University of Stuttgart | ANALYTIC COMPUTING

Machine Learning (SS 23)

Assignment 10: Neural Networks

Mojtaba Nayyeri
Mojtaba.Nayyeri@ipvs.uni-stuttgart.de

Akram Sadat Hosseini
Akram.Hosseini@ipvs.uni-stuttgart.de

Nadeen Fathallah
Nadeen.Fathallah@ipvs.uni-stuttgart.de

Rodrigo Lopez Portillo Alcocer
rodrigo.lopez-portillo-alcocer@ipvs.uni-stuttgart.de

Tim Schneider
timphillip.schneider@ipvs.uni-stuttgart.de

Osama Mohammed
osama.mohammed@ipvs.uni-stuttgart.de

Daniel Frank
daniel.frank@ipvs.uni-stuttgart.de

Make sure to list the full names of all participants, matriculation number, study program, and B.Sc. or M.Sc on the first page. Optionally, you can *additionally* upload source files (e.g., PPTX files). If you have any questions, feel free to ask them in the exercise forum in ILIAS.

Submission is open until Monday, 10th of July 2023, 12:00 noon.

Formalizing Neural Networks

Create a Feed forward Neural Network with 3 hidden layers and an input dimension of 5. Each hidden layer uses the tanh activation function. Each layer includes a bias term. The FNN is a multi-class classifier for with 3 classes $\{-1, 0, 1\}$. Choose the appropriate output function. In your definition, provide the function for the forward-pass and the dimensions of each weight matrix and bias vector. Afterwards, define an appropriate loss function for your multi-class classification neural network. Assume that the loss function operates on batches of size 32. Then, extend the loss function to punish miss-classifications of class -1 more heavily than the other classes. Use a parameter $\alpha \geq 0$ to control this weighting. The extended loss function should be equivalent to the original loss function if $\alpha = 0$.

Solution: Input: $x = h^{(0)} \in \mathbb{R}^5$.

First hidden layer:$h^{(1)} = g_1(W^{(0)}h^{(0)} + b^{(0)}) \in \mathbb{R}^{l_1}, W^{(0)} \in \mathbb{R}^{l_1 \times 5}, b^{(0)} \in \mathbb{R}^{l_1}$,

Second hidden layer: $h^{(2)} = g_2(W^{(1)}h^{(1)} + b^{(1)}) \in \mathbb{R}^{l_2}, W^{(1)} \in \mathbb{R}^{l_2 \times l_1}, b^{(1)} \in \mathbb{R}^{l_2}$,

Third hidden layer: $h^{(3)} = g_3(W^{(2)}h^{(2)} + b^{(2)}) \in \mathbb{R}^{l_3}, , W^{(2)} \in \mathbb{R}^{l_3 \times l_2}, b^{(2)} \in \mathbb{R}^{l_3}$,

Output: $\hat{y} = h^{(o)} = g_4(W^{(o)}h^{(3)} + b^{(o)}) \in \mathbb{R}^{l_o}$.

The activation functions are as follows:

$g_1(z) = g_2(z) = g_3(z) = tanh(z)$;

$$g_4(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{3} e^{z_j}} \quad \text{(softmax function)},$$
$$W^{(o)} \in \mathbb{R}^{l_3 \times 3}, \quad b^{(3)} \in \mathbb{R}^o$$

Loss function (cross entropy, one-hot encoded y): $\mathcal{L} = \frac{-1}{32} \sum_{i=1}^{32} \sum_{j=1}^{3} y_{ij} log \hat{y}_{ij}$.

Extended loss function (indicator function [.]): $\mathcal{L} = \frac{-1}{32} \sum_{i=1}^{32} \sum_{j=1}^{3} (1 + \alpha[y_{ij} = -1]) y_{ij} log \hat{y}_{ij}$.

Neural Networks Example

Let us define a feedforward neural network for binary classification. The network takes a two-dimensional vector $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top$ as input and outputs a scalar $\hat{y} \in \mathbb{R}$.

