INEB- PSI Technical Report 2006-2

Neural network software tool development in C#

Alexandra Oliveira

aao@fe.up.pt

Supervisor:

Professor Joaquim Marques de Sá

December 2006

INEB - Instituto de Engenharia Biomédica

FEUP/DEEC, Rua Dr. Roberto Frias, 4200-645 PORTO

Contents

Introduction **2**

C# and Microsoft Visual C# 2005 Express Edition **3**

The neural network software tool **6**

First version 6

Second version 10

Bibliography **19**

Introduction

One of our main objectives is to create a software tool with a friendly graphical interface to implement neural networks. This program must have a icon-driven graphical interface that allows the user to specify a block diagram representation of a neural network.The block diagram is composed of linkable icons, representing different components of a neural network (inputs, neural network architecture, learning algorithm, display outputs,....) chosen from a library. As new features, this software must include the algorithms developed by the researchers involved in the ENTNETs project.

The language chosen to develop the tool was C#. C# is an object-oriented programming language developed by Microsoft as part of their .NET initiative [2]. The IDE (integrated development environment) is Microsoft Visual C# 2005 Express Edition.

We started by familiarizing ourselves with C# and it's IDE and by the end of the second semester year 2006, we have built a neural network software tool that implements a multilayer perceptron (MLP) with two different learning algorithms and using mean squared error as cost function, for classification problems with two classes.

C# and Microsoft Visual C# 2005 Express Edition

The Microsoft Visual C# Express Edition is an IDE (integrated development environment) that allows a quick and easy creation of Windows applications (see figure 1).

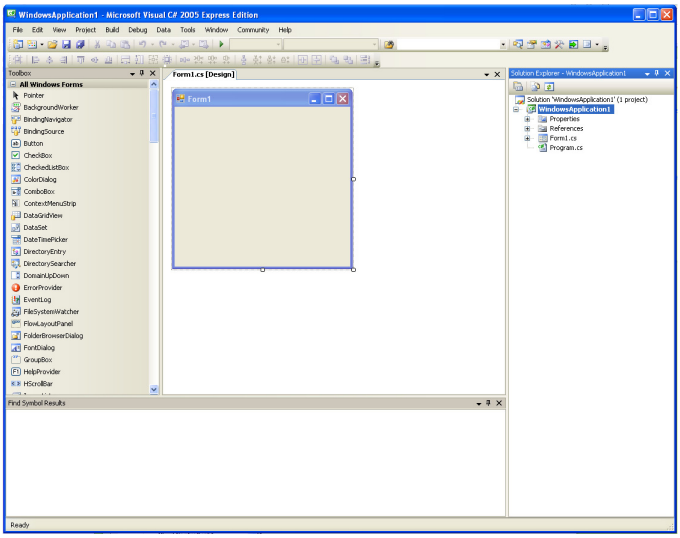

Figure 1: Microsoft Visual C# Express Edition

The toolbox contains drag-and-drop controls and components to create a Windows applications. Controls are grouped into logically-named categories like Menus and Toolbars, Data, Common Dialogs and more. To add controls to Windows Forms we just have to click on the control and drag the control onto the form[4]. The events and properties of the components can be changed in the Property Window, shown in figure 2. The events are listed in the menu represented by the “lightning” and new events can be added by double-clicking on the name of the event.

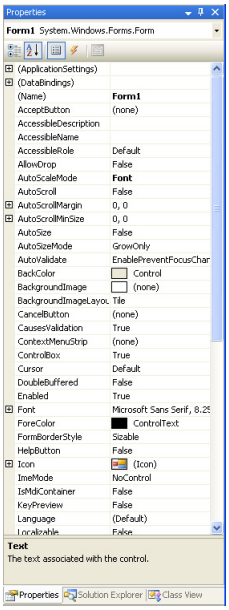

Figure 2: Properties window

The programming language used in Microsoft Visual C# 2005 Express Edition is C#. This programming language is an object-oriented programming language developed by Microsoft.

