DEEP LEARNING ASSIGNMENT 1

Gladys ROCH

January 22, 2021

1 Computer Vision Part

1.1 Latent Variable Models

1.2 Decoder: The Generative Part of the VAE

Question 1.
We want to estimate the probability of x_n, $p(x_n)$. Because we know the probability $p(z_n)$ and $p(x_n|z_n)$ we can proceed by forward (ancestral) sampling. Forward sampling consists in sampling from the parent node (here Z) to the children nodes (X): we sample a z_i from the standard normal distribution, then, given the value of z_n we draw x_n.

Question 2.
First, the Monte Carlo Integration consists in evaluating the integrand at random points of the integration domain and then multiplying the average by the length of the integration domain. The value converges towards the true value of the integral when the number of sampling points tends to infinity.

The issue with the Monte-Carlo approximation is that when z has a high dimensionality, the number of samples needed to reach an accurate estimate of the log-likelihood may be extremely large.

1.3 KL Divergence

Question 3.
The KL divergence account for the amount of information lost when the law of probability q is used to approximate p.

• If p = q (for example if $\mu_q = \mu_p = 0$ and $\sigma_q^2 = \sigma_p^2 = 1$), then $D_{KL}(p||q) = 0$.
• The KL divergence is large when the two distributions p and q are very different from each other. Let's take p the standard normal distribution $\mu_p = 0$ and $\sigma_p = 1$ and $\mu_q = 10$ and $\sigma_q = 1$. In this case $D_{KL}(p||q) = 49.5$

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

Question 4.
Equation (16) gives :

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

Since the KL divergence between $q(Z|x_n)$ and $p(Z|x_n)$ is positive, it easily comes that the right-hand of the initial Eq.(16) is a lower bound for $\log p(x_n)$.

We optimize the lower-bound, instead of optimizing the log-probability $\log p(x_n)$ directly, because the latter involves the intractable estimation of $p(x_n)$.

Question 5.

Looking at $\log p(x_n) - KL(q(Z|x_n)||p(Z|x_n))$ we can see that when the ELBO is pushed up, two things can happen:
• the KL divergence can decrease to zero (in this case the q approximates the p accurately),
• the log-probability can increase.

1.5 Specifying the Encoder $q_{(z_n|x_n)}$

Question 6.
The per-sample loss used to optimize the ELBO is :$L = \frac{1}{N}\sum_{n=1}^{N}(L_n^{recon} + L_n^{reg})$.

• $L_n^{recon} = -E_{q_\phi(z|x_n)}[\log p(x_n|Z)]$ is called the reconstruction loss term because aims at maximizing the log likelihood, which makes the generated image, $p(x_n|Z)$, more correlated to the latent variable (Z), which makes the model more deterministic.
•$L_n^{reg} = KL(q(Z|x_n)||p(Z))$ is called the regularization term because minimizing the KL forces p to be similar to the input $q(.|x_n)$ (i.e. forces p to be a Gaussian distribution).

Question 7.
$L_n^{recon} = -E_{q_\phi(z|x_n)}[\log p(x_n|Z)]$
but $q_\phi(z_n|x_n) = N(z_n|_\phi(x_n), diag(\Sigma_\phi(x_n)^2))$
and $p(x|z) = \prod_{n=1}^{N} Bern(x_n^{(m)}|f_\theta(z_n)_m)$
thus, $L_n^{recon} =$

On the other hand,
$L_n^{reg} = KL(q(Z|x_n)||p(Z)) = \log\frac{\sigma_p}{\sigma_q} + \frac{\sigma_q^2+(\mu_q-\mu_p)^2}{2\sigma_p^2} - \frac{1}{2}$
but $p(z_n) = N(0, I_D)$
and $q_\phi(z_n|x_n) = N(z_n|_\phi(x_n), diag(\Sigma_\phi(x_n)))$
thus, $L_n^{reg} = -\log\sigma_q + \frac{\sigma_q^2+\mu_q^2-1}{2}$

1.6 The Reparametrization Trick

Question 8.
The act of sampling usually prevent us from computing the gradient of the loss function because the sampling layer is a non-continuous operation and has therefore no gradient. Backpropagation does not support stochastic layers.

The solution to this problem is to move the sampling from a computational layer to an input layer via a change of variable. Instead of sampling z_n from the gaussian distribution $q(z_n|x_n)$ we sample ϵ_n from a standard normal distribution $\mathcal{N}(0,1)$ (which independent from the network) and then apply to the sample an affine transformation ($z_n = \Sigma_\phi(x_n)^2 \odot \epsilon_n + z_n|_\phi(x_n)$). This way the backpropagation only deals with a stochastic input and and affine layer.

1.7 Putting things together: Building a VAE

Question 9.
The architectures of the encoder and decoder are symetrical.

