Multilayer feed-forward backpropagation networks

Transfer function types:

purelin – linear function
logsig – sigmoid function
tansig – hyperbolic tangent function

Main command creating multilayer feed-forward backpropagation network without a division of data into learning and testing parts:

```
net = newff(M,[S1 ... Sn],{F1 ... Fn})
```

M – matrix defining minimum and maximum value for each input
S1 ... Sn – number of neurons on succeeding layers (the last must agree with number of outputs)
TF1 ... TFn – transfer functions of succeeding layers (possible are: `purelin`, `logsig`, `tansig`)

For example the command:

```
net = newff(minmax(inp), [10 6], {'logsig', 'purelin'});
```

will create two-layer network, with 10 neurons on the hidden layer and 6 on the oytput layer. Neurons on the hidden layer will have sigmoid transfer function and neurons on the output layer will have linear transfer function.

```
net = newff(minmax(inp), [8 5 1], {'tansig', 'tansig', 'logsig'});
```

will create three-layer network, with 8 neurons on the I hidden layer (with hyperbolic tangent transfer function), 5 neurons on the II hidden layer (also with hyperbolic tangent transfer function) and 1 neuron on the output layer (with sigmoid transfer function).

Main command creating multilayer feed-forward backpropagation network with a division of data into learning and testing parts:

```
net = newff(inp,out,[h1 h2],{F1 ... Fn});
```

inp – matrix of input data
out – vector (or matrix) of given output data
[h1 h2] – vector definig number of neurons on succeeding hidden layers
`{F1 ... Fn}`– vector definig transfer functions of all succeeding layers

Examples:

The network with 1 hidden layer. On the hidden layer we have 5 neurons with hyperbolic tangent activation function and on the output layer neurons have linear activation function.

```
net = newff(inp,out,[5],{'tansig', 'purelin'});
```

The network with 2 hidden layers. We have 10 neurons with hyperbolic tangent activation function on the I hidden layer, 4 neurons with sigmoid function on the II hidden layer and neurons with linear function on the output layer.

```
net = newff(inp,out,[10 4],{'tansig', 'logsig','purelin'});
```

Example for 1-input data:

```
inp = load('data_1D_sin1_i.txt'); inp = inp';
out = load('data_1D_sin1_o.txt'); out = out';

net = newff(minmax(inp),[4 1],{'logsig', 'purelin'}); % no data division
%net = newff(inp,out,[4],{'logsig', 'purelin'});      % with data division
net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
net = init(net); % random initation of weights
net.trainParam.epochs = 100;

figure(1)
inp1 = linspace(min(inp),max(inp),1000);
out1 = sim(net, inp1);
plot(inp1,out1,'r'); hold on;
plot(inp,out,'.b')
title('Characteristic of the model before learning')

net = train(net, inp, out);

figure(2)
out2 = sim(net, inp1);
plot(inp1,out2,'g'); hold on;
plot(inp,out,'.b')
title('Characteristic of the model after learning')
```

Example for 2-input data:

```
inp = load('data_2D_5_i.txt'); inp = inp';
out = load('data_2D_5_o.txt'); out = out';

net = newff(minmax(inp),[5 1],{'tansig', 'logsig'}); % no data division
%net = newff(inp,out,[5],{'tansig', 'logsig'});      % with data division
%net.outputs{2}.processFcns = {};
net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
net = init(net); % random initation of weights
net.trainParam.epochs = 100;

figure(1)
out1 = sim(net, inp);
x_lin = linspace(min(inp(1,:)),max(inp(1,:)),50);
y_lin = linspace(min(inp(2,:)),max(inp(2,:)),50);
[X,Y] = meshgrid(x_lin,y_lin);
Z = griddata(inp(1,:), inp(2,:), out1, X, Y, 'cubic');
mesh(X,Y,Z)
hold on
plot3(inp(1,:), inp(2,:), out,'.');
title('Surface of the model before learning')

net = train(net, inp, out);

figure(2)
out2 = sim(net, inp);
x_lin = linspace(min(inp(1,:)),max(inp(1,:)),50);
y_lin = linspace(min(inp(2,:)),max(inp(2,:)),50);
[X,Y] = meshgrid(x_lin,y_lin);
Z = griddata(inp(1,:), inp(2,:), out2, X, Y, 'cubic');
mesh(X,Y,Z)
hold on
plot3(inp(1,:), inp(2,:), out,'.');
title('Surface of the model after learning')
```

```
figure(3) % plot should be done only for binary classification task
%plot(inp(1,:), inp(2,:),'.'); hold on
plotpv(inp,out); hold on
contour(X,Y,Z,[0.5])
title('Separation line')

figure(4)
x_lin = linspace(min(inp(1,:)),max(inp(1,:)),50);
y_lin = linspace(min(inp(2,:)),max(inp(2,:)),50);
[X,Y] = meshgrid(x_lin,y_lin);
Z = griddata(inp(1,:), inp(2,:), out, X, Y, 'cubic');
mesh(X,Y,Z)
hold on
plot3(inp(1,:), inp(2,:), out,'.');
title('Interpolated data surface')

sim(net, [0.5;-0.4])    % test the network output for exemplary point
                        % (0.5 , -0.4)
```

