SS 2025
© Prof. Georgia Koppe

Course Coordinator: Andre Herz
andre.herz@iwr.uni-heidelberg.de

Machine Learning Essentials

Exercise Sheet 06

Due: 16.06.2025 11:15

This sheet covers convolutions, convolutional neural networks and their implementations via pytorch. The first exercise asks you to explicitly compute convolutions, both for the discrete and the continuous case. In the second exercise, a classifier is trained using the hand sign dataset, which consists of images of hand gestures. Finally, you will compare a (non-linear) convolutional autoencoder with a linear one to represent images in a lower dimensional latent space.

Regulations

Please submit your solutions via Moodle in teams of **3** students. The coding tasks must be completed using the template `MLE25_sheet06.ipynb`. Each submission must include **exactly** one file:

- Upload a `.pdf` file containing both your Jupyter notebook and solutions to analytical exercises.

The Jupyter notebook can be exported to pdf by selecting **File** → **Download as** → **pdf** in JupyterLab. If this method does not work, you may print the notebook as a pdf instead. Your analytical solutions can be either scanned handwritten solutions or created using LaTeX.

Exercise 1: Convolutions of Continuous and Discrete Variables

1. In probability theory and machine learning, we often deal with the sum of random variables. For instance, a signal might be corrupted by additive noise, or we might combine predictions from different uncertain sources. If X and Y are independent continuous random variables with pdf $f_X(x)$ and $f_Y(y)$ respectively, the PDF of their sum $Z = X + Y$, denoted $f_Z(z)$, can be found using the **convolution** of their individual PDFs:

$$f_Z(z) = (f_X * f_Y)(z) = \int_{-\infty}^{\infty} f_X(x) f_Y(z - x)\, dx$$

This integral essentially calculates the probability density for $Z = z$ by summing over all possible ways this sum can be achieved (i.e., all possible values of x such that $y = z - x$). The Gaussian distribution holds a special place in probability and ML due to its many convenient properties. One such property relates to its behaviour under convolution. Your task is to demonstrate this property by proving that the convolution of two univariate Gaussian PDFs results in another Gaussian PDF. Specifically, let $X \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Y \sim \mathcal{N}(\mu_2, \sigma_2^2)$ be two **independent** Gaussian random variables. Compute the PDF $f_Z(z)$ of their sum $Z = X + Y$ using the given equation. Show that the resulting $f_Z(z)$ is the PDF of a Gaussian distribution and determine its mean μ_Z and variance σ_Z^2 in terms of $\mu_1, \sigma_1^2, \mu_2, \sigma_2^2$. You may use the standard result that for constants a and $b^2 > 0$:

$$\int_{-\infty}^{\infty} \exp\left(-\frac{(x-a)^2}{2b^2}\right) dx = \sqrt{2\pi b^2}.$$

(This is equivalent to stating that the integral of a properly normalized Gaussian pdf over its domain is 1).

(4 pts.)

2. Consider a (grayscale) image of height H and width W. We now apply a square filter of shape $(K_H \times K_W)$ where $K_H < H$ ans $K_W < W$ to the image, that is, we compute the two-dimensional discrete convolution of the image and the filter. The image is first padded on all four sides with p pixels of some constant value. For both directions, a stride s is to be used. Derive equations for the dimensions H_{out} and W_{out} of the convoluted image.

(2 pts.)

Exercise 2: CNN Classifier

1. Complete the pytorch code to build the classifier of the given architecture.

(4 pts.)

2. Define a custom dataset and dataloader to loop over during training.

(1 pts.)

3. Implement the training function. Train classifiers with learning rates 10^{-1}, 10^{-3}, 10^{-4} and 10^{-5}. Visualize the train and validation loss as well as the train and validation accuracy over epochs for each model and comment on it.

(4 pts.)

4. Apply the best model from Task 3 to the test set and compute the test accuracy. How does it differ from the validation error?

(1 pts.)

Exercise 3: CNN Autoencoder

Autoencoders (AEs) are types of neural networks that learn to represent compressed data in a latent space. They consist of an encoder, which gradually reduces the feature dimension m to the latent dimension $d < m$, and a decoder, which reconstructs the original image using the latent code. The resulting bottleneck forces the model to learn an efficient representation of the input. The reconstruction error is minimized using the mean squared error and gradient descent:

$$\text{Data} : \mathbf{X} \in \mathbb{R}^{n \times m}$$
$$\text{Encoder } E : \mathbb{R}^m \to \mathbb{R}^d$$
$$\text{Decoder } D : \mathbb{R}^d \to \mathbb{R}^m$$
$$\text{Reconstruction Error} : \frac{1}{n} \sum_i^n \|D(E(\mathbf{X}_{i,:})) - \mathbf{X}_{i,:}\|_2^2.$$

Note that the AE is an example of unsupervised learning, since learning the lower-dimensional embedding does not rely on labeled data. In the case of convolutional AEs, the encoder and the decoder consist of convolutional layers, activation functions and pooling layers. To achieve the desired code dimension, a linear layer is added.
To bring the latent representation back to the original dimension, the decoder has to upsample its input. For this we use transposed convolutions. These insert zeros between the pixels and then perform a normal convolution. Look here for a visualization.

1. Implement the convolutional AE of the provided architecture. Make use of the sequential container.

(4 pts.)

2. Implement the training function using the MSE as a loss function. Train the AE with latent dimensions 3, 10, 100, and 250. Visualize the train and test loss. For one sample, plot both the original image and the reconstruction for each model. Comment on your results.

(4 pts.)

3. Implement the PCA AE and train it with a latent dimension of 10. You can reuse the training function from Task 2. Plot again the train and test loss as well as the image

and its reconstruction. Compare your findings with the respective convolutional AE. Name further differences between the two model types that affect their application.

(3 pts.)

4. Select the convolutional AE with latent dimension 3 and encode a number of images. Create a 3d scatter plot of the code and use the classes as colors. Do the classes form separated clusters? If two classes are not well separable in their code representation, does this agree with the their labels?

(3 pts.)

Note: The trained AE could now also be used as a generative model. To do so, one would fit a probability distribution to the latent code (e.g. using a histogram in the simplest case). To generate new images, code is sampled from the distribution and transformed into images using the decoder.