Back Propagation Neural Networks User Manual

Author: Lukáš Civín
Library: BP_network.dll
Runnable class: NeuralNetStart

Content:

1 Introduction to Back-Propagation multi-layer neural networks

Lots of types of neural networks are used in data mining. One of the most popular types is multi-layer perceptron network and the goal of the manual has is to show how to use this type of network in Knocker data mining application.

Multi-layer perceptron is usually used for classification or prediction methods of data mining. User usually wants the network to split input data into some groups (classify them). There are always some necessary steps in lifetime cycle:

- Preparing data (and also understanding them)
- Preparing network
- Training network
- Classifying

We presume that you have prepared data. This document will help you with preparing and training network to get the best network for classification as possible. After all, data classification is very simple.

The network is trained using Back-propagation algorithm with many parameters, so you can tune your network very well. Remember, you can use only numbers (type of integers, float, double) to train the network! And it is presumed that all data are normalized into <0,1> interval. **The better you prepare your data, the better results you get.**

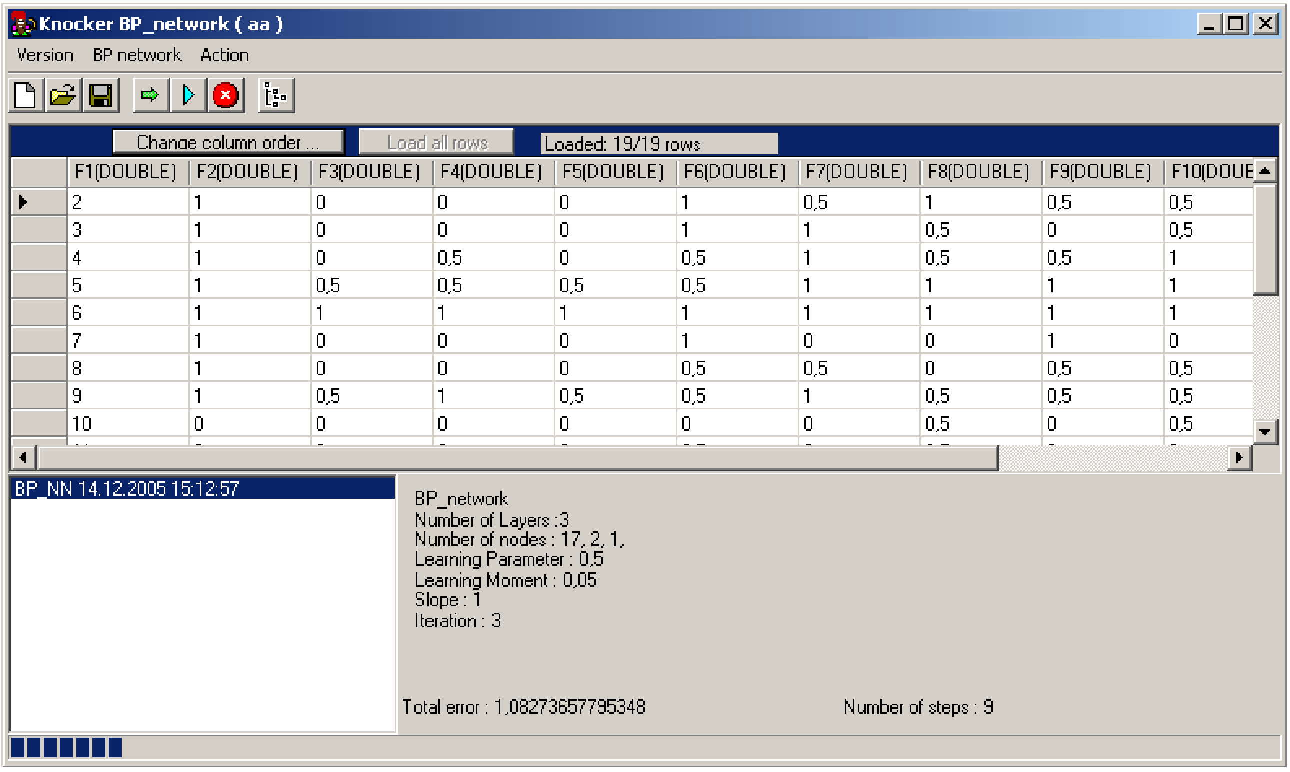
Detailed information about MBA can be found at following links:
http://en.wikipedia.org/wiki/Artificial_neural_network#Multi-layer_perceptron
http://www.tek271.com/articles/neuralNet/IntoToNeuralNets.html
http://cortex.snowseed.com/neural_networks.htm

2 BP network – User interface

This module consists of Main window, visualizing window and some other dialogs.

2.1 Main Window

You can see a menu, tool bar, data-grid and list of prepared networks with their parameters in the main window. You can manage everything (see below for details) from menu. The most important commands from menu are also placed on the tool bar. You can see data of the current version in the data-grid. The list of prepared networks helps you to manage and use more networks together.



	F1(DOUBLE)	F2(DOUBLE)	F3(DOUBLE)	F4(DOUBLE)	F5(DOUBLE)	F6(DOUBLE)	F7(DOUBLE)	F8(DOUBLE)	F9(DOUBLE)	F10(DOUE
▶	2	1	0	0	0	1	0,5	1	0,5	0,5
	3	1	0	0	0	1	1	0,5	0	0,5
	4	1	0	0,5	0	0,5	1	0,5	0,5	1
	5	1	0,5	0,5	0,5	0,5	1	1	1	1
	6	1	1	1	1	1	1	1	1	1
	7	1	0	0	0	1	0	0	1	0
	8	1	0	0	0	0,5	0,5	0	0,5	0,5
	9	1	0,5	1	0,5	0,5	1	0,5	0,5	0,5
	10	0	0	0	0	0	0	0,5	0	0,5

BP_NN 14.12.2005 15:12:57

BP_network
Number of Layers :3
Number of nodes : 17, 2, 1,
Learning Parameter : 0,5
Learning Moment : 0,05
Slope : 1
Iteration : 3

Total error : 1,08273657795348

Number of steps : 9

2.1.1 Main Menu

Main menu has several items. Brief description of all the items is following.

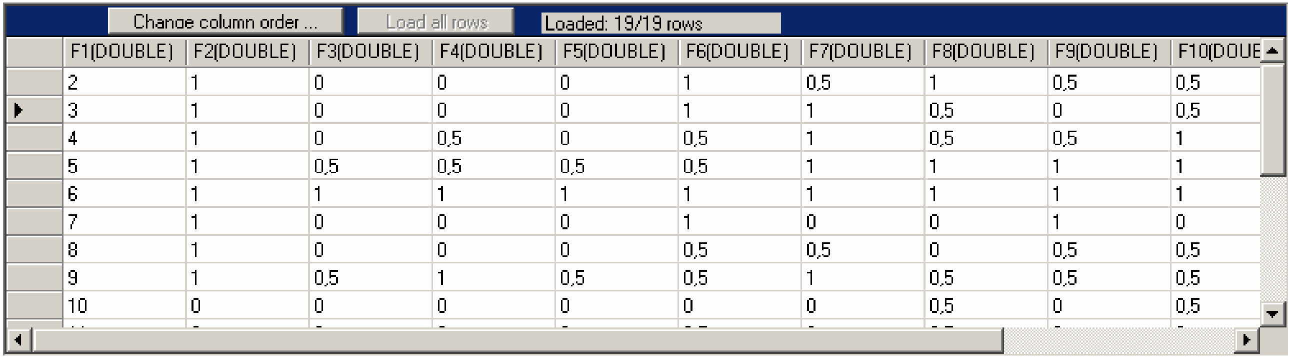
- Version
  - *Get new version* – chooses a new version of data (versions are prepared with Knocker main module) and fills the data-grid with actual data
  - *Select Columns*
    - ✓ *Input Columns* – When you train network, you use only some columns of prepared data as input data. Thus choose columns from version that you want network to use as input values.
    - ✓ *Output Columns* – When you train network, you use some columns as output data. So choose columns from version, that you want network to use as desired output values.
- BP network
  - *New Network... (Ctrl+N)* 🗋 - Starts dialog to create a new BP network.

- *Open Network... (Ctrl+O)* - Opens saved network from a special XML file. This file is in a specific format, please, use only files created by this module.
- *Save Network... (Ctrl+S)* - Saves created network into a special XML file. You have to specify a name and a location of the network.
- *Delete* – Deletes chosen network from the memory (not the file from disk!).
- *Update* – Updates chosen network. Remember, that only certain parameters of the network can be changed.
- *Visualize network* – Opens a new window with the network visualization
- *Error visualizing* – Opens a new window with graph of global errors of the network during the cycles.

- Action
  - *Train (F5)* - Trains the network through given data until stopping parameter is reached.
  - *Step Train (Shift+F5)* - Trains the network through given data only by one cycle.
  - *Cancel Training* - Stops the training of the network.
  - *Classify... (F6)* – Starts a small wizard to classify data with trained network

2.1.2 Data-Grid

You can see current data of chosen version in the data grid. So when module starts, or choose Version> Get new version in the Menu, this grid will be filled with current data values.



Loaded: 19/19 rows

	F1(DOUBLE)	F2(DOUBLE)	F3(DOUBLE)	F4(DOUBLE)	F5(DOUBLE)	F6(DOUBLE)	F7(DOUBLE)	F8(DOUBLE)	F9(DOUBLE)	F10(DOUE
	2	1	0	0	0	1	0,5	1	0,5	0,5
▶	3	1	0	0	0	1	1	0,5	0	0,5
	4	1	0	0,5	0	0,5	1	0,5	0,5	1
	5	1	0,5	0,5	0,5	0,5	1	1	1	1
	6	1	1	1	1	1	1	1	1	1
	7	1	0	0	0	1	0	0	1	0
	8	1	0	0	0	0,5	0,5	0	0,5	0,5
	9	1	0,5	1	0,5	0,5	1	0,5	0,5	0,5
	10	0	0	0	0	0	0	0,5	0	0,5

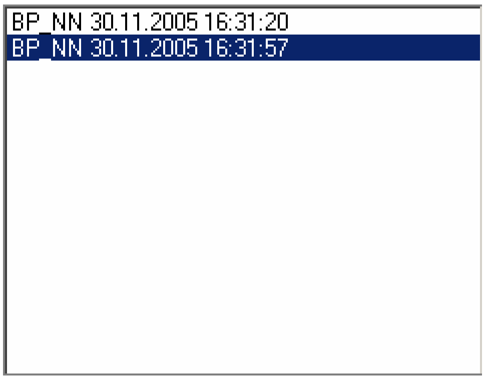
There you can see values of currently selected version. Columns have got same names as in database and the type of values is also mentioned.

Data-grid is read only, so you can't edit data. If you want to, use Main Knocker module.

It is possible to see only some of the columns and also you can change their positions. Use button "Change column order" to start a dialog. There, select the columns you want to see and move them to the right places you want.

2.1.3 List of the networks

You can work with several networks in the same time. It is useful when you try to train networks in different ways. You can compare them very simply after training at one place and find the best one.

