Variational Auto Encoders (VAEs)

Aashish Patel(190014)
Prateek Jain(190634)

April 2022

1 Introduction

Variational autoencoders (VAEs) is a learning techniques for learning latent representations. They help in modelling of the underlying probability distribution when only the samples from the underlying pdf are given.

E.g. creating new images of certain class, when some images from the class are given.

2 Goal

Given some samples x from a probability distribution generate new samples from it. Let $x^{(1)}, x^{(2)}, ...x^{(N)}$ be the observed samples from the target distribution which we want to model.

The two pdf's being represented by the graphical plate are

$$p_\theta(x|z) = p(x|z;\theta) \tag{1}$$

and

$$p(z;\theta) = p_\theta(z) \tag{2}$$

$$p_\theta = \int (p_\theta(x|z) \cdot p_\theta(z) \cdot d(z) \tag{3}$$

The value of $p_\theta(x)$ may be intractable here, i.e. we can not compute it. Therefore, we look towards training the model. The first thing that we need to train the model is:-

Figure 1: Graphical Plate

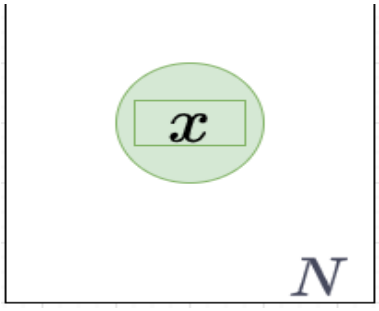
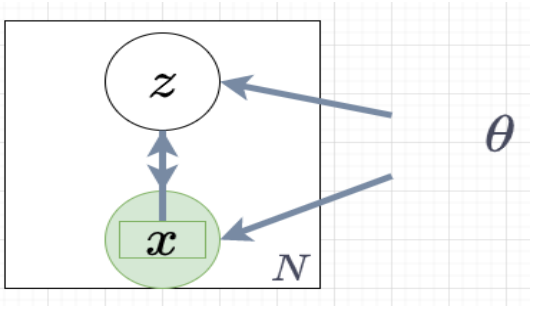


Figure 2: Graphical Plate



3 Loss Function

Loss Function: Here in our present problem, we only have the samples x from the distribution and we don't have any underlying p^* (actual target distribution), we can't use KL divergence to compare the calculated distribution with the target distribution.

Objective Function: Maximize the log-likelihood for $x^{(1)}, x^{(2)}, ...x^{(N)}$

$$max_\theta(log(p_\theta(x^{(1)}, x^{(2)}, x^3, ...x^{(N)}))) \tag{4}$$

$$= max_\theta[\sum_{i=1}^{N}(log(p_\theta(x^{(i)})))] \tag{5}$$

where, $p_\theta(x^{(1)}), p_\theta(x^{(2)}), ...p_\theta(x^{(N)})$ are identical and independent distributions.

Marginalizing over z

$$log(p_\theta(x^{(i)})) = log(\int(p_\theta(x^{(i)}, z) \cdot dz)) \tag{6}$$

As this is intractable hence we need to use Jensen's Inequality. Introducing a new term $q_\phi(z|x)$ to make the Jensen's Inequality applicable to Eq. 6.

Using the Jensen's Inequality,

$$log(p_\theta(x^{(i)})) = log(\int(p_\theta(x^{(i)}, z) \cdot dz)) \tag{7}$$

$$= log(\int(\frac{p_\theta(x^{(i)}, z)}{q_\phi(z|x^{(i)})}) \cdot q_\phi(z|x^{(i)}) \cdot dz) \tag{8}$$

$$>= \int(log(\frac{p_\theta(x^{(i)}, z)}{q_\phi(z|x^{(i)})})) \cdot q_\phi(z|x^{(i)}) \cdot dz = L \tag{9}$$

Now our new objective is to :-

$$maximize_{\theta,\phi} L(\theta, \phi; x^{(i)}) \tag{10}$$

Maximizing Eq 10 would automatically mean that we are maximing Eq 7, due to the inequality in Eq 9. $L(\theta, \phi; x^{(i)})$ is known as Variatioinal Lower Bound or ELBO(Evidence Lower Bound).

