Informatics in Education, 2011, Vol. 10, No. 2, 225–232

Visual NNet: An Educational ANN's Simulation Environment Reusing Matlab Neural Networks Toolbox

Emilio GARCÍA-ROSELLÓ, Jacinto GONZÁLEZ-DACOSTA,
María J. LADO, Arturo J. MÉNDEZ,
Baltasar GARCÍA PÉREZ-SCHOFIELD, Fátima FERRER
Faculty of Computer Sciences of University of Vigo
Campus As Lagoas, s/n. 32004 Ourense, Spain
e-mail: {erosello,jdacosta,mrpepa,mrarthur}@uvigo.es

Received: December 2010

Abstract. Artificial Neural Networks (ANN's) are nowadays a common subject in different curricula of graduate and postgraduate studies. Due to the complex algorithms involved and the dynamic nature of ANN's, simulation software has been commonly used to teach this subject. This software has usually been developed specifically for learning purposes, because the existing general packages often lack of a convenient user interface, and are too complex or inadequate for these goals. Since ANN's algorithms, types and applications grow regularly, this solution becomes more and more complex and inefficient. In this paper, we present Visual NNet, a learning-oriented ANN's simulation environment, which overcomes this problem by reusing Matlab Neural Networks Toolbox (MNNT), a well-known, comprehensive and robust ANN implementation. Visual NNet combines an on-purpose learning oriented design with the advantages of an ANN's implementation like MNNT. Furthermore, reusing MNNT has done Visual NNet development more cost-effective, fast and reliable.
Keywords: artificial neural networks, simulation software, reusability.

1. Introduction

Artificial Neural Networks (ANN's) are nowadays a well-known data processing method used in a growing range of applications, from pattern recognition to multidimensional data classification (see for example research journals like Neural Computing & Applications). Therefore, its presence in the educational curricula of different graduate and postgraduate studies has been more and more common in the last years. ANN's internal algorithms are based on relatively complex mathematical grounds, which are not within the scope of the interests of some curricula. For that reason some references focuses on a global understanding of neural networks and their usage from an intuitive perspective (Veelenturf, 1995).

However, in order to achieve a meaningful learning of ANN's concepts, behavior and practical applications, apart from the mathematical and algorithmic concerns, it is important to illustrate their structural and dynamic aspects, giving to students the chance of using them without necessarily having to deal with their underlying complexity. Moreover, this qualitative view of ANN's can help their quantitative understanding (Ploetzner and Spada, 1998). Computer-based modeling and simulation environments can be an excellent way to facilitate students to build ANN's models and to reach both a theoretical and procedural significant understanding of those concerns (Jonassen and Henning, 1999; Jonassen et al., 2007). This approach has been successfully used and reported in many works (see for example García Roselló *et al.* (2003), Gokbulut and Tekin (2006) or Gonzalez *et al.*, (2003)). In most of these references, an on-purpose application has been developed. This could seem curious, since there are many very robust and comprehensive implementations of ANN's, such as SNNS (University of Stuttgart, 2009), PDP++ (University of Colorado Boulder, 2007), Neuron (Yale University, 2005) or Matlab Neural Networks Toolbox (MNNT). Moreover, some of them are open software, and obviously less expensive than the development of a new on-purpose application. This could be explained by the need of a specifically learning-oriented software environment, which offers a simple and intuitive user interface, and a suitable support to teach ANN's concepts, since those aspects have been traditionally more valued than robustness or state-of-the-art features.

Nevertheless, new ANN's algorithms with increasing complexity are regularly coming up, and the range of problems and type of ANN's applied to solve them is also continuously growing up; therefore, this development of ad-hoc learning oriented ANN's applications turns to be also more complex and time-consuming. An efficient solution should consist in developing applications that comply with particular learning requirements, especially concerning user interface and complexity level, but reusing existing high-quality ANN's implementations for the internal processing. Unfortunately this is a challenging goal, because most of those implementations are not intended to be reused, or cannot be separated from their own user interface (Méndez *et al.*, 2007). In this paper, we explain the way to overcome this problem, in order to develop Visual NNet, a simulation environment that combines a very simple and easy-to-use learning-oriented user interface with the state-of-the-art and well-proven implementation of ANN's of the MNNT. In this way, Visual NNet development was clearly shortened and simplified, since algorithmic complexity relies on MNNT, and, furthermore, this provides a more robust and comprehensive ANN's functionality.

