UNIVERSITÀ DI PISA

Scuola di Dottorato in Ingegneria "Leonardo da Vinci"

Corso di Dottorato di Ricerca in
Ingegneria dell' Informazione

Tesi di Dottorato di Ricerca

Computational Intelligence for classification and forecasting of solar photovoltaic energy production and energy consumption in buildings

Eleonora D'Andrea

Anno 2013

UNIVERSITÀ DI PISA

Scuola di Dottorato in Ingegneria “Leonardo da Vinci”

Corso di Dottorato di Ricerca in Ingegneria dell’Informazione

Tesi di Dottorato di Ricerca

Computational Intelligence for classification and forecasting of solar photovoltaic energy production and energy consumption in buildings

Autore:

Eleonora D’Andrea __________________

Relatori:

Prof. Beatrice Lazzerini ________________

Prof. Francesco Marcelloni ________________

Ing. Marco Cococcioni ________________

Anno 2013

SSD ING-INF/05

*“Non sono le specie più forti quelle
che sopravvivono e nemmeno le più
intelligenti, ma quelle in grado di
rispondere meglio al cambiamento”*

*“It is not the strongest of the species
that survive, nor the most intelligent, but
the one most responsive to change”*

[Charles Darwin]

Sommario

Questa tesi presenta una serie di nuove applicazioni di tecniche di Computational Intelligence in problemi del settore energetico, con particolare riferimento alla valutazione dell'energia prodotta da impianti fotovoltaici e alla valutazione dei consumi energetici di edifici. Infatti, di recente, grazie anche alla crescente evoluzione tecnologica, il settore energetico ha attirato l'attenzione della comunità di ricerca scientifica nel proporre strumenti utili in problemi di efficienza energetica negli edifici e nella gestione della produzione di energia solare. Affronteremo quindi due tipologie di problemi.

Il primo problema è legato alla gestione efficiente degli impianti solari fotovoltaici, per esempio, per controllare efficacemente le prestazioni e per la ricerca di guasti, o per la pianificazione della distribuzione di energia elettrica in rete. Questo problema è stato affrontato con due approcci diversi: un approccio di previsione e un approccio di classificazione fuzzy per stimare la produzione di energia, a partire dalla conoscenza di alcune variabili ambientali. Il sistema di previsione sviluppato è in grado di riprodurre la curva giornaliera di energia prodotta dai pannelli solari dell'impianto, con un orizzonte temporale di previsione di un giorno. Il sistema sfrutta reti neurali e modelli di analisi di serie temporali. Il sistema di classificazione fuzzy, invece, estrae una certa conoscenza linguistica relativa alla quantità di energia prodotta dall'impianto, sfruttando una base di regole fuzzy ottimale e impiegando algoritmi genetici. Il modello sviluppato è il risultato di una nuova metodologia di tipo gerarchico per la costruzione di sistemi fuzzy, che può essere applicata in molteplici settori.

Il secondo problema è legato alla efficienza energetica degli edifici, allo scopo di ottenere benefici quali la riduzione del costo o la pianificazione del carico, ed è stato affrontato proponendo un sistema di previsione dei consumi energetici negli uffici. Il sistema sviluppato sfrutta una rete neurale per stimare il consumo di energia per illuminazione in un intervallo di tempo di alcune ore, a partire da considerazioni sulla luce naturale esterna disponibile.

Abstract

This thesis presents a few novel applications of Computational Intelligence techniques in the field of energy-related problems. More in detail, we refer to the assessment of the energy produced by a solar photovoltaic installation and to the evaluation of building's energy consumptions. In fact, recently, thanks also to the growing evolution of technologies, the energy sector has drawn the attention of the research community in proposing useful tools to deal with issues of energy efficiency in buildings and with solar energy production management. Thus, we will address two kinds of problem.

The first problem is related to the efficient management of solar photovoltaic energy installations, e.g., for efficiently monitoring the performance as well as for finding faults, or for planning the energy distribution in the electrical grid. This problem was faced with two different approaches: a forecasting approach and a fuzzy classification approach for energy production estimation, starting from some knowledge about environmental variables. The forecasting system developed is able to reproduce the instantaneous curve of daily energy produced by the solar panels of the installation, with a forecasting horizon of one day. It combines neural networks and time series analysis models. The fuzzy classification system, rather, extracts some linguistic knowledge about the amount of energy produced by the installation, exploiting an optimal fuzzy rule base and genetic algorithms. The developed model is the result of a novel hierarchical methodology for building fuzzy systems, which may be applied in several areas.

The second problem is related to energy efficiency in buildings, for cost reduction and load scheduling purposes, and was tackled by proposing a forecasting system of energy consumption in office buildings. The proposed system exploits a neural network to estimate the energy consumption due to lighting on a time interval of a few hours, starting from considerations on available natural daylight.

Table of Contents

List of Acronyms and Abbreviations

Acronym	Meaning
AC	Alternating Current
ANN	Artificial Neural Network
APE	Absolute Percentage Error
AR	Auto-Regressive
ARIMA	Auto-Regressive Integrated Moving Average
CI	Computational Intelligence
CF	Certainty Factor
DB	Data Base
DC	Direct Current
DP	Dominance Percentage
ECL	Energy Consumption due to Lighting
EP	Electric Power
EU	European Union
FLT	Fuzzy Logic Toolbox
FRBC	Fuzzy Rule-Based Classifier
FRM	Fuzzy Reasoning Method
FS	Fuzzy System
GA	Genetic Algorithm
GFS	Genetic-fuzzy system
HFRBC-GA	GA-optimized Hierarchical approach to Fuzzy Rule-Based Classifiers
HVAC	Heating, Ventilation and Air Conditioning
KB	Knowledge Base
MAPE	Mean Absolute Percentage Error
MCF	Multiple Certainty Factor
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NARX	Non-linear Auto-Regressive with eXogenous input
PRTools	Pattern Recognition Tools
PV	Photovoltaic
R	Coefficient of determination in Regression
RB	Rule Base
RMSE	Root Mean Squared Error
RT	Relevance Threshold
SCF	Single Certainty Factor
TD	Time Delay

List of Figures

List of Tables

Introduction

In this thesis Computational Intelligence (CI) techniques are employed in applications regarding energy systems. Many papers exist concerning applications of CI to perform forecasting, classification, and pattern recognition in the energy field (e.g., heating, ventilation and air-conditioning (HVAC) systems, power-generation, load forecasting, building's energy consumption, wind speed forecasting, solar irradiation estimation, etc.)

Computational intelligence includes several techniques, e.g., artificial neural networks, genetic algorithms, expert systems, and fuzzy systems. Each kind of technique allows building systems suited to solve a certain class of problem. Moreover, various hybrid systems may be created, as combinations of two or more of the systems previously mentioned, to exploit the advantages of both the techniques simultaneously. In addition, a brief description of forecasting through time series analysis is presented.

The thesis is organized as follows. Chapters 1 through 3 provide an overview of the computational intelligence techniques adopted in this thesis. In particular, Chapter 1 recalls main theory concepts about neural networks, describes the classical multi-layer perceptron neural network and the non-linear auto-regressive with exogenous input neural network model (NARX). In addition, some notions about forecasting and time series analysis are briefly recalled. Chapter 2 regards fuzzy rule-base classification systems. Particularly, the fuzzy rule-based classifier `frbc` is presented, along with the fuzzy reasoning method and the methodology for automatic generation of rules from data, that `frbc` adopts. Finally, Chapter 3 provides an overview about genetic algorithms and a brief description of genetic-fuzzy hybrid systems. Chapters 4 through 6 contain the novelty of the thesis, i.e., the developed methodologies and the experimental results achieved. Two main problems are addressed. First, the management of energy production in solar photovoltaic (PV) installations, by classification and forecasting, and, second, the forecasting of energy consumption in buildings.

The interest for the first problem arises from the necessity of forecasting and/or classification tools for the analysis of energy production in solar PV installations. In fact, solar energy is becoming a valid alternative to traditional energy and, as a consequence, PV installations have spread in recent years. In addition, the monitoring of the performance of solar panels has become a key issue for the improvement of the efficiency of the PV installation, as well as for finding faults or for efficiently planning the energy distribution. Among other things, PV-based power generators are discontinuous energy sources, owing to the influence of weather conditions. For all these reasons, a set of management tools is needed to correctly exploit the PV installation productivity.

The problem is tackled from two different points of view. On the one hand, we propose a general methodology to forecast solar energy production with a forecasting horizon of one day. The forecasting system, presented in Chapter 4, consists of a NARX time series analysis model implemented using a feed-forward neural network with tapped delay lines. The system, starting from some knowledge about solar irradiation, is able to faithly reproduce the instantaneous curve of daily produced energy, as well as to estimate the total daily produced energy. From another point of view, a further issue in PV energy production management is the lack of a fuzzy approach to data classification to make the final user able to make decisions easily regarding energy production management. In this way, even the non-expert user of PV systems might be able to understand the results and make smarter decisions, as we deal with class labels. Regarding this issue, we developed in Chapter 5 a fuzzy rule-based classification system aimed to classify the energy produced by a PV panel based on two environmental variables, i.e., the irradiation and the temperature of the panel. At the same time, we propose a novel hierarchical method to construct fuzzy classifiers, by performing an input domain space analysis with the aim of generating an optimal fuzzy rule base avoiding the generation of too many, unnecessary rules. The developed model results from the merging of a number of fuzzy systems built on input domain regions increasingly smaller. Each fuzzy system is developed exploiting the fuzzy rule-based classifier `frbc`. The model is actually a genetic-fuzzy system, as the model parameters are optimized by a real-coded genetic algorithm.

The motivation for dealing with the second problem, i.e., the forecasting of energy consumption in buildings, stems from considerations about the large amount of energy consumed in buildings, also in reference to the political campaigns concerning energy efficiency and energy savings promoted by several countries. The chance of knowing building's energy consumption in real time or even in advance could bring several benefits, such as cost reduction, energy management and control, and load scheduling in the electrical grid. Chapter 6 is devoted to address this problem. In particular, we refer to the electric lighting energy consumption in offices. The reason is that it is well known that electric lighting energy consumption is a big component of office buildings energy consumption. The proposed method uses an artificial neural network to forecast the average energy consumption on a time interval of a few hours, exploiting mainly the information about natural daylight, in terms of solar irradiation. The novelty of the proposed method stands mainly in the design of the forecasting system, which does not need any kind of information about the building to estimate its consumption.

Finally, Chapter 7 provides the thesis conclusions and future work, and Appendix A reports a guideline on how to implement a generic classifier (such as `frbc`) in the PRTools framework.

The research presented in this thesis was developed entirely using the toolboxes existing in the Matlab® environment. Additionally, the PRTools toolbox was used for pattern recognition concerns.

1

Artificial neural networks

1.1 Introduction

Artificial Neural Networks (ANNs) are data-driven intelligent systems having the capability to learn, remember and generalize. They were created to reproduce the learning process of the human brain by learning the relationship between input parameters and output variables based on previously recorded data [13, 98].

The human brain is a complex calculator, non-linear, massively parallel with abilities like learning, generalization and adaptability. Moreover it is fault tolerant. It is constituted by an extremely large number of simple processing elements (biological neurons) with many interconnections, thus being able to perform complex computations.

Thus, artificial neural networks have been developed following the structure and the functioning of biological neurons in the human nervous system.

Neural networks are widely applied in areas such as prediction, classification, recognition and control. Applications of artificial neural networks are in many fields: pattern classification, clustering, function approximation, prediction, optimization, and control.

In this chapter, a brief introduction to artificial neural networks is presented in Sections 1.2 and 1.3. Then, in Section 1.4 we address the multilayer perceptron neural network and the *backpropagation* training algorithm. Next, in Section 1.5, we present a description of main concepts about time series forecasting and finally we present a neural network model suited for forecasting purposes.

1.2 The biological neuron

A *neuron* is a special biological cell that processes information. It is composed of i) a cell body called *soma,* ii) many branched extensions called *dendrites*, through which the neuron receives electricity signals from other neurons, and iii) a filamentous extension, called *axon*, through which the electrical signals are transmitted to other neurons. The point of connection between two neurons (the terminal of the axon of one neuron and the dendrite of another one) is called *synapse*. A simplified model of the biological neuron is depicted in Fig. 1.1.

As we said, a neuron receives signals (impulses) from other neurons through its dendrites and transmits signals generated by its cell body along the axon. We may refer to the dendrites as the inputs of the neuron, while to the axon as the output of the neuron.

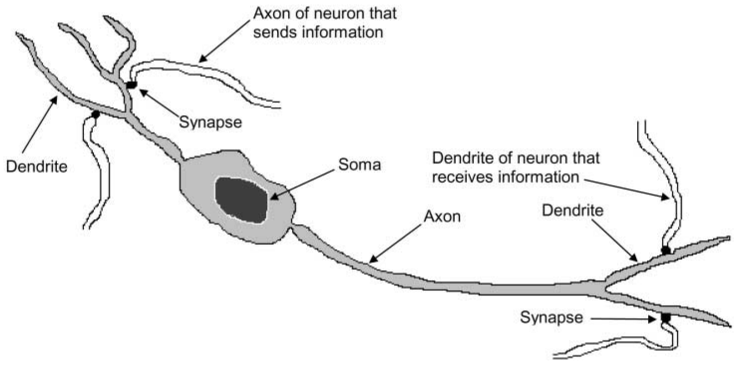

Axon of neuron that sends information
Synapse
Dendrite
Soma
Dendrite of neuron that receives information
Axon
Dendrite
Synapse

Figure 1.1. – *A simplified model of the biological neuron.*

A neuron is activated by electric impulses coming from other neurons when an electric potential difference between the inside and the outside of the cell occurs. Then, if the sum of received inputs exceeds a certain threshold, the neuron fires an electrical impulse along its axon. This electrical impulse causes the release of certain chemicals, called neurotransmitters, from the terminals of the axon, which in turn may influence other neurons. The neurotransmitters diffuse across the synaptic gap, to enhance or inhibit (depending on the type of the synapse) the tendency of the receptor neuron to fire electrical impulses.

Further, the brain is able to adjust the connections between the neurons based on its experience, that is, it is able to learn.

In the brain, the various areas cooperate, influencing each other and contributing to the achievement of a specific task, without the need for a centralized control. In addition, the brain is fault tolerant, that is, if a neuron or one of its connections is damaged, the brain continues to function, although with slightly degraded performance.

1.3 An overview of neural networks

As already stated, an ANN is a collection of simple processing units individually interconnected. In its basic computational form, a neuron appropriately processes a set of input signals coming from other neurons or sources [118].

ANNs resemble the human brain in two ways: first the knowledge is acquired by the network through a learning process, i.e., the training, then it is stored by adjusting the synaptic weights [58].

To artificially reproduce the human brain we need to build a network of very simple elements having the same characteristics of biological neurons:

a) a parallel and distributed structure;

b) the capability of learning from previous experience and thus to generalize, i.e., to produce outputs corresponding to inputs not encountered during training;
c) a graceful degradation (fault tolerance) capability.

ANNs operate like a “black box”, in the sense that they do not require any information about the system to represent a non-linear relationship between input and output variables, any time a new input set is under examination.

Several algorithms exist to set the weights in order to make the outputs match the desired result. *Supervised* learning algorithms adjust network weights using input-output data. The most frequently used supervised algorithm is the well-known *backpropagation* algorithm [131]. *Unsupervised* learning algorithms change weight values according to input values only, so this mechanism is also called self-organization.

1.3.1 The perceptron

The simplest form of neural network is the *perceptron* formed by a single artificial neuron with adjustable synaptic weights and bias. The weights represent connection strengths, and their values are established during the training process. The perceptron, developed by Rosenblatt in 1958 [128], receives input signals from other neurons through its incoming connections, it calculates the weighted sum of the inputs (i.e., the sum of the products of the weights and the inputs) and the result is passed through an activation function (e.g., the sigmoid function with bias *b*). If this value is above *b* the neuron fires and takes the activated value, otherwise it takes the deactivated value. The scheme of the perceptron is depicted in Fig. 1.2. More in detail, the relation between inputs and output is expressed by the following equation:

$$y = f_b\left(\sum_{j=1}^{N} x_j w_j + b\right), \tag{1.1}$$

where f_b is the *activation function* having bias b, x_j, (j = 1, …, N) is the j-th input to the neuron, and w_j is the weight associated with the j-th input.

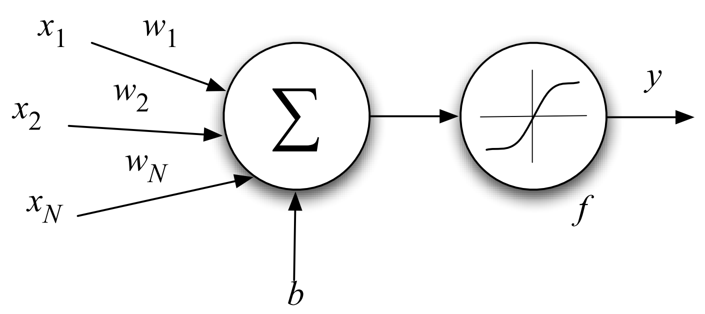


Figure 1.2. – *The scheme of the perceptron.*

Due to its simplicity, the perceptron can only solve linearly separable classification problems. However, by using more than one perceptron together, we may correspondingly perform non-linearly separable classification problems.

The most used ANN architecture and training algorithm are, respectively, the multi-layer feed-forward neural network, which includes one or more hidden layers, and the Levenberg-Marquardt *backpropagation* (abbreviation for "backward propagation of errors") training algorithm, which shows good generalization capability and simplicity [143].

In this thesis two kinds of neural networks, namely the Multi-Layer Perceptron (MLP) neural network and the Non-linear Auto-Regressive with eXogenous input (NARX) neural network, have been used for energy analysis, classification and forecasting, so they will be described in depth in the following sections. Besides, the literature about the neural network's main concepts is very extensive [16, 46, 121, 127].

1.4 The multilayer perceptron neural network

A multilayer perceptron neural network is a feed-forward network model which may represent a non-linear mapping between an input vector and an output vector. It is obtained connecting an arbitrary number of perceptrons, and thus, it consists of neurons arranged in layers, with each layer fully connected to the next one. The input signal propagates through the network in the forward direction, from the input layer to the output layer. Each connection has a weight associated with it.

In an MLP there are three kinds of layers: i) the *input* layer which receives the input signals, ii) one or more *hidden* layers where the processing takes place, and iii) the *output* layer which provides the output. Neurons in each layer are characterized by a specific activation function. The input layer merely passes the input vector to the network without any computation. Neurons in the hidden layers, referred to as *hidden neurons*, usually have a non-linear activation function. The number of hidden neurons is chosen experimentally to minimize the average error across all training patterns.

Figure 1.3 depicts an MLP network with two inputs, two outputs, and one hidden layer having four hidden neurons.

Multiple layers of neurons with non-linear transfer functions allow the network to learn non-linear relationships between input and output vectors. However, the universal approximation theorem has proved that an MLP with a single hidden layer having the sigmoid as activation function, can almost approximate any function that maps an input interval of real numbers to an output interval of real numbers [32, 58]

For the output layer, linear activation functions are often used. However the transfer function depends on the kind of the problem the MLP has to solve: a linear transfer function is used, e.g., for function fitting problems, while a sigmoid transfer function is more suited for pattern recognition problems to constrain the output of the network to assume values in a predefined range, so as to identify classes.

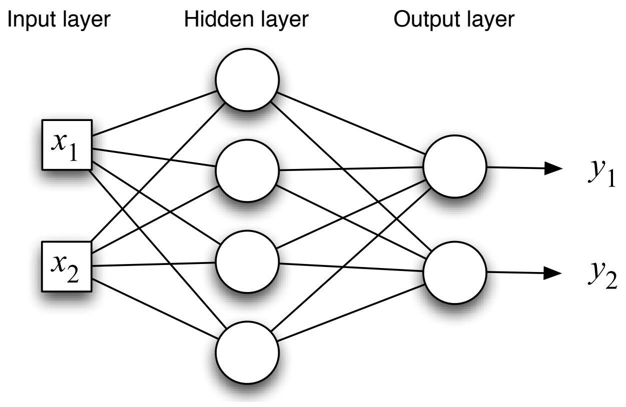


Figure 1.3. – *The scheme of a MLP neural network having two inputs, two outputs and one hidden layer with four neurons.*

The most commonly used activation functions are summarized in Table 1.1. In this thesis we adopted an MLP neural network to forecast the energy consumption in a building, as better explained in Chapter 6.

Table 1.1. – *Some common activation functions.*

Activation function	**Equation**
Sigmoid	$f(z)=\frac{1}{1+e^{-z}}$
Hyperbolic tangent	$f(z)=\tanh(z)$
Linear	$f(z)=az+b$

1.4.1 The *backpropagation* training algorithm

The most popular training technique of an MLP network is the well-known *backpropagation* training algorithm [131], a supervised learning technique that looks for the global minimum of the error function in the weight space using the method of gradient descent.

The idea is to present the network a set of matched input and desired output patterns, called *training set*. The output given from the network for each training pattern is compared with the desired output, by evaluating the error. This error is used to adjust the weights in the network so as to reduce the overall error of the network.

After providing the network with a cycle of training patterns (*epoch*), the process is repeated many times until the output of the network produces a satisfactory error. Then, the weights are held and from now on the trained network is able to generalize and correctly answer to new, unseen input data.

The combination of weights which minimizes the error function is considered to be a solution of the learning problem.

The *backpropagation* algorithm can be executed in two versions: *online* or *batch*, dependently on the way the network weights are adjusted. In the former, the weights are adapted after each pattern has been presented to the network, while in the latter, they are adapted at the end of each epoch.

The *backpropagation* algorithm is the most computationally straightforward algorithm for training the multilayer perceptron [49]. Some problems may occur during the training, thus they will be briefly described below.

The network may be trapped in a local minimum of the error function. In fact, the error surface can contain more than one minimum, i.e., a global minimum and a few local minima. Two learning parameters (*learning rate* and *momentum*) should be adjusted in order to avoid the problem. The parameters act on the step size used during the iterative gradient descent process.

Another kind of problem that may occur during the training of a neural network is *overfitting*. It occurs when the error on the training set is very small, while the error on some new patterns not presented during training is extremely large. The reason for this is that the network has learned perfectly the training examples, but has not learned how to generalize, thus the error on new data easily grows.

The solution to this problem is the *early stopping* method: this method trains the network with only a part of the available data (*training set*), while the remaining data are split into two sets, namely, *validation set* and *test set*. During the training process, patterns from the training set are used to update the weights in the usual way, while patterns from the validation set are presented to the network to check its generalization capability when the training is still in progress. As soon as the error on the validation set starts to grow, and continues to grow for a given number of epochs, it means that *overfitting* has started, so the training is stopped and the weights corresponding to the minimum validation error, before the occurrence of *overfitting*, are held.

Finally the generalization capability of the trained network is tested on the test set.

1.5 Neural networks for time series forecasting

Prediction or forecasting is a special type of dynamic filtering in which past values of one or more time series are used to predict future values of the unknown time series. Formally, a time series is a set of values that describes the evolution of a phenomenon over time, stemming from a process for which a mathematical description is unknown. Thus, usually the future behavior of a time series cannot be exactly predicted, indeed, it can be estimated [41].

Several mathematical models are present in the literature to solve prediction problems, such as regression analysis [29, 56, 81], time series analysis [57], and neural networks [9, 96, 138].

In time series modeling, the *Non-linear Auto-Regressive model with eXogenous inputs* (NARX) is an important class of discrete-time non-linear systems, where the current value of an unknown (endogenous) time series is related to the past values of the same series, and to the current and past values of the exogenous (independent) time series. More in detail, this model allows to forecast the future values of the output time series $y(t)$, knowing the past values of the same endogenous series $y(t)$ and the past values of the exogenous series $x(t)$. The equation model is the following:

$$y(t) = f(x(t-1),...,x(t-d_x), y(t-1),..., y(t-d_y)), \qquad (1.2)$$

where, $x(t\text{-}i)$ and $y(t\text{-}i)$ denote the exogenous input and the output of the model at a previous discrete time step i, respectively, and d_x *and* d_y denote the maximum delays considered for the two time series. Though these values can be different for the two time series involved, usually $d_x = d_y = d$ is the preferred choice.

The NARX model can be implemented in *open loop* mode or *closed loop* mode. In the *open loop* mode, the system computes the output without using the feedback. Generally, to obtain a more adaptive control, it is necessary to feed the output of the system back as input. This type of system is called a *closed loop* system.

Figure 1.4 shows the model block diagram in *closed loop* mode. This means that the output of the system is fed back as input.

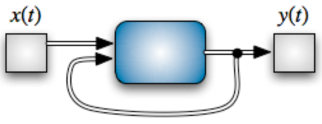


Figure 1.4. – *NARX model block diagram in closed loop mode.*

The NARX model is one of the most general time series analysis models [87, 88], as it is the straightforward generalization of the linear Auto-Regressive (AR) model to the non-linear case.

In Equation (1.2), the *forecast horizon*, i.e., the number of data points to be forecasted, is equal to one. The prediction can also involve several steps ahead. If the time horizon is greater than one, two different forecasting schemes exist: i) an *iterative* scheme, which aims to predict a variable at a given time step and the obtained prediction is used as input for the forecasting of the next time step; ii) a *direct* scheme, which aims to predict the next n time steps from the same input data [145].

The time series prediction process usually involves 5 steps [41]:

1. preprocessing of the data: perform smoothing or normalization, remove the outliers;

2. identification of the model: select the architecture, set an appropriate number of layers and hidden neurons for each layer, set learning parameters values;
3. training of the network (usually in open loop mode);
4. validation of the trained network: compute the error, check the prediction ability of the network;
5. use of the network for forecasting (usually in closed loop mode).

Regarding previous step 4, several metrics exist to evaluate the error and the goodness of the network.

The most frequently used are: the Mean Absolute Percentage Error (MAPE), the Mean Square Error (MSE), the Root Mean Square Error (RMSE). To assess the goodness of the forecasting model, also the *coefficient of determination* R^2 can be used. It indicates how well a regression line fits a set of data. It provides a measure of how well future outcomes are likely to be predicted by the model. R^2 near 1 indicates a good fitting of the model to the data, R^2 closer to 0 indicates bad fitting.

The following equation expresses the coefficient of determination:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}, \tag{1.3}$$

where, n is the number of data, x_i and $\hat{x}_i$ denote the actual and predicted values of data, and $\overline{x}$ is the mean of the actual data.

The numerator is called *residual sum of squares* and it is a measure of the variability of the forecasting error. The denominator is called *total sum of squares* and measures how much variation there is in the data (with respect to their mean value).

1.5.1 The NARX neural network

ANN models may be used as alternative methods to autoregression models in engineering analyses and predictions [77, 92]. Thanks to their ability to model complex non-linear functions or unknown functions by learning from examples, neural networks are particularly suited to implement NARX models.

A NARX neural network is the network used to implement a NARX model. It is a special kind of Time Delay (TD) neural network, i.e., a feed-forward network with a tapped delay line associated with inputs and output, particularly suited to time-series prediction.

The non-linear function f of Equation (1.2) is generally unknown and can be approximated, for example, by a neural network. The resulting architecture is then called a NARX neural network and it is shown below in Fig. 1.5.

Considering Fig. 1.5, the output of the neural network is the predicted next value in the output time series, computed as a function of the exogenous time series and the output time series, which is fed back as endogenous input to the network. Two delays are used on both the time series. The hidden layer presents ten neurons. The associated model equation is $y(t) = f(x(t-1), x(t-2), y(t-1), y(t-2))$.

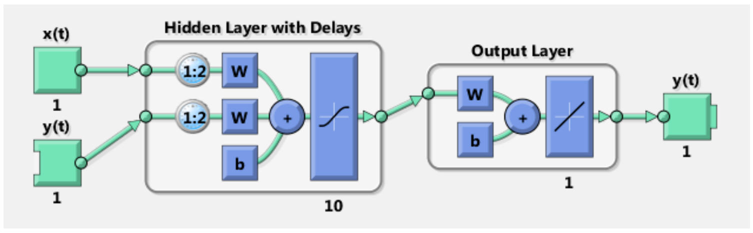


Figure 1.5. – *The scheme of a simple NARX neural network (the figure was produced in the Matlab® environment).*

In this thesis we adopted a NARX neural network used with the iterative scheme to forecast the energy production in a photovoltaic installation, as better described in Chapter 4.

2

Fuzzy systems

2.1 Introduction

Fuzzy systems (FSs) are methodologies based on fuzzy set theory to represent and process linguistic information and have been introduced by Zadeh [164].

The mathematical theory of *fuzzy logic* is used to simulate the process of human reasoning, i.e., approximate reasoning. Fuzzy logic is an extension of classical logic (in which any statement is either true or false) where several degrees of truth are allowed [22]. A *fuzzy set* is a generalization of a classical set in which the membership function is a continuous function with values in the interval [0, 1] in place of two discrete values {0} and {1} of a classical set.

A set of fuzzy rules is used to specify an input-output mapping between the input variables (i.e., linguistic variables) and the output class.

Each rule is represented as a conditional statement of the kind "**if** premise **then** consequence", where the premise and the consequence are expressed in terms of fuzzy sets and linguistic variables. Rules are usually fixed basing on some expert's previous knowledge, or drawn directly from data. In fact, extracting fuzzy rules from data allows modeling the relationships existing in the data by means of if-then rules that are easy to understand, verify and extend [24].

Let us now spend a few words about the main benefit of fuzzy systems: interpretability. Sometimes it is interesting not only to have an accurate classifier at our disposal, but also to have an easily interpretable classifier capable to let understand its operating ("white box" model). For this reason fuzzy systems are widely adopted in real-modeling problems where vagueness and uncertainty can be represented by using linguistic variables, instead of classical variables, thus obtaining more interpretable results.

However, not all the fuzzy classifiers are necessarily interpretable. According to the recent literature [7, 109], there are many concerns to be addressed and constraints to be included during the design stage to guarantee interpretability. Typically, interpretability of fuzzy rules depends on three issues: i) the fuzzy partition should be both complete and distinguishable (i.e., understandable linguistic terms should be easily assigned to the fuzzy sets of the partition); ii) the fuzzy rules in the rule base should be consistent, i.e., there should not be any conflicting rules; iii) the number of variables in the rule premise should be as small as possible [94].

Fuzzy systems have been applied to a wide variety of fields ranging from control, signal processing, communications, classification and soft computing.

The chapter has the following outline. Section 2.2 recalls the main concepts of fuzzy classifiers, while Section 2.3 presents the Fuzzy Rule-Based Classifier (FRBC) employed in this work, i.e., the generation of the rule base from data and the fuzzy inference method adopted.

2.2 Overview of Fuzzy Rule-Based Classifiers

Fuzzy rule-based classifiers (FRBCs) consist of two main components: the Knowledge Base (KB) and the Fuzzy Reasoning Method (FRM). The KB is made up by a Rule Base (RB) and a Data Base (DB). The former contains a set of fuzzy if-then rules, while the latter contains the semantic parameters of the fuzzy sets that model the linguistic variables used in the fuzzy rules. Each rule specifies a subspace of the pattern space using the fuzzy set in the antecedent part in the rule. The FRM maps any input pattern X to its predicted class C using the information provided in the KB. An FRBC scheme is represented in Fig. 2.1.

