HW3 – Solutions

Thomas Marchioro

April 2023

Exercise 1

(a) In order to derive the loss function for a conditional variational autoencoder (cVAE), we need to compute the ELBO for the conditional log-likelihood $\log p(x|y)$, analogously to the non-conditional case:

$$\log p(x|y) = \log \left[\int_Z p(x,z|y) \frac{q(z|x,y)}{q(z|x,y)} dz \right] \tag{1}$$

$$= \log \left[\int_Z q(z|x,y) \frac{p(x,z|y)}{q(z|x,y)} dz \right] \tag{2}$$

$$\overset{\text{Jensen}}{\geq} \int_{\mathcal{Z}} q(z|x,y) \log \frac{p(x,z|y)}{q(z|x,y)} dz \tag{3}$$

$$= \int_{\mathcal{Z}} q(z|x,y) \log \frac{p(x|z,y)p(z|y)}{q(z|x,y)} dz \tag{4}$$

$$= \int_{\mathcal{Z}} q(z|x,y) \left(\log p(x|z,y) - \log \frac{q(z|x,y)}{p(z|y)} \right) dz \tag{5}$$

$$= \underbrace{\int_{\mathcal{Z}} q(z|x,y) \log p(x|z,y) dz}_{-\mathcal{L}_{\text{Rec}}} - \underbrace{\int_{\mathcal{Z}} q(z|x,y) \log \frac{q(z|x,y)}{p(z|y)} dz}_{\mathcal{L}_{\text{KLD}} = D_{\text{KL}}(q(z|x,y) \| p(z|y))} . \tag{6}$$

The formulation is identical to standard VAE but with an additional conditional variable y. In the case of MNIST, y represents the digits class. Implementation-wise, this means that both the encoder (which determines $q(z|x,y)$) and the decoder (which determines $p(x|z,y)$) should receive the condition as input. Also the latent variable distribution $p(z|y)$ depends on y, but in practice we always choose it to be $\mathcal{N}(0, I)$ for all values of y.

(b) The cVAE model can be easily obtained by slightly modifying the linear architecture of the non-conditional VAE from tutorial 6[1]. The encoder network should receive as input the image x and the one-hot encoding y of the desired label. Likewise, the decoder network should receive the concatenation of the latent variable z and the one-hot encoded class y. Therefore, the size of the input layer of both networks should be increased by 10 (the number of classes). Once this is done, the loss function can be kept unchanged. One epoch of training is all that is needed to generate digits of reasonable quality.

(c) The results obtained by the cVAE after one epoch of training are shown in figure 1.

[1] `https://github.com/thomasmarchioro3/hy673_tutorial06`

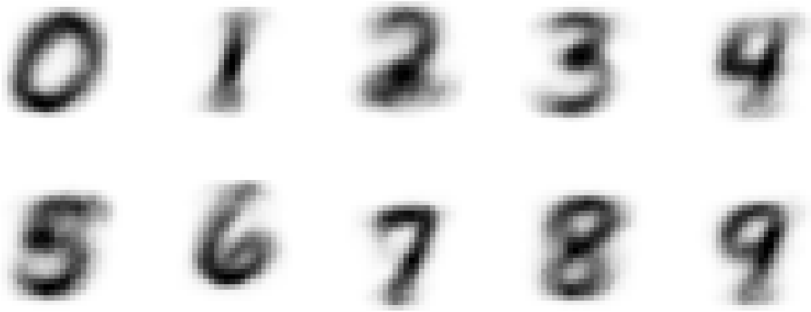

Figure 1: Digits generated by cVAE after one epoch of training.

Exercise 2

(a) The cVAE used in this exercise could be obtained by changing the architecture implemented in the previous exercise. Additionally, one should notice that the curves are not bounded in $[0, 1]$, so binary cross-entropy should be replaced by the mean squared error (or another suitable distance measure) in the reconstruction loss.

(b) Figure 2 shows examples of time series generated by the cVAE for $p = 0.01$ and $p = 5.01$.

