Simulation and Implementation of a Neural Network in a Multiagent System

D. Oviedo, M. C. Romero-Ternero, M. D. Hernández, A. Carrasco, F. Sivianes and J. I. Escudero

Abstract This paper presents the simulation and the implementation of a model of a neural network applied to a multiagent system by using the Neuroph framework. This tool enables several tests to be carried out and verify which structure is the best structure of our neural network for a specific application. In our case, we simulated the neural network for a sun-tracking control system in a solar farm. Initial implementation shows good results in performance, thereby providing an alternative to traditional solar-tracking systems.

Keywords Multiagent system · Neural network · Control systems · Position control · Solar-tracking · Solar energy · Renewable energy

1 Introduction

Multiagent technology has demonstrated its utility for the implementation of control systems, where it is necessary to control a distributed hardware platform [4, 9]. This requires agents to provide not only control capabilities, but also prediction and optimization capabilities.

The topic proposed in this paper refers to the simulations of a neural network model, whose design and implementation is described in detail in [6]. This model has been integrated in a multiagent system to support the prediction of the optimal movement patterns of solar panels for solar tracking.

Our neural network presents an adaptive two-dimensional auto-tracking control method suitable for a wide range of weather conditions. It utilizes real-time control

D. Oviedo · M. C. Romero-Ternero (✉) · M. D. Hernández · A. Carrasco
F. Sivianes · J. I. Escudero
Departamento de Tecnología Electrónica, Universidad de Sevilla,
Avda. Reina Mercedes s/n, 41012, Seville Spain
e-mail: mcromerot@us.es

strategies and control parameter auto-optimization methods to realize flexible timing tracking and photoelectric tracking control.

This paper is organized as follows: Sect. 2 presents the proposed neural network, while in Sect. 3, the simulations, implementation, and integration of the neural network with the multiagent system are described. Section 4 shows the tests performed and the results obtained. Finally, the concluding remarks are drawn in Sect. 5.

2 Intelligence and Learning Model for Agents

The model developed is a neural network that can determine an appropriate movement pattern to maximize performance of a general control system at a particular time of day for a specific year, in the environment in which the solar panel is located. This network considers the previous position of the solar panel regarding previous instants, and therefore predicts the coordinates of the subsequent movement (which are close). Furthermore, training and output feedback is analyzed by comparing energy performance. The network adjusts to the movements or output patterns that are most suitable.

The neural network model described is integrated in the CARISMA system [5]. CARISMA presents a multiagent architecture to obtain an integrated system to supervise solar farm infrastructures. This system is able to monitor the environment by collecting data from distributed sensors. It provides both efficient maintenance and prevention of failures in solar farms in terms of acting over the environment or to initialize automatic recommendations to the telecontrol operators.

In order to provide the software agents with the ability to identify behavior patterns that optimize the performance of the solar panels controlled by the system, full integration into the system of our modeled neural network is required. To achieve this inference, the agents responsible for the overall control of the system implement a neural network that enables them to determine patterns of behavior that improve and optimize the overall system capacity to obtain solar energy. When the neural network finds patterns that improve behavior, the agent reports them to the expert system already implemented in CARISMA.

The expert system is then responsible for carrying out actions that introduce optimal states to the system and that serve as a learning element for the network through feedback.

Figure 1 shows the modeled neural network and how it interacts with the expert systems included in CARISMA.

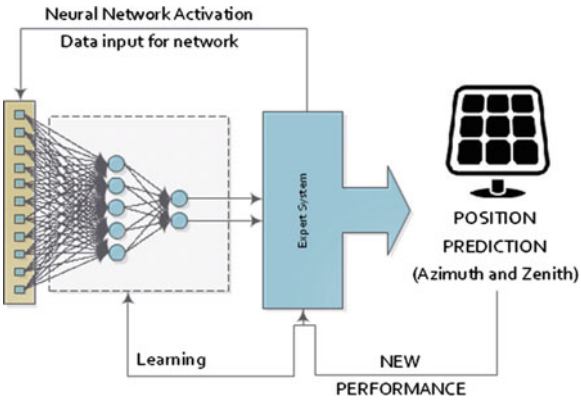

Fig. 1 Architecture decision-making control to optimize energy efficiency

3 Simulations and Implementation Details

To integrate the neural network into the expert system, it is first necessary to validate the neural network model through simulations and then implement this model. The following sections describe the various stages and elements that are necessary to attain a correct final implementation of the system.

3.1 Simulation of Neural Network

The modeled networks should be validated using a set of simulations that helps to refine certain elements of the neural network. Therefore, simulation phases are performed prior to the final implementation of the network and its integration into the CARISMA system.

