NEURAL PATTERNS EDITOR, A JAVA APPLICATION FOR EDITING AND ANALYSING PATTERNS FOR TRAINING OR TESTING NEURAL NETWORKS

Alexandru Ene[1], Andrei Ene[2]
[1]Department of Electronics, Computer and Electrical Engineering, University of Pitești, Argeș, România
[2]Computer Science student, University of Pitesti Romania,
[1] alexandru.ene@upit.ro, [2] abi.moonbow@yahoo.com

Keywords- neural network, training patterns, graphical editor

Abstract – *Neural Patterns Editor is a graphical application used for creating binary patterns that are used with neural networks (for the training or for the testing). It is also used for storing the patterns in text files, editing, viewing, deleting and for showing relationships between patterns. The application is written in the Java language.*

I. INTRODUCTION

Feed forward neural networks are parallel devices that consist from interconnected processing units (artificial neurons) that are grouped in layers. They have an input layer, one or more hidden layers and an output layer. These networks are used in pattern recognition applications, their main advantage being that they do not need an exact mathematical algorithm for solving the problem of recognition. Instead, they learn to recognize an image or to classify it, using a set of predefined examples, in the same manner as human beings learn to recognize hand written digits or letters [2]. The examples that are used by the neural network to learn to classify an image are called training patterns. Each training pattern consists of two sets of data:
- the input part of the pattern that are the data that apply to all the inputs of the network.
- the output part of the pattern that are the ideal outputs of the neural network, when the input part of the pattern is applied on the inputs of the network.

The feed forward neural networks are trained to learn to classify a set of patterns through the backpropagation algorithm [1][4] :

- specify a ERROR for learning all the patterns

- initialize the weights with small random values

-set flag hasConverged to false

-while**(not hasConverged) **do

begin

for** all training patterns **do

begin

forward propagate the current input pattern (compute all the outputs of the neurons)

compute the learning error for the current pattern

backpropagate (compute the new weights, using delta rule)

end for

compute E, the total error of learning

if E<=ERROR than

set flag hasConverged to 1

end while

If the network does not converge i.e. if it cannot learn the training set of patterns in a specified maximum number of epochs the first thing that has to be done is to repeat the training. By repeating the training, the backpropagation algorithm will start using different values for the weights (because these are randomly initialized). So is possible now for the network to converge. This is the simplest method to be tried in order to solve the non-convergence problem.

But if the network still does not converge we have to look carefully over the training patterns

[4]. A bad training set is typical for the non-convergence of feed forward neural networks.

A feed forward neural network can have only binary inputs (only the input values 0 or 1), only analog inputs (for instance any value in the interval [0, 1] or mixed inputs (some inputs are binary, some are analog).

In this paper we present a software application called Neural Patterns Editor, that is very useful for binary feed forward neural networks, for creating and editing training patterns.

II. THE MAIN COMPONENTS OF THE APPLICATION

Neural Patterns Editor has four main components:

- graphical editor
- patterns viewer
- file splitter
- relationships discovery module

Graphical editor allows the user to create and to save binary patterns, as illustrated in the Fig. 1.

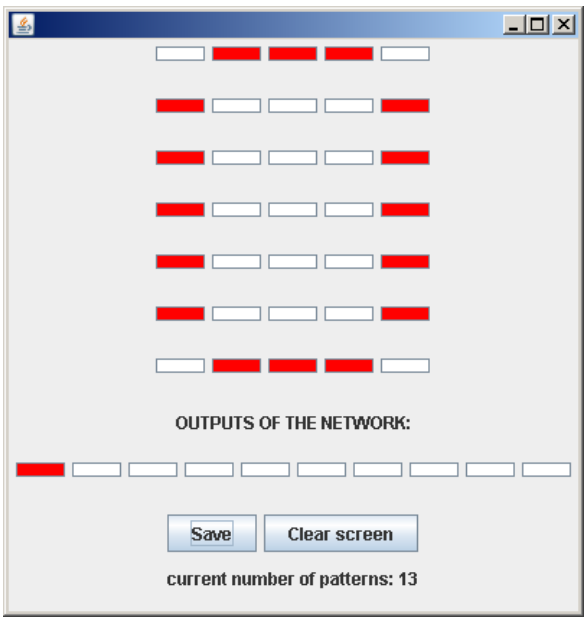

Fig. 1 Graphical editor

The current pattern is saved in a text file, patterns.txt, at the end of it. All saved patterns are written in this file. For the example shown in figure 1, the following lines are written at the end of patterns.txt:

0 1 1 1 0
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1
0 1 1 1 0
1 0 0 0 0 0 0 0 0 0

The current pattern is not saved if the outputs of the pattern are not correctly completed. There has to be a single and only a single activated output, as illustrated in figure 2 and figure 3.