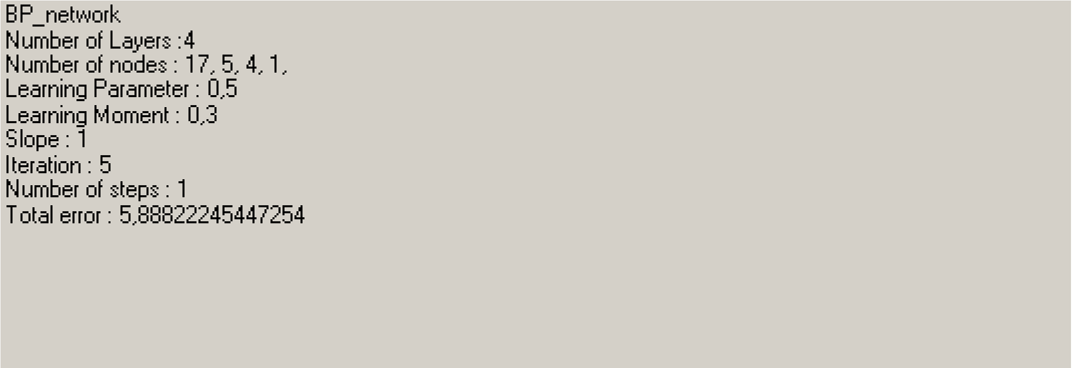


BP_NN 30.11.2005 16:31:20
BP_NN 30.11.2005 16:31:57

You can add a new network into the list by creating a network (*BP network>New Network...*), or loading a network from the file (*BP network>Open Network...).* Date and time is signalized.
The network you choose from the list becomes active and each next operation is affecting this network!

2.1.4 Parameters of Networks

Parameters of the chosen network are shown in this area.



BP_network
Number of Layers :4
Number of nodes : 17, 5, 4, 1,
Learning Parameter : 0,5
Learning Moment : 0,3
Slope : 1
Iteration : 5
Number of steps : 1
Total error : 5,88822245447254

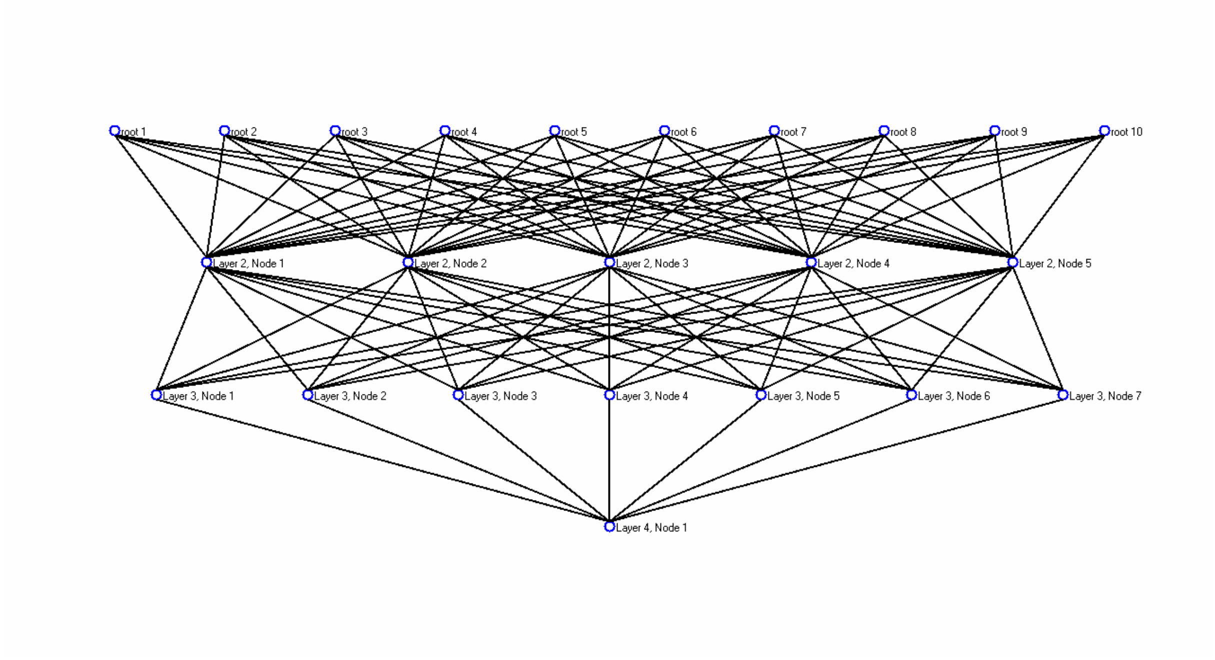
Most of parameters are explained in the chapter "Creating a network".
Number of steps – Shows how many cycles have already been done through given data during the training.
Total error – Shows current error of the network.

2.1.5 Progress bar

Sometimes it is easy to show you progress of the network training (when you are using halting parameter "*number of steps*" or "*timeout*").

2.2 Creating a new network

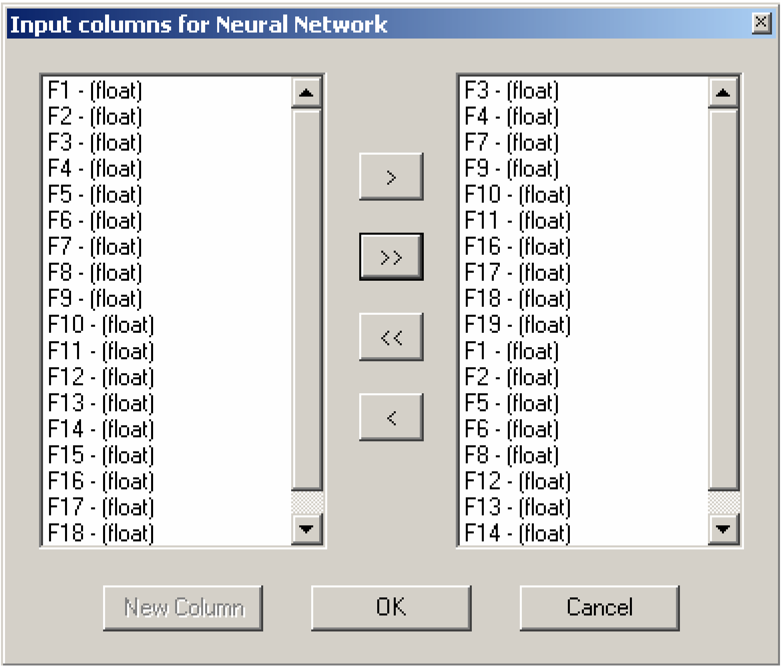
We will go through all steps you have to do to create a new multi-layer perceptron network in this chapter. New network could look similar to this one:



2.2.1 Defining input and output columns

Our network is a set of layers (with nodes and edges). You do not need to estimate number of input and output nodes of your network, because it depends on the given data! So, the first and the last layer have the number of nodes according to the number of chosen columns of your data version. But we have not set input and output columns for the network yet, so we do it now.

- ***Input columns*** **–** values in these columns are generally well known both in training data and in classifying data. You must know which columns you plan to use as input for classification. In Menu: Version>Select Columns>Input Columns. This dialog will appear:



F1 - (float)
F2 - (float)
F3 - (float)
F4 - (float)
F5 - (float)
F6 - (float)
F7 - (float)
F8 - (float)
F9 - (float)
F10 - (float)
F11 - (float)
F12 - (float)
F13 - (float)
F14 - (float)
F15 - (float)
F16 - (float)
F17 - (float)
F18 - (float)

F3 - (float)
F4 - (float)
F7 - (float)
F9 - (float)
F10 - (float)
F11 - (float)
F16 - (float)
F17 - (float)
F18 - (float)
F19 - (float)
F1 - (float)
F2 - (float)
F5 - (float)
F6 - (float)
F8 - (float)
F12 - (float)
F13 - (float)
F14 - (float)

You can see all columns of the current version in the left rectangle. The right one tells you which columns will be used as input columns.

Use:

> - To copy one selected column from source into chosen input columns

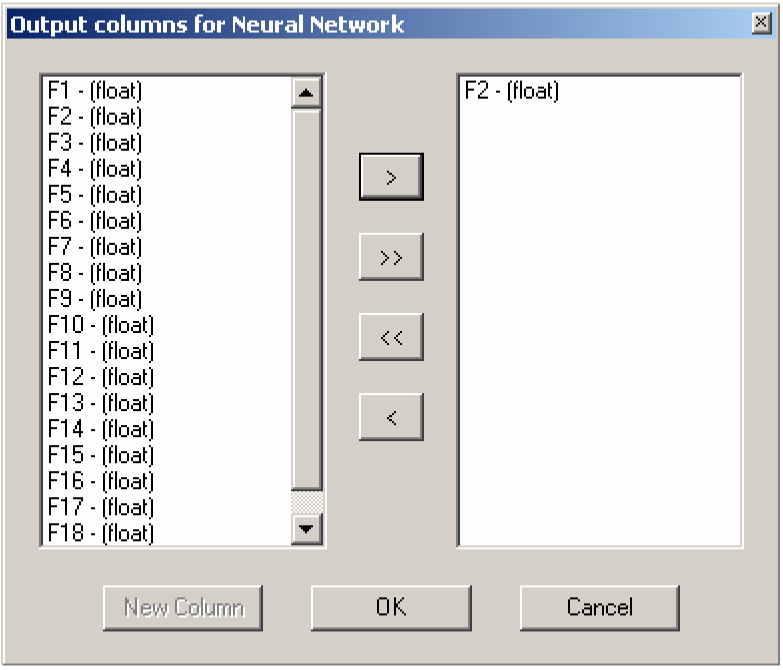
>> - To copy all items from the source into chosen input columns

<< - To move all items from chosen input columns into the list of all columns

< - To move one selected item from chosen input columns into the source

When you have selected all input columns, press *"OK"* and your selection will be saved. When you press *"Cancel"*, your selection will not be saved.

- **Output columns** - values in these columns are generally not well known. We know them only in training data, but we want to find them in classification. These values are usually the "goal" values you want to reach, so it is easy to select them in the dialog (way of selection is almost the same as for input columns), Menu: Version> Select Columns>Output Columns:



Output columns for Neural Network

F1 - (float)
F2 - (float)
F3 - (float)
F4 - (float)
F5 - (float)
F6 - (float)
F7 - (float)
F8 - (float)
F9 - (float)
F10 - (float)
F11 - (float)
F12 - (float)
F13 - (float)
F14 - (float)
F15 - (float)
F16 - (float)
F17 - (float)
F18 - (float)

F2 - (float)

You can see all columns of the current version in the left rectangle. The right one tells you which columns will be used as output columns. Use the buttons in the same way as those described in the section about input columns.