Regarding the type of neural network, we decided to use a fully connected multilayer back-propagation network often called a multilayer perceptron (MLP) [3], which allows time series predictions to be obtained. Feed-forward networks can be applied directly to problems of this form if the data provided are suitably pre-processed.

In the normalization of the input data to train the network, theoretical minimum and maximum data are used depending on the area where the tests are executed (Seville, Spain, in our case). For example, the temperature parameter remains within it has been found that under normal conditions, a range from 0 to 45 °C. For those cases where one of the input parameters lies outside the specified target range, the system does not use the neural network to calculate the new coordinates. In this case, the expert system is directly responsible for the decision.

Figure 2 shows the topology for the first network trained. This is a fully connected multilayer network with one input vector of 13 parameters and one output vector with 2 parameters corresponding to the subsequent azimuth and zenith coordinates for the solar panels. The input and output vectors are interconnected by a hidden layer which is composed of three artificial neurons.

The number of neurons in the hidden layer can be increased in accordance with to the needs of network adjustment. The input values considered in the network are:

- Azimuth and Zenith solar-panel coordinates
- Solar time UTC
- Azimuth and Zenith sun coordinates
- Date
- Solar radiation (Kw/m^2)
- Temperature
- Humidity
- Wind speed
- Atmospheric Pressure
- Weather conditions (sunny, cloudy, rainy, etc.)
- Energy performance (kWh).

The simulation process is performed in order to determine various parameters of the network (maximum error, learning rate, momentum, impulse, etc.), to train the network through a training pattern preset based on theoretical maximum energy efficiency and to check its behavior in a variety of tests. The number of iterations required to train the network and to minimize the error are analyzed from the results of these simulations. The first results for our network, shown in Fig. 2, were inadequate, since the error was insufficiently low to consider the structure of this network as valid.

The number of hidden-layer neurons was therefore increased by following approximation rules [2]. First we tested the network with five neurons in the hidden layer and then with eight hidden neurons. Tests concluded that the latter type of network gives the best results for this problem, as shown in Fig. 3.

This network of eight neurons (and an additional "bias" neuron) in the hidden layer achieves better results in the tests performed, and shows minimal errors in most cases and a faster learning rate, as shown in Figs. 4 and 5. This neural network has been set with a maximum error of 0.01, a learning rate of 0.2, and a momentum of 0.5.

This network is selected and implemented in the CARISMA system, and other conditions of learning and operating over the time are included, as discussed in the following section.

