Exercises Week 2

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

February 2024
(Version 1.2)

This week's exercises are on **normalizing flows** 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 2.1 Consider a linear transformation $f : \mathbb{R}^D \to \mathbb{R}^D$ given by $f(\mathbf{x}) = -\mathbf{x} + \mathbf{a}$ for some $\mathbf{a} \in \mathbb{R}^D$. Show that this transformation is *volume-preserving*, c.f., section 3.1.1 of the textbook (Tomczak 2022).

Exercise 2.2 Let $F(\mathbf{x}) = f_K \circ f_{K-1} \circ \cdots \circ f_2 \circ f_1(\mathbf{x})$, where $f_k : \mathbb{R}^D \to \mathbb{R}^D$ for $k = 1, \ldots, K$ are invertible functions. Show that the inverse of F is given by $F^{-1}(\mathbf{z}) = f_1^{-1} \circ f_2^{-1} \circ \cdots \circ f_{K-1}^{-1} \circ f_K^{-1}(\mathbf{z})$.

Hint: You can, e.g., show that $F^{-1} \circ F(\mathbf{x}) = \mathbf{x}$ and $F \circ F^{-1}(\mathbf{z}) = \mathbf{z}$, and you do not need to do a very formal proof.

Exercise 2.3 Consider the function $h = g \circ f$ that composed of the two functions $f : \mathbb{R}^D \to \mathbb{R}^D$ and $g : \mathbb{R}^D \to \mathbb{R}^D$. Show that $|\det \mathbf{J}_h| = \left|\det \mathbf{J}_g\right|\left|\det \mathbf{J}_f\right|$ by following the steps:

1. Show that $\mathbf{J}_h = \mathbf{J}_g \mathbf{J}_f$.
   (a) Use the chain rule to write down the expression of (i, j)'th entry $\mathbf{J}_h$, i.e., $\frac{\partial h_i}{\partial x_j}$.
   (b) Use the definition of matrix multiplication to write down (i, j)'th entry of $\mathbf{J}_g \mathbf{J}_f$.
2. Use that the determinant distributes over multiplication to arrive at the final results.

Hint: The chain rule for $\frac{\partial h_i}{\partial x_j}$ can be expressed as $\frac{\partial h_i}{\partial x_j} = \nabla g_i(f(\mathbf{x})) \cdot \frac{\partial f}{\partial x_j}$.

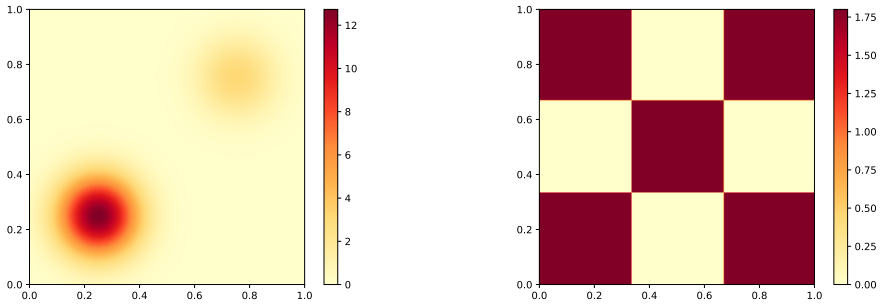

Figure 1: *Left:* The density of the TwoGaussians toy model, which is an uneven mixture of to Gaussian distributions. *Right:* The density of the Chequerboard toy model, which is an even mixture of five uniform densities.

2 Programming exercises

The main task in this week's programming exercise, is to implement the *masked coupling layer* from Real NVP (Dinh et al. 2017). We will start out with the two simple toy datasets shown in figure 1, then we will move on to learning a flow on MNIST, and finally use a flow as a prior in a VAE. This week, we have provided you with two files:

- `flow.py` contains a modular (incomplete) implementation of a normalizing flow.
- `ToyData.py` contains the code for generating data from the two toy models.

Masked coupling layer The advantage of using the masked coupling layer (Dinh et al. 2017) over a regular coupling layer, is that it allows for arbitrary partitioning of the variable in the coupling layer. Let $T : \mathbb{R}^D \to \mathbb{R}^D$ be the masked coupling transformation that maps $\mathbf{z}$ to $\mathbf{z}'$. Then let $\mathbf{b} \in \{0, 1\}^D$ denote a binary mask, where $b_i = 1$ means that z_i is left unchanged by the coupling layer and $b_i = 0$ means that z_i is transformed by the coupling layer. We can then express the masked coupling layer as the transformation

$$\mathbf{z}' = \mathbf{b} \odot \mathbf{z} + (\mathbf{1} - \mathbf{b}) \odot \Big(\mathbf{z} \odot \exp\big(s(\mathbf{b} \odot \mathbf{z})\big) + t(\mathbf{b} \odot \mathbf{z}) \Big) \tag{15}$$

where $\odot$ is the element wise product, $s : \mathbb{R}^D \to \mathbb{R}^D$ is the network that calculates the scaling and $t : \mathbb{R}^D \to \mathbb{R}^D$ is the network that calculates translation of the affine transformation. Note that the input to these networks are masked by multiplying $\mathbf{z}$ by the mask, and that $\mathbf{b} \odot \mathbf{z} = \mathbf{b} \odot \mathbf{z}'$.

