SAPIENZA
UNIVERSITÀ DI ROMA

Neural and Fuzzy Neural Approaches to Energy Time Series Prediction

Department of Information Engineering,
Electronics and Telecommunications

Ph.D. in
Information and Communications Technologies
XXXI Cycle

Advisor
Prof. Massimo Panella

Candidate
Antonello Rosato

This thesis has been evaluated by the two following external referees:

- Prof. Barbara Cannas, University of Cagliari, Electric and Electronic Engineering Dept.
- Prof. Amedeo Andreotti, Department of Electrical Engineering and Information Technology - University of Naples "Federico II".

Neural and Fuzzy Neural Approaches to Energy Time Series Prediction
Ph.D. thesis. Sapienza - University of Rome

Version: February 11, 2019
Author's email: antonello.rosato@uniroma1.it

Abstract

The analysis of time series has a significant value in a lot of various and diverse broad contexts, such as engineering, medicine, finance, physics and social sciences. Its importance stems from the possibility to grasp a better knowledge of the underlying processes involved in the field of interest. One of the core objectives of time series analysis is prediction. In fact, the forecasting of future values of any time series is very valuable, especially in applications were adapting to later evolutions of a given system is profitable and often mandatory. Historically, time series prediction has been implemented mainly with statistical models, but with the advent of new machine learning paradigms such as neural networks, which mimic the human intuitive learning approach, the horizon of possibilities for time series methods has been largely widened. Also, pairing neural networks with the ability of fuzzy logic to deal with the uncertainty of data, can give precious results in terms of accuracy for time series forecasting algorithms. The main objective of this work is to investigate these new machine learning approaches to solve the prediction problem in the energy production context, which has a big challenging application impact. The ability to forecast energy related time series, such as energy commodities prices, electrical load of power systems and energy production, in the short and middle term is a key issue to allow a high-level penetration of the distributed generation into the grid infrastructure. Forecasting energy production is mandatory for dispatching and distribution issues at the transmission system operator level, as well as the electrical distributor and power system operator levels. Also, forecasting the energy production of renewable energy plants is today an essential tool for asset owners. Practically, it has direct economic implications on the net operating income of the plants, whose generated energy is sold in competitive electricity markets. Usually, energy related time series are affected by a wide variety of impurities and wild behaviors which call for forecasting algorithms whose generalization capability must be accordingly greater than other classical models. To this extent, in this dis-

sertation innovative prediction techniques based on neural and fuzzy neural paradigms are developed and tested on real-world application cases, to assess the reliability and strength of these techniques in operational contexts. In detail, energy production forecasting is carried out in different environments, such as single PV cell production, isolated grid, and multiple plants. A study is also done on other energy-related quantities, namely load and energy price. The machine learning models used in this work can be listed as: mixture of gaussian neural network, radial basis function neural network, adaptive neuro-fuzzy inference system, higher-order neuro-fuzzy inference system and echo-state network. The results of the extensive experiments reported in this thesis give a complete outlook on the performance of such methods, highlighting their versatility and good accuracy in the energy time series context. Thus, neural and fuzzy neural approaches can be considered as a reliable solution for the energy forecasting problem.

Contents

List of Figures

List of Tables

Part I

Background and Introduction

Chapter 1

Introduction

1.1 Motivation

Forecasting the evolution of real-world complex systems is one of the grand challenges of modern applied science. Thanks to widespread smart technology, time series data can be gathered almost ubiquitously from different environments, capturing the behaviours of the underlying regulating processes. Thus, time series provide a tractable mean to predict and monitor the evolution of the system although its dynamics pose a challenging entanglement. In most real-world systems, these predominant dynamics are inherently nonlinear and non-stationary; they generate time series with aperiodic patterns even under steady state. To the same extent, the evolution of real-world systems under transient conditions contributes to the budding of multiple non linearities.