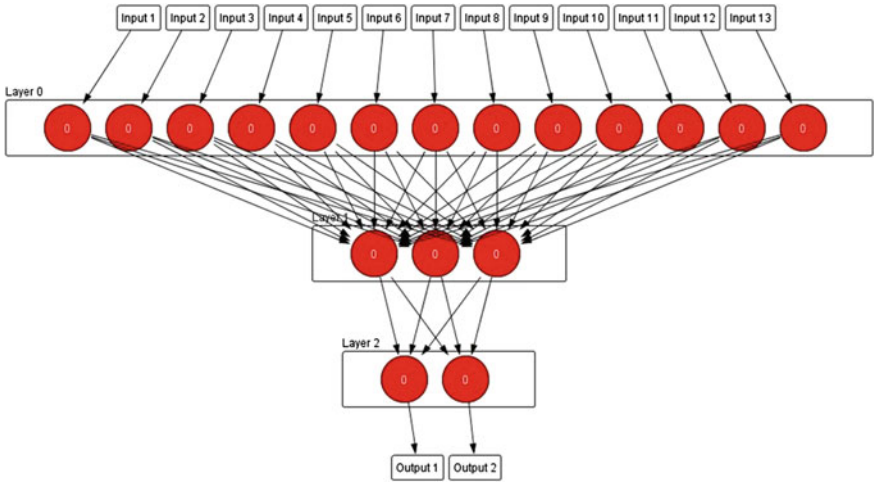

Fig. 2 First neural network simulation

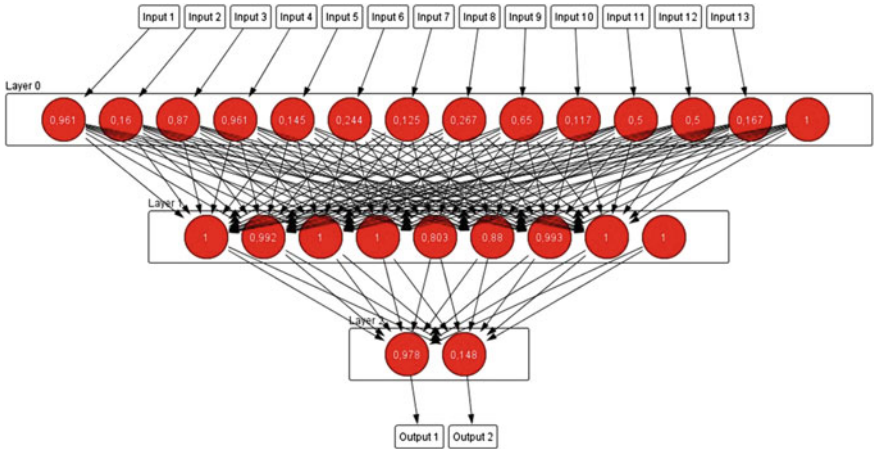

Fig. 3 Final topology of the neural network selected through simulation

3.2 Simulation of Neural Network

For the implementation of our neural network, the Neuroph framework [8] is used since it is the most suitable framework for integration into the CARISMA system. Neuroph offers a complete solution for creation, simulation, and implementation of neural networks in Java language. It provides a Java class library for integration into applications and a GUI tool "easyNeurons" for the creation and training of neural networks. Additionally, Neuroph is suitable for time-series prediction [7]. From the point of view of implementation, the expert system is responsible for

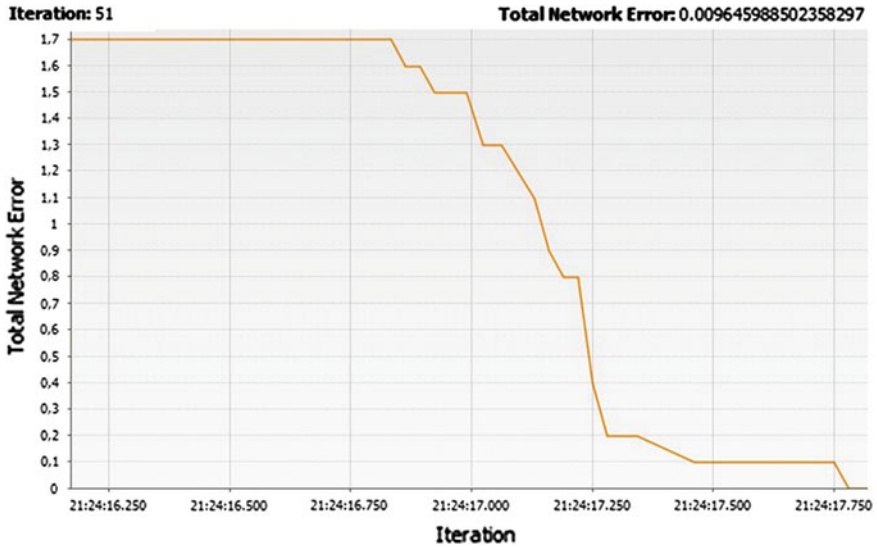

Fig. 4 Graphical comparison of the number of learning iterations and the total error on the neural network with eight neurons in the hidden layer (Minimum Test Time)

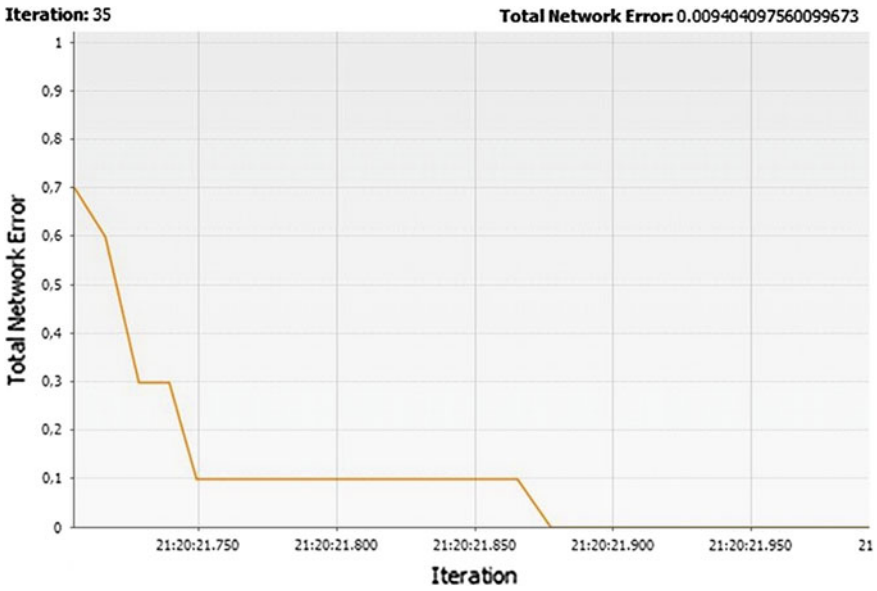

Fig. 5 Graphical comparison of the number of learning iterations and the total error on the neural network with eight neurons in the hidden layer (Maximum Test Time)