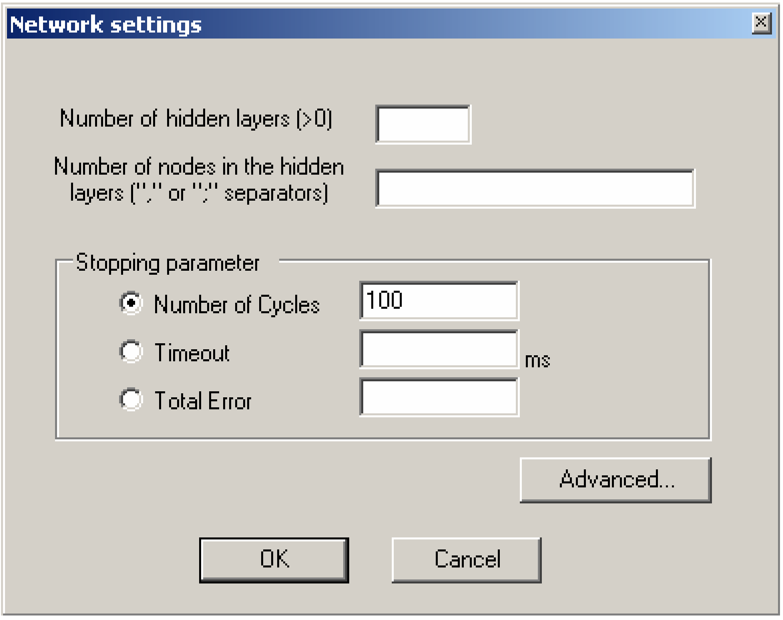
Tip: You can use the same columns for input and output. In that case you train identity.

When you have selected all output columns, press *"OK"* and your selection will be saved. When you press *"Cancel"*, your selection will be lost.

2.2.2 Defining a new network

Every network has some parameters that are very important for the successful network classification. We can't give you a cookbook which parameters are the best, because you will have to identify them according to the situation.

You can start dialog for creating network from toolbar (🗋) or from Menu: BP Network>New Network... or by pressing shortcut Ctrl+N.



Network settings

Number of hidden layers (>0)

Number of nodes in the hidden layers ("," or ";" separators)

Stopping parameter

● Number of Cycles 100

○ Timeout ms

○ Total Error

Initially, the dialog is almost empty. You have to fill some parameters before the net can be created.

- *Number of hidden layers* – Every multi-layer-perceptron network has input and output layer and we have already defined them (previous chapter). But if we want to create an "intelligent" network, it will be necessary to have at least one hidden layer. So, fill in the number of hidden layers.

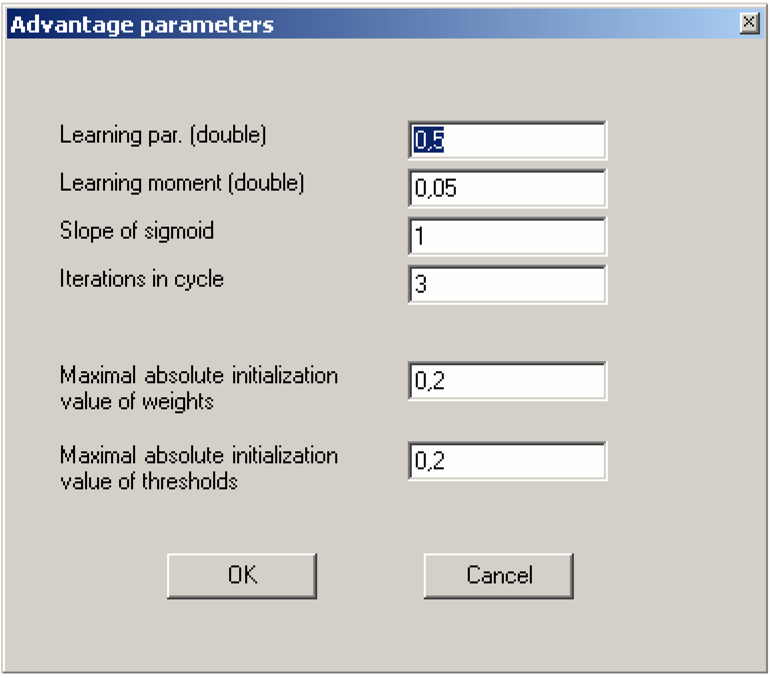
  Tip: do not try to make a network with more than 2 hidden layers, it doesn't bring better results and training is then very difficult.

- *Number of nodes in the hidden layers* – This is one of the best places for experiments with networks. According to the number of layers, fill in the number of nodes for each layer. When you have more than one hidden layer, use "," or ";" separators between numbers.

  Example: you filled in 2 hidden layers, so in this field you can insert "7;10" nodes

- *Stopping parameter* – Network training must finish sometimes. Be careful when you are editing this parameter, because it could take a very long time before stopping parameter is reached on huge amount of data. Just one stopping parameter is always used according to the selected radio-button.
  - *Number of Cycles* – "teacher" is going through all given data for several **steps** (also cycles) during the training. When you have checked this option, network will read the data **k**-times (**k** is the number inserted) – be careful with huge amount of data!
  - *Timeout* – Training will stop after this amount of millisecond (do not use this when you are going to step training)
  - *Total Error* – Critical parameter of the network is its total error. It is simply difference between desired output and real output. So, training will stop, when Total Error of the network is lower than this value (in %) – be careful with unstable training patterns!
- *Advanced button* – If you want to experiment more with your network, you can set some "advanced" parameters of the network. But you don't have to.

  After clicking on the button, new dialog appears:



Advantage parameters

Learning par. (double) 0,5

Learning moment (double) 0,05

Slope of sigmoid 1

Iterations in cycle 3

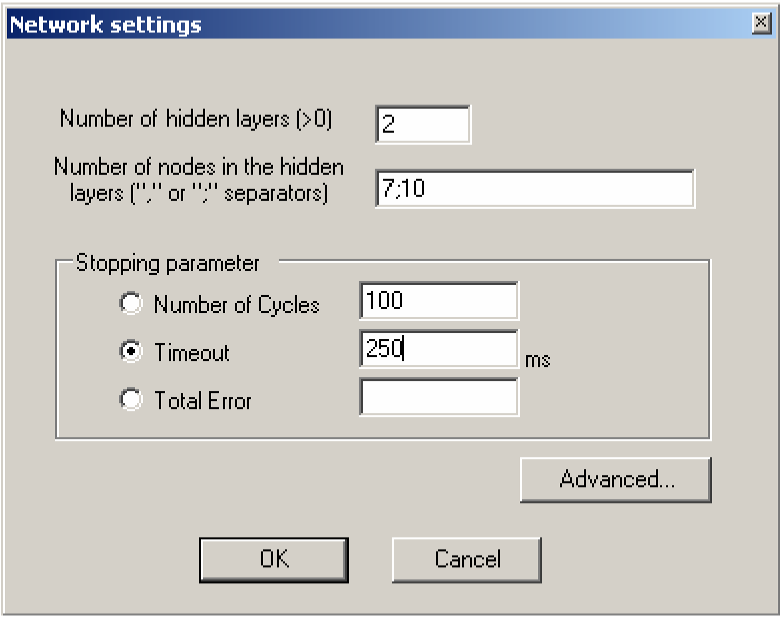
Maximal absolute initialization value of weights 0,2

Maximal absolute initialization value of thresholds 0,2

- *Learning parameter* –This parameter is a learning parameter in the formula for back-propagation training – it says, how big the modification step will be
- *Learning moment* - This parameter is a moment parameter in the formula for back-propagation training – it says, how strong the training keeps its direction against the gradient of error function
- *Slope of sigmoid* - This parameter is slope of sigmoid in a formula for back-propagation training – it says, how steep the sigmoid function will be
- *Iterations in cycle* – it says, how many times the network will be trained on one pattern during one cycle (if there are 100 patterns, network will take every pattern **k**-times)
- *Maximal absolute initialization value of weights* – The weights of edges (between nodes in layers) have to be initialized. This value is the maximum initial weight for each edge.
- *Maximal absolute initialization value of thresholds* – The thresholds of nodes have to be initialized. This value is the maximum initial threshold for each node.

When you have selected all output columns, press "*OK*" and your selection will be saved. When you press "*Cancel*", your selection will be lost.

You will be back in the first "basic dialog".



Network settings

Number of hidden layers (>0) 2

Number of nodes in the hidden layers ("," or ";" separators) 7;10

Stopping parameter

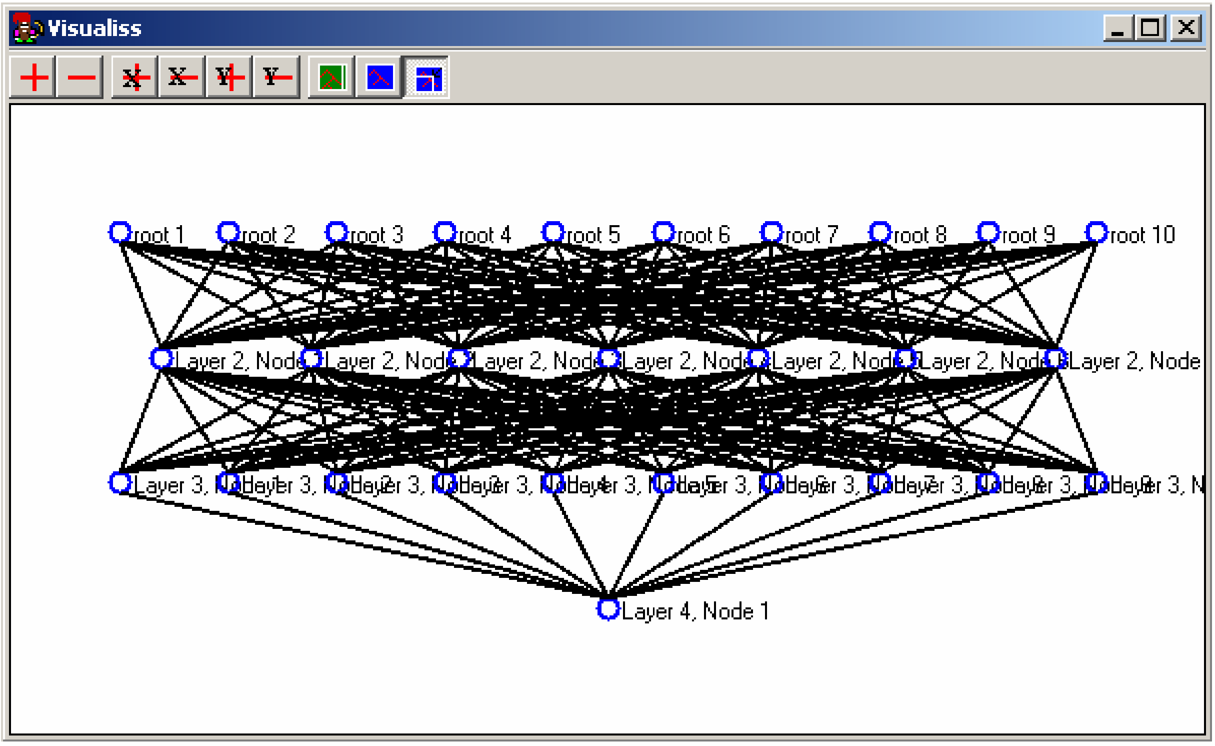
○ Number of Cycles 100

● Timeout 250 ms

○ Total Error

- *OK button* – program controls given parameters and creates a new BP network

After this definition, the visualization window will appear according to the created network. You can find more information about this window in a separate chapter named “Visualization”.



