Submission until: **21.06.2015**
Discussion on: 23.06.2015
Submission as upload to your groups stud.IP folder as *groupNumber_sheet8.zip* or *groupNumber_sheet8_mnist.zip* if optional task included.

Assignment 1 *(Multi-Layer Perceptron (8p))*

1. Draw multi-layer perceptron to solve each given logical function below:
   (a) $x_1 \oplus x_2$ (xor)
   (b) $(x_1 \wedge x_4) \vee (x_2 \wedge x_3)$
   (c) $x_1 \vee (x_2 \wedge x_3)$
2. Prove that derivatives of the following activation functions are true:
   (a) binary sigmoid function $f(x) = \frac{1}{1+\exp(-\lambda x)}$ with $f'(x) = \lambda f(x)[1 - f(x)]$, where λ is a constant.
   (b) bipolar sigmoid function $f(x) = \frac{2}{1+\exp(-x)} - 1$ with $f'(x) = \frac{1}{2}[1 + f(x)][1 - f(x)]$
3. Binary sigmoid has its function values in the range $(0, 1)$, while bipolar sigmoid function has its function values in the rage $(-1, 1)$. We want to extend sigmoid function to have function values in the range of (a, b). Write down a general sigmoid function and its derivative.

Assignment 2 *(Backpropagation (4p))*

(a) How to avoid local minima in backpropagation?

(b) Explain the generalization and avoiding overfitting.

(c) To prevent overly large weights which cause the high sensitivity of inputs, we apply the quadratic regularization term in the error function. Use gradient descent to minimize this error function.

Assignment 3 *(Backpropagation (12p))*

Nomenclature:

t	Output target vector
α	learning rate
δ	Portion of error correction weight adjustment

Algorithm:

1. Initialize weights to small random values
2. While stopping condition is false:

   For each training pair:

   3 Each input unit $(X_i, i = 1..n)$ broadcasts the signal to the layer above

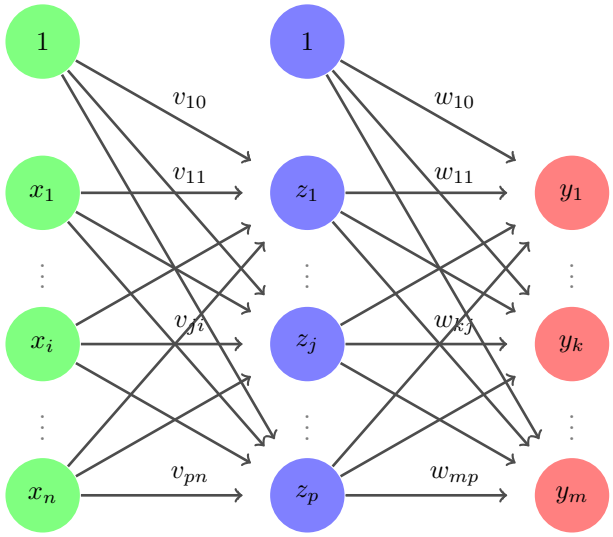

Figure 1: Backpropagation neural network with one hidden layer

4 Each hidden unit $(Z_j, j = 1..p)$ sums its weighted input signals z_in_j and applies its activation function to compute its output signal z_j and sends this signal to all units in the layer above (output units)

5 Each output unit $Y_k, k = 1..m$ sums its weighted input signals y_in_k, and applies its activation function to compute its output signal y_k.

6 Each output unit has a target pattern corresponding to the training pattern, computes its error information term

$$\delta_k = (t_k - y_k) f'(y\_in_k)$$

The weight correction term used to update weight later, can be calculate from

$$\triangle w_{kj} = \alpha \delta_k z_j$$

Sends δ_k to units in the layer below.

7 Each hidden unit Z_j sums its delta inputs from units in layer above,

$$\delta\_in_j = \Sigma_k \delta_k w_{kj} \ ,$$

multiplies by the derivative of its activation function to calculate its error information term,

$$\delta_j = \delta\_in_j f'(z\_in_j)$$

The weight correction term used to update v_{ji} later can be calculated from

$$\triangle v_{ji} = \alpha \delta_j x_i$$

8 Each output unit Y_k updates its bias and wights:

$$w_{kj}(new) = w_{kj}(old) + triangle w_{kj}$$

Each hidden unit Z_j updates its bias and weights:

$$v_{ji}(new) = v_{ji}(old) + triangle v_{ji}$$

9. Test stop condition

Tasks:

1. Implement backpropagation for multi-layer perceptron for one hidden layer. Choose any training and test samples with two input features and output which can be represented as one neuron or use the sample from previous tasks to train the network. You may need to change the suitable input and output range for the data that fit the selected activation function. Select an appropriate learning rate, activation function, stop condition and number of neurons in hidden layer. Do not forget to provide your dataset with your code. Plot the training and test data on the same figure. What if you change to a bigger learning rate and smaller learning rate, discuss the result. (10p)
2. Apply momentum from p.55 to update the weights. Select a momentum parameter which is constrained to be in the range from 0 to 1, exclusive of the end points. It should be bigger than the learning rate. Compare the result from traditional weight update. (2p)

Optional (digit recognition):

Figure 2: Examples of handwritten digits from MNIST[a]

[a] http://yann.lecun.com/exdb/mnist/

Examples of handwritten digits MNIST database, http://yann.lecun.com/exdb/mnist/, are shown in the Figure.

1. Adapt the implemented backpropagation to the given small set of MNIST database, mnist2000.mat. This dataset consists of 1200 images for training and each 400 images to validate and to test the model. The data are stored in Matlab's cell along with their target outputs. Each image has a size of 28×28 pixels which stored as a vector of size 784. Input images are already converted from the gray-scale value into the range between 0 and 1. The targets are numbers from 0 to 9. What to consider are for example (i) how to represent the training pattern and target to the network i.e. number of output units represent number 0 to 9, (ii) what is a suitable activation function (iii) what is a good learning rate and momentum parameter, and (iv) how long to train the net.
2. File mnist2000.mat contains six cells. To display the first image from the first cell, for Matlab:

```
load('mnist2000.mat');
x = mnist{1}; % first cell is for training pattern
t = mnist{2}; % second cell is for target number
% show first image in the training pattern which
% stored in 1 vector size 784:
imshow( reshape(x(1,:), [28,28]), [] ); % number 5
% check the target vector at first position
t1 = t(1,1); % return 5
```

For Python:

```
mat = io.loadmat('mnist2000.mat')
M = mat['mnist']
x = M[0,0]; # first cell return ndarray of size 1200
    x784
t = M[0,1]; # second cell retrun ndarray of size 1200
    x1
# get first image in the traing pattern
x1 = x[0,:].reshape((28,28))
plt.imshow(x1.T, cmap='gray') #display
plt.show()
```

3. You may apply other techniques which discussed in the lecture to improve network's stability and speed. Plot accuracy of your test patterns vs. number of simulations step and submit your file as *groupNumber_sheet8_mnist.zip*. ***Challenge*** is to get the highest average score of accuracy of test patterns with least number of simulation steps.