The rest of this paper is organized as follows. Firstly, we describe the materials and methods used to develop Visual NNet. Next, Visual NNet functionality is explained. Finally, the conclusions are detailed.

2. Materials and Methods

The main goal for developing Visual NNet was to provide to both students and teachers with a complete and easy-to-use ANN's learning environment, which offers a broad range of ANN types and algorithms, reusing an existing ANN's implementation, instead of building it from the scratch. The MNNT implementation was chosen because of its comprehensiveness (it implements any type of current ANN and a very large set of algorithms), robustness, regular updating, and also because of the large spreading of Matlab in university education.

The integration of MNNT functionality in another application requires reusing Matlab engine. However, we have previously explained that this is not always easy task, as there are important limitations or difficulties to reuse those existing packages (García Roselló *et al.*, 2007). To overcome this problem, we resorted to a solution that our research group has previously developed, aiming at improving reusability of this type of proprietary environments. This solution basically consists in a structured framework of reusable components, called IMO.Net for Matlab, that encapsulates and allows for the fully integration of the functionality of the proprietary environment in other applications. This framework includes a first-level component set that allows for reusing the common Matlab functionality, and several second-level component sets that offer more specific-domain features. One of those second-level component sets, called IMO.Net for Neural Networks Library, encapsulates the MNNT complexity and offers an intuitive object-oriented API (Méndez *et al.*,2007), making very easy to reuse it. It is beyond the scope of this article to describe in depth this framework, although detailed explanations can be found elsewhere (García Roselló *et al.*, 2007; Méndez *et al.*, 2007).

ANN's are basically data processing algorithms, and an application like Visual NNet has to offer to the users a simple way to input data to an ANN, like training and simulation patterns; and to visualize output data, such as simulation results, error adjustment rates or topological classifications (for ANN's like Self-Organizing Maps, for example). Instead of building our own solution for those requirements, which would require some adaptation and learning effort from users, we have chosen to reuse and integrate in Visual NNet an existing worksheet application. Since it is common that data susceptible of being processed with ANN's were managed and stored using these tools, and that the majority of the users are familiar with their usage, this solution makes easier to work with Visual NNet and to process results. At the same time, it also simplifies its development. Particularly, we have selected to integrate Microsoft Excel in Visual NNet because of its wide spreading.

The resulting architecture of Visual NNet is shown in Fig. 1. As it can be seen, users only have to deal with Visual NNet graphical user interface (GUI) and with worksheet software, but neither with details of the subjacent MNNT nor Matlab engine. Users are actually unaware of the fact that MNNT is used as ANN's implementation.

For the internal design, we have followed a typical Model-View-Controller architectural pattern (MVC) (Krasner and Pope, 1988). The model (a particular ANN implementation and the API to work with) is provided by the classes of the IMO.Net for Neural Networks Library. The view and the controller classes are implemented as part of the Visual NNet GUI; they are in charge of rendering the model in a suitable way for the user to be able to see and to interact with it, and they are responsible of making the pertinent calls to the model in response to user requests and events.

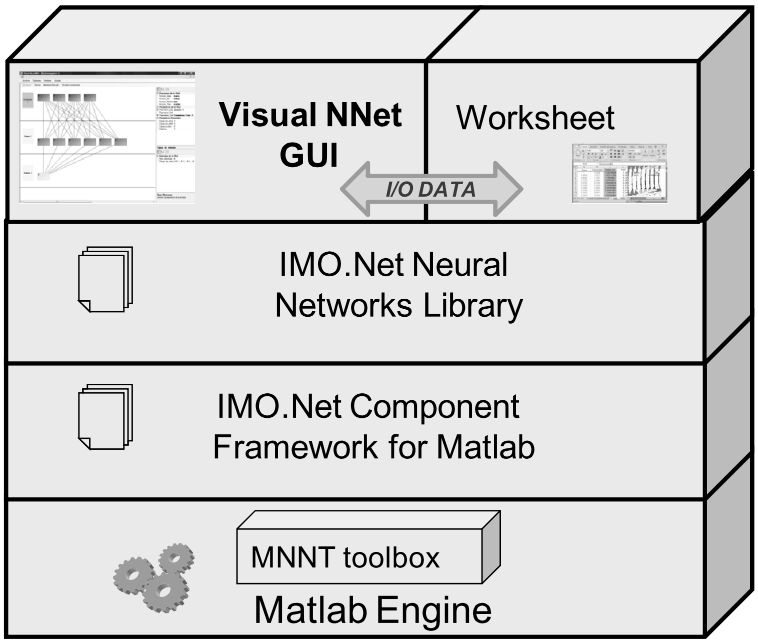

Visual NNet GUI
Worksheet
I/O DATA
IMO.Net Neural Networks Library
IMO.Net Component Framework for Matlab
MNNT toolbox
Matlab Engine