The ECMA standard lists these design goals for C# [2]:

- C# is intended to be a simple, modern, general-proposed, object-oriented programming language.
- The language, and implementations there of, should provide support for software engineering principles such as strong type checking, array bounds checking, detection of attempts to use uninitialized variables, and automatic garbage collection. Software robustness, durability, and programmer productivity are important.
- The language is intended for use in developing software components suitable for deployment in distributed environments.
- Source code portability is very important, as it programmer portability.
- Support for internationalization is very important.
- C# is intended to be suitable for writing applications for both hosted and embedded systems, ranging from the very large that use sophisticated operating systems, down to the very small having dedicated functions.

- Although C# applications are intended to be economical with regards to memory and processing power requirements, the language was not intended to complete directly on performance and size with C or assembly language.

Therefore, we decided to use Microsoft Visual C# to build the new neural network software tool.

The neural network software tool

After choosing the programming language we started by constructing a library with the basic mathematical support for a neural network. At the same time, we explored C# language as well as Microsoft Visual C# Express Edition features. During the study of C#, we constructed some graphical controls that were included in the first version of the software tool.

In September, appeared a very useful neural network library written in C#, so we investigated this library. With a solid knowledge of the code we have rewritten and integrated these functionalities in the fist version.

First version

The first version of the written software included a set of basic routines with the purpose of aiding the construction of neural networks algorithms. These routines were grouped into a library and consist of the following functions:

1. Matrix arithmetics (addition, subtraction, product);
2. Matrix partitioning;
3. Matrix sorting of rows or columns;
4. Transpose matrix;
5. Elementary operations on rows or columns of matrices (finding the minimum/maximum element and the position of it, adding or deleting rows, altered some rows elements);
6. Matrix normalization;
7. Vector dot product;
8. Initializing matrices.

9. Classification matrix;

10. Matrix of initial random weights.

The first version of the graphical interface is shown in the figure 3.

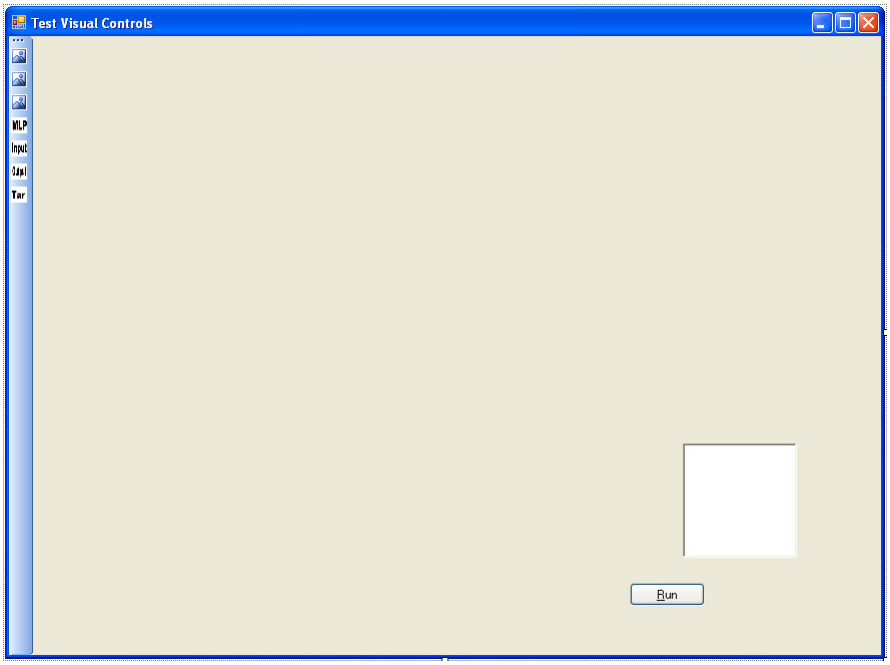

Figure 3: First version of the graphical interface

In the tool strip menu we created four buttons that add new controls to the window application. The first and last buttons create controls related to input data and targets data, respectively. These controls are shown in the figure 4.

