Institute for Computer Science VI, Autonomous Intelligent Systems, University of Bonn

Dr. N. Goerke
Endenicher Allee 19a, 53115 Bonn, Tel: +49 228 73-4167
E-Mail: goerke (at) ais.uni-bonn.de
http://www.ais.uni-bonn.de/WS1920/4204_L_NN.html

Exercises for module Technical Neural Networks (MA-INF 4204), WS1920

Exercises sheet 2, due: Monday 21.10.2019

14.10.2019

Group	Name	8	9	10	11	12	13	14	Σ Sheet 2
A	Alessandro Riccardi	1	1	2	2	4	1	3	14
A*	Vladimir Fedoseev	1	1	2	2	4	1	3	14

(temporarily)

Remark:
Please hand in the solutions in paper before the start of the lecture (before 12:15).
Staple the solutions and make the assignment sheet the cover page.

! **Assignment 8** (1 Point)
Write down the δ-rule.
Explain <u>all</u> parts of the formula with a short sentence.

Assignment 9 (1 Point)
Define the term: *learning* for a technical system.
Please try to use a formal, generally applicable definition.

! **Assignment 10** (2 Points)
A given MLP with two hidden layers, N-H1-H2-M MLP shall be replaced by a second MLP (N-H-M) with only one single hidden layer, but (almost) the same number of weights.
Derive a formula for the number H of neurons in that hidden layer, and compute a value for H to replace a 5-21-30-4 MLP. Hint: please do not forget the BIAS.

! **Assignment 11** (2 Points)
Show by calculation that the first derivatives of the two typical transferfunctions for MLPs (*tanh* and *logistic function*) can be expressed by the transferfunctions themselves.

in every exame

! Assignment 12 (4 Points)

Prove in a strict formal way, **analytically**, that a simple Perzeptron (with step function) without a hidden layer is not capable to implement the Boolean function XOR.

Assignment 13 (1 Point)

Show by calcluation that the transferfunction $tanh(z)$ is identical to the shifted and rescaled Fermi-function (also called logistic function) $f_{\text{log}}(z) = \frac{1}{1+e^{-z}}$.

! Assignment 14 (4 Points)

Derive a new learning rule * for a Multi-Layer-Perceptron.
Start from the new objective function (cost function, error function) E^* and derive the new learning rule in analogy to Backpropagation of Error. Write down all calculation steps, and give the formulas for calculating the δ^* in output- and hidden layer.

$$^pE^*(w_{ij}) = \frac{1}{2}\sum_{m=0}^{M} ({^p\hat{y}_m} - {^py_m})^4$$

Programming-Assignment PA-A (5 Points, due 21.10.2019)

Implement a 2-layer Perzeptron (one input-layer, one output-layer) as a Java, C/C++ or Python program. The Perzeptron shall have an N-dimensional binary input **X**, an M-dimensional binary output **Y**, and a BIAS-weight for implementing the threshold. (N shall be less than 101, and M less than 30), initialize all weights randomly between $-0.5 \leq w_{n,m} \leq +0.5$

Implement further the **possibilities** to train the Perzeptron using the perzeptron learning rule with patterns (^pX, ^pY) that have been read in from a file named `PA-A-train.dat` (P shall be less than 200), and a possibility to read the weights $w_{n,m}$ from a file.

Please hand in your solution for the programming assignment by E-Mail to the tutor of your exercise group, **before Monday 21.10.2019, 12:00.**

The solution must contain your source code (C, C++, Java or Python), and a description how to compile and run your program.

Please try to make your program as platform independent as possible, and make sure, that your program is compiling and running from the console, without the need for a specific IDE or Software-Development-Kit.
Write down the commands how to compile and run your program from a console (terminal).

In addition, make sure that your program is running correctly, is producing the required results, and that your source code contains valid, and useful comments.

TNN

• Assignmnt 8

δ-rule = $\Delta w_{nm} = \eta \cdot {}^{p}\delta_m \cdot {}^{p}\widetilde{out}_h$

η = learning rate, this controls how quickly the model can learn and adapt to the problem.

${}^{p}\delta_m$ = is the definition of: $({}^{p}\hat{y}_m - {}^{p}y_m) \cdot f'({}^{p}net_m)$. Where $({}^{p}\hat{y}_m - {}^{p}y_m)$ is the difference between the teacher (expected value) and the output$_m$ (computed value).

