2AMU20 Generative Models

Homework Assignment 3

Release date: Friday, 4th June 2021
Due date: Friday, 18th June 2021, 23:59

Solve the tasks below and describe your solution in a report in pdf format. Make sure that your report is written in a clean and accessible manner. Answer all questions asked and report the results. Please also explain your reasoning and intermediate steps. For tasks which require you to code, submit all the code you used to solve the task. Put the code for each task in a separate sub-folder called `task_x` where `x` is the corresponding task number, zip all the code into one file and submit it additionally to you report on Canvas. You can solve the assignment sheet in teams of 2 students. Please state the names and student numbers of both students in your pdf report.

Task 1 — Implement and Train a VAE [30 points]

Implement a VAE using (de)convolutional neural networks with `pytorch`.[1] Construct your model so that its latent space $\mathcal{Z}$ is K-dimensional, with $K > 1$, and such that $p(\mathbf{X} \mid \mathbf{z})$ is (multivariate) Gaussian with independent components, i.e. such that

$$p(\mathbf{X} \mid \mathbf{z}) = \prod_{d=1}^{D} p(X_d \mid \mathbf{z}) = \prod_{d=1}^{D} \mathcal{N}(X_d \mid \mu_d(\mathbf{z}), \sigma_d^2(\mathbf{z})) \quad \text{for all } \mathbf{z} \in \mathcal{Z},$$

where $D \in \mathbb{N}$ is the number of features in the data, where $\mathbf{X} = (X_1, \ldots, X_D)$, and where $\boldsymbol{\mu}(\mathbf{z}) = (\mu_1(\mathbf{z}), \ldots, \mu_D(\mathbf{z}))$ and $\boldsymbol{\sigma}^2(\mathbf{z}) = (\sigma_1^2(\mathbf{z}), \ldots, \sigma_D^2(\mathbf{z}))$ are the outputs of the *decoder* network of the VAE, when evaluated in $\mathbf{z}$.

Make sure your code can load and use the MNIST (digits) dataset; verify that it contains 60.000 training images and 10.000 testing images. Normalize the pixel values are represented as numerical values in the range $[0, 1]$ (by dividing by 255). Split the training data such that you can use the first 50.000 images for training, and the last 10.000 images for validation.

Implement a routine to train your model on the 50.000 training images by maximising the (estimated) ELBO, using `pytorch`'s autodiff functionality and build-in optimisers (e.g. Adam). Use a batch size of your choice. Use your code to answer the following questions:

[1] You are free to use another library of your choice that provides auto-differentiation and optimisation functionality, such as Tensorflow or Jax; in that case you may read any instance of "`pytorch`" in this assignment as meaning your choice of library.

Task 1a — Model Setup [5 points] In your report, briefly describe the network architecture(s) that you used in your model. Name the library that you used in your implementation (just mention `pytorch` if you decide to use that), and describe the optimisation routine (e.g. Adam) along with any training hyperparameters that differ from the default settings.

Task 1b — Model Training [10 points] Provide an explicit expression for $\log p(\mathbf{x} \mid \mathbf{z})$—the (conditional) log-likelihood of your model (i.e. given $\mathbf{z}$) with respect to a single data-point $\mathbf{x}$—*using the notation from this assignment as much as possible.*[2]

Provide a graph that shows how the ELBO (computed over the 50.000 training images) increases with the number of training epochs. At each training epoch, also compute the ELBO on the *validation* part of the data, and also show how this quantity changes with the number of training epochs.

Important: Make absolutely sure that your code does not keep track of gradients when computing the ELBO on the validation data, since otherwise it will be incorporated in training; this is explicitly not the intention. In `pytorch`, you can use

```
with torch.no_grad():
    ... # your evaluation code
```

to deactivate gradient tracking.

Task 1c — Stopping Criterion [5 points] Define a reasonable stopping criterion to terminate the training routine, and implement it in your code. Briefly describe and motivate your choice in the report.

Hint: To prevent having to constantly re-train your model, and therefore waste time, it may be a good idea to implement some functionality to save and load your model once it has finished training; `pytorch` provides functionality for this. You do not need to report on this, it is merely for your own convenience.

