Megi Dervishi
megi.dervishi@ens.fr

Assignment report
Deep Learning MVA 2021-2022

13/02/22

1 Variational Autoencoder

1. We have
$$p(x_n, z_n) = p(z_n)p(x_n|z_n) = N(z_n|0, I_D) \prod_{i=1}^{M} \text{Bern}(x_n^{(m)}|f_\theta(z_n)_m) \tag{1}$$
Hence, in order to sample our model we can first sample z_n from $N(0, I_D)$, called our prior distribution, and then sample from $p(x_n|z_n)$ which is known, and is called our posterior distribution. This process is called ancestral sampling.

2. Monte Carlo integration suffers from the curse of dimensionality. Indeed suppose that for our algorithm to achieve reasonable accuracy we have to sample the integration space with some density ρ. Then the number of samples required is $\sim \rho V$ where V is the volume of our integration space. However $V \sim \ell^d$ where ℓ is the characterstic distance of our integration space and d the dimension of our space. Hence the number of points needed to get a good result increase exponentially with the dimension of the space.

3. Let $q \sim N(\mu_q, \sigma_q^2)$ and $p \sim N(\mu_p, \sigma_p^2)$. Then we have that
$$p(x) = \frac{1}{\sigma_p\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu_p}{\sigma_p}\right)^2\right) \tag{2}$$
and hence
$$\log p(x) = -\log\sigma_p - \log\sqrt{2\pi} - \frac{1}{2}\left(\frac{x-\mu_p}{\sigma_p}\right)^2 \tag{3}$$
and similarly for q. Then
$$D_{KL}(p||q) = \mathbb{E}_p[\log(p) - \log(q)] \tag{4}$$
and using 3 we get
$$D_{KL}(p||q) = \mathbb{E}_p\left[\log\frac{\sigma_q}{\sigma_p} - \frac{1}{2}\left(\frac{x-\mu_p}{\sigma_p}\right)^2 + \frac{1}{2}\left(\frac{x-\mu_q}{\sigma_q}\right)^2\right]. \tag{5}$$
Now using the linearity of the expectation we can re-write
$$D_{KL}(p||q) = \log\frac{\sigma_q}{\sigma_p} - \frac{1}{2\sigma_p^2}\mathbb{E}_p\left[(x-\mu_p)^2\right] + \frac{1}{2\sigma_q^2}\mathbb{E}_p\left[(x-\mu_q)^2\right]. \tag{6}$$
Now notice that the expectancy of the second term can be re-written as
$$\mathbb{E}_p\left[(x-\mu_p)^2\right] = \mathbb{E}_p\left[(x-\mathbb{E}_p[x])^2\right] = \sigma_p^2. \tag{7}$$
The third term is trickier but can be re-expressed as
$$\mathbb{E}_p\left[(x-\mu_q)^2\right] = \mathbb{E}_p\left[((x-\mu_p)+\mu_p-\mu_q)^2\right] = \mathbb{E}_p\left[(x-\mu_p)^2\right] + \mathbb{E}_p\left[2(x-\mu_p)(\mu_p-\mu_q)\right] + (\mu_p-\mu_q)^2 \tag{8}$$
using the same re-writing as in 7 for the first term and the fact that $\mathbb{E}_p\left[(x-\mu_p)\right] = \mathbb{E}_p\left[x\right] - \mu_p = 0$ for the second term, we get
$$\mathbb{E}_p\left[(x-\mu_q)^2\right] = \sigma_p^2 + (\mu_q-\mu_p)^2. \tag{9}$$
Putting everything together we obtain
$$D_{KL}(p||q) = \log\frac{\sigma_q}{\sigma_p} + \frac{\sigma_p^2 + (\mu_q-\mu_p)^2}{2\sigma_q^2} - \frac{1}{2} \tag{10}$$
Hence we can see that for $\sigma_q = \sigma_p$ and $\mu_q = \mu_p$ then $D_{KL}(p||q) = 0$. Indeed as expected in this case the two distributions are identical. More generally if $\sigma_q = \sigma_p(1+\epsilon_\sigma)$ and $\mu_q = \mu_p + \epsilon_\mu$ then
$$D_{KL}(p||q) = \log(1+\epsilon_\sigma) + \frac{\sigma_p^2+\epsilon_\mu^2}{2\sigma_p^2(1+\epsilon_\sigma)^2} - \frac{1}{2} \simeq \epsilon_\sigma - \frac{\epsilon_\sigma^2}{2} + \frac{1}{2}(1-\epsilon_\sigma^2) + \frac{1}{2}\epsilon_\mu^2 = \epsilon_\sigma + \frac{1}{2}(\epsilon_\mu^2+1) - \epsilon_\sigma^2 \tag{11}$$

