Variational Autoencoder : A Deep Generative Model

Ayush Raina, Anushka Dassi, Arnav Bhatt

Prof. Chiranjib Bhattacharyya

***Abstract*—Autoencoders are deep generative models. They are widely used for tasks such as image generation, data compression, denoising and capturing the most important features of the data. Autoencoders contains encoder and decoder networks. Variational Autoencoders introduce probabilistic modelling into encoding process in which we learn the latent distribution, which enables generation of data which are similar to training data.**

1. Introduction

1.1. Neural Networks

Given a training set $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$, where $x^{(i)}$ is the input and $y^{(i)}$ is the output. Neural networks are used to learn a function $h_\theta(x)$ which maps input x to output y. The function $h_\theta(x)$ is parameterized by θ which are the weights of the neural network. The neural network is trained by minimizing the loss function $J(\theta)$ by using optimization algorithms like gradient descent. In our case we will not have any output $y^{(i)}$ for the input $x^{(i)}$. We need to learn the latent representation of the input data $x^{(i)}$.

2. Autoencoders

2.1. Architecture

This is also a neural network which consists of two parts: Encoder(ϕ),Decoder(θ) (where ϕ, θ are the parameters)and bottleneck layer whose dimension is much less than input layer. Encoder network learns the latent representation of the input data and decoder network learns to reconstruct the input data from the latent representation. The loss function is defined as the difference between input and output data.

2.2. Aim

We want output of the network as input itself. The main challenge is that input has to pass from a bottleneck layer whose dimension is much less than input layer. We can say that the network learns to compress the data to hidden state.

2.3. Objective

In this case we are dealing with images and we want our output image to be as similar as input image. Since our image passes through encoder and decoder network we can set up the loss function as follows:

$$\boxed{min \sum_{i=1}^{m} || x^{(i)} - Decoder_\theta(Encoder_\phi(x^{(i)})) ||^2}$$

To minimise this loss function we use backpropagation algorithm. We will now discuss some methods which we will use in further sections.

3. Expectation Maximization Algorithm

3.1. Normal EM Algorithm

In one line we can say EM algorithm is used to perform maximum likelihood estimation in the presence of latent variables. Suppose ψ are the parameters we want to learn and $z^{(i)}$ are the latent variables. Then we perform these two steps iteratively until convergence:

E-Step: For all i, set $\boxed{Q_i(z^{(i)}) = p(z^{(i)}|x^{(i)};\psi)}$

M-step: $\boxed{\psi^{(new)} = argmax_\psi \sum_{i=1}^{m} ELBO(x^{(i)}; Q_i, \psi)}$

where ELBO is the evidence lower bound.

The problem with this is in E-step where we assumed that posterior distribution can be computed. Suppose $z \sim N(0, I_{k\times k})$ and $x|z \sim NeuralNetwork(z^{(i)};\theta)$. Now it is almost impossible to calculate the posterior distribution $p(z^{(i)}|x^{(i)};\psi)$. Now only option left is to approximate the posterior distribution. One way to do this is to use MCMC EM Algorithm, but that requires knowledge of markov chains, instead we will be using Variational Inference.

3.2. Variational Inference