Task 1d — Model Testing [10 points] Use your trained model to *reconstruct* images from the *testing* part of the dataset, which if you did everything correctly, should not have been used in the training of your model. You obtain these reconstructions as follows:

1. Pass a testing image $\mathbf{x}$ through your encoder to obtain (the parameters of) the distribution $q(\mathbf{Z} \mid \mathbf{x})$;
2. Draw a sample $\mathbf{z}$ from $q(\mathbf{Z} \mid \mathbf{x})$;
3. Pass the sample $\mathbf{z}$ through the decoder to obtain (the parameters of) the distribution $p(\mathbf{X} \mid \mathbf{z})$;
4. Sample an image $\mathbf{x}'$ from $p(\mathbf{X} \mid \mathbf{z})$.

[2]That is, use $\boldsymbol{\mu}(\mathbf{z})$ and $\boldsymbol{\sigma}^2(\mathbf{z})$; *please* do not try to expand these into expressions for the neural networks...

In your report, include a graphic showing the first 32 images from the testing data, side-by-side with their respective reconstructions. Please represent this graphic using an 8-by-8 grid that fits on (at most) a single page of your report.

Moreover, generate some completely new images with your model, by sampling $\mathbf{z}$ from the prior $p(\mathbf{Z})$, passing this sample through the decoder, and sampling $\mathbf{x}'$ from $p(\mathbf{X} \mid \mathbf{z})$. Include some of these samples in your report, but don't overdo it; display them in a grid that fits on (at most) a single page of your report.

Task 2 — Choice of Conditional Data Distributions [25 points]

We will now (re)consider the choice of distributions with which we constructed $p(\mathbf{X} \mid \mathbf{z})$.

Task 2a — Reflection [5 points] Do you think that the choice of Gaussian distribution(s) was appropriate for the model $p(\mathbf{X} \mid \mathbf{z})$ that you constructed for **Task 1**? Consider both practical and theoretical aspects, and motivate your answer.

Task 2b — Beta Distributions [5 points] Change your model description so that $p(\mathbf{X} \mid \mathbf{z})$ is an (independent) product of Beta distributions,[3] i.e. such that

$$p(\mathbf{X} \mid \mathbf{z}) = \prod_{d=1}^{D} p(X_d \mid \mathbf{z}) = \prod_{d=1}^{D} \mathrm{Beta}(X_d \mid \alpha_d(\mathbf{z}), \beta_d(\mathbf{z})) \quad \text{for all } \mathbf{z} \in \mathcal{Z},$$

where $\alpha_d(\mathbf{z})$ and $\beta_d(\mathbf{z})$ are the parameters of the Beta distribution for the d-th feature of the data, as generated by the *decoder* network when evaluated in $\mathbf{z}$.

Provide an explicit expression for $\log p(\mathbf{x} \mid \mathbf{z})$—the (conditional) log-likelihood of your model (i.e. given $\mathbf{z}$) with respect to a single data-point $\mathbf{x}$—*using the notation from this assignment as much as possible.* Modify the code that you wrote for **Task 1** to represent these changes; mention and motivate any changes you make to the network architecture(s) in the encoder and decoder modules.

Train your model and and provide graphs of the ELBO as in **Task 1b**, as well as reconstructions and samples as in **Task 1d**. You may use the stopping criterion that you devised for **Task 1c**; if you change anything to this, make sure to mention and motivate this in your report.

Hint: Depending on your implementation, it can happen that your code becomes numerically unstable because it attempts to evaluate $\log(\textit{term})$, where *term* happens to be 0. In particular, this might happen for pixel values which are exactly 0 or 1. You can avoid this by adding a small constant $\epsilon > 0$ to terms which might evaluate to 0 and to which the log is applied.

Hint: You might use `torch.lgamma`, which is a numerical stable implementation of $\log(\Gamma)$, where Γ is the Gamma function.

[3] Please consult a statistics book to look up the Beta distribution, or search it on the internet.

Task 2c — Categorical Distributions [5 points] Write a routine that *discretises* the pixel values of the MNIST dataset into $k > 1$ *bins*; you are free to choose k (mention and motivate your choice in the report), but be wary of using too many.

Now let $\mathbf{X}$ denote the random variables over these discretised datapoints, and change your model description so that $p(\mathbf{X} \mid \mathbf{z})$ is an (independent) product of Categorical distributions over k outcomes, i.e. such that

$$p(\mathbf{X} \mid \mathbf{z}) = \prod_{d=1}^{D} p(X_d \mid \mathbf{z}) = \prod_{d=1}^{D} \mathrm{Cat}(X_d \mid \boldsymbol{\pi}_d(\mathbf{z})) \quad \text{for all } \mathbf{z} \in \mathcal{Z},$$

where $\boldsymbol{\pi}_d(\mathbf{z})$ is a vector containing the parameters of the Categorical distribution over k outcomes, for the d-th feature of the data, as generated by the *decoder* network when evaluated in $\mathbf{z}$.