The inverse of the transformation is given by an expression similar to the inverse of the regular couple layers, i.e.,

$$\mathbf{z} = \mathbf{b} \odot \mathbf{z}' + (\mathbf{1} - \mathbf{b}) \odot \Big(\big(\mathbf{z}' - t(\mathbf{b} \odot \mathbf{z}')\big) \odot \exp\big(-s(\mathbf{b} \odot \mathbf{z}')\big) \Big), \tag{16}$$

and the log determinant of the Jacobian is given by

$$\log\left|\det \mathbf{J}_T(z)\right| = \sum_{d=1}^{D}(1-b_i)s_i(\mathbf{b}\odot\mathbf{z}). \tag{17}$$

Exercise 2.4 In the provided code (`flow.py`), the class `MaskedCouplingLayer` implements the masked coupling layer from Real NVP (Dinh et al. 2017), but it does not implement the forward transformation, the inverse transformation and the corresponding calculations of the log determinant of the Jacobian.

In this exercise you should complete the following two functions such that:

- `MaskedCouplingLayer.forward(...)` returns $T(\mathbf{z})$ and $\log\left|\det \mathbf{J}_T(\mathbf{z})\right|$.
- `MaskedCouplingLayer.inverse(...)` returns $T^{-1}(\mathbf{z}')$ and $\log\left|\det \mathbf{J}_{T^{-1}}(\mathbf{z}')\right|$.

Use the TwoGaussians datasets (c.f., figure 1) for testing the model. Adjust the number of coupling layers and the architecture of the networks to get a good fit to the density (by qualitative assessment). Make sure to write your implementation such that it works on data of more than two dimensions.

Optional: Can you also fit a flow to the Chequerboard dataset? It is difficult to find an architecture that gives a good fit.

Exercise 2.5 Use the flow implementation from exercise 2.4 to train a flow on dequantized MNIST. You do not need to implement a dequantization layer, as you can perform this transformation on MNIST when you load it. The following code will load MNIST such that (1) each pixel value is transformed to $[0, 1]$, (2) each pixel value is dequantized, and (3) each picture is flatten to have dimension $28 \cdot 28 = 784$ such that you can use fully connected layers for the networks in the masked coupling layer.

```
datasets.MNIST('data/', train=True, download=True,
               transform=transforms.Compose([
                transforms.ToTensor(),
                transforms.Lambda(lambda x: x + torch.rand(x.shape)/255),
                transforms.Lambda(lambda x: x.flatten())
               ])
              )
```

For stability, you need to add the tanh activation function at the end of the scale network. Implement the following two masking strategies:

- Each layer uses a random initialized masking (note that the mask is saved in the `MaskedCouplingLayer` such that is not randomized at every call to the transformation).
- Use the chequerboard masking (see figure 2) from Real NPV and invert the mask at each layer.

Implement functionally for saving samples from the flow and qualitatively asses the sampling quality of MNIST.

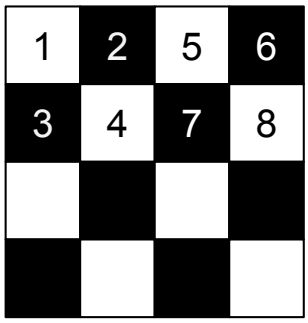

Figure 2: The chequerboard masking proposed and illustrated by Dinh et al. (2017), where pixies are masked out in a chequerboard pattern.

Exercise 2.6 Extend the VAE with Bernoulli output (`vae_bernoulli.py`) from week 1 to use a flow prior based on the model from exercise 2.4. For your implementation of the VAE with the flow prior:

- Evaluate the test set ELBO. Do you see better performance?
- Plot the samples from the approximate posterior and samples from the prior. Do the match well?

Hint: You can use the same the formulation of the ELOB as you used for the MoG prior.

References

Dinh, L, J Sohl-Dickstein, and S Bengio (2017). "Density estimation using Real NVP". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=HkpbnH9lx.

Tomczak, JM (2022). *Deep Generative Modeling*. Springer.