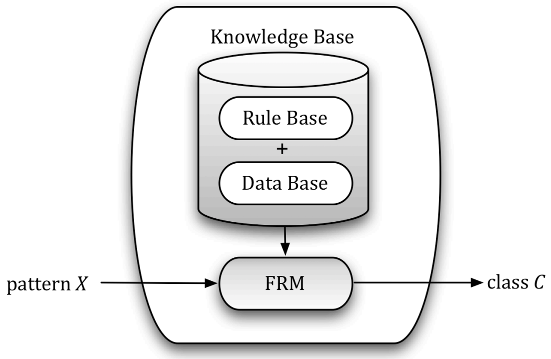


Figure 2.1. – *A fuzzy rule-based classifier.*

The design of a FRBC consists basically in finding a compact set of fuzzy if-then classification rules derived from human experts or from domain knowledge, able to model the input-output behavior of the system. The next step is to combine these rules in order to associate a given input to a single conclusion which is the output.

In the following we show three typical formulations of fuzzy if-then rules for classification.

Let us consider an M-class classification problem related to M classes $\{C_j\}_{j=1}^{M}$ in an F-dimensional input space, and a set of (F+1) fuzzy partitions $P_f = (A_1^f, ..., A_{Q_f}^f)$, f=1,…(F+1), on F input variables $X_1,…,X_f$ and one output variable $X_{(F+1)}$; each partition consists of Q_f membership functions. Moreover, let $\{\mathbf{x}^t, o^t\}_{t=1}^{N}$ be a set of N

input-output pairs to be classified, where $(x_1^t,...,x_F^t)$ is the feature vector associated with pattern $\mathbf{x}^t$, and o^t the associated class index.

Under the previous assumptions, an FRBC rule base made of a set of L rules $\{R_k\}_{k=1}^L$ can be expressed in different ways. In the next we report the three most common kinds of rules used for pattern classification problems [30]:

i) fuzzy rules with a single consequent class:

$$R_k : \textbf{if } X_1 \textit{ is } A^1_{\delta_{k,1}} \textit{ and} ... X_F \textit{ is } A^F_{\delta_{k,F}} \textbf{ then } X_{(F+1)} \textit{ is } C_{\delta_{k,(F+1)}}, \tag{2.1}$$

where k is the rule index, $\delta_{k,f}$ is the index of the fuzzy set to be used for variable f in rule k, f = (1,…, F), and $C_{\delta_{k,(F+1)}}$ means that the output class associated with rule k is the one having index $\delta_{k,(F+1)}$;

ii) fuzzy rules with a single consequent class and a single certainty factor (SCF) γ_k, i.e., the weight associated with the rule, with a value in the interval [0, 1]:

$$R_k : \textbf{if } X_1 \textit{ is } A^1_{\delta_{k,1}} \textit{ and} ... X_F \textit{ is } A^F_{\delta_{k,F}} \textbf{ then } X_{(F+1)} \textit{ is } C_{\delta_{k,(F+1)}} \textit{ with } \gamma_k; \tag{2.2}$$

iii) fuzzy rules with multiple consequent classes and multiple certainty factors (MCF) γ_k^j (j = 1,…, M):

$$R_k : \textbf{if } X_1 \textit{ is } A^1_{\delta_{k,1}} \textit{ and...and } X_F \textit{ is } A^F_{\delta_{k,F}} \textbf{ then } X_{(F+1)} \textit{ is } C_1 ... C_M \textit{ with } \gamma_k^1 ... \gamma_k^M. \tag{2.3}$$

The degree of satisfaction $\mu^{k,t}_{f,\delta_{k,f}}$ of the generic condition "X_f *is* $A^f_{\delta_{k,f}}$", corresponding to pattern $\mathbf{x}^t$, is computed in the same way for the three kinds of rules, as the membership function value associated with fuzzy set $A^f_{\delta_{k,f}}$, in the f-th component $x_f^t : \mu^{k,t}_{f,\delta_{k,f}} = A^f_{\delta_{k,f}}(x_f^t)$.

2.3 The implementation of a FRBC: `frbc`

In this thesis, we will consider only rules with a single consequent class and a single certainty factor (see Equation (2.2)), which represents the certainty degree of the classification in the specified class for an input pattern belonging to the fuzzy subspace identified by the rule antecedent. These rules provide the best tradeoff between flexibility and complexity. Furthermore they naturally fit the Matlab® Fuzzy Logic Toolbox (FLT) rule base structure.

The FRBC used in the present thesis, presented in [28] was developed under the Pattern Recognition Toolbox (PRTools) [42], the *de facto* standard toolbox for classification in Matlab®. `frbc` follows the PRTools base philosophy, e.g., use of fast and heuristic-based training algorithms, function reuse, powerful and concise syntax, automatic training from data, total compatibility with the Matlab® environment. A brief description of PRTools framework and the instructions on how to implement a new classifier in PRTools are presented in Appendix A, and supporting material can be found in [42, 151].

In the following sections we explain the method employed to automatically learn the KB form data and the general model of fuzzy reasoning used.

2.3.1 The rule base construction

The main issue in the development of fuzzy classifiers is the proper training, that is, the creation of an efficient rule base. The rule base can be derived from the expert previous knowledge or more easily can be derived directly from numerical input-output pairs. In the literature, many approaches have been proposed for generating fuzzy rules from numerical data, such as heuristic approaches [2, 70, 93], neuro-fuzzy techniques [103, 110, 111, 112, 150], clustering methods [3, 130], and genetic algorithms [19, 51, 68, 73].

In order to meet the PRTools base philosophy (see Appendix A), we are interested in existing batch-mode oriented approaches to automatically generate fuzzy rules from data, that are well-assessed and widely accepted, that need few free parameters to be specified, and that are associated with fast heuristic training methods. In our opinion, among the approaches existing in the literature, the technique that best meets these requirements is the Wang and Mendel method extended to classification problems, an adaptation of the well-known Wang and Mendel method for regression problems [154]. This method assumes that the fuzzy partitions of the input and output variables are provided by the user. A typical and easy way to achieve this is to resort to uniform fuzzy partitions consisting, for input variables, of a limited number of fuzzy sets, while, for output variables, of as many fuzzy sets as there are classes. So, the Wang and Mendel method is the training technique available in `frbc` [28]. This method seems to be simpler and with less construction time than a comparable neural network, maintaining the comparability of the results [106]. In addition, it allows to combine in the same framework both numerical and linguistic information [154].

In the following, the formal steps of the Wang and Mendel algorithm extended to classification problems are briefly introduced:

STEP A.	generate a uniform fuzzy partition of the input domain, made of Q_f, $f = 1,\dots, F$, fuzzy sets for each input variable f (usually $Q_f = Q = 3, 5, 7$ or 9);
STEP B.	generate a uniform fuzzy partition of the output variable, made of M fuzzy sets (one set for each class);
STEP C.	generate an initial raw rule base (one rule for each training pattern);
STEP D.	remove duplicated rules;
STEP E.	compute certainty factors;
STEP F.	remove conflicting rules (i.e., rules having the same **if** part but different **then** parts) by keeping, for each set of conflicting rules, only the rule with the highest CF.

More in detail, this algorithm generates the fuzzy rule base assuming a uniform fuzzy partition for the input variables. Thus, the domain of each input variable is divided into $Q = 2N + 1$ regions, typically $N \in \{1,2,3,4\}$ (STEP A). The length and the number of the regions may be different for the considered variables. A fuzzy membership function is then defined for each region. Typically this function has its maximum value in the middle point of the region and assumes its minimum value in the central points of the two neighboring regions, although different definitions are possible. Figure 2.2 shows an example of a fuzzy partition built according to the Wang and Mendel approach, on a generic variable y whose domain interval $[y^-, y^+]$ has been divided into Q=5 regions (N=2).

Doing so, the thresholds identify a grid on the input variable space. So, for instance, if we have two input variables and we define two evenly spaced thresholds for each variable, we obtain a uniform 9-area grid.

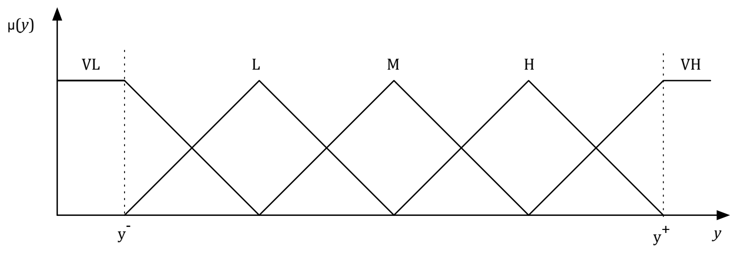


Figure 2.2. – *An example of a fuzzy partition for the input variables, built according to the Wang and Mendel approach.*

Regarding STEP B, the fuzzy sets used to represent the output partition are usually fuzzy singletons, as we deal with a classification problem. Since each data pair generates a fuzzy rule in the rule base (STEP C), there will possibly be some duplicate rules and some conflicting rules, i.e., rules having the same **if** parts but different **then** parts. The duplicated rules are simply deleted (STEP D), while to solve a conflict, a CF is assigned to each conflicting rule of a set. The CF is defined so as to take into account the importance of each rule in the entire rule base (STEP E). The winning rule, within a conflicting set, is the one that has the maximum CF. The other rules of the set are discarded (STEP F).

From the field of data mining [5], the CF is tipically computed as the confidence of the fuzzy association rule $A_k \Rightarrow C_{\delta_k}$, corresponding to the fuzzy rule R_k: **if** A_k **then** C_{δ_k}, where A_k represents the antecedent part of the fuzzy rule, and C_{δ_k} the class appearing in the consequent. The CF is calculated as follows:

$$\gamma_k = \sum\nolimits_{t:\mathbf{x}^t \in class\, C_{\delta_k}} \Omega^{k,t} \Big/ \sum\nolimits_{t=1}^{N} \Omega^{k,t}, \tag{2.4}$$

where $\Omega^{k,t}$ is the strength of activation of the antecedent of rule R_k for the t-th pattern, and N is the number of training patterns.

However, in past research, many heuristic measures have been proposed to specify the weight of a fuzzy classification rule [72, 93, 166]. Nozaki *et al.* [115]

proposed a method of learning rule weight using Reward & Punishment, in which, considering the classification of a pattern using the single winner FRM, the weight of the winner rule is increased or decreased depending on whether the pattern has been correctly classified or not. In other relevant methods [66, 72], the computation consists of two phases. First, the certainty factor is calculated as the confidence of the fuzzy rule (as in Equation (2.4)), then, the certainty factor is refined with a measure that depends on the specific method, with the aim to improve the classification performance.

We recall that any shape and number for membership functions can be selected. Clearly, the higher the number of membership functions, the bigger the accuracy obtained. On the other hand, a large number of membership functions leads to a large rule base dimension and, consequently, to a higher complexity.

2.3.2 The fuzzy reasoning method

The FRM available in `frbc` [28] for MCF rules is a general model of fuzzy reasoning for combining information provided by different rules. It is an extension presented in [30] of the fuzzy classifier defined by [82].

In the following, we recall the steps of the FRM applied to each input pattern $\mathbf{x}^t$:

STEP 1.	determine, for each rule, the strength of activation of the antecedent, say *matching degree*;
STEP 2.	compute, for each rule, the *association degree* of the pattern with the class specified by the rule;
STEP 3.	compute, for each rule, the *stressed association degree* by emphasizing the association degree;
STEP 4.	determine the *soundness degree* of the classification of the pattern $\mathbf{x}^t$;
STEP 5.	assign pattern $\mathbf{x}^t$ to the class that has the maximum soundness degree.

For each step of the inference process, several operators can be selected, thus giving origin to different inference methods.

In particular, as regards STEP 1, the matching degree Ω for pattern $\mathbf{x}^t$ and rule R_k is calculated as the AND operator (any T-norm) between the membership function values:

$$\Omega^{k,t} = \mathrm{I}_{f=1}^{F} \mu_{f,\delta_{k,f}}^{k,t}, \quad k = 1,\dots,L, \quad t = 1,\dots,N. \tag{2.5}$$

The AND operators available in `frbc` are the minimum and the product.

In STEP 2, the association degree is computed by applying a combination operator h to the matching degree $\Omega^{k,t}$ and the certainty factor γ_k as follows (possible choices for h are product and minimum):

$$b^{k,t} = h(\Omega^{k,t}, \gamma_k), \quad k = 1,\dots, L, \quad t = 1,\dots,N. \tag{2.6}$$

In STEP 3 the association degree is stressed by applying a stress function g so as, e.g., to increase higher values and decrease lower ones:

$$B^{k,t} = g(b^{k,t}), \quad k = 1,\ldots, L\,,\ t = 1,\ldots,N. \tag{2.7}$$

We have considered two stress functions, namely *No_Stress* function g_1 (identity function) and *Square_SquareRoot* function g_2, as defined hereafter:

$$g_1(z) = z \ \ \forall z \in [0,1] \tag{2.8}$$

$$g_2(z) = \begin{cases} z^2 \ if \ z < 0.5 \\ \sqrt{z} \ if \ z \geq 0.5\,. \end{cases} \tag{2.9}$$

In STEP 4, the soundness degree $\hat{o}_j^t$ associated with each output class j is computed by applying an aggregation function Γ to the S_j^t positive association degrees $a_j^{s,t}$:

$$\hat{o}_j^t = \Gamma(a_j^{s,t}), \quad s = 1,\ldots, S_j^t,\ j = 1,\ldots,M \tag{2.10}$$

where:

$$(a_j^{1,t},\ldots,a_j^{S_j^t,t}) = (B_j^{k,t} : B_j^{k,t} > 0,\ k = 1,\ldots, L). \tag{2.11}$$

We have considered seven different aggregation functions (maximum, normalized addition, arithmetic mean, quasi-arithmetic mean, Sowa and-like, Sowa or-like, Badd operator). Table 2.1 shows the mathematical equations of the aggregation functions [30]. For each function we indicate the value of the free parameter (if existent) used in the experiments.

In particular, the use of the maximum operator leads to the implementation of the classical FRM, which classifies a new example with the consequent of the fuzzy rule with the greatest degree of association. Although it is used by the majority of FRBCs, it loses the information provided by other rules.

Finally, STEP 5 computes the predicted class index $\hat{o}^t$, associated with pattern $\mathbf{x}^t$, as:

$$\hat{o}^t = \underset{j=1,\ldots,M}{\arg\max}\,(\hat{o}_j^t). \tag{2.12}$$

In the present thesis, `frbc` has been employed to forecast the energy production from a solar photovoltaic installation in order to help the manager of the installation in the control and the dispatch of the energy in the electrical grid, as better described in Chapter 5.

Table 2.1. – *Mathematical equations for the aggregation functions.*

Aggregation function	**Mathematical equation**	**Value of free parameter**
Maximum	$\Gamma_{MAX} = \max\limits_{j=1\ldots M}\{a_1 \ldots a_{s_j^t}\}$	-
Normalized addition	$\Gamma_{NORMADD} = \sum\limits_{i=1}^{s_j^t} a_i \Big/ \max\limits_{j=1\ldots M} \sum\limits_{i}^{s_j^t} a_i$	-
Arithmetic mean	$\Gamma_{ARIMEAN} = \sum\limits_{i=1}^{s_j^t} a_i \Big/ s_j^t$	-
Quasi-arithmetic mean	$\Gamma_{QARIMEAN} = \left[\frac{1}{s_j^t} \cdot \sum\limits_{i=1}^{s_j^t} (a_i)^p\right]^{-p}, p \in R$	p=50
Sowa and-like	$\Gamma_{SOWAAND} = \alpha \cdot \min\{a_1 \ldots a_{s_j^t}\} + (1-\alpha) \cdot \Gamma_{ARIMEAN}$, $\alpha \in [0,1]$	$\alpha = 0.5$
Sowa or-like	$\Gamma_{SOWAOR} = \alpha \cdot \max\{a_1 \ldots a_{s_j^t}\} + (1-\alpha) \cdot \Gamma_{ARIMEAN}$, $\alpha \in [0,1]$	$\alpha = 0.9$
Badd operator	$\Gamma_{BADD} = \sum\limits_{i=1}^{s_j^t} a_i^{p+1} \Big/ \sum\limits_{i=1}^{s_j^t} a_i^p$, $p \in R \cdot$	p=50

3

Genetic algorithms and hybrid systems

3.1 Introduction

Since their first introduction by Holland in 1975 [61], genetic algorithms (GAs) have attracted a lot of interest in the research community. GAs are search algorithms capable of solving a wide range of problems that traditional methods have difficulty to solve (large scale combinatorial optimization problems or complex search space with multiple optima), by using the principles inspired by natural genetics [37, 50, 100].

The basic idea is to describe the optimization problem and its solution with an individual having a set of characteristics (i.e., parameters of the problems) and then to make *evolve* a population of individuals toward the optimal solution for the problem. Usually we refer to individuals with the term *chromosome,* and to each characteristic with the term *gene*. Each chromosome is typically coded as a binary string. However, real-coded GAs have shown better performance than binary-coded GAs in many optimization problems [75].

GAs simulate the evolutionary cyclic process of a population of individuals, with each cycle representing a *generation*. Within each generation, genetic operators are applied to obtain a new population made of better individuals. The quality of each individual is measured by means of a *fitness function*, which indicates the adaptability of the individual to the environment, i.e., the probability to survive. In a GA this relates to the probability to be part of the next generation population.

The main advantage of GAs is that they do not need a mathematical description of the problem. Thanks to their nature, GAs are successfully employed in optimization problems in many areas (e.g., economics, mathematics, computational science, engineering) [50, 61]. GAs may be used alone or as part of hybrid systems (e.g., genetic-fuzzy systems, neuro-genetic systems).

In this chapter, an overview of genetic algorithms along with their functioning is presented in Section 3.2, while a brief description of hybrid systems, with particular reference to genetic-fuzzy hybrid systems, is presented in Section 3.3.

3.2 Genetic algorithms: main concepts

A genetic algorithm requires: i) a genetic representation of the solution domain,

ii) a *fitness function* to evaluate the goodness of each solution, iii) methods and associated probabilities values for recombining chromosomes (*crossover*) and reconsider possibly useful lost genetic material (*mutation*), and iv) a *selection* mechanism to choose the best chromosomes.

A GA works with populations of chromosomes, that evolve according to the natural evolution, towards a better population, thus toward better solutions. In fact, they operate simultaneously on a set of solutions rather than on one solution.

Evolution is a method of searching among a big number of solutions, by applying genetic operators similar to the corresponding in nature. Among genetic operators the most used are *crossover*, *mutation* and *selection*.

The evolution usually starts from a population of randomly generated individuals. Each new population is built with the best individuals of previous generations (selected according to their *fitness*), with the aim of propagate to next generations the best genetic heritage.

More in detail, the evolution from one generation to the other involves three steps. First, individuals of the current population are evaluated. Second, those with higher fitness are selected from the current population to form a new population in the next generation. Third, genetic operators (crossover and mutation) are applied to selected parents to generate offspring. Then the population is evaluated again. The algorithm terminates when a convergence criterion is met, i.e., a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

Figure 3.1 summarizes the functioning of a simple GA [140].

```
GA {
        generate random population;
        evaluate population;
        while termination criterion not reached {
                select solutions for next population;
                perform crossover and mutation;
                evaluate population;
        }
}
```

Figure 3.1. – *The functioning of a simple GA in pseudocode.*

3.2.1 GA operators

After the genetic representation and the fitness function are defined, a GA initializes a population of solutions and then improves it through an iterative application of the genetic operators. Genetic operators are used to make evolve the current population towards a heterogeneous set of individuals in order to obtain a global convergence and a complete exploration of the search space.

In the following, we briefly explain the GA operators: *selection*, *crossover* and *mutation*.

Selection is the process of choosing the breeding chromosomes (mating pool) in the current population for reproduction of individuals to be inserted in the next population. Since it is expected that better parents generate better offspring, parent solution chromosomes with higher fitness have a higher probability to be selected, similarly to what happens in natural selection. Several selection operators exist, and usually the selection probability is proportional to the fitness of the chromosome.

Crossover is used to combine the genes of two individuals (parents) to produce a new individual (offspring) that inherits characteristics from both parents. There are several ways to combine parent chromosomes. Crossover usually takes place according to a crossover probability (ranging from 0 to 1) that should ensure both *exploitation* (ability of convergence) and *exploration* (good ability to explore the search space). The simplest crossover is called *one-point crossover:* the parent chromosomes are split into two parts at a random position and then the left part of one is combined with the right part of the other and vice versa. Other kinds of crossover are *multi-point crossover* or *uniform crossover*.

Mutation is merely a random modification of one or more genes in a chromosome, to reintroduce the genetic material lost or to avoid the convergence to local optima. The aim of this operation is to deeply modify the chromosome, so as to explore areas of the solution space that have not yet been observed. However, to ensure the convergence of the genetic algorithm, the mutation probability is very low, i.e. in the range [0.001, 0.01]. In the literature several kinds of mutation exist (e.g., *uniform mutation*, *non-uniform mutation*, etc.).

3.3 Hybrid systems: genetic-fuzzy systems

GAs are often used to produce intelligent hybrid systems. Among existing hybrid systems, we recall only genetic-fuzzy system (GFS), as this kind of hybrid system is the one employed in this thesis.

A GFS is a fuzzy system whose parameters are optimized by a learning process based on a GA. Genetic learning processes can be involved in a fuzzy system at different levels of complexity. In the literature, many paper deal with GFSs.

In the simplest case, the GA is used to optimize some free parameters of the hybrid model [33] as done in Chapter 5 of this thesis.

In more complex cases, the GA is used to tune: i) the fuzzy partition parameters (membership functions) of the fuzzy system, as done in [134], where a GA models the linguistic labels of the fuzzy sets, ii) the fuzzy rule set of the fuzzy system, as done in [90, 137], or iii) both the fuzzy partition parameters and the fuzzy rules, as done in [165], where a GA is used to optimize shape and parameters of the fuzzy membership functions, and number and size of the fuzzy rules.

In other approaches, a GA is used to optimize the set of rules and the fuzzy reasoning method of the FRBC, thus producing single-objective and multi-objective GFS [60, 64, 65], or it is applied to perform feature selection [152].

4

One day-ahead forecasting of PV energy production by means of neural networks and time series analysis

4.1 Introduction

Renewable energy refers to energy generated by natural sources, which are naturally replenished, such as sunlight (solar energy), wind (eolic energy), and tides (tidal energy).

As conventional fossil fuel energy sources (e.g., coal, oil, gas, etc.) are diminishing and global warming is increasing [80], renewable energy, and in particular, solar energy, is receiving heightened attention as a potentially widespread approach to sustainable energy production [142], thus becoming a valid alternative to traditional energy since it is considered economical and non-polluting [160] besides practically inexhaustible [126, 142, 147, 157]. Furthermore, according to European Union (EU), renewable energies will be able to contribute between 33% and 40% to the total electricity production in Europe by 2020. Particularly, photovoltaic (PV) energy could provide 12% of European electricity demand by 2020 [43].

For these reasons PV installations have spread in recent years [116]. The solar PV total world capacity has increased dramatically: from 9.4 GW in 2007 to 23.2 GW in 2009, and to 70 GW in 2011 [126].

A photovoltaic installation consists of a series of solar panels that using sunlight energy generates directly usable electricity thanks to the PV effect. A PV panel (see Fig. 4.1) is composed in its turn of individual PV cells. Since a single PV panel can produce only a limited amount of power, normally several panels are connected together to form a generation system called PV array, to which an inverter is connected that measures the production power of that array, and converts the DC power in AC power, as requested by the electrical network. Due to their modularity, PV installations can be configured in almost any way to supply most loads. Generally, a PV installation includes one or more PV arrays, an inverter for each array, batteries, and wiring to connect all the PV installation [15, 125].

With the diffusion of PV systems the monitoring of the performance of solar panels has become a key issue, so as to detect efficiency losses or effectively plan the energy distribution, for instance in smart grid installations.

This can be done by estimating the forecasted energy and comparing it with the real produced energy.

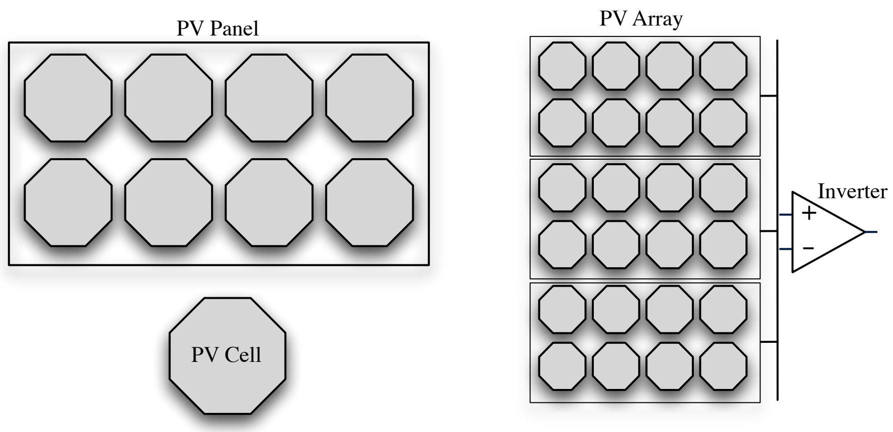


Figure 4.1. – *Photovoltaic elements: PV cell, PV panel, and PV array.*

However the main drawback of solar energy is its availability due to the unpredictability nature of solar irradiation.

Although a lot of people work in this research area, they are mainly concerned with forecasting solar radiation [12, 95, 96, 100, 107], whereas only a few are concerned with the forecasting of solar energy production directly.

The literature related to the renewable energy field presents many approaches to forecasting load, wind speed or solar irradiation. The most widely used techniques include regression methods [29, 56, 81], neural networks [9, 96, 138], and time series analysis [57]. However most of the existing methodologies present some drawbacks such as high average accuracy error, dependence on the particular design of the PV installation, and inability to provide real-time prediction [13, 119, 123].

The methodology proposed in this chapter represents a flexible and easy-to-use methodological approach to the forecasting of energy production in solar PV installations, using time series analysis and neural networks. The aim is to develop and validate a one day-ahead forecasting model by adopting an artificial neural network with tapped delay lines to implement the NARX time series model for the prediction of the energy production of the following day. The potential benefits of having energy production predictability are obvious: knowing the energy production ahead of time is useful in automatic power dispatch and load scheduling, and energy control [139].

The goal of the proposed approach is twofold. On the one hand, energy production forecasting is important for improving the efficiency of the PV installation, as well as for finding faults. On the other hand, energy production forecasting is vital for efficiently planning the energy distribution [31]. In particular, the chance to forecast the energy production up to 24 hours can become of the utmost importance in decision-making processes, with particular reference to grid-connected photovoltaic plants. Moreover, this approach could be particularly useful in smart grid systems, which are able to make better operational decisions using ahead predictions [122].

Variability of weather, in particular of solar irradiation, is maybe the main difficulty faced by PV installation operators [86] so that good forecasting tools are required for the appropriate integration of renewable energy into the power system [81]. Among various prediction models, such as analytic, stochastic, and empirical, neural networks are fairly able to correctly model the nonlinear nature of dynamic processes. Actually, an artificial neural network is able to reproduce an empirical, possibly nonlinear, relationship between some inputs and one or more outputs [14]. Forecasting is one of the most interesting nonlinear applications of neural networks. Indeed, the measurements of environmental parameters are generally provided in the form of time series which are suitable to use neural networks for prediction purposes [17]. Moreover, neural networks are fault tolerant, i.e., are able to handle noisy or incomplete data [77].

4.1.1 Outline of the chapter

The chapter has the following structure. Section 4.2 describes the experimental data collected from two PV installations located in Italy, Section 4.3 presents the proposed methodology to correctly set up the NARX-neural network model for forecasting the solar energy production. Section 4.4 presents and discusses the achieved results. Finally, concluding remarks are provided in Section 4.5.

4.2 Description of the solar PV dataset

The available data were collected from two PV ground installations of solar panels 500 meters far away from each other, located in Apulia, Italy. We employed data only from one installation for the development of the model, as, under conditions of proper working, we can assume that the two installations are correlated. We then successfully tested the developed model also on the second installation.

According to the literature, the two most significant environmental parameters are temperature and irradiation on the solar panels [9, 80, 81]. The first one refers to the surface of the panel exposed to the sunlight, while the second one is the quantity of sun radiation incident on the panel with respect to the whole surface of the panel and to all the electromagnetic spectrum frequencies.

Actually, we employed only the irradiation input, similarly to what happens in PV plant sizing, since the total solar radiation is considered as the most important parameter in the performance prediction of renewable energy systems [99]. The first input, instead, was used only for offline checks. Figure 4.2 shows the irradiation trend for six consecutive days of Winter. Days 1, 2, 5, and 6 correspond to clear or partly cloudy days, while days 3 and 4 correspond to completely overcast days. The output parameter is the produced energy from each PV installation. Data, including also the sampling timestamp (date and time), were collected every 15 minutes during 1 year, from October 2009 to September 2010.

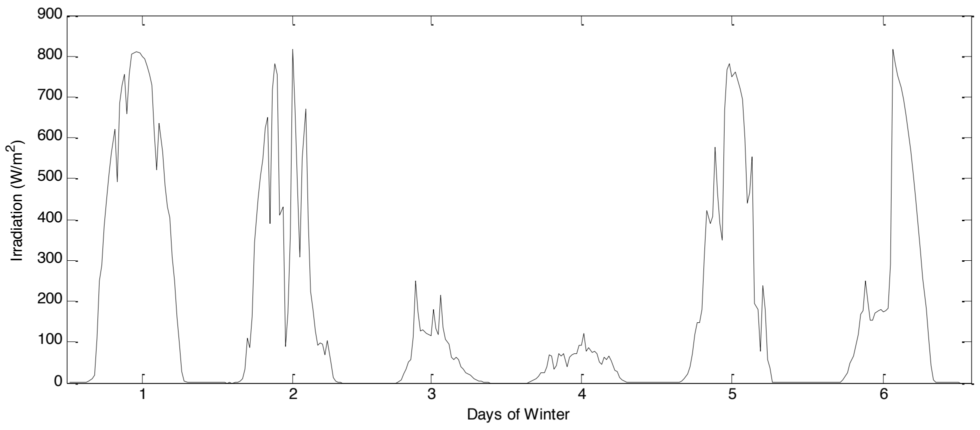


Figure 4.2. – *Solar irradiation trend for six days of Winter.*

4.3 The proposed NARX-neural network model for solar PV energy forecasting

The proposed methodology presented in [26, 27] aims to provide an easy-to-use tool for correctly configuring the best predictor for energy production in a PV installation based only on the collected historical data, with no concern about possible relationships between plant attributes (like positions, construction, etc.) and predictor parameters. More in detail, we aim to forecast the energy production up to 24 hours, given the environmental parameters of an appropriate number of previous days that compose the training window.