Nowadays, the electric industry is transforming mainly due to the penetration of distributed grid and to the increasing need of smart and efficient management of resources and assets. Consequently, prediction problems arise from strategic analysis in production, transmission and distribution of energy. In this framework, momentous significance is given to the prediction of power outputs, loads and prices, because they have direct economic implications on the net operating income of the players.

The establishment of the pivotal role of renewable energy sources has greatly impacted the structure and handling of the current grids' infrastructure. The cardinal characteristic of RES-based systems is the intrinsic intermittence of the natural phenomena on which these resources rely. Thus, great attention must be put on the possible forecasting techniques. Also, the reliability, resiliency and flexibility needed to seamlessly incorporate smart

and private grids into the general network call for an accurate prediction of future conditions, to ensure balance in load and generation.

In the RES environment, solar photovoltaic energy production has an uncertain and non-dispatchable nature that poses serious problem for the energy management and operational planning of the involved grids. To this end, prediction of the amount of power fed by these renewable energy plants into substations, is a necessary requirement to optimize short and middle term operational decisions. In addition, accurate short-term forecasts are mandatory in the PV case for regulatory and load following issues.

1.2 Scope of the work

Time series analysis has been historically applied in the economics field mainly as a statistical tool for finance applications because of the direct implications it has on the recordings of economic quantities. In general, time series can also be gathered from a variety of different physical phenomena and their analysis results to be quite useful for a plethora of diverse fields: environment monitoring, demographics, process engineering, industry management among others.

In the past, time series analysis was carried out traditionally using statistical methods based on simple descriptive technique and on auto-regressive models. Instead, in more recent years, the use of machine learning techniques was predominant also in time series applications. This led to the employment of well-known paradigms (such as neural network) to the time-series world, making possible the realization of new models with much higher generalization capabilities for describing time series.

In the time series analysis context, an important role is played by time series prediction or forecasting which can be employed as a tool in many energy-related industrial fields. In fact, this is one of the main area of interest because of its great usefulness in terms of real world applications and aid to the management and production sides, enabling the much discussed 'smart' environment. In this context, a lot of attention is gathered by the application of novel neural and fuzzy neural paradigms to the prediction problem because of their promising performances in dynamic real-world scenarios. In this work, Neural and Fuzzy Neural models are studied with applications to the energy production field, focusing especially on the prediction of energy price, energy production, load and management.

1.3 Organization

The proposed work is structured in three main parts: the first one is an introduction to time series analysis, the second one deals with time series prediction and relative applications, the third one collects other relevant contributions made by the author on other less related subjects. In detail, this thesis is organized as follows:

- **Part one**
  - *Chapter 1* describes motivations and scope of the work.
  - *Chapter 2* introduces time series analysis.
- **Part two**
  - *Chapter 3* introduces time series prediction, with description of the models.
  - *Chapter 4* illustrates in depth the Neural and Fuzzy Neural models for time series prediction with applications.
  - *Chapter 5* presents the conclusion and discuss the results of the applications.
- **Part three**
  - *Chapter 6* describes a novel distributed clustering algorithm.
  - *Chapter 7* explains the effect on finite precision metric on a specific type of Neural model.
  - *Chapter 8* Illustrates a study on water quality prediction and monitoring using the same algorithms presented in part two.

1.4 Research Contributions

The main contribution of this dissertation is the study and the development of forecasting methodologies for real-world energy time series, based on neural and fuzzy neural paradigms. It is presented in Part II in this work, and details of the research contributions are as follows.

- A. Rosato, R. Altilio, R. Araneo, M. Panella, "Embedding of Time Series for the Prediction in Photovoltaic Power Plants", *Environment and Electrical Engineering (EEEIC), 2016 IEEE 16th International Conference on*, 2016.

- A. Rosato, R. Altilio, R. Araneo, M. Panella, "Takagi-Sugeno Fuzzy Systems Applied to Voltage Prediction of Photovoltaic Plants", *Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC/ICPS Europe), 2017 IEEE International Conference on, 1-6*, 2017.

- A. Rosato, R. Altilio, M. Panella "A New Learning Approach for Takagi-Sugeno Fuzzy Systems Applied to Time Series Prediction", *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on, 1-6*, 2017.

