3MD3020: Deep Learning
MVA Master

Assignment 1

Due: January 22, 2020

Information about the exercise:

- Please complete the first assignment **individually**.
- Please send your answers on a **PDF** file together with your source code in order to produce the answers.
- For your answers please provide short and clear descriptions with figures where ever this is possible.
- Please do not forget to write your name on the reports and submit your assignment on 3md3020@gmail.com by the day of the deadline, using as title "3md3020_Assignment".
- The total number of points is 130 to cover also the optional questions.
- Good luck!

A. Computer Vision Part

VAEs leverage the flexibility of neural networks (NN) to learn and specify a latent variable model. Mathematically, they are connected to a distribution ($p(x)$) over x in the following way: $p(x) = \int p(x|z)p(z)dz$. This integral is typically too expensive to evaluate, which VAEs have resolved in a way that you learn in this assignment.

0.1 Latent Variable Models

A latent variable model is a statistical model that contains both observed and unobserved (i.e. latent) variables. Assume a dataset $D = \{x_n\}_{n=1}^{N}$, where $x_n \in \{0,1\}^M$. For example, x_n can be the pixel values of a binary image.

$$z_n \sim N(0, I_D) \tag{1}$$

$$x_n \sim p_X(f_\theta(z_n)) \tag{2}$$

where f_θ is some function – parameterized by θ – that maps z_n to the parameters of a distribution over x_n. For example, if p_X would be a Gaussian distribution we will use $f_\theta : R^D \to (R^M, R_+^M)$ for a mean and covariance matrix, or if p_X is a product of Bernoulli distributions, we have $f_\theta : R^D \to [0,1]^M$. Here, D denotes the dimensionality of the latent space. Note that our dataset D does not contain z_n, hence z_n is a latent (or unobserved) variable in our statistical model. In the case of a VAE, a (deep) NN is used for $f_\theta(\cdot)$.

0.2 Decoder: The Generative Part of the VAE

In the previous section, we described a general graphical model which also applies to VAEs. In this section, we will define a more specific generative model that we will use throughout this assignment. This will later be refered to as the decoding part (or decoder) of a VAE. For this assignment, we will assume the pixels of our images x_n in D are Bernoulli (p) distributed.

$$p(z_n) = N(0, I_D) \tag{3}$$

$$p(x_n|z_n) = \prod_{m=1}^{M} Bern(x_n^{(m)}|f_\theta(z_n)_m) \tag{4}$$

where $x_n^{(m)}$ is the m-th pixel of the n-th image in D, and $f_\theta{:}R^D \to [0,1]^M$ is a neural network parameterized by θ that outputs the means of the Bernoulli distributions for each pixel in x_n.

Question 1 [5 points] Describe the steps needed to sample from such a model. (Hint: ancestral sampling)

Now that we have defined the model, we can write out an expression for the log probability of the data D under this model:

$$\log(p(D)) = \sum_{n=1}^{N} \log p(x_n) = \sum_{n=1}^{N} \log \int p(x_n|z_n)p(z_n)dz_n = \sum_{n=1}^{N} \log E_{p(z_n)}[p(x_n|z_n)] \tag{5}$$

Evaluating $\log p(x_n) = \log E_{p(z_n)}[p(x_n|z_n)]$ involves a very expensive integral. However, Eq. 5 hints at a method for approximating it, namely Monte-Carlo Integration. The log-likelihood can be approximated by drawing samples $z_n^{(l)}$ from $p(z_n)$:

$$\log(p(x_n)) = \log E_{p(z_n)}[p(x_n|z_n)] \tag{6}$$

$$\approx \log \frac{1}{L} \sum_{l=1}^{L} p(x_n|z_n^{(l)}), \quad z_n^{(l)} \sim p(z_n) \tag{7}$$

If we increase the number of samples L to infinity, the approximation would be equals to the actual expectation. Hence, the estimator is unbiased and can be used to approximate $\log p(x_n)$ with a sufficient large number of samples.

Question 2 [5 points] Although Monte-Carlo Integration with samples from $p(z_n)$ can be used to approximate $\log p(x_n)$, it is not used for training VAE type of models, because it is inefficient. In a few sentences, describe why it is inefficient and how this efficiency scales with the dimensionality of z.

0.3 KL Divergence

Before continuing our discussion about VAEs, we will need to learn about another concept that will help us later: the Kullback-Leibner divergence (KL divergence). It measures how different one probability distribution is from another:

$$D_{KL}(q||p) = -E_{q(x)}[\log \frac{p(X)}{q(X)}] = -\int q(x)[\log \frac{p(x)}{q(x)}]dx, \tag{8}$$

where q and p are probability distributions in the space of some random variable X.

