AI5031: Machine learning, exercise 6

1 Implementing a ReLU layer

Write a python function *relu(X)* which takes a numpy array of arbitrary shape and returns a copy in which ReLU has been applied to every element!
In the main program, implement these test cases:

- $[1, 0, -1] \rightarrow [1, 0, 0]$
- $[1, 0, 10] \rightarrow [1, 0, 10]$
- $[-1, 0, -10] \rightarrow [0, 0, 0]$

2 Implementing an affine layer

Write a Python function *affine(X, W,b)* which applies the transformation $XW+\vec{b}$ to a matrix X (rows are data samples), where the bias vector $\vec{b}$ is added row-wise. In the main program, implement these test cases for $W = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}$, $\vec{b} = [1, -1, 0]$ and

- $X = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}$
- $X = \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}$
- $X = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$

3 Implementing a DNN

Re-use the multi-sample softmax function constructed last week and implement a DNN with the following structure:

0:Input-1:Affine-2:ReLU-3:Affine-4:Softmax.

The input should be $X = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}$.

The weight matrices and biases should be declared as global variables at the beginning of the program. They should have the values: $W^{(1)} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}$, $\vec{b}^{(1)} = [1, -1, 0]$, $W^{(3)} = \begin{pmatrix} 4 & 0 \\ 0 & -1 \\ 0 & -1 \end{pmatrix}$, $\vec{b}^{(3)} = [1, -1]$.