ADVANCED NEURAL TECHNIQUES FOR ENERGY MANAGEMENT

F. Garzia, G.M. Veca
Università di Roma "La Sapienza"

ABSTRACT

It is presented a powerful system to manage and control the energy in building. It is based on neural technology to adapt its management strategy to the controlled environment. The advanced neural techniques adopted is properly chosen to be implemented in a low computation capabilities device such a common electronic microcontroller, since it is characterized by a reduced number of learning operations and therefore of learning time.

1. INTRODUCTION

Different system architectures [1-4] can be used to manage and control the energy flows inside a building. They differ for the features and performances and obviously for the cost necessary to install them [5-7].
Once a particular hardware platform [8] has been chosen, it is necessary to implement a proper software that implements and executes the desired energy management strategy. The choice of the energy strategy needs to know in advance the needs of the final users together with their energy consumption temporal behaviour, that is a certain number of data must be collected, usually, for a long time.
It is possible to avoid this kind of problem using adapting strategies such as the one offered by the neural networks [9-12].
Neural networks offer the great advantage of learning the behaviour of the final users, together with a great generalisation capability that make them able to face new situations adopting a mixed behaviour between the learned ones.
The input can be represented, for example, by the presence of people inside a room, the outside temperature, the inside temperature, the time, the date and other data that are useful to characterise the desired application and so on.
The output data are represented by the desired energy management strategies as a function of the input data that act directly on the electrical and the air conditioner installations.

2. DEFINITION OF THE PROBLEM

Different parameters such as presence, light intensity, temperature, humidity, etc., can be used as inputs to the neural network to control the energy flows, using nets characterized by a growing complexity. Since we use as hardware a microcontroller with quite limited computation resources, it is necessary to reduce as more as possible the complexity of the net to reduce the computation duty and increase its velocity.
For this reason we decide to use as input only the presence of people inside the room, that is a binary information, and as output the switching on/off of the electrical loads, that is a binary information too.
The neural net has therefore to learn the occupation state of the controlled room, that is to predict when it is possible and necessary to switch on or off the electrical loads as a function of the previous occupation states.
The information about the presence is acquired by means of a sensor of presence while the output is executed by means of a switcher.

3. THE IMPLEMENTED NEURAL NETWORK

The neural net is fed with the last N values of the presence parameter, that are temporally spaced according to the desired precision, and the net must predict the next value of the output that is the net must learn to predict the state of occupation of the room basing on the previous occupation states of the room.
The used neural net is the so called feedforward net that is composed by two layers of neurons, all of them using log-sig functions. The first layers is composed by L neurons with N inputs. The second layer is composed by one neuron with a number of inputs equal to the number L of neurons output, as shown in the following figure.

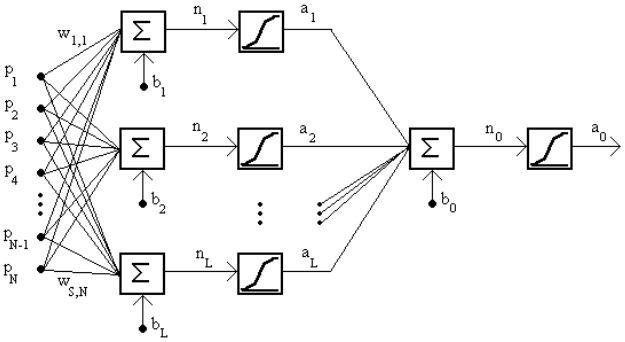

Fig.1 Neuronal model used

This net differs from the one studied previously [9] for different aspects that are illustrated in the following. The first and immediate one is the topology of the net: two layers-multiple neurons this one, one layer-one neuron the previous one [9]. The used configuration ensures the presented net to remember on yearly basis more than on daily basis. In this way the net is capable of remembering a sudden changed situation, represented for example by a rainy day in summer, whereas the previous net needs to a certain number of days to adapt to the new situation.
The used net is able to learn using a supervised training method where a set of Q couples of vectors

$[\mathbf{p_1},\mathbf{t_1}],[\mathbf{p_2},\mathbf{t_2}],....,[\mathbf{p_Q},\mathbf{t_Q}]$ are presented to the net, being **p** an input vector and **t** a target vector, and the sum of the average of the square errors between the output of the net with a given input and the desired output is calculated:

$$\frac{1}{Q}\sum_{j=1}^{Q} e_j{}^2 = \frac{1}{Q}\sum_{k=1}^{Q}\left[t_j - a_j\right]^2 . \qquad (1)$$

The weights and the biases are adjusted to reduce the error expressed by the expression (1), starting from the weights closer to the output and going towards the weights closer to the inputs, following a back path with respect to the input data, that gives the name of backpropagation to this kind of learning algorithm.