We decided to implement the NARX model (see Chapter 1) using a feed-forward neural network with tapped delay lines, having one hidden layer. The hidden neurons are characterized by a hyperbolic tangent sigmoid function while the output neuron has a linear transfer function.

The tasks to be performed are basically the following: i) choice of the training window width; ii) choice of the sampling frequency; iii) choice of the number of delays; iv) choice of the number of hidden neurons. In the following we will use the term *structural parameters* to refer to the number of hidden neurons and the number of delay elements, and the term *configuration parameters* to mention the training window width and the sampling frequency.

Based on heuristic considerations, we propose: i) to consider the training window width as a multiple of the day, ii) to use balanced training sets obtained by randomly extracting the same number of training samples from each day of a window, and iii) to assess the performance of the NARX model based on the mean performance obtained on all the predicted days within periods of one month. This last choice aims to avoid the strong dependence of the performance results, achieved by a given neural network, on the particular day to be predicted (which might be atypical).

In the following we will describe the operation steps for the development of the neural model.

4.3.1 Choice of the structural and configuration parameters

Based on the previous considerations, we make use of training windows having width *w* from a minimum of 7 days to a maximum of 30 days before the predicted day. Furthermore, for the sake of simplicity, we kept the 15 minute sampling frequency.

As previously stated, we decided to take into account the mean performance obtained on all the predicted days within periods of one month. We considered one month for each season. More in detail, we performed the experiments on the days of the first complete month of each season (e.g., being the beginning of winter on December 21st, January is chosen as the first complete month of winter), having a total of four months.

We heuristically decided to try a number of hidden neurons (*h*) ranging from 8 to 20 with step 1, and a number of delays (*d*) ranging from 3 to 10 with step 1.

Each kind of experiment has been repeated 30 times, once fixed the neural network configuration and structure, to mitigate the effects, in terms of convergence, of the random initialization typical of neural networks.

For each month considered, we found a set of windows that appear most often, i.e., at least in a given percentage (80% in our case) of the trials of each experiment, as the best windows among the 24 possible windows. This set is $w = 9$, 12, 15, 18, 21.

The best results were obtained with a neural network with 10 hidden neurons, and 3 delay elements on both the exogenous (irradiation) and endogenous (produced energy) variables. These values represent the best compromise between efficiency and simplicity. In particular, to set the number of delays to 3 means to use the information pertinent to the three quarters of hour immediately before each predicted sample.

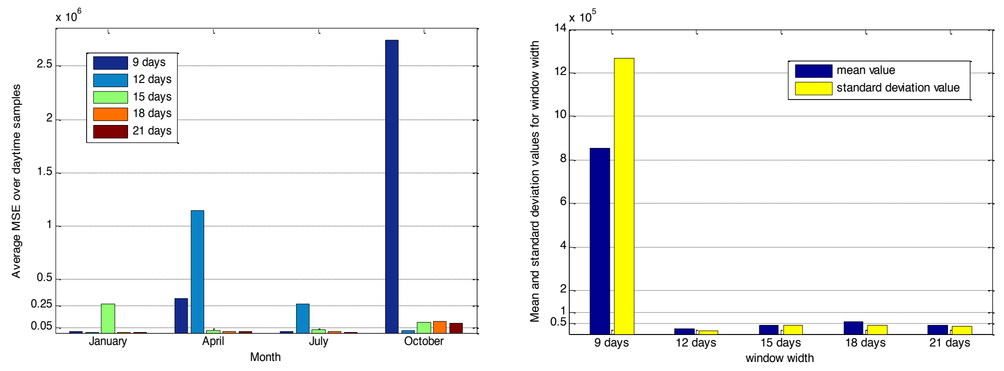
We used the Mean Square Error (MSE) defined in Equation (4.1), where t_i and o_i are, respectively, the target and predicted instantaneous energy values of sample *i*, and *S* is the number of samples of the considered day:

$$MSE = \frac{1}{S}\sum_{i=1}^{S}(t_i - o_i)^2. \tag{4.1}$$

Figure 4.3(a) shows the results related to window widths equal to 9, 12, 15, 18 and 21 days, respectively, for the four months. For each window width the figure shows the MSE made during the daytime (hours of daylight, approximately from 6 a.m. to 8 p.m.) of each predicted day, averaged over all the days of the considered month.

To compare the performances of the different neural models related to the five considered windows, we have evaluated the mean value and the standard deviation of the previously computed average MSEs for each window width over the four months. Figure 4.3(b) shows the mean value and the standard deviation of the five neural models (corresponding to the five windows) over the four months (January, April, July and October). As we can easily observe in Fig. 4.3(b), the worst window is that with 9 days width, so we decided to discard it.

Regarding the number of delays, as stated before, we chose to use 3 delays. Actually, we have experimentally found that, by increasing the number of delays, we can slightly improve the performance of the network. Anyhow, this depends on the window width value chosen and on the month considered, therefore it makes no sense to increase the complexity of the network. In a sense, we may say that the window width *w* lets the network be "aware" of the specific season or more in general the temporal context, while the number of delays *d* lets the network detect the particular real-time climatic variation. Figure 4.4 illustrates the iterative forecasting scheme employed.



(a) (b)

Figure 4.3. – *Forecasting performances obtained using irradiance only over the daytime samples for whole months (January, April, July and October).* (a) *Average MSE.* (b) *Mean and standard deviation of the average MSEs for each window width.*

Moreover, we have experimentally found that there does not exist an optimum value of *w* effective for all the seasons. Nevertheless, we have realized that in most cases, (about) 10 days is the minimal window width able to provide the network with the correct temporal context (season). In fact, the typical way to provide data for solar energy climatology has monthly, annual and 10-days granularity [74].

The previous result has been confirmed by repeating the experiments on the remaining months of each season.

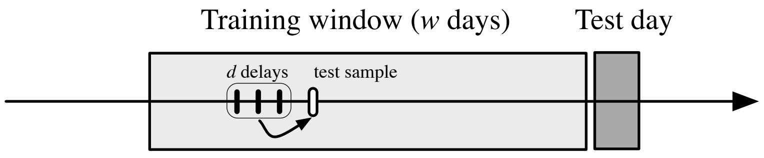


Figure 4.4. – *Forecasting scheme adopted.*

4.3.2 Model refinement

The goal of this phase is to propose a way to improve the model performance by trying to resolve possible irregularities present in the data collected from the specific PV installation.

During the experiments we noticed the presence of error spikes corresponding to days having a mean MSE sensibly higher than the other days of the same month. The position of these spikes (and the related days) is not fixed as the window width varies. More precisely, if a given day shows a high error for a specific window maybe the following day shows a similar high error for a different window.

So, to resolve this irregularity of the model, we decided to add one more input parameter to improve the performance, and, at the same time, to favor the regularization of the occurrence of the previous error spikes.

As we have a timestamp associated with each sample, we decided to add the hour input, maintaining fixed the structural parameters of the network. In fact, given a temporal context (e.g., a season or a month), the daily irradiation values at the same time of the day tend to be very similar, especially considering the values far from the maximum (approximately corresponding to midday).

Stated in other terms, by adding the hour input, we want to let the network be more aware of the concept of succession of samples during the day, with the hour representing the specific time at which the samples have been collected, so as to exploit some kind of regularity in the sun irradiation cycle.

So we performed the previous experiments with one more input, i.e., the hour input.

The resulting network model is depicted in Fig. 4.5. The model has 2 inputs, 10 hidden neurons with hyperbolic tangent sigmoid transfer function and 1 output with linear transfer function; the output is fed back as input; 3 delays are used on all the inputs. *W* and *b* represent, respectively, the weight matrix and the bias.

We found that the network that employs irradiation and hour inputs performs better compared to the one that uses only the irradiation input (see Fig. 4.6). We have measured a performance improvement of about one order of magnitude for all window widths.

Figure 4.6(a) shows the daytime MSEs of each predicted day, averaged over all the days of the considered month and related to window widths equal to 12, 15, 18 and 21 days, respectively, while Fig. 4.6(b) shows the mean value and the standard deviation of the average MSEs for the four months (January, April, July and October). From Fig. 4.6(b) we can notice that the best window width is w=15 as it corresponds to the lowest mean value and standard deviation, although the three window widths 12, 15 and 18 are comparable with each other. On the other hand, these three windows are sensibly better than *w*=21. For this reason we decided to eliminate *w*=21 from further analysis.

We performed a following assessment phase to attempt to select the best window width w, among the previous three found (12, 15, 18), for the particular temporal context (season). We carried out our experiments on the remaining months of each season.

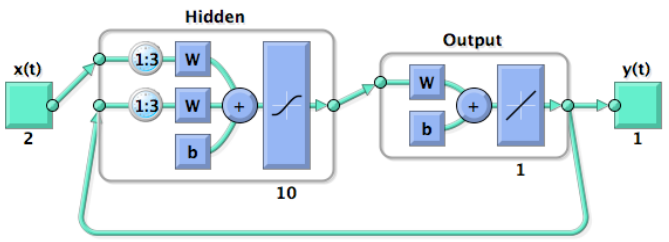
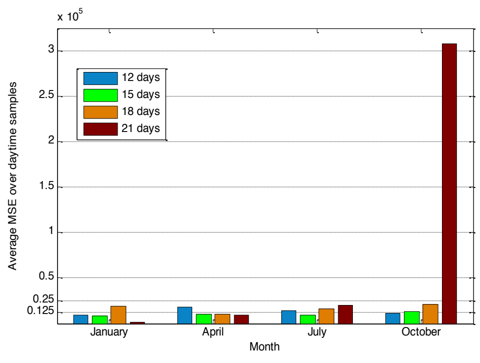
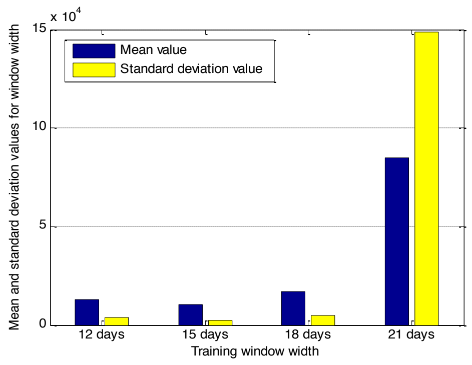


Figure 4.5. – *NARX neural network model employed in the experiments (the figure was produced in the Matlab® environment).*





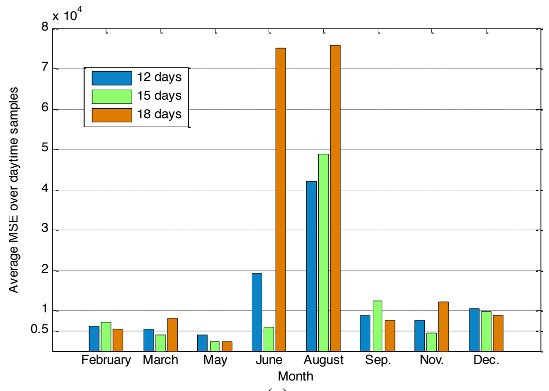
(a) (b)

Figure 4.6. – *Forecasting performances obtained using irradiance and hour inputs over the daytime samples for the months of January, April, July and October.* (a) *Average MSE.* (b) *Mean and standard deviation of the average MSEs for each window width.*

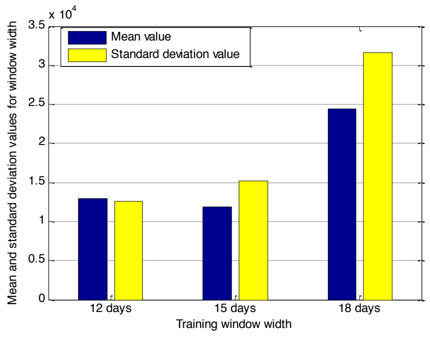
Figure 4.7(a) shows the goodness of the adopted methodology on the remaining months. The achieved performance is in line with the previous results. As we can see, the optimal window width results to be 15 days. Figure 4.7(b) shows the mean value and the standard deviation of the average MSEs for the three windows over the remaining eight months (February, March, May, June, August, September, November, and December).

From Fig. 4.7(b) we can notice that the best window width is *w*=15 as it corresponds to the lowest mean value although the standard deviation is slightly higher than that of 12 days. On the other hand, the differences among the three window widths are in fact negligible. For practical reasons, in the experiments described in the next section we decided to adopt *w*=15.

Table 4.1 summarizes the parameter values of the chosen final configuration.



(a)



(b)

Figure 4.7. – *Forecasting performances obtained using irradiance and hour inputs over the daytime samples for the months of February, March, May, June, August, September, November, and December.* (a) *Average MSEs.* (b) *Mean and standard deviation of the average MSEs for each window width.*

Table 4.1. – *Parameters for the final NARX neural network model.*

Parameter	**Value**
Structural parameters	
Number of hidden layers	1
Number of hidden neurons (*h*)	10
Number of delay elements (*d*)	3
Transfer functions	Hyperbolic tangent sigmoid (hidden layer), linear (output layer)
Training algorithm	Levenberg-Marquardt
Maximum number of learning epochs	30
Early stopping criterion	6 validation failures
Exogenous input variables	*irradiation*, *hour*
Endogenous input variable	*produced energy*
Configuration parameters	
Training window width (*w*)	15 days
Sampling frequency (*s*)	15 minutes

4.4 *Experimental results*

The analysis described in the previous section shows that the neural network with 10 neurons in the hidden layer and 3 delay elements is the best structure for the proposed problem. Moreover, the best performance was obtained employing the hour input along with the irradiation input.

Finally, we have also verified that the best results were achieved using training window widths of 12, 15 or 18 days. As already stated, we adopted *w*=15.

In the following two sub-sections we analyze, respectively, the prediction of the instantaneous energy on the continuous daily horizon and the prediction of the total (accumulated) energy produced over the whole day.

4.4.1 Prediction of the instantaneous energy

Since the goal of the following experiments is the comparison between the target produced energy and the predicted energy on the continuous daily horizon consisting of samples taken every 15 minutes, we considered the unsigned absolute instantaneous error for all samples of a day.

For the aim of these experiments we need to take into account the presence of "missing" days in the used dataset. A "missing" day is a day in which the system was down or under maintenance. Of course, meteorological changes possibly occurred during missing days cannot be correctly modeled by the forecasting system with the consequence that the prediction of one or more days following a missing one may produce high errors. In the following, we will distinguish two different categories of badly performing days, called, respectively, *atypical* days and *unpredictable* days. The difference between the two types of bad days is, respectively, the absence or presence of at least one missing day in the training window.

Figures 4.8-4.10 show the comparison, sample by sample, between the real energy (target) and the predicted energy of some randomly chosen days of the year. More in detail, Fig. 4.8 regards four days chosen at random among *well performing days*, Fig. 4.9 regards two days chosen at random among *atypical days*, while Fig. 4.10 concerns two randomly extracted *unpredictable days*. Each of these figures shows also the unsigned absolute instantaneous error, made on each sample of the day. We can notice that the performance error in Figs. 4.8(c), 4.8(d), 4.8(g), 4.8(h) is quite small with respect to the nominal energy production values shown in Figs. 4.8(a), 4.8(b), 4.8(e), 4.8(f).

In Fig. 4.9 we can see two examples of badly performing days (atypical days), having the predicted energy curve with a quite irregular trend. With the aim of interpreting these results, we checked the data at our disposal and we found out that the bad performance of the atypical days under consideration might be due to a rapid change in the panel temperature values. In fact, if rapid changes in solar radiation or temperature occur during the predicted day, the produced power can sensibly change and the prediction error might increase [18, 135].

Figure 4.10 shows two examples of badly performing days (unpredictable days), which present a noticeably evident difference between real and predicted energies.

By analyzing the results achieved in this sub-section, we can conclude that, regarding the atypical days, the forecasting performance could be sensibly improved only if we had a trustworthy prediction of changes in the panel temperature values at our disposal. On the other hand, as far as unpredictable days are concerned, improved forecasting performance could be easily achieved by using a complete dataset without missing days.

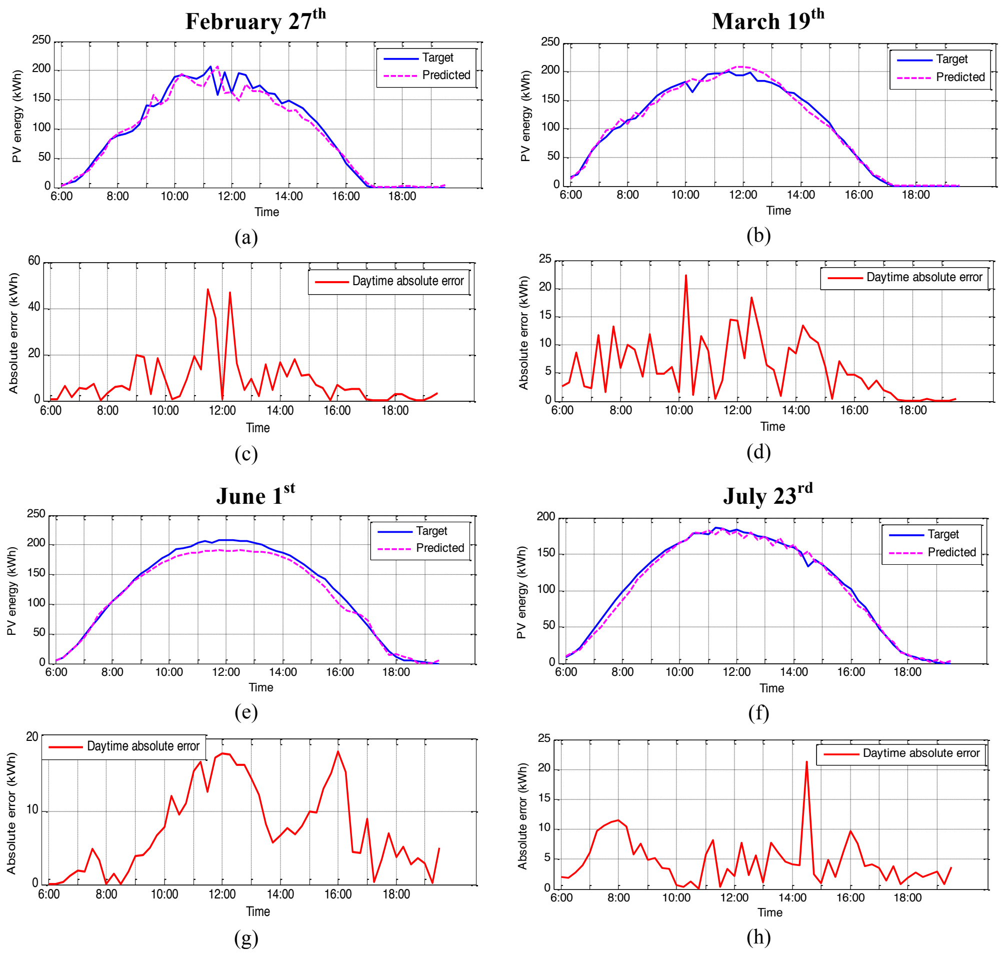


Figure. 4.8. – *Results on four well performing days chosen at random.* (a) (b) (e) (f) *Comparison between the real energy and the predicted energy*. (c) (d) (g) (h) *Associated daytime absolute instantaneous error.*

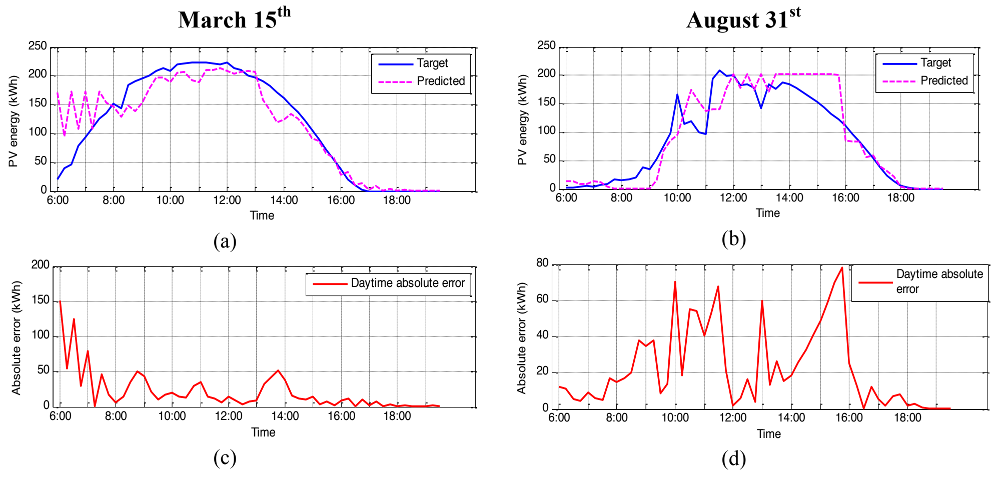


Figure 4.9. – *Results on two atypical days chosen at random.* (a) (b) *Comparison between the real energy and the predicted energy.* (c) (d) *Associated daytime absolute instantaneous error.*

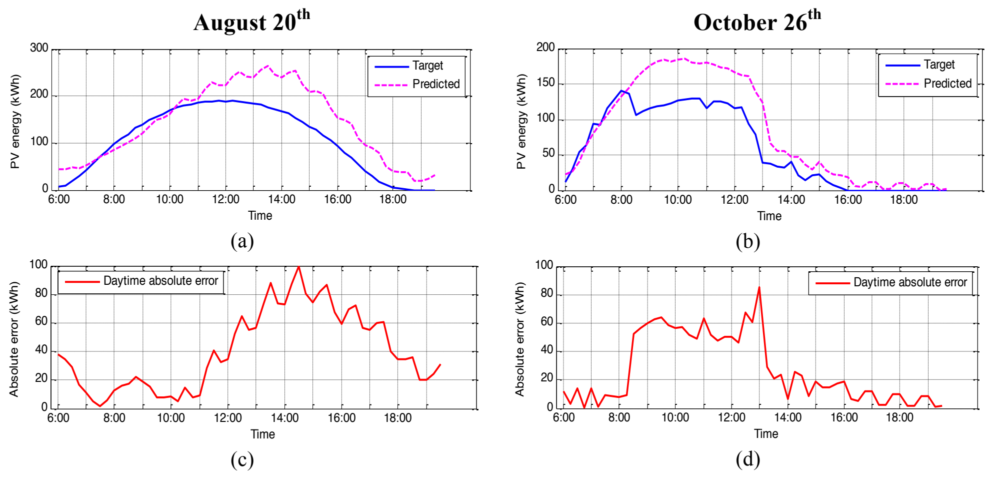


Figure 4.10. – *Results on two unpredictable days chosen at random.* (a) (b) *Comparison between the real energy and the predicted energy.* (c) (d) *Associated daytime absolute instantaneous error.*

4.4.2 Prediction of the accumulated energy

Usually in the renewable energy field, and in particular in PV plants, the total (accumulated) produced energy is considered instead of the instantaneous produced energy. So we take into account also the daily accumulated produced energy. Among other things, this typically allows to achieve an error reduction at the end of the day due to a compensation effect.

Table 4.2 compares the daily accumulated energy values predicted by the network with the real ones for the eight test days referred to above. In addition, we have considered the Absolute Percentage Error (APE) for each test day according to the equation: $APE = 100 \cdot (t - o) / t$, where *t* and *o* are, respectively, the real and predicted daily accumulated energy.

Table 4.2. – *Comparison between real and predicted accumulated energy.*

Predicted day	**Accumulated energy (kWh)**		**APE (%)**
	Real	***Predicted***	
February 27th	5110	4915.59	3.80
March 19th	5999	6062.04	1.05
June 1st	6556	6277.71	4.24
July 23rd	5779	5652.25	2.19
March 15th	6455	6414.31	0.63
August 31st	4784	5006.53	4.65
August 20th	5821	7788.51	33.8
October 26th	3254	4617.11	41.9

As we can see from Table 4.2, using the accumulated energy we obtain an acceptable error for all the considered days, independently of the dynamic behavior of the curve representing the instantaneous predicted energy, with a maximum value of 4.65% for August 31st.

We can observe that an apparently quite bad day, e.g., the 15th of March (Figs. 4.9(a) and 4.9(c)), has actually achieved the best performance among all the eight days (absolute percentage error less than 1%) thanks to the compensation effect.

To evaluate the accuracy of the proposed model, we computed the accumulated energy for each day of the four seasons and we compared it with the target accumulated energy.

Figures 4.11-4.14 shows the comparison between the real and predicted daily accumulated energy, with reference to all days of the four seasons. Please note that the figures may refer to a different number of days due to the lack of data pertinent to days in which the system was down or under maintenance.

From Figs 4.11-4.14, we can see that the predicted energy generally fits quite well the target energy although error spikes may appear. Once again, the bad performance of the badly performing days might be due to rapid changes in solar radiation or temperature during the predicted day [74, 135], or to the presence of missing days in the training window.

Furthermore, we computed the error made on each season as the average of the errors made on the daily accumulated energy values pertinent to the days of that season.

We used the Mean Absolute Percentage Error (MAPE) defined in Equation (4.2), where t_i and o_i are, respectively, the target and predicted accumulated energy values of day *i*, and *N* is the number of forecasted days of the season:

$$MAPE = \frac{100}{N} \cdot \sum_{i=1}^{N} \frac{|t_i - o_i|}{t_i}. \tag{4.2}$$

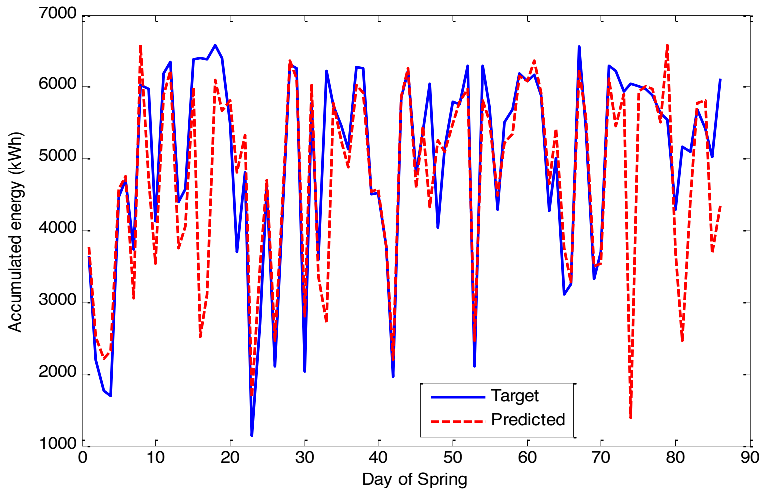


Figure 4.11. – *Target and predicted daily accumulated produced energy for Spring.*

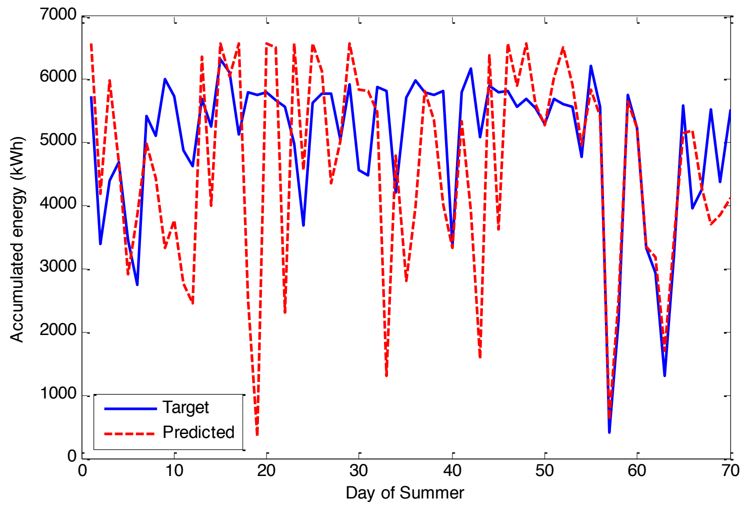


Figure 4.12. – *Target and predicted daily accumulated produced energy for Summer.*

Table 4.3 compares our method with the classical persistence method, which provides as forecasting value the last known value of the time series. The table shows the seasonal MAPEs made by the persistence method and by the neural model in the prediction of the daily accumulated energy values over all days of each season. It can be seen that the persistence method achieves results significantly worse than our neural network-based NARX model for all the four seasons.

The results achieved by the neural model compare favorably (even though obtained on different datasets and with a different technique) with those obtained in [162], where the average prediction error per day from April to September is about 26% of the measured power.

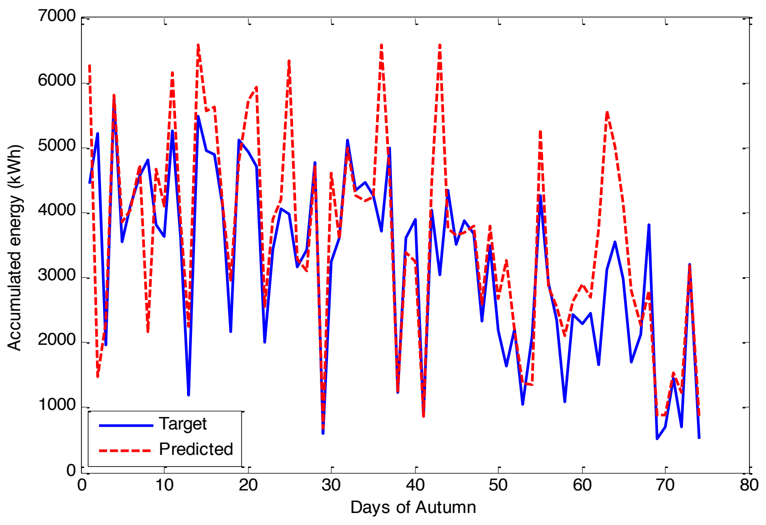


Figure 4.13. – *Target and predicted daily accumulated produced energy for Autumn.*

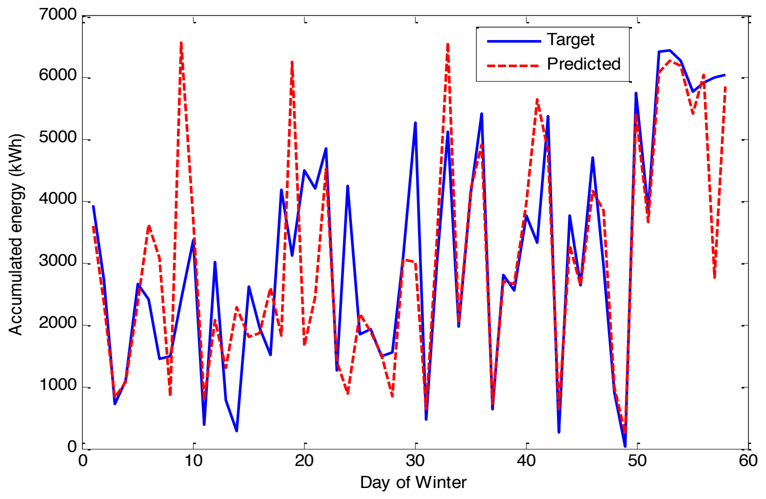


Figure 4.14. – *Target and predicted daily accumulated produced energy for Winter.*

Finally, with the aim of investigating the seasonal MAPE in Table 4.3, in Fig. 4.15 we show the error histograms related to the APE made on the daily accumulated energy for the four seasons.