We formally define this neural network as:

$$h = ReLU(W^h x + b^h)$$
$$\hat{y} = W^o h + b^o.$$

with the initialized parameters:

$W^h = \begin{bmatrix} w^h_{11} & w^h_{12} \\ w^h_{21} & w^h_{22} \\ w^h_{31} & w^h_{32} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.4 \\ 0.8 & 0.1 \\ 2.0 & 0.5 \end{bmatrix} \in \mathbb{R}^{3\times 2}$ is the hidden layer matrix, $b^h = \begin{bmatrix} 1.0 & 1.0 & -4.0 \end{bmatrix}$ is the hidden layer bias, and $W^o = \begin{bmatrix} w^o_1 & w^o_2 & w^o_3 \end{bmatrix} = \begin{bmatrix} 0.5 & -1.0 & 0.0 \end{bmatrix} \in \mathbb{R}^{1\times 3}$ is the output weight with its corresponding bias $b^o = 3.0$.

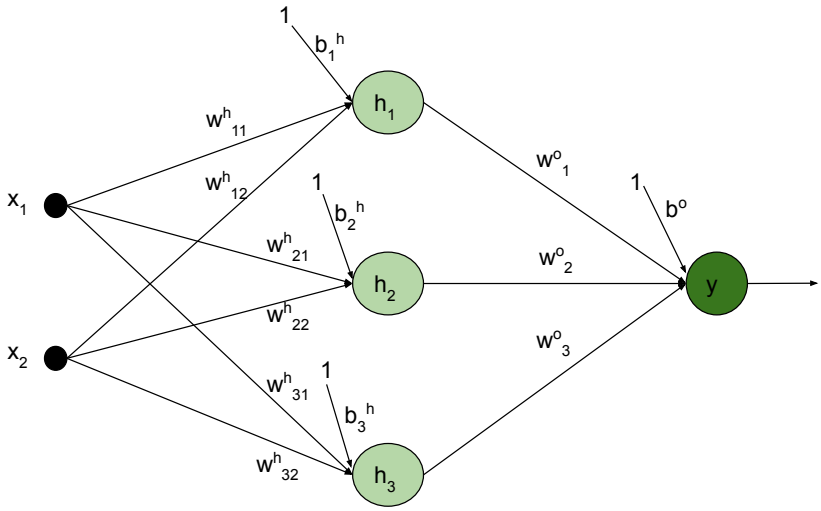

Figure 1 Neural Network

1) Compute the forward pass for $\begin{bmatrix} 1.0 & 2.0 \end{bmatrix}^\top$. Show each computation step.

$$z = W^h x + b^h = \begin{bmatrix} 0.2 & 0.4 \\ 0.8 & 0.1 \\ 2.0 & 0.5 \end{bmatrix} * \begin{bmatrix} 1.0 & 2.0 \end{bmatrix}^\top + \begin{bmatrix} 1.0 \\ 1.0 \\ -4.0 \end{bmatrix}$$

$$= \begin{bmatrix} 1.0 \\ 1.0 \\ 3.0 \end{bmatrix} + \begin{bmatrix} 1.0 \\ 1.0 \\ -4.0 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 2.0 \\ -1.0 \end{bmatrix}$$

$$h = ReLU(z) = max(0, z) = \begin{bmatrix} 2.0 \\ 2.0 \\ 0.0 \end{bmatrix}$$

$$\hat{y} = W^o h + b^0 = \begin{bmatrix} 0.5 & -1.0 & 0.0 \end{bmatrix} * \begin{bmatrix} 2.0 \\ 2.0 \\ 0.0 \end{bmatrix} + 3.0 = 2.0$$

2) What loss function would you use to train the given neural network for binary classification? Define the loss function formally. Compute the loss for the prediction made in the previous task given a true label of $y = -1$. Hint: Check the chapter "Feed-forward networks predicting a Bernoulli distribution" of the lecture slides on neural networks

We use softplus loss function, which is defined as follows:

$\mathcal{L} = log(1 + exp((1 - 2y)\hat{y}))$

The loss for $y = -1$ is computed as follows:

$\mathcal{L} = log(1 + exp((1 - 2(-1))2)) \approx 6.$

Implementing a Feedforward Neural Network

Download the provided Jupyter notebook Assignment10.ipynb and the dataset *airfoil_self_noise.csv*. Follow the instructions in the Jupyter notebook.