Provide an explicit expression for $\log p(\mathbf{x} \mid \mathbf{z})$—the (conditional) log-likelihood of your model (i.e. given $\mathbf{z}$) with respect to a single data-point $\mathbf{x}$—*using the notation from this assignment as much as possible.* Modify the code that you wrote for **Task 1** to represent these changes; mention and motivate any changes you make to the network architecture(s) in the encoder and decoder modules.

Train your model and and provide graphs of the ELBO as in **Task 1b**, as well as reconstructions and samples as in **Task 1d**. You may use the stopping criterion that you devised for **Task 1c**; if you change anything to this, make sure to mention and motivate this in your report.

Hint: When attempting to reconstruct and sample images from this model, you will need some way to translate bin-assignments back into numerical values in the $[0, 1]$ range; feel free to use whatever method you think is appropriate (e.g. the midpoints), but make sure to document this in your report.

Task 2d — Bernoulli Distributions [5 points] Let us now change the model description to a distribution $p(\mathbf{B} \mid \mathbf{z})$ over the space $\{0, 1\}^D$ of binary images, as an (independent) product of Bernoulli distributions

$$p(\mathbf{B} \mid \mathbf{z}) = \prod_{d=1}^{D} p(B_d \mid \mathbf{z}) = \prod_{d=1}^{D} \mathrm{Bern}(B_d \mid \pi_d(\mathbf{z})) \quad \text{for all } \mathbf{z} \in \mathcal{Z}, \tag{1}$$

where $\pi_d(\mathbf{z})$ is the parameter of the Bernoulli distribution for the d-th feature, as generated by the *decoder* network when evaluated in $\mathbf{z}$. Note that your model now describes binary (i.e. black-and-white) pixel data, which is not representative of the MNIST data, as those are grayscale images.

In order to have this model make sense, we will therefore need to modify the interpretation of the data. To this end, again note that the MNIST pixel intensity values lie in the range $[0, 1]$. Hence, we can interpret an image from this dataset as *itself* representing a probability distribution: the d-th feature (i.e. pixel) x_d of an image $\mathbf{x}$ can be interpreted as the probability that the d-th pixel should be "on" (i.e. white) or "off" (i.e. black). If we assume that all these per-pixel probabilities are independent under the distribution represented by the grayscale image, this yields a distribution $p(\mathbf{B} \,|\, \mathbf{x})$ over the space $\{0,1\}^D$ of binary images that can be written as

$$p(\mathbf{B} \,|\, \mathbf{x}) = \prod_{d=1}^{D} \mathrm{Bern}(B_d \,|\, x_d)\,. \tag{2}$$

Now, rather than the usual expression for the ELBO, let your training method instead maximise

$$-H\big(p(\mathbf{B} \,|\, \mathbf{x}), p(\mathbf{B} \,|\, \mathbf{z})\big) - \mathrm{KL}\big(q(\mathbf{Z} \,|\, \mathbf{x}) \,||\, p(\mathbf{Z})\big)\,,$$

where the first summand is the (negative) cross-entropy between $p(\mathbf{B} \,|\, \mathbf{x})$ and $p(\mathbf{B} \,|\, \mathbf{z})$, given by

$$\begin{aligned} -H\big(p(\mathbf{B} \,|\, \mathbf{z}), p(\mathbf{B} \,|\, \mathbf{x})\big) &= \mathbb{E}_{p(\mathbf{B} \,|\, \mathbf{x})}[\log p(\mathbf{b} \,|\, \mathbf{z})] \\ &= \sum_{d=1}^{D} x_d \log \pi_d(\mathbf{z}) + (1 - x_d) \log(1 - \pi_d(\mathbf{z}))\,, \end{aligned}$$

with $\pi_d(\mathbf{z})$ as in Equation (1).

Modify the code that you wrote for **Task 1** to represent these changes; mention and motivate any changes you make to the network architecture(s) in the encoder and decoder modules.

