3A-MDS-E2-DL

Introduction to deep learning assignment

Hamdi BEL HADJ HASSINE

1.Variational autoencoders

Question 1

Let $n \in \{1, .., N\}$. Since we know the distribution of z_n and the conditional distribution $(x_n|z_n)$, we can use ancestral sampling to sample x_n values:

1. Sample z_n from $\mathcal{N}(0, I_D)$
2. Compute $f_\theta(z_n)$ by passing z_n to the neural net f_θ
3. For every $m \in \{1, .., M\}$, sample $x_n^{(m)}$ from Bern$(f_\theta(z_n)_m)$

And we obtain the x_n vector.

Question 2

Monte-Carlo Integration is inefficient because the number of samples L needs to be large to accurately approximate the integral, and here computing each sample is expensive since it needs to be passed through a neural network.

Let's provide an approximation of the complexity needed to accurately evaluate an integral. First we can define that the sampling is "accurate" if the samples cover the whole integration space. To approximate this, we can assume that the samples form a grid of values (e.g. 10 values) between a minimum value and a maximum value for each component of z. Then if dim(z)=1, we need 10 values, if dim(z)=2, we need 100 values, and generally we need 10^D values. Therefore we conclude that Monte Carlo integration is inefficient in large dimensions because the number of samples needed to properly cover the integration space is exponentially large in D.

Question 3

We can derive the KL-divergence of two univariate gaussians $q = N\left(\mu_q, \sigma_q^2\right)$ and $p = N\left(\mu_p, \sigma_p^2\right)$ as follows:

$$\begin{aligned} D_{KL}(q||p) &= -\int q(x)\log p(x)dx + \int q(x)\log q(x)dx \\ &= \frac{1}{2}\log\left(2\pi\sigma_p^2\right) + \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} - \frac{1}{2}\left(1 + \log 2\pi\sigma_q^2\right) \\ &= \log\frac{\sigma_p}{\sigma_q} + \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} - \frac{1}{2} \end{aligned}$$

The KL divergence is the smallest when it's equal to 0, for p=q. We can also give as example $\left(\mu_q, \mu_p, \sigma_q^2, \sigma_p^2\right) = (0, 0.1, 1, 1)$ so $D_{KL}(q||p) = 0.005$ is small.

But for $\left(\mu_q, \mu_p, \sigma_q^2, \sigma_p^2\right) = (0, 1, 1, 0.0001)$, $\quad D_{KL}(q||p) = 9994.9$ is large.

Question 4

From Eq. 16 we can write:

$$\begin{aligned}\log p\left(x_n\right) &= KL\left(q\left(Z \mid x_n\right) || p\left(Z \mid x_n\right)\right) + E_{q(z_n|x_n)}\left[\log p\left(x_n \mid z_n\right)\right] - KL\left(q\left(Z \mid x_n\right) || p(Z)\right) \\ &\geq E_{q(z_n|x_n)}\left[\log p\left(x_n \mid z_n\right)\right] - KL\left(q\left(Z \mid x_n\right) || p(Z)\right)\end{aligned}$$

because for given distributions p,q,

$$D_{KL}(q||p) = -E_{q(x)}[\log \frac{p(x)}{q(x)}] \leq -log\left(E_{q(x)}[\frac{p(x)}{q(x)}]\right) = -log\left(\int q(x)\frac{p(x)}{q(x)}dx\right) = log(1) = 0$$

where in the first inequality we use Jensen inequality and concavity of the log.

Therefore the right-hand side of Eq. 16 is a lower bound on $\log p(x_n)$. We have seen in Q2 that computing and optimizing $\log p(x_n)$ is expensive, so instead we optimize its lower bound which, for a good choice of q, will be easier to optimize (and the KL term can be computed analytically).

Question 5

When the lower bound is pushed up, $\log p\left(x_n\right) - KL\left(q\left(Z \mid x_n\right) \| p\left(Z \mid x_n\right)\right)$ is pushed up too. There are two possibilities:

1. $\log p\left(x_n\right)$ increases: The log-likelihood of the data increases, meaning that our model fits the data better, and that by optimizing the lower bound we actually optimized the original objective $\log p(x_n)$, which is good.