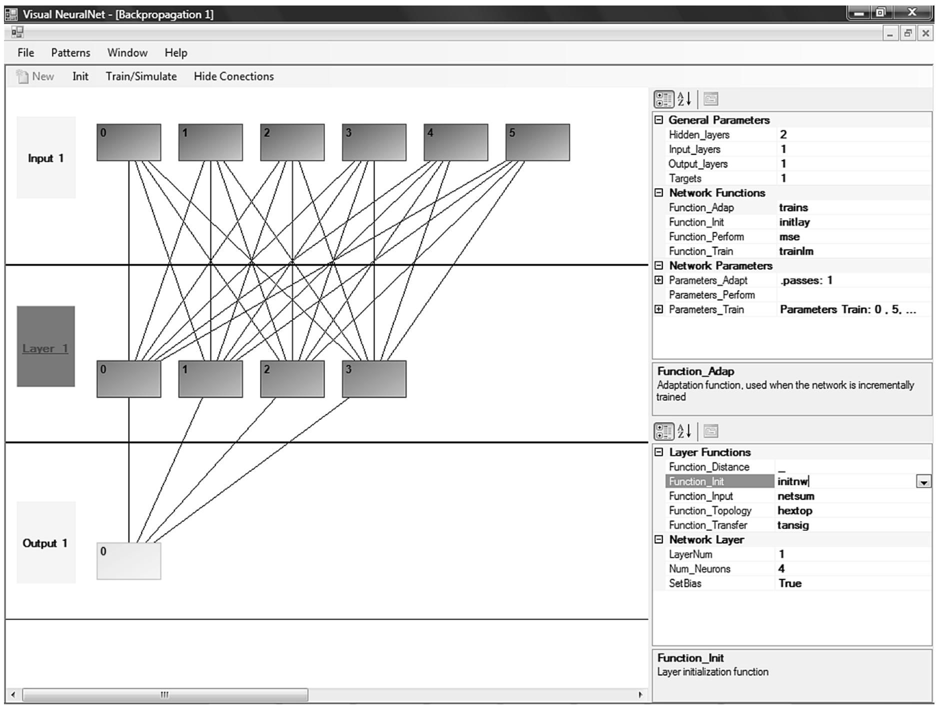
Fig. 1. Visual NNet architecture components.

3. Visual NNet Functionality

Visual NNet allows the user for creating, configuring, training and simulating an ANN through an intuitive and easy-to-use GUI. The user has first to select the type of ANN that he wants to create, choosing it in the corresponding menu option, and then sets its size. As it will be obvious from the previously explained, Visual NNet virtually supports all the types of ANN of the MNNT, notably simple and multilayer perceptrons, backpropagation networks, radial basis function networks, or self-organizing maps, to mention some of the more frequently used in ANN's courses. Once created, the ANN will be shown in a new window. Visual NNet has in fact a MDI (Multiple Document Interface), thus allowing for creating and working with several ANN's simultaneously, as each one will have its own child window. This feature is very useful for some learning purposes, such as, for example, activities including the comparative assessment with distinct ANN's, algorithms, or parameter values, which can be proposed to students.

An ANN is shown depending on its type. Feedforward-like networks are depicted in a visually intuitive way: each neuron is painted as a box that receives input connections from previous layer neurons and has output connections to next layer neurons. Layers are drawn from top to bottom, with input layer at the top and output layer at the bottom (Fig. 2).

Topological networks like self-organizing maps are typically shown as a bidimensional map with a dot for each neuron, and connections between the neurons within a Euclidean distance of 1.



General Parameters

Hidden_layers	2
Input_layers	1
Output_layers	1
Targets	1

Network Functions

Function_Adap	trains
Function_Init	initlay
Function_Perform	mse
Function_Train	trainlm

Network Parameters

Parameters_Adapt	.passes: 1
Parameters_Perform	
Parameters_Train	Parameters Train: 0 . 5. ...

Function_Adap
Adaptation function, used when the network is incrementally trained

Layer Functions

Function_Distance	_
Function_Init	initnw
Function_Input	netsum
Function_Topology	hextop
Function_Transfer	tansig

Network Layer

LayerNum	1
Num_Neurons	4
SetBias	True

Function_Init
Layer initialization function

Fig. 2. Screenshoot of visual NNet showing a backpropagation network. The properties panels on the right allow user easily to see and modify all network parameters and functions.