Exercise 3

(a) Training an energy-based model with noise contrasive estimation (NCE) means minimizing the loss function

$$\mathcal{L}_{\theta,Z} = \frac{1}{n}\sum_{i=1}^{n}\left[f_\theta(x_i) + \mathrm{LSE}(f_\theta(x_i), \log Z + \log q(x_i)) + f_\theta(y_i) + \mathrm{LSE}(f_\theta(y_i), \log Z + \log q(y_i))\right] \quad (7)$$

w.r.t. the model parameters θ and the normalization constant Z (in practice, we directly learn $\log Z$). The distribution q is set to be $\mathcal{N}(0, \sigma_r^2 I)$.

The architecture and hyperparameters used to train the model are detailed in figure 3. Since the datas is low-dimensional, a simple multilayer perceptron with 4 fully-connected layers and leaky ReLU activation function is sufficient to learn the log-energy function f_θ. The normalization constant is initialized as $\log(Z) = 1$.

The model is trained for 40 epochs with batch size 128 and learning rate 10^{-3}.

(b) Using $\sigma_r = 3$ for the standard deviation of the initial noise, the generated samples obtained via Langevin sampling with $T = 1000$ do not produce the complete Swiss roll, as shown in figure 4 (on the right). This is expected since Langevin sampling iteratively brings noise observations towards one of the closest high-energy point. The noise samples (shown in figure 4, on the left) are within a distance of 10 from the origin, due to the small standard deviation. Therefore, in order to cover the whole Swiss roll, the standard deviation must be set higher.

After repeating the training and inference process with $\sigma_r = 6$, Langevin sampling actually allows to generate the complete Swiss roll, as shown in figure 5. With $T = 1000$, points generated with step size $\epsilon = 0.1$ and $\epsilon = 0.01$ mostly follow the Swiss roll distribution, with the exception of few outliers for the latter. For $\epsilon = 0.001$, instead, many points have not yet reached the high-energy region, which is expected since a small step size requires more steps to converge.