It is possible to demonstrate that for this kind of net the error has a quadratic expression and the performance index can show a global minimum, a weak minimum or no minimum, depending on the input vectors. The function expressed by eq.(1) is also called performance function since it expresses how correctly the approximate the desired output.

The back propagation learning methods belong to the so called gradient descent algorithms, where a proper performance function is used to measure the degree of learning of a net, following its reduction according to the more descending paths.

The simplest gradient descent algorithm is the represented by the Widrow-Hoff learn algorithm where the weight matrix **W** and the bias vector **b** are iteratively updated according to:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha\mathbf{e}(k)\mathbf{p}^T(k) \qquad (2a)$$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\mathbf{e}(k) \qquad (2b)$$

until a convergence takes place. In eqs.(3) **e** is the error vector and α is the learning rate. If α is too large learning occurs fast but if it is too large the algorithm can become unstable, diverging and increasing the error instead of reducing it. To avoid divergence the learning rate must be less than the reciprocal of the largest eigenvector of the correlation matrix $\mathbf{p}^T\mathbf{p}$ of the input vectors.

Properly trained backpropagation networks tend to give reasonable answers when presented with inputs that they have never seen. Typically, a new input will lead to an output similar to the correct output for input vectors used in training that are similar to the new input being presented. This generalization property makes it possible to train a network on a representative set of input/output target pairs and get good results without training the network on all possible input/output pairs.

We already said that there are many variations of the back propagation algorithm, some of which will be briefly illustrated in the following.

The simplest implementation of backpropagation learning updates the network and biases in the direction in which the performance function decreases most rapidly, that is the negative of the gradient. One iteration of this algorithm can be written as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k\mathbf{g}_k \qquad (3)$$

where $\mathbf{x}_k$ is a vector of current weights and biases. $\mathbf{g}_k$ is the current gradient and α_k is the learning rate.

The simplest technique consists in moving the weight and the biases in the direction of the negative gradient of the performance function. This is an efficient and simple technique that can sometimes stop in local minimums, without reaching the global minimum of the performance function. To avoid this kind of problem and to increase the velocity of convergence it is possible to use the so called steepest descent with momentum. Momentum allows a network to respond not only to the local gradient, but also to recent trends in the error surface. Acting like a low pass filter, momentum allows the network to ignore small features in the error surface. Without momentum a network may get stuck in a shallow local minimum while if it is present, a network can slide through such a minimum. Momentum can be added to the back propagation learning by making weight changes equal to the sum of a fraction of the last weight change and the new change suggested by the backpropagation rule. The magnitude of the effect that the last weight change is allowed to have is mediated by a momentum constant which is a number between 0 and 1. When the momentum constant is 0 a weight change is based solely on the gradient. When momentum constant is 1 the new weight is set equal the last weight change and the gradient is ignored.

A proper variation is represented by the variable learning rate, that keep the learning size as large as possible while keeping learning stable. The learning rate is made responsive to the complexity of the local error surface. It ensures a higher learning velocity with respect to the other algorithms.

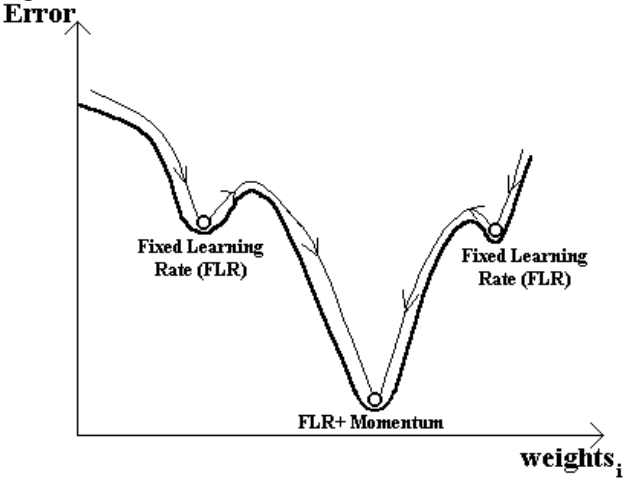

Fig.2 Steepest descent and steepest descent with Momentum: error behavior

Fast and precise learning technique are represented by the so called quasi-Netwon algorithm, that are based on the basic step of the Newton's method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k \qquad (4)$$

