Deep Learning 1 - Homework 3

Pedro M.P. Curvo
MSc Artificial Intelligence
University of Amsterdam
`pedro.pombeiro.curvo@student.uva.nl`

Part 1

1.1

To sample an image using the decoder f_θ from the generative model described in this section, we first need to sample a latent vector z from the prior distribution $p(z)$. In this case, the prior distribution is a standard multivariate Gaussian distribution, so we can sample z from a standard normal distribution, i.e., $z \sim \mathcal{N}(0, I_D)$. Then, we need to compute the pixel probabilities using the decoder f_θ. For this, we pass the sampled latent vector z through the decoder f_θ, which is a neural network parameterized by θ. This will map the latent vector z to the probabilities of the Categorical distribution for each pixel x_m in the image. $f_\theta(z) \to (p_1, p_2, ..., p_k)^M$. Here: $p_m = (p_{m1}, p_{m2}, ..., p_{mk})$ are the event probabilities for pixel m being in one of the k categories. M is the number of pixels in the image. The, we can sample pixel values x_n for each pixel m by sampling from the Categorical distribution $Cat(x_m|f_\theta(z)_m)$. Where, $x_m \sim Cat(p_m)$. This will generate the image by sampling each pixel value independently based on the probabilities computed by the decoder. Finally we combine the pixel values x_m to from the image x_n.

1.2

Monte Carlo integration with samples from $p(z_n)$ can be used to approximate the expectation of the log-likelihood. However, is inefficient for training VAE models because it requires a large number of samples to accurately estimate the log-likelihood. This problem appears because the dimensionality of the latent space z affects the number of samples needed to cover the entire latent space adequately. As the dimensionality of the latent space increases, the space becomes sparser, and more samples are needed to cover the space to capture the posterior distribution $p(z|x)$, which is often highly concentrated in certain regions of the latent space (as shown by the blue contours in Figure 2).

This results in a exponentially increasing number of samples needed to accurately estimate the log-likelihood, that is, $L \to \infty$ as $D \to \infty$. This makes Monte Carlo integration impractical for high-dimensional latent spaces due to the computational cost of sampling a large number of points to estimate the log-likelihood accurately. Therefore, other methods are used to estimate the log-likelihood in VAE models.

1.3

From Equation 10, we have:

$$\log p(x_n) - KL(q_\theta(z_n|x_n)||p(z_n|x_n)) = \mathbb{E}_{q_\theta(z_n|x_n)}[\log p(x_n|z_n)] - KL(q_\theta(z_n|x_n)||p(z_n))$$

Rearranging the equation we get:

$$\log p(x_n) = \mathbb{E}_{q_\theta(z_n|x_n)}[\log p(x_n|z_n)] - KL(q_\theta(z_n|x_n)||p(z_n)) + KL(q_\theta(z_n|x_n)||p(z_n|x_n))$$

Now, we know the KL divergence is always non-negative, since it is a measure of the difference between two distributions by measuring how much one probability distribution diverges from a second, expected probability distribution. It is zero if and only if the two distributions are the same. Mathematically, we have that:

$$\begin{aligned} KL(q_\theta(z_n|x_n)||p(z_n)) &\geq 0 \\ KL(q_\theta(z_n|x_n)||p(z_n|x_n)) &\geq 0 \end{aligned}$$

With this we have that:

$$\begin{aligned} \log p(x_n) &= \mathbb{E}_{q_\theta(z_n|x_n)}[\log p(x_n|z_n)] - KL(q_\theta(z_n|x_n)||p(z_n)) + KL(q_\theta(z_n|x_n)||p(z_n|x_n)) \\ &\geq \mathbb{E}_{q_\theta(z_n|x_n)}[\log p(x_n|z_n)] - KL(q_\theta(z_n|x_n)||p(z_n)) \end{aligned}$$

Therefore, the right-hand side of the equation $\mathbb{E}_{q_\theta(z_n|x_n)}[\log p(x_n|z_n)] - KL(q_\theta(z_n|x_n)||p(z_n))$ is always less than or equal to the true log-likelihood $\log p(x_n)$. Hence, the right-hand side of the equation is a lower bound on the true log-likelihood.