This window is for your information.

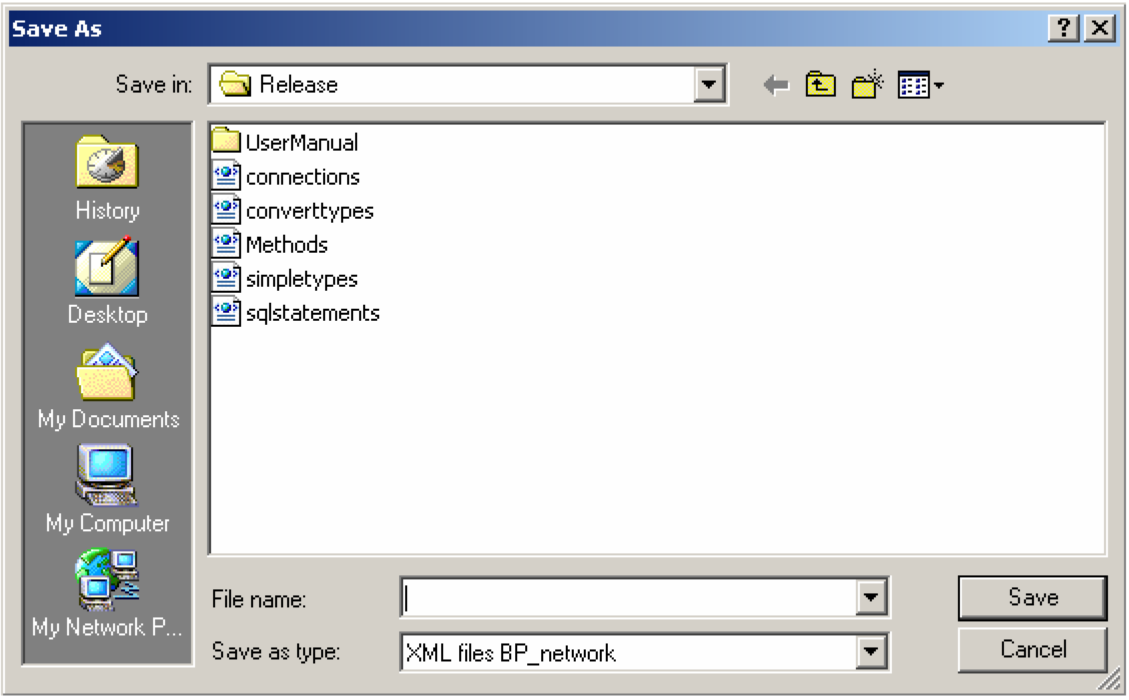
2.3 Saving network to file

You can create a network, train it, use for classification, but after all, sometimes you want to save it into a file. You can archive your networks and later use them again.

The network is saved into a special XML file. If you want to edit this XML file, be very careful! It has a special format and unauthorized editing could bring wrong or impossible loading (opening) network.

Before you save the network, you have to choose the right network from the list of the networks (you can see its parameters to the left from the list). Then choose Menu: BP network>Save network or shortcut Ctrl+S or button.

Common dialog appears:


Save As

Save in: Release

UserManual

connections

converttypes

Methods

simpletypes

sqlstatements

File name:

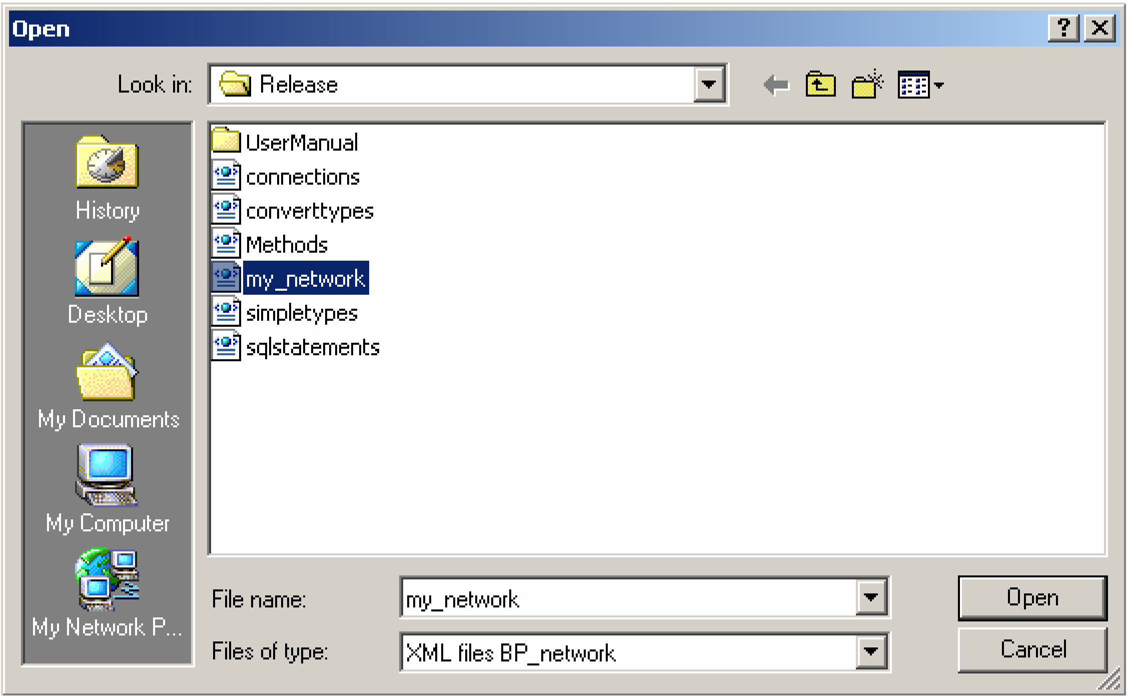
Save as type: XML files BP_network

You are asked for a location and name of your network. If you press button “Save”, your network will be saved immediately.

2.4 Opening Network from file

If you have some networks already saved, you can open them and load them into a memory. **Be sure, that you are loading only networks created by this module!**

Selecting Menu: BP network>Open Network or shortcut Ctrl+O or button, then appears well known dialog:



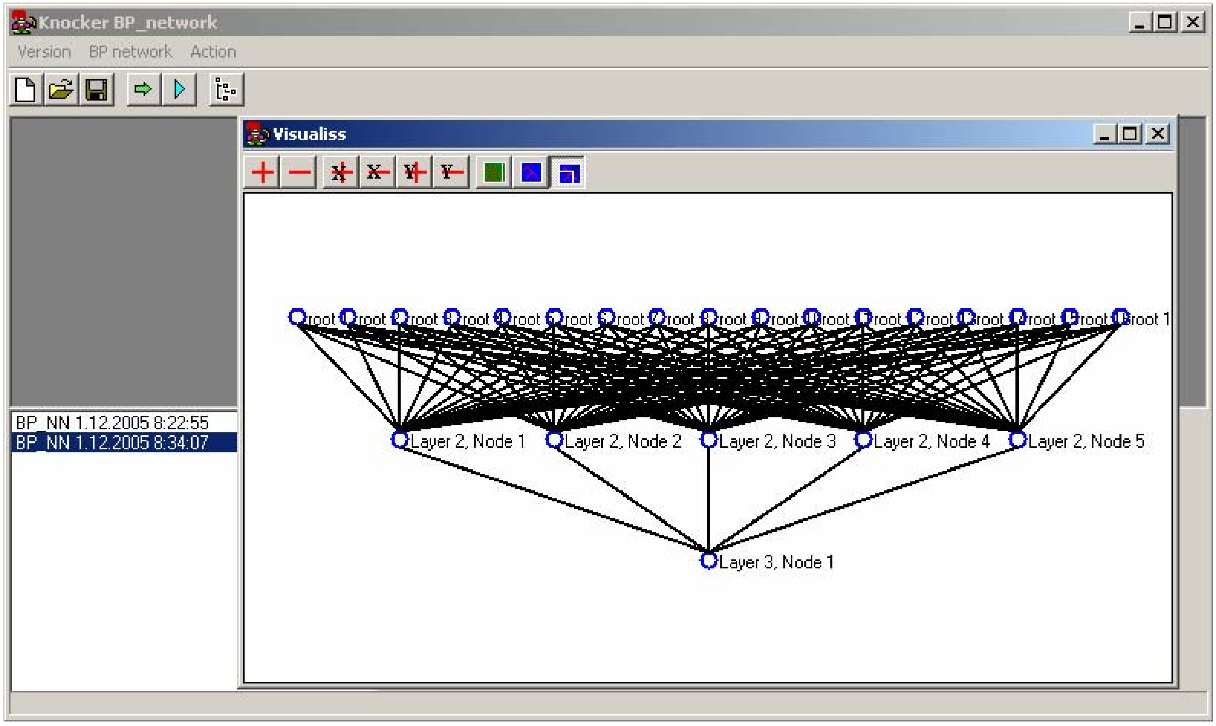
Look in: Release

UserManual
connections
converttypes
Methods
my_network
simpletypes
sqlstatements

File name: my_network

Files of type: XML files BP_network

Find your saved network and select it. Program tries to load this network into the memory after clicking "Open" button. This network then appears both in a visualizing window and in the list of networks as you can see (the second network from the list on the left was just loaded):



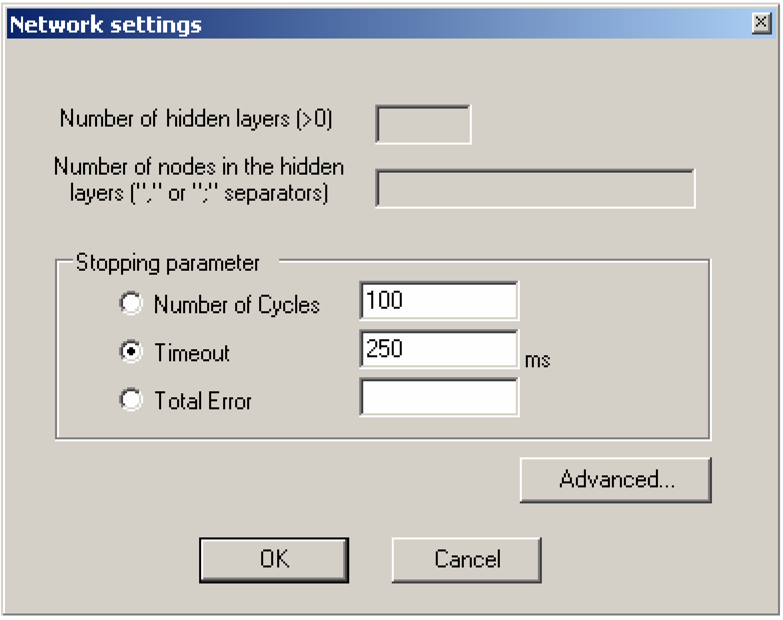
BP_NN 1.12.2005 8:22:55
BP_NN 1.12.2005 8:34:07

2.5 Deleting Network from memory

Every created network is saved in a memory and you can delete it. Just select it in the list of networks and choose Delete in Menu: BP network>Delete.