being $\mathbf{A}_k$ the Hessian matrix (a matrix composed by the second derivatives) of the performance function at the current values of the weights and biases. Newton's method often converges faster that the other method but it is characterized by a certain calculation complexity due

to the need of computating the Hessian matrix of the net. For this reason it is very useful to use the quasi-Newton method, that use an approximated Hessian matrix that is updated at each iteration of the algorithm. A very efficient quasi-Newton method is represented by the Levenberg-Marquardt algorithm that is capable of reaching the typical second order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares, such as eq.(1), then the Hessian matrix can be approximated as:

$$\mathbf{H}=\mathbf{J}^T\mathbf{J} \qquad (5)$$

and the gradient can be computed as:

$$\mathbf{g}=\mathbf{J}^T\mathbf{e} \qquad (6)$$

where **J** is the Jacobian matrix, which contains first derivatives of the network error with respect to the weights and biases, and e is a vector of network errors. The Jacobian matrix can be computed through a standard finite difference technique that is, given a function F of more variables (x_1, x_2,….., x_N) the first derivatives $\frac{\partial F}{\partial x_i}$ of F with respect to the generic variables x_i can be calculated as:

$$\frac{\partial F}{\partial x_i} \cong \frac{F(x_1,..,x_i+\Delta x_i,...,x_N)-F(x_1,..,x_i,...,x_N)}{\Delta x_i} \qquad (7)$$

if Δx_i is little enough. The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$\mathbf{x}_{k+1}=\mathbf{x}_k-[\mathbf{J}^T\mathbf{J}+\mu\mathbf{I}]^{-1}\mathbf{J}^T\mathbf{e} \qquad (8)$$

When the scalar μ is zero, this is jus Newton's method, using approximate Hessian matrix. When μ is large, this become gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift towards Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function . In this way, the performance function will always reduced at each iteration of the algorithm.

	Fixed learning rate (FLR)	FLR+ momentum	Variable learning rate (VLR)	VLR+ momentum	Levemberg Marquardt
Time	30	6.8	8.4	8.6	1
Train cycles	163	31	18	14	1
Floating point operations	5.4	1.21	1.52	2.15	1

Table 1: Comparison between the different learning algorithms (normalized to the Levemberg-Marquardt algorithm)

Once explained the different and more common learning algorithms for the feedforwards neural networks we show the operative scheme of the used neural net.

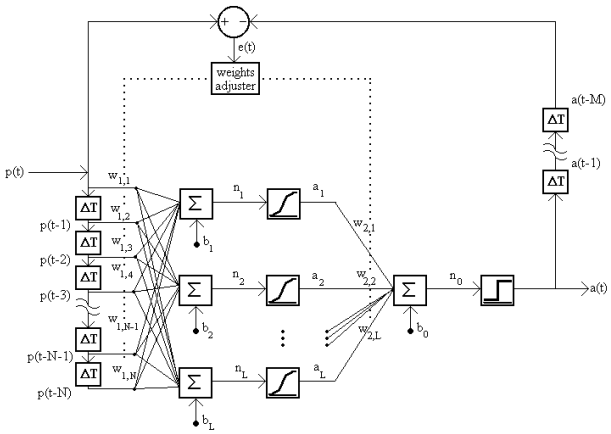

Fig.3 Operative scheme of the used neural network

The elements pointed with ΔT are delay elements that give as output their input after a time interval equal ΔT. The system acts as an advanced predictive filter that estimates the actual value of the input variable once known N previous values of the same variable.

The actual value of the presence variable, that is a binary variable, is fed into the system that calculates the predicted value of the presence variable, necessary to decide if the system can switch the electrical loads on or off. The system uses the actual value of the presence variable and its previous N-1 values.

The prediction interval can be decided by introducing a certain number of delays elements in the back loop. In fact, since the back loop is used to calculate the error between the actual value and the predicted value, if we want the net to predict what is the value of the presence variable M·time intervals ΔT in advance, it is necessary to introduce M delays units. In this way the M-th delay unit gives at its output the presence variable predicted M time intervals in advance and we wish this variable to be equal to the actual presence variable. The comparison between these two variables is made by the error unit that executes their difference: if the error is different from zero the weight adjuster unit trains the net to improve the prediction capability of the system. If the error is equal to zero it means that the system was able to predict the actual value of the presence value M time intervals in advance and that it works well. The neural net is trained by adjusting the weights and the bias of the net using the Levenberg-Marquardt algorithm expressed by eq.(8).