Question 3 [5 points] Assume that q and p in Eq. 8, are univariate gaussians: $q = N(\mu_q, \sigma_q^2)$ and $p = N(\mu_p, \sigma_p^2)$. Give two examples of $(\mu_q, \mu_p, \sigma_q^2, \sigma_p^2)$: one of which results in a very small, and one of which has a very large, KL-divergence: $D_{KL}(q||p)$.

In VAEs, we usually set the prior to be a normal distribution with a zero mean and unit variance: $p = N(0,1)$. For this case, we can actually find a closed-form solution of the KL divergence:

$$KL(q,p) = -\int q(x)\log p(x)dx + \int q(x)\log p(x)dx \tag{9}$$

$$= \frac{1}{2}\log(2\pi\sigma_p^2) + \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} - \frac{1}{2}(1 + \log 2\pi\sigma_q^2) \tag{10}$$

$$= \log\frac{\sigma_p}{\sigma_q} + \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} - \frac{1}{2} \tag{11}$$

$$= \frac{\sigma_q^2 + \mu_q^2 - 1 - \log\sigma_q^2}{2} \tag{12}$$

For simplicity, we skipped a few steps in the derivation. You can find the details here if you are interested (it is not essential for understanding the VAE). We will need this result for our implementation of the VAE later.

0.4 The Encoder: $q_\phi(z_n|x_n)$ - Efficiently evaluating the integral

In the previous section 0.2, we have developed the intuition that we only want to sample z_n for which $p(z_n|x_n)$ is not close to zero - in order to compute the Monte-Carlo approximation. Unfortunately, the true posterior $p(z_n|x_n)$ is as difficult to compute as $p(x_n)$ itself. To solve this problem, instead of modeling the true posterior $p(z_n|x_n)$, we can learn an approximate posterior distribution, which we refer to as the variational distribution. This variational distribution $q(z_n|x_n)$ is used to approximate the (very expensive) posterior $p(z_n|x_n)$ and to more efficiently integrate $\int p(x_n|z_n)p(z_n)dz_n$.

Now, we have all the tools to derive an efficient bound on the log-likelihood $\log p(D)$. We start from Eq. 5 where the log-likelihood objective is written, but for simplicity in notation we write the log-likelihood $\log p(x_n)$ only for a single datapoint.

$$\log p(x_n) = \log E_{p(z_n)}[p(x_n|z_n)] = \log E_{p(z_n)}[\frac{q(z_n|x_n)}{q(z_n|x_n)}p(x_n|z_n)] = \log E_{q(z_n|x_n)}[\frac{p(z_n)}{q(z_n|x_n)}p(x_n|z_n)] \tag{13}$$

$$\geq E_{q(z_n|x_n)} \log[\frac{p(z_n)}{q(z_n|x_n)}] = E_{q(z_n|x_n)}[\log p(x_n|z_n)] + E_{q(z_n|x_n)} \log[\frac{p(z_n)}{q(z_n|x_n)}] = E_q(z_n|x_n)[\log p(x_n|z_n)] - KL(q(Z|x_n)||p(Z)) \tag{14}$$

We have derived a bound on $\log p(x_n)$, exactly the thing we want to optimize, where all terms on the right hand side are computable. Let's put together what we have derived again in a single line:

$$\log p(x_n) \geq E_{q(z_n|x_n)}[\log p(x_n|z_n) - KL(q(Z|x_n)||p(Z))] \tag{15}$$

The right side of the equation is referred to as the evidence lowerbound (ELBO) on the log-probability of the data. This leaves us with the question: How close is the ELBO to $\log p(x_n)$? With an alternate derivation[1] we can find the answer. It turns out the gap between $\log p(x_n)$ and the ELBO is exactly $KL(q(Z|x_n)||p(Z|x_n))$ such that:

$$\log p(x_n) - KL(q(Z|x_n)||p(Z|x_n)) = E_{q(z_n|x_n)}[\log p(x_n|z_n)] - KL(q(Z|x_n)||p(Z)) \tag{16}$$

Now, let's optimize the ELBO. For this, we define our loss as the mean negative lower bound over samples:

$$\mathbf{L}(\theta, \phi) = -\frac{1}{N}\sum_{n=1}^{N} E_{q_\phi(z|x_n)}[\log p_\theta(x_n|Z)] - D_{KL}(q_\phi(Z|x_n)||p_\theta(Z)) \tag{17}$$

Note, that we make an explicit distinction between the generative parameters θ and the variation parameters ϕ.