2. $KL\left(q\left(Z \mid x_n\right) \| p\left(Z \mid x_n\right)\right)$ decreases: This means that the lower bound we are optimizing gets closer to the objective $\log p(x_n)$, and that our approximate posterior q gets closer to the real posterior p, which is also good.

We can also have a combination of those two possibilities.

Question 6

Let $x_n \in D$. The expectation in $\mathbf{L_n^{recon}} = -E_{q_\phi(z|x_n)}\left[\log p_\theta\left(x_n \mid Z\right)\right]$ is approximated using one sample z_n sampled from the distribution $q_\phi\left(z_n \mid x_n\right)$, i.e. from $\mathcal{N}\left(\mu_\phi\left(x_n\right), \operatorname{diag}\left(\Sigma_\phi\left(x_n\right)\right)\right)$. Then $\mathbf{L_n^{recon}} \simeq -logp_\theta(x_n|z_n)$ is the negative log-likelihood of the data x_n given the distribution of the image generated by the decoder $f_\theta(z_n)$. In other words, this measures the (non-)similarity between the dataset image x_n and the generated image $f_\theta(z_n)$, hence the name reconstruction loss.

$\mathbf{L_n^{reg}}$ is the KL divergence between the approximation $q\phi(Z \mid x_n)$ and the standard gaussian $p_\theta(Z)$ and penalizes approximations that are too far from $\mathcal{N}(0, I_D)$, i.e. from 0, hence the name regularization loss.

Question 7

Let $x_n \in D$. The expectation in $\mathbf{L_n^{recon}} = -E_{q_\phi(z|x_n)}\left[\log p_\theta\left(x_n \mid Z\right)\right]$ is approximated using one sample z_n sampled from the distribution $q_\phi\left(z_n \mid x_n\right)$, i.e. from $\mathcal{N}\left(\mu_\phi\left(x_n\right), \operatorname{diag}\left(\Sigma_\phi\left(x_n\right)\right)\right)$. Then

$$\mathbf{L_n^{recon}} \simeq -logp_\theta(x_n|z_n) = -\sum_{m=1}^{M} x_n^{(m)} \log f_\theta(z_n)_m + (1 - x_n^{(m)}) \log(1 - f_\theta(z_n)_m)$$

Denote $\Sigma = \text{diag}(\Sigma_\phi(x_n))$

$$
\begin{aligned}
\mathbf{L_n^{reg}} &= D_{KL}(q_\phi(Z \mid x_n) \| p_\theta(Z)) \\
&= \int [\log(q_\phi(z \mid x_n)) - \log(p_\theta(z))] \times q_\phi(z \mid x_n)\, dz \\
&= \int \left[\frac{1}{2} \log \frac{|I_D|}{|\Sigma|} - \frac{1}{2}(z - \mu_\phi(x_n))^T \Sigma^{-1} (z - \mu_\phi(x_n)) + \frac{1}{2} z^T z \right] \times q_\phi(z \mid x_n)\, dz \\
&= \frac{1}{2} \log \frac{1}{|\Sigma|} - \frac{1}{2} \text{tr} \left\{ E[(z - \mu_\phi(x_n))(z - \mu_\phi(x_n))^T] \, \Sigma^{-1} \right\} + \frac{1}{2} E[z^T z] \\
&= \frac{1}{2} \log \frac{1}{|\Sigma|} - \frac{1}{2} \text{tr} \, (I_D) + \frac{1}{2} \mu_\phi(x_n)^T \mu_\phi(x_n) + \frac{1}{2} \text{tr}(\Sigma) \\
&= \frac{1}{2} \left[\log \frac{1}{|\Sigma|} - D + \text{tr}(\Sigma) + \mu_\phi(x_n)^T \mu_\phi(x_n) \right] \\
&= \frac{1}{2} \left[\sum_{i=1}^{D} (\Sigma_\phi(x_n)_i - \log \Sigma_\phi(x_n)_i) - D + \mu_\phi(x_n)^T \mu_\phi(x_n) \right].
\end{aligned}
$$

where in the fourth equality we use the property
$E\left[(z-m)^T A (z-m)\right] = (\mu_\phi(x_n) - m)^T A (\mu_\phi(x_n) - m) + \text{Tr}(A\Sigma)$ and in the fifth equality $E[(z - \mu_\phi(x_n))(z - \mu_\phi(x_n))^T] = \Sigma$