Hence we see that to a possible example for a very small divergence would be

$$D_{KL}(\mu_p, \sigma_p || \mu_p, \sigma_p + 0.1) \simeq 0.1 \tag{12}$$

Contrarily an example of a very big divergence would be

$$D_{KL}(0, \sigma || 10\sigma, \sigma) = \frac{\sigma^2 + 100\sigma^2}{2\sigma^2} - \frac{1}{2} = 50 \tag{13}$$

4. From equation 16 of the assignment, we can re-write

$$\log p(x_n) = \text{ELBO} + D_{KL}(q(Z|x_n)||p(Z|x_n)) \tag{14}$$

however note that the Kullback-Leibler divergence is non-negative hence $D_{KL}(q(Z|x_n)||p(Z|x_n)) \geq 0$, hence

$$\log p(x_n) \geq \text{ELBO} \tag{15}$$

We optimize the lower bound ELBO instead of optimizing $p(x_n)$ directly because as we have seen in Question 2, computing $p(x_n)$ is extremely costly and completely intractable in high dimensions, however ELBO is tractable. Note also that $D_{KL}(q(Z|x_n)||p(Z|x_n))$ is also intractable to compute by the no-free-lunch theorem.

5. When maximizing ELBO one of two things can happen
   (a) Either $D_{KL}(q(Z|x_n)||p(Z))$ decreases, meaning that our posterior distribution q becomes 'closer' to the prior distribution p.
   (b) Either the expectancy $\mathbb{E}_{q(z_n|x_n)} \log p(x_n|z_n)$ increases, meaning that our estimation of the latent variables is becoming more accurate.

6. The reconstruction loss is called in such a way because it is a term that forces the model to predict good latent variables which accurately describe the model and hence allow to reconstruct the image. The regularization loss is called in such a way because it forces the posterior distribution not to wander 'too far away' from the actual prior distribution, hence acting as a regularization.

7. We first detail the regularization loss. As said in Section 1.3 we take $p \sim N(0, 1)$ as is usual to do for VAEs. Furthermore using equation (18) from the assignment, we get

$$\mathbf{L_n^{reg}} = D_{KL}(q(z_n|x_n)||p_\theta(z_n)) = D_{KL}(N(z_n|\mu_\phi(x_n), \text{diag}(\Sigma_\phi(x_n)))||N(0, 1)) \tag{16}$$

Now using the formula derived in Question 3 (or equation (12) in the assignment) we get

$$\mathbf{L_n^{reg}} = \frac{1}{2}\left[\mu_\phi(x_n)^2 + \text{Tr diag}(\Sigma_\phi(x_n)) - M - \log|\text{diag}(\Sigma_\phi(x_n))|\right] \tag{17}$$

using the identities for the trace and the determinant of a diagonal matrix we get

$$\mathbf{L_n^{reg}} = -M + \frac{1}{2}\sum_{m=1}^{M} \mu_\phi(x_n)_m^2 + \Sigma_\phi(x_n)_m - \log \Sigma_\phi(x_n)_m \tag{18}$$

Now for the reconstruction loss, as is said in the previous question, we can approximate the expectancy by sampling and hence

$$\mathbf{L_n^{recon}} = -\mathbb{E}_{q_\phi(z_n|x_n)}[\log p_\theta(x_n|z_n)] \simeq -\frac{1}{L}\sum_{\ell=1}^{L} \log \theta p(x_n|z_n^{(\ell)}) \text{ where } z_n^{(\ell)} \sim q_\phi(z|x_n) \tag{19}$$

and as is said in the previous question it is customary for VAEs to take $L = 1$. The loss term is now explicit.