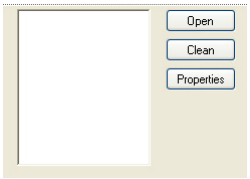

Figure 4: Data control

The Open button of this control opens a data file, where variables are separated by a semi coma and patterns are separated by new line. The button Clear, deletes the data written in the text box. The button Properties creates a new control that is shown in 5.

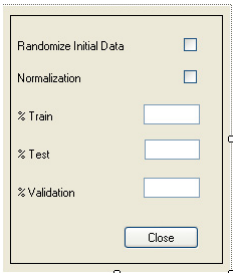

Figure 5: Data properties control

The first check box allows the user to shuffle the patterns of the initial data. The Normalization check box allows the user to scale the initial data to the range chosen by him. The limits of the range appears in a control added when the user checks the Normalization box. The %Train, %Test and the %Validation text boxes allow the use to crate a partitioning of the data matrix.

The MLP Button of the tool strip menu creates a control related to an MLP neural network. This control is shown in 6.

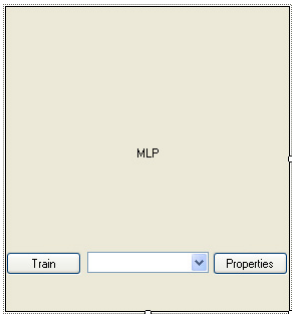

Figure 6: MLP control

The button train creates the control show in 7.

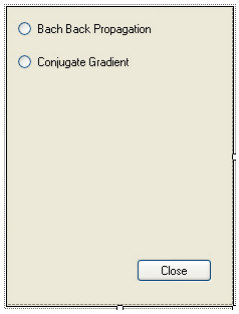

Figure 7: MLP learning algorithm control

The first check box, when selected, opens a control, shown in 8, where the user can choose the back propagation algorithm parameters.

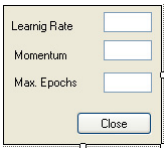

Figure 8: Backpropagation algorithm parameters control

The second check box is not working.

The combo box of the control MLP is used to choose the cost function that, in this version, is just the mean squared error (mse). Finally the Properties button creates the control shown in 9.

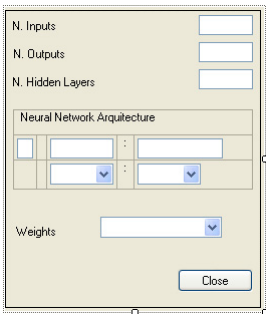

Figure 9: MLP properties control

In this control the user can choose the neural network architecture: the number of input and outputs are fixed so one can only choose the number of hidden layer and the number of neuron in each hidden layer. Having chosen the number of layers, the user can choose the activation function of each layer as well as the initial weights for the neuron. However, this last functionality is not working.

Finally, the Output tool strip menu button just creates a control to display the outputs of the neural network. This control is shown in the figure 10.

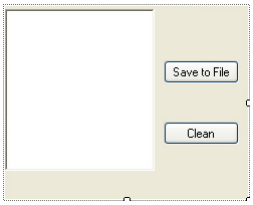

Figure 10: Output control

During this work appeared a useful neural network library written in C# , so we started to explore it and rewrite it so that the software had the mathematical and graphical functionalities aimed by the group. As a result of this work the second version of the software was developed based on the first version and on the mentioned library.

Second version

Franck Fleury made a C# library very useful for the program the we are developing [1]. However, this library presents some mathematical limitations and the graphical interface is not at all what we desired. Anyway, this library served as an inspiration for the second version of the software.

Fleury's interface is shown in the figure 11.

