Variational Autoencoder

Latent variable models form a rich class of probabilistic models that can infer hidden structure in the underlying data. In this post, we will study variational autoencoders, which are a powerful class of deep generative models with latent variables.

1 Representation

Consider a directed, latent variable model as depicted in Figure 1. In this model, z and x denote the latent and observed variables respectively. The joint distribution expressed by this model is given as:

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

From a generative modeling perspective, this model describes a generative process for the observed data, x, using the following procedure:

- First sample z from the prior distribution $p(z)$. For our purposes, we will consider that the prior is a standard normal distribution $N(0, I_d)$ where d is the dimension of the latent space. We denote this by $z \sim p(z)$.
- The we sample x from the posterior distribution $p(x|z)$. We denote this by $x \sim p(x|z)$.

Figure 1: Latent Variable Model

If one adopts the belief that the latent variables z somehow encode semantically meaningful information about x, it is natural to view this generative process as first generating the "high-level" semantic information about x first before fully generating x. In this lab we will be concerned with learning the joint distribution $p(x, z)$.

Given a dataset $D = x^{(1)}, x^{(2)}, ..., x^{(N)}$, we are interested in the following learning and inference tasks

- Find the parameters θ of $p_\theta(x, z)$ that best fit D.
- Given a sample x and am model $p_\theta(x, z)$, what is the posterior distribution $q_\phi(z|x)$.

2 Learning Directed Latent Variable Models

One way to measure how closely $p_\theta(x, z)$ fits the observed dataset D is to measure the Kullback-Leibler (KL) divergence between the data distribution (which we denote as $p_{data}(x)$) and the model's marginal distribution $p_\theta(x) = \int p_\theta(x, z)dz$.

Using the equivalent Maximum-likelihood (minimizing the KL-divergence is equivalent to maximizing the log-likelihood of the data), our objective becomes:

$$\max_\theta \sum_{x \in D} \log p(x) \tag{2}$$

It turns out that for a singe data-point x we can deduce a lower-bound (evidence lower-bound) defined as follows:

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z)\right] - D_{KL}(q_\phi(z|x) \parallel p(z))$$

For the full derivation please refer to the original paper[1]

Notice that the posterior distribution, $q_\phi(z|x)$, it is parameterized by ϕ and that the likelihood distribution, $p_\theta(x|z)$, is parameterized by θ. $p(z)$ is the prior distribution which we consider to be a standard normal $N(0, I_d)$, where d is the dimension of the latent representation.

3 Implementation Details

We are going to model the two distributions, $q_\phi(z|x)$ and $p_\theta(x|z)$, using two neural networks, called encoder and decoder. Figure 2 depicts our setup.

Before going further with the training procedure let's first discuss a bit more the lower-bound objective. Our new objective contains two terms: :

- $\mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z)\right]$ this is called the reconstruction term. Notice that this term requires estimation by sampling. Fortunately, as long as we have a decent batch-size, we can use a single point estimate. Furthermore, we

[1] https://arxiv.org/pdf/1312.6114.pdf

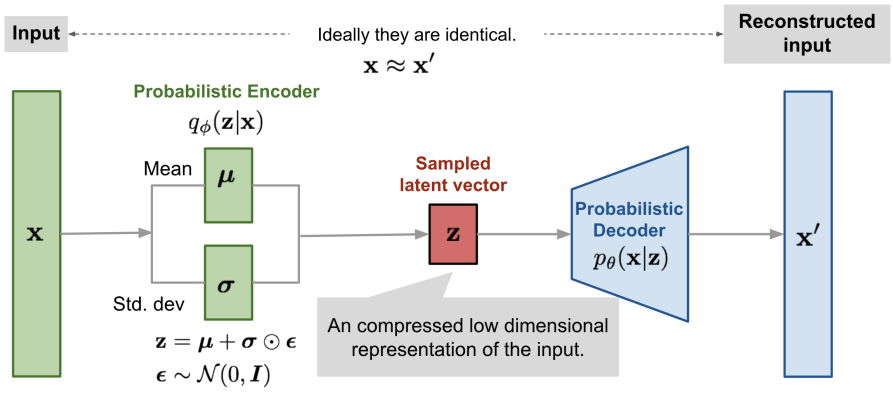

Figure 2: VAE.

can replace this term by a simple reconstruction loss such as suqared error, between the output of the decoder and the encoder input data, $|X_{in} - X_{dec}|_2^2$.

- $D_{KL}(q_\phi(z|x) \parallel p(z))$ has a closed form for normal distributions as you will see in the labs implementation.

Now, let's discuss the architecture of the model. As depicted in Figure 2 the encoder is a neural network parameterized by ϕ, that has two output heads, one for the mean and one for the standard deviation of the posterior distribution $q_\phi(z|x)$. Each of the output produces a d-dimensional tensor. The decoder is another neural network that has as input a d-dimensional vector obtained by sampling (we will discuss later), and produces a tensor in the observation(input) space.

To make the encoder differentiable with respect to the reconstruction loss, we apply the reparametrization trick as follows:

- First sample ϵ from a standard normal $N(0, I_d)$: $\epsilon \sim N(0, I_d)$.
- Multiply the sample by the standard deviation (output of the encoder) and add the mean (also output of the encoder): $\mu + \sigma \odot \epsilon$, where $\odot$ represent element-wise multiplication.
- The output is then passed to the decoder.

3.1 Algorithm

The training procedure can be summarized as follows:

Algorithm 1: VAE Training

Consider Q - encoder network, P - decoder network. 1. Sample a batch of data, X;
2. $\mu(X), \sigma(X) = Q(X)$;
3. $\epsilon \sim N(0, I_d)$;
4. $z = \mu(X) + \epsilon \odot \sigma(X)$;
5. $X_{rec} = P(z)$;
6. $Loss(\phi, \theta; X) = |X_{rec} - X|_2^2 + D_{KL}(N(0, I_d) \parallel N(\mu(X), \sigma^2(X)))$;
7. Differentiate with respect to ϕ and θ and apply gradient step.;
8. Repeat 1 - 7 till convergence.

4 Acknowledgements

Some of the materials presented in this lab were borrowed from here.