Hamburg University of Technology
Institute of Mathematics
Chair Numerical Mathematics

Jens-Peter M. Zemke
Finn Sommer
Lars Stietz

Advanced Machine Learning
Winter Semester 2025/2026
Exercise Sheet 1

Week 2, 20.10.2025 – 24.10.2025

Homework:

Exercise 1: ((1+1+1)+2+1 points) level ●●○ relevance ●●●

a) Construct feedforward neural networks based on ReLU activations that implement
   (i) logical AND ($\wedge$) of two logical variables,
   (ii) logical OR ($\vee$) of two logical variables,
   (iii) logical XOR of two logical variables.
   Here, $0 \in \mathbb{R}$ encodes False and $1 \in \mathbb{R}$ encodes True, e.g., $0 \wedge 0 = 0$, $1 \vee 0 = 1$.

b) Can you find three feedforward neural networks based on ReLU activations that implement these three logical binary operators but with only *one* hidden layer such that the first weight matrix and first bias vector is the *same* for all three?

c) Prove that there does not exist a feedforward neural network based on ReLU activations that implements XOR with zero hidden layers.

Exercise 2: (2 points) level ●○○ relevance ●●○
Is it possible to construct a feedforward neural network with no hidden layer that implements the logical XOR based on a different activation function than ReLU? Justify your answer.

Exercise 3: (1+1+2 points) level ●○○ relevance ●●○

a) Read the Wikipedia entry on the Basic Linear Algebra Subprograms (BLAS),

   https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms.

b) Find out which variant of the BLAS your installed variant of NumPy uses by invoking

```
import numpy as np
np.__config__.show()
```

c) Implement a Python script that computes for given $n, k \in \mathbb{N}$ the product of two matrices $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times k}$ using NumPy and
   (i) nk scalar products (BLAS LEVEL 1),
   (ii) k matrix-vector products (BLAS LEVEL 2),
   (iii) one matrix-matrix product (BLAS LEVEL 3).
   How fast is each variant, say, e.g., for $n = 10.000$ and $k = 100$? You might have to reduce these numbers depending on your computer.

Exercise (in class):

Exercise 4: (2+2 points) level ●○○ relevance ●●●

Let H denote the Heaviside activation function,

$$H(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0. \end{cases}$$

Sketch the area of points $\mathbf{x} \in \mathbb{R}^2$ that are classified as one, i.e., $f(\mathbf{x}) = 1$ using the following neural network,

$$\mathbf{a}_1 = \mathbf{x}, \quad \mathbf{z}_1 = \mathbf{W}_1\mathbf{a}_1 + \mathbf{b}_1, \quad \mathbf{a}_2 = H(\mathbf{z}_1), \quad \mathbf{z}_2 = \mathbf{W}_2\mathbf{a}_2 + \mathbf{b}_2, \quad \mathbf{a}_3 = H(\mathbf{z}_2), \quad f(\mathbf{x}) = \mathbf{a}_3,$$

where

$$\mathbf{W}_1 = \begin{pmatrix} 0 & 1 \\ 1 & -1 \\ -1 & -1 \end{pmatrix}, \quad \mathbf{b}_1 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \quad \mathbf{W}_2 = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}, \quad \mathbf{b}_2 = \begin{pmatrix} -3 \end{pmatrix}.$$

Exercise 5: (2 points) level ●○○ relevance ●●●
Implement the `ReLU` activation function as a callable Python class, i.e., work with the methods `__init__(self)` and `__call__(self, x)` inside a class named `ReLU`. This class should be able to handle NumPy arrays.