It is possible to see from the comparative table 1 that the used net ensures a reduced number of training cycles, a reduce number of floating point operations and therefore a reduced learning time with respect to the net studied previously [9] that uses the Widrow-Hoff rule by means of equations (3) which belong to the fixed learning rate algorithms family.

From the figure 2 it is also possible to see that the used learning algorithm ensures the net to reach a global

minimum of the error whereas the net studied previously is capable of reaching only local minimums of the error.
Since the presence variable is a binary variable that can assume only the values 1 or 0 and the learning function is sigmoidal, the output of the neuron could assume any value. The output variable has to control electrical loads and it must assume only the value 1 or 0, as the input variable: for this reason a step transfer function is selected for the output neuron.

4. SYSTEM IMPLEMENTATION AND RESULTS

The basic practical implementation of this kind of net has already been discussed [9] and it is not repeated here for brevity.
In the studied net, to ensure a high degree of precision, it is necessary to extend as more as possible the number of past input presence variables: the best results are obtained if a 24 hour period is used as input that is a total of 144 inputs (24 hour multiplied 6 samples per hour).
Since the complexity of the net, related to the number of neurons L of the first layer depends on the variability of the input data, that is similar input patterns need a low number of neurons to be properly recognized while very different input patterns (owed, for example, to great weather variability that produces a great variability of the occupation state of the room) need a quite high number of neurons to be properly recognized, it has been introduced a parameter called “day variability” (DV) that represents the variation degree between two subsequent days. It consists, for all the 144 samples points used by the system, in the calculation of the absolute value of the difference between the desired output $O_D(i)$ of the system on the actual day and the desired output $O_{D-1}(i)$ of the system on the previous day, both taken at the same sample time i:

$$DV = \frac{\sum_{i=1}^{144} \left| O_D(i) - O_{D-1}(i) \right|}{144}. \qquad (9)$$

From the given definition it is evident that if a considered day is characterized by a DV equal to 1 it is totally different from the previous day (the system must switch on whereas in the previous day it had to switch off and vice versa) while if a considered day is characterized by a DV equal to zero it is exactly equal to the previous day.
The DV parameter is very useful in characterizing the variability of the input data that greatly affects the performance of the net and therefore its complexity.
The first design parameter of the net is the number of neurons L that strongly influences the switching error of the system. It is evident that a low value of the DV parameter, that means input data characterized by a low variability, needs a reduced number L of neurons for the correct working of the net, while a high value of the DV parameter needs a elevated number of neurons to learn the variation features of the input data and therefore to switch correctly the system.
Different tests on the net were made varying the number of neuron L and the DV parameter of the input data to check the system behavior from the switching error point of view.
In figure 4 the relative switching error as a function of the number L of neurons for different values of DV parameter, is shown. It is possible to see that the relative switching error decreases when the number of neurons L increases.
It is also possible to see that the more the DV parameter increases (that is the variability of the input data increases) and the more it is necessary to increase the number of neurons L to obtain a given relative switching error.

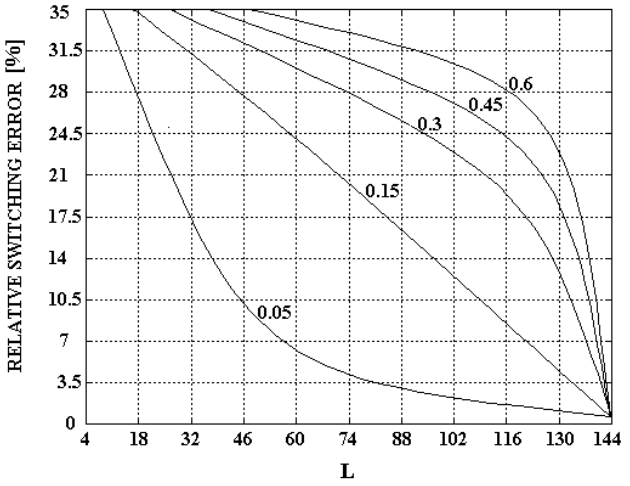

Fig.4 Relative switching error as a function of the number of neuron, for different values of DV parameter.