determining when the system makes use of a neural network in order to attain the subsequent movement for a given solar panel. Therefore, the expert system provides solar-panel environment data as the current position. Additionally, the expert system will be responsible for neural network learning, by comparing energy efficiency obtained from a particular movement. When output power from the system is less due to an executed movement, then the expert system reports the difference obtained to the neural network to adjust the weights. If output power is increased, then the expert system does not request new predictions from the neural network, and it sets suitable weights for future requests.

From the point of view of code, the creation of the network is encapsulated in the procedure "initializeNN()". For network initialization, we have made use of a temporary directory, where at least the initial training file for the network must be defined. This training file is a plain text file with fields separated by a semicolon (;). It contains the input values of the network and the best theoretical output result to obtain the maximum energy solar performance. This file can be edited with new data obtained from the network. Its use is obviated if the training option is adopted, since the network is trained with each output movement, thereby resulting in increasing performance from previously obtained movements under the same conditions. The data supplied to the neural network must be normalized with values ranging from 0 to 1, as indicated above. This aspect is dynamically controlled.

In order to make the network available during the operation of CARISMA once the network is trained, the neural network is stored in the file "myMLPerceptron.nnet", where it is encapsulated in a binary level. This network can be loaded and launched into the system from any agent. The procedure "execNN()" is called when the rules are triggered if the performance of the solar panels is inadequate. This procedure implements the following functionality:

- Launching the neural network for the current data.
- Comparing performance with the theoretical output, with a normal solar-tracking system, with fixed plates and a log of maximum energy efficiency for the same conditions.
- Training the network when a good result on the energy performance is obtained.
- Ensuring that the calculations do not run for nonlogical input values (for example, at night).
- Collecting times and system performance data.
- Storing returns and positions of the solar panels in a log file.

Finally, it has to be remembered that the network must be able to interact with the expert system integrated into the agent. Therefore, we implemented rules that are executed when the current energy efficiency (given by the current position of the solar panels) is less than the expected theoretical performance, which is reported by an alarm. In this case, the expert system updates the neural network using a process called "communicateNN()".

4 Testing and Experimental Results

We used the PeMMAS system [1] for testing, and internal variables and scripts of the system for measurement control. With this tool, data on the use of system resources, flight times of global messages by type of agent, and processing times in behaviors can be obtained. It may also be used for other external measures of system agents. The information obtained by PeMMAS agents can be processed to generate various reports to provide the analysis of the multiagent system.

The set of tests are focused on analyzing the energy efficiency obtained in one day, based on a history of performance measures obtained on various days of the year. These measures have three variants: fixed-position solar panels, solar panels with classic solar tracking (based on the theoretical solar position), and maximum energy efficiency obtained at various times of the day. The neural network is trained with the data obtained for panels with a traditional solar-tracking system. From this training, the neural network is available in the CARISMA system and runs on each of the agents responsible for controlling a solar panel.

After various tests were performed, we obtained an average of the most significant measures for the set of all agents. From the results obtained, as shown in Table 1, we conclude that the integration of a Neural Network has enabled a slight increase in the production of energy by the system. This increase has been at an average of approximately 3 %, compared to that obtained with a traditional solar-tracking system. The subsequent movements predicted by the neural network are the same as those expected in over 10 % and close to those expected in more than 25 %. However, a large number of unsuitable movements are also obtained (approximately 15 %).

These results are derived from historical data training. If no such data were available, the neural network response would initially be significantly worse and it would take several cycles of testing under the same conditions to obtain responses with an increased energy efficiency. Similarly, when the network is trained in real time, then it provides better results since the weights of the neurons adapt more appropriately.

Regarding the battery of tests designed to test the impact of using neural networks on the consumption in terms of processor and disk in the system, the results are satisfactory but considerable quantities of memory are employed, in particular if the network is used simultaneously with other types of inference systems. As shown in Table 1, the execution of the neural network has no impact on the system, and CPU consumption is less than 0.5 % as a result of the limited and controlled use by the expert system. However, the neural network is higher in terms of RAM consumption, and reaches an average consumption of 3 mb.