$f'({}^{p}net_m)$ is the derivative of the transfer function $\frac{\partial y_m}{\partial\, {}^{p}net_m}$

${}^{p}out_h$ = is the output (result of the transfer function) of the neuron ~~g~~ = h of the hidden layer above

1

• Assignment 9

Learning for a technical system means changing or adapting its parameters to reach a specific goal or objective. This is possible by analyzing data from the process and using an algorithm that permit to change those parameters of the model.

1

• Assignment 11

$$f_{logistic}(z) = \frac{1}{1+e^{-z}} \quad \rightarrow \quad f'_{logistic}(z) = \frac{1' \cdot (1+e^{-z}) - 1\,(1+e^{-z})'}{(1+e^{-z})^2} =$$

$$= \frac{e^{-z}}{(1+e^{-z})^2} \Rightarrow \underset{\text{chain rule}}{\text{for the}} \Rightarrow \frac{1+e^{-z}-1}{(1+e^{-z})^2} = \frac{1+e^{-z}}{(1+e^{-z})^2} - \left(\frac{1}{1+e^{-z}}\right)^2 = \frac{1}{(1+e^{-z})} - \left(\frac{1}{1+e^{-z}}\right)^2$$

$$= f_{logistic}(z) \cdot \left(1 - f_{logistic}(z)\right)$$

$\longrightarrow$

$$f\,\tanh(z) = \left(\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}\right) \qquad f'\,\tanh(z) = \frac{\partial}{\partial z}\left(\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}\right) =$$

$$= \frac{(e^{z}-e^{-z})'(e^{z}+e^{-z}) - (e^{z}\cdot e^{-z})\cdot(e^{z}+e^{-z})'}{(e^{z}+e^{-z})^2} = \frac{\sinh(z)'\cdot\cosh(z) - \sinh(z)\cdot\overset{\cosh(z)}{\vee}}{(e^{z}+e^{-z})^2}$$

$$= \frac{(e^{z}+e^{-z})^2-(e^{z}-e^{-z})^2}{(e^{z}+e^{-z})^2} = 1 - \frac{\sinh^2(z)}{\cosh^2(z)} = 1-\tanh^2(z)$$

ssignment 10.

Number of weights in the initial MLP:

$(N+1)H_1 + (H_1+1)H_2 + (H_2+1)M$ ('+1' due to BIASes)

Number of weights in the desired MLP:

$(N+1)H + (H+1)M + C$ (C is a relatively small constant, so the numbers of weights can be 'almost' equal).

$NH_1 + H_1 + H_1H_2 + H_2 + H_2M + \cancel{M} - C =$

$= NH + \underset{H}{\cancel{H}} + HM + \cancel{M}$

$NH_1 + H_1 + H_1H_2 + H_2 + H_2M - C = H(N+M+1)$

$H = \frac{H_1(N+1) + H_2(M+1) + H_1H_2 - C}{N+M+1}$, so C is needed for making this fraction equal to an integer, since the number of neurons cannot be not an integer. Then it can be rewritten as

$$H = \left\lceil \frac{H_1(N+1) + H_2(M+1) + H_1H_2}{N+M+1} \right\rceil$$ (ceiling function).

For 5-21-30-4 MLP the total number of weights is 910.

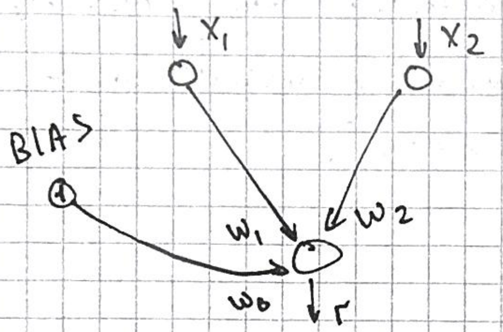
$$H_{5-21-30-4} = \left\lceil \frac{21(5+1) + 30(4+1) + 21\cdot 30}{5+4+1} \right\rceil = \left\lceil \frac{126 + 150 + 630}{10} \right\rceil =$$

$$= \left\lceil \frac{906}{10} \right\rceil = 91$$

✓ 2

Assignment 12

Since XOR is a function mapping a 2D vector (x_1, x_2) to a scalar r, the structure of a Perzeptron that aims to implement is: 2 input neurons, 1 output neuron.



