 University of Stuttgart | 

Machine Learning Exercise (SS 21)

Assignment 8: Neural Networks (Solution)

Dr. Decky Aspandi
decky.aspandi-latif@ipvs.uni-stuttgart.de

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

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

This assignment sheet consists of 5 pages with the following 3 tasks:

- Task 1: Formalizing Neural Networks (30 Points) 2
- Task 2: Feedforward Neural Network: Theory (30 Points) 3
- Task 3: Feedforward Neural Network: Programming (40 Points) 5

Submit your solution in ILIAS as a single PDF file.[1] Make sure to list 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 Tuesday, 29 June 2021, 23:59.

[1] Your drawing software probably allows to export as PDF. An alternative option is to use a PDF printer. If you create multiple PDF files, use a merging tool (like pdfarranger) to combine the PDFs into a single file.

Task 1: Formalizing Neural Networks (30 Points)

Task (30 Points):
Create a Feed forward Neural Network with 2 hidden layers and an input dimension of 12. 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: $x = h^{(0)} \in^{12}$

First hidden layer: $h^{(1)} = g^{(1)}\left(W^{(1)^T} h^{(0)} + b^{(1)}\right) \in^{n}$

Second hidden layer: $h^{(2)} = g^{(2)}\left(W^{(2)^T} h^{(1)} + b^{(2)}\right) \in^{m}$

Output: $\hat{y} = h^{(3)} = g^{(3)}\left(W^{(3)^T} h^{(2)} + b^{(3)}\right) \in^{3}$

Further definitions:

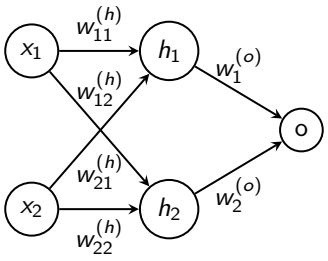
$$g^{(1)}(z) = \tanh(z),\ W^{(1)} \in^{12\times n},\ b^{(1)} \in^{n}$$
$$g^{(2)}(z) = \tanh(z),\ W^{(2)} \in^{n\times m},\ b^{(2)} \in^{m}$$
$$g^{(3)}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{3} e^{z_j}} \text{ (softmax function)},\ W^{(3)} \in^{m\times 3},\ b^{(3)} \in^{3}$$

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

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

Task 2: Feedforward Neural Network: Theory (30 Points)

The following FNN for regression takes two inputs x_1 and x_2 and outputs a scalar o:



The hidden layer, consisting of h_1 and h_2, uses the ReLU activation function. The output layer uses the identity. The weights are $w_{11}^{(h)} = 0.5$, $w_{12}^{(h)} = -0.2$, $w_{21}^{(h)} = 0.6$, $w^(h)_{22} = 0.5$, $w_1^{(o)} = 1$, $w_2^{(o)} = -1$.

Solution: Assumption for this solution: a weight w_{ij} connects a node x_i from the previous layer with a node x_j from the next layer, i.e. $x_j = w_{ij}\, x_i$.

1. **Task (10 Points)**: Compute the forward-pass for the input $x = (2, -0.5)$.

   Solution:

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ -0.5 \end{pmatrix}$$

$$W^{(h)} = \begin{pmatrix} w_{11}^{(h)} & w_{12}^{(h)} \\ w_{21}^{(h)} & w_{22}^{(h)} \end{pmatrix} = \begin{pmatrix} 0.5 & -0.2 \\ 0.6 & 0.5 \end{pmatrix}, \; W^{(o)} = \begin{pmatrix} w_1^{(o)} \\ w_2^{(o)} \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$g^{(h)}(z) = \max(0, z), \; g^{(o)}(z) = z$$

$$a = {W^{(h)}}^T x = \begin{pmatrix} 0.7 \\ -0.65 \end{pmatrix}$$

$$h = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = g^{(h)}(a) = g^{(h)}\left(\begin{pmatrix} 0.7 \\ -0.65 \end{pmatrix}\right) = \begin{pmatrix} 0.7 \\ 0 \end{pmatrix}$$

$$o = g^{(o)}({W^{(o)}}^T h) = 0.7$$

2. **Task (10 Points)**: Compute the squared error loss of the forward-pass for the true value $y = 1.5$.

   Solution: $L(o, y) = (o - y)^2 = (0.7 - 1.5)^2 = 0.64$

3. **Task (10 Points)**: Given the computed error, adjust the weight $w_{11}^{(h)}$ via back-propagation with gradient descent using a learning rate of 0.1.

   Solution:

$$\frac{\partial L(o, y)}{\partial o} = 2(o - y) = -1.6$$

$$\frac{\partial o}{\partial h_1} = \frac{\partial w_1^{(o)} h_1 + w_2^{(o)} h_2}{\partial h_1} = w_1^{(o)} = 1$$

$$\frac{\partial g^{(h)}(z)}{\partial z} = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

$$\frac{\partial h_1}{\partial a} = \frac{\partial g^{(h)}(a)}{\partial a} = \frac{\partial g^{(h)}(0.7)}{\partial 0.7} = 1$$

$$\frac{\partial a}{\partial w_{11}^{(h)}} = \frac{\partial w_{11}^{(h)} x_1 + w_{21}^{(h)} x_2}{\partial w_{11}^{(h)}} = x_1 = 2$$

Combining the gradients:

$$\frac{\mathrm{d}L(o,y)}{\mathrm{d}w_{11}^{(h)}} = \frac{\mathrm{d}L(o,y)}{\mathrm{d}o} \frac{\partial o}{\partial h_1} \frac{\partial h_1}{\partial a} \frac{\partial a}{\partial w_{11}^{(h)}} = -3.2$$

Update the weight $w_{11}^{(h)}$:

$$w_{11}^{(h)^{\text{new}}} = w_{11}^{(h)} - \eta \frac{\mathrm{d}L(o,y)}{\mathrm{d}w_{11}^{(h)}} = 0.5 - 0.1 \cdot (-3.2) = 0.82$$

FYI: Check if updating weight improved the result and reduces the error:

$$o^{\text{new}} = 1.34, L(o^{\text{new}}, y) = 0.0256$$

Task 3: Feedforward Neural Network: Programming (40 Points)

Task (40 Points): Please download the Jupyter notebook *Assignment8-FNN-Q.ipynb* and the dataset *airfoil_self_noise.csv*. Follow the instructions in the Jupyter notebook.

Solution in the Jupyter notebook *Assignment8-FNN-Q-Solution.ipynb*