At the right side of the window two panels of properties are displayed (Fig. 2). The first one allows the user for easily consulting and modifying the general parameters and functions of the network, notably the function that will be used to train the network, the learning rate, etc. The second panel shows the properties corresponding to the network element currently selected by the user. That is, if the user clicks on a layer or a particular neuron of the graphic drawing of the network, this panel will automatically show its properties, like bias settings or input function for a layer, or connections weights for a neuron. The most of those parameters and functions, when modifiable, can be selected from a list of permitted values, which avoids mistakes and simplifies the usage. A help text is also displayed on the bottom of the panel corresponding to a brief explanation of the currently selected property.

To provide input data for both training patterns and simulation of ANN's, as we pointed above and in order to simplify and make more intuitive those tasks, Visual NNet allows users to directly use data contained in Excel worksheets. In this way, to define a dataset the user only has to open the worksheet where data are stored (or create a new sheet) and select the corresponding range of cells. Then he/she has to select the option of creating a new dataset in Visual NNet, assigning it a name. Visual NNet will automatically detect the selected range in Excel and link it to the assigned name. Visual NNet does not make a copy of the selected data, but it just registers the information needed to localize and retrieve them when needed. In this way, the users can define all the datasets they want. Then, when they want to train or simulate an ANN, the users have simply to select the dataset name to be used as input from a dropdown listbox. The information about the defined datasets is automatically stored by Visual NNet so it is directly available from one work session to another.

Output data produced by ANN's training or simulation are also transparently exported and displayed in an Excel worksheet by Visual NNet. That simplifies output data handling and analysis, such as performing statistical calculi or creating graphics, since the majority of the users know Excel functionality. For supervised training, typical data outputted by Visual NNet include error rate, expected and real output, to allow for an easy understanding of the performance of the ANN training. Obviously for simulation tasks, only real data output is shown.

It is important to stand out that, as it could be seen from the previous explanation of Visual NNet features, the user has no to directly deal with Matlab or MNNT. Visual NNet completely hides this aspect making it totally transparent to the user, which only has to work with its GUI and can be totally unaware of the fact that ANN's processing is done by MNNT.

4. Conclusions

In this paper we presented Visual NNet, a learning software environment for simulating ANN's developed with the aim of offering a specifically learning-oriented user interface but without renouncing to provide robust, complete and state-of-the art ANN features. To reach this goal with affordable costs and time, the choice was done of reusing MNNT, a well-know ANN's implementation. Thanks to the usage of the IMO.Net framework the integration of the full functionality of MNNT in Visual NNet was very simplified, and very little effort had to be put in this part of the development, compared to what should be required if ANN's algorithms and data structures had to be implemented. This also permitted to concentrate more resources in the user interface design and the learning-related features, and to drastically save time and costs without making any sacrifices in terms of functionality. By contrary, reusing MNNT provided Visual NNet with clearly more comprehensive ANN's features than it surely should have if we had to develop them from the scratch, and, at the same time, a specific learning-oriented design, which was very relevant for our aims.

Furthermore, the employ of Excel as either input source or output destination simplifies the data management and it does not require new learning from the users. In fact, this is an additional example of reutilization of proprietary environments. We can conclude that the reutilization of existing proprietary environments, of wide functionality, into on-purpose applications with educative aims, and with a friendly user interface, provides advantages in order to develop high quality educational software in a fast way.

Visual NNet will be shortly incorporated as a part of a Posgraduate course in the field of neural networks bases, to be held in the University of Vigo, Spain, in a near future.

Acknowledgements. This work has been developed as part of a research project funded by Xunta de Galicia (Spain) with identification key PGIDIT06SIN30501PR.

References

GarciaRoselló, E., Garcia Perez-Schofield, J., González Dacosta, J., Pérez Cota, M. (2003). Neuro-lab: a highly reusable software-based environment to teach artificial neural networks. *Computer Applications in Engineering Education*, 11(2), 93–102.

Garcia Roselló, E., Lado, M., Méndez, A., González Dacosta, J., Pérez, M. (2007). A component framework for reusing a proprietary computer-aided engineering environment. *Advances in Engineering Software*, 38, 256–266.

Gokbulut, M., Tekin, A. (2006). An educational tool for neural network control of brushless dc motors. *International Journal of Engineering Education*, 22(1), 197–204.

