Laboratory 1
Multilayer Perceptron

1. Task specification

Write a program which implements multilayer perceptron of the structure given below in Fig. 1 (without BIAS neuron denoted as "1" on the sketch).

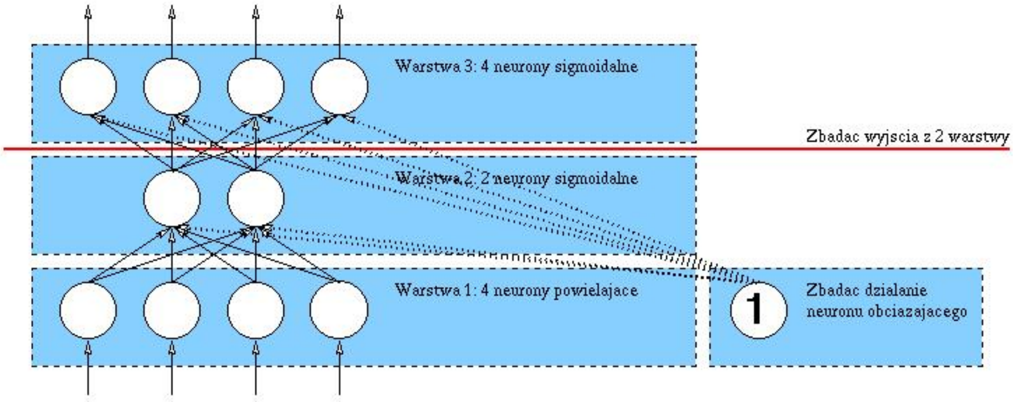

Warstwa 3: 4 neurony sigmoidalne
Zbadac wyjscia z 2 warstwy
Warstwa 2: 2 neurony sigmoidalne
Warstwa 1: 4 neurony powielajace
1
Zbadac dzialanie neuronu obciazajacego

Fig. 1. Multilayer perceptron

Perceptron has three layers input listed below

- input layer with 4 neurons,
- hidden layer with 2 neurons,
- output layer with 4 neurons.

It should be taught (trained) the following training patterns

1. `([1,0,0,0],[1,0,0,0]),`
2. `([0,1,0,0],[0,1,0,0]),`
3. `([0,0,1,0],[0,0,1,0]),`
4. `([0,0,0,1],[0,0,0,1]),`

using error backpropagation method which was discussed during laboratory.

How many patterns out of the four above can be trained, all of them or not?

In the next step add a BIAS neuron denoted as "1" in the Fig. 1 and using the same backpropagation method try to repeat the training process. Now can all patterns be trained? The BIAS neuron is the normal neuron with output always equal to one and which outputs are connected to the last additional weights of each neuron in the hidden and output layers.

Please report what values are returned by the output neurons for each of the training patterns after the training process has finished. Please report also what are the outputs of the hidden layer neurons after the training process for each of the training patterns put on the networks' inputs. Can you observe some regularity regarding how the four training patterns are coded inside the network by the hidden neurons?

2. Backpropagation method as discussed at the laboratory

In the following figures the photo shoots are depicted regarding the backpropagation algorithm discussed at he laboratory for a multilayer perceptron without the additional BIAS neuron. The algorithm stays absolutely analogous when the BIAS neuron is added.