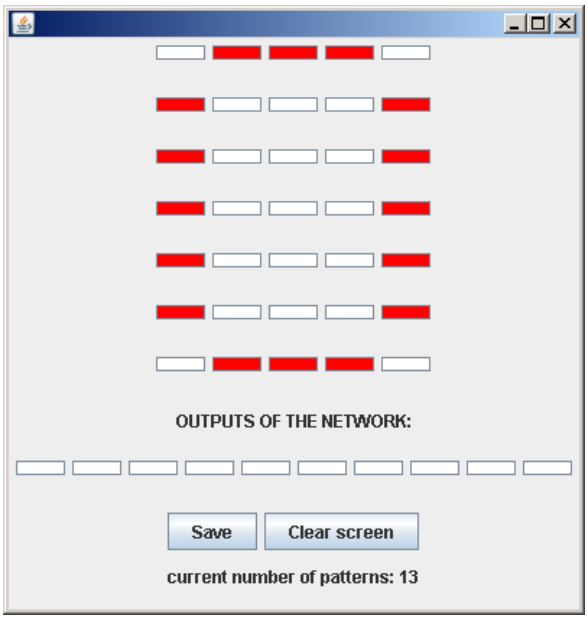

Fig. 2 The current pattern is not saved if the outputs of the pattern are not correctly completed

Fig. 3 Warning message

The current pattern is not saved also if the inputs of this pattern have been saved before as another pattern. In the case the user receives a warning like this:

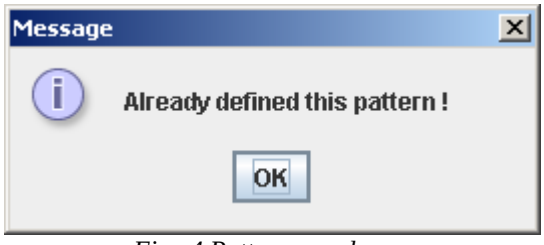

Fig. 4 Pattern saved message

So, the software prevents duplication of the patterns. A pattern that is good (its input part) is recorded in memory, in a bidimensional array(patt[ ][ ]). All good patterns (the input part of a pattern) are stored in this array. So, at the beginning of array patt[ ][ ] is stored the input part of the first training pattern, after it, in the same array is stored the second pattern, etc. A good pattern is also saved permanently in the text file patterns.txt.

We present briefly the algorithm that verifies if a new pattern has a duplicate. We use the following notations:

NP - current number of patterns
nL – number of lines of a pattern
nC – number of columns of a pattern

The algorithm is:

for i=0 to NP do
begin
set flag isDifferent to false
for j=0 to nL do
for k=0 to nC do
*if currentPattern[j][k] ≠ patt[i*nL+j][k] than*
set flag isDifferent to true
end

Patterns viewer

This tool allows the user to visualize all the patterns that are stored in the file patterns.txt . The user can navigate through the patterns, in both directions, using the buttons *Back* and *Next*.

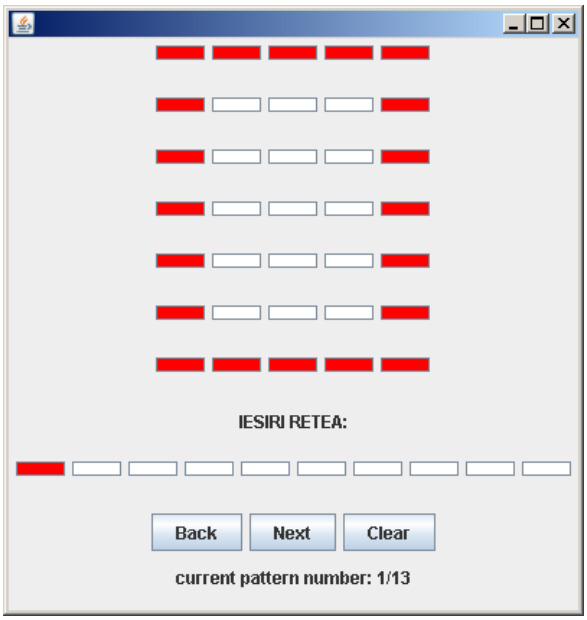

Fig. 5 Pattern delete

If the user considers that a pattern is not good, he may delete it permanently from the patterns file, using the button *Delete* as illustrated in figures 5 and 6.

File splitter is a software tool that copies all patterns from the same class (those that have the same outputs) from the big patterns.txt file to a separate file. So, for a neural network that learns to recognize the digits from 0 to 9, the file splitter tool will create 10 files: patt0.txt that groups all patterns for the 0 digit, patt1.txt that groups all patterns for the 1 digit, etc. For these new created files, the **patterns viewer** tool can be used, in order to view them or to delete some patterns.