Each seasonal histogram shows the frequency with which an error value is made in the considered season, i.e., how many days of that season collected that error value.

Figure 4.15(a) shows that in 52 Spring days, which represent more than 60% of all the considered Spring days (see Fig. 4.11), the error made is lower than 10%, while only a few days have errors significantly higher. Similarly, from Figs. 4.15(b), 4.15(c), 4.15(d), we can see that, respectively, 27, 36 and 29 days (i.e., 39%, 51% and 50% of the considered days for that season, respectively (see Figs. 4.12, 4.13, 4.14)) produce an error lower than 10%. Regarding higher errors the same considerations as those made for Spring hold.

Table 4.3. – *Comparison between the persistence method and the NARX neural network.*

Predicted season	**MAPE (%)** **Persistence method**	**MAPE (%)** **NARX neural network**
Spring	31.16	12.2
Summer	38.13	21.1
Autumn	84.59	26
Winter	77.84	23.9

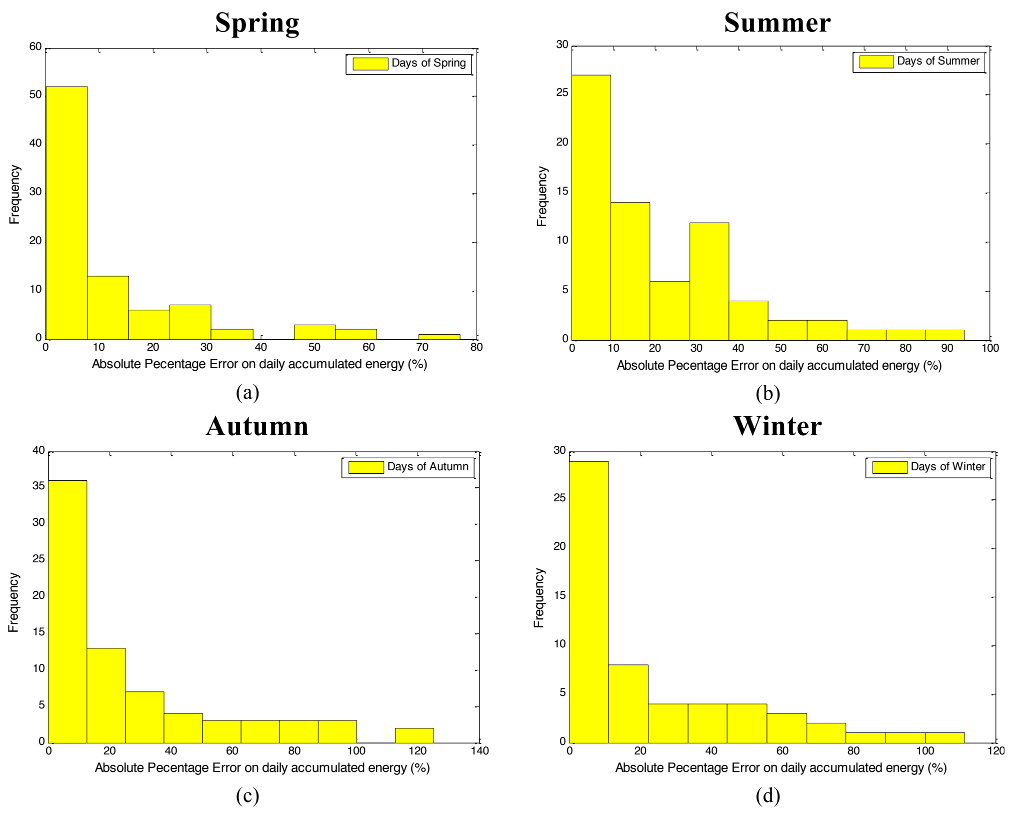


Figure 4.15. – *Error histograms for* (a) *Spring,* (b) *Summer,* (c) *Autumn, and* (d) *Winter.*

In order to correctly interpret the results shown above we have tried to identify which are the days that produce the worst errors, e.g., errors higher than 30% for

Spring, or higher than 40% for Autumn. We have found out that almost all these days follow immediately (after one or two days) a “missing” day, that is a day in which the system was down or under maintenance. As already stated, missing days are not included in the used dataset, so that their information cannot be used by the forecasting system. As a consequence, the prediction of one or more days following a missing one may produce high errors, especially when there have been meteorological changes that result not to be correctly modeled.

In Fig. 4.16 we show the error histogram related to the APE made on the daily accumulated energy for the best days of Spring (those corresponding to an error lower than 10%). Similar histograms (not shown) can be obtained for Summer, Autumn and Winter.

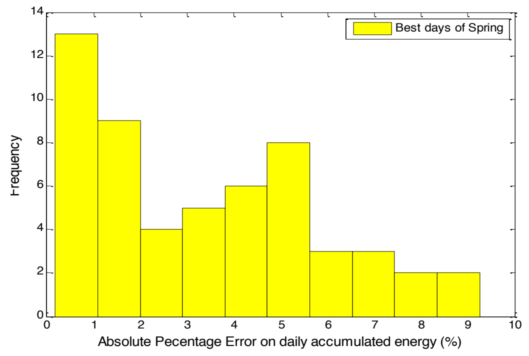


Figure 4.16. – *Error histogram for the best days of Spring (daily error lower than 10%).*

Finally, we tested the developed forecasting system on the second PV installation obtaining practically the same results as those shown above (the differences are so negligible to make the presentation of such results unnecessary).

Based on the previous considerations, we can observe that the compensation effect resulting from the use of the daily accumulated energy makes the developed forecasting system suitable to be effectively and profitably used for one day-ahead forecasting of energy production.

4.4.3 Discussion

In the previous sections we have shown that a powerful nonlinear forecasting technology (neural networks), employed within a NARX model, is able to predict the energy production in a PV installation, provided that (see Table 4.1):

- the right exogenous inputs are used (irradiance and hour of the day);
- the right NARX model is used (i.e., the correct number of tapped delays *d*);
- the right neural network structure is used (particularly, the number of hidden neurons *h*);
- the right training window size (*w*) is adopted;
- the right sampling frequency (*s*) is used.

In our experience the other neural network parameters (namely, training algorithm, maximum number of learning epochs, early stopping criterion, etc.) have a lower impact on the forecasting performance.

The proposed methodology does not depend on the characteristics of the specific PV installation, such as geographical location, panel's inclination, etc. Its degrees of freedom make it suitable to be applied to any other PV installation.

4.5 Concluding remarks

In this chapter we have presented a general methodology to solve a problem of one day-ahead forecasting of solar energy production. We implemented the time series analysis model NARX by means of a feed-forward neural network with tapped delay lines. In the training process we use the solar irradiation data and the hour as exogenous inputs, and the PV energy production data as endogenous input.

Experimental results have showed that the proposed neural network can faithfully reproduce the curve of daily produced energy so as to predict the daily accumulated energy with seasonal mean absolute percentage errors ranging from a minimum of 12.2% (Spring) to a maximum of 26% (Autumn). These results were achieved despite the presence of missing days (about 21% of the days of one year) in the used training windows. The proposed methodology has been validated by showing that it significantly outperforms the persistence method, a frequently used benchmark in this kind of applications.

Future work will focus on integrating the proposed system with a weather forecasting system, so as to estimate directly the environmental variables. This could be useful when the input data are not available and anyway could increase the prediction performance.

Moreover, due to the good results obtained, we may extend the forecasting horizon to a multiple of the day, so as to strengthen the long-term forecasting analysis. Similarly, we may change the actual forecast time-step of 15 minutes to hourly time-steps.

The forecasting system resulting from the proposed methodology could be profitably used to control the energy distributing grid. Indeed, since PV-based power generators are discontinuous, being influenced by weather conditions, this discontinuity has to be mitigated using alternative power sources, like gas turbines and thermal power plants, which have short start-up time (of the order of a few hours). This means that knowing the energy produced by a grid-connected PV installation 24 hours ahead of time is enough to prepare the start-up of such alternative power sources. The importance of the present study stems exactly from this consideration. In addition, it is widely recognized that an accurate forecasting tool for energy production is a key component of a smart grid, especially when coupled with an energy consumption predictor.

5

A hierarchical approach to multi-class fuzzy classifiers for PV energy production

5.1 Introduction

In the last two decades fuzzy rule-based systems have been extensively applied to pattern classification thanks to their capability to achieve good trade-offs between accuracy and interpretability [45, 76, 90, 129, 134]. In particular, interpretability of a fuzzy rule-based system is typically measured in terms of complexity of the rule base, and depends on such factors as comprehensibility of fuzzy partitions of the domains of the involved linguistic variables, number of input variables, number of conditions in the antecedent of each rule, and number of fuzzy rules. In its simplest form, a fuzzy rule-based classifier is a system consisting of fuzzy if-then rules having a class label as consequent.

When designing a fuzzy classifier two main issues must be considered: fuzzy classifier identification and fuzzy parameter optimization. Major issues in fuzzy classifier identification are i) how to choose the membership functions of linguistic variables, ii) how to generate the fuzzy rules, and iii) how to determine the output class.

A large number of methods for extracting fuzzy rules directly from numerical data have been proposed, thus making prior knowledge about the data unnecessary. These methods include heuristic procedures [2, 63, 70, 114, 156], neuro-fuzzy techniques [79, 102, 103, 111, 146, 150], clustering methods [3, 130], genetic algorithms [8, 19, 51, 52, 65, 68, 71, 133, 137, 165], fuzzy decision trees [21, 155, 163], and data mining techniques [38, 39, 62, 71].

The antecedent part of fuzzy rules may contain single-dimensional fuzzy sets obtained by partitioning each input dimension. Antecedent fuzzy sets may, e.g., have pre-specified linguistic values with fixed membership functions obtained by homogeneously partitioning each axis of the pattern space [67] or may be purposely defined by domain experts. Alternatively, multi-dimensional antecedent fuzzy sets may be generated by applying a clustering algorithm to sample input-output data [2, 3]. Sometimes, these multi-dimensional antecedent fuzzy sets are projected onto each axis of the input space to improve the interpretability of the clusters produced [129, 136]. In all cases, the output class associated with each fuzzy subset (either grid cell, identified by the partitions on the input dimensions, or cluster) of the pattern space is derived from the training samples belonging to that subset.

Of course, the performance of a fuzzy rule-based classifier (FRBC) depends on the grain size of the fuzzy partition of the pattern space: a too coarse fuzzy partition may cause many misclassifications while a too fine fuzzy partition may miss to generate fuzzy if-then rules due to lack of training samples in the corresponding areas of the input space. A possible solution is to simultaneously use different partitions with different resolutions at the expense of a high number of fuzzy rules, especially in high-dimensional spaces [70]. A different solution is selective partitioning, in which the input regions where classes overlap are further partitioned with a higher resolution level [103].

Other alternatives are possible. E.g., Ait Kbir *et al.* [6] propose the construction of a compact fuzzy classification system by using a method of hierarchical fuzzy partition based on 2N-tree recursive decomposition of the feature space. In [54], a hierarchical fuzzy partition is generated independently over each dimension in an ascending way by aggregating fuzzy sets. In [85], the authors adopt a fuzzy entropy measure to partition the pattern space into non-overlapping decision regions and to select relevant features for classification purposes. In [47], a hierarchical fuzzy rule-based classification system is proposed for imbalanced datasets. Basically a finer granularity of the fuzzy partitions is applied in the boundary areas between the classes.

As far as fuzzy parameter optimization is concerned, several optimization techniques have been applied to set the fuzzy system parameters based on the training samples. These include the type and shape of fuzzy membership functions, and the number and structure of fuzzy rules. E.g., genetic algorithms [67], and evolutionary multi-objective approaches [68, 69, 71] are adopted to cope with the combinatorial explosion of the number of rules. In [165], the authors adopt a simultaneous genetic algorithm-based optimization of fuzzy partitions, shape and parameters of fuzzy membership functions, number and size of fuzzy rules. In [134], interval-valued fuzzy sets with a post-processing genetic tuning step of their parameters are used to model the linguistic labels. Li *et al.* [90] propose a hybrid co-evolutionary genetic algorithm for learning approximate fuzzy rules, by using a q-nearest neighbor replacement method to coevolve a population of rules, and a local search method. A classifier is built by extracting rules with minimal redundancy from the final population. Setnes *et al.* [137] apply fuzzy clustering to produce an initial TSK fuzzy rule set, then they use a real-coded GA to simultaneously optimize the rule antecedents and the consequents. Wu *et al.* [159] adopt a functional-link-based neural fuzzy network where the consequence of each rule is a nonlinear combination of the input variables. Tung *et al.* [148] propose the self-organizing Yager-based hybrid neural fuzzy inference system: initial clusters are found in the input-output space by means of the Gaussian Discrete Incremental Clustering technique, and fuzzy rules are generated through the Wang and Mendel method. Wang *et al.* [153] apply the Mapping-Constrained Agglomerative clustering method to identify the cluster configuration of a given dataset for the construction of an initial classifier structure. The linear and nonlinear parameters of the classifier are then optimized, respectively, by a recursive least squares algorithm and a modified Levenberg-Marquardt algorithm. Ishibuchi *et al.* [73] propose the combination of two fuzzy genetic learning approaches (i.e., Michigan and Pittsburgh) into a single hybrid algorithm for designing fuzzy rule-based classifiers. Abonyi *et al.* [4] use a decision tree-based initialization of the fuzzy rule-based classifier for feature selection and initial partitioning of the input domains. The initial fuzzy classifier is optimized by similarity-driven rule reduction and a multi-objective genetic algorithm based on redundancy and accuracy. Heuristic methods for rule weight specification are proposed in [72]. In [115], an adaptive method based on reward and punishment is applied to automatically adjust the weights of fuzzy rules. In [167], a hill-climbing search algorithm is adopted for learning rule weights.

In this chapter we propose an easy-to-use approach for extracting fuzzy rules from available data by employing the Wang and Mendel algorithm for the generation of the rule base. The fuzzy system developed in [33] is obtained exploiting a hierarchical scheme, as a combination of fuzzy models built on input domain regions increasingly smaller, according to a multi-level grid-like partition. Only the necessary partitions are built, in order to avoid the explosion of the number of rules with the increase of the hierarchical level. The fuzzy system employs the fuzzy rule-based classifier `frbc` [28], presented in Section 2.3. The optimal values of some key parameters of the proposed method are found by means of a real-coded genetic algorithm.

5.1.1 Context of application

To illustrate the proposed approach we refer to a real-world dataset consisting of input/output pairs collected from a photovoltaic (PV) installation: the inputs are the temperature of the solar panel and the irradiation, the output is the produced energy.

The reason for this is the following. PV installations (see Section 4.1) are typically used as energy sources for the electric grid. Major issues in electric grid management (in particular, smart grids) are efficiency and reliability, which require, among other things, fast and easy understanding by the grid operator of both the electricity demand and the electricity supply (energy production). With the aim of helping the grid operator to promptly make his/her decisions, we propose to model in linguistic terms the decision process and the elements on which that process operates, as previously done in [34, 35]. More precisely, we deal with this issue as a fuzzy classification problem. We divide the values of energy production into three classes (*low*, *medium*, *high*), each modeled by a fuzzy set. We also model the environmental variables (namely, temperature and irradiation) as linguistic variables. Then we build fuzzy rules directly from data so as to associate pairs of values of the two environmental variables with a specific value of produced energy. In this way, the manager of a PV plant can gain enough information from the system so as to perform appropriate functional operations for the installation, even if the exact energy production value is not known [55].

The proposed approach is also applied to some well-known benchmark datasets and the results are compared with those obtained by other authors using different techniques.

5.1.2 Outline of the chapter

The chapter has the following structure. Section 5.2 describes the real-world experimental data used to illustrate the proposed method; Section 5.3 introduces the proposed hierarchical approach to fuzzy classifier construction; Sections 5.4 and 5.5 present the application of the methodology to the real-world dataset and the obtained results. Finally, Section 5.6 is devoted to validate our classifier, by comparing it with other classifiers present in the literature on some benchmark classification problems, namely the Fisher's Iris data, the Wine data, the Wisconsin breast cancer data and the Pima Indians diabetes data. Lastly, concluding remarks are provided in Section 5.7.

5.2 Description of the real-world experimental dataset

The real-world data used to highlight the characteristics of the proposed method for building fuzzy classifiers were collected during five months (from March to July 2009) from a PV installation, consisting of an array of solar panels, located in Italy. The nominal power of this PV installation measured by the associated inverter is 6.45 KW. Data were collected every day during daylight, with a sampling frequency of 15 minutes.

As stated previously, among the environmental data, temperature of the solar panel and irradiation play the most significant role to evaluate the energy production of a PV installation [9, 80, 81], so they are chose as input parameters (see Section 4.2. for a more detailed description).

The output parameter is the energy production related to the PV array and measured by the associated inverter.

Data have been adapted in order to use them in a fuzzy classification problem, as explained in the following. Before beginning the data analysis, we needed to transform the energy numerical values into class labels. For the sake of simplicity, we operated a uniform partition on the output domain by identifying three intervals corresponding to three output classes (*Low*, *Medium*, *High* energy production, M=3). Then, we associated each output pattern with the energy label corresponding to its interval. In Fig. 5.1 we can see the scattering of the dataset (7303 samples) over the two dimensions and the distribution of the samples over the three output classes.

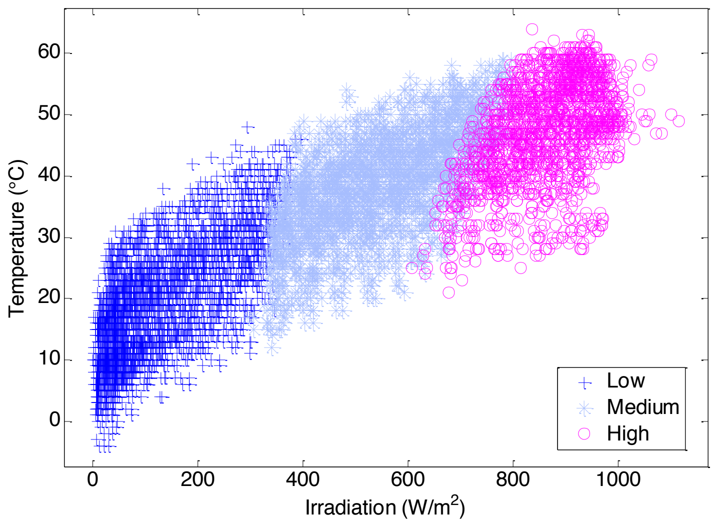


Figure 5.1. – *Scatter diagram of the PV dataset and distribution of the samples over the three output classes (Low, Medium, High).*

5.3 A hierarchical approach to fuzzy classifier construction

In this section we introduce the proposed methodology [33], which consists of a first step, a second iterative step and a final third step. Let us make some general considerations before describing each step in greater detail. Both in the first step and at each iteration of the second step we build a grid, respectively, on the whole input space and on a portion of the input space. Whatever the case, our aim is to find *univocal mapping* areas, i.e., input areas mostly containing patterns associated with the same class label. For each such area, an appropriate number of training samples are randomly extracted and used to generate fuzzy rules that model that area. Since we are interested in collecting training samples according to the real distribution of the available input patterns in relation with each output class, whenever we need to construct a grid in the input portion under consideration we should adopt an *ad hoc* non-uniform partition, e.g., based on the distribution of the input samples in the feature space. On the other hand, the `frbc` method expects a uniform partition of the input space. Thus, for a good compromise between efficiency and computational cost, we chose to perform a non-uniform grid partitioning of the original input space only in the first step, while we decided to adopt uniform grid partitioning of the relevant input area in all iterations of the subsequent second step. Of course, appropriate scaling will let the non-uniform grid partition correspond to an equivalent uniform partition used by the `frbc` system.

In practice, our objective is to split the input domain into *univocal mapping* areas with possibly different grain size, and to build a separate set of fuzzy rules to model each such area. Let us now describe more thoroughly the three steps of the methodology.

5.3.1 First step: first-level grid partitioning

In the first step, applied to the original input space, we carry out the following actions:

i) we apply the *k*-means clustering algorithm [91] separately to each input dimension;

ii) we use the separation thresholds between the clusters for:
   - ii.1) building a non-uniform grid (made of $k \times k$ areas) in the input space, and
   - ii.2) constructing a non-uniform fuzzy partition on each input dimension consisting of k membership functions;

iii) we analyze separately each area of the grid previously built in order to discriminate among *insignificant*, *univocal mapping* and *to-subgrid* areas. More precisely,
   - iii.1) an *insignificant* area is any grid area A containing a total number N_A of input samples below a predefined *first-step relevance threshold* RT_1 (the value of RT_1 depends on the specific problem under consideration); each such area is eliminated from further consideration;
   - iii.2) a *univocal mapping* area A is any non-insignificant area in which there exists a *dominant majority class*, i.e, the class associated with the majority of the samples falling in that area, such that the number N_A^+ of majority class samples is greater than, or equal to, a given percentage, say *first-step dominance percentage* (DP_1), of the numerousness N_A of samples falling in A;
   - iii.3) each non-univocal and non-insignificant grid area is called *to-subgrid* area: each such area will undergo the second iterative step;

iv) for each *univocal mapping* area A, a random extraction of $K = \min(perc \cdot N_a^+, S)$ majority class samples is performed, with *perc*, appropriately chosen, representing a percentage of N_A^+, and S, appropriately chosen, being a problem-dependent upper bound of samples of the same class that can be extracted from the same area. The extracted samples will be used to generate, through `frbc`, the pertinent fuzzy classification rules that model the considered area;

v) we build the *first-level fuzzy model* by training `frbc` with all the samples extracted from all the univocal mapping areas previously found.

To complete the description of the first step of the methodology we must mention that, since we use the Wang and Mendel method implemented in `frbc`, which builds a uniform fuzzy partition of each input feature space, two more operations must be performed within action ii), namely:

ii.3) for each input feature we build a uniform fuzzy partition, consisting of *k* fuzzy sets, using the Wang and Mendel method implemented in `frbc`;

ii.4) for each input feature, we use non-uniform scaling to transform the previous uniform partition into the corresponding non-uniform partition (built at stage ii.2): all the feature values are scaled from their original interval to the new interval, maintaining the proportionality.

5.3.2 Second (iterative) step: deeper-level grid partitioning

The second step is applied to each *to-subgrid* area, which has been found either in the first step or at any iteration of the second step itself. For a given *to-subrid* area A we perform the following actions:

i) we build a uniform hard partition (consisting of *k* intervals) on each dimension of *A*, so as to construct a deeper-level uniform grid of the area itself;

ii) we identify the *insignificant*, *univocal mapping* and *to-subgrid* areas inside the new grid. Similarly to what done before, first we eliminate from further consideration any insignificant area of the new grid, by using the *second-step relevance threshold* RT_2^i, with i, $i \geq 1$, representing the iteration number of the second step; then we identify the univocal mapping areas based on the *second-step dominance percentage* DP_2^i with *i*, $i \geq 1$, having the same meaning as before; finally, each *to-subgrid* area of the new grid will undergo the second iterative step, thus giving origin to one more iteration. Of course, *second-step relevance thresholds* RT_2^i, $i \geq 1$, will typically decrease with the increase of the iteration number *i*, while *second-step dominance percentages* DP_2^i, $i \geq 1$, may vary according to the iteration number *i*;

iii) we identify the minimum (hyper)rectangle (see Fig. 5.2) containing all the samples falling inside the univocal mapping areas included in *A*; we construct a uniform fuzzy partition, consisting of *k* membership functions, on each dimension of the (hyper)rectangle, thus producing a fuzzy partition of the (hyper)rectangle itself; then we generate a *deeper-level fuzzy model* for the (hyper)rectangle by training `frbc` with an appropriate number $K = \min(perc \cdot N_a^+, S)$ of majority class samples extracted from each univocal mapping area a related to the hyper(rectangle).

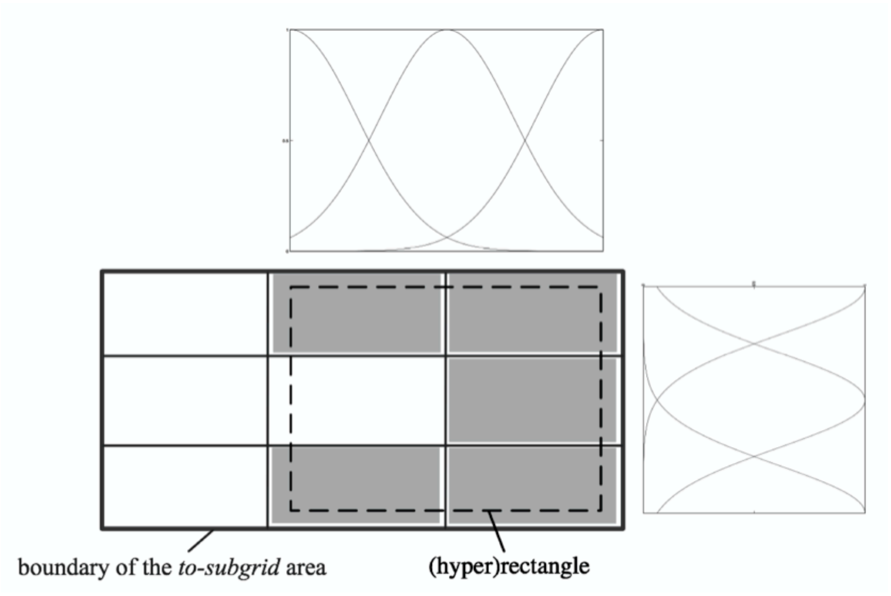


Figure 5.2. – *A to-subgrid area containing five univocal mapping areas (colored areas), and the (hyper)rectangle (dashed line) including all the samples inside the univocal mapping areas.*

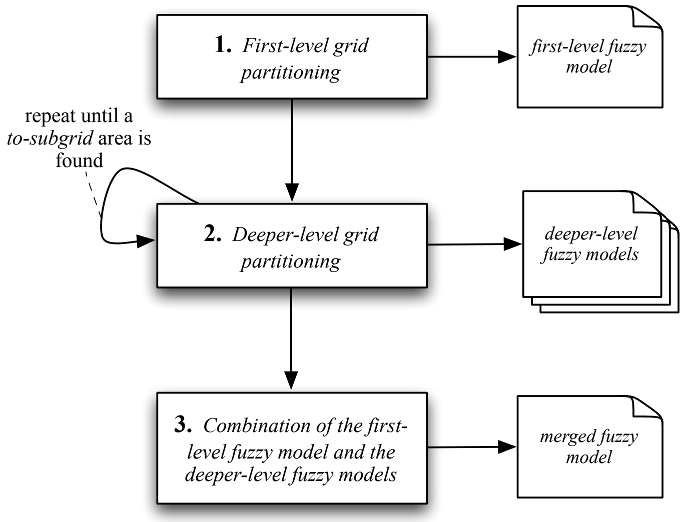
5.3.3 Third step: final fuzzy model generation

In the third step, we generate the final fuzzy model, called *merged fuzzy model*, as the union of the *first-level fuzzy model* and all the *deeper-level fuzzy models* built during the hierarchical process. The fuzzy sets for each input variable of the merged fuzzy model are the union of the fuzzy sets (for that input) of all the models built. The merged fuzzy rule base is the union of the rule bases of all the fuzzy models.

We observe that there may be input domain regions modeled by more than one fuzzy set (e.g., the larger one built at first level analysis and the narrower, and therefore more specific ones, built at higher level analysis).

Figure 5.3 depicts the steps described above and shows the objects resulting from each step.

In this way the final fuzzy model is obtained through a hierarchical process, by merging fuzzy models built on input domain regions increasingly smaller, as the result of the construction of appropriate grids on the pertinent areas of the input domain. The objective is to exploit the input domain space in an effective way, avoiding unnecessary analysis and thus the generation of too many, irrelevant rules. The proposed hierarchical method allows us to extract an *ad hoc* training dataset, so as to build the final `frbc` system as well as possible. This is a key feature when dealing, e.g., with non-uniformly distributed data. Actually, with reference to the used real-world dataset, months and days within a month may be typically different, due to the highly variable climatic conditions, so a random selection of training samples would not meet efficient training requirements.



1. First-level grid partitioning
first-level fuzzy model
repeat until a to-subgrid area is found
2. Deeper-level grid partitioning
deeper-level fuzzy models
3. Combination of the first-level fuzzy model and the deeper-level fuzzy models
merged fuzzy model

Figure 5.3. – *Steps of the proposed hierarchical methodology and resulting objects.*

5.3.4 GA-based parameter optimization

A genetic algorithm (GA) is an optimization process based on the mechanics of natural selection and genetics to produce better populations, according to a fitness function. Generally, GAs start with a randomly generated initial population of chromosomes, representing candidate solutions to the problem at hand, and evolve toward populations having a better fitness by applying genetic operators such as crossover and mutation.

We apply a GA to optimize the following parameters (i refers to the iteration number of the second step): the *relevance thresholds* RT_1 and RT_2^i, $i \geq 1$, the *dominance percentages* DP_1 and DP_2^i, $i \geq 1$, the *maximum number S of samples extracted* from a given grid area (valid for the first step and all iterations of the second step), the minimum *rule weight w* (valid for the first step and all iterations of the second step), and the rule *weight modifiers* Δw_1 and Δw_2^i, $i \geq 1$. In particular, the last two parameters aim, respectively, to control the complexity of the whole rule base, and to enhance/inhibit the influence of the rules of a given step/iteration. The maximum number of iterations is fixed heuristically.

We adopt real-coded chromosomes. The range of possible values of each gene is chosen in heuristic way based on the specific dataset under consideration. When appropriate, integer approximations of real numbers are adopted. The fitness function is the classification error of the fuzzy classifier.

5.4 Application of the proposed methodology to the real-world dataset

In this subsection we show the application of our methodology to the PV dataset. For the sake of simplicity, we adopt *k*=3 in the *k*-means algorithm. Further, we consider the percentage *perc* of majority class samples to extract from each univocal mapping area equal to 70%.

5.4.1 First step

In this step, we perform the actions described in Section 5.3.1. Figure 5.4 shows the non-uniform partitions obtained by the *k*-means algorithm (*k*=3) on the input features and the scattering of the original dataset over the three output classes on the 9 areas of the grid. In the figure, *i*1 and *i*2, and *t*1 and *t*2 represent, respectively, the separation thresholds for irradiation and temperature. The number on each area identifies the grid area. As we expected, two areas were found to be *insignificant* (areas 3 and 7). Insignificant areas correspond, for instance, to incompatible or unusual input conditions, such as high irradiance and low temperature at the same time, or vice versa.