Train your model and and provide graphs of the modified objective function as in **Task 1b**, as well as reconstructions and samples as in **Task 1d**. You may use the stopping criterion that you devised for **Task 1c**; if you change anything to this, make sure to mention and motivate this in your report.

Hint: For the image reconstruction and sampling, you may get better results by instead using the posterior means $\mathbb{E}_{p(\mathbf{B} \,|\, \mathbf{z})}[\mathbf{B}]$, rather than samples $\mathbf{b} \sim p(\mathbf{B} \,|\, \mathbf{z})$. If you decide to use these, make sure to mention this in your report.

Task 2e — Further Reflection [5 points] Which of the model and data combinations that you considered so far—Gaussian distributions with $[0,1]$ -valued data, Beta distributions with $[0,1]$-valued data, Categorical distributions with discretised (binned) data, and Bernoulli distributions with re-interpreted data—do you feel is most appropriate for a generative model that attempts to approximate the (unknown) distribution $p_*(\mathbf{X})$ from which the MNIST dataset was sampled? Is there another data or model description that hasn't been covered by these questions and that you feel is even more appropriate for this task? Consider both practical and theoretical aspects, and motivate your answer.

Task 3 — Investigate the Latent Space [20 points]

For this task, we will investigate the structure of the latent variables $\mathbf{Z}$, and see how it captures structure that was implicitly present in the data. You may use whichever model-data combination you preferred from **Task 1** and **Task 2**—just be sure to mention your choice in the report.

Task 3a — Visualizing the Latent Space [5 points] Modify your code so that the latent space is two-dimensional, i.e. such that $\mathcal{Z} = \mathbb{R}^2$, and train your model to completion using the appropriate objective function and stopping criterion.

Evaluate the first 1000 datapoints $\mathbf{x}_i$, $i = 1, \ldots, 1000$, of the *testing* data using the *trained encoder* network. Visualise the network outputs $\boldsymbol{\mu}(\mathbf{x}_i)$ on a 2-dimensional plot, color-coding the datapoints based on the class-labels y_i, i.e. such that you use a distinct color for every type of digit. Include this graphic in your report.

Task 3b — Visualizing the Latent Space, using PCA [10 points] Modify your code so that the latent space is K-dimensional, i.e. such that $\mathcal{Z} = \mathbb{R}^K$, with $K \geq 10$. Train your model to completion using the appropriate objective function and stopping criterion.

Evaluate the first 1000 datapoints $\mathbf{x}_i$, $i = 1, \ldots, 1000$, of the *testing* data using the *trained encoder* network. Store the network outputs $\boldsymbol{\mu}(\mathbf{x}_i)$ in a matrix $\mathbf{U}$ with k columns and 1000 rows.

Perform a *Principal Component Analysis* on $\mathbf{U}$, and create a 2D scatterplot of the rows of $\mathbf{U}$ projected onto the first two principal components. As in **Task 3a**, color-code these datapoints based on the class-labels y_i. Include this graphic in your report.

Hint: You do not need to write the code for the PCA yourself; feel free to use whichever library or python functionality that you prefer.

Task 3c — Interpolation in the Latent Space [5 points] Rather than directly *reconstruct* images, one can perform operations on the encoded datapoints by working in the latent space, and then using the decoder part of the model to translate these operations to the data-domain.

Make sure you have your favorite variant of the model (Gaussian, Beta, Bernoulli, Categorical) trained to completion. Now perform the following operations:

1. Take two data points $\mathbf{x}$ and $\mathbf{x}'$ from the *testing* data, making sure that these images have different class labels y and y';
2. Obtain samples $\mathbf{z}$ and $\mathbf{z}'$ from, respectively, $q(\mathbf{Z} \,|\, \mathbf{x})$ and $q(\mathbf{Z} \,|\, \mathbf{x}')$;
3. With $\lambda \in [0, 1]$, compute the linear interpolation $\mathbf{z}_\lambda = \lambda \mathbf{z} + (1 - \lambda)\mathbf{z}'$;
4. Obtain a sample $\mathbf{x}_\lambda$ from $p(\mathbf{X} \,|\, \mathbf{z}_\lambda)$.

In your report, include a figure containing a grid such that, on each row, the leftmost entry is $\mathbf{x}$, the rightmost entry is $\mathbf{x}'$, and the $k > 1$ entries in between are given by $\mathbf{x}_{\lambda_i}$, with λ_i on a uniform k-partition of $[0, 1]$.