1.4

ELBO consists of two terms:

$$ELBO(x_n) = \mathbb{E}_{q_\theta(z_n|x_n)}[\log p(x_n|z_n)] - KL(q_\theta(z_n|x_n)||p(z_n))$$

The first term, $\mathbb{E}_{q_\theta(z_n|x_n)}[\log p(x_n|z_n)]$, represents the expected log-likelihood of the data x_n given the latent variable z_n. However, this value stays the same regardless of how $q(z_n|x_n)$ is chosen. The second term, $KL(q_\theta(z_n|x_n)||p(z_n))$, represents the KL divergence between the approximate posterior $q_\theta(z_n|x_n)$ and the prior $p(z_n)$. As $q_\theta(z_n|x_n)$ approaches the true posterior $p(z_n|x_n)$, the KL divergence term decreases towards zero. This is because the KL divergence is minimized when the two distributions are the same, $q(z_n|x_n) = p(z_n|x_n)$. With that being said, as the KL divergence term decreases, the ELBO increases. becoming closer to the true log-likelihood $\log p(x_n)$. Ideally, when $q(z_n|x_n) = p(z_n|x_n)$, the ELBO is equal to the true log-likelihood $\log p(x_n)$.

So:

$$\begin{aligned} \lim_{q_\theta(z_n|x_n)\to p(z_n|x_n)} ELBO(x_n) &= \lim_{q_\theta(z_n|x_n)\to p(z_n|x_n)} \mathbb{E}_{q_\theta(z_n|x_n)}[\log p(x_n|z_n)] - KL(q_\theta(z_n|x_n)||p(z_n)) \\ &= \mathbb{E}_{p(z_n|x_n)}[\log p(x_n|z_n)] - \lim_{q_\theta(z_n|x_n)\to p(z_n|x_n)} KL(q_\theta(z_n|x_n)||p(z_n)) \\ &= \mathbb{E}_{p(z_n|x_n)}[\log p(x_n|z_n)] - KL(p(z_n|x_n)||p(z_n)) \\ &= \mathbb{E}_{p(z_n|x_n)}[\log p(x_n|z_n)] - 0 \\ &= \mathbb{E}_{p(z_n|x_n)}[\log p(x_n|z_n)] \\ &= \log p(x_n) \end{aligned}$$

And that is why the main goal of training a VAE is to maximize the ELBO, as it is a lower bound on the true log-likelihood $\log p(x_n)$. And this corresponds to minimizing the KL divergence between the approximate posterior $q_\theta(z_n|x_n)$ and the prior $p(z_n)$.

1.5

The names **reconstruction** loss and **regularization** loss are used because:

- **Reconstruction Loss**: The term $\mathbb{E}_{q_\theta(z_n|x_n)}[\log p(x_n|z_n)]$ is the expected log-likelihood of the data x_n given the latent variable z_n. Hence, it is a measure of how well the model can reconstruct the observed data x_n from the latent variable z_n. Meaning, it directly corresponds to the task of reconstructing the original input, being the main objective of the model. Therefore, it is called the **reconstruction** loss.
- **Regularization Loss:** The term $KL(q_\theta(z_n|x_n)||p(z_n))$ is the KL divergence and ensures that the learned posterior distribution $q_\theta(z_n|x_n)$ is close to the prior distribution $p(z_n)$. This term regularizes the model by preventing the posterior distribution from deviating too much from the prior distribution. Therefore, it is called the **regularization** loss.

1.6

When the prior distribution $p(z_n)$ is and variational posterior $q_\theta(z_n|x_n)$ are not Gaussian, the KL divergence term $KL(q_\theta(z_n|x_n)||p(z_n))$ cannot be computed simply in closed form. To overcome this, we can use the Monte Carlo approximation to estimate the regularization term.