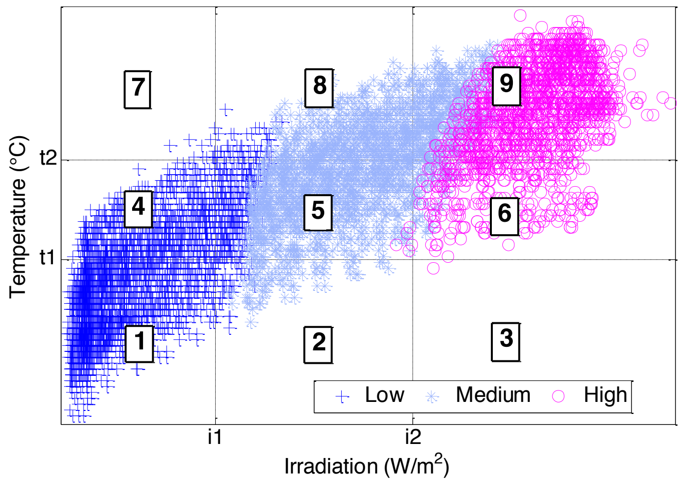


Figure 5.4. – *Partition of the input domain by applying the k-means clustering algorithm (with k=3) and identification of 9 areas (numbered from 1 to 9) on the grid.*

Figures 5.5(a) and 5.5(b) show, respectively, the uniform partition built on one input feature (irradiation) space by `frbc`, and the corresponding non-uniform partition based on the *k*-means algorithm. Both partitions consist of two-sided Gaussian membership functions. We chose two-sided Gaussian membership functions in the first level of analysis since they are known to be very accurate, provide complete coverage of the modeled space, and allow easy scaling from the uniform partition to the non-uniform one. Similar considerations hold for the temperature input.

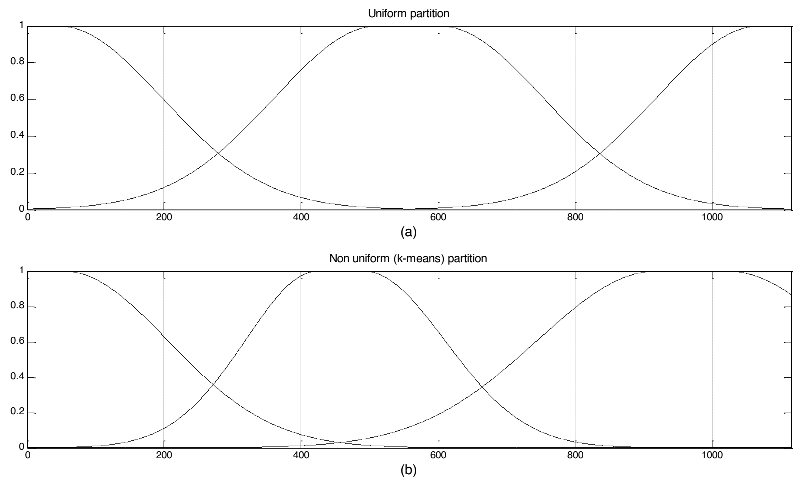


Figure 5.5. – *Scaling from a uniform partition* (a) *to a non-uniform one* (b). *In both cases, two-sided Gaussian membership functions are used.*

Next, we need to take into account separately each area of the grid previously built, in order to discriminate among *insignificant*, *univocal mapping* and *to-subgrid* areas. The bar diagrams in Figs. 5.6(a) and 5.6(b) shows the distribution of the samples in the areas of the first-level grid. Fig. 5.6(a) simply indicates the numerousness of the samples in each area, whereas Fig. 5.6(b) shows the class distribution in each area (for better clarity, we used a logarithmic scale on the y-axis). As we can see only a few regions present samples from one class, while the other regions present samples belonging to at least two classes.

With reference to Fig. 5.7, which shows the first-level and second-level grid partitions, two areas (i.e., 3 and 7) are found to be *insignificant* and so they are discarded. Areas 1, 4, 6 and 8 are *univocal mapping* areas, that is, we can find samples mostly from one class. In particular, *Low energy* class samples in areas 1 and 4, *Medium energy* class samples in area 8, and *High energy* class samples in area 6. Each such area is candidate to represent a possible input state for the system. Finally, *to-subgrid* areas 2, 5 and 9 are marked for further analysis.

5.4.2 Second step

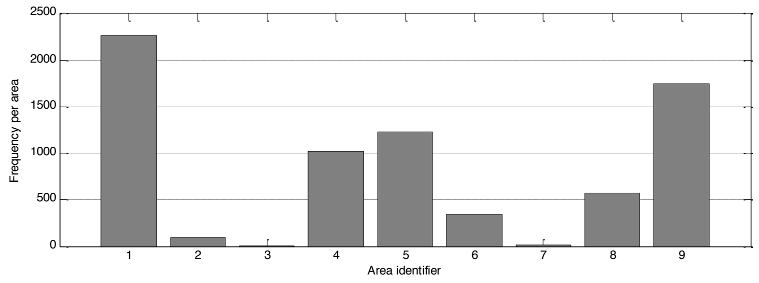
During this step, we analyze each *to-subgrid* area (in this case, areas 2, 5 and 9 in Fig. 5.7) in a similar way as done in the previous step with the following differences:

- the initial partition built on each dimension is a uniform hard partition instead of non-uniform;
- the *deeper-level fuzzy model* is built on the minimum (hyper)rectangle containing the *univocal mapping* areas included in the *to-subgrid* area

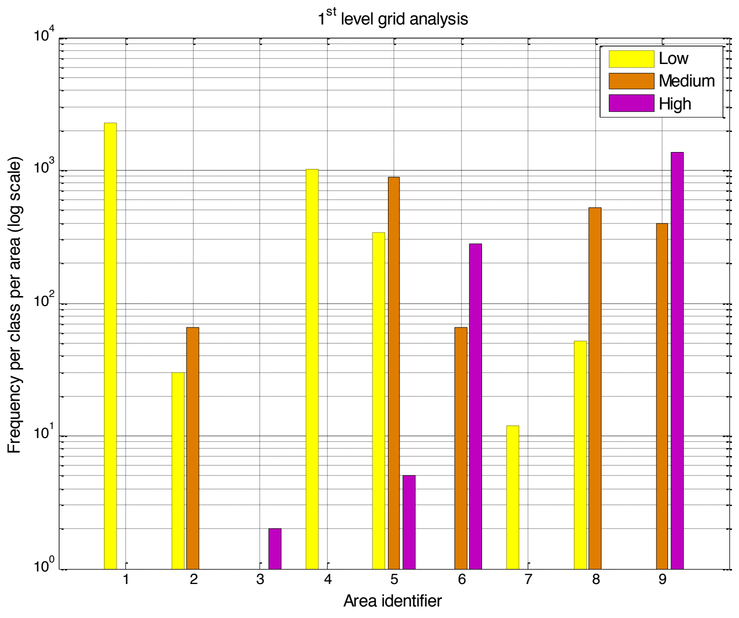
under consideration;

- the fuzzy partitions are made of Gaussian membership functions.

The process is repeated until no area needs to be further divided.



(a)



(b)

Figure 5.6. – *Distribution of samples on the areas of the first-level grid:* (a) *total, and* (b) *per class (logaritmic scale on y-axis).*

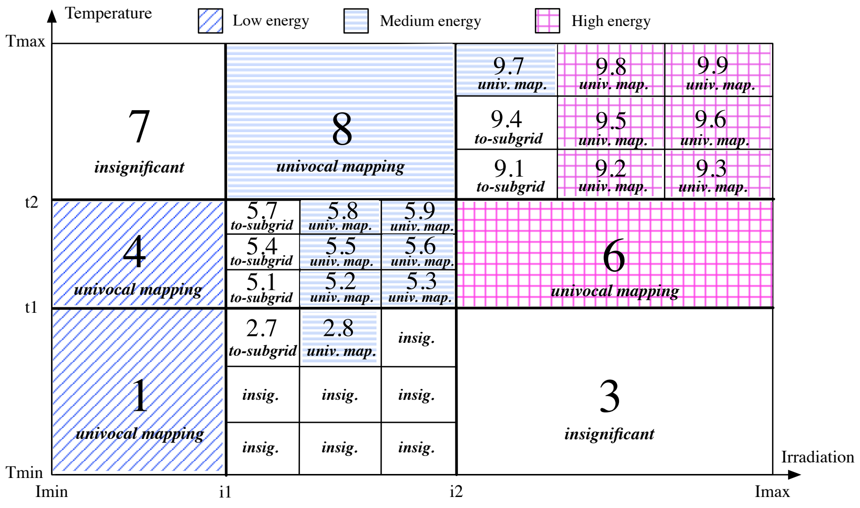


Figure 5.7. – *The grid partitions of the original input space obtained by applying the first step and the first iteration of the second step of the methodology (the dot notation is used to indicate sub-areas).*

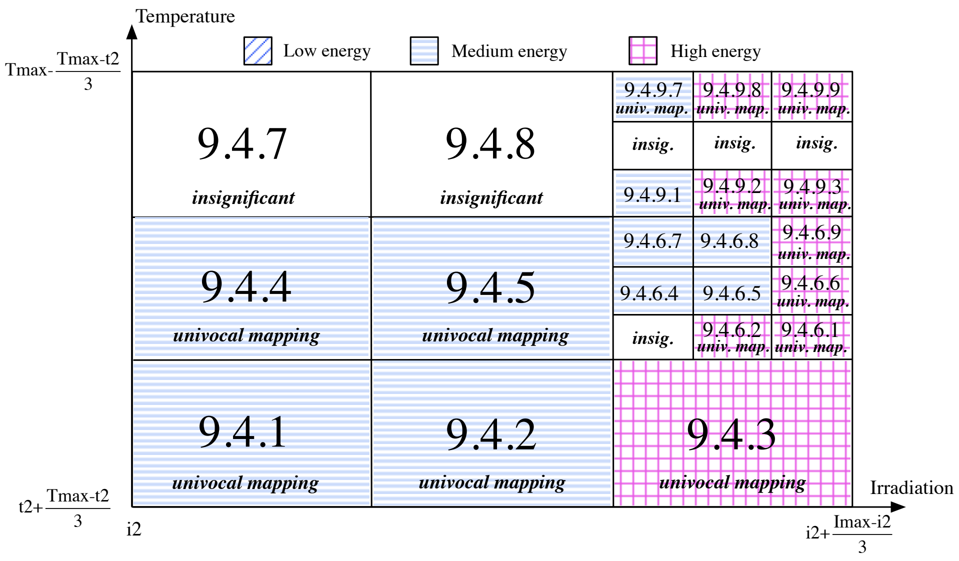


Figure 5.8. – *The grid partition of area 9.4 obtained through the second and third iterations of the second step of the methodology.*

Figure 5.8 shows the grid partitioning for the second and third iterations of the second step in sub-area 9.4 (which is marked "*to-subgrid*" in Fig. 5.7). Two areas, namely 9.4.6 and 9.4.9, are marked *to-subgrid*, so a further analysis level is required. In the third iteration of the second step only *insignificant* and *univocal mapping* areas are found, so the analysis of area 9.4 and its sub-areas ends at this iteration. We wish to highlight dot notation used, e.g., *A.a* is used to indicate a sub-area *a* inside a given area *A*.

Figure 5.9 shows the hierarchical decomposition tree representing the analysis performed on the PV dataset: the tree shows four levels of analysis, which imply three repetitions of the second step of the methodology. More in detail, in Fig. 5.9, the root, labeled "*start*", represents the whole input domain space, while each node represents an input domain subspace increasingly smaller with the hierarchy level. Dashed line rectangles close up *insignificant* areas; solid line (colored) rectangles represent *univocal mapping* areas, while double solid line rectangles correspond to *to-subgrid* areas.

For each *to-subgrid* area found at a given analysis level, a partition is built on that to-subgrid area in the following analysis level. For example, three *to-subgrid* areas are found at first level (namely, areas 2, 5 and 9), thus three corresponding partitions will be built at second level within each level: each partition is identified by an order number (which is the same as the order number of the corresponding *to-subgrid* area in the preceding hierarchical level); so, with reference to Fig. 5.9, the first partition at the second level (consisting of *to-subgrid* area 7, *insignificant* areas 1, 2, 3, 4, 5, 6 and 9, and *univocal mapping* area 8) is pertinent to area 2, the second partition is related to area 5, finally, the third partition regards area 9.

Each set of univocal mapping areas represents a leaf in the decomposition tree. For each leaf, a fuzzy model `frbc` is built, for a total of 23 fuzzy models. We observe that the decomposition of a grid area may actually not generate any fuzzy system, e.g., due to the lack of significant sub-areas found with the decomposition.

5.4.3 Third step

In the third step we build the final fuzzy model, by merging the 23 fuzzy models previously generated. The *merged* fuzzy model consists of 83 rules and 51 and 38 fuzzy sets, respectively, for the irradiation input variable and the temperature input variable.

The generic *k*-th rule has the following format (see Equation (2.2)):

R_k: **If** irradiation **is** $l\_p\_label_I$ **and** temperature **is** $l\_p\_label_T$ **then** energy **is** $label_E$ **with** γ_k ,

where, with reference to Fig. 5.9, *l* is the level in the decomposition tree, *p* is the order number of the partition on level *l*, $label_I$, $label_T$ and $label_E$ are the labels associated, respectively, with irradiation, temperature and energy, and can be, separately, either "*Low*", "*Medium*" or "*High*".

So, e.g., the rule:

If irradiation **is** 2_3_*Low* **and** temperature **is** 2_3_*Medium* **then** energy **is** *Medium* **with** 0.39*,*

identifies a region of the input domain, to which the labels '2_3_*Low'* for irradiation and '2_3_*Medium'* for temperature correspond. The consequent class for the output variable energy is ‘*Medium’* with a certainty factor of 0.39.

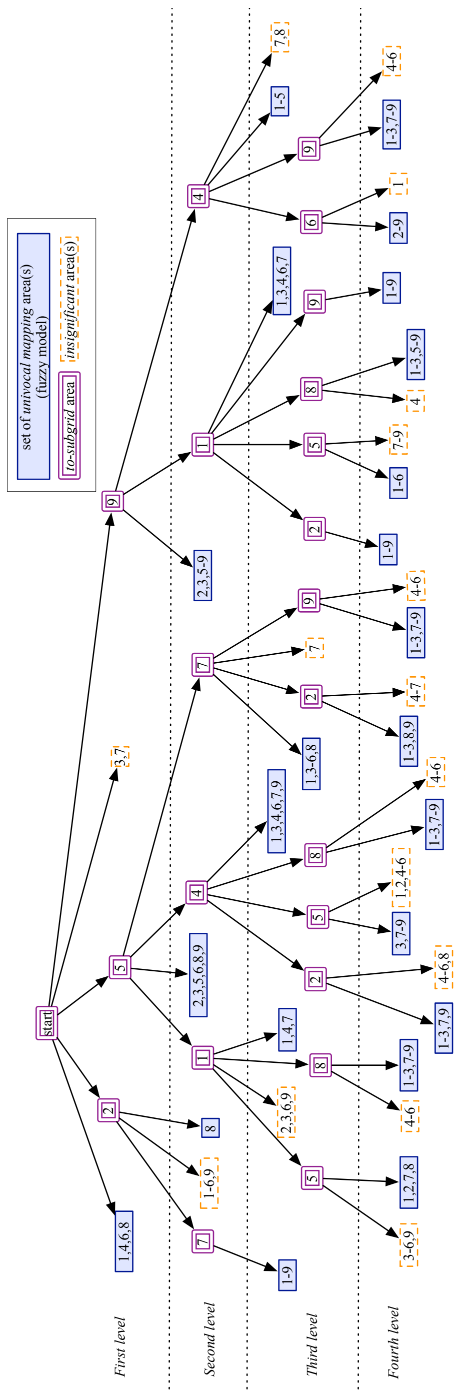


Figure 5.9. – *Hierarchical decomposition tree.*

More precisely, ‘2_3_*Low’* means that the considered region of the input domain is the *low* part of the irradiation input in the *third* partition built at the *second* level. Similarly, ‘2_3_*Medium’* means that the considered region of the input domain is the *medium* part of the temperature input in the *third* partition among the *second* level partitions. Thus, referring to Fig. 5.9, the rule corresponds to the partition of area 9.

In Fig. 5.10 we show a subset of the rule base of the merged fuzzy model. More in detail, we show 11 rules: i) the first-level rules regarding areas 1 and 4, ii) the second-level rules regarding the partition of area 9, iii) the third-level rules regarding the partition of area 9.4. Please note the use of a compact notation: *irr*, *temp* and *en* stand for irradiation, temperature and energy, respectively, while *L*, *M* and *H* refer to *Low*, *Medium* and *High*, respectively.

Figure 5.11 shows the fuzzy sets present in the rules of Fig. 5.10, used to model the linguistic variables irradiation and temperature, respectively.

Hereafter we explain the interpretation of the rules in Fig. 5.10. The rules of the first level refer to the initial input domain partition. More in detail, rules R_1 and R_2 refer to areas 1 and 4, respectively (see Fig. 5.11). The rules of the second level refer, as said earlier, to area 9. The rules of the third level refer to area 9.4. Indeed, considering, e.g., rule R_{44}, we can easily see, from the used fuzzy set labels (‘3_6_*L*’ for *irr* and ‘3_6_*L*’ for *temp*), that we are referring to the *sixth* partition built at the *third* level (see Figs. 5.8 and 5.9).

As we can see, the final fuzzy system contains a reasonable number of easily interpretable linguistic rules.

Some first-level rules

R_1: **If** *irr* **is** *L* **and** *temp* **is** *L* **then** *en* **is** *L* **with** 1.45
R_2: **If** *irr* **is** *L* **and** *temp* **is** *M* **then** *L* **with** 1.45
. . .

Some second-level rules

R_{16}: **If** *irr* **is** 2_3_*L* **and** *temp* **is** 2_3_*H* **then** *en* **is** *M* **with** 0.45
R_{17}: **If** *irr* **is** 2_3_*M* **and** *temp* **is** 2_3_*L* **then** *en* **is** *H* **with** 0.45
R_{18}: **If** *irr* **is** 2_3_*M* **and** *temp* **is** 2_3_*M* **then** *en* **is** *H* **with** 0.44
R_{19}: **If** *irr* **is** 2_3_*M* **and** *temp* **is** 2_3_*H* **then** *en* **is** *H* **with** 0.44
R_{20}: **If** *irr* **is** 2_3_*H* **and** *temp* **is** 2_3_*M* **then** *en* **is** *H* **with** 0.45
. . .

Some third-level rules

R_{44}: **If** *irr* **is** 3_6_*L* **and** *temp* **is** 3_6_*L* **then** *en* **is** *M* **with** 0.85
R_{45}: **If** *irr* **is** 3_6_*L* **and** *temp* **is** 3_6_*M* **then** *en* **is** *M* **with** 0.85
R_{46}: **If** *irr* **is** 3_6_*L* **and** *temp* **is** 3_6_*H* **then** *en* **is** *M* **with** 0.85
R_{47}: **If** *irr* **is** 3_6_*M* **and** *temp* **is** 3_6_*H* **then** *en* **is** *M* **with** 0.79
. . .

Figure 5.10. – *Part of the final rulebase (compact notation).*

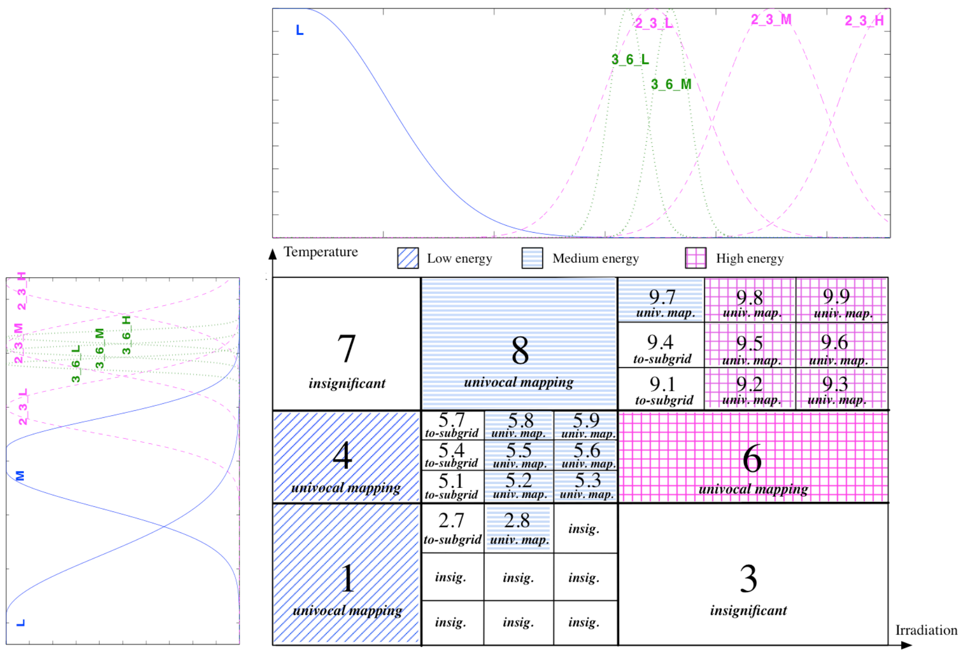


Figure 5.11. – *Fuzzy sets built by the hierarchical method to model areas 1, 4, 9, and 9.4.*

5.4.4 Genetic optimization

With reference to the PV dataset, based on heuristic considerations, we considered four hierarchical levels, i.e., three iterations of the second step of the methodology. Thus, a chromosome contains the following real genes (corresponding to the parameters to optimize): i) the *relevance thresholds* RT_1 and RT_2^i, *i*=1, 2, 3, ii) the *dominance percentages* DP_1 and DP_2^i, *i*=1, 2, 3, iii) the *maximum number S of samples extracted*, iv) the *minimum rule weight w*, and v) the *weight modifiers* Δw_1 and Δw_2^i, *i*=1, 2, 3.

For the sake of simplicity, in the experiments we set $RT_2^3 = RT_2^2 = RT_2^1$ and $DP_2^3 = DP_2^2 = DP_2^1$. Figure 5.12 depicts the final structure of the chromosome.

RT_1	RT_2^1	DP_1	DP_2^1	S	w	Δw_1	Δw_2^1	Δw_2^2	Δw_2^3

Figure 5.12. –*Structure of the GA chromosome used with the PV dataset.*

5.5 *Experimental results on the PV dataset*

In the experiments, conducted in Matlab®, on the PV dataset we started by fixing the GA-optimized model parameters based on heuristic considerations. Then, we chose the fuzzy inference process parameters of the fuzzy model based and on a preliminary analysis (see Table 5.1) in which we tried all the 56 possible inference process parameters configurations. Each configuration was tested 10 times on 10 different test sets randomly generated from the available data. Table 5.1 shows the maximum correct classification values.

Then, we performed the genetic optimization. We used stochastic uniform selection, scattered crossover with probability 0.8, and uniform mutation with probability 0.01. The population consisted of 30 individuals and the maximum number of generations was 300. These values are shown in Table 5.2, which summarizes the values for the final rule base parameters, the fuzzy inference process parameters, the GA parameters and the GA-optimized model parameters.

For each chromosome, we used the values of the genes of that chromosome to perform 30 experiments on 30 different training and test sets randomly generated from the available data. Finally we computed the fitness as the mean correct classification value on the 30 test sets. The best chromosome achieved a mean classification performance of 97.38%, with a maximum classification performance of 97.91%. The model parameters contained in the best chromosome are shown in Table 5.2.

Table 5.1. – *Application of the multi-class fuzzy classifier on the PV dataset for 56 different FRMs (best results in bold).*

Aggregation function	**Stress function**	**And operator, Implication operator**			
		Minimum, Product	***Product, Product***	***Product, Minimum***	***Minimum, Minimum***
Badd operator	***stress***	97.32	97.05	96.78	97.01
	no stress	97.41	97.13	96.78	97.1
Normalized addition	***stress***	97.32	97.05	96.78	97.01
	no stress	97.41	97.13	96.78	97.11
Arithmetic mean	***stress***	97.30	97.17	96.82	97.07
	no stress	**97.53**	97.34	96.93	97.11
Maximum	***stress***	97.11	97	96.7	97.09
	no stress	97.11	97	96.7	97.09
Sowa Or-like	***stress***	97.11	97	96.7	97.09
	no stress	97.11	97	96.7	97.09
Sowa And-like	***stress***	**97.53**	97.17	96.82	97.07
	no stress	97.30	97.34	96.93	97.11
Q.-arithmetic mean	***stress***	97.11	97	96.72	97
	no stress	97.11	96.97	96.72	97.07

Table 5.2. – *Final parameters of the merged fuzzy model for the classification of solar energy production.*

Parameter name	**Value**
Final rule base parameters	
Number of input variables	$F = 2$
Number of fuzzy sets per input variable	$Q_1 = 51$, $Q_2 = 38$
Shape of fuzzy sets	*Two-sided Gaussian, Gaussian*
Number of output classes	$M = 3$
Number of rules	$L = 83$
Fuzzy inference process parameters	
AND operator (T-norm)	*minimum*
Implication (h) operator	*product*
Stress function, g	*Square_SquareRoot* (see Equation (2.17))
Aggregation function, Γ	*Sowa and-like* (see Table 2.1)
GA parameters	
Selection	*stochastic uniform*
Crossover	*scattered* ($P_C = 0.8$)
Mutation	*uniform* ($P_M = 0.01$)
Number of individuals per population	30
Maximum number of generations	300
GA-optimized model parameters	
Relevance thresholds	$RT_1 = 15,\ RT_2^3 = RT_2^2 = RT_2^1 = 2$
Dominance percentages	$DP_1 = 80\%,\ DP_2^3 = DP_2^2 = DP_2^1 = 50\%$
Maximum number of samples extracted	$S = 145$
Minimum rule weight	$w = 0.95$
Weight modifiers	$\Delta w_1 = 0.45,\ \Delta w_2^1 = -0.55,\ \Delta w_2^2 = -0.15,\ \Delta w_2^3 = 0$

The available data at our disposal consisted of 7303 samples; based on the parameter values in Table 5.2, only 2614 (about 36%) of them were used for training.

5.6 Validation on benchmark datasets and discussion

This section aims to validate the proposed hierarchical methodology for building fuzzy classifiers. We apply the fuzzy system built following our approach (called HFRBC-GA from now on) to some well-known benchmark datasets, namely, the Fisher's Iris data, the Wisconsin breast cancer data, the Pima Indians diabetes data, and the Wine data (all available at the UCI machine learning repository [48]). We compare the mean classification performance achieved by our method in 30 executions with those obtained by other classifiers in the literature on the above-mentioned datasets.

5.6.1 Iris dataset

The Iris dataset is a commonly used benchmark for classification problems [85, 161] and it consists of 150 samples belonging to three different species of Iris flower, namely, *Setosa*, *Virginica*, and *Versicolor*. Each sample is represented by four numerical features: petal length, petal width, sepal length and sepal width. The dataset is perfectly balanced with respect to the classes.

We applied our method to this dataset and we compare our results with those achieved by some authors in the literature [4, 6, 73, 85, 90, 134, 137, 148, 153, 159].

Table 5.3 shows the results achieved by the aforementioned authors on the Iris dataset along with our results (first row of Table 5.3). For each system we report the mean number of rules generated, the number of features used for classification, the total number of fuzzy sets employed for all features, and the mean test set accuracy. Sometimes, where appropriate, we show also the maximum test set accuracy (in brackets). In the table we used the symbol '-' when no information is available. Please, note that in the first column of this table and the following ones we adopt the model acronym used by the author(s), if available; otherwise, we introduce a new acronym in quotation marks.

In this and in the following experiments, we adopted the forward feature selection to decrease the input space dimensionality. The two features used by our method are sepal length and sepal width.

The table shows that our method achieved the mean accuracy of 100%.

The values of the GA-optimized model parameters for the Iris dataset and the others benchmark datasets are shown in Table 5.4.

Table 5.3. – *Classification results on Iris dataset (best result in bold).*

Model	Mean # rules	# Features	Total # fuzzy sets	Mean (max.) test set accuracy (%)
HFRBC-GA	7.1	2	6	**100**
"Fuzzy DT" [4]	3	4	-	96.11
"HFP-2^n tree" [6]	392	4	-	95.83
HGBML [73]	10	4	-	94.67
FEBFC [85]	-	4	-	96.7
HCGA [90]	3.3	-	-	96.22
FH-GBML [134]	-	4	12	97.33
"TSK-GA" [137]	3	4	12	99.4
SoHyFIS-Yager [148]	16.33	4	-	95.66 (97.98)
SANFIS [153]	3	4	11	97.47
FNFN [159]	3.4	4	12	98.1

Table 5.4. – *Values of the GA-optimized model parameters for the benchmark datasets.*

GA-optimized model parameters	**Value**
Iris dataset	
Relevance Thresholds	$RT_1 = 5,\ RT_2^1 = 3$
Dominance percentages	$DP_1 = 80\%,\ DP_2^1 = 50\%$
Maximum number of samples extracted	$S = 6$
Minimum rule weight	$w = 0.6$
Weight modifiers	$\Delta w_1 = 0.23,\ \Delta w_2^1 = 0,\ \Delta w_2^2 = 0,\ \Delta w_2^3 = 0$
Wisconsin breast cancer dataset	
Relevance Thresholds	$RT_1 = 4,\ RT_2^1 = 7$
Dominance percentages	$DP_1 = 80\%,\ DP_2^1 = 50\%$
Maximum number of samples extracted	$S = 12$
Minimum rule weight	$w = 0.99$
Weight modifiers	$\Delta w_1 = 0.17,\ \Delta w_2^1 = -0.02,\ \Delta w_2^2 = 0,\ \Delta w_2^3 = 0$
Pima Indians diabetes dataset	
Relevance Thresholds	$RT_1 = 5,\ RT_2^1 = 2$
Dominance percentages	$DP_1 = 80\%,\ DP_2^1 = 50\%$
Maximum number of samples extracted	$S = 12$
Minimum rule weight	$w = 0.77$
Weight modifiers	$\Delta w_1 = 0.47,\ \Delta w_2^1 = -0.17,\ \Delta w_2^2 = 0,\ \Delta w_2^3 = 0$
Wine dataset	
Relevance Thresholds	$RT_1 = 8,\ RT_2^1 = 9$
Dominance percentages	$DP_1 = 80\%,\ DP_2^1 = 50\%$
Maximum number of samples extracted	$S = 6$
Minimum rule weight	$w = 0.99$
Weight modifiers	$\Delta w_1 = 0.28,\ \Delta w_2^1 = -0.06,\ \Delta w_2^2 = 0,\ \Delta w_2^3 = 0$

5.6.2 Wisconsin breast cancer dataset

The Wisconsin breast cancer dataset contains 699 samples representing two kinds of cancer (*Benign*, *Malignant*). The dataset involves nine features (clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses) and it is unbalanced (458 *benign cancer* samples, 241 *malignant cancer* samples). Since 16 samples contain missing values, we actually used 683 samples.

