Exercises Week 1

Advanced Machine Learning (02460)
Technical University of Denmark
Jes Frellsen

February 2024 (Version 1.1)

This week's exercises are on **deep latent variable models** and consist of theoretical exercises that will prepare you for the written exam and a programming exercise that will prepare you for the project(s).

1 Theoretical exercises

Exercise 1.1 Consider probabilistic principal component analysis (PPCA), as described in section 4.2 of the textbook (Tomczak 2022). The PPCA model has three parameters σ^2, $\mathbf{b}$ and $\mathbf{W}$. Given a data set $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, show that the maximum likelihood estimate of $\mathbf{b}$ is the mean of the data set, i.e., $\hat{\mathbf{b}} = \bar{\mathbf{x}}$. Do this by following the steps:

1. Based on equation (4.6), write the log-likelihood function, $\ell(\sigma^2, \mathbf{b}, \mathbf{W}) = \ln p(\mathcal{D} \mid \sigma^2, \mathbf{b}, \mathbf{W}) = \sum_{n=1}^{N} \ln p(\mathbf{x}_n \mid \sigma^2, \mathbf{b}, \mathbf{W})$.
2. Set the derivative of the log-likelihood with respect to $\mathbf{b}$ equal to zero.

Hint: For a symmetric matrix $\mathbf{W}$ and vectors $\mathbf{x}$ and $\mathbf{b}$, we have

$$\frac{\partial}{\partial \mathbf{b}} (\mathbf{x} - \mathbf{b})^\intercal \mathbf{W} (\mathbf{x} - \mathbf{b}) = -2\mathbf{W}(\mathbf{x} - \mathbf{b}), \tag{1}$$

see, e.g., equation (86) by Petersen and Pedersen (2012).

Exercise 1.2 As discussed in section 4.3.1 of the textbook (Tomczak 2022), the second term of the ELBO in equation (4.17), can be written as a KL-divergence, i.e.,

$$\begin{aligned} \text{ELBO}(\mathbf{x}) &= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}[\ln p(\mathbf{x} \mid \mathbf{z})] - \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}[\ln q_\phi(\mathbf{z} \mid \mathbf{x}) - \ln p(\mathbf{z})] && (2) \\ &= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}[\ln p(\mathbf{x} \mid \mathbf{z})] - \text{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z})] && (3) \end{aligned}$$

where the KL-divergence of density $f(\mathbf{x})$ from density $g(\mathbf{x})$ is defined as

$$\text{KL}[f \,\|\, g] = \int f(\mathbf{x}) \ln \frac{f(\mathbf{x})}{g(\mathbf{x})} \, \mathrm{d}\mathbf{x}. \tag{4}$$

When training VAEs, we often use that this KL-divergence has a closed for expression for many distributions, e.g., when both the approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ and the prior $p(\mathbf{z})$ are multivariate Gaussian.

1. Show that the two expressions for the ELBO in equations (2) and (3) are equivalent, i.e., that $\mathbb{E}_{z\sim q_\phi(\mathbf{z}|\mathbf{x})}[\ln q_\phi(\mathbf{z} \mid \mathbf{x}) - \ln p(\mathbf{x})] = \mathrm{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z})]$.
2. Show that the KL divergence between two *univariate* Gaussian distributions has the following closed form

$$\mathrm{KL}[\mathcal{N}(\mu_1, \sigma_1) \,\|\, \mathcal{N}(\mu_2, \sigma_2)] = \ln \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}. \tag{5}$$

Remark: The above questions can easily be answered by ChatGPT or a Google search. However, we encourage you to do the derivations yourself, as these are insightful, and you will not have access to ChatGPT or Google during the exam.

Exercise 1.3 Consider the two-level hierarchical VAE, as described in section 4.5 of the textbook (Tomczak 2022). Starting from the standard ELBO, e.g., equation (4.17), show that the ELBO for the two-level VAE can be written as equation (4.82), i.e.,

$$\begin{aligned}\mathrm{ELBO}(\mathbf{x}) = \mathbb{E}_{(\mathbf{z}_1,\mathbf{z}_2)\sim Q(\mathbf{z}_1,\mathbf{z}_2)}\Big[&\ln p(\mathbf{x} \mid \mathbf{z}_1) \\ &- \mathrm{KL}[q(\mathbf{z}_1 \mid \mathbf{x}) \,\|\, p(\mathbf{z}_1 \mid \mathbf{z}_2)] - \mathrm{KL}[q(\mathbf{z}_2 \mid \mathbf{z}_1) \,\|\, p(\mathbf{z}_2)]\Big].\end{aligned} \tag{6}$$

2 Programming exercises

In this week's programming exercise, you will work with variational autoencoders (VAEs). We are going to consider a binarised version of the MNIST dataset, where pixels with values over 0.5 are set to 1 and pixels with values less than 0.5 are set to 0.