Question 8

The act of sampling prevents us from computing $\nabla \mathbf{L}_\phi = \frac{\partial \mathbf{L}}{\partial z_n} \frac{\partial z_n}{\partial \phi}$ because we cannot compute the gradient of a non-deterministic (hence non-continuous and non-differentiable) variable $z_n(\phi)$. The reparametrization trick solves the problem by redefining $z_n = \mu_\phi(x_n) + \text{diag}(\Sigma_\phi(x_n))^{1/2} \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, I_D)$ is treated as a fixed input. By doing this we can compute $\frac{\partial z_n}{\partial \phi}$ and $\nabla \mathbf{L}_\phi$

Question 9

The encoder is a three-layer MLP with two output layers, one for μ and one for σ. It takes a 28*28 input and returns two z_dim outputs. The decoder is a three-layer MLP which takes a z_dim input and returns a 28*28 output. The hyperparameters are the dimensions of the two hidden layers of the encoder (500,250) and the decoder (250,500), the latent dimension z_dim (20), the learning rate (0.0005) and the batch size (100). After defining the model we define the loss (sum of reconstruction and regularization loss) and for each epoch we sample a minibatch, forward pass it through the model, calculate the loss then backpropagate the error (see notebook for code).

Question 10

We obtain the following generated samples:

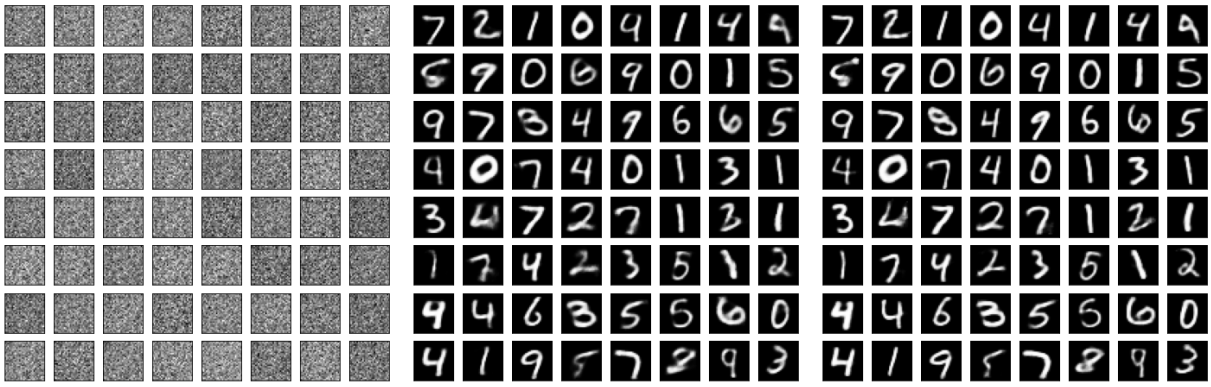

Figure 1: Generated samples after 0 epochs (left), 10 epochs (middle), 80 epochs (right)

Before training, the generated images are random noise. After 10 epochs, we obtain good results: the digits are recognizable and resemble the original MNIST digits, although some of them are blurry. After 80 epochs the output is slightly improved: most images become a bit less blurry, but the difference isn't very noticeable.

Overall the quality of the generated images is good (it is hard to distinguish them from the original MNIST images), but we also note that in rare cases the generated images contain blurry artifacts (e.g. fifth row, second column).