8. When performing gradient descent we need first of all to compute the gradients of the loss with respect to our parameters. There are no problems computing $\nabla_\theta \mathbf{L}$ however there is a problem for $\nabla_\phi \mathbf{L^{recon}}$ since

$$\nabla_\phi \mathbb{E}_{q_\phi}[f(z)] \neq \mathbb{E}_{q_\phi}[\nabla_\phi f(z)] \tag{20}$$

for any function f. However let us re-express

$$z = g(\varepsilon, \phi, x) \text{ and } \varepsilon \sim p(\varepsilon) \tag{21}$$

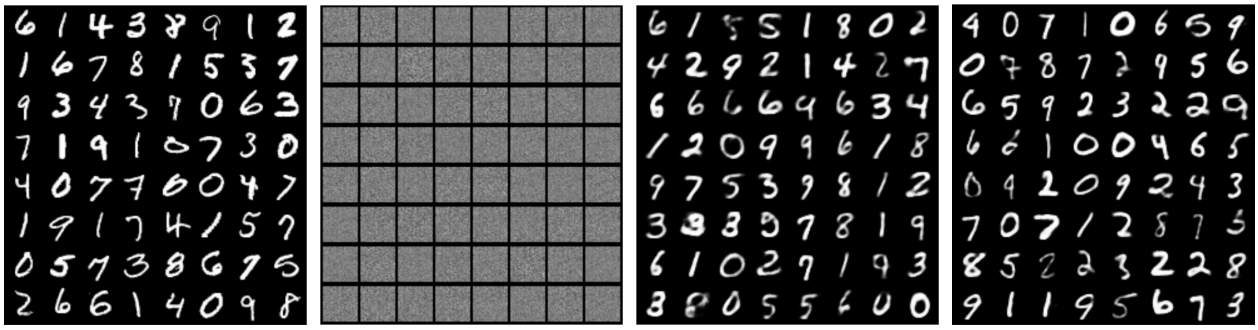
where g has to be C^1 in all its arguments. Then we can re-write the previous gradient as

$$\nabla_\phi \mathbb{E}_{q_\phi}[f(z)] = \nabla_\phi \mathbb{E}_{p(\varepsilon)}\left[f(g(\varepsilon, \phi, x)\right] = \mathbb{E}_{p(\varepsilon)}\left[\nabla_\phi f(g(\varepsilon, \phi, x)\right] = \mathbb{E}_{p(\varepsilon)}[\nabla_\phi f(z)] \tag{22}$$

which solves our problem.

9. For the VAE Encoder I have a 2 layer MLP with input size 784, hidden size 512, output size twice the dimension of the latent variable space and ReLU activations. We need to output twice the dimension of the latent variable space because we need to encode both the mean and the diagonal of the co-variance matrix. For the decoder I have a 2 layer MLP with input size the dimension of the latent variable space, hidden size 512, output size 784 and ReLU activations. I used an Adam optimizer with a learning rate of 10^{-4} a batch size of 128 and trained for 80 epochs.

10. As we can see in Figure [1], the VAE is able to model quite well the MNIST dataset. However we can notice that even after 80 epochs VAE still has some trouble modelling different 'penstrokes'. We see some numbers which are almost transparent for example. Nonetheless the quality of the digits is extremely good.



(a) Ground Truth (b) Before Training (c) After 10 epochs. (d) After 80 epochs.

Figure 1: 64 Samples after different training steps.