Question 4 [5 points] Explain how you can see from Eq. 16 that the right hand side had to be a lower bound of the log-probability $\log p(x_n)$. Why must we optimize the lower-bound, instead of optimizing the log-probability $\log p(x_n)$ directly?

Question 5 [5 points] Now, looking at the two terms on left-hand side of Eq. 16: Two things can happen when the lower-bound is pushed up. Can you describe what these two thing are?

0.5 Specifying the Encoder $q_\phi(z_n|x_n)$

In VAE, we have some freedom to choose the distribution $q_\phi(z_n|x_n)$. In essence, we want to choose something that can closely approximate $p(z_n|x_n)$, but we are also free to a select distribution that makes our life easier. We will do exactly that in this case and choose $q_\phi(z_n|x_n)$ to be factored multivariate normal distribution, i.e.,

$$q_\phi(z_n|x_n) = N(z_n|\mu_\phi(x_n), diag(\Sigma_\phi(x_n))), \tag{18}$$

where $\mu_\phi : R^M \rightarrow R^D$ maps an input image to the mean of the multivariate normal over z_n and $\Sigma_\phi : R^D \rightarrow R^M_{>0}$ maps the input image to the diagonal of the covariance matrix of that same distribution Moreover, $diag(v)$ maps a K-dimensional (for any K) input vector v to a $K \times K$ matrix such that for $i, j \in \{1, ..., K\}$.

$$diag(v)_{ij} = \begin{cases} v_i, & \text{if } i = j. \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

Question 6 [5 points] The loss in Eq. 17 can be rewritten in terms of per-sample losses:

$$\mathbf{L} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{L_n^{recon}} + \mathbf{L_n^{reg}}) \tag{20}$$

where $\mathbf{L_n^{recon}} = -E_{q_\phi(z|x_n)}[\log p_\theta(x_n|Z)]$ and $\mathbf{L_n^{reg}} = D_{KL}(Q_(Z|x_n)||p_\theta(Z))$ can be seen as a reconstruction loss term and an regularization term, respectively. Explain why the names reconstruction and regularization are appropriate for these two losses. (Hint: Suppose we use just one sample to approximate the expectation $E_{\phi(z|x_n)}[p_\theta(_n|Z)]$ – as is common practice in VAEs.)

[1]This derivation is not done here, but can be found in for instance Bishop sec 9.4

Question 7 [5 points] Now we have defined an objective (Eq. 17) in terms of an abstract model and variational approximation, we can put everything together using our model definition (Equations 1, 2) and definition of $q_\theta(z_n|x_n)$ (Eq. 18), and we can write down a single objective which we can minimize. Write down expressions (including steps) for $\mathbf{L_n^{recon}}$ and $\mathbf{L_n^{reg}}$ such that we can minimize $\mathbf{L} = \sum_{n=1}^{N}(\mathbf{L_n^{recon}} + \mathbf{L_n^{reg}})$ as our final objective. Make any approximation explicit. (Hint: look at Eq. 9 for $\mathbf{L_n^{reg}}$).

0.6 The Reparametrization Trick

Although we have written down (the terms of) an objective in Question 7, we still cannot simply minimize this by taking gradients with regard to θ and ϕ. This is due to the fact that we sample from $q_\phi(z_n|x_n)$ to approximate the $E_{q_\phi(z|x_n)}[\log p_\theta(x_n|Z)]$ term. Yet, we need to pass the derivative through these samples if we want to compute the gradient of the encoder parameters, i.e., $\nabla_\phi \mathbf{L}(\theta, \phi)$. Our posterior approximation $q_\phi(z_n|x_n)$ is parameterized by ϕ. If we want to train $q_\phi(z_n|x_n)$ to maximize the lower bound, and therefore approximate the posterior, we need to have the gradient of ϕ with respect to the lower-bound.

Question 8 [5 points] Passing the derivative through samples can be done using the reparameterization trick. In a few sentences, explain why the act of sampling usually prevents us from computing $\nabla_\phi \mathbf{L}$, and how the reparameterization trick solves this problem. (Hint: you can take a look at Figure 4 from the tutorial by Carl Doersch)

0.7 Putting things together: Building a VAE

Given everything we have discussed so far, we now have an objective (the evidence lower boundor ELBO) and a way to backpropagate to both θ and ϕ (i.e., the reparametrization trick). Thus, we can now implement a VAE in PyTorch to train on MNIST images. We will model the encoder $q(z|x)$ and decoder $p(x|z)$ by a deep neural network each, and train them to maximize the data likelihood.