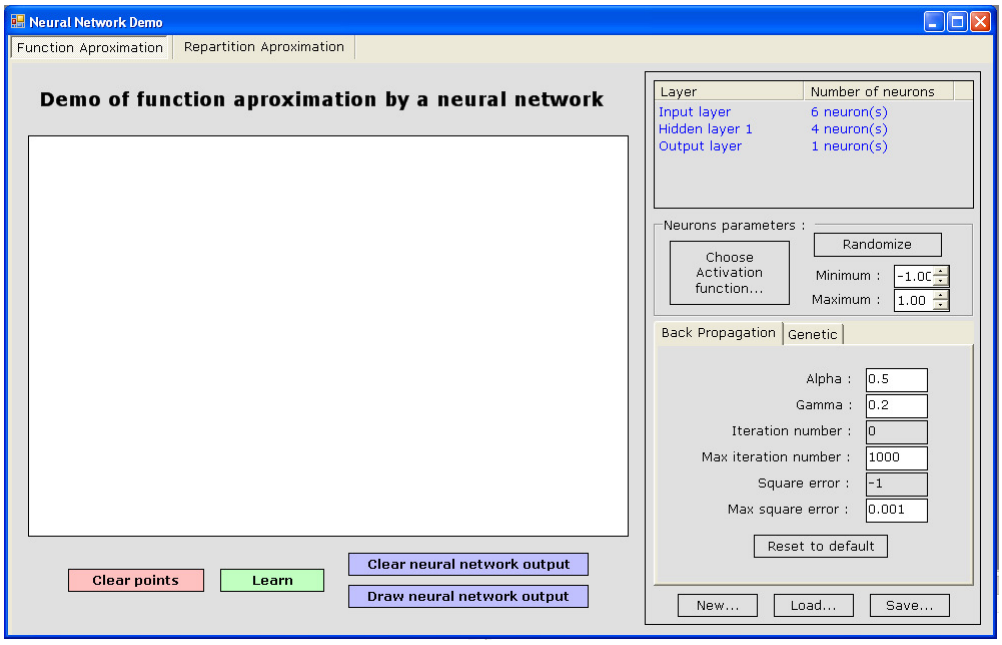

Figure 11: Frank Fleury graphical interface for implementing neural networks

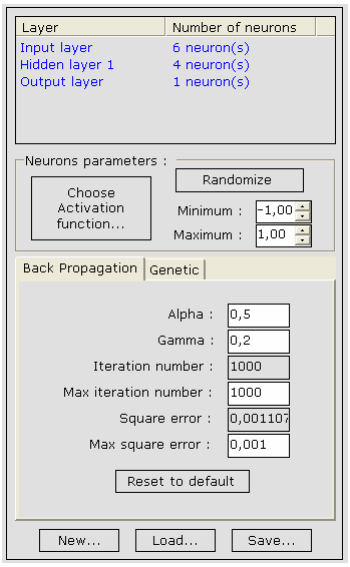

Figure 12: Neural network editor control

In the panel shown in 12 the user can design a neural network. The top of the panel is about network structure; we can see here 3 layers of neurons (processing units with sigmoid activation function). By clicking on a particular layer, we can edit the properties of that layer, namely the activation function and the synaptic weights of the layer or of one particular neuron of that layer. This can be done in the control shown in 13.