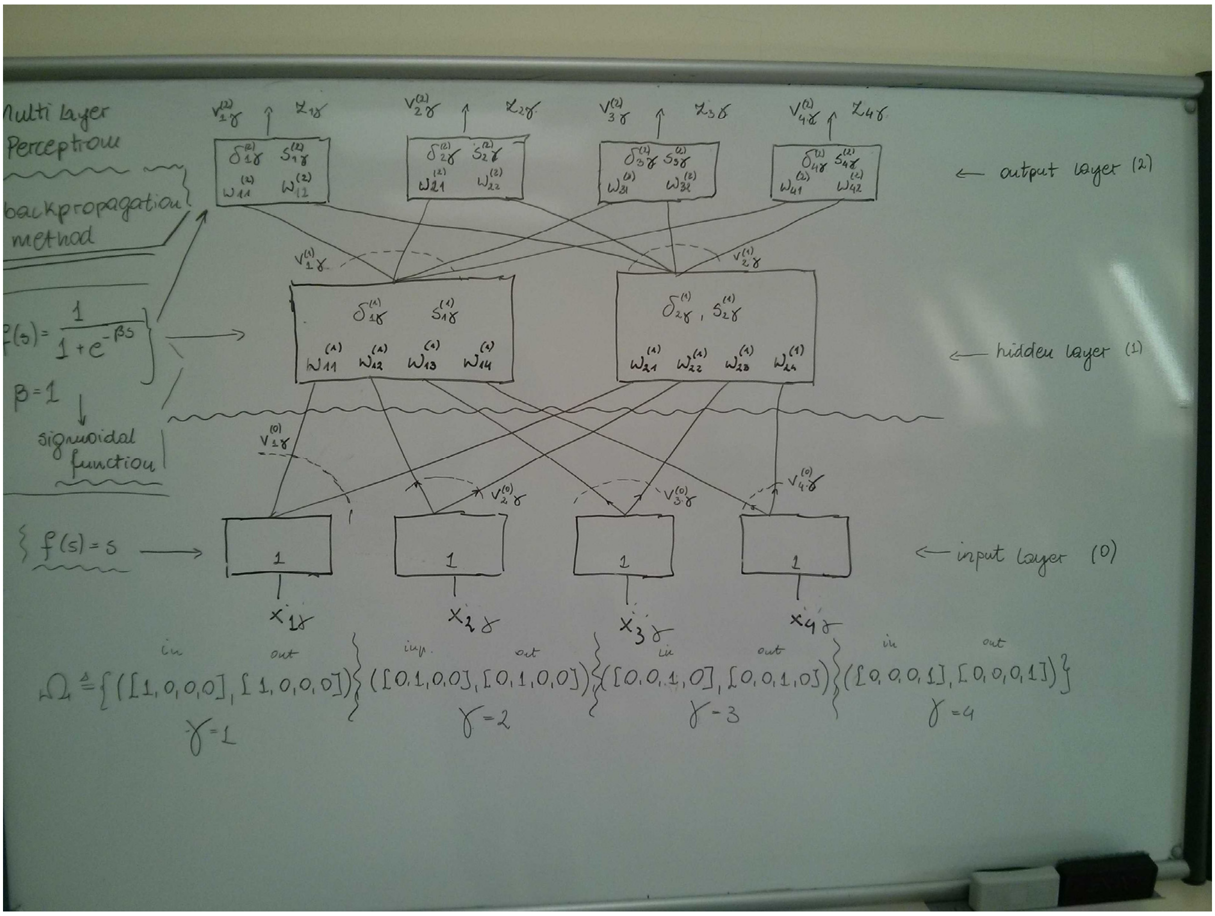
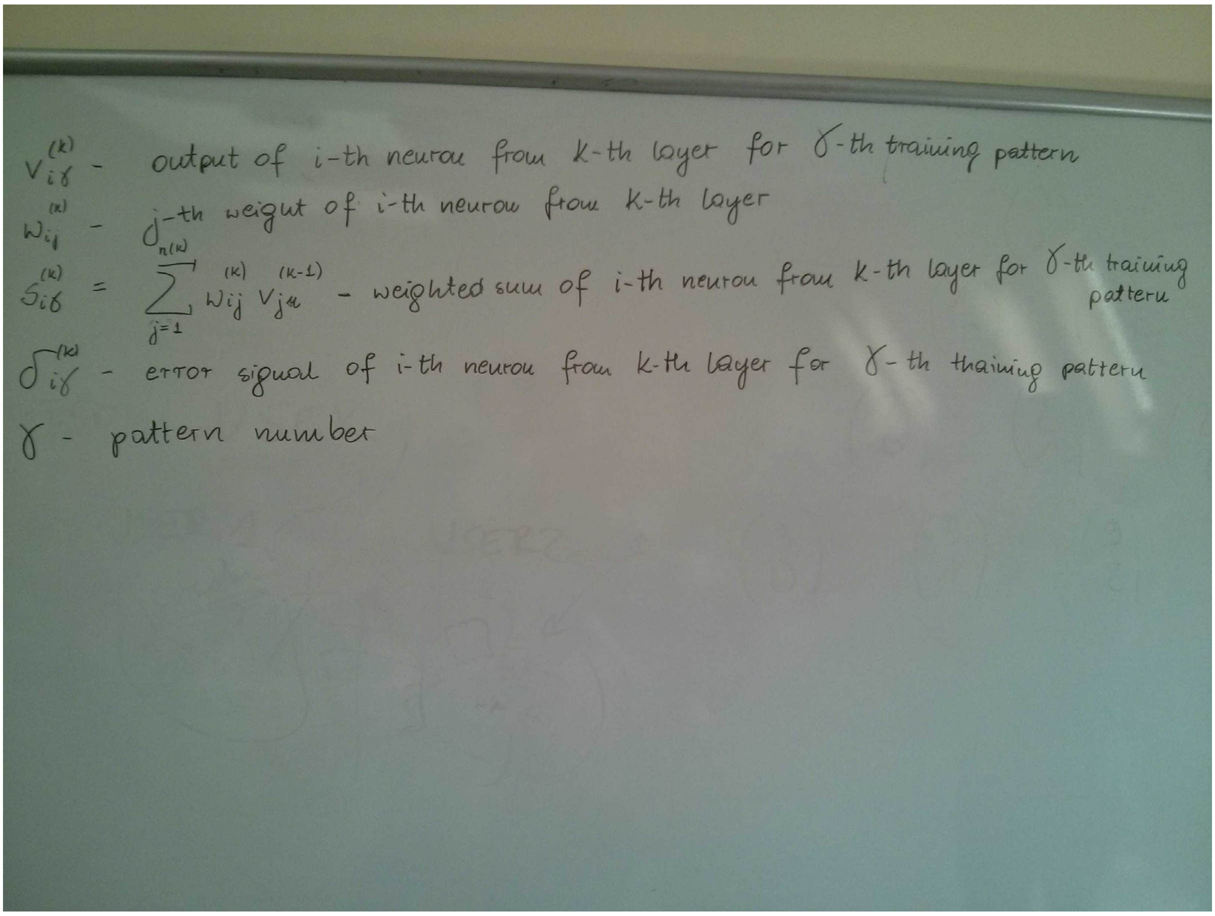