$$L(\theta, \phi; x^{(i)}) = \int(log(\frac{p_\theta(x^{(i)}|z) \cdot p_\theta(z)}{q_\phi(z|x^{(i)})})) \cdot q_\phi(z|x^{(i)}) \cdot dz \tag{11}$$

Using the definition of KL Divergence and Expectation, we can also write Eq 11 as :-

$$- D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + E_{z\sim q_\phi(z|x^{(i)})}[log(p_\theta(x^{(i)}|z))] \tag{12}$$

Now we would have to use Neural Networks to define the pdf's $q_\phi(z|x^{(i)})$ and $p_\theta(x^{(i)}|z)$.

Note: The KL Divergence between 2 Gaussians is given by

$$KL(p,q) = \log(\sigma_2/\sigma_1) + (\sigma_1^2 + (\mu_1 - \mu_2)^2)/2\sigma_2^2 - 1/2 \tag{13}$$

where σs and μs are corresponding to PDFs p and q respectively in order.

4 Description of PDF's involved

How to find and characterize the different pdf's:

- $p_\theta(z)$

  We will keep this fixed and choose $N(z; 0, I)$ as the fixed pdf for this.

- $p_\theta(x^{(i)}|z)$

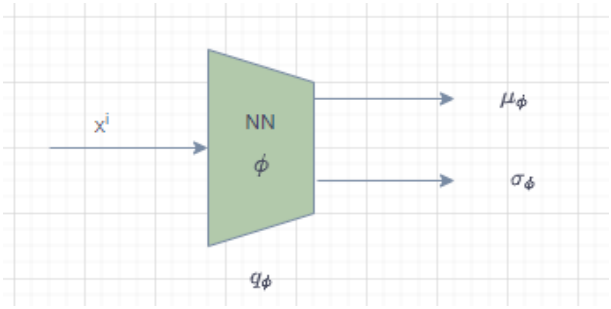
  We would use a Neural Network to model it. A Neural Network cannot output a pdf, but it can output samples from a pdf or parameters of a pdf. Here we would parametrize $p_\theta(x^{(i)}|z)$ as a Gaussian Distribution $\mathcal{N}(x; u_\theta(z), \sum_\theta(z))$ and try to estimate the parameters μ and $\sum$ of this Gaussian by taking them as the output of the Neural Network.

- $q_\phi(z|x^{(i)})$

  We would use a Neural Network to model it. A Neural Network cannot output a pdf, but it can output samples from a pdf or parameters of a pdf. Here we would parametrize $q_\phi(z|x^{(i)})$ as a Gaussian Distribution $\mathcal{N}(z; u_\phi(x^{(i)}), \sum_\phi(x^{(i)}))$ and try to estimate the parameters μ and $\sum$ of this Gaussian by taking them as the output of the Neural Network. n

  Note: For simplification, we can assume sigma to be a diagonal matrix

Figure 3: Encoder



5 Training

While training, the objective is to maximize the Loss function $L(\theta, \phi; x^{(i)})$. We use gradient based optimization techniques. Here, we need we need to compute the gradient of the loss function with respect to θ and ϕ given x^i.

- Compute the gradient with respect to θ: $\nabla_\theta L(\theta, \phi; x^{(i)})$

$$L(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + E_{z\sim q_\phi(z|x^{(i)})}[log(p_\theta(x^{(i)}|z))] \tag{14}$$