Example for data with number of inputs greater than 2:

```
inp = load('data_8D_diabet_i.txt'); inp = inp';
out = load('data_8D_diabet_o.txt'); out = out';

net = newff(minmax(inp),[10 1],{'tansig', 'logsig'}); % no data division
%net = newff(inp,out,[10],{'tansig', 'logsig'});      % with data division
%net.outputs{2}.processFcns = {};
net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
net = init(net); % random initation of weights
net.trainParam.epochs = 100;

% plots for the specified output:
nr_output = 1;
figure(1)
y = sim(net, inp);
plot(abs(y(nr_output,:)-out(nr_output,:)),'r'); hold on;

net = train(net, inp, out);

y1 = sim(net, inp);
plot(abs(y1(nr_output,:)-out(nr_output,:)),'g'); hold off;
title('Error for samples before (red) and after (green) learning');

figure(3)
plot(1:length(inp),out(nr_output,:),'.b', ...
1:length(inp),y(nr_output,:),'.r', 1:length(inp),y1(nr_output,:),'.g');
title('Output for samples: given (blue), before learning (red), after
learning (green)')
```

Task to do:

1. Prepare data and find the simplest network realising correctly XOR logic gate.
2. For chosen data files:
   a. look at the data: (how many inputs and outputs they have? what kind of inputs and outputs they have? what are the ranges of inputs and outputs? are we dealing with the task of classification or approximation?),

b. choose the network architecture (number of network inputs and outputs depends on data, choose the number of hidden layers – 1 or 2, choose the number of neurons on each hidden layer, choose the transfer function of neurons on the hidden layer and on the output layer),
c. perform learning, find the error for the learning data and observe the shape of network characteristic.

Prepare the report

For each chosen data create the table:

Network architecture	Activation functions	Error for the learning data

Description of the data files

Files	No of samples	Ino of inputs	No of outputs	Description
data_1D_4 data_1D_6 data_1D_7 data_1D_9		1	1	
data_1D_sin1 data_1D_sin2 data_1D_sin3		1	1	Sine function sampled with different density
data_1D_sin1a data_1D_sin2a data_1D_sin3a		1	1	Noisy sine function sampled with different density
data_2D_1 data_2D_2 data_2D_3 data_2D_4 data_2D_5 data_2D_a		2	1	
data_2D_percep		2	1	
data_2D_elusage		2	1	The average monthly power consumption depending on the month and the average monthly temperature
data_3D_captain	269	3	1	Output: assessment of the degree of danger of the ship made by captain-expert (0, 0.5, 1) Inputs: speeds of the ship
data_5D_parity	32	5	1	Output = 1 if the sum of inputs is even, 0 if not
data_5D_build	4208	5	3	Inputs: hour, output temperature, humidity, sunshine, wind power Outputs: consumption of cold water, hot water and electricity in the building
data_6D_transac	128	6	1	Inputs: amount, company index, hour, type of person realising the transaction, day of week, type of day (free or not) Output: the credibility of transaction in %
data_8D_diabet	768	8	1	Inputs: patients data Output: risk of diabetes
data_9D_glass	214	9	6	Inputs: physico-chemical parameters of glass Outputs (0,1) define the class the glass belons to
data_35D_heart	920	35	1	Inputs: normalised patients data Outputs: risk of heart disease