Multi Layer Perceptron
backpropagation method
f(s) = 1 / (1 + e^{-βs})
β = 1
sigmoidal function
f(s) = s
output layer (2)
hidden layer (1)
input layer (0)
v(2)1γ z1γ
v(2)2γ z2γ
v(2)3γ z3γ
v(2)4γ z4γ
δ(2)1γ s(2)1γ w(2)11 w(2)12
δ(2)2γ s(2)2γ w(2)21 w(2)22
δ(2)3γ s(2)3γ w(2)31 w(2)32
δ(2)4γ s(2)4γ w(2)41 w(2)42
v(1)1γ
v(1)2γ
δ(1)1γ s(1)1γ w(1)11 w(1)12 w(1)13 w(1)14
δ(1)2γ, s(1)2γ w(1)21 w(1)22 w(1)23 w(1)24
v(0)1γ
v(0)2γ
v(0)3γ
v(0)4γ
1
1
1
1
x1γ
x2γ
x3γ
x4γ
in out
Ω ≜ {([1,0,0,0],[1,0,0,0]) ([0,1,0,0],[0,1,0,0]) ([0,0,1,0],[0,0,1,0]) ([0,0,0,1],[0,0,0,1])}
γ=1
γ=2
γ=3
γ=4

Fig. 2. Multilayer perceptron photo shoot



$v_{i\gamma}^{(k)}$ - output of i-th neuron from k-th layer for γ-th training pattern