Table 5.5 shows the results achieved by our classifier (first row) and those of some models available in the literature [4, 39, 47, 54, 73, 85, 90, 134, 153, 159].

We achieved a mean classification accuracy of 98.32% with a maximum classification accuracy of 98.44% outperforming the other models (except for [54]). However, we adopted fewer features than [54]. The values of the GA-optimized model parameters of the methodology are shown in Table 5.4.

The three features employed in our experiments are clump thickness, uniformity of cell size, and bare nuclei.

Table 5.5. – *Classification results on Wisconsin cancer dataset (best result in bold).*

Model	**Mean # rules**	**# Features**	**Total # fuzzy sets**	**Mean (max.) test set accuracy (%)**
HFRBC-GA	14.7	3	9	98.32 (98.44)
"Fuzzy DT" [4]	2	2	3	96.82
ABA [39]	-	-	-	95.10
HFRBCS [47]	-	9	-	88.24
HFP [54]	7.8	5	-	**98.4**
HGBML [73]	20	9	-	96.68
FEBFC [85]	-	6	-	95.14
HCGA [90]	3.1	-	-	96.09
FH-GBML [134]	-	9	-	95.75
SANFIS [153]	2	9	18	96.3
FNFN [159]	1.6	9	-	98.3

5.6.3 Pima Indians diabetes dataset

The Pima Indian diabetes dataset contains 768 samples belonging to two different classes (*Diabetes positive*, *Diabetes negative*) and described by eight features (number of times pregnant, plasma glucose concentration, diastolic blood pressure, triceps skin fold thickness, 2-Hour serum insulin, body mass index, diabetes pedigree function, and age). The dataset presents a significant class overlap [Chang 20] that usually makes it difficult to obtain high classification accuracy.

The dataset is unbalanced (268 *diabetes positive* samples, 500 *diabetes negative* samples).

Table 5.6 shows the achieved results (first row) compared with some models found in the literature [4, 39, 47, 73, 134].

Our mean accuracy, obtained using only three features, outperforms all the models considered. We achieved a mean classification accuracy of 78.31% with a maximum of 80.30%. The three features are: number of times pregnant, plasma glucose concentration, and diabetes pedigree function. The values of the GA-optimized model parameters tailored to this classification problem are shown in Table 5.4.

Table 5.6. – *Classification results on Pima Indians diabetes dataset (best result in bold).*

Model	**Mean # rules**	**# Features**	**Total # fuzzy sets**	**Mean (max.) test set accuracy (%)**
HFRBC-GA	63.6	3	90	**78.31** (80.30)
“Fuzzy DT” [4]	11.2	8	-	73.05
ABA [39]	-	-	-	74.8
HFRBCS [47]	-	8	-	68.72
HGBML [73]	20	8	-	75.83
FH-GBML [134]	-	8	-	75.91

5.6.4 Wine dataset

The Wine dataset contains 178 samples representing Italian wines belonging to three different cultivars. Each wine is described by thirteen features resulting from chemical analysis (alcohol, malic acid, ash, alcalinity of ash, magnesium, total phenols, flavanoids, non flavanoid phenols, poanthocyanins, color intensity, hue, OD280/OD315 of diluted wines, and proline). The class balancing is the following: 59, 71, and 48.

Table 5.7 (first row) shows that the results we achieved using only three features outperform all the considered models [4, 54, 72, 73, 90, 134, 137, 153, 159].

The three features considered are alcohol, flavanoids, and non flavanoid phenols. We achieved a mean classification accuracy of 99.41% with a maximum of 100%. The values of the GA-optimized parameters of the methodology tailored to the Wine classification problem are shown in Table 5.4.

Table 5.7. – *Classification results on Wine dataset (best result in bold).*

Model	**Mean # rules**	**# Features**	**Total # fuzzy sets**	**Mean (max.) test set accuracy (%)**
HFRBC-GA	10.2	3	9	**99.41** (100)
“Fuzzy DT” [4]	3.6	13	-	91.22
HFP [54]	6.8	4	-	89.2
HGBML [73]	10	13	-	95.06
“RW” [72]	15	13	-	95.51
HCGA [90]	4.9	-	-	95.66
FH-GBML [134]	-	13	-	93.79
“TSK-GA” [137]	3	9	21	98.3
SANFIS [153]	3	13	34	99.4
FNFN [159]	1.2	13	-	99.1

5.7 Concluding remarks

In this chapter we have proposed a hierarchical method to construct a fuzzy classifier by merging fuzzy systems built on input domain regions increasingly smaller, as the result of the creation of appropriate grids on the input domain. The aim is to exploit the easiness of use and the interpretability of the fuzzy approach along with a methodology of input domain space analysis which builds an optimal fuzzy rule base avoiding the generation of too many, unnecessary rules. The model parameters are optimized by a real-coded GA.

We developed a fuzzy classifier aimed at classifying the energy produced by a PV panel as either low, medium, or high based on the irradiation and the temperature of the panel. Experimental results have showed mean and maximum classification performances of 97.38% and 97.91%, respectively, on the test sets of 30 repetitions of the classification experiment.

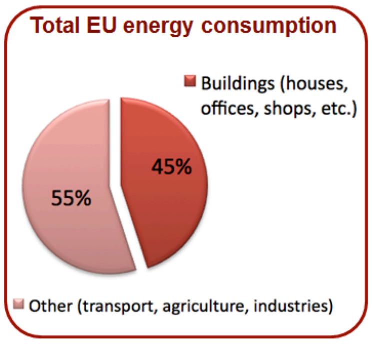
The performance of the proposed approach has also been successfully validated by building fuzzy classifiers for some well-known benchmark datasets. The achieved results outperform those obtained by other methods found in the literature.

6

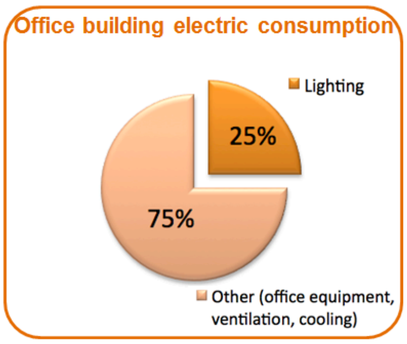
Neural network-based forecasting of energy consumption due to lighting in office buildings

6.1 Introduction

Energy consumption in buildings is one of the fastest growing sectors. It is estimated that the amount of the energy currently consumed in the European buildings is about 40–45% of the total European energy consumption [40, 44], as shown in Fig. 6.1(a). Buildings include shops, houses, offices, etc., but office buildings represent the largest share. In particular, as regards electricity consumption in office buildings (see Fig. 6.1(b)), earlier works have shown that electric lighting is a big component of electricity consumption: it accounts for about 25% of total electricity consumption [44, 83, 158]. The remainder 75% is due to HVAC (Heating, Ventilation, and Air Conditioning) and office equipment (PCs, printers, etc.).



(a)



(b)

Figure 6.1. – *Statistics about energy consumption.* (a) *Total European energy consumption.* (b) *Office buildings electric consumption.*

Although the electric power consumption due to lighting is not the highest one in a building, it is present throughout the working day and it deserves to be taken into account alone for forecasting purposes as it represent one quarter of the total office electric consumption.

The potential benefits of knowing energy consumption, in real time or even in advance, can be useful for several purposes, ranging from cost reduction, improved energy control, and smarter load scheduling in the electric grid, especially in the case of smart grids. In addition, the European Commission has adopted a plan to reduce energy consumption of 20% by 2020 [43, 44, 97], by promoting energy efficiency, so the possibility of energy consumption forecasting is of the utmost importance.

On the one hand, electric Energy Consumption due to Lighting (ECL) could be directly estimated knowing the lighting equipment: the kind and the number of lights existing in the office and their operating time. From another point of view, one could expect that the amount of ECL should be inversely proportional to the amount of natural daylight in the office, so electric ECL could be estimated starting from some knowledge about natural daylight. In this case, we need to know i) the global solar irradiation model, ii) the position of the building, its orientation with respect to the path of the sun, and the kind of glazed surfaces (i.e., windows effect of shading devices), iii) the weather conditions, in particular the sky conditions, which can influence the measured value of solar irradiation and thus the quantity of daylight available, and subsequently the values of energy consumption in buildings [78].

Actually, the main problem of such models is that all this kinds of information are not always easy to achieve. In addition, we need to take into account the unpredictable component given by occupants' needs and behavior regarding the use of lights.

Hence, we propose a way of estimating the electric ECL, using mainly the solar irradiation data and assuming a fairly good behavior of the occupants, already discussed in [36]. By "good behavior" we mean a rational, "green" behavior of people that try to assure the necessary visual comfort inside the office [59, 113] by paying attention to energy savings.

Several techniques have been traditionally used for energy use forecasting. Among them, we can recall some time series analysis classical techniques such as ARIMA [1, 132] and regression [23, 104, 120]. Unfortunately, the correlation between solar irradiation and electric energy consumption is highly nonlinear, thus making classical techniques not well suited to solve this kind of problems. So in the last few years there has been a growing interest in computational intelligence tools, in particular, neural networks [25, 53, 105, 144, 158], expert systems [124], genetic algorithms [149], and hybrid systems, i.e., systems resulting, e.g., from the integration of neural networks and fuzzy logic [108, 117, 141]. In particular, the literature has demonstrated the superior capability of neural networks over conventional methods, thanks to the high potential to model non-linear processes, such as individual buildings energy consumption [78].

Hence, in this chapter, we propose an artificial neural network model to forecast the electric ECL of a small office building, starting from some knowledge about the external daylight, without having to know any kind of information about the building, the lighting equipment, and the occupancy of the office, as usually happens when using simulation tools. In fact, to have to know in advance all these kinds of information may be a drawback [59]. The two key points of the methodology are: i) the construction of a proper reference solar irradiation curve, and ii) the division of the working day into an appropriate number of time intervals.

6.1.1. Outline of the chapter

This chapter is organized as follows. Section 6.2 describes the experimental dataset concerning an office building located in Italy, Section 6.3 presents the proposed model to predict the electrical ECL. More in detail, first, we describe the analysis and elaboration of data, and then, we discuss the design of the forecasting model by setting the values of some model parameters. Section 6.4 shows the performed experiments and the achieved results, and, finally, Section 6.5 provides concluding remarks and future work.

6.2 Description of the building consumption dataset

The data used in this work were collected from the sensors of an office building located in Tuscany, Italy. The data were measured every 15 minutes during six months, from April to September 2011 and consisted of i) solar irradiation outside the building, measured by a meteorological station, ii) sampling timestamp (date and time), and iii) lighting electricity consumption, expressed in terms of active power averaged over 15-minute intervals, measured by a multimeter and related to the use of lights in four rooms.

In addition, we acquire from data one more input: the day of the week, in order to model further cyclic activities, such as cleaning tasks, periodic meetings, etc. We chose not to use an explicit information regarding the kind of day (working or weekend) in order to maintain the model as general as possible.

Table 6.1. – *Characteristics of the building.*

Characteristic name	**Characteristic value**
Location	Tuscany, Italy
Office specifications	Ground floor, four rooms
Obstacles	An obstacle (a tree) obscures the irradiation sensor at about 10 a.m.
Usual office (business) operation hours	Monday to Friday: 9 a.m.–9 p.m.; Otherwise: closed (occasionally open for maintenance)
Cleaning schedule	Tuesday to Friday (7 a.m.–8 a.m.) and on Saturday morning (the actual time is not fixed)

The input variables of the model are the results of a processing made on the available data, as better explained in the following. The output variable is the average active power over intervals of a few hours (time interval), which is known to be the real power transformed into work and represents the real consumption of the time interval. Moreover, we had some information about the cleaning schedule, the working hours, and the presence of obstacles in front of the irradiation sensor (see Table 6.1). These data helped us to correctly analyze and interpret the experimental results.

6.3 The proposed model

The proposed model consists of an artificial neural network and aims at forecasting the energy consumption of an office building, over intervals of a few hours, due to lighting.

As we are concerned with a small office, we deal with very small values of consumption if compared with those found in the literature, which are related to big buildings or include the total HVAC systems consumption. Furthermore, the values we are concerned with are highly irregular.

The correlation between solar irradiation and electric energy consumption is not straightforward. Actually, when a building is in use, the intensity and the stability of the solar irradiation can be considered as the factors that have the greatest influence on the decrease and increase in lighting consumption. By intensity we mean the absolute value of irradiation at every considered instant. It is obvious that the artificial lights are used because the value of the lighting is not enough inside the building. If in a hypothetical case the solar irradiation had the same time evolution every day, the electric power consumption would not vary significantly from one day to another. Actually, the electric power consumption trend presents many differences between days as Fig. 6.2 clearly shows.

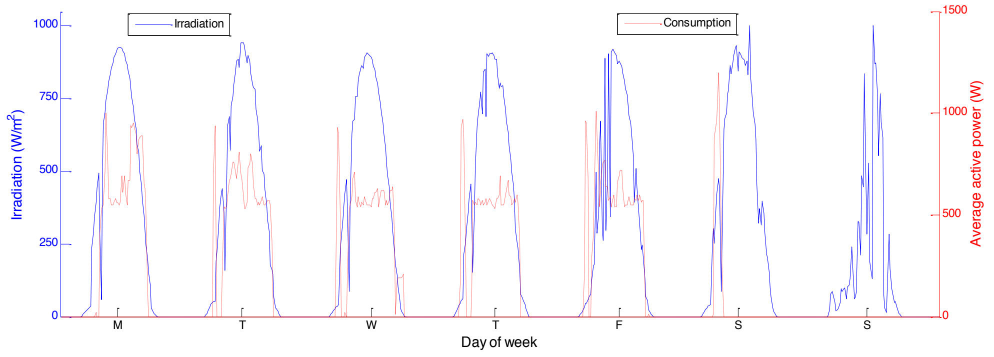


Figure 6.2. – *Evolution of consumption (red dotted line) and irradiation (blue solid line) from Monday May 9th to Sunday May 15th.*

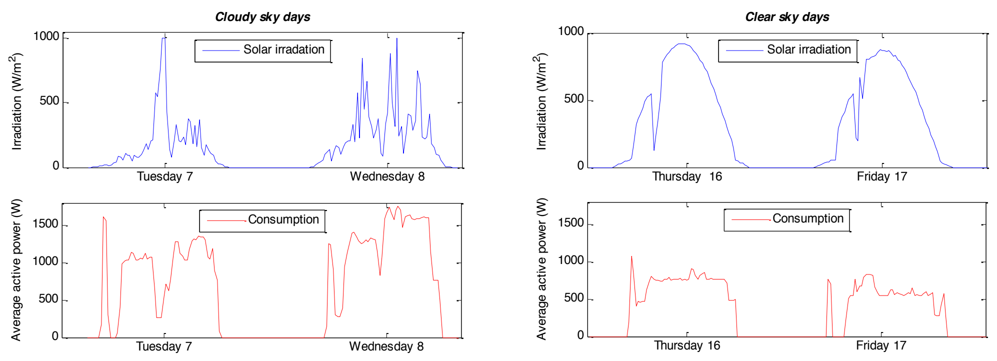
Figure 6.2 shows the evolution of the electrical consumption and solar irradiation for seven days of a typical Spring week (from May 9th, Monday, to May 15th, Sunday). Please note the partial and total absence of consumption on Saturday and Sunday, respectively.

Therefore we need to make some considerations to exploit the available data so as to reproduce the relation between solar irradiation and electrical lighting consumption.

6.3.1 Effects of climatic contest: analysis of solar irradiation

The first consideration regards the analysis of the solar irradiation trend. Solar irradiation and thus daylight availability and intensity mainly depend on geographic latitude and on climatic contest. In particular, latitude-related variations are caused by changes in the sun position in the sky with the latitude during the day or during the year. Climate-related variations, also called *sky conditions*, can be classified as clear, partly cloudy, and overcast [84, 89] and may deeply affect energy consumption, as shown in Fig. 6.3.

We noticed that *cloudy sky days* (Fig. 6.3(a)) present quite a lot of variations in the solar irradiation values (probably due to weather fluctuations, such as clouds moving caused by the wind), and to these days correspond a mostly irregular and pretty high electrical consumption. Instead, *clear sky days* (Fig. 6.3(b)), characterized by a regular solar irradiation trend, present a regular and lower electrical consumption. Besides, we have confirmed the above mentioned hypothesis, by considering the average daily consumptions (e.g., 640 W on Wednesday 8th, and 359 W on Thursday 16th).



(a) (b)

Figure 6.3. – *Solar irradiation and electrical consumption for four days of June with different sky conditions*: (a) *two cloudy sky days*, and (b) *two clear sky days.*

It is clear that there is a relation between the daily solar irradiation trend and the electrical consumption due to the use of lighting. It seems that if external daylight is enough, lights in the office are less used.

Moreover, another consideration regarding solar irradiation is the following. We found that, given the same sky conditions in two consecutive days of a given climatic period, the daily solar irradiation curve slightly changes from one day to the following (see the first three days in Fig. 6.4). On the other hand, if the sky conditions in the considered climatic period change, the irradiation curves differ from each other to a greater extent (see the last day in Fig. 6.4). In any case, over periods of about *one month*, most daily irradiation curves appear to be very close to each other. In addition, a month can be considered as an important time unit from the weather point of view [74], and works in the literature often deal with monthly analysis [10, 11].

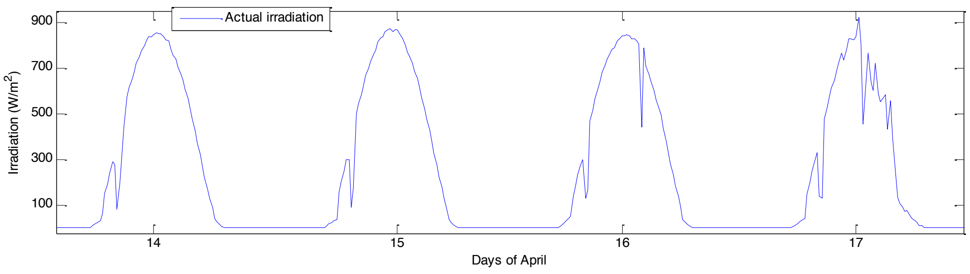
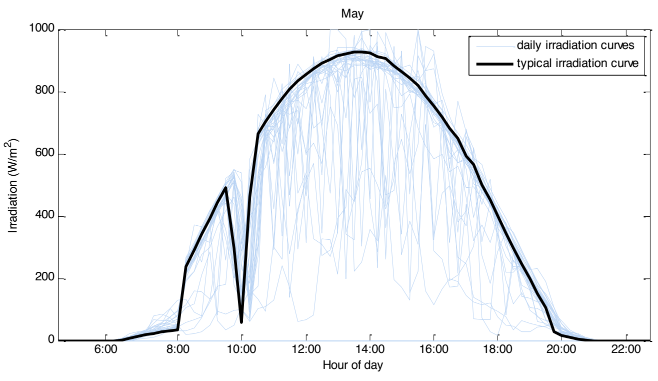


Figure 6.4. – *Solar irradiation curves for four consecutive days of April.*

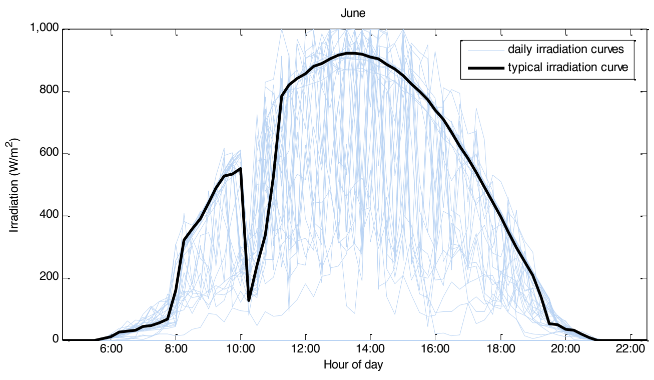
Based on these considerations, we can consider a single solar irradiation *reference* curve for each month, by simply using the curve that best approximates all the curves of the month. From now on we will call this curve *ideal typical irradiation curve*. Figure 6.5(a), 6.5(b) and 6.5(c) show the ideal typical irradiation curve for May, June, and September respectively. More precisely, from an operation point of view, for each month we have performed the following steps. After superimposing the irradiation curves relative to all days of that month (as done in Fig. 6.5), we have noticed that most curves almost coincide, in the sense that their difference is very small in all points. So, after removing the few curves that represent outliers, we have generated the curve that best approximates all the involved curves.

In Fig. 6.6 we show the typical ideal irradiation curves for the six months available, from April to September. In Figs. 6.5 and 6.6 we can also notice that the discontinuity present in all curves at about 10 a.m., due to the presence of a tree that obscures the irradiation sensor, slightly changes as a result of the movement of the sun. Our choice to keep this discontinuity in the ideal typical curves stems from the will to use this curve to faithfully model the particular spatial context of the considered building.

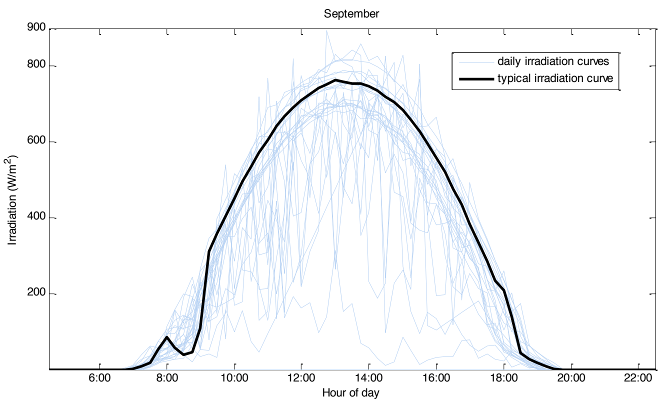
We may say that the reference solar irradiation curve represents the *typical day* of the month. So we can use the typical ideal irradiation curve to characterize each day in the month or distinguish among the different kinds of days. More precisely, we exploit the difference between the ideal curve and the actual daily irradiation curve, so as to highlight how the considered day differs from the typical day of that month.



(a)



(b)



(c)

Figure 6.5. – *Daily irradiation curves and corresponding typical ideal irradiation curve (black line) for* (a) *May,* (b) *June and* (c) *September.*

Figures 6.7(a) and 6.7(b) show the typical ideal irradiation curves and actual irradiation curves for four days (from Tuesday September 6th to Friday September 9th) and the corresponding differences, respectively. The third day (i.e., September

8th) in Fig. 6.7(b) presents a very small difference so we may consider that day as a typical day. In fact, the actual irradiation curve perfectly follows the ideal curve. On the contrary, the other days of Fig. 6.7 present a not negligible difference between the ideal and actual irradiation curve, so we can use this difference between the daily actual irradiation curve and the typical ideal curve as further input parameter to our system, as better explained in the following.

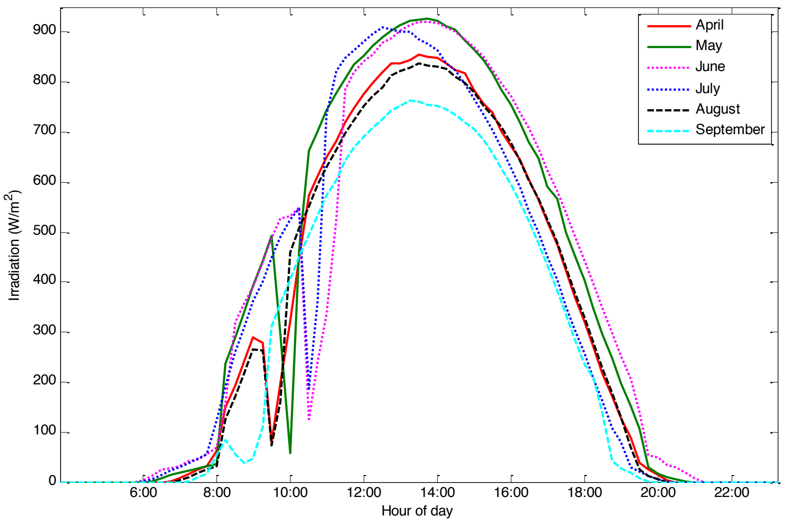
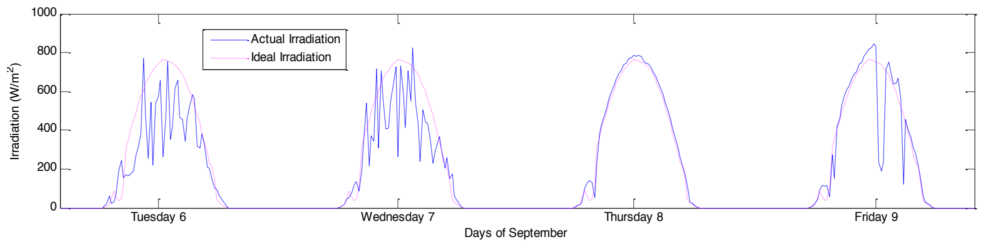
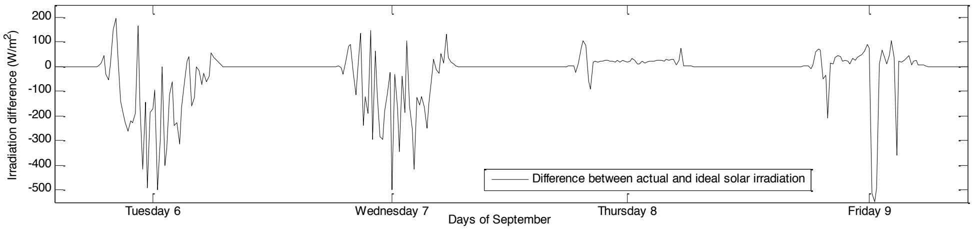


Figure 6.6. – *Typical ideal irradiation curves for six months (April to September).*



(a)



(b)

Figure 6.7. – (a) *Typical ideal irradiation curve (magenta dotted line) and actual irradiation curve (blue solid line) for four days of September*. (b) *Difference between the two irradiation curves.*

6.3.2 Analysis of energy consumption based on the office use

The third consideration concerns the analysis of the energy consumption in relation with the specific use of the office taken into account.

Figure 6.8 shows the evolution of the energy consumption during a typical working day in terms of average active power. The working time goes from 9 a.m. to 9 p.m., so we have considered the energy consumption in this time interval, even though the presence of a spike at about 8 o'clock in correspondence of the cleaning time of the office.

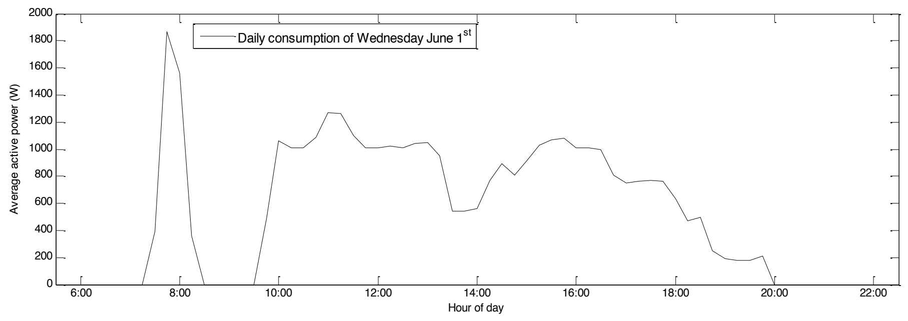


Figure 6.8. – *Evolution of the energy consumption for a typical working day. Please note that, for the considered day, the working time ends at 8 p.m..*

We have tried to split the working day into a small number of intervals with the aim of taking into account the use of the office building in the various parts of the day. In order to identify the most appropriate number of these intervals, we have tried several combinations and we finally found out that the working time can be profitably subdivided into three intervals of *four* hours each, as shown in Fig. 6.9. We used the average value within each considered time interval instead of all the instantaneous values because this is the usual practice found in the literature, although relative to daily average values [158].

Our goal is to predict the average energy consumption of a given time interval based on information pertinent to the previous interval. Of course, in order to perform the prediction of the electrical consumption related to the first interval, we added a service interval, called *interval 0*, from 5 a.m. to 9 a.m. (the dotted interval in Fig. 6.9).

Finally, by analyzing the energy consumption, we found daily and weekly cyclicity in the data. Figure 6.10 shows the ECL in two consecutive weeks of April, randomly selected among the available data. From Monday (Mo) to Friday (Fr) there is a considerable variability of electric power consumption in the working hours, and on Saturdays (Sa) there is only consumption in a little span of one hour. In addition, in four days a week (from Tuesday (Tu) to Friday) and on Saturday morning at a non-fixed time, there is a spike, relative to a high electric power use for about one hour outside of the working time in correspondence with the cleaning time of the office. Clearly, on Sundays (Su) the energy consumption is completely absent.

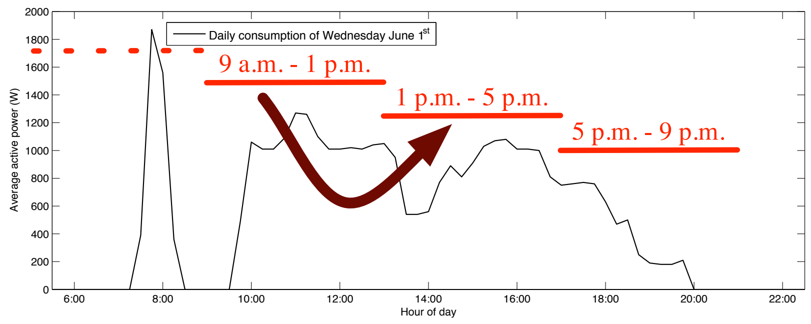


Figure 6.9. – *Identification of the time intervals over the working time.*

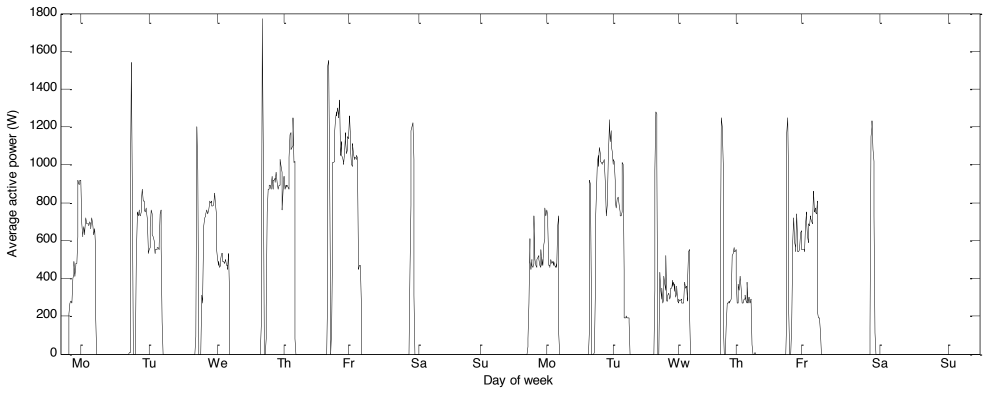


Figure 6.10. – *Temporal evolution of the energy consumption for two consecutive weeks of April.*

6.3.3 Discussion

After the processing of data we are now able to design the forecasting model, by setting the values of the model parameters, and by fixing the inputs and the output of the network.