The provided file `vae_bernoulli.py` contains a modular and simple implementation of a VAE with

- a Gaussian prior, $p(\mathbf{z})$,
- a product of Bernoulli likelihood, $p(\mathbf{x}|\mathbf{z})$,
- a fully connected encoder and decoder network.

The implementation makes use of `torch.distributions` for the various distributions, which substantially simplifies the code.

It also implements a command line parser, so you can train a VAE on the CPU with a latent dimension of $M = 10$ for 5 epochs and with a batch size of 128 and save the final model to the file `model.pt` using the command

```
python3 vae_bernoulli.py train --device cpu --latent-dim 10 \
        --epochs 5 --batch-size 128 --model model.pt
```

After the model has been trained, you can sample from the trained model (saved in `model.pt`) and save the samples in `samples.png`, using

```
python3 vae_bernoulli.py sample --device cpu --latent-dim 10 \
        --model model.pt --samples samples.png
```

In the following exercises, you will modify this VAE implementation.

Exercise 1.4 In this first exercise, you should just inspect the code in `vae_bernoulli.py`. Answer the following questions:

- How is the reparametrisation trick handled in the code?
- Consider the implementation of the ELBO. What is the dimension of `self.decoder(z).log_prob(x)` and of `td.kl_divergence(q, self.prior.distribuion)`?
- The implementation of the prior, encoder and decoder classes all make use of `td.Independent`. What does this do?
- What is the purpose using the function `torch.chunk` in `GaussianEncoder.forward`?

Exercise 1.5 Add the following functionally to the implementation (`vae_bernoulli.py`) of the VAE with Bernoulli output distributions:

- Evaluate the ELBO on the binarised MNIST test set.
- Plot samples from the approximate posterior and colour them by their correct class label. Implement it such that you, for latent dimensions larger than two, $M > 2$, do PCA and project the sample onto the first two principal components (e.g., using scikit-learn).

Exercise 1.6 Extend the VAE with Bernoulli output distributions (`vae_bernoulli.py`) to use a mixture of Gaussian prior (MoG). We recommend using the `MixtureSameFamily` class from `torch.distributions`, but you are also welcome to implement it from scratch. For your implementation of the VAE with the MoG prior:

- Evaluate the test set ELBO. Do you see better performance?
- Plot the samples from the approximate posterior. How does it differ from the model with the Gaussian prior? Do you see better clustering?

Remark: You will need to change the formulation of the ELOB.

Exercise 1.7 So far, we have considered a binarised version of MNIST. We will now consider the pixel values in MNIST to be continuous and experiment with different output distributions. You can load the continuous MNIST training set using

```
datasets.MNIST('data/', train=True, download=True,
               transform=transforms.Compose([
                   transforms.ToTensor(),
                   transforms.Lambda(lambda x: x.squeeze())
                ]))
```

Implement a multivariate Gaussian output distribution. You should both experiment with learning the variance of each pixel and having a fixed variance for all pixels.

- How is the qualitative sample quality?
- Rathen then sampling from $p(\mathbf{x}|\mathbf{z})$, try to sample $\mathbf{z} \sim p(\mathbf{z})$ and then show the mean of the output distribution, $p(\mathbf{x}|\mathbf{z})$. Does the mean qualitatively look better?

Optional: You can also try the Continuous Bernoulli output distribution, see Loaiza-Ganem and Cunningham (2019) for more details.

Exercise 1.8 (*Optional*) Extend the VAE with Bernoulli output distributions (`vae_bernoulli.py`) to use a CNN based encoder and decoder. For your new implementation:

- When sampling from a trained model, do you see a qualitative improvement in sample quality?
- Evaluate the test set ELBO. Do you see better performance?

A Working with the code

We recommend working with the code locally in a code editor. To work with the code locally, you need to have Python 3.8 or later installed and PyTorch version 2.1.2. For details on installing PyTorch, see https://pytorch.org/get-started/locally/. We recommend that you use a modern editor, such as Visual Studio Code or PyCharm.

You are also welcome to paste the code a Google Colab notebook, but it is not something we directly support in this course.

A.1 DTU HPC Cluster

To get access to GPUs for the exercise, you can use the DTU HPC cluster, as you learned in the Deep Learning course (02456), where you were also given an comprehensive guide to the system. Otherwise, we refer to the HPC documentation, particularly

- Accessing Cluster
- Installing Python packages
- Using GPUs

References

Loaiza-Ganem, G and JP Cunningham (2019). "The continuous Bernoulli: fixing a pervasive error in variational autoencoders". In: *Advances in Neural Information Processing Systems*. Ed. by H Wallach, H Larochelle, A Beygelzimer, et al. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/f82798ec8909d23e55679ee26bb26437-Paper.pdf.

Petersen, KB and MS Pedersen (2012). *The Matrix Cookbook*. Version 20121115. URL: http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html.
Tomczak, JM (2022). *Deep Generative Modeling*. Springer.