The number of neurons L as a function of DV parameter for different values of relative switching error is shown in figure 5.

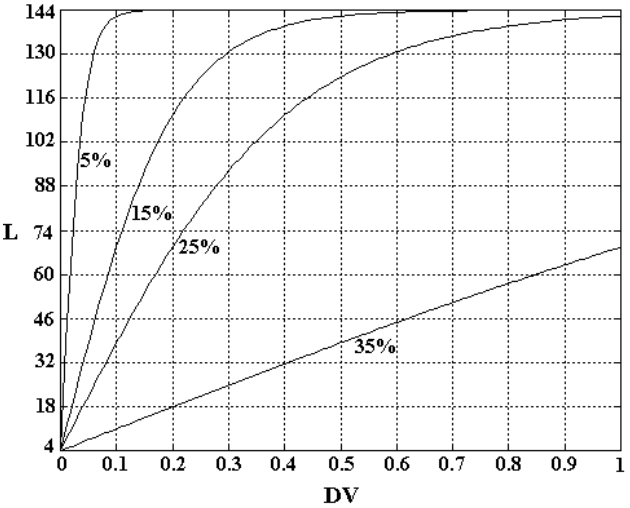

Fig.5 Number of neurons L as a function of DV for different values of relative switching error.

It is possible to see that when DV parameter increases (high variability of input data) the number of neurons L increases rapidly if a reduce switching error is desired while it increases slowly if a greater switching error is tolerated.

Another important factor to be considered is the number of floating point operations (NFPO) that must be executed to train the net, since we wish to use a limited computation resources microcontroller. It is obvious that NFPO grows not only with the number of neurons L but also with the DV parameter, since a highly variable input set needs a greater number of operations to ensure the convergence of the learning algorithm. This is a very important factor that must be considered to avoid the microcontroller to spend the most of time in computation operations, neglecting the other controlling duties.
In figure 6 the NFPO as a function of the number of neurons L, for different values of DV, is shown.

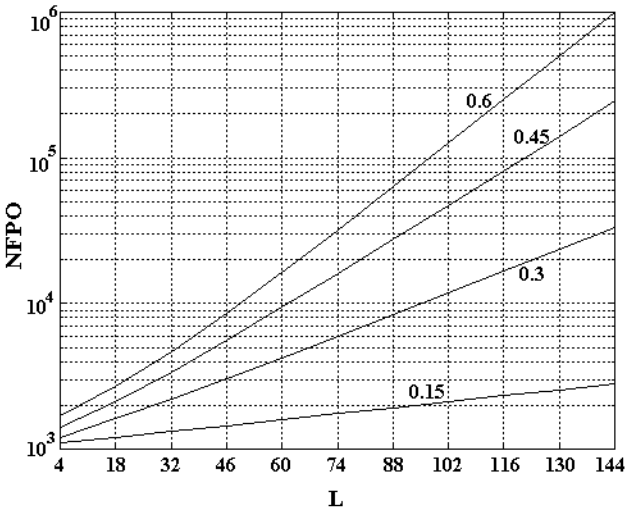

Fig.6 NFPO as a function of the number of neurons L, for different values of DV.

The learning time depends on the specific microcontroller used and on the time necessary to execute a floating point operation (FPO): for this reason, in figure 6, it has been indicated the NFPO so that it is possible to know immediately the learning time when the system is implemented on different hardware platforms characterized by a different FPO execution time.
The used net needs a certain training time before predicting the behavior of the presence variable inside the controlled room with a definite degree of precision: the higher the number of neurons L and the shorter is the training period (different from the learning time that depends on NFPO as explained before).
The training time also depends on the variability of the input data, related to the DV parameter according to the eq.(9), that can be reduced, as usual, increasing the number of neuron L according to the results obtained.
Different trainings of the net were performed to derive the minimum number of input data, which is equivalent to the training days in a real system, necessary to obtained a certain relative switching error. The training were performed varying the number L of neurons to test the behavior of the net and considering a day variability DV equal to 0.3.
The results are shown in figure 7, where the training days as a function of neuron L, for different values of relative error, are plotted.

It is possible to see that to obtain a reduced switching error it is necessary to increase the number of training days or increase the number L of neurons, considering anyway that the curves tend to present an asymptotic behavior above a certain value.
It is evident that if DV increases the net needs a longer training period since it has to learn the variability features of the input data and vice versa.