The regularization term in the VAE is the KL divergence between the variational distribution $q_\theta(z_n|x_n)$ and the prior distribution $p(z_n)$. Mathematically, this term is defined as:

$$D_{KL}(q_\theta(z_n|x_n)||p(z_n)) = \mathbb{E}_{q_\theta(z_n|x_n)}[\log \frac{q_\theta(z_n|x_n)}{p(z_n)}]$$

Assuming the closed-form expression for the KL divergence is intractable, we can use the Monte Carlo approximation to estimate the KL divergence. First, we sample L latent vectors $z_n^{(l)}$ from the variational distribution $q_\theta(z_n|x_n)$, where $l = 1, 2, ..., L$. Then, we compute the log-ratio of the variational distribution and the prior distribution for each sample:

$$\log \frac{q_\theta(z_n^{(l)}|x_n)}{p(z_n)}$$

where $q_\theta(z_n^{(l)}|x_n)$ is the probability density of the variational distribution evaluated at the sample $z_n^{(l)}$. Finally, we compute the Monte Carlo estimate of the KL divergence as:

$$D_{KL}(q_\theta(z_n|x_n)||p(z_n)) \approx \frac{1}{L}\sum_{l=1}^{L} \log \frac{q_\theta(z_n^{(l)}|x_n)}{p(z_n)}$$

Then, we can use this approximation in the loss function:

$$\mathbb{L}_{regularization,n} \approx \frac{1}{L}\sum_{l=1}^{L} \log \frac{q_\theta(z_n^{(l)}|x_n)}{p(z_n)}$$

This methods allows us to estimate the KL divergence term when the prior and variational posterior are not Gaussian. The more samples we use, the more accurate the estimate will be, but it will also increase the computational cost of training the model. This method can be used for any arbitrary prior and variational posterior distributions, as long as we can sample from the variational distribution and compute the log-ratio of the two distributions.

It is also important to note the in this case the sampling is more efficient than the Monte Carlo integration for estimating the log-likelihood. Sampling from $p(z_n)$ can lead to a more inefficient sampling because the prior may not capture the actual structure of the data, that is, many samples may be drawn from regions of z_n that do not contribute to $p(x_n|z_n)$, leading to a poor estimate of the log-likelihood, as seen in the image, where the prior distribution is a standard normal distribution. However, $p(z_n|x_n)$ is more concentrated in specific regions of the latent space (the blue contours in the image). However, by sampling from $q_\theta(z_n|x_n)$, we can sample from a distribution that is more likely to capture the structure of the data. The variational distribution $q_\theta(z_n|x_n)$ is designed to be a good approximation of the true posterior $p(z_n|x_n)$, Hence, samples drawn from $q_\theta(z_n|x_n)$ are more likely to be in regions of the latent space that contribute to the log-likelihood. In other words, the samples drawn from $q_\theta(z_n|x_n)$ are concentrated in areas where the posterior is high, which makes tgese samples more useful. This lead to a more efficient Monte-Carlo integration because fewer samples are needed to estimate the log-likelihood accurately.

1.7

When we sample directly from a distribution $q_\theta(z_n|x_n)$ to estimate the expectation $\mathbb{E}_{q_\theta(z_n|x_n)}[\log p(x_n|z_n)]$, the sampling part is non-differentiable with respect to the parameters θ of the model. This means that the gradients with respect to the encoder parameters θ cannot be computed directly since the sampling operation does not provide a smooth path for backpropagation. This is because the sampling operation introduces a discontinuity in the computation graph, which makes it impossible to compute the gradients using standard backpropagation.

The reparameterization trick is a technique used to make the sampling operation differentiable with respect to the parameters θ. The trick involves reparameterizing the random variable z_n as a deterministic function of the encoder outputs $\mu_\theta(x_n)$ and $\sigma_\theta(x_n)$, and a noise variable ϵ sampled from a fixed distribution, such as a standard normal distribution, i.e., $\epsilon \sim \mathcal{N}(0, I_D)$. This way the sample z_n can be expressed as:

$$z_n = \mu_\theta(x_n) + \sigma_\theta(x_n) \odot \epsilon$$

With this, the sampling operation is now differentiable with respect to the parameters θ of the model. Allowing us to compute the gradients of the loss function with respect to the parameters θ using backpropagation through the encoder network.

1.8

REVISE CODE

1.9

TODO: Put images here

By looking at the images generated by the VAE model before training, middle of training, and after training, we can see that initially we only have noise in the images, as the model has not learned to generate meaningful images yet. As the model trains, the images start to become more recognizable, and we start to see some numbers appearing in the images resembling the digits in the MNIST dataset. At the end of training, I see that the images didn't improve much from the middle of training. I believe this is the case, since the regularization loss keeps increasing.

1.10

RUN CODE, PUT IMAGES, WRITE TEXT

Part 2

2.1

TODO

2.2

TODO

2.3

TODO