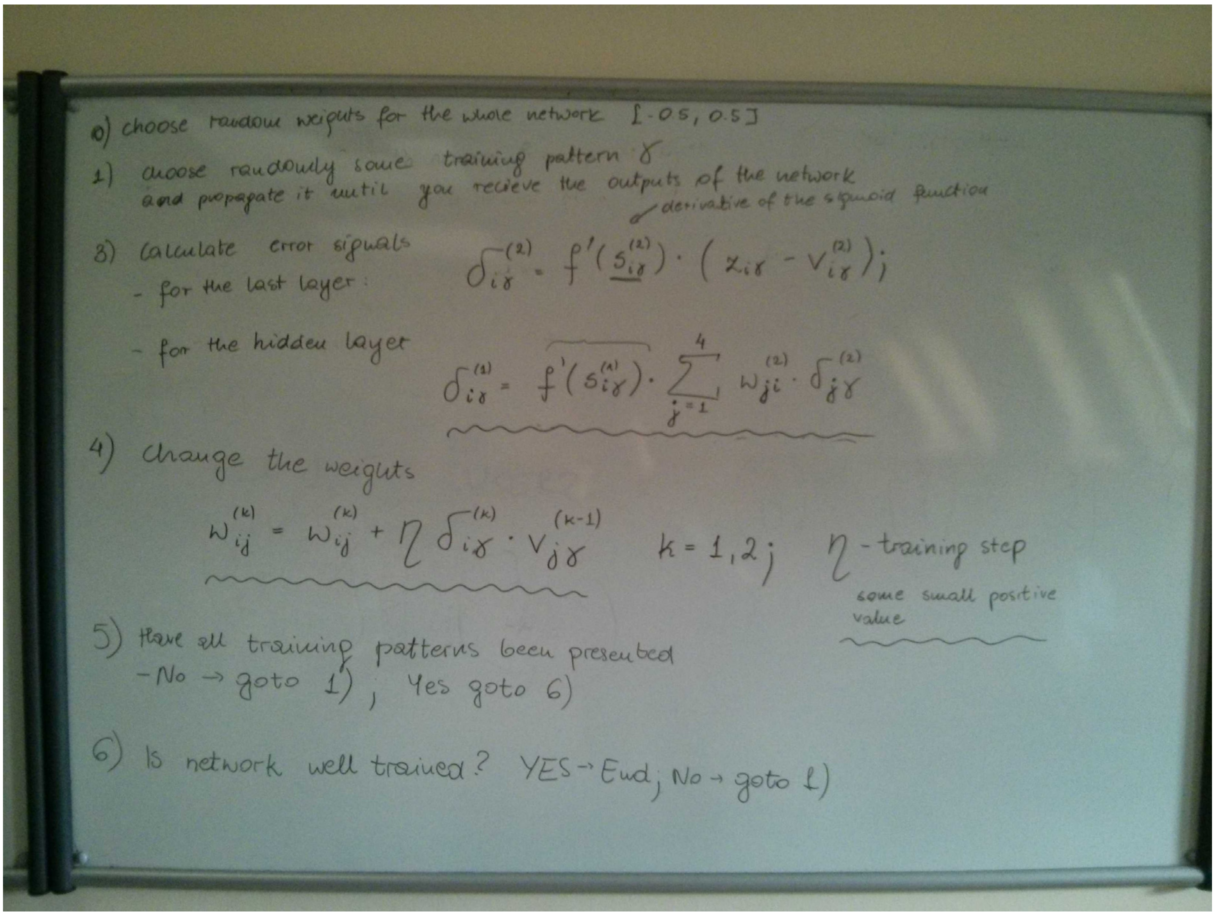
$w_{ij}^{(k)}$ - j-th weigut of i-th neuron from k-th layer

$s_{i\gamma}^{(k)} = \sum_{j=1}^{n(k)} w_{ij}^{(k)} v_{j\gamma}^{(k-1)}$ - weighted sum of i-th neuron from k-th layer for γ-th training pattern

$\delta_{i\gamma}^{(k)}$ - error signal of i-th neuron from k-th layer for γ- th thaining pattern

γ - pattern number

Fig. 3. Symbols used



0) choose random weights for the whole network [-0.5, 0.5]

1) choose randomly some training pattern γ
and propagate it until you recieve the outputs of the network

3) calculate error signals
- for the last layer:

derivative of the sigmoid function

$$\delta_{i\gamma}^{(2)} = f'(\underline{s_{i\gamma}^{(2)}}) \cdot (z_{i\gamma} - v_{i\gamma}^{(2)});$$

- for the hidden layer

$$\underline{\delta_{i\gamma}^{(1)} = \overline{f'(s_{i\gamma}^{(1)})} \cdot \sum_{j=1}^{4} w_{ji}^{(2)} \cdot \delta_{j\gamma}^{(2)}}$$

4) change the weights

$$\underline{w_{ij}^{(k)} = w_{ij}^{(k)} + \eta \, \delta_{i\gamma}^{(k)} \cdot v_{j\gamma}^{(k-1)}} \quad k = 1,2; \quad \eta \text{ - training step}$$

$\underline{\text{some small positive value}}$

5) Have all training patterns been presented
-No → goto 1), Yes goto 6)

6) Is network well trained? YES → End; No → goto 1)

Fig. 4. Backpropagation algorithm