2.6 Updating network in memory

After creating, loading, training network, you can even more update some parameters. After selecting the network in the list choose Menu: BP network>Update. Already mentioned dialog from creating network appears:



Network settings

Number of hidden layers (>0)

Number of nodes in the hidden layers ("," or ";" separators)

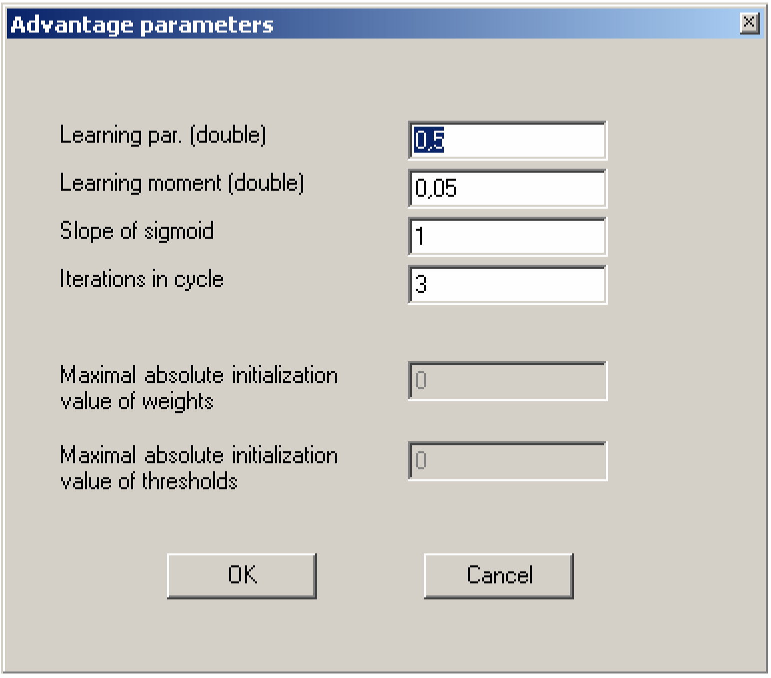
Stopping parameter

○ Number of Cycles 100

● Timeout 250 ms

○ Total Error

You can see that only some of the parameters can be changed. Some advanced parameters can be updated too:



Advantage parameters

Learning par. (double) 0,5

Learning moment (double) 0,05

Slope of sigmoid 1

Iterations in cycle 3

Maximal absolute initialization value of weights 0

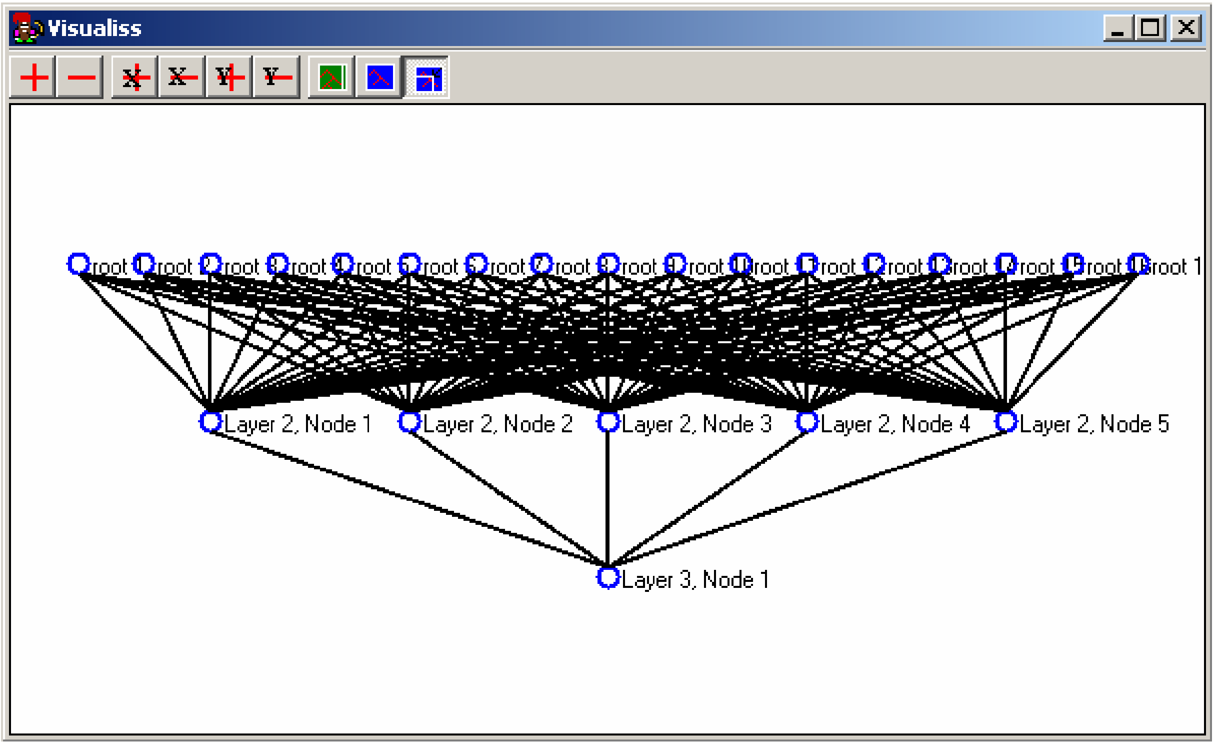
Maximal absolute initialization value of thresholds 0

Again, you have to click on OK button to save your changes.

2.7 Visualizing network

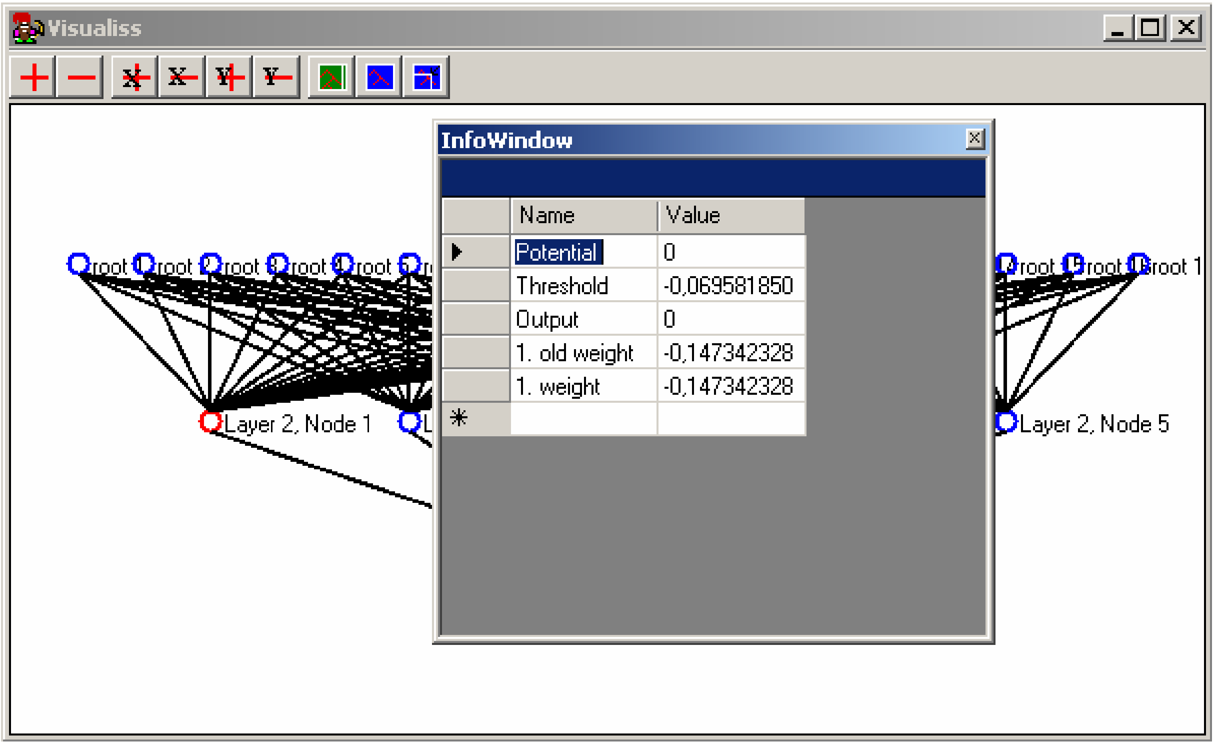
If you want to study your network, you should have idea about its structure. We have already seen a window with visualized network, but we haven't spoken about it yet. We are going to describe the structure now.

In Menu: BP network>Visualize network or clicking on button. A special window appears.



It has two parts:

- *Toolbar* – helps you working with the visualized network more comfortable
  - o + − Zoom in and Zoom out the network
  - o X+ X− Y+ Y− Zoom in/out axis independently
  - o Set exact space of each node in the pixels
  - o Fit the tree to the window
  - o Fit the tree to the window always when resizing the window. Toggle button, default is pushed
- *Network view* – shows the structure of currently selected network. **Be careful, input nodes are at the top part of the view!** (Named root...). You can see input node(s), nodes of hidden layer(s) and output node(s). There are edges between them too. When you click on the edge, additional information about this node and its edges appears. Example:

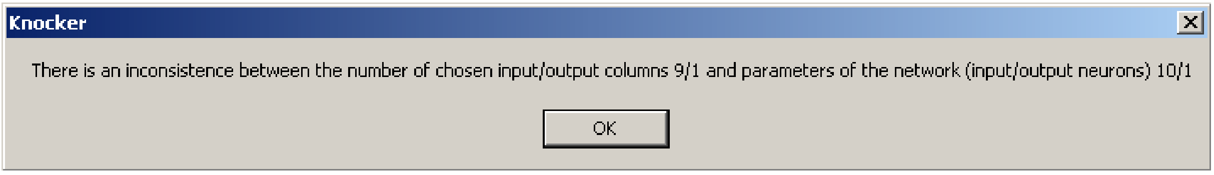


Name	Value
Potential	0
Threshold	-0,069581850
Output	0
1. old weight	-0,147342328
1. weight	-0,147342328

Selected node has a red color now. Try clicking on other nodes to see their parameters.

2.8 Training network

You can train a selected (you have to select one!) network from the list to get resulting network. Be careful while working with several networks with different input and output. You have to correct the number of input/output columns of data of current network according to number of input/output nodes before training.



Knocker

There is an inconsistence between the number of chosen input/output columns 9/1 and parameters of the network (input/output neurons) 10/1

Example: You have defined a first network with 3/6 (I/O) nodes, second network with 4/6 nodes. If you want to train the first network, you will have to select input/output columns again, because the network has 3 nodes, but there are still 4 input columns selected. Columns selection was described in chapter "*Creating Network*".

You can train network in two ways:

- *Train until stopping parameter is reached* – The network will be trained through given data, until the stopping parameter (which was defined during the phase of creating network) is fulfilled.