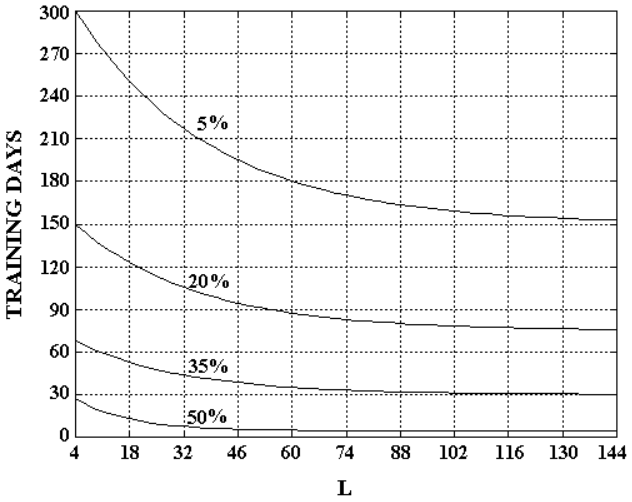

Fig.7 Number of training days as a function of the number of neurons L, for different values of relative switching error.

The proposed neural net shows a higher degree of flexibility in the choice of the relative switching error with respect to the net proposed previously [9] since it is possible to increase the number of neuron L to reduce the mentioned error at will, whereas the other net shows a higher and unpredictable error that depends on which local minimum of the error surface the learning algorithm has been able to reach.

5. CONCLUSIONS

A versatile system, based on neural technology, to manage and control the energy in buildings has been presented. It is able to change and adapt its management strategy to the controlled environment. The advanced neural technique adopted is properly designed to be implemented in a low computation capabilities device such a common electronic microcontroller.
The proposed neural net has been studied into details obtaining some useful design parameters that allows to achieve the desired results as a function of the features of the environmental where the system operates and as a function of the hardware platform used.

6. REFERENCES

[1] S. Mc Clelland, “Intelligent building”, New York: Springer-Verlag, 1989.
[2] J.A.Bernaden, R.E.Neubauer, “The intelligent building sourcebook”, Lilburn: Fairmont Press, 1988.
[3] V. Bradshaw, K. E. Miller ,”Building Control Systems”, John Wiley & Sons.

[4] V. Boed, “Networking and Integration of Facilities Automation Systems”, CRC Press.
[5] M. Eyke, ”Building Automation Systems: a Practical Guide to Selection and Implementation”, Blackwell Science (UK).
[6] D. A. Wacker, “Complete Guide to Home Automation”, Popular Woodworking.
[7] B. Gerhart, “Home Automation Guide for Builders”, McGraw-Hill Publishing.
[8] F. Garzia, G. M. Veca, “Smart Automatic Control of Energy Flows in Building”, DUE 2000, Cape Town.
[9] F. Garzia, G. M Veca, “Neural technique for energy management in building”, DUE 2001, Cape Town.
[10] N. K. Bose, P. Liang, “Neural Network Fundamentals with Graphs, Algorithms, and Applications”, McGraw- Hill Publishing Company, 1998.
[11] S. I. Gallant, “Neural Network Learning and Expert Systems”, MIT Press, 1997.
[12] W. T. Miller, “ Neural Networks for Control (Neural Network Modeling and Connectionism), MIT Press, 1999.

7. AUTHORS

Principal Author: Fabio Garzia holds a degree in Electronics Engineering and a PhD in Applied Electromagnetism from the University of Rome “La Sapienza”, Italy. He is appointed professor at the degree in Safety and Security Engineering of the same university. He is also responsible for the Applied Security Laboratory. His mail address is:

Dipartimento di Ingegneria Elettrica
Università degli Studi di Roma "La Sapienza"
Via Eudossiana, 18
00184 Roma, Italy

Co-author: Giuseppe M. Veca is full professor of Electrical Engineering at the University of Rome "La Sapienza", Italy. He is Senior Member of IEEE and Dean of the degree in Electrical Engineering. His fields of research are EMC/EMI, Magnetic Devices and Power Apparatus and Systems. His mail address is:

Dipartimento di Ingegneria Elettrica
Università degli Studi di Roma "La Sapienza"
Via Eudossiana, 18
00184 Roma, Italy

Presenter: The paper is presented by Fabio Garzia