- A. Rosato, R. Altilio, R. Araneo, M. Panella, "Prediction in Photovoltaic Power by Neural Networks", *Energies 10 (7), 1003*, 2017.

- A. Rosato, R. Altilio, R. Araneo, M. Panella, "A Smart Grid in Ponza Island: Battery Energy Storage Management by Echo State Neural Network", *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC/ICPS Europe), 1-4*, 2018.

- A. Rosato, R. Altilio, R. Araneo, M. Panella, "Neural Network Approaches to Electricity Price Forecasting in Day-Ahead Markets", *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC/ICPS Europe), 1-5*, 2018.

- A. Rosato, R. Araneo, M. Panella, "A Distributed Algorithm for the Cooperative Prediction of Power Production in PV Plants", *IEEE Transactions on Energy Conversion*, 2018.

Other contributions In Part III of this dissertation, other research contributions are presented. Literature works on those matters are listed as follows.

- R. Fierimonte, M. Barbato, A. Rosato, M. Panella, "Distributed learning of random weights fuzzy neural networks", *Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on, 2287-2294*, 2016.

- A. Rosato, R. Altilio, M. Panella, "Finite Precision Implementation of Random Vector Functional-Link Networks", *Digital Signal Processing (DSP), 2017 22nd International Conference on, 1-5*, 2017.

- R. Altilio, A. Rosato, M. Panella, "A Nonuniform Quantizer for Hardware Implementation of Neural Networks", *Circuit Theory and Design (ECCTD), 2017 European Conference on, 1-4*, 2017.

- H.A. Nascimiento Silva, G. Laneve, A. Rosato, M. Panella, "Retrieving Chlorophyll-a Levels, Transparency and TSS Concentration from Multispectral Satellite Data by Using Artificial Neural Networks", *Progress in Electromagnetics Research Symposium-Fall (PIERS-FALL) 2017, 2876-2883*, 2017.

- A. Rosato, R. Altilio, M. Panella, "An unsupervised learning algorithm for distributed environment", *Soft Computing*, submitted in 2018.

- A. Rosato, R. Altilio, M. Panella, "On-line Learning of RVFL Neural Networks on Finite Precision Hardware", *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on, 1-5*, 2018.

- H.A. Nascimiento Silva, A. Rosato, R. Altilio, M. Panella, "Water Quality Prediction Based on Wavelet Neural Networks and Remote Sensing", *2018 International Joint Conference on Neural Networks (IJCNN), 1-6*, 2018.

- R. Altilio, A. Rosato, M. Panella, "A Sparse Bayesian Model for Random Weight Fuzzy Neural Networks", *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 1-7*, 2018.

Chapter 2

Time Series Analysis

2.1 Definition, description and examples

In this chapter we will give a comprehensive description of the properties and characteristics of a time series. First of all, we can define it as a collection of observations sequentially made through time and sequentially ordered by it [1]. This set of statistics can be occur naturally in many fields such as, but not limited to: economics, finance, environment, engineering, medicine. It is useful to begin with illustrating some examples of time series in diverse fields, to be able to intuitively grasp some knowledge on how a time series can look like.

Financial and Economic Time Series

In economics it is possible to find many recorded time series pertaining to prices, stock values, profits. They are usually relative to financial quantities recorded over a period of time that can vary from hours to years. In Fig. 2.1 an example of a financial time series is reported.

Environmental Time Series

Time series can also stem from physical quantities, linked to meteorology or geophysical studies. A classical example is the recorded temperature over time, as reported in Fig. 2.2. In this field, many applications can derive from monitoring these physical quantities by analysing the time series.