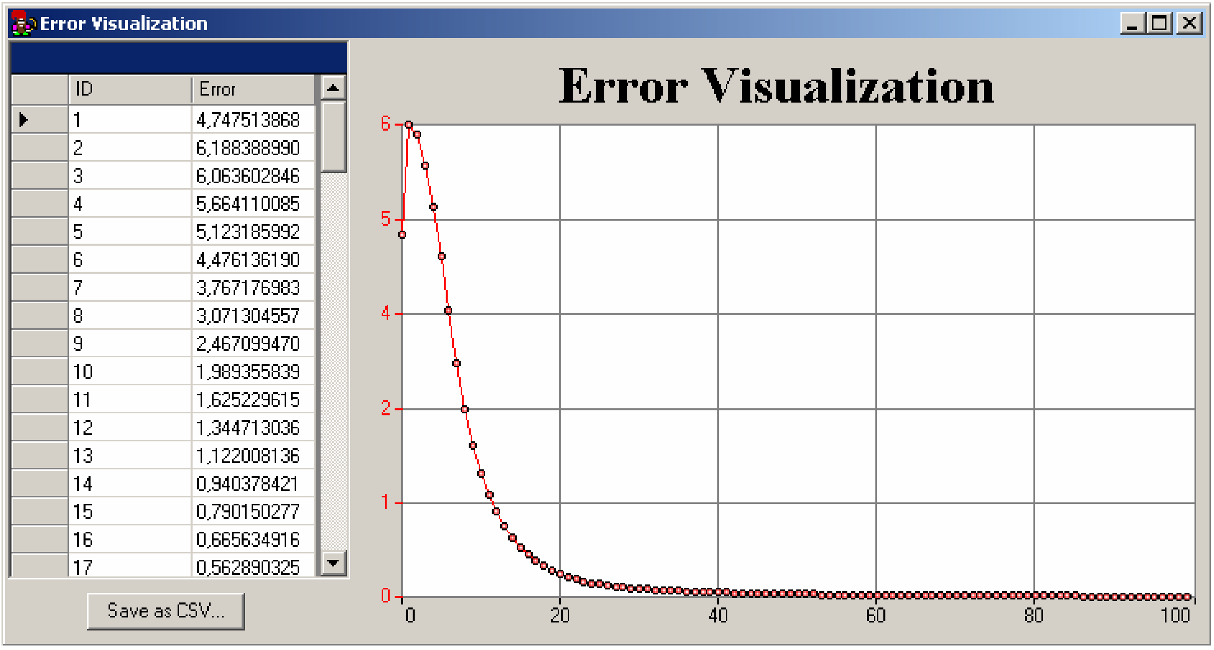
  You can train from Menu: Action>Train or by shortcut F5 or by pressing the button ▷

  When you start this training, you can stop it untimely with button ✖ (Menu Action>Stop Training).
- *Step training* – You can step training, so it stops after each cycle through data. It is useful especially when you want to check error of the network, weights of the edges and thresholds after each step.
  You can step training from Menu: Action>Step Train or by shortcut Shift+F5 or by pressing the button ➡

2.9 Error Visualization

One of the most important values of the network is its global error (energetic function). It is often very useful to monitor it. Global error should reach 0, but sometimes, when network is not able to train some patterns, it is above 0. To decrease it, try to restart training with another parameters (learning par., learning moment, number of cycles...).

It is possible to visualize the trend of error function during the training cycles of selected network in this module. In Menu: BP network>Error Visualization, a window appears (in this example – after 100 cycles):

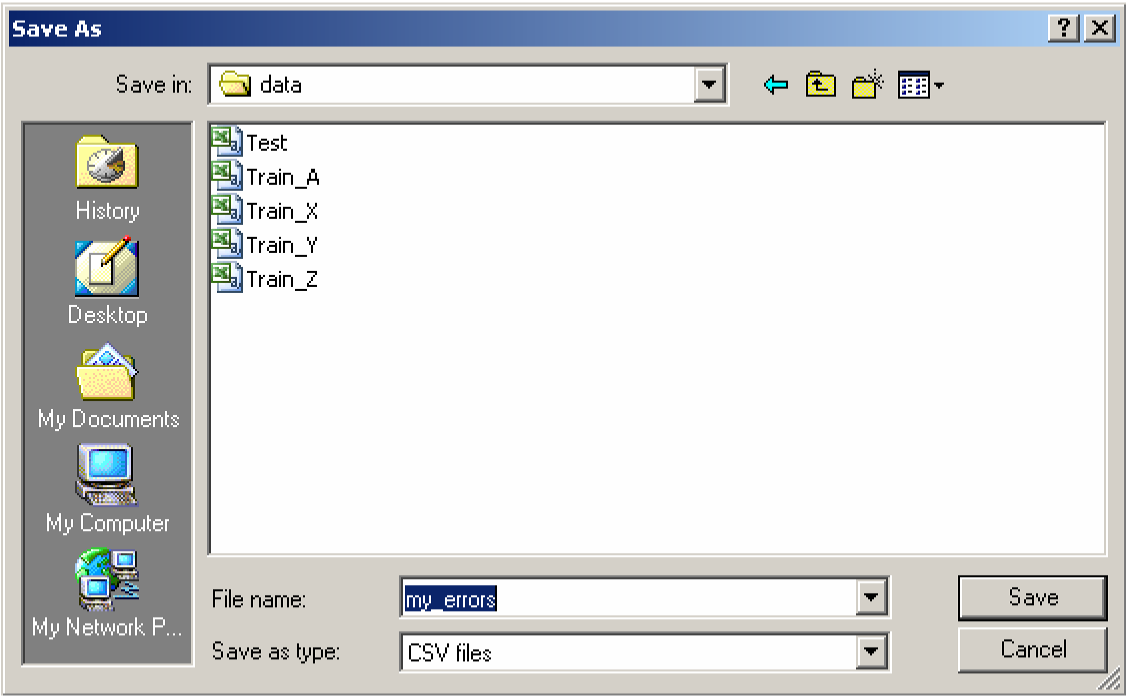


ID	Error
1	4,747513868
2	6,188388990
3	6,063602846
4	5,664110085
5	5,123185992
6	4,476136190
7	3,767176983
8	3,071304557
9	2,467099470
10	1,989355839
11	1,625229615
12	1,344713036
13	1,122008136
14	0,940378421
15	0,790150277
16	0,665634916
17	0,562890325

There are two logical parts:

- *Data-grid* – informs about the error in each cycle. It is possible to see exact value of the error function. Scroll down to see all values.
- *Graph* – Shows the values from data-grid

It is possible to save values into the "csv" file (MS EXCEL can open it). By clicking button "Save as CSV..." save dialog appears:



File name: my_errors

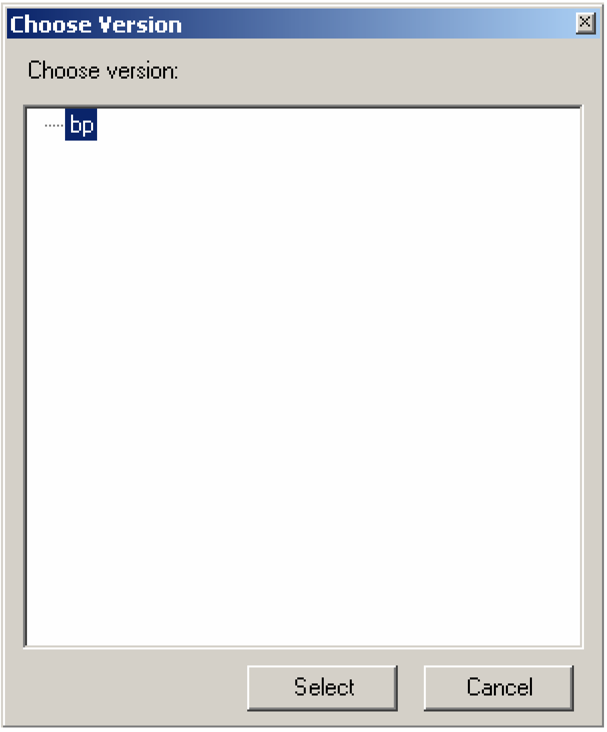
Save as type: CSV files

After clicking on “Save” a new file is created with error data and you can work with them later (e.g. in excel).

2.10 Getting a new version of data

More than one version of data can be prepared in the main Knocker module. We usually want to use them all without closing Neural Networks module.

In Menu: Version>Get New Version, you will see a dialog for choosing a new version of data. All versions already created are here.



The highlighted version will be selected as actual version after clicking on “Select” button.

Be careful, with new data. Sometimes is necessary to redefine your input and output columns (if the data has another structure) and some networks wouldn’t work correctly with new versions.

Now you can work with new version of data.

Tip: Prepare your training data as one version and data for classification as second version. When you train your network, you can easily switch to the second version and you can classify data immediately.

2.11 Data classification

Classification is the last step and after all the work, it is like a prize. There are some steps which are necessary for successful classification:

- Trained network (the network is created or loaded and trained)
- Data for classification (usually special version of data)

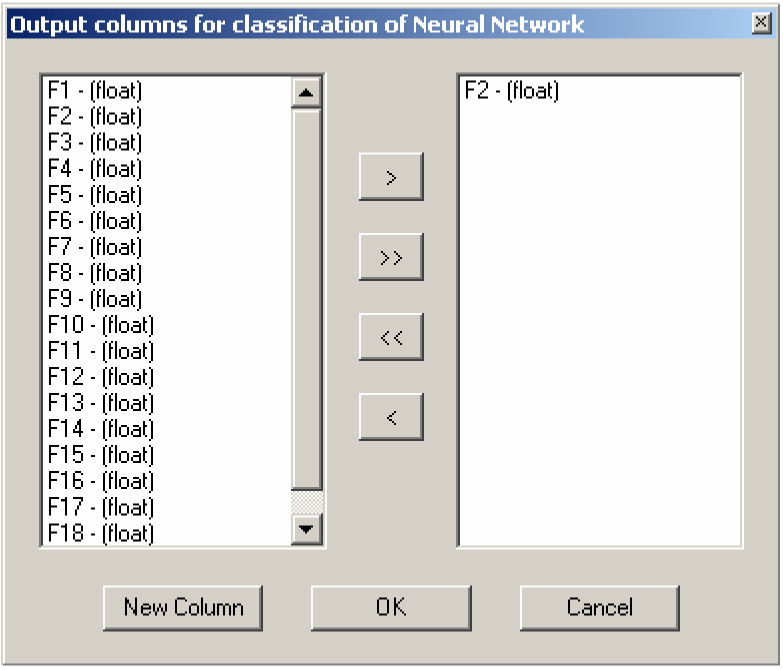
Classification is solved as a "wizard". You have to go through some dialogs before your data are classified. If you press "Cancel" button, all steps will be lost. "OK" button is a gate to next dialog.
In Menu: choose Action>Classify or use shortcut F6 to start classification sequence.