2. Backpropagation on Generative Adversarial Networks

Question 11

$$L\left(\theta_d, \theta_g\right) = \frac{1}{n}\sum_{i=1}^{n} \log D\left(X^i\right) + \log\left(1 - D\left(G\left(Z^i\right)\right)\right)$$

$$\Rightarrow \nabla_{\theta_d} L\left(\theta_d, \theta_g\right) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{D\left(X^i\right)} \nabla_{\theta_d} D\left(X^i\right) - \frac{1}{1 - D\left(G\left(Z^i\right)\right)} \nabla_{\theta_d} D\left(G\left(Z^i\right)\right)$$

Question 12

Like in the last question we can write:

$$\frac{\partial L\left(\theta_d, \theta_g\right)}{\partial z_d^{L_d}} = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{D\left(X^i\right)} \frac{\partial D\left(X^i\right)}{\partial z_d^{L_d}} - \frac{1}{1 - D\left(G\left(Z^i\right)\right)} \frac{\partial D\left(G\left(Z^i\right)\right)}{\partial z_d^{L_d}}$$

where $D(X^i) = g_d^{L_d}(z_d^{L_d}) = \sigma(z_d^{L_d})$, so:

$$\frac{\partial D\left(X^i\right)}{\partial z_d^{L_d}} = \sigma'(z_d^{L_d}) = \sigma(z_d^{L_d})(1 - \sigma(z_d^{L_d})) = D(X^i)(1 - D(X^i))$$

and similarly when the input is $G(Z^i)$ we have $D(G(Z^i)) = g_d^{L_d}(z_d^{L_d}) = \sigma(z_d^{L_d})$, so $\frac{\partial D\left(G\left(Z^i\right)\right)}{\partial z_d^{L_d}} = \sigma(z_d^{L_d})(1 - \sigma(z_d^{L_d})) = D(G(Z^i))(1 - D(G(Z^i)))$

$$\Rightarrow \frac{\partial L\left(\theta_d, \theta_g\right)}{\partial z_d^{L_d}} = \frac{1}{n}\sum_{i=1}^{n} \Big((1 - D(X^i)) + D(G(Z^i))\Big)$$

Question 13

Let $l \in \{1, .., L_d - 1\}$. We denote $(z_i^{l+1})_{i \in \{1,..,k_{l+1}\}}$ the elements of the vector z_d^{l+1}, $(w_i^{l+1})_{i \in \{1,..,k'_{l+1}\}}$ the vectors of the rows of W_d^{l+1}, and g_j^l the activation function of neuron j of layer l, i.e. $g_d^l(z_d^l) = (g_1^l(z_1^l), .., g_{k_l}^l(z_{k_l}^l))$.

We use the the multivariable chain rule:

$$\frac{\partial L\left(\theta_d, \theta_g\right)}{\partial z_d^l} = \sum_{i=1}^{k_{l+1}} \frac{\partial L\left(\theta_d, \theta_g\right)}{\partial z_i^{l+1}} \frac{\partial z_i^{l+1}}{\partial z_d^l}$$

We have $z_i^{l+1} = {w_i^{l+1}}^T g_d^l(z_d^l) = \sum_{j=0}^{k_l} (w_i^{l+1})_j (g_d^l(z_d^l))_j = \sum_{j=0}^{k_l} (w_i^{l+1})_j (g_j^l(z_j^l))$

$$\Rightarrow \frac{\partial z_i^{l+1}}{\partial z_j^l} = (w_i^{l+1})_j (g_j'^l(z_j^l)) = (w_i^{l+1})_j (g_d'^l(z_d^l))_j$$

$$\Rightarrow \frac{\partial z_i^{l+1}}{\partial z_d^l} = w_i^{l+1} \odot g_d'^l(z_d^l)$$

where the $\odot$ operator denotes component-wise vector multiplication.