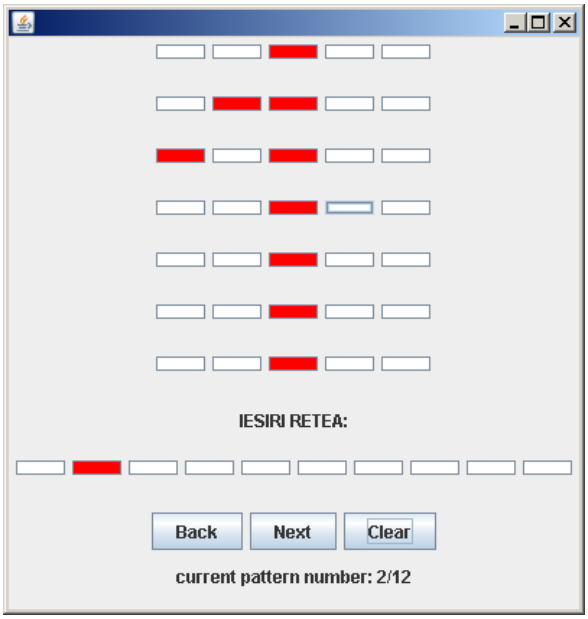

Fig. 6 Pattern delete example 2

Relationships discovery module is a software tool that uses each file created by **file splitter** tool, in order to find relationships between patterns from the same class. It finds out which are the two most different patterns from the same class and which are the two patterns that have the most resemblance. This software tool is useful when the neural network does not succeed to learn the patterns set, and we have therefore to examine them closely.

III. EXAMPLE OF AN APPLICATION

In [3] the authors had presented a method for the selection of the weights set of a feed forward neural network. To illustrate this method, there was used a feed forward neural network that had to classify a 9x9 image of black and white pixels

in three classes: vertical line, horizontal line and a cross line. The authors used an 81-20-3 feed forward neural network (81 input neurons, 20 hidden neurons and 3 output neurons) and they had developed 15 training patterns in order to train the network (5 patterns for each output class). These patterns were manually edited in a text file using Notepad editor.

For this task we could better use the Neural Patterns Editor. Firstly, the configuration file has to be created. It has three lines: the first line in this file specifies the number of lines of a pattern, the second line specifies the number of columns of a pattern and the third line specifies the number of outputs of the neural network. So the configuration file for this specific problem is:

9
9
3

Then we use our graphical editor tool in order to build and save all 15 patterns, and then we use the patterns viewer in order to examine the patterns.

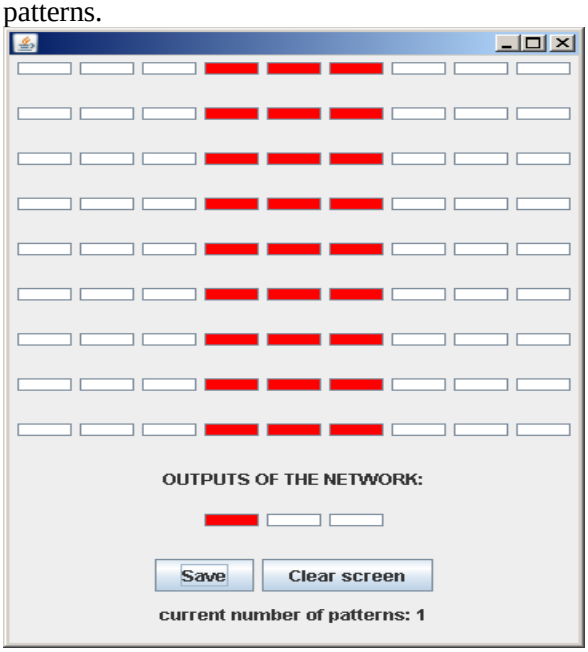

Fig. 7 Typcal pattern for a vertical line

A typical pattern for a vertical line, built with Neural Patterns Editor for this problem, is presented in Fig. 7.

After all the 15 patterns have been built, the resulted file, patterns.txt, is used by the specific neural software [3] to train the network.

If the network does not converge in the specified number of epochs and if this is caused by the training set of patterns, we have the possibility to analyze the patterns using file splitter tool in order to group the patterns from the same output class and to use the relationships discovery module in order to analyze comparatively the patterns from the same category.

IV. CONCLUSIONS

In this paper it was presented a Java graphical application that is used for creating binary patterns for the training or testing of feed forward neural networks.

This application is also used for storing the patterns in text files, editing the patterns (modify or delete), viewing the already stored patterns and for showing some relationships between patterns.

This graphical editor could be very easy modified in order to create patterns that are used with other types of neural networks (for instance for Hopfield neural networks).

REFERENCES

[1] A. Blum, Neural networks in C++, John Wiley,1992.

[2] Anton, C-tin, Stirbu, C., Badea R.V., "Automatic Hand Writer Identification Using the Feed Forward Neural Networks", World Congress on Internet Security (WorldCIS 2011), February 21-23, 2011, London, UK, pp. 304-307.

[3] Al. Ene, C. Stirbu, "Weights set selection method for feed forward neural networks" ECAI 2013, International Cionference 5th edition, 27-29 iunie, Pitesti

[4] Al. Ene, C. Stirbu, Retele neuronale. Teorie si aplicatii in Java, Ed. Universitatii din Pitesti, 2008.