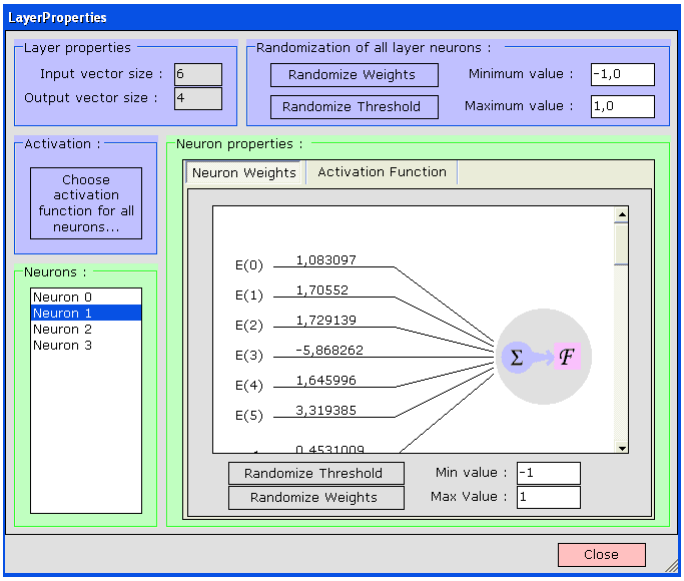

Figure 13: Layer editor control

Under the table that displays the neural network architecture, we can adjust neuron properties for the whole network (activation function of all neurons and randomize all weights). The next panel is about the learning algorithm. Two algorithms are implemented and we can choose which one we want to use by clicking on the tab and adjust it's parameters. At the bottom of the panel we have the ability to perform "new", "load" and "save" operations of the neural network [1]. The activation functions can be chosen on the panel shown in 14.

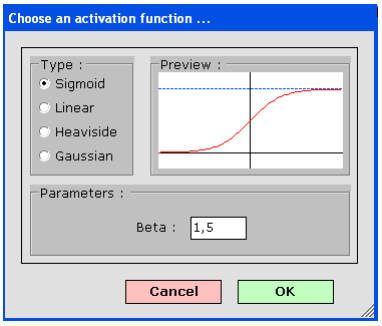

Figure 14: Activation function selecting control

The "new neural network" frame lets we create his/her own structure of neural network and is shown in the figure 15.

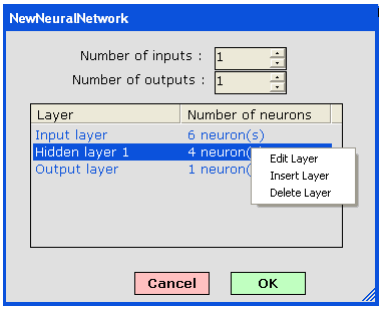

Figure 15: New neural network control

The features on this neural network library that are more important to our software tool are described below:

1. Ability to solve problems of function approximation and classification;
2. The classification problem is a problem of separating two classes of points;
3. The data is inputted to the neural network by clicking in a white panel (in classification problems this point has different colors that are displayed if the user clicks on the right or left mouse buttons);
4. Four activation functions:
   - Sigmoid;
   - Linear;
   - Heaviside;
   - Gaussiana.
5. All weights are initialized, by default, to zero;
6. The initial weights can be altered by generating random values in the interval $[-1, 1]$. This interval is set by default but can be altered by the user;
7. The network is fully connected;
8. By default, the activation function is Sigmoid and this activation function can be different for each neuron or each layer;
9. The first layer is a processing units;

10. The default learning algorithm is the sequential back propagation;

11. By default, the error threshold is 0.001 and the maximum epochs is 1000;

12. The learning rate 0.5 and the momemtum is 0.2 are the values by default for the back propagation algorithm parameters.

Most of these features were maintained but the most part of the graphical interface was rewritten. As the second version of the neural network was built based on the Fleury's library we will now describe the main differences between the two software tools. First of all the new software as the window form shown in 16.