2 Back propagation on Generative Adversarial Networks.

11. We have that

$$\nabla_{\theta_d} L(X;\theta_d,\theta_g) = \nabla_{\theta_d}\left[\log D(x) + \log(1 - D(G(Z)))\right] = \frac{\nabla_{\theta_d} D(X)}{D(X)} - \frac{\nabla_{\theta_d} D(G(Z))}{1 - D(G(Z))} \tag{23}$$

12. We have that

$$D(X) = g_d^{L_d}(\cdots g_d^2(W_d^2(\overbrace{g_d^1(\underbrace{W_d^1 X}_{z_d^1})}^{A_d^1}))\cdots) \tag{24}$$

A_d^2 over the whole expression; z_d^2 under the whole expression.

More specifically for this question we can simply write

$$D(X) = g_d^{L_d}(z_d^{L_d}) = \text{sigmoid}(z_d^{L_d}) \tag{25}$$

Now notice that

$$\frac{\text{sigmoid}'(z)}{\text{sigmoid}(z)} = \text{sigmoid}(-z) = 1 - \text{sigmoid}(z) \tag{26}$$

and similarly

$$\frac{\text{sigmoid}'(z)}{1 - \text{sigmoid}(z)} = \text{sigmoid}(z) \tag{27}$$

so applying this to the result of our previous question we obtain

$$\frac{\partial D(X)}{\partial z_d^{L_d}} = \text{sigmoid}(-z_d^{L_d}(X)) - \text{sigmoid}(z_d^{L_d}(G(X))) \tag{28}$$

13. A basic application of the chain rule tells us that

$$\frac{\partial L(\theta_d,\theta_g)}{\partial z_d^\ell} = \frac{\partial L(\theta_d,\theta_g)}{\partial z_d^{\ell+1}} \frac{\partial z_d^{\ell+1}}{\partial z_d^\ell} \tag{29}$$

Now the second term is given by

$$\frac{\partial z_d^{\ell+1}}{\partial z_d^\ell} = \frac{\partial}{\partial z_d^\ell} g_d^\ell(W_d^{\ell+1} z_d^\ell) = {g'_d}^\ell(W_d^\ell z_d^\ell) W_d^{\ell+1} \tag{30}$$

so plugging it back upstairs we get

$$\frac{\partial L(\theta_d, \theta_g)}{\partial z_d^\ell} = \frac{\partial L(\theta_d, \theta_g)}{\partial z_d^{\ell+1}} {g'_d}^\ell(W_d^\ell z_d^\ell) W_d^{\ell+1} \tag{31}$$

14. Similarly we have that

$$\frac{\partial L(\theta_d, \theta_g)}{\partial g(z_i, \theta_g)} = \frac{\partial L(\theta_d, \theta_g)}{\partial z_d^1} \frac{\partial z_d^1}{\partial g(z_i, \theta_g)} = \frac{\partial L(\theta_d, \theta_g)}{\partial z_d^1} W_g^1 \tag{32}$$

since as we have seen before we have that $z_d^1 = W_g^1 g$.

15. Using the chain rule once more we get

$$\boldsymbol{\nabla}_{\theta_g} L(\theta_d, \theta_g) = \frac{\partial L(\theta_d, \theta_g)}{\partial g(z_i, \theta_g)} \boldsymbol{\nabla}_{\theta_g} g(z_i, \theta_g) \tag{33}$$

16. Now we can simply apply gradient ascent for the discriminator

$$\theta_d^{t+1} = \theta_d^t + \alpha \boldsymbol{\nabla}_{\theta_d} L(\theta_d, \theta_g) \tag{34}$$

17. And a gradient descent for the generator

$$\theta_g^{t+1} = \theta_g^t - \alpha \boldsymbol{\nabla}_{\theta_g} L(\theta_d, \theta_g) \tag{35}$$

3 Multi-Lingual Translation.

18. Succesive chain rule applications yield

$$\frac{\partial e}{\partial x_\ell} = \frac{\partial e}{\partial x_L} \frac{\partial x_L}{\partial x_\ell} = \frac{\partial e}{\partial x_L} \prod_{i=\ell}^{L} \frac{\partial x_{i+1}}{\partial x_i} \tag{36}$$