2.11.1 Selecting input columns

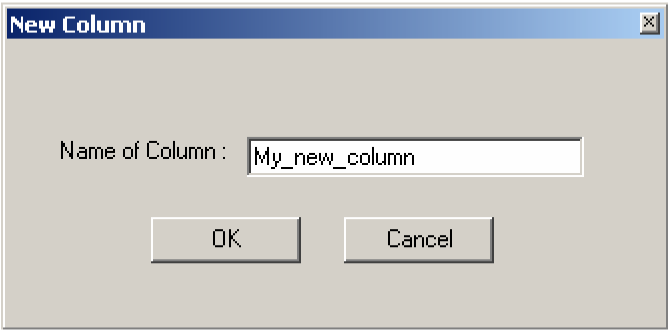
Be sure that you have selected correct columns for classification. Number of input columns should be the same as number of input nodes. This dialog was described in chapter 2.2.1

2.11.2 Selecting output columns

This dialog is a little bit different from the dialog in chapter 2.2.1 because it has active button "New Column".

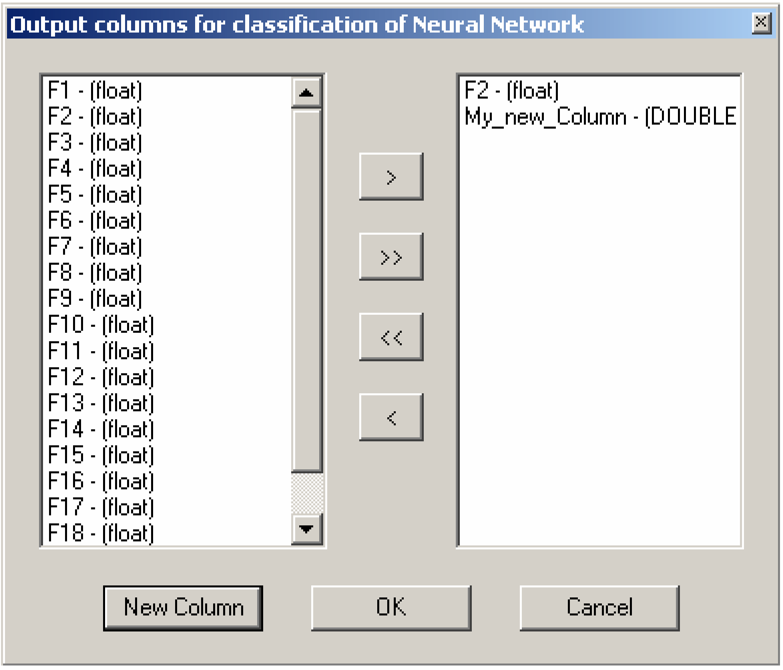


When you don't have "output" column prepared, you can do it by clicking on this button. Additional dialog appears:



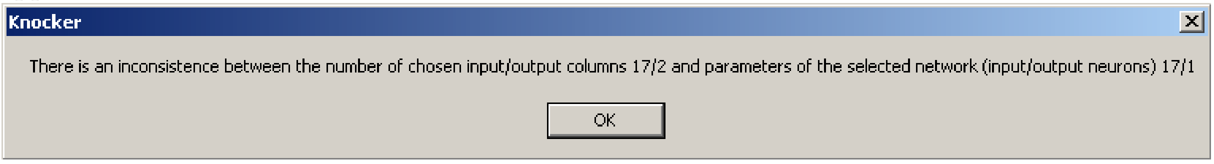
Name of Column : My_new_column

Insert the name of the new column. This column is type of double and it will be added to the new table with classified data. Click "OK" to add this column into your set of columns.



Be careful! The number of output columns should be the same as the number of output nodes in trained network.

If we have just one output node, we have to correct the selection. If we didn't correct it, error message would appear.



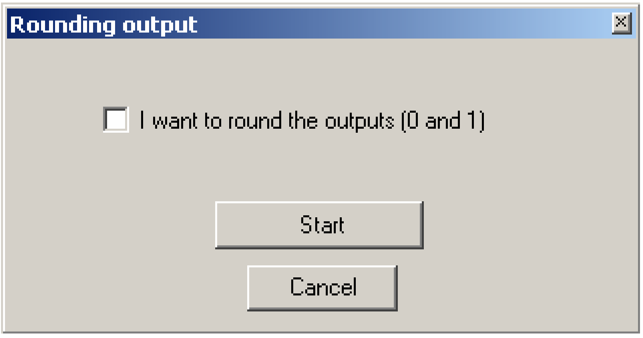
There is an inconsistence between the number of chosen input/output columns 17/2 and parameters of the selected network (input/output neurons) 17/1

If both selections are correct, next dialog will appear.

Be careful! If you choose the same column for input and output, initial value of the column will be overwritten with the new, value of classification.

2.11.3 Rounding the output

The output from the network is always between 0 and 1. Sometimes, output should be 0 **or** 1. In such case, check the check box.

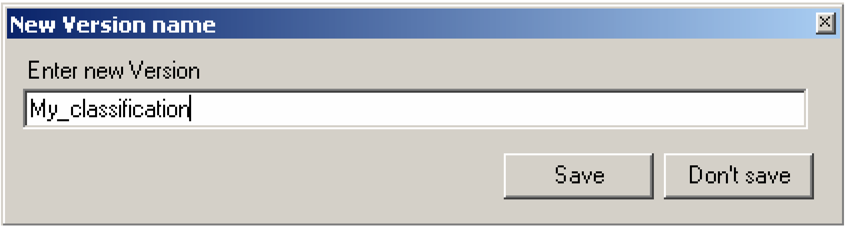


□ I want to round the outputs (0 and 1)

After clicking Start, everything is prepared to start classification.

2.11.4 Name of the new version

Classification inserts new values into the version; so new version must be created. Fill in its name. Its creation is the last step before classification.

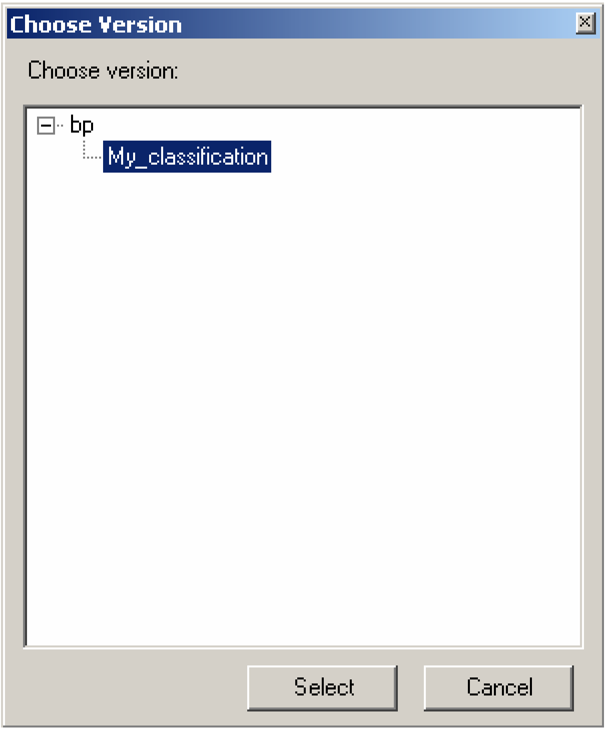


Enter new Version

My_classification

Click “Save” and classification starts.

Tip: If you want to see your new classified data then select your new version in “Menu: Version->Get New Version”:



Choose version:

- bp
  - My_classification

You will see your new data in the data-grid.

3 Neural Network – Tutorials

There will be described one common task for back-propagation neural networks in this chapter – identity(7). We would like our network to learn all patterns – all 7-bit binary numbers (input is a 7-bit binary number and the output should be the same 7-bit binary number).

3.1 Creating a new network

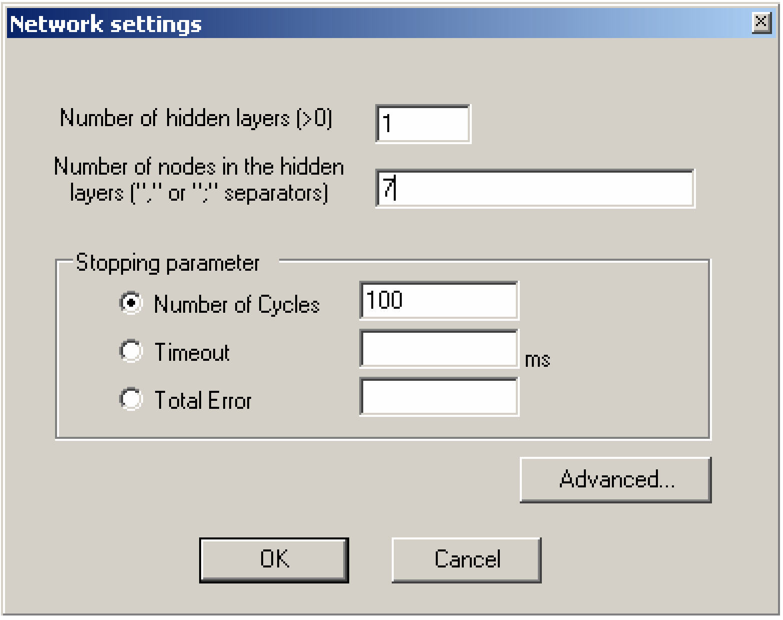
We have already prepared the version with data. It is available as Identity7.csv

1. **Selecting input/output columns**

As it is described in chapter 2.2.1, choose all 7 columns as input columns and then all 7 columns as output columns.

2. **Creating a new network**

As it is described in chapter 2.2.2, create a new network. You can leave all parameters default, set only one layer and 7 nodes in this layer as it is shown in the screenshot and click "OK":



Network settings

Number of hidden layers (>0): 1

Number of nodes in the hidden layers ("," or ";" separators): 7

Stopping parameter

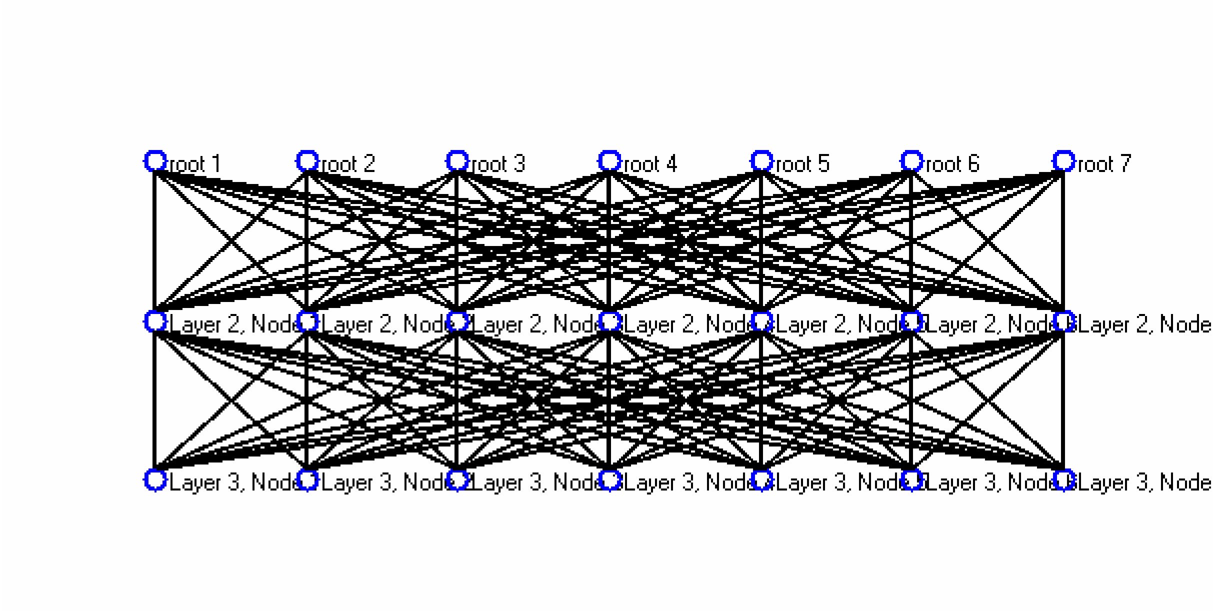
● Number of Cycles: 100

○ Timeout: ms

○ Total Error:

3.2 Training network

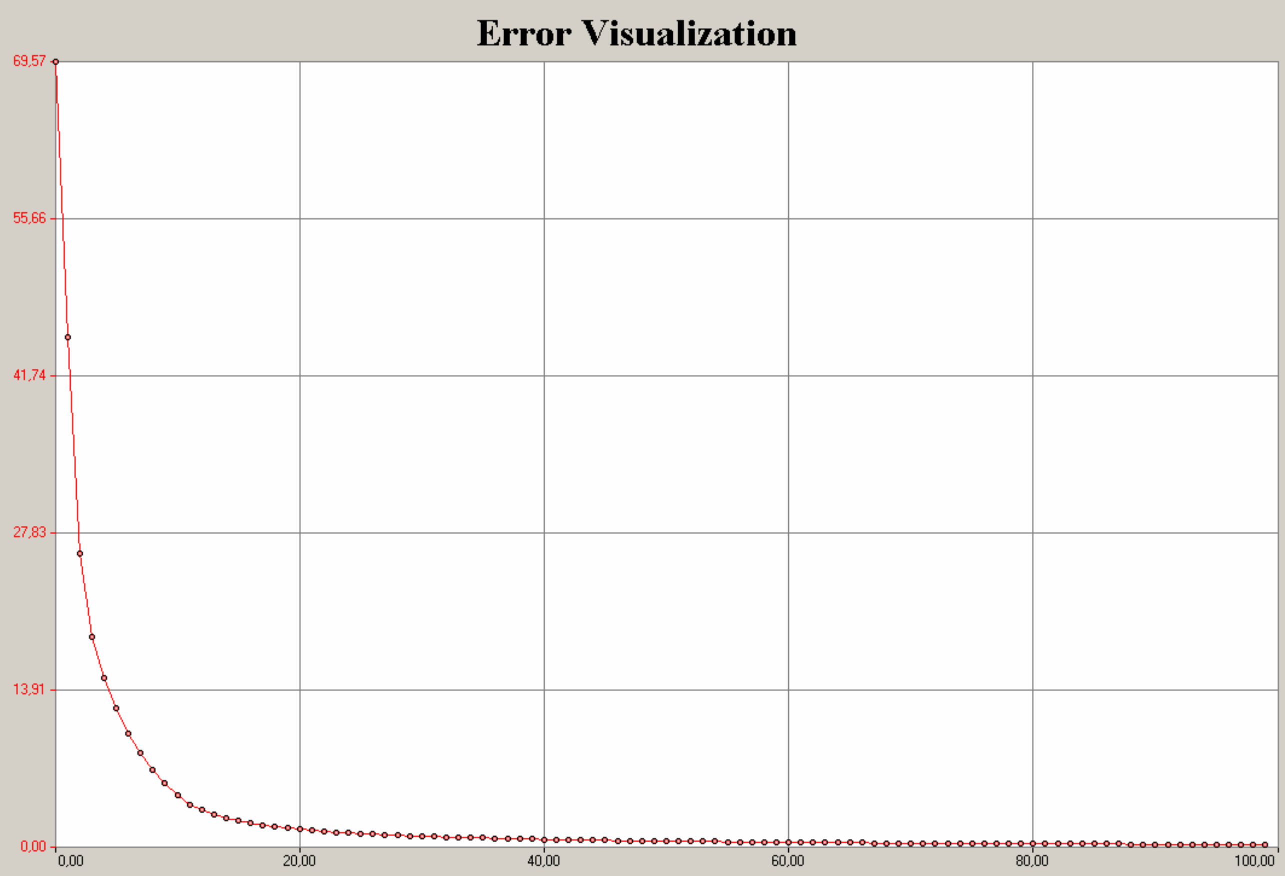
Now there is a network available and it should be learned.



There are two ways, how to train the network. It is described in chapter 2.8. 100 cycles is enough for this type of network. We start clicking ▷ button. Progress bar rises immediately.

3.3 Error visualization

After the training, we can study the error function, as it is described in chapter 2.9. Error function looks like this:

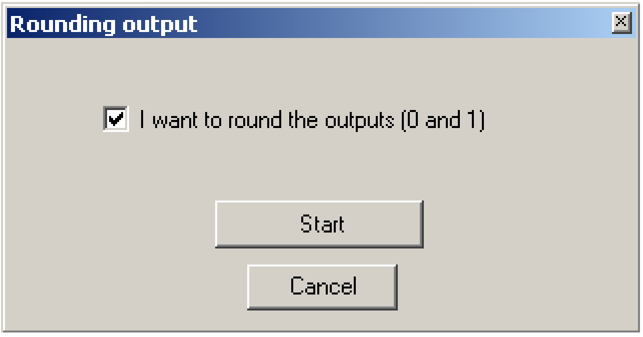


We can see, that this network has learned the numbers very well.

Tip: Try it with only 6 nodes in the hidden layer and you will see that it is impossible to train the network so well.

3.4 Classification

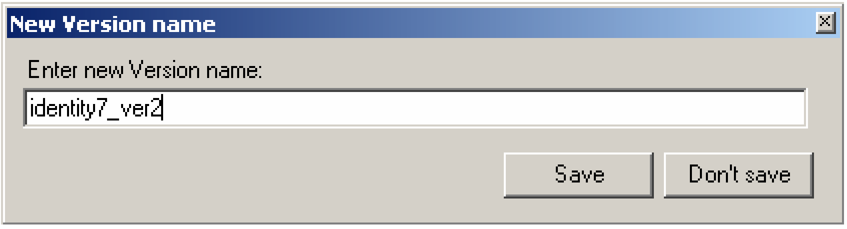
In the classification, we would like our network to show us, what it has learned. We use the source table as destination table, so after the classification, we would like to have a table with all 7-bit binary numbers again.
A process of classification was described in chapter **Error! Reference source not found.**.
At the beginning (2.11.1), we will choose all 7 columns as input columns. Next step (2.11.2), we will choose the same – all 7 columns, because we want to overwrite the old ones.
In the next dialog, we show our will, that output should be 0 or 1 (2.11.3) and click start:



Rounding output

☑ I want to round the outputs (0 and 1)

The last step is only to insert the name of the new version that will be created (2.11.4).



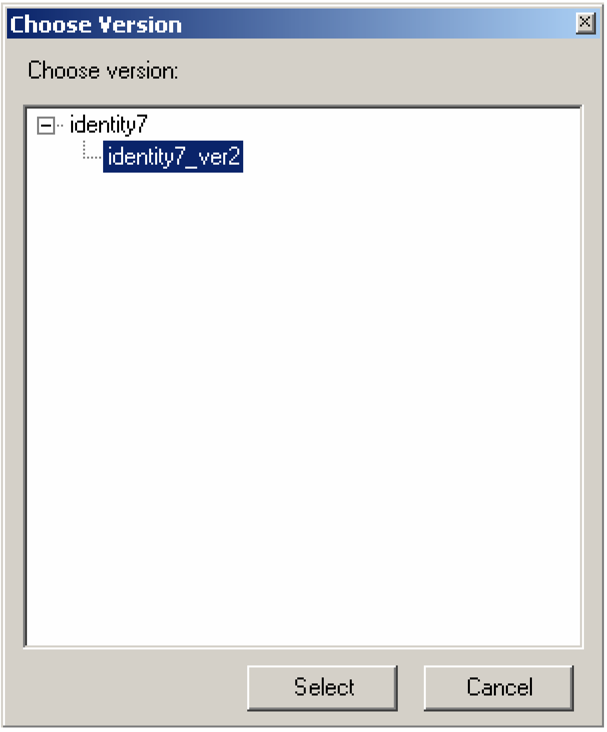
New Version name

Enter new Version name: identity7_ver2

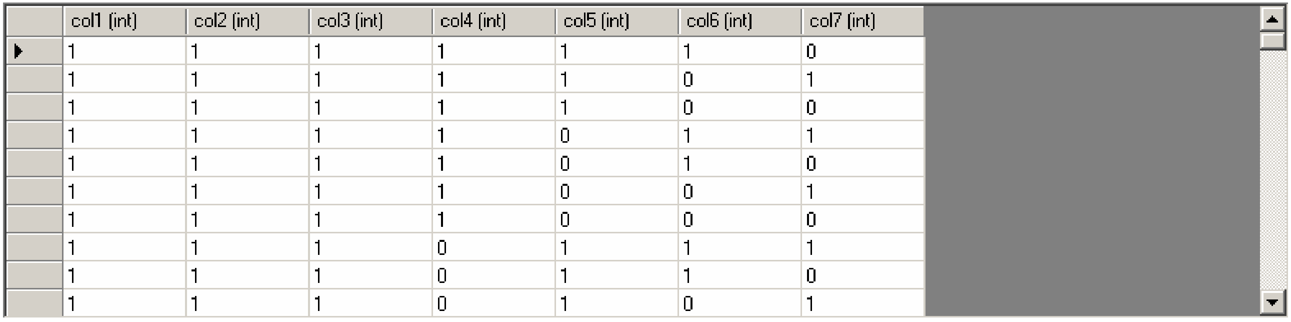
A classification starts.

3.5 Where can you see the result?

After the classification, choose Menu: Version>Get New Version and similar window appears:



A new version is already there. Select it and the new data will be shown in the data-grid.



	col1 (int)	col2 (int)	col3 (int)	col4 (int)	col5 (int)	col6 (int)	col7 (int)
▶	1	1	1	1	1	1	0
	1	1	1	1	1	0	1
	1	1	1	1	1	0	0
	1	1	1	1	0	1	1
	1	1	1	1	0	1	0
	1	1	1	1	0	0	1
	1	1	1	1	0	0	0
	1	1	1	0	1	1	1
	1	1	1	0	1	1	0
	1	1	1	0	1	0	1

4 Requirements

Necessary components for correct running of this module:

- all common components of main application Knocker
- BP_network.dll
- DMTransformStruct.dll
- PtreeViewer.dll
- GuiExt.dll
- Gui.dll
- DasNetBarChart.dll

The main runnable class is NeuralNetStart in BP_network.dll.

5 Samples

You will find some interesting sample data in CSV (Knocker friendly) format as a part of the distribution.

- identity7.csv (data from the tutorial, not-data mining, but common exercise for BP network)
- neuralnet.csv (the first column is the primary key, the second is the output)
- it is possible to find large datasets in http://www.kdnuggets.com/datasets/