Exercise for Machine Learning (SS 20)

Assignment 8: Neural Networks

Prof. Dr. Steffen Staab, steffen.staab@ipvs.uni-stuttgart.de
Alex Baier, alex.baier@ipvs.uni-stuttgart.de
Janik Hager, janik-manel.hager@ipvs.uni-stuttgart.de
Ramin Hedeshy, ramin.hedeshy@ipvs.uni-stuttgart.de
Analytic Computing, IPVS, University of Stuttgart

Submit your solution in Ilias as either PDF for theory assignments or Jupyter notebook for practical assignments.
Mention the names of all group members and their immatriculation numbers in the file.
Submission is possible until the following Monday, 15.06.2020, at 14:00.

1 Formalizing Neural Networks

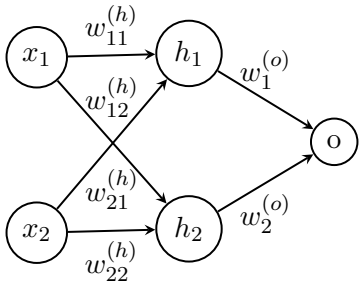
Define an FNN with 2 hidden layers and an input dimension of 12. Each hidden layer uses the tanh activation function. Each layer includes a bias term. The FNN is a multi-class classifier for with 3 classes ($\{-1, 0, 1\}$). Choose the appropriate output function. In your definition, provide the function for the forward-pass and the dimensions of each weight matrix and bias vector.
Define an appropriate loss function for your multi-class classification neural network. Assume that the loss function operates on batches of size 32.
Extend the loss function to punish misclassifications 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: $x = h^{(0)} \in \mathbb{R}^{12}$
First hidden layer: $h^{(1)} = g^{(1)}\left(W^{(1)^\top} h^{(0)} + b^{(1)}\right) \in \mathbb{R}^n$
Second hidden layer: $h^{(2)} = g^{(2)}\left(W^{(2)^\top} h^{(1)} + b^{(2)}\right) \in \mathbb{R}^m$
Output: $\hat{y} = h^{(3)} = g^{(3)}\left(W^{(3)^\top} h^{(2)} + b^{(3)}\right) \in \mathbb{R}^3$
Further definitions:

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

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

2 Feedforward Neural Network: Theory

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 $w_{11}^{(h)} = 0.5$, $w_{12}^{(h)} = -0.2$, $w_{21}^{(h)} = 0.6$, $w^(h)_{22} = 0.5$, $w_1^{(o)} = 1$, $w_2^{(o)} = -1$.

1. Compute the forward-pass for the input $x = (2, -0.5)$.
2. Compute the squared error loss of the forward-pass for the true value $y = 1.5$.
3. Given the computed error, adjust the weight $w_{11}^{(h)}$ via back-propagation with gradient descent using a learning rate of 0.1.

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. Forward-pass:

$$
\begin{aligned}
x &= \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ -0.5 \end{pmatrix} \\
W^{(h)} &= \begin{pmatrix} w_{11}^{(h)} & w_{12}^{(h)} \\ w_{21}^{(h)} & w_{22}^{(h)} \end{pmatrix} = \begin{pmatrix} 0.5 & -0.2 \\ 0.6 & 0.5 \end{pmatrix},\ W^{(o)} = \begin{pmatrix} w_1^{(o)} \\ w_2^{(o)} \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \\
g^{(h)}(z) &= \max(0, z),\ g^{(o)}(z) = z \\
a &= {W^{(h)}}^\top x = \begin{pmatrix} 0.7 \\ -0.65 \end{pmatrix} \\
h &= \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = g^{(h)}(a) = g^{(h)}\left(\begin{pmatrix} 0.7 \\ -0.65 \end{pmatrix}\right) = \begin{pmatrix} 0.7 \\ 0 \end{pmatrix} \\
o &= g^{(o)}({W^{(o)}}^\top h) = 0.7
\end{aligned}
$$

2. Squared error loss: $L(o, y) = (o - y)^2 = (0.7 - 1.5)^2 = 0.64$
3. Back-propagation:

$$\frac{\delta L(o,y)}{\delta o} = 2(o-y) = -1.6$$
$$\frac{\delta o}{\delta h_1} = \frac{\delta w_1^{(o)} h_1 + w_2^{(o)} h_2}{\delta h_1} = w_1^{(o)} = 1$$
$$\frac{\delta g^{(h)}(z)}{\delta z} = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$
$$\frac{\delta h_1}{\delta a} = \frac{\delta g^{(h)}(a)}{\delta a} = \frac{\delta g^{(h)}(0.7)}{\delta 0.7} = 1$$
$$\frac{\delta a}{\delta w_{11}^{(h)}} = \frac{\delta w_{11}^{(h)} x_1 + w_{21}^{(h)} x_2}{\delta w_{11}^{(h)}} = x_1 = 2$$

Combining the gradients:

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

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:

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

3 Feedforward Neural Network: Programming

Please download the Jupyter notebook *assignment8.ipynb* and the dataset *airfoil_self_noise.csv*. Follow the instructions in the Jupyter notebook.
Solution: Solution in the Jupyter notebook *assignment8_sol.ipynb*