Now supposing $x_{i+1} = \text{LayerNorm}(x_i + \mathcal{A}(x_i))$ we obtain

$$\frac{\partial e}{\partial x_\ell} = \frac{\partial e}{\partial x_L} \prod_{i=\ell}^{L} \text{LayerNorm}'(x_i + \mathcal{A}(x)) \left(1 + \mathcal{A}'(x_i)\right) \tag{37}$$

If instead we have that $x_{i+1} = x_i + \text{LayerNorm}(\mathcal{A}(x_i))$ then we get

$$x_L = x_\ell + \sum_{i=\ell}^{L-1} \text{LayerNorm}(\mathcal{A}(x_i)) \tag{38}$$

so we can immediately compute the partial derivative and we get

$$\frac{\partial e}{\partial x_\ell} = \frac{\partial e}{\partial x_L} \left(1 + \frac{\partial}{\partial x_\ell} \sum_{i=\ell}^{L-1} \text{LayerNorm}(\mathcal{A}(x_i))\right) \tag{39}$$

19. Notice how the first choice (equation (22) in the assignment) leads to an error which is expressed by the product of the gradient at every step, while on the other hand the second choice (equation (23) in the assignment) leads to the contribution being expressed as a sum. This fundamental difference makes it so that the second option is much more stable, as with the first option gradients can very quickly either go to 0 (vanish) if they are successively smaller than 1 or explode if they are successively bigger than 1. This instability is further accentuated the deeper the network is, as more terms will appear in the product.

20. In order to observe these vanishing/exploding gradients I simply look at the norm of the gradients in the last layer during the training iterations. I measured both for a shallow (1 layer) transformer as well as for a deeper (5 layer) transformer and compared the difference with taking the norm before or after the iterative additive step as can be seen in Figure 2. As expected we see that taking the normalization before leads to more stable results.

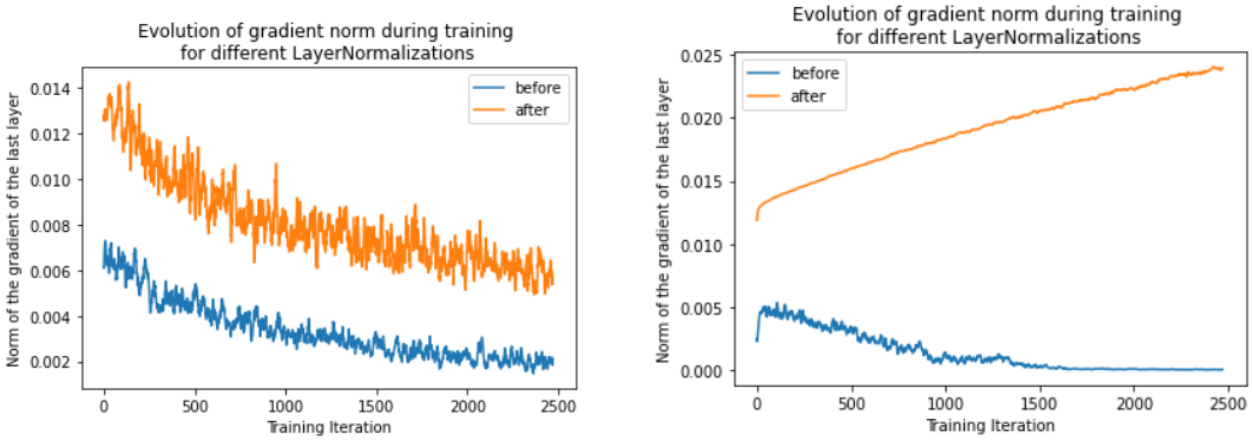


Figure 2: Evolution of the norm of the gradient of the last layer during training, comparing whether to put the normalization before or after adding to x_ℓ and the difference between a shallow (left.) /deep (right.) network.