Then, $net_m = w_1 x_1 + w_2 x_2 + w_0$.
For the step function is $SGN(net_m - w_0)$, so for any w_0 the ~~ras~~ value of $SGN(net_m - w_0)$ should comply to XOR:

x_1	x_2	r		
0	0	0		$SGN(0 \cdot w_1 + 0 \cdot w_2 - w_0) = 0$
0	1	1	OR	$SGN(0 \cdot w_1 + 1 \cdot w_2 - w_0) = 1$
1	0	1		$SGN(1 \cdot w_1 + 0 \cdot w_2 - w_0) = 1$
1	1	0		$SGN(1 \cdot w_1 + 1 \cdot w_2 - w_0) = 0$

OR
$$-w_0 < 0$$
$$w_2 - w_0 > 0$$
$$w_1 - w_0 > 0$$
$$w_1 + w_2 - w_0 < 0 \quad (*)$$

$w_0 > 0$ and $w_2 - w_0 > 0$ gives $w_2 > 0$

$w_0 > 0$ and $w_1 - w_0 > 0$ gives $w_1 > 0$

given ~~$w_2 > 0$~~, $w_2 - w_0 > 0$ and $w_1 > 0$, $w_1 + w_2 - w_0 > 0$, which contradicts with $(*)$. ▨

[illegible]

signment 14

$$\Delta w = -\eta \frac{\partial\, {}^{p}E(w_s)}{\partial w_s}$$

For the output layer: ${}^{p}\delta_m = -\frac{\partial\, {}^{p}E(w_{hm})}{\partial\, net_m}$

$$\frac{\partial\, {}^{p}E(w_{hm})}{\partial w_{hm}} = \frac{\partial}{\partial w_{hm}} \frac{1}{2} \sum_{j=1}^{M} \left(\hat{y}_j - y_j(w_{hm})\right)^4$$

$$= \frac{\partial\, {}^{p}E(w_{hm})}{\partial\, net_m} \cdot \frac{\partial\, net_m}{\partial w_{hm}}$$

$$\frac{\partial\, net_m}{\partial w_{hm}} = \frac{\partial}{\partial w_{hm}} \sum_{g=0}^{H} \widetilde{out}_g \cdot w_{gm} = {}^{p}\widetilde{out}_h$$

$$\frac{\partial\, {}^{p}E(w_{hm})}{\partial\, net_m} = \frac{\partial\, {}^{p}E}{\partial y_m} \cdot \frac{\partial y_m}{\partial\, net_m} \longrightarrow \frac{\partial y_m}{\partial\, net_m} = \frac{\partial f(net_m)}{\partial\, net_m} = f'(net_m)$$

(y_m - transfer function of net_m)

$$\frac{\partial\, {}^{p}E}{\partial y_m} = \frac{\partial}{\partial y_m} \frac{1}{2} \sum_{j=1}^{M} \left(\hat{y}_j - y_j\right)^4 =$$

$$= \frac{1}{2} \frac{\partial}{\partial y_m} \left(\hat{y}_m - y_m\right)^4$$ only $j=m$ part contributes

$$= \frac{1}{2} 4 \left(\hat{y}_m - y_m\right)^3 \cdot \frac{\partial}{\partial y_m}(-y_m)$$

$$= -2\left(\hat{y}_m - y_m\right)^3$$

$$\Rightarrow \frac{\partial\, {}^{p}E(w_{hm})}{\partial w_{hm}} = -2\left(\hat{y}_m - y_m\right)^3 \cdot f'(net_m) \cdot {}^{p}\widetilde{out}_h$$

3

$$\Delta w = \eta \cdot 2\left(\hat{y}_m - y_m\right)^3 \cdot f'(net_m) \cdot {}^{p}\widetilde{out}_h$$

$\Rightarrow$ $\boxed{\delta_m = -2\left(\hat{y}_m - y_m\right)^3 \cdot f'(net_m)}$ (δ for ~~the~~ an output neuron) and $\boxed{\Delta w = \eta\, {}^{p}\delta_m\, {}^{p}\widetilde{out}_h}$ (learning rule)

For the hidden layer: how??

$$\boxed{\delta_h = \left(\sum_{k=1}^{K} \delta_k \cdot w_{hk}\right) \cdot f'(net_h)}$$ and $\boxed{\Delta w = \eta\, \delta_h \cdot \widetilde{out}_g}$ (learning rule)

δ for the hidden layer

δ and w for hidden layer, each $\delta_k = \delta_m$.