Figure 4: Decoder

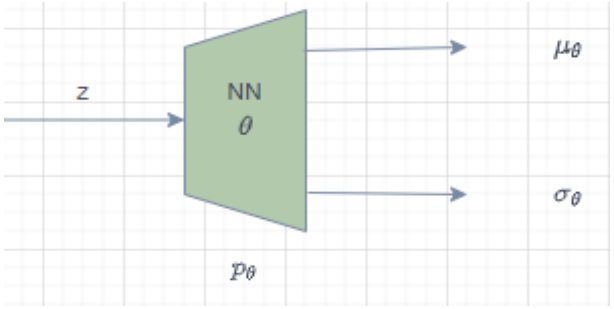
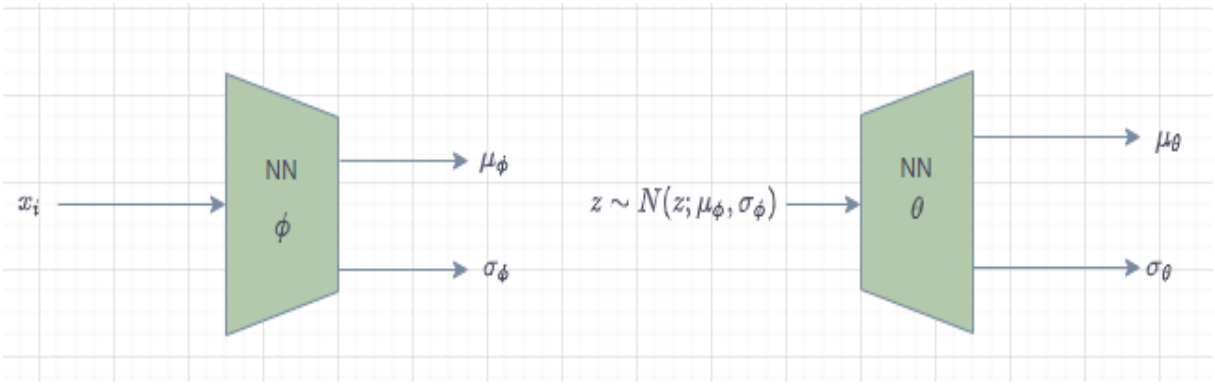


Figure 5: Full VAE along with Encoder and Decoder Part



since $p_\theta(z)$ is fixed, we need to differentiate only the second term with respect to θ

$$E_{z\sim q_\phi(z|x^{(i)})}[log(p_\theta(x^{(i)}|z))] \tag{15}$$

Using Monte-Carlo approximations, the expectation value is:

$$E = 1/N \sum_{l=1}^{N} log(p(x^{(i)}|z^n)); z \sim q_\phi(z|x^{(i)}) \tag{16}$$

In this way, the derivative of the above expression is easy to compute since it is a normal distribution

- Compute the gradient with respect to ϕ: $\nabla_\phi L(\theta, \phi; x^{(i)})$

  There are two methods by which we can compute the gradient

  – **Gradient Based Direct Method**

$$\nabla_\phi E = \nabla_\phi \int (q_\phi(z|x^{(i)}) log(p_\theta(x^{(i)}|z)) dz) \tag{17}$$

$$\nabla_\phi E = \int (\nabla_\phi q_\phi(z|x^{(i)}) log(p_\theta(x^{(i)}|z)) dz) \tag{18}$$

Using Monte Carlo Approximation,

$$\nabla_\phi E \approx 1/N \sum_{n=1}^{L} log(p(x^{(i)}|z)) \nabla_\phi log(q_\phi(z|x^{(i)})) \tag{19}$$

But this is not a good estimator as it has very high variance.

– **Reparametrization Trick**

Let

$$q_\phi(z|x^i) = \mathcal{N}(z; a, b^2) \tag{20}$$

where b^2 is the diagonal matrix entries

and

$$z = \epsilon b_\phi + a_\phi; \epsilon = \mathcal{N}(\epsilon; 0, 1) \tag{21}$$

$$\nabla_\phi E_{q_\phi}(z|x^i)[log(p_\theta(x^i|z)] = \nabla_\phi E_{\epsilon\sim\mathcal{N}(\epsilon;0,1)}[log(p_\theta(x^i|\epsilon b_\phi + a_\phi)] \tag{22}$$

Using Monte Carlo Approximation,

$$\nabla_\phi E_{q_\phi}(z|x^i)[log(p_\theta(x^i|z)] \approx \nabla_\phi[1/L \sum_{l=1}^{L} log(p_\theta(x^i|\epsilon b_\phi + a_\phi)]; \epsilon = \mathcal{N}(\epsilon; 0, 1) \tag{23}$$

Figure 6: Full VAE along with Encoder and Decoder Part