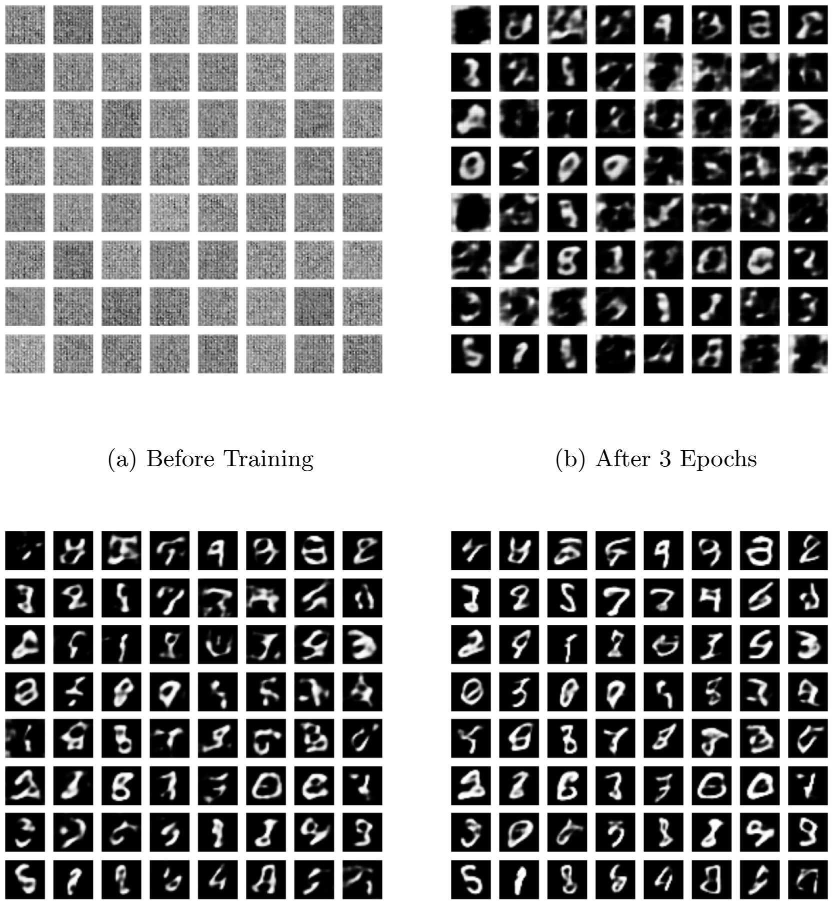
Architecture of the Encoder :
• 2 identical Conv2D layers with kernel of size 3 and Relu Activation, the first one with 64 filters and the second one with 32 filters
• 1 Flatten layer to prepare tensor for dense layer
• 1 Dense layer of dimension 2× latent-space dimension (=40)

Architecture of the Decoder :
- 1 Dense layer with Relu Activation
- 1 Reshape layer (inverse of Flatten)
- 3 identical Conv2DTranspose layers (inverse of Conv2D) with kernel of size 3 and Relu Activation, the first one with 64 filters, the second one with 32 filters, and the thrid one with 1 filter.

The .sample method passes random samples from $\mathcal{N}(0,1)$ into the decoder of the VAE.

Hyperparameters :
•train$_s ize = 6000$
$batch_s ize = 128$
$test_s ize = 10000$
$epoch = 80$

Question 10.


(a) Before Training (b) After 3 Epochs

(c) After 10 Epochs (d) After 80 Epochs

Figure 1: Samples of generated digits from VAE throughout training. The random vector used for generation is kept constant to better see the improvement. After 80 epochs the edges are more continuous and some digits are identifiable.

2 Natural Language Processing Part

2.1 RNNs

Question 12.
The motivation for Attention is for the decoder to focus on different parts of the source tokens. The attention weights are somewhat aligned between source and target words, this means that at each step of translation the decoder looks at the previous, following or corresponding word from the source vector.
The pytorch tutorial also shows that the attention is shifted by one word between the input and the output translation. I believe this is due to the token ¡SOS¿ from witch the decoder builds the first word of the output. Since we have an ¡SOS¿ in the output but not in the input the attention is shifted by one.

When we remove the ¡EOS¿ the network does not understand when the sentence is over and thus either fails to translate the entire sequence, or translates padding with dots.

Question 13.
We need attention because the encoding step compresses all the meaning of the sentence into a fixed length vector. However at each step of the translation different parts of the sequence can be more useful than others. The attention thus enables the decoder to focus on specific parts of the input. Without the attention layer the network learns much more slowly and we would need a lot more data to achieve the same result as with the attention.

Question 14.
Teacher forcing is a method for training recurrent neural network models more quickly and efficiently, that uses the ground truth from a prior time step as input to the current step. The idea behind it is that at the beginning of the training, the network doesn't know anything (random weights). Hence, to help it get started with the generation of the first few tokens, we give the encoder the first tokens of the correct target sequence.

Teacher forcing allows for faster convergence but presents instability risks.

2.2 Transformers

Question 15.

Question 16.
According to the study of Ruibin Xiong et al. 2020, (On Layer Normalization in the Transformer Architecture), putting the normalisation layer before the self-attention layer (architecture of Eq 22., called Pre-LN Transformer in the article), allows to remove the learning rate warm-up stage that usually prevent the unstability of the gradient descent at the beginning of the training of Post-LN transformers (Eq 21 architecture). Eq 22 allows for well-behaved gradients at the initialization and thus faster convergence and largely reduced training-time.