Regarding the values of the model parameters, we can state that:

a) the climatic period considered to build the reference irradiation curve is, as stated before, *one month*;
b) the dimension of each time interval is *four hours*. So, being the working day of twelve hours, we split each day in *three intervals*.

The input parameters to our system are the following: i) day, ii) month, iii) time (hour and minutes), iv) mean difference between actual and ideal irradiation for the considered time interval, v) instantaneous difference between actual and ideal irradiation at that time, vi) average actual irradiation for the considered interval. All these input parameters are pertinent to a given time interval. The output parameter from our system is the average energy consumption, expressed in terms of average active power, for the following interval. Table 6.2 shows the inputs and output of the system, and their units of measurement.

Table 6.2. – *Inputs and output variables of the neural network.*

Variable name	**Unit of measurement**
Input variables	
Day	Encoding: from 1 to 7
Month	Encoding: from 1 to 12
Timestamp (hour, minute)	Encoding: hour:minute
Mean difference for the considered time interval between actual irradiation and ideal irradiation	W/m^2
Difference between actual and ideal irradiation for the considered timestamp	W/m^2
Average actual irradiation for the considered time interval	W/m^2
Output variable	
Average energy consumption for the following time interval	W

6.4 Experimental results

We used a feed-forward neural network having one hidden layer, to implement the proposed approach in Matlab®. The transfer functions for the hidden neurons and the output neuron are, respectively, hyperbolic tangent sigmoid and linear functions.

By using a trial-and-error strategy, we tried different neural configurations by varying the number of hidden neurons from 10 to 30 with step 1. For each of the training, validation and test sets, the same number of, respectively, training, validation and test samples is randomly extracted for each month so as not to create unbalanced sets. The percentages of samples extracted for each month are 60%, 10% and 30%, respectively, for the three sets.

Each kind of experiment has been repeated 30 times. The best configuration resulted to be a neural network with 28 hidden neurons in the hidden layer. Table 6.3 summarizes the parameter values of the chosen neural configuration.

The aim of the performed experiments was the comparison between the average actual electrical consumption in a given time interval and the average predicted electrical consumption in the same interval.

For each month, we computed the average Mean Squared Error (MSE) on all the intervals of the days of the month, according to the following equation:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(EC_{act}(i) - EC_{pred}(i))^2 , \tag{6.1}$$

with N representing the number of intervals of the considered month, and EC_{act} and EC_{pred} being the average *actual* electrical consumption and the average *predicted* electrical consumption respectively, for each considered interval. Then, so as to obtain fair values, we repeated the same experiment 30 times with different training and test sets and we averaged the MSEs obtained over the 30 trials, by obtaining an average error MSE_{av} for each month.

Table 6.3. – *Learning parameters for the final neural network.*

Parameter	**Value**
Number of hidden layers	1
Number of hidden neurons	28
Hidden Layer tansfer function	Hyperbolic tangent sigmoid
Output Layer tansfer function	Linear
Training algorithm	Levemberg-Marquardt
Number of training samples	60% of available data
Number of validation samples	10% of available data
Number of test samples	30% of available data

Furthermore, in order to make the error easier to understand, we computed for each month the corresponding average RMSE (Root Mean Squared Error), $RMSE_{av}$, and the normalized RMSE ($RMSE_n$) given by the following equation:

$$RMSE_n = \frac{RMSE_{av}}{EP_{av}} , \tag{6.2}$$

where EP_{av} is the average electric power for the considered month, i.e., the average consumption of the month.

Table 6.4 shows the average monthly MSE (MSE_{av}), the corresponding RMSE ($RMSE_{av}$), the monthly average electric power (EP_{av}), and the normalized RMSE ($RMSE_n$), for the six months. Please note that in the available data there are some missing days in correspondence with sensor maintenance. In particular, there are 6, 1, 6 and 6 missing days for April, May, August and September. Obviously, these numbers have been taken into account for computing the MSE values.

As we can see from Table 6.4, we achieve good results for April, May, June and July. Poorer results are obtained in August and September. In both cases, the high error is most likely due to the presence of frequent weather changes within the same day, as we observed by analyzing the irradiation curves of August and September. In such conditions the electric power consumption can be sensibly different from what expected, thus causing an increased prediction error. Further, in August, we must also take into account the partial, non-regular daily occupancy of the office building during summer vacation, as testified by the low monthly electric lighting consumption. This situation obviously increases the prediction error.

Therefore, we can state that the RMSE error ranges from 4.84% to 38.27% of the monthly average electric power, with an average value of 17.25%. In absolute terms, the minimum and maximum RMSEs are 18.45 W and 110.05 W, obtained for August and September, respectively.

Finally, to assess our results we applied the persistence method, which assumes as target value to forecast the average energy consumption of a given interval, the average energy consumption of the same interval of the previous day. We achieved an average monthly MSE one order of magnitude greater than the values shown in Table 6.4.

Table 6.4. – *Monthly performances of the forecasting system.*

Month	**MSE_{av}**	**$RMSE_{av}$ (W)**	**EP_{av} (W)**	**$RMSE_n$ (%)**
April	548.1	23.41	483.13	4.84
May	2058.9	45.37	408	11.12
June	2647.3	51.45	487	10.56
July	1816.8	42.62	307.24	13.87
August	340.72	18.45	48.23	38.27
September	12113	110.05	443.23	24.83

6.5 Concluding remarks

In this chapter a novel method to predict the ECL, mainly based on solar irradiation, has been described. The forecasting system predicts the average active power in a given time interval of the working day exploiting the difference between a reference ideal irradiation curve (pertinent to the specific month) and the actual irradiation curve in the previous time interval . In this way the electric energy consumption is essentially influenced by the quantity of available external daylight.

We used a feed-forward artificial neural network, which was applied to a case study concerning a small office building located in Italy. In the experiments, made on the data pertinent to six months, the average RMSE error represents 17.25% of the monthly average electric power.

As a future work, we are planning to extend the presented analysis to the heating and cooling consumption, in order to build a monitoring and simulation tool able to estimate the total HVAC consumption of an office building.

Moreover, it could be interesting to integrate the proposed system with a weather forecasting system, so as to estimate directly the solar irradiation values.

7

Thesis conclusions and future work

The aim of this thesis was the design and the development of some Computational Intelligence novel methodologies for applications regarding energy systems. More in detail, two main problems have been addressed. First, the management of energy production in solar PV installations, by classification and forecasting, second, the forecasting of energy consumption in buildings.

In the early chapters, we recalled the main concepts about artificial neural networks, fuzzy rule-based classification systems, and hybrid systems exploiting genetic algorithms. In addition, we revised some notions about forecasting and time series analysis. In the following chapters, we presented the developed methodologies applied to the problems described above.

Regarding the management of energy production in solar PV installations, we faced the problem by building a one day-ahead energy production forecasting model. The model, resulting from a flexible and easy-to-use approach, uses the NARX time series analysis model and a neural network with tapped delay lines to reproduce the curve of daily produced energy starting from some knowledge about solar irradiation. Despite the existence of some methods for energy forecasting problems, the main novelty of our approach is the proposal of a general methodology, consisting of a sequence of steps to perform in order to find the optimal structure of the neural network (particularly, number of hidden neurons and number of delay elements) and the best configuration of the neural predictor (namely, the training window width and the sampling frequency). As a future work, thanks to the good results obtained, we may extend the forecasting horizon to a multiple of the day, so as to strengthen the long-term forecasting analysis. Besides, it would be of interest to integrate the proposed system with a weather forecasting system, so as, e.g., to estimate directly the environmental variables.

The management of energy production in solar PV installations can be tackled also from a different point of view, that is, as a fuzzy classification problem of energy production, given the values of the irradiation and the temperature of the panel, so as to linguistically describe how the inputs of the fuzzy classification system (i.e., the temperature of the PV panel and the solar irradiation) relate to the class (*low*, *medium*, *high*) of the energy production. The main advantages of our approach are easier interpretability and versatility, as we deal with class labels. In addition, the model parameters are optimized by means of a genetic algorithm. The fuzzy classifier results from the union of fuzzy systems (`frbc`), built on input regions increasingly smaller, according to a hierarchical multi-level grid-like partition. Only the necessary partitions are built, in order to guarantee high interpretability and to avoid the explosion of the number of rules as the hierarchical level increases. The fuzzy classifier was also successfully applied to several benchmark datasets, thus proving the validity of the methodology.

Lastly, we address the problem of the forecasting of energy consumption due to lighting in office buildings. The system was developed as a feed-forward artificial neural network, which predicts the energy consumption related to a few hours, using some knowledge about external daylight, without having to know any kind of information about the building. As a future work, we aim to build a system for estimating the total HVAC consumption of an office building, by extending the analysis to the heating and cooling consumption. Furthermore, it would be of interest to build a simulation tool to simulate energy optimization actions.

Acknowledgement

We wish to acknowledge Samares, Via Giuntini, 25, 56023, Navacchio di Cascina, Pisa (Italy), for having provided the sets of experimental data used in this thesis, namely three PV dataset related to three different PV installations sited in Italy, and the energy consumption dataset related to an office building located in Italy.

References

[1] Abdel-Aal R.E., Al-Garni A.Z., “Forecasting monthly electric energy consumption in eastern Saudi Arabia using univariate time-series analysis”, *Energy*, vol. 22, n. 11, 1997, pp. 1059–1069.

[2] Abe S., Lan M.-S., “A method for fuzzy rules extraction directly from numerical data and its application to pattern classification”, *IEEE Transanctions on Fuzzy Systems*, vol. 3, n. 1, 1995, pp. 18–28.

[3] Abe S., Thawonmas R., “A fuzzy classifier with ellipsoidal regions”, *IEEE Transactions on Fuzzy Systems*, vol. 5, n. 3, 1997, pp. 358–368.

[4] Abonyi J., Roubos J.A., Szeifert F., “Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision-tree initialization”, *International Journal of Approximate Reasoning*, vol. 32, n. 1, 2003, pp. 1–21.

[5] Agrawal R., Srikant R., “Fast algorithms for mining association rules”, Proceedings of the 20th International Conference on Very Large Databases, Santiago, Chile, 1994, pp. 487–499.

[6] Ait Kbir M., Benkirane H., Maalmi K., Benslimane R., “Hierarchical fuzzy partition for pattern classification with fuzzy if-then rules”, *Pattern Recognition Letters*, vol. 21, 2000, pp. 503–509.

[7] Alonso J.M., Magdalena L., González-Rodríguez G., “Looking for a good fuzzy system interpretability index: an experimental approach”, *Journal of Approximate Reasoning*, vol. 51, n. 1, 2009, pp. 115–134.

[8] Angelov P.P., Buswell R.A., “Automatic generation of fuzzy rule based models from data by genetic algorithms”, *Information Sciences*, vol. 150, n. 1–2, 2003, pp. 17–31.

[9] Ashraf I., Chandra A., “Artificial neural network based models for forecasting electricity generation of grid connected solar PV power plant”, *International Journal of Global Energy*, vol. 21, n. 1–2, 2004, pp. 119–130.

[10] Azadeh A., Ghaderi S.F., Sohrabkhani S., “A simulated-based neural network algorithm for forecasting electrical energy consumption in Iran”, *Energy Policy*, vol. 36, n. 7, 2008, pp. 2637–2644.

[11] Azadeh A., Ghaderi S.F., Sohrabkhani S., “Forecasting electrical consumption by integration of Neural Network, time series and ANOVA”, *Applied Mathematics and Computation*, vol. 186, n. 2, 2007, pp. 1753–1761.

[12] Azadeh A., Maghsoudi A., Sohrabkhani S., “An integrated artificial neural networks approach for predicting global radiation”, *Energy Conversion and Management*, vol. 50, n. 6, 2009, pp. 1497–1505.

[13] Barbounis T.G., Theocharis J.B., Alexiadis M.C., Dokopoulos P.S., “Long-term wind speed and power forecasting using local recurrent neural network models”, *IEEE Transactions on Energy Conversion*, vol. 21, n. 1, 2006, pp. 273–284.

[14] Benghanem M., Mellit A., Alamri S.N., “ANN-based modelling and estimation of daily global solar radiation data: A case study”, *Energy Conversion and Management*, vol. 50, n. 7, 2009, pp. 1644–1655.

[15] Bhattacharya T.K., **Design and development of solar photovoltaic systems**, vol. 1, New Delhi: Publications & Information Directorate, 1991.

[16] Bishop C., **Neural networks for pattern recognition**, Oxford University Press, Oxford, UK, 1995.

[17] Cadenas E., Rivera W., “Short term wind speed forecasting in La Venta, Oaxaca, Mexico, using artificial neural networks”, *Renewable Energy*, vol. 34, n. 1, 2009, pp. 274–278.

[18] Caputo D., Grimaccia F., Mussetta M., Zich R.E, “Photovoltaic plants predictive model by means of ANN trained by a hybrid evolutionary algorithm”, Proceedings of the 2010

International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 18–23 July 2010, pp. 1–6.

[19] Casillas J., Cordón O., del Jesus M.J., Herrera F., “Genetic feature selection in a fuzzy rule-based classification system learning process for high-dimensional problems”, *Information Sciences*, vol. 136, n. 1–4, 2001, pp. 135–157.

[20] Chang X., Lilly J.H., “Evolutionary design of a fuzzy classifier from data”, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, n. 4, 2004, pp. 1894–1906.

[21] Chen Y., Wang T., Wang B., Li Z., “A Survey of fuzzy decision tree classifier”, *Fuzzy Information and Engineering*, vol. 1, n. 2, 2009, pp. 149–159.

[22] Cherkassky V., “Fuzzy Inference Systems: A Critical Review”, in **Computational Intelligence: soft computing and fuzzy-neuro integration with applications**, pp. 177-197, O. Kaynak *et al.*, Springer Berlin Heidelberg, 1998.

[23] Cherkassky V., Chowdhury S.R., Landenberger V., Tewari S., Bursch P., “Prediction of electric power consumption for commercial buildings”, Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN), 31 July–5 August 2011, pp. 666–672.

[24] Chiu S., “Extracting fuzzy rules from data for function approximation and pattern classification”, ch. 9 in **Fuzzy Information Engineering: A Guided Tour of Applications**, D. Dubois, *et al.*, John Wiley and Sons, 1997.

[25] Chow T.W.S., Leung C.T., “Neural network based short-term load forecasting using weather compensation”, *IEEE Transactions on Power Systems*, vol. 11, n. 4, 1996, pp. 1736–1742.

[26] Cococcioni M., D'Andrea E., Lazzerini B., “24-hour-ahead forecasting of energy production in solar PV systems”, Proceeding of the 11th International Conference on Intelligent Systems Design and Applications (ISDA), Córdoba, Spain, 22–24 November 2011.

[27] Cococcioni M., D'Andrea E., Lazzerini B., “One day-ahead forecasting of energy production in solar photovoltaic installations: An empirical study”, *Intelligent Decision Technologies*, vol. 6, 2012, pp. 1–14.

[28] Cococcioni M., D'Andrea E., Lazzerini B., “Providing PRTools with fuzzy rule-based classifiers”, Proceedings of the 2010 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Barcelona, Spain, 18–23 July 2010, pp. 1–8.

[29] Cococcioni M., D'Andrea E., Lazzerini B., Volpi S.L., “Short-time forecasting of renewable production energy in solar photovoltaic installations”, Proceedings of 2010 International Conference on Competitive and Sustainable Manufacturing, Products and Services (APMS), Como, Italy, 11–13 October 2010b.

[30] Cordón O., del Jesus M.J., Herrera F., “A proposal on reasoning methods in fuzzy rule-based classification systems”, *International Journal of Approximate Reasoning*, vol. 20, n. 1, 1999, pp. 21–45.

[31] Costa A., Crespo A., Navarro J., Lizcano G., Madsen H., Feitosa E., “A review on the young history of the wind power short-term prediction”, *Renewable and Sustainable Energy Reviews*, vol. 12, 2008, pp. 1725–1744.

[32] Cybenko G., “Approximations by superimpositions of sigmoidal functions”, *Mathematics of Control, Signals and Systems*, vol. 2, n. 4, 1989, pp. 303–314.

[33] D'Andrea E., Lazzerini B., “A hierarchical approach to multi-class fuzzy classifiers”, *Expert Systems with Applications*, 2013, vol. 40, n. 9, 2013, pp. 3828-3840.

[34] D'Andrea E., Lazzerini B., “Computational intelligence techniques for solar photovoltaic system applications”, Proceedings of the 2nd Conference on Sustainable Internet and ICT for Sustainability (SustainIT), Pisa, Italia, 4–5 October 2012c.

[35] D'Andrea E., Lazzerini B., “Fuzzy forecasting of energy production in solar PV systems”, Proceeding of the 2012 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Brisbane, Australia, 10–15 June 2012, pp. 1–8.

[36] D'Andrea E., Lazzerini B., Leon del Rosario S., “Neural network-based forecasting of energy

consumption due to electric lighting in office buildings", Proceedings of the 2nd Conference on Sustainable Internet and ICT for Sustainability (SustainIT), Pisa, Italia, 4–5 October 2012b.

[37] Davis L., **Handbook of Genetic Algorithms**, Van Nostrand Reinhold, 1991.

[38] De Cock M., Cornelis C., Kerre E.E., "Elicitation of fuzzy association rules from positive and negative examples", *Fuzzy Sets and Systems*, vol. 149, n. 1, 2005, pp. 73–85.

[39] del Jesus M.J., Hoffmann F., Navascues L.J. Sanchez L., "Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms", *IEEE Transactions on Fuzzy Systems*, vol. 12, n. 3, 2004, pp. 296–308.

[40] Doukas H., Patlitzianas K.D., Iatropoulos K., Psarras J., "Intelligent building energy management system using rule sets", *Building and Environment*, vol. 42, n. 10, 2007, pp. 3562–3569.

[41] Drossu R., Obradovic Z., "Rapid design of neural networks for time series prediction", *IEEE Computational Science and Engineering*, vol. 3, n. 2, 1996, pp. 78–89.

[42] Duin R.P.W., Juszczak P., de Ridder D., Paclik P., Pekalska E., Tax D.M.J., PRTools: The Matlab Toolbox for Pattern Recognition (http://www.prtools.org/), 2004, version 4.2.3.

[43] EREC (European Renewable Energy Council), "Renewable Energy Technology Roadmap: 20% by 2020", Brussels, 2010.

[44] European Commission, "Energy 2020: A strategy for competitive, sustainable and secure energy", COM(2010), Brussels, 2010.

[45] Fakhrahmad S.M., Zare A., Zolghadri Jahromi M., "Constructing accurate fuzzy rule-based classification systems using apriori principles and rule-weighting", in **Intelligent Data Engineering and Automated Learning**, pp. 547–556, H. Yin *et al.*, Springer Berlin Heidelberg, 2007.

[46] Fausett L., **Fundamentals of neural networks**, Prentice Hall, Englewood Cliffs, N.J., 1994.

[47] Fernández A., del Jesus M.J., Herrera F., "Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets", *International Journal of Approximate Reasoning*, vol. 50, n. 3, 2009, pp. 561–577.

[48] Frank A., Asuncion A., UCI Machine Learning Repository (http://archive.ics.uci.edu/ml). Irvine, CA: University of California, School of Information and Computer Science, 2010.

[49] Gardner M.W., Dorling S.R., "Artificial neural networks (the multilayer perceptron). A review of applications in the atmospheric sciences", *Atmospheric Environment*, vol. 32, n. 14–15, pp. 2627–2636,

[50] Goldberg D.E., **Genetic Algorithms in Search, Optimization and Machine Learning**, Addison-Wesley, 1989.

[51] Gómez-Skarmeta A.F., Valdés M., Jiménez F., Marín-Blázquez J.G., "Approximative fuzzy rules approaches for classification with hybrid-GA techniques", *Information Sciences*, vol. 136, n. 1–4, 2001, pp. 193–214.

[52] Gonzalez A., Perez R., "SLAVE: A genetic learning system based on an iterative approach", *IEEE Transactions on Fuzzy Systems*, vol. 7, n. 2, 1999, pp. 176–191.

[53] Gonzalez-Romera E., Jaramillo-Moran M.A., Carmona-Fernandez D., "Monthly electric energy demand forecasting based on trend extraction", *IEEE Transactions on Power Systems*, vol. 21, n. 4, 2006, pp. 1946–1953.

[54] Guillaume S., Charnomordic B., "Generating an interpretable family of fuzzy partitions from data", *IEEE Transactions on Fuzzy Systems*, vol. 12, n. 3, 2004, pp. 324–335.

[55] Gutiérrez P.A., Salcedo-Sanz S., Hervas-Martinez C., Carro-Calvo L., Sanchez-Monedero J., Prieto L., "Evaluating nominal and ordinal classifiers for wind speed prediction from synoptic pressure patterns", Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA), Córdoba, Spain, 22–24 November 2011, pp. 1265–1270.

[56] Haida T., Muto S., "Regression based peak load forecasting using a transformation technique",

IEEE Transactions on Power Systems, vol. 9, n. 4, 1994, pp. 1788–1794.

[57] Hamilton J.D., **Time series analysis**, Princeton University Press, 1994.

[58] Haykin S.S., **Neural networks. A comprehensive foundation**, Second Edition, Prentice-Hall, New Jersey, 1999.

[59] Hernandez Neto A., Sanzovo Fiorelli F.A., "Comparison between detailed model simulation and artificial neural network for forecasting building energy consumption", *Energy and Buildings*, vol. 40, n. 12, 2008, pp. 2169–2176.

[60] Herrera F. "Genetic Fuzzy systems: Taxonomy, current research trends and prospects", *Evolutionary Intelligence*, vol. 1, 2008, pp. 27–46.

[61] Holland J.H., **Adaptation in Natural and Artificial Systems**, University of Michigan Press, 1975.

[62] Hu Y.-C., Tzeng G.-H., "Elicitation of classification rules by fuzzy data mining", *Engineering Applications of Artificial Intelligence*, vol. 16, n. 7–8, 2003, pp. 709–716.

[63] Huang C., Moraga C., "Extracting fuzzy if-then rules by using the information matrix technique", *Journal of Computer and System Sciences*, vol. 70, n. 1, 2005, pp. 26–52.

[64] Ishibuchi H., Kuwajima I., Nojima Y., "Effectiveness of designing fuzzy rule-based classifiers from Pareto-optimal rules", 2008 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Hong Kong, China, 1–6 June 2008, pp.1185–1192.

[65] Ishibuchi H., Murata T., Turksen I.B., "Single-objective and two objective genetic algorithms for selecting linguistic rules for pattern classification problems", *Fuzzy Sets and Systems*, vol. 89, n. 2, 1997, pp. 135–150.

[66] Ishibuchi H., Nakashima T., "Effect of rule weights in fuzzy rule-based classification systems", *IEEE Transactions on Fuzzy Systems*, vol. 9, n. 4, 2001, pp. 506–515.

[67] Ishibuchi H., Nakashima T., Murata T., "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems", *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 29, n. 5, 1999, pp. 601–618.

[68] Ishibuchi H., Nakashima T., Murata T., "Three-objective genetics-based machine learning for linguistic rule extraction", *Information Sciences*, vol. 136, n. 1–4, 2001, pp. 109–133.

[69] Ishibuchi H., Nojima Y., Kuwajima I., "Evolutionary multiobjective design of fuzzy rule-based classifiers", in **Computational Intelligence: a compendium,** pp. 641–685. J. Fulcher *et al.*, Springer Berlin Heidelberg, 2008.

[70] Ishibuchi H., Nozaki K., Tanaka H., "Distributed representation of fuzzy rules and its application to pattern classification", *Fuzzy Sets and Systems*, vol. 52, 1992, pp. 21–32.

[71] Ishibuchi H., Yamamoto T., "Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining", *Fuzzy Sets and Systems*, vol. 141, 2004, pp. 59–88.

[72] Ishibuchi H., Yamamoto T., "Rule weight specification in fuzzy rule-based classification systems", *IEEE Transactions on Fuzzy Systems*, vol. 13, n. 4, 2005, pp. 428–435.

[73] Ishibuchi H., Yamamoto T., Nakashima T., "Hybridization of fuzzy GBML approaches for pattern classification problems", *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 35, n. 2, 2005, pp. 359–365.

[74] Ivancheva J., Koleva E., "An estimation of the global solar radiation over Bulgaria", Proceedings of International Conference on Water Observation and Information System for Decision Support (BALWOIS), Ohris, Republic of Macedonia, 27–31 May 2008.

[75] Janikow C.Z., Michalewicz Z., "An experimental comparison of binary and floating point representations in genetic algorithms", Proceedings of the 4th International Conference on Genetic Algorithms, San Diego, CA, July1991, pp. 31-36.

[76] Jin Y., "Fuzzy modeling of high-dimensional systems. Complexity reduction and Iinterpretability improvement", *IEEE Transactions on Fuzzy Systems*, vol. 1, 2000, pp. 212–

221.

[77] Kalogirou S.A., “Applications of artificial neural-networks for energy systems”, *Applied Energy*, vol. 67, n. 1–2, 2000, pp. 17–35.

[78] Karatasou S., Santamouris M., Geros V., “Modeling and predicting building’s energy use with artificial neural networks: methods and results”, *Energy and Buildings*, vol. 38, 2006, pp. 949–958.

[79] Kasabov N., “Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems”, *Fuzzy Sets and Systems*, vol. 82, n. 2, 1996, pp. 135–149.

[80] Kottas T.L., Boutalis Y.S., Karlis A.D., “New maximum power point tracker for PV arrays using fuzzy controller in close cooperation fuzzy cognitive networks”, *IEEE Transactions on Energy Conversion*, vol. 21, n. 3, 2006, pp. 793–803.

[81] Kudo M., Takeuchi A., Nozaki Y., Endo H., Sumita J., “Forecasting electric power generation in a photovoltaic power system for an energy network”, *Electrical Engineering in Japan*, vol. 167, n. 4, 2009.

[82] Kuncheva L.I., “On the equivalence between fuzzy and statistical classifiers”, *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, vol. 4, n. 3, 1996, pp. 245–253.

[83] Lam J.C., Chan R.Y.C., Tsang C.L., Li D.H.W., “Electricity use characteristics of purpose-built office buildings in subtropical climates”, *Energy Conversion and Management*, vol. 45, n. 6, 2004, pp. 829–844.

[84] Lam J.C., Li D.H.W., “Luminous Efficacy of daylight under different sky conditions”, *Energy Conversion and Management*, vol. 37, n. 12, 1996, pp. 1703–1711.

[85] Lee H.-M., Chen C.-M., Chen J.-M., Jou Y.-L., “An efficient fuzzy classifier with feature selection based on fuzzy entropy”, *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 31, n. 3, 2001, pp. 426–432.

[86] Lei M., Shiyan L., Chuanwen J., Hongling L., Yan Z., “A review on the forecasting of wind speed and generated power”, *Renewable and Sustainable Energy Reviews*, vol. 13, n. 4, 2009, pp. 915–920.

[87] Leontaritis I.J., Billings S.A., “Input-output parametric models for non-linear systems. Part I: deterministic non-linear systems”, *International Journal of Control*, vol. 41, n. 2, 1985, pp. 303–328.

[88] Leontaritis I.J., Billings S.A., “Input-output parametric models for non-linear systems. Part II: stochastic non-linear systems”, *International Journal of Control*, vol. 41, n. 2, 1985, pp. 329–344.

[89] Li D.H.W., Lam J.C., “An analysis of climatic parameters and sky condition classification”, *Building and Environment*, vol. 36, n. 4, 2001, pp. 435–445.

[90] Li M., Wang Z., “A hybrid coevolutionary algorithm for designing fuzzy classifiers”, *Information Sciences*, vol. 179, n. 12, 2009, pp. 1970–1983.

[91] MacQueen J.B., “Some Methods for classification and Analysis of Multivariate Observations”, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, 1967, vol. 1, pp. 281–297.

[92] Makridakis S.G., Wheelwright S.C., Hyndman R.J., **Forecasting: methods and applications**, Third Edition, Wiley, 1998.

[93] Mansoori E.G., Zolghadri Jahromi M., Katebi S.D., “A weighting function for improving fuzzy classification systems performance”, *Fuzzy Sets and Systems,* vol. 158, n. 5, 2007, pp. 583–591.

[94] Marques Nogueira T., de Arruda Camargo H., “Fuzzy rule base generation through conditional clustering”, 19th Brazilian Symposium on Artificial Intelligence (SBIA), Salvador, Brazil, 26-30 October 2008.

[95] Marquez R., Coimbra C.F.M., “Forecasting of global and direct solar irradiance using stochastic learning methods, ground experiments and the NWS database”, *Solar Energy*, vol. 85, n. 5,

2011, pp. 746–756.

[96] Martin L., Zarzalejo L.F., Polo J., Navarro A., Marchante R., Cony M., “Prediction of global solar irradiance based on time series analysis: Application to solar thermal power plants energy production planning”, *Solar Energy*, vol. 84, n. 10, 2010, pp. 1772–1781.

[97] Mateo F., Carrasco J.J., Sellami A., Millán-Giraldo M., Domínguez M., Soria-Olivas E., “Machine learning methods to forecast temperature in buildings”, *Expert Systems with Applications*, vol. 40, n. 4, 2013, pp. 1061–106.

[98] McCulloch W.S., Pitts W., “A logical calculus of ideas immanent in nervous activity”, *The bulletin of mathematical biophysics*, vol. 5, n. 4, 1943, pp. 115–133.

[99] Mellit A., Benghanem M., Kalogirou S.A., “An adaptive wavelet-network model for forecasting daily total solar-radiation”, *Applied Energy*, vol. 83, n. 7, 2006, pp. 705–722.

[100] Mellit A., Massi Pavan A., “A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected PV plant at Trieste, Italy”, *Solar Energy*, vol. 84, n. 5, 2010, pp. 807–821.

[101] Mitchell M., **An introduction to genetic algorthms**, MIT press, 1998.

[102] Mitra S., Hayashi Y., “Neuro-fuzzy rule generation: survey in soft computing framework”, *IEEE Transactions on Neural Networks*, vol. 11, n. 3, 2000, pp. 748–768.

[103] Mitra S., Kuncheva L.I., “Improving classification performance using fuzzy MLP and two-level selective partitioning of the feature space”, *Fuzzy Sets and Systems*, vol. 70, 1995, pp. 1–13.

[104] Moghram I., Rahman S., “Analysis and evaluation of five short-term load forecasting techniques”, *IEEE Transactions on Power Systems*, vol. 4, n. 4, 1989, pp. 1484–1491.

[105] Mohamed E.A., Mansour M.M., El-Debeiky S., Mohamed K.G., “Egyptian unified grid hourly load forecasting using artificial neural network”, *International Journal of Electrical Power & Energy Systems*, vol. 20, n. 7, 1998, pp. 495–500.