Gonzalez, E.J., Hamilton, A., Moreno, L., Aguilar, R.M. (2003). Neural networks teaching using Evenet-2000. *Computer Applications in Engineering Education*, 11(1), 1–5.

Jonassen, D.H., Henning, P. (1999). Mental models: knowledge in the head and knowledge in the world. *Educational Technology*, 39(3), 37–42.

Jonassen, D., Howland, J., Marra, R., Crismond, D. (2007). *Meaningful learning with technology*, 3rd edn. London, UK, Prentice Hall.

Krasner, G.E., Pope, S.T. (1988). A cookbook for using the model-view-controller user interface paradigm in smalltalk-80. *Journal of Object-Oriented Programming*, 1(3), 26–49.

Méndez, A., García Roselló, E., Lado, M., González Dacosta, J., Pérez, M. (2007). Integrating Matlab neural networks toolbox functionality in a fully reusable software component library. *Neural Computing & Applications*, 16, 471–479.

Ploetzner, R., Spada, H. (1998). Constructing quantitative problem representations on the basis of qualitative reasoning. *Interactive Learning Environments*, 5, 95–107.

The Mathworks (2009). Matlab. `http://www.mathworks.com/`.

University of Colorado Boulder. (2007). $PDP++$. `http://psych.colorado.edu/~oreilly/PDP++/PDP++.htm`.

University of Stuttgart. (2009). *SNNS-Stuttgart Neural Network Simulator.* Stuttgart Neural Network Simulator. `http://www.ra.cs.uni-tuebingen.de/SNNS/`.

Veelenturf, L.P. (1995). *Analysis and Applications of Artificial Neural Networks*. London, Prentice Hall.

Yale University. (2005). *Neuron*. `http://www.neuron.yale.edu/neuron/`.

E. García-Roselló is lecturer at the Department of Computer Sciences of the University of Vigo. He's got his PhD with a thesis on reusability in educational and scientific software. His research work, together with the other authors, is mainly centered on reusability and component-oriented software engineering and its applications to educational software.

J. González is lecturer at the Department of Computer Sciences of the University of Vigo. He's got his PhD with a thesis on educational software engineering. His research work is mainly centered on educational software engineering.

M.J. Lado is an associate professor at the Department of Computer Science of the University of Vigo, Spain. She received a PhD degree in physics in 1999 from the University of Santiago de Compostela. Her research interests include computer-aided diagnosis in breast and chest/CT imaging, as well as computer-aided systems for teaching purposes, and biomedical signal processing.

A.J. Méndez is an associate professor at the Department of Computer Science of the University of Vigo, Spain. He received his PhD degree in physics in 1996 from the University of Santiago de Compostela. His research interests include computer-aided diagnosis and educational software, and biomedical signal processing.

B.G. Perez-Schofield is an associate professor at the Department of Computer Science of the University of Vigo, Spain. His research interests include persistence and object orientation, especially as related to virtual machines and dynamic languages. He received his PhD in computer science from the University of Vigo.

F. Ferrer has got an MSc degree in computer sciences at the University of Vigo, and she has been collaborating actively with the other authors in several research projects, as the one presented in this paper.

„Visual NNet“: mokomoji dirbtinių neuroninių tinklų modeliavimo aplinka, naudojanti „Matlab Neural Networks Toolbox“

Emilio GARCÍA-ROSELLÓ, Jacinto GONZÁLEZ-DACOSTA, María J. LADO, Arturo J. MÉNDEZ, Baltasar GARCÍA PÉREZ-SCHOFIELD, Fátima FERRER

Dirbtiniai neuroniniai tinklai – tai informacijos apdorojimo struktūros, netiksliai imituojančios kai kuriuos gyvųjų organizmų smegenyse vykstančius informacijos apdorojimo procesus. Šiandieninėje akademinėje bendruomenėje tai populiari tema, kurios mokoma tiek bakalauro, tiek magistro studijų metu. Dirbtinių neuroninių tinklų algoritmų sudėtingumas lemia, kad jų mokymui yra naudojama programinė modeliavimo įranga. Ši programinė įranga buvo specialiai sukurta mokymui, nes egzistuojančiuose bendruosiuose paketuose trūksta patogios vartotojo sąsajos. Straipsnio autoriai pristato mokymui skirtą dirbtinių neuroninių tinklų modeliavimo programą „Visual NNet“, kuri realizuojama naudojantis „Matlab Neural Networks Toolbox“.