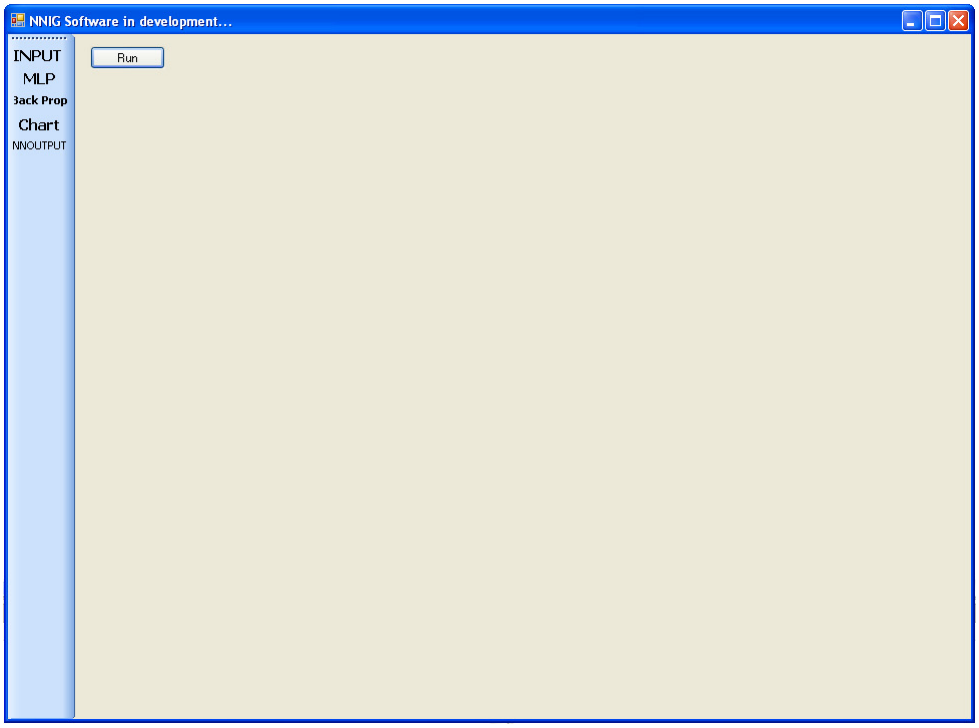

Figure 16: Second version window form

This first tool strip button creates the control shown in 17.

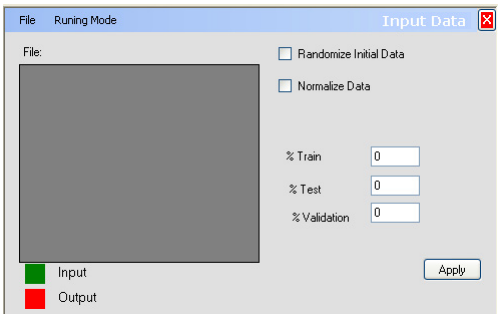

Figure 17: Data control

The input data most be written in the same format as in the prevision version and the behavior of the Randomize Initial Data button was maintained. By default, the last data column is the target column and the others are set as input. This default option can be altered trough a context menu (see figure 18).

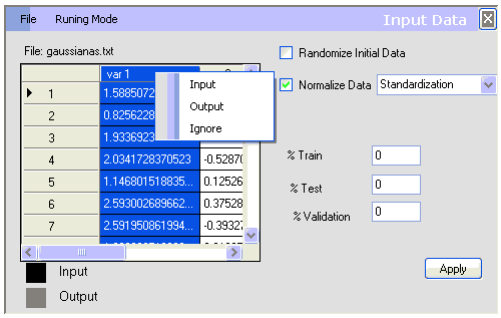

Figure 18: Data control showing the context menu and one pre-processing data methods

Two pre-processing data methods were added: mean centering and standardization. These options are displayed in the combo box that appeare when the user activates the Normalize Data check box.

The neural network editor was reformulated and is shown in picture 19.

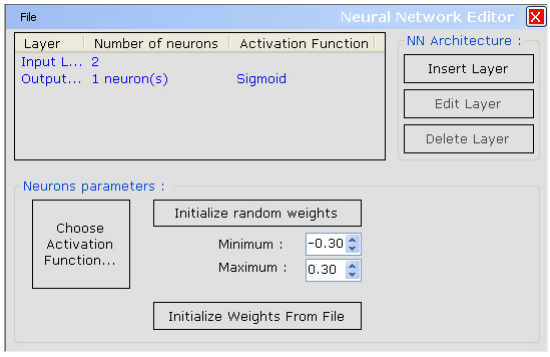

Figure 19: Neural network editor control

Now, the first layer is the input layer. The neurons of this layer are just the variables of the neural network, so the neurons of this layer are not processing units. The default neural network that appears is set according to choices made by the user for input and outputs. One can add layers with a chosen number of neurons and can delete some hidden layers. All layers, except the input layer, can be edited in the control shown in 20.