$$\begin{aligned}
\Rightarrow \frac{\partial L\left(\theta_d, \theta_g\right)}{\partial z_d^l} &= \sum_{i=1}^{k_{l+1}} \frac{\partial L\left(\theta_d, \theta_g\right)}{\partial z_i^{l+1}} w_i^{l+1} \odot g_d'^l(z_d^l) \\
&= g_d'^l(z_d^l) \odot \sum_{i=1}^{k_{l+1}} \frac{\partial L\left(\theta_d, \theta_g\right)}{\partial z_i^{l+1}} w_i^{l+1} \\
&= g_d'^l(z_d^l) \odot \sum_{i=1}^{k_{l+1}} \left(\frac{\partial L\left(\theta_d, \theta_g\right)}{\partial z_d^{l+1}}\right)_i w_i^{l+1} \\
&= g_d'^l(z_d^l) \odot \left({w_d^{l+1}}^T \frac{\partial L\left(\theta_d, \theta_g\right)}{\partial z_d^{l+1}}\right)
\end{aligned}$$

where in the last equality we use the property $Ax = \sum_{i=1}^n A^{(i)} x_i = \sum_{i=1}^n (A^T)_{(i)} x_i$ with $A^{(i)}$ the columns of A and $A_{(i)}$ the rows of A.

Question 14

Here the input of the first layer of the discriminator ($g(z_i, \theta_g)$) is supplied post-activation, but this is equivalent to supposing that there exists a layer 0 having its activation g_d^0 equal to the identity and its output $g_d^0(z_d^0) = z_d^0 = g(z_i, \theta_g)$. Then we can easily verify that this is a special case of Q13 for $l = 0$ and that the intermediate steps remain correct, and we obtain:

$$\frac{\partial L\left(\theta_d, \theta_g\right)}{\partial g(z_i, \theta_g)} = \mathbf{1}_k \odot \left({w_d^1}^T \frac{\partial L\left(\theta_d, \theta_g\right)}{\partial z_d^1}\right) = \left({w_d^1}^T \frac{\partial L\left(\theta_d, \theta_g\right)}{\partial z_d^1}\right)$$

Question 15

Using the multivariable chain rule we can write:

$$\nabla_{\theta_g} L(\theta_d, \theta_g) = \sum_{i=1}^{n} \frac{\partial L(\theta_d, \theta_g)}{\partial g(z_i, \theta_g)} \frac{\partial g(z_i, \theta_g)}{\partial \theta_g}$$
$$\Rightarrow \nabla_{\theta_g} L(\theta_d, \theta_g) = \sum_{i=1}^{n} \frac{\partial L(\theta_d, \theta_g)}{\partial g(z_i, \theta_g)} \nabla_{\theta_g} g(z_i, \theta_g)$$

Question 16

The discriminator is optimized to maximize the log-likelihood $L(\theta_d, \theta_g)$, so θ_d^{t+1} is updated in the direction of the gradient:

$$\theta_d^{t+1} = \theta_d^t + \alpha \nabla_{\theta_d} L(\theta_d, \theta_g)$$

Question 17

The generator is optimized to minimize the log-likelihood $L(\theta_d, \theta_g)$, so θ_g^{t+1} is updated in the opposite direction of the gradient:

$$\theta_g^{t+1} = \theta_g^t - \alpha \nabla_{\theta_g} L(\theta_d, \theta_g)$$

3. Multi-lingual Translation

Question 18

In the original transformer architecture we apply LayerNorm after self-attention:

$$x_{l+1} = LayerNorm(x_l + \mathcal{A}(x_l))$$

We can write $x_{l+1} = LayerNorm(z_l)$ with $z_l = x_l + \mathcal{A}(x_l)$. Then

$$\frac{\partial x_{l+1}}{\partial x_l} = \frac{\partial x_{l+1}}{\partial z_l} \frac{\partial z_l}{\partial x_l} = \frac{\partial LayerNorm(z_l)}{\partial z_l} \left(1 + \frac{\partial \mathcal{A}(x_l)}{\partial x_l}\right)$$

