Technische Universität Hamburg
Institut für Mathematik
Lehrstuhl Numerische Mathematik

Jens-Peter M. Zemke
Jonas Grams

Mathematics of Neural Networks
winter semester 2021/2022
exercise sheet 1

Exercise 1: ((1+1+1)+2+1 points)

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+2 points) Read the Wikipedia entry

https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Arnold_representation_theorem

and interpret the Kolmogorov–Arnold representation theorem (a) and the variant by Sprecher (b) each as a special type of feedforward neural network.

Exercise 3: (1+1+2 points)

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 4: ((2+2+2)+2+2 points)

a) Implement a feedforward neural network in Python.

(i) Implement the `ReLU` activation function as class in the provided `activation.py` script which has a method `evaluate(self, x)` that performs the application of the `ReLU` activation function

(ii) Implement a class `DenseLayer` in the given `layers.py` script which
- is initialized with integers specifiying the number of inputs and outputs, and an activation function (which is `ReLU` by default)
- has the attributes `W` and `b` for the weight matrix and the bias of this layer
- has the method `evaluate(self, a)` that performs the evaluation on the
- has the methods `set_weights` and `set_bias` for setting the weight matrix and the bias of a layer.

(iii) Implement a class `SequentialNet` in the provided `networks.py` script which
- has the attributes `layers` that stores all layers of the network and an integer `no` indicating the current number of outputs
- is initialized by an integer indicating the number of inputs and an (optional) list of layers
- has the method `evaluate(self,x)` that performs the feed forward with input `x`.

b) Write a Python script that tests the feed forward process for some given neural net. You may use your results from exercise 1.

c) (optional) Add a method `draw()` to the class `SequentialNet` that draws the neural network using circles for neurons and lines between them for the connecting weights.