[106] Monfared M., Rastegar H., Kojabadi H.M., “A new strategy for wind speed forecasting using artificial intelligent methods”, *Renewable Energy*, vol. 34, 2009, pp. 845–848.

[107] Mubiru J., Banda E.J.K.B., “Estimation of monthly average daily global solar irradiation using artificial neural networks”, *Solar Energy*, vol. 82, n. 2, 2008, pp. 181–187.

[108] Nadimi V., Azadeh A., Pazhoheshfar P., Saberi M., “An adaptive-network-based fuzzy Inference system for long-term electric consumption forecasting (2008–2015): A case study of the Group of Seven (G7) industrialized nations: U.S.A., Canada, Germany, United Kingdom, Japan, France and Italy”, Proceedings of the 2010 Fourth UKSim European Symposium on Computer Modeling and Simulation (EMS), Pisa, Italy, 17–19 November 2010, pp. 301–305.

[109] Nauck D., “GNU fuzzy”, Proceedings of 2007 International Conference on Fuzzy Systems (FUZZ-IEEE), London, UK, 23–26 July 2007, pp. 1–6.

[110] Nauck D., Kruse R. “NEFCLASS-J – A JAVA-based soft computing tool”, in **Intelligent Systems and Soft Computing**, pp. 139–160, B. Azvine *et al.*, Springer Berlin Heidelberg, 2000.

[111] Nauck D., Kruse R., “A neuro-fuzzy method to learn fuzzy classification rules from data”, *Fuzzy Sets and Systems*, vol. 89, n. 3, 1997, pp. 77–288.

[112] Nauck D., Kruse R., “NEFCLASS - A neuro-fuzzy approach for classification of data”, Proceedings of 1995 ACM Symposium on Applied Computing, Nashville, TN, 1995, pp. 461–465.

[113] Nicol J., Humpreys M., “Adaptive thermal comfort and sustainable thermal standards for buildings”, *Energy and Buildings*, vol. 34, n. 6, 2002, pp. 563–572.

[114] Nozaki K., Ishibuchi H., Tanaka H., “A simple but powerful heuristic method for generating fuzzy rules from numerical data”, *Fuzzy Sets and Systems*, vol. 86, n. 3, 1997, pp. 251–270.

[115] Nozaki K., Ishibuchi H., Tanaka H., “Adaptive fuzzy rule-based classification systems”, *IEEE Transactions on Fuzzy Systems*, vol. 4, n. 3, 1996, pp. 238–250.

[116] Oliver M., Jackson T., "The market for solar photovoltaics", *Energy Policy*, vol. 27, 1999, pp. 371–385.

[117] Padmakumari K., Mohandas K.P., Thiruvengadam S., "Long term distribution demand forecasting using neuro fuzzy computations", *International Journal of Electrical Power & Energy Systems*, vol. 21, n. 5, 1999, pp. 315–322.

[118] Panda S.S., Chakraborty D., Pal S.K., "Drill wear prediction using neural network architectures", *International Journal of Knowledge-based and Intelligent System,* vol. 12, n. 5–6, 2008, pp. 327–338.

[119] Paoli C., Voyant C., Muselli M., Nivet M., "Solar radiation forecasting using ad-hoc time series preprocessing and neural networks", *Solar Energy*, vol. 84, n. 12, 2010, pp. 2146–2160.

[120] Papalexopoulos A.D., Hesterberg T.C., "A regression-based approach to short-term system load forecasting", *IEEE Transactions on Power Systems*, vol. 5, n. 4, 1990, pp. 1535–1547.

[121] Patterson D.W., **Artificial neural networks. Theory and applications**, Prentice Hall, Simon & Schuster, Singapore, 1996.

[122] Potter C.W., Archambault A., Westrick K., "Building a smarter smart grid through better renewable energy information", Proceedings of 2009 Power Systems Conference and Exposition (PSCE), IEEE/PES, Seattle, WA, 15–18 March 2009, pp. 1–5.

[123] Pousinho H.M.I., Mendes V.M.F., Catalão J.P.S., "Neuro-fuzzy approach to forecast wind power in Portugal", Proceedings of 2010 International Conference on Renewables Energies and Power Quality (ICREPQ), Granada, Spain, 23–25 March 2010, pp. 1–4.

[124] Rahman S., Hazim O., "A generalized knowledge-based short-term load-forecasting technique", *IEEE Transactions on Power Systems*, vol. 8, n. 2, 1993, pp. 508–514.

[125] Ramakumar R., Bigger J.E., Photovoltaic systems, *IEEE Proceedings*, vol. 81, n. 3, 1993, pp. 365–77.

[126] REN21, "Renewables 2012 Global Status Report", Paris: REN21 Secretariat, France, 2012.

[127] Rojas R., **Neural networks. A systematic introduction**, Springer, Berlin, 1995.

[128] Rosenblatt F., **Principles of neurodynamics: perceptrons and the theory of brain mechanisms**, Spartan Books, Washington D.C., 1962.

[129] Roubos H., Setnes M., "Compact and transparent fuzzy models and classifiers through iterative complexity reduction", *IEEE Transactions on Fuzzy Systems*, vol. 9, n. 4, 2001, pp. 516–524.

[130] Roubos J.-A., Setnes M., Abonyi J., "Learning fuzzy classification rules from labeled data", *Information Sciences*, vol. 150, 2003 pp. 77–93.

[131] Rumelhart D.E., Hinton G.E., Williams R.J., "Learning representations by back-propagating errors", *Nature,* vol. 323, 1986, pp. 533–536.

[132] Saab S., Badr E., Nasr G., "Univariate modeling and forecasting of energy consumption: the case of electricity in Lebanon", *Energy*, vol. 26, n. 1, 2001, pp. 1–14.

[133] Sánchez L., Couso I., Corrales J.A., Cordón O., del Jesus M.J., Herrera F., "Combining GP operators with SA search to evolve fuzzy rule based classifiers", *Information Sciences*, vol. 136, n. 1–4, 2001, pp. 175–191.

[134] Sanz J., Fernández A., Bustince H., Herrera F. "A genetic tuning to improve the performance of fuzzy rule-based classification systems with interval-valued fuzzy sets: degree of ignorance and lateral position", *International Journal of Approximate Reasoning*, vol. 52, 2011, pp. 751–766.

[135] Senjyu T., Takara H., Uezato K., Funabashi T., "One-hour-ahead load forecasting using neural network", *IEEE Transactions on Power Systems*, vol. 17, n. 1, 2002, pp. 113–118.

[136] Setnes M., Babuska R., Verbruggen B., "Rule-based modeling: precision and transparency", *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, vol. 28, n. 1, 1998, pp. 165–169.

[137] Setnes M., Roubos H. "GA-fuzzy modeling and classification: complexity and performance", *IEEE Transactions on Fuzzy Systems*, vol. 8, n. 5, 2000, pp. 509–522.

[138] Sfetsos A., “A novel approach for the forecasting of mean hourly wind speed time series”, *Renewable Energy*, vol. 27, n. 2, 2002, pp. 163–174.

[139] Song Y.D., “A new approach for wind speed prediction”, *Wind Engineering*, vol. 24, n. 1, 2000, pp. 35–47.

[140] Srinivas M., Patnaik L.M., “Genetic algorithms: a survey”, *Computer*, vol. 27, n. 6, 1994, pp. 17–26.

[141] Srinivasan D., Swee T.S., Cheng C.S. Chan E.K., “Parallel neural network-fuzzy expert system strategy for short-term load forecasting: system implementation and performance evaluation”, *IEEE Transactions on Power Systems*, vol. 14, n. 3, 1999, pp. 1100–1106.

[142] Sukamongkol Y., Chungpaibulpatana S., Ongsakul, W., “A simulation model for predicting the performance of a solar photovoltaic system with alternating current loads”, *Renewable Energy*, vol. 27, 2002, pp. 237–258.

[143] Sulaiman S.I., Rahman T.K.A., Musirin I., Shaari S., “Performance analysis of two-variate ANN models for predicting the output power from grid-connected photovoltaic system”, *International Journal of Power, Energy and Artificial Intelligence* (IJPEAI), vol. 2, n. 1, 2009, pp. 72–76.

[144] Taylor J.W., Buizza R., “Neural network load forecasting with weather ensemble predictions”, *IEEE Transactions on Power Systems*, vol. 17, n. 3, 2002, pp. 626–632.

[145] Troncoso Lora A., Riquelme J.C., Martìnez Ramos J.L., Riquelme Santos J.M., Gomez Exposito A., “Influence of kNN-based load forecasting errors on optimal energy production”, in **Progress in Artificial Intelligence**, pp. 189–203, F.M. Pires *et al.*, Springer Berlin Heidelberg, 2003.

[146] Tsang E.-C.-C., Wang X.-Z., Yeung D.-S., “Improving learning accuracy of fuzzy decision trees by hybrid neural networks”, *IEEE Transactions on Fuzzy Systems*, vol. 8, n. 5, 2000, pp. 601–614.

[147] Tsoutsosa T., Frantzeskakib N., Gekas V., “Environmental impacts from the solar energy technologies”, *Energy Policy*, vol. 33, 2005, pp. 289–296.

[148] Tung S.W., Queck C., Guan C., “SoHyFIS-Yager: A self-organizing Yager based hybrid neural fuzzy inference system”, *Expert Systems with Applications*, vol. 39, 2012, pp. 12759–12771.

[149] Tzafestas S., Tzafestas E., “Computational Intelligence Techniques for Short-Term Electric Load Forecasting”, *Journal of Intelligent & Robotic Systems*, vol. 31 n. 1, 2001, pp. 7–68.

[150] Uebele V., Abe S., Lan M.-S., “A neural network based fuzzy classifier”, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, n. 2, 1995, pp. 353–361.

[151] van der Heijden F., Duin R.P.W., De Ridder D., Tax D.M.J., **Classification, Parameter Estimation and State Estimation: An Engineering Approach Using MATLAB**, John Wiley & Sons, 2004.

[152] Vieira S., Sousa J.M.C., Runkler T.A., “Multi-criteria ant feature selection using fuzzy classifiers”, in **Swarm Intelligence for Multi-objective Problems in Data Mining**, pp. 19–36, C.A. Coello Coello *et al.*, Springer Berlin Heidelberg, 2009.

[153] Wang J.-S., Lee C.S.G., “Self-adaptive neuro-fuzzy inference systems for classification applications”, *IEEE Transactions on Fuzzy Systems*, vol. 10, n. 6, 2002, pp. 790–802.

[154] Wang L.-X., Mendel J.M., “Generating fuzzy rules by learning from examples”, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, n. 6, 1992, pp. 1414–1427.

[155] Wang X.-Z., Tsang E.C.C., Yeung D.-S., “A comparative study on heuristic algorithms for generating fuzzy decision trees”, *IEEE Transactions on Systems, Man and Cybernetics– Part B: Cybernetics*, vol. 31, n. 2, 2001, pp. 215–226.

[156] Wang X.-Z., Wang Y.-D., Xu X.-F., Ling W.-D., Yeung D.-S., “A new approach to fuzzy rule generation: Fuzzy extension matrix”, *Fuzzy Sets and Systems*, vol. 123, n. 3, 2001, pp. 291–306.

[157] Wilamowski B.M., Li X., “Fuzzy system based maximum power point tracking for PV system”, 28th IEEE Annual Conference of the Industrial Electronics Society (IECON), Sevilla, Spain, 5–

8 November 2002, vol. 4, pp. 3280–3284.

[158] Wong S.L., Wan K.K.W., Lam T.N.T., "Artificial neural networks for energy analysis of office buildings with daylighting", *Applied Energy*, vol. 87, n. 2, 2010, pp. 551–557.

[159] Wu C.-F., Lin C.-J., Lee C.-Y., "A functional neural fuzzy network for classification applications", *Expert Systems with Applications*, vol. 28, 2011, pp. 6202–6208.

[160] Xie H., Liu L., Ma. F., Fan H., "Performance prediction of solar collectors using artificial neural networks", Proceedings of 2009 International Conference on Artificial Intelligence and Computational Intelligence (AICI), Shanghai, China, 2009, pp. 573–576.

[161] Xiong N., Litz L., Ressom H., "Learning premises of fuzzy rules for knowledge acquisition in classification problems, knowledge and information systems", vol. 4, n. 1, 2002, pp. 96–111.

[162] Yona A., Senjyu T., Saber A.Y., Funabashi T., Sekine H., Chul-Hwan K., "Application of neural network to 24-hour-ahead generating power forecasting for PV system", Proceedings of Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, Pittsburgh, Pennsylvania, 20–24 July 2008, pp. 1–6.

[163] Yuan Y., Shaw M. "Induction of fuzzy decision trees", *Fuzzy Sets and Systems*, vol. 69. n. 2, 1995, pp. 125–139.

[164] Zadeh L.A., "Fuzzy sets", *Information and Control*, vol. 8, n. 3, 1965, pp. 338–353.

[165] Zhou E., Khotanzad A., "Fuzzy classifier design using genetic algorithms", *Pattern Recognition*, vol. 40, 2007, pp. 3401–3414.

[166] Zolghadri Jahromi M., Mansoori E.G., "Weighting fuzzy classification rules using receiver operating characteristics (ROC) analysis", *Information Sciences*, vol. 177, n. 11, 2007, pp. 2296–2307.

[167] Zolghadri Jahromi M., Taheri M. "A proposed method for learning rule weights in fuzzy rule-based classification systems", *Fuzzy Sets and Systems*, vol. 159, 2008, pp. 449–459.

A

How to implement a generic classifier in PRTools

First we introduce the PRTools framework[1], then, the basic elements of PRTools, namely the `dataset` and `mapping` objects, in the sense of object oriented programming (OOP), and then we recall the phases for building a new generic classifier. Lastly we show the main steps of the implementation in PRTools of the classfier `frbc` presented in Chapter 2.

A.1 PRTools framework

The Pattern Recognition Toolbox (PRTools) is the *de-facto* standard toolbox for classification in Matlab, and is freely available for academic research. It offers many classifiers, like linear and quadratic discriminant classifiers, decision trees and neural networks. In addition, many base functions are available for training and testing classifiers, and plotting of decision boundaries. Furthermore, many types of feature selection (e.g., individual, forward, backward, branch-and-bound) could be performed. In the following, we outline main characteristic of PRTools.

PRTools follows a base philosophy that is:

a) *powerful and concise syntax-oriented*: PRTools exploits the Matlab support to object-oriented programming and overloads many operators (plus, times, disp, etc.). In this way, the syntax associated with the training of a classifier and its test is very concise. Operator concatenation also allows for an even more concise syntax: many basic tasks can be performed by including operators and objects into a single expression;
b) *command line-oriented*: PRTools is not equipped with a graphical user interface: each task can be performed by entering command line instructions at the Matlab prompt or by using scripts/functions;
c) *multiple test set-oriented*: PRTools decouples the training phase from the

[1] For a detailed documentation see: i) van der Heijden F., Duin R.P.W., De Ridder D., Tax D.M.J., **Classification, Parameter Estimation and State Estimation: an Engineering Approach Using MATLAB**, John Wiley & Sons, 2004, and ii) Duin R.P.W., Juszczak P., de Ridder D., Paclik P., Pekalska E., Tax D.M.J., PRTools: The Matlab Toolbox for Pattern Recognition (http://www.prtools.org/), 2004, version 4.2.3.

test phase. To this aim, it maintains the data structure representing the input-output mapping learnt during training, so that this mapping can subsequently be exploited to test the classifier on as many test sets as needed;

d) *oriented to the reuse of existing Matlab toolboxes*: PRTools does not intend to reinvent the wheel: if something is already available in Matlab then it reuses it. For instance, neural network-based classifiers exploit the Matlab Neural Networks Toolbox;

e) *oriented to as few free parameters as possible*: classification algorithms may have a lot of free parameters to be set by the user. PRTools tends to limit the number of these parameters as much as possible, by privileging algorithms that need as few parameters as possible, and/or default (sometimes optimized) values for most of them;

f) *oriented to fast and heuristic-based training algorithms*: the goal of PRTools is not to provide the most accurate results, achieved by means of very complex optimization methods, such as simulated annealing, evolutionary algorithms, swarm intelligence, etc. Instead, it frequently uses good heuristics to determine good classifiers quite quickly. For instance, `treec`, which is the PRTools implementation of decision trees, does not involve global optimization meta-heuristics, but instead it uses well-assessed and efficient heuristics;

g) *data-driven oriented*: PRTools is oriented to the automatic training of classifiers from data, without requiring any intervention by a human expert in the specific application domain;

h) *oriented to well-assessed and widely accepted methodologies*: PRTools privileges well-assessed and widely accepted design methodologies, instead of more recent and cutting the edge ones. This choice allows for algorithms that perform generally well (although not superlatively well) in most of the application domains, in place of algorithms that perform well only in particular circumstances, and poorly in many other cases;

i) *oriented to the combination with other functions*: PRTools easily allows passing a classification algorithm as parameter of other higher-level functions. In this way, e.g., a feature selection function is able to call any classifier and to rank features based on the performance achieved by that classifier, once trained using the subset of features under assessment;

j) *batch-mode oriented*: PRTools is not oriented to online classification of data streams, but instead it assumes that all the data are available at the beginning.

A.2 Basic elements of PRTools

A.2.1 The `dataset` object

The `dataset` object is one of the two fundamental elements in PRTools. A `dataset` consists of a structure which contains both the patterns and the corresponding class labels. Further, it contains other useful information regarding the input domain, the class a-priori probabilities, etc.

PRTools also defines several member functions for the `dataset` class, which are used to set and get field values, display selected information, and so on. It also overloads some Matlab® operators and functions, thus making the syntax very concise and powerful by exploiting operator concatenation, priority, etc.

A.2.2 The `mapping` object

Datasets are transformed by mappings, so a `mapping` is an object that may store the information about transformations to be made on data (e.g., in case of a neural network, the network architecture and its parameters), the routine to be used to perform the mapping itself, and the routine for training. It is used to transform data (e.g., normalization and scaling), to build classifiers, etc.

PRTools supports the following four types of `mapping`:

a) *untrained mappings*: they are empty mappings which store the specific mapping routine that has to be used when they are trained on a specific dataset. They can also store user-defined options (e.g., the value of k for a k-nearest neighbor classifier);
b) *trained mappings*: they store all the information gathered during the training process of a classifier, so as to classify (map) as many datasets as needed in the future;
c) *fixed mappings*: they transform dataset objects, but in this case the type of transformation is defined by the user (scaling, normalization, etc.);
d) *combiners*: they are mappings used to combine multiple mappings (associated to an ensemble of classifiers) into a combined mapping.

A classifier is handled as follows: the function that has to be used to train the classifier is stored in a `mapping` object, as well as the function used to map a data set. When an object of type `dataset` is "multiplied by" a `mapping` object associated with a particular classifier, the relative training function is invoked on the provided dataset.

As an example let us look at the following lines of code:

```
U=ldc          % builds an untrained mapping U with ldc
               (Linear Bayes Classifier)
W=tr*U         % perform training with dataset tr and
               builds a trained mapping W
V=ldc(tr)      % builds a trained mapping V directly
e=testc(ts,W)  % computes the misclassification error of W
               on a test dataset ts
```

In PRTools each classifier has a constructor, which, by calling the mapping constructor, builds a `mapping` object after performing input check and other minor operations. In the following we describe how the constructor for a generic classifier x has to be implemented. It is good practice to name such constructor xc.

A.3 The construction of a generic trained classifier xc

Implementing a new classifier under PRTools requires the implementation of three functions in an appropriate way. In the following we provide a template for implementing a generic classifier in PRTools, which may be useful for other researchers interested in extending PRTools.

After analyzing some classifiers in PRTools, we can point out that the new classifier xc should be able to perform the following operations:

1. the construction of a mapping object: the classifier;
2. the training of the classifier;
3. the application of a dataset to the trained classifier (i.e., the classification);
4. the building of a dataset as the result of the classification consistently with PRTools routines like testc, etc.

So the source code of a new classifier xc will consist of three main parts: the classifier constructor, the training phase and the mapping phase (the classification).

A.3.1 The classifier constructor xc

This is the main routine of the classifier. The classifier xc should be able to handle the following types of call:

```
U=xc([],params)    % builds an untrained classifier U with
                   classifier xc and sets some training
                   parameters params
V=tr*U             % trains the untrained classifier U with
                   dataset tr
W=xc(tr,params)    % builds directly a trained classifier W
                   with xc, specifying some training
                   parameters, on dataset tr
D=ts*W             % uses W to perform the classification of
                   a test dataset ts
```

A.3.2 The training phase xc_train

This phase performs the training of the classifier. The processing could be part of the main file xc or could be put apart. It requires as input a training dataset tr and, optionally, the parameters needed for training params, and provides a trained mapping as output. The routine xc_train is called when a trained classifier W is directly build (W=xc(tr,params)) or to train an untrained classifier U previously built (V=tr*U).

A.3.3 The mapping phase xc_map

For each classifier in PRTools, a mapping routine xc_map is needed. When a trained mapping is applied to a dataset (D=ts*W) the routine xc_map is called. The dataset D contains the result of the classification. So the previous instruction is equivalent both to D=map(ts,W) and to D=xc_map(ts,W). Otherwise, if the mapping routine is applied to an untrained classifier U, (W=tr*U), the training of the classifier is performed and the routine returns a trained mapping. The previous instruction is equivalent to W=map(tr,U).

A.4 The implementation of **frbc** *in PRTools*

To implement frbc in PRTools we have to provide the following functions: the constructor (frbc), the training function (frbc_train) and the mapping function (frbc_map). Once provided these functions, PRTools will automatically support, for example, the following statements:

```
W1=frbc(tr)                 % trains frbc W1 directly with
                              default options
W2=frbc(tr,opts)            % trains frbc W2 directly using
                              options opts
U=frbc([],opts)             % builds an untrained mapping U
                              with frbc
V=tr*U                      % trains the untrained mapping U
                              on dataset tr
[W,R]=faetself(tr,frbc,f,ts) % apply forward feature selection
                            using frbc (find the feature that
                            used alone gives the best result,
                            then selects the second feature
                            that with the first one gives the
                            best result, and so on.
plotc(V)                    % plot the decision boundary for
                              V
e=testc(ts,V)               % computes the misclassification
                              error of V on test dataset ts
```

A.4.1 **frbc** (the constructor)

This is the main routine of the classifier. We will analyze the structure of the source code through a flowchart (Fig. A.1), which summarizes the fundamental parts of the routine.

This function builds a PRTools mapping object, which represents a classifier. What is interesting here is that, within the mapping object, the data related to the FRBC are stored in a Matlab FLT consistent FIS structure. This structure can be retrieved whenever necessary using the function frbc_getfis, and used in combination with other FLT functions (e.g., evalfis, mfedit, ruleview, showrule, etc.).

frbc supports many options. They are provided through the input parameter opts which is a structure containing the parameters for the training and the fuzzy reasoning. The options fields are: i) the number of fuzzy sets per variable, ii) the membership function type, iii) the certainty factor computing method. Moreover, as regards the fuzzy reasoning method, the user can choose different operators among the supported ones: i) the AND operator, ii) the *association degrees* operator, iii) the aggregation function, iv) the *stress* method, etc. Default options are handled by a specific routine, which also allows the customization of all the above mentioned options. The same function provides some shortcuts (e.g., classical FRM) to useful combinations of options. Moreover, the user can change the options fields of a predefined shortcut directly from the command line.

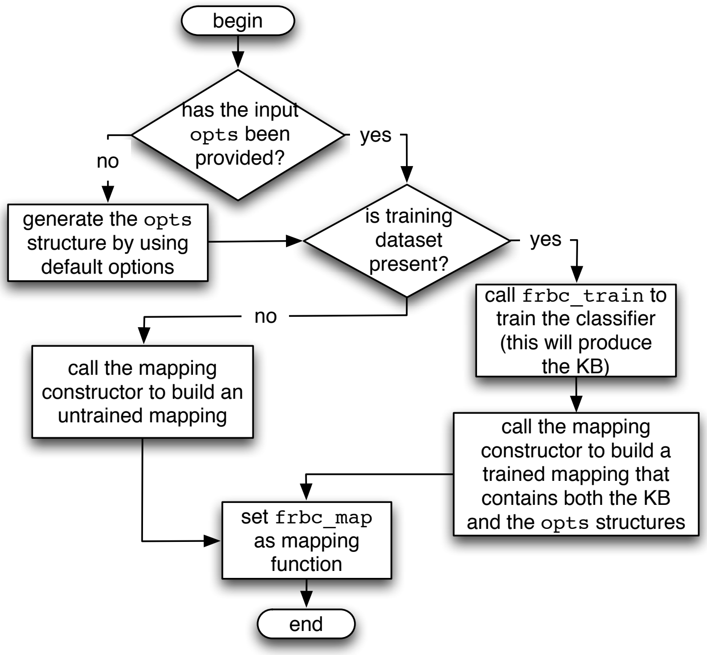

begin
has the input opts been provided?
no
yes
generate the opts structure by using default options
is training dataset present?
yes
call frbc_train to train the classifier (this will produce the KB)
no
call the mapping constructor to build an untrained mapping
call the mapping constructor to build a trained mapping that contains both the KB and the opts structures
set frbc_map as mapping function
end

Figure A.1. – *Flowchart describing the actions of the* `frbc` *constructor.*

A.4.2 `frbc_train` (the training function)

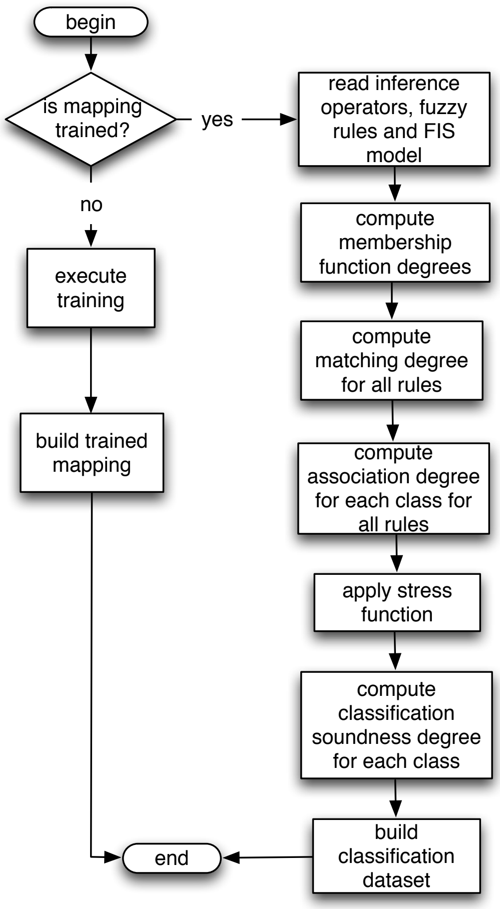
This routine trains the classifier according to the Wang and Mendel approach[2] for generating fuzzy rules from data, presented in Section 2.3.1. This function, first, generates an initial raw rule base, then generates the final rule base, by performing the following steps: a) removes ambiguous rules, b) removes duplicated rules, c) computes certainty factors, d) removes conflicting rules, i.e., rules with same antecedent but different consequent class (only the rule with the highest certainty

[2] Wang L.-X., Mendel J.M., “Generating fuzzy rules by learning from examples”, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, n. 6, 1992, pp. 1414–1427.

factor is kept).

A.4.3 `frbc_map` (the mapping function)

This is the mapping routine (the flowchart shown in Fig. A.2), which contains the processing required to apply the FRM, as described in Section 2.3.2. It corresponds to the `evalfis` function under the Matlab FLT.



begin
is mapping trained?
yes
no
read inference operators, fuzzy rules and FIS model
compute membership function degrees
compute matching degree for all rules
compute association degree for each class for all rules
apply stress function
compute classification soundness degree for each class
build classification dataset
execute training
build trained mapping
end

Figure A.2. – *Flowchart describing the actions of the mapping function* `frbc_map`.

A.4.4 `frbc`: some usage example

The tests carried out with `frbc` have reproduced the results concerning fuzzy rule-based classifiers in the literature[3].

In addition, we report some interesting examples on how to use `frbc`, combined with existing PRTools functions.

We build and train `frbc` using an artificial dataset of two-class elements with a banana shaped distribution. Then we plot the decision boundaries of two well-known PRTools classifiers (the Linear Bayes Classifier `ldc` and the K-Nearest Neighbor Classifier `knnc`) against `frbc` to compare them (see Fig. A.3). We can easily see from Fig. A.3 the good performance achieved by the `frbc` classifier.

The following lines of code have been used to perform this test:

```
tr=gendatb(500)      % generates a banana-shaped distribution
                       training dataset of 500 elements
W=frbc(tr,opts)      % trains frbc W directly using options opts
                       on dataset tr
V1=ldc(tr)           % trains the Linear Bayes Classifier V1
                       directly on tr
V2=knnc(tr)          % trains the K-Nearest Neighbor Classifier
                      V2 directly on tr
scatterd(tr)         % plots the training dataset
plotc(W,V1,V2)       % plot the decision boundaries for W, V1 and
                       V2
```

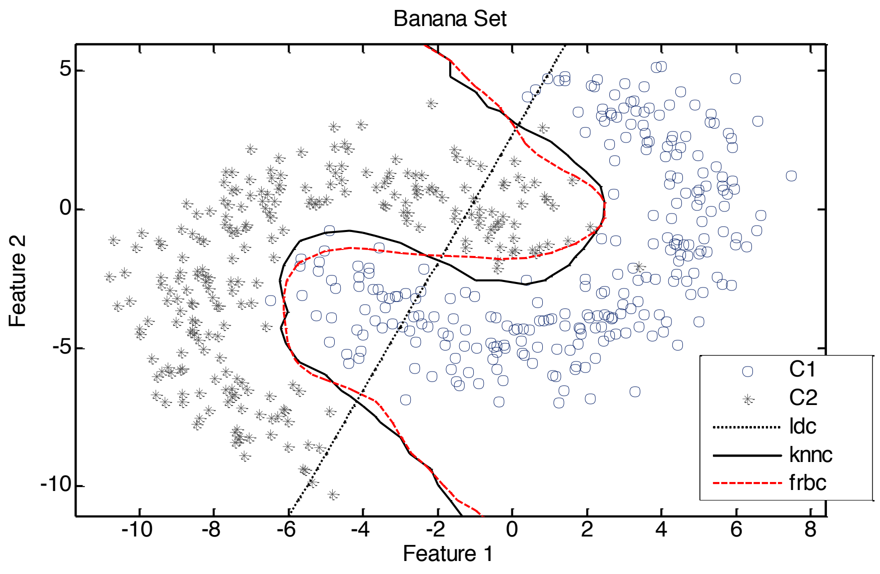


Figure A.3. – *Decision boundaries for* `knnc`, `ldc` *and* `frbc` *(dashed red line) on a two-class dataset.*

[3] Cordón O., del Jesus M.J., Herrera F., "A proposal on reasoning methods in fuzzy rule-based classification systems", *Int. Journal of Approximate Reasoning*, vol. 20, n. 1, 1999, pp. 21–45.