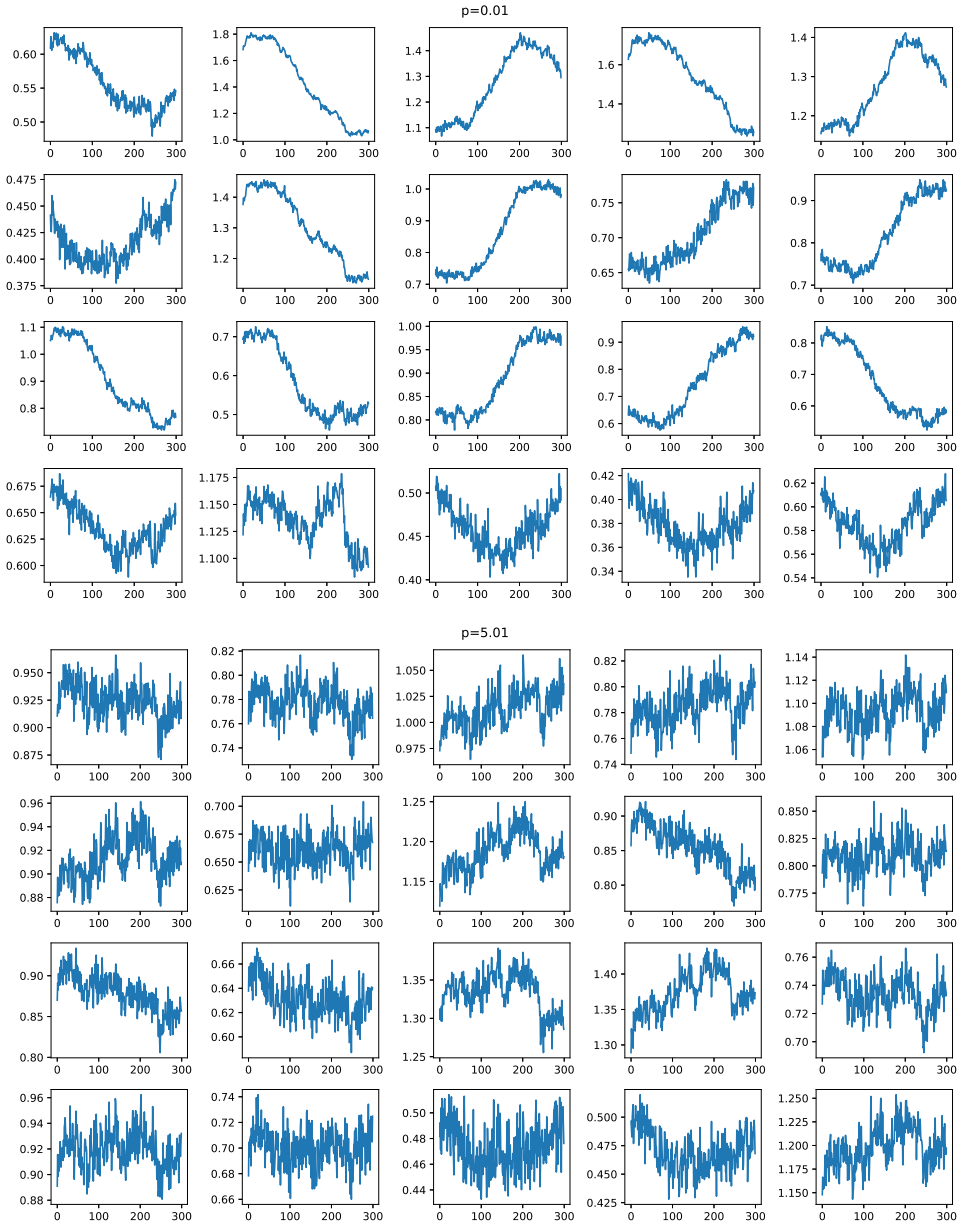

Figure 2: Sequences generated by the cVAE for $p = 0.01$ and $p = 5.01$.

Linear(2, 128)
Leaky ReLU(0.2)
Linear(128, 128)
Leaky ReLU(0.2)
Linear(128, 128)
Leaky ReLU(0.2)
Linear(128, 1)

Epochs	40
Batch size	128
Learning rate η	10^{-3}
Langevin step size ϵ_{train}	0.01

Figure 3: Architecture (to read from top to bottom) and hyperparameters used to train the energy-based model.

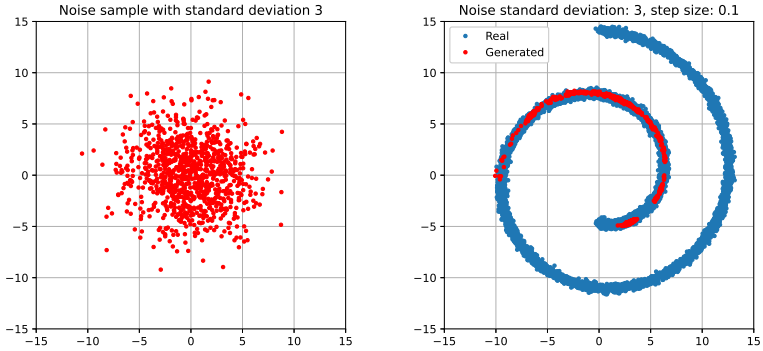

Figure 4: Samples generated starting from $x_0 \sim \mathcal{N}(0, 3^2 I)$.

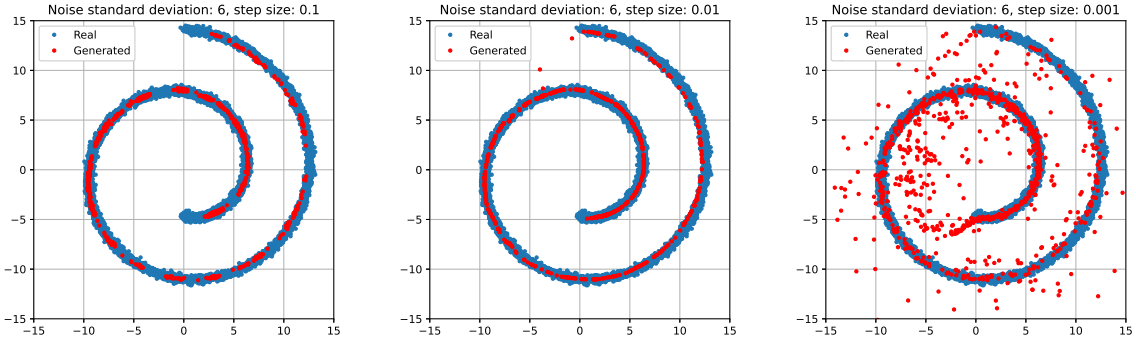

Figure 5: Samples generated starting from $x_0 \sim \mathcal{N}(0, 6^2 I)$ with step size $\epsilon = 0.1, 0.01, 0.001$ (from left to right).