University of Stuttgart | ANALYTIC COMPUTING

Machine Learning (SS 23)

Assignment 11: Neural Networks (Continue) and PyTorch

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, 17th of July 2023, 12:00 noon.

Note:

This assignment builds upon the previous Neural Networks assignment from last week, encompassing additional components such as backpropagation and the implementation of the training process for a neural network.

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 $b^o = 3.0$.

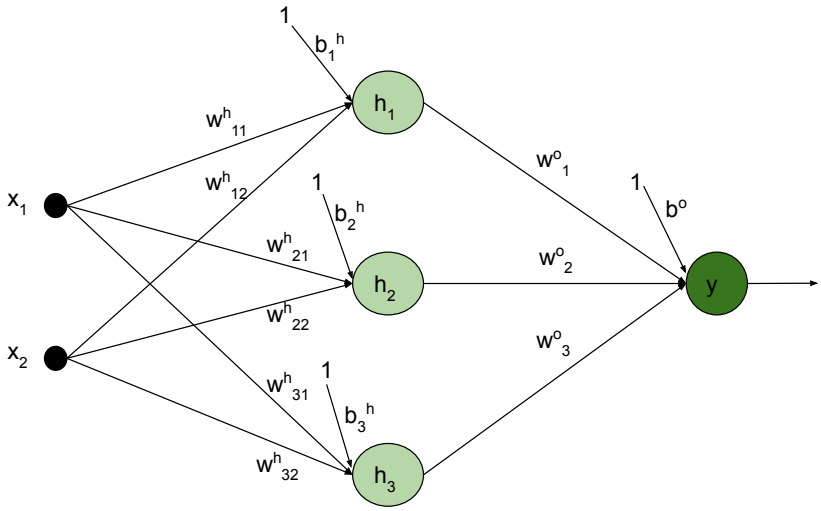

Figure 1 Neural Network

1) Compute the forward pass for $\begin{bmatrix} 1.0 & 2.0 \end{bmatrix}^\top$. Show each computation step (From assignment 10).

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

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

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

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

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 $y = -1$ (From assignment 10). 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:

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

3) Given the computed loss and a learning rate $\eta = 0.1$, adjust the weight W^h_{12} using backpropagation with gradient descent. Hint: Find an example step-by-step computation of backpropagation in footnote [1].

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \sigma((1 - 2y)\hat{y})\,(1 - 2y) = 3\sigma\,(3\hat{y}) = 3\sigma(6) \approx 3$$
$$\frac{\partial \hat{y}}{\partial h_1} = \frac{\partial 0.5h_1 - 1h_2 + 0h_3 + 3}{\partial h_1} = 0.5$$
$$\frac{\partial h_1}{\partial z_1} = \frac{\partial \max(0, z_1)}{\partial z_1} = 1$$
$$\frac{\partial z_1}{\partial W^h_{12}} = \frac{\partial (W_{h11}x_1 + W_{h12}x_2 + b^h_1)}{\partial w^h_{12}} = x_2 = 2$$

Now we can chain this operations to compute the gradient required for updating w^h_{12} :

$\frac{\partial \mathcal{L}}{\partial w^h_{12}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial h_1} \times \frac{\partial h_1}{\partial z_1} \times \frac{\partial z_1}{\partial W^h_{12}} = 3 \times 0.5 \times 1 \times 2 = 3.$

This gradient can now be used to update w^h_{12}:

$w^h_{12} = w^h_{12} - 0.1.3 = 0.4 - 0.3 = 0.1.$

[1] https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example

Implementing a Feedforward Neural Network

Download the provided Jupyter Notebook Assignment11_PyTorch_P1.ipynb, Assignment11_PyTorch_P2.ipynb, and the dataset $airfoil\_self\_noise.csv$. Follow the instructions in the Jupyter Notebook. (Forward pass was given in assignment 10, implementation of backward and training process are required).