Question 9 [5 points] Build a Variational Autoencoder, and train it on the binarized MNIST dataset. Both the encoder and decoder should be implemented as an MLP. Following standard practice – and for simplicity – you may assume that the number of samples used to approximate the expectation in $\mathbf{L_n^{recon}}$ is 1. Use a latent space size of $z_{dim} = 20$. In your report, provide a short description (no more than 10 lines) of the used architectures for the encoder and decoder, any hyperparameters and your training steps.

Question 10 [5 points] Plot 64 samples (8×8grid) from your model at three points throughout training (before training, after training 10 epochs, and after training 80 epochs). You should observe an improvement in the quality of samples. Describe shortly the quality and/or issues of the generated images.

(Optional) Question 11 [10 points] In the previous questions, we have used MLPs for modeling the encoder and decoder. Next, implement a convolutional encoder and decoder network. For the architecture you can use a U-Net [1] like architecture. In your report, provide a short description (no more than 10 lines) of the used architectures for the encoder and decoder, any hyperparameters and your training steps. Also include samples of the model as in Question 10. What differences can you see compared to the MLP networks.

B. Natural Language Processing Part

0.8 RNNs

In this assignment you will be asked to implement a seq2seq model and test it on a translation task. Have a look at this pytorch tutorial: and answer the following questions. Provide empirical evidence to your answers. Each point should be answered in a few concise paragraphs commenting any resulting graph.

Question 12. [10 points] Analyze how attention is distributed along the tokens. Do you observe any special patterns for EOS/SOS tokens? What happens if you remove EOS? Do you have any interpretation for such behaviour?

Question 13. [5 points] Why is attention so important in this model? What would happen if it gets removed?

Question 14. [5 points] What is teacher forcing? Why would you want to use teacher forcing?

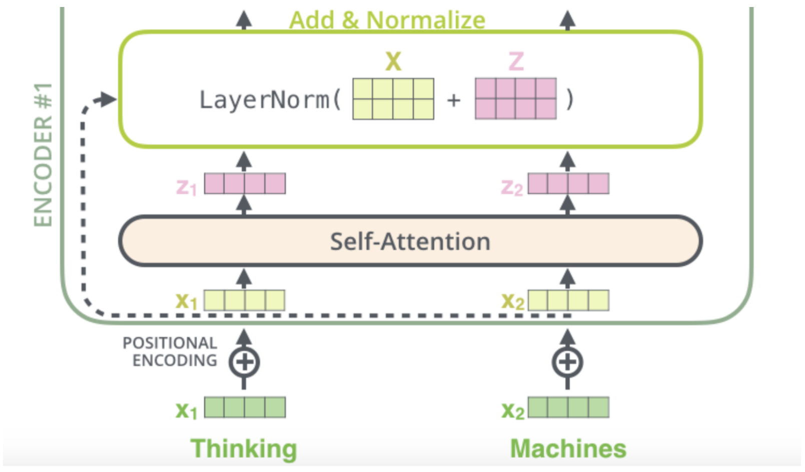

Figure 1: Original sequence of layers.The image is coming from this source

0.9 Transformers

The original Transformer architecture [2] proposed the sequence of layers shown in Fig. 1. Note in particular that the normalization occurs after the self-attention operation, with an intermediate addition of the input embedding (the residual connection). In equation, and denoting by $\mathcal{A}(x)$ the self-attention layer:

$$x_{\ell+1} = \mathit{LayerNorm}(x_\ell + \mathcal{A}(x)) \tag{21}$$

where for simplicity we suppose an architecture without a feed-forward layer. An alternative which is often used is to do the layer normalization before the attention layer:

$$x_{\ell+1} = x_\ell + \mathcal{A}(\mathit{LayerNorm}(x_\ell)) \tag{22}$$

Question 15. [15 points] Derive analytically the impact of that different sequence of layers, when doing back-propagation during training. (Hint: If you denote by e the error produced in one pass, and x_L the top-most layer, then by applying chain rule you obtain:)

$$\frac{\partial e}{\partial x_\ell} = \frac{\partial e}{\partial x_L}\frac{\partial x_L}{\partial x_\ell}$$

Derive then $\frac{\partial x_{\ell+1}}{\partial x_\ell}$ (no need of actually deriving $\frac{\partial \mathit{LayerNorm}(x)}{\partial x}$).

Question 16. [15 points] Explain in a few paragraphs what do you think are the consequences of training (very deep models) of those derivations when using Eq 22 instead of Eq. 21?

(Optional) Question 17. [20 points] Show empirically the impact of this. This is, design a measure and plot that measure against training steps for a standard task (eg: language modelling), comparing those plots between a deep and shallow model. Finding what exactly to measure is an important part of the problem.

References

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.