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 $p(z^{(i)}|x^{(i)};\psi)$.

4. Approximating the distribution

There are many methods to approximate the posterior distribution. One of the methods we are going to discuss is Variational Inference.

4.1. Variational Inference

This is used to approximate the complex distributions. We know that $\boxed{log(P(x)) = ELBO(x;Q) + D_{KL}(Q||P_{z|x})}$ where $P_{z|x}$ is the posterior distribution of z given x. We want Q to be as close as possible to $P_{z|x}$, which means $D_{KL}(Q||P_{z|x}) \longrightarrow 0$. If we maximise ELBO with respect to Q, then we are minimising $D_{KL}(Q||P_{z|x})$(because LHS is constant) and minimising the KL Divergence means Q is close to $P_{z|x}$. Hence we can say $\boxed{P_{z|x} \sim argmax_{q\sim Q} ELBO(x;q)}$, where we will assume all $q's$ are in the same family of distributions(Q) called as variational family. These are distributions over vectors of dimension k.

4.2. Mean Field Variational Inference

In this method we assume that variational family factorizes,$Q(z) = \prod_{i=1}^{k} q_i(z^{(i)})$, which means each latent variable $z^{(i)}$ is independent.

4.3. Coordinate Ascent/Descent

This is the technique we used in EM algorithm. In E-step all paramters(ψ) were fixed and in M-step, all $Q_i's$ were fixed and we optimised the parameters. This is called as Coordinate Ascent/Descent. In Variational Autoencoders we use gradient descent approach.

5. Variational Autoencoders

5.1. Setting up the problem

We have input data $x^{(i)}$, latent variables $z^{(i)}$ are continuous random variables, each $z^{(i)} \sim N(0, I_{k\times k})$, $x^{(i)}|z^{(i)} \sim N(g(z^{(i)};\theta), \sigma^2 I_{d\times d})$. Here $g(z^{(i)};\theta)$ is some neural network(generally decoder model) with parameters θ. We will us variational inference to approximate the posteiror distribution $p(z^{(i)}|x^{(i)})$. For this we will choose our variational family as $\boxed{Q_i = N(q(x^{(i)};\phi), diag(\upsilon(x^{(i)};\psi)^2))}$.

Here $q(x^{(i)};\phi)$ is a neural network with parameters ϕ which generates the mean and $\upsilon(x^{(i)};\psi)$ is a neural network with parameters ψ which generates the variance of the distribution. We will now derive the ELBO for this model.Since we have choosen our variational family as Normal distribution, this is also called Ammortized Inference.

Further if we notice that our covariance matrix of variational family Q is diagonal, which mean all the latent variables are independent, that means indirectly we are making mean field assumption. Now we can write the ELBO as follows:

$$\boxed{ELBO(\theta,\phi,\psi) = \sum_{i=1}^{n} \mathbb{E}_{z^{(i)}\sim Q_i}[log(\frac{p(x^{(i)},z^{(i)};\theta)}{Q_i(z^{(i)})})]} \text{ where}$$

$Q_i = N(q(x^{(i)};\phi), diag(v(x^{(i)};\psi)^2))$. Now we need to maximise this ELBO with respect to θ, ϕ, ψ. Here are the updates for the parameters:
1. $\theta = \theta + \eta\nabla_\theta ELBO(\theta,\phi,\psi)$
2. $\phi = \phi + \eta\nabla_\phi ELBO(\theta,\phi,\psi)$
3. $\psi = \psi + \eta\nabla_\psi ELBO(\theta,\phi,\psi)$

5.2. Calculating Gradient with respect to θ

Let us calculate $\nabla_\theta ELBO(\theta,\phi,\psi)$. This can be written as: $\nabla_\theta ELBO(\theta,\phi,\psi) = \sum_{i=1}^{n} \nabla_\theta \mathbb{E}_{z^{(i)}\sim Q_i}[log(\frac{p(x^{(i)},z^{(i)};\theta)}{Q_i(z^{(i)})})]$. Since Q_i does not depend on θ, we will have $\nabla_\theta Q_i(z^{(i)}) = 0$ and we can take the gradient inside the expectation, so we get $\sum_{i=1}^{n} \mathbb{E}_{z^{(i)}\sim Q_i}[\nabla_\theta log(p(x^{(i)},z^{(i)};\theta))]$

5.3. Calculating Gradient with respect to ϕ

Since the sampling distribution Q_i depends on ϕ, we cannot swap the expectation and gradient. So we need to use the Reparametrization Trick. We know that if $x \sim N(\mu,\sigma^2)$ is equivalent to $x = \mu + \sigma\xi$, where $\xi \sim N(0,1)$. Hence we can say $z^{(i)} \sim N(q(x^{(i)};\phi), diag(v(x^{(i)};\psi)^2))$ is equivalent to say $\boxed{z^{(i)} = q(x^{(i)};\phi) + diag(v(x^{(i)};\psi))\xi^{(i)}}$, where $\xi^{(i)} \sim N(0, I_{k\times k})$. With this reparametrization we now have converted $\mathbb{E}_{z^{(i)}\sim Q_i}[log(\frac{p(x^{(i)},z^{(i)};\theta)}{Q_i(z^{(i)})})]$ to

$$\boxed{\mathbb{E}_{\xi^{(i)}\sim N(0,I_{k\times k})}[log(\frac{p(x^{(i)}, q(x^{(i)};\phi) + diag(v(x^{(i)};\psi))\xi^{(i)};\theta)}{Q_i(q(x^{(i)};\phi) + diag(v(x^{(i)};\psi))\xi^{(i)})})]}.$$

Now we can take the gradient inside the expectation and we can calculate the gradient with respect to ϕ.