Figure 2.1: Crude Oil Price in Euro, from year 2008 to 2018

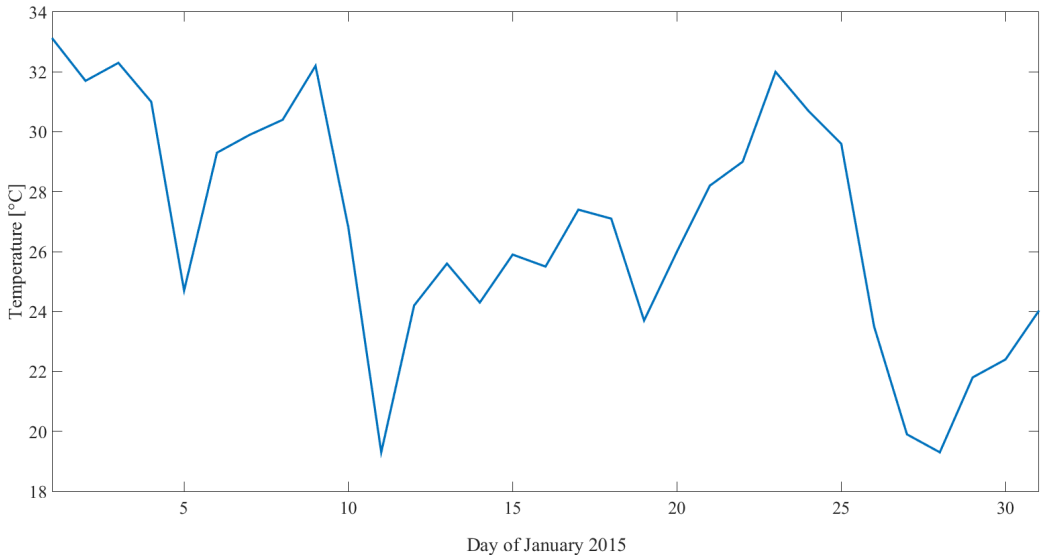


Figure 2.2: Maximum Daily Temperature in °C for Perth Airport, Australia

Other Fields

Example of time series can also be found in multiple other fields such as: demographic as in Fig. 2.3, process analysis as in Fig. 2.4, medicine as in Fig. 2.5 and so on.

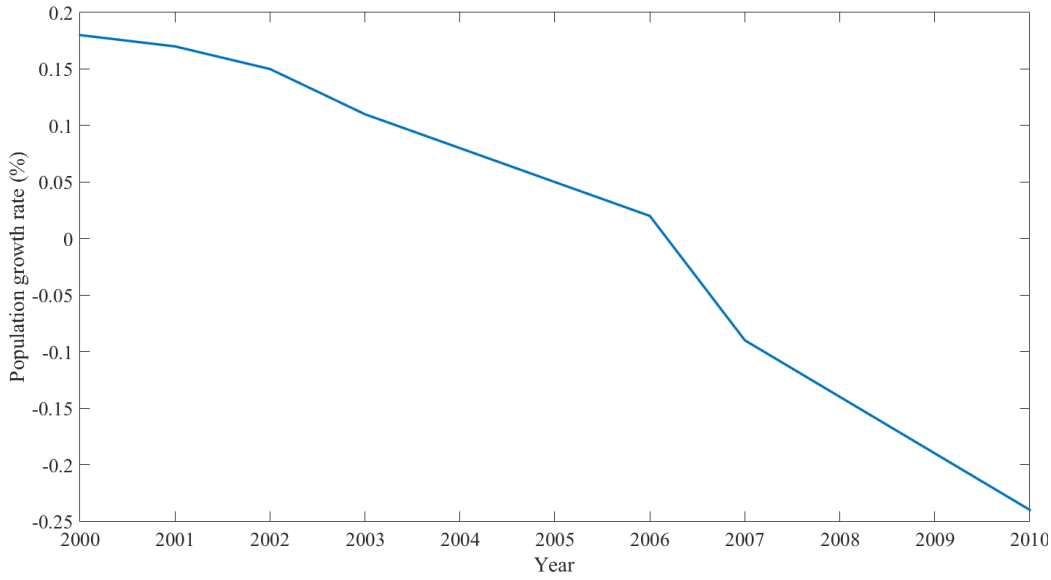


Figure 2.3: Average annual percent change in the population of Japan