You are free to choose k and the number of rows of this figure, but please don't overdo it; make sure that it fits on (at most) a single page of your report.

Task 4 — Inference Without the Encoder [25 points]

For this task, we will attempt to estimate the distribution $p(\mathbf{Z} \,|\, \mathbf{x})$ *without* using the encoder, and see why this is useful. You may use whichever model-data combination you preferred from **Task 1** and **Task 2**—just be sure to mention your choice in the report.

Task 4a — Reconstruction without the Encoder [10 points] Consider a single datapoint $\mathbf{x}$ from the *testing* data. Let $q(\mathbf{Z} \,|\, \Psi)$ be diagonal Gaussian with independent components over $\mathcal{Z}$, with parameter vector Ψ – i.e., Ψ contains the mean vector and the diagonal of the covariance matrix.

Maximise the ELBO for $\mathbf{x}$ with respect to Ψ, i.e. compute

$$\Psi_* = \arg\max_{\Psi} \mathbb{E}_{q(\mathbf{Z} \,|\, \Psi)}[\log p(\mathbf{x} \,|\, \mathbf{z})] - \mathrm{KL}\big(q(\mathbf{Z} \,|\, \Psi) \,||\, p(\mathbf{Z})\big) \,,$$

using stochastic gradient descent, together with the 1-sample Monte Carlo estimate and the "reparameterisation trick" to approximately evaluate $\mathbb{E}_{q(\mathbf{Z} \,|\, \Psi)}[\log p(\mathbf{x} \,|\, \mathbf{z})]$.

After convergence—use and document your choice of stopping criterion—generate a sample $\mathbf{z}$ from $q(\mathbf{Z} \,|\, \Psi_*)$, and reconstruct $\mathbf{x}$ by sampling $\mathbf{x}'$ from $p(\mathbf{X} \,|\, \mathbf{z})$.

Repeat the above procedure for several different datapoints from the *testing* data, making sure to optimise Ψ separately each time. In your report, include a figure containing a 3-column grid where on each row, you display the original datapoint $\mathbf{x}$, the reconstructed datapoint $\mathbf{x}'$ that you obtained in the manner described above, and a reconstruction $\mathbf{x}''$ that you obtained as in **Task 1d**.

Task 4b — Image Completion [15 points] We will finally use the above techniques to perform *image completion.* To this end, consider a single datapoint $\mathbf{x}$ from the *testing* data, and consider the partition of the variables $\mathbf{X}$ into parts $\mathbf{X}_L$ and $\mathbf{X}_R$, representing respectively the left- and right half of the image. Our goal will be to construct a complete image $\mathbf{x}'$ given only the left half $\mathbf{x}_L$; that is, we want to sample $\mathbf{x}'_R$.

Modify the code that you wrote for **Task 4a** to instead maximise

$$\Psi_* = \arg\max_{\Psi} \mathbb{E}_{q(\mathbf{Z} \,|\, \Psi)}[\log p(\mathbf{x}_L \,|\, \mathbf{z})] - \mathrm{KL}\big(q(\mathbf{Z} \,|\, \Psi) \,||\, p(\mathbf{Z})\big) \,,$$

using the same details as before. Note that you can evaluate $p(\mathbf{x}_L \,|\, \mathbf{z})$ by simply evaluating the model in *only* the variables of the left half of the image, and ignoring all variables in the right half (i.e. simply discard the mean and variance for the missing pixels).

Once the optimisation of Ψ has converged—use and document your choice of stopping criterion—generate a sample $\mathbf{z}$ from $q(\mathbf{Z} \,|\, \Psi_*)$, and reconstruct the right half of the image by sampling $\mathbf{x}'_R$ from $p(\mathbf{X}_R \,|\, \mathbf{z})$.

Repeat the above procedure for several different datapoints from the *testing* data, making sure to optimise Ψ separately each time. In your report, include a figure containing a 3-column grid where on each row, you display the original datapoint $\mathbf{x}$, an image whose left side consists of $\mathbf{x}_L$ and whose right side contains only black pixels, and an image whose left side consists of $\mathbf{x}_L$ and whose right side consists of $\mathbf{x}'_R$.

Task 5 — Peer Review [0 points]

Each group member must write a single paragraph outlining their opinion on the work distribution within the group. Did every group member contribute equally? Did you split up tasks in a fair manner, or jointly worked through the exercises? Do you think that some members of your group deserve a different grade from others?