Additionally, we have also analyzed the response times of expert systems integrated with a neural network in the agents of our system. As shown in Table 2, the response times (for inferences) in milliseconds obtained for each type of agent did not differ from those obtained for the operation of expert systems alone. These

Table 1 Average results obtained by the neural network in predicting movements of a solar panel for performance optimization

Global measures on the system		
LIM_ NUMBER_AGENTS	27	Number of agents in the system
LIM_CAPACITY_PROC	16.78 %	% Average of CPU used
LIM_CAPACITY_STORAGE	54.56 %	% Average of memory used in the system
Neural network measures		
RN_TRAINING	916.6	ms
RN_RESPONSE SE	1374.72	ms
RN_CPU	0.272 %	% Average of CPU used in the system
RN_MEM	3079.4	Amount (in KB) of memory used in the system
RN_QUALITY TRUE	10.60 %	Expected resolution rate
RN_QUALITY FALSE	15.3 %	Wrong resolution rate
RN_QUALITY_RESOLUTION	5 %	Unexpected resolution rate
RN_ENERGY_EFFICIENCY_GLOBAL	2.72 %	Global energy efficiency rate obtained

Table 2 Average response times for the integrated expert system in a CARISMA agent when used in conjunction with a neural network

Time for inferences in neural network		
TIME_INFERENCE_AT	990.7	ms
TIME_INFERENCE_AC	940.7	ms
TIME_INFERENCE_AO	1070.02	ms
TIME_INFERENCE_ADS	683.5	ms

measures provided an average of one second per inference. Therefore, it can be concluded that the use of the neural network does not affect the use of other systems.

5 Conclusions

We have developed a complete multiagent system responsible for the supervision of an automated set of solar farms. The agents of this system have been equipped with artificial intelligence through the combination of expert systems and neural networks. Once the system has been verified through tests and simulations, the responses obtained from this system are suitable, and require very little human intervention.

The integration of the inference engine based on a neural network in an agent allows the system to optimize the energy efficiency obtained by adjusting the position of the solar panels to the variables of the particular environment and the position of the sun. This type of system enables the decision-making agents to adapt to their environment.

Future lines of work include the optimization of the neural network to improve the approaches on the movements of the solar panels, considering other environmental parameters. Therefore, one possible improvement would come from the implementation of a fully connected recurrent neural network, which, according to theoretical studies, would provide the best results.

Acknowledgements The work described in this paper has been funded by the Consejería de Innovación, Ciencia y Empresas (Junta de Andalucía) with reference number P08—TIC-03862 (CARISMA Project).

References

1. Carrasco A, Hernández MD, Sivianes F, Romero-Ternero MC, Oviedo D, Escudero JI (2014) PeMMAS: a tool that studing the performance of multi-agent systems developed in JADE, IEEE Transactions on Human-Machine Systems 44(2):180–189
2. Flórez RF, Fernández JMF (2008) Las Redes Neuronales Artificiales: fundamentos teóricos y aplicaciones prácticas. Netbiblo, Oleiros
3. Hertz J, Krogh A, Palmer RG (1991) Introduction to the theory of neural computation. Santa Fe Institute studies in the sciences of complexity: Lecture notes. Addison-Wesley, Massachusetts
4. Jennings NR, Bussmann S (2003) Agent-based control systems. IEEE Control Syst Mag 23:61–74
5. Oviedo D, Romero-Ternero MC, Hernández MD, Carrasco A, Sivianes F, Escudero JI (2010) Architecture for multiagent-based control systems, vol 79. Springer, Berlin, pp 97–104
6. Oviedo D, Romero-Ternero MC, Carrasco A, Sivianes F, Hernandez MD, Escudero JI (2013) Multiagent system powered by neural network for positioning control of solar panels: an optimization for sun tracking systems. In: 39th annual conference of the IEEE industrial electronics society (IECON 2013), Vienna, 10–13 Nov 2013
7. Paluch M, Jackowska-Strumillo J (2013) Prediction of closing prices on the stock exchange with the use of artificial neural networks. Image Process Commun 17(4):275–282
8. Sevarac Z, Goloskokovic I, Tait J, Carter-Greaves L, Morgan A (2013) Neuroph. http://neuroph.sourceforge.net
9. Wang Z, Yang R, Wang L (2010) Multiagent control system with intelligent optimization for smart and energy-efficient buildings. In: IECON 2010—36th annual conference on IEEE industrial electronics society, pp 1144, 1149, 7–10 Nov 2010