Let $l_0 \in \{1, .., L-1\}$. By recursively applying the chain rule we can write the error backpropagation at layer l_0 as:

$$\frac{\partial e}{\partial x_{l_0}} = \frac{\partial e}{\partial x_L} \frac{\partial x_L}{\partial x_{L-1}} .. \frac{\partial x_{l_0+1}}{\partial x_{l_0}} = \frac{\partial e}{\partial x_L} \prod_{l=l_0}^{L-1} \frac{\partial x_{l+1}}{\partial x_l}$$

$$\Rightarrow \frac{\partial e}{\partial x_{l_0}} = \frac{\partial e}{\partial x_L} \prod_{l=l_0}^{L-1} \left[\frac{\partial LayerNorm(z_l)}{\partial z_l} \left(1 + \frac{\partial \mathcal{A}(x_l)}{\partial x_l}\right)\right]$$

In the alternative approach we apply LayerNorm before self-attention:

$$x_{l+1} = x_l + \mathcal{A}(LayerNorm(x_l)) = x_l + f(x_l)$$

with $f(x) = \mathcal{A}(LayerNorm(x))$. Therefore $\frac{\partial x_{l+1}}{\partial x_l} = 1 + \frac{\partial f(x_l)}{\partial x_l}$. We can reexpress this relation using a telescoping sum:

$$\begin{aligned}
x_L &= x_{l_0} + \sum_{l=l_0}^{L-1}(x_{l+1} - x_l) = x_{l_0} + \sum_{l=l_0}^{L-1} f(x_l) \\
\Rightarrow \frac{\partial x_L}{\partial x_{l_0}} &= 1 + \sum_{l=l_0}^{L-1} \frac{\partial f(x_l)}{\partial x_{l_0}} \\
\Rightarrow \frac{\partial e}{\partial x_{l_0}} &= \frac{\partial e}{\partial x_L}\frac{\partial x_L}{\partial x_{l_0}} = \frac{\partial e}{\partial x_L}\left(1 + \sum_{l=l_0}^{L-1} \frac{\partial f(x_l)}{\partial x_{l_0}}\right) \\
\Rightarrow \frac{\partial e}{\partial x_{l_0}} &= \frac{\partial e}{\partial x_L}\left(1 + \sum_{l=l_0}^{L-1} \frac{\partial \mathcal{A}(LayerNorm(x_l))}{\partial x_{l_0}}\right)
\end{aligned}$$

Question 19

When using Eq. 22 (the original version, which applies LayerNorm after self-attention), the error backpropagation equation at layer l_0 contains two products of $(L - l_0)$ gradients:

$$\frac{\partial e}{\partial x_{l_0}} = \frac{\partial e}{\partial x_L}\prod_{l=l_0}^{L-1}\left[\frac{\partial LayerNorm(z_l)}{\partial z_l}\right]\prod_{l=l_0}^{L-1}\left(1 + \frac{\partial \mathcal{A}(x_l)}{\partial x_l}\right)$$

This makes the model prone to vanishing gradients or exploding gradients, and the deeper the model, the more this effect is amplified. For example if we suppose $L = 21$, $l_0 = 1$ and $\frac{\partial \mathcal{A}(x_l)}{\partial x_l} = 0.1$, then $\prod_{l=l_0}^{L-1}\left(1 + \frac{\partial \mathcal{A}(x_l)}{\partial x_l}\right) \simeq 3325$ (exploding gradient), or if we suppose $\frac{\partial LayerNorm(z_l)}{\partial z_l} = 0.5$ then $\prod_{l=l_0}^{L-1} \frac{\partial LayerNorm(z_l)}{\partial z_l} \simeq 1.10^{-6}$ (vanishing gradient). Generally if we assume the gradients to be constant then the products are equivalent to an exponentional term in the number of layers, which explains again why this architecture makes training deep models harder because of the vanishing/exploding gradients problem.

When we use Eq. 23 instead (applying LayerNorm before self-attention), we only have a sum of the gradients:

$$\frac{\partial e}{\partial x_{l_0}} = \frac{\partial e}{\partial x_L}\left(1 + \sum_{l=l_0}^{L-1} \frac{\partial \mathcal{A}(LayerNorm(x_l))}{\partial x_{l_0}}\right)$$

In this case, even if the model is very deep, we don't have an exponential term in the number of layers so it is less prone to vanishing/exploding gradients and works better especially when using deep models.