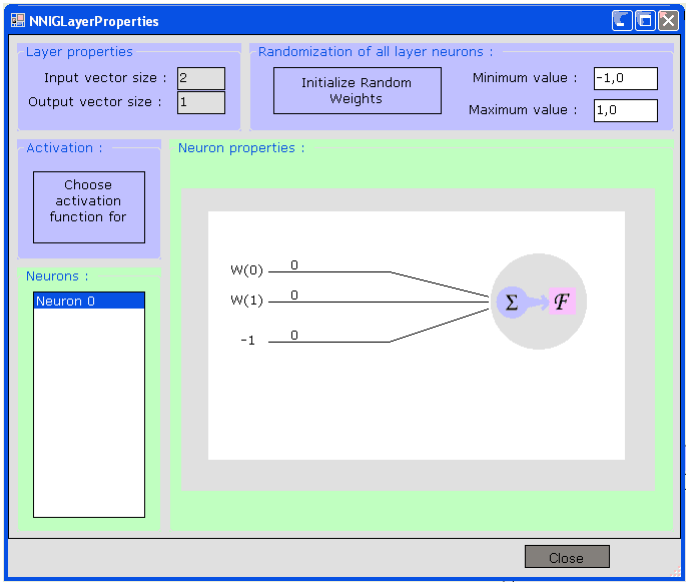

Figure 20: Layer editor control

Now, all the neurons of a particular layer have the same activation function and the bias is considered a synaptic weight.

The hyperbolic tangent activation function was added and the activation function selecting control is almost the same.

Was added a new method for initializing the synaptic weights (the initial weights are read from a file).

In this software we have two learning algorithms: the sequential back propagation (from Fleury) and the batch mode back propagation that can be chosen in the combo box of the control shown in 21 that is created when clicking in the tool strip menu button Back Prop.

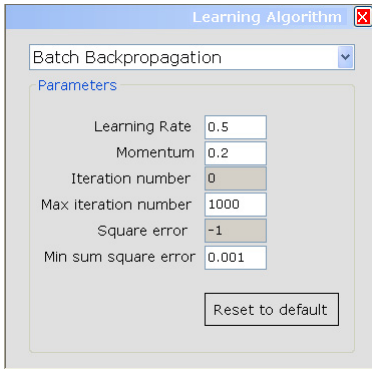

Figure 21: Learning algorithms selecting control

The button Run of the form starts the learning process and the result can be viewed by the value changing of the control specifications , as well as by the mean squared error chart and by the displayed output data. These new controls are created by clicking on the Chart or on the NNOutput tool strip menu buttons and are shown in the figures 22 and 23 respectively.

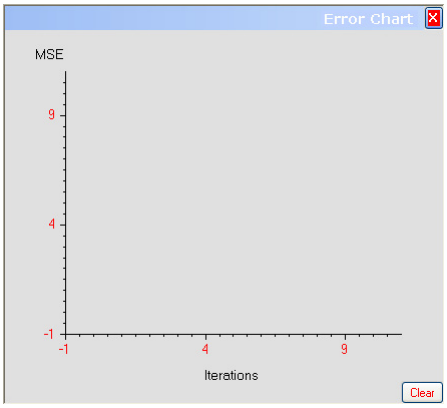

Figure 22: Chart control

In this control the clear button deletes the previous chart drawn.

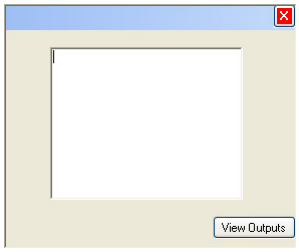

Figure 23: Output control

This second version of the software is working well and with good results. We applied it to known datasets and were able to validate the results by conducting the same experiments in the same conditions as other software products.

Bibliography

[1] Fleury, F. ; *http*: *//franck.fleurey.free.fr/NeuralNetwork/index.htm*

[2] *http* : *// en.wikipedia.org/wiki/C_Sharp*

[3] *http*: *//en.wikipedia.org/wiki/Microsoft_Visual_C_Sharp*

[4] *http*: *//msdn.microsoft.com/vstudio/express/visualcsharp/features/visuallydesign/*

[5] Pereira, V.;*O Guia Prático do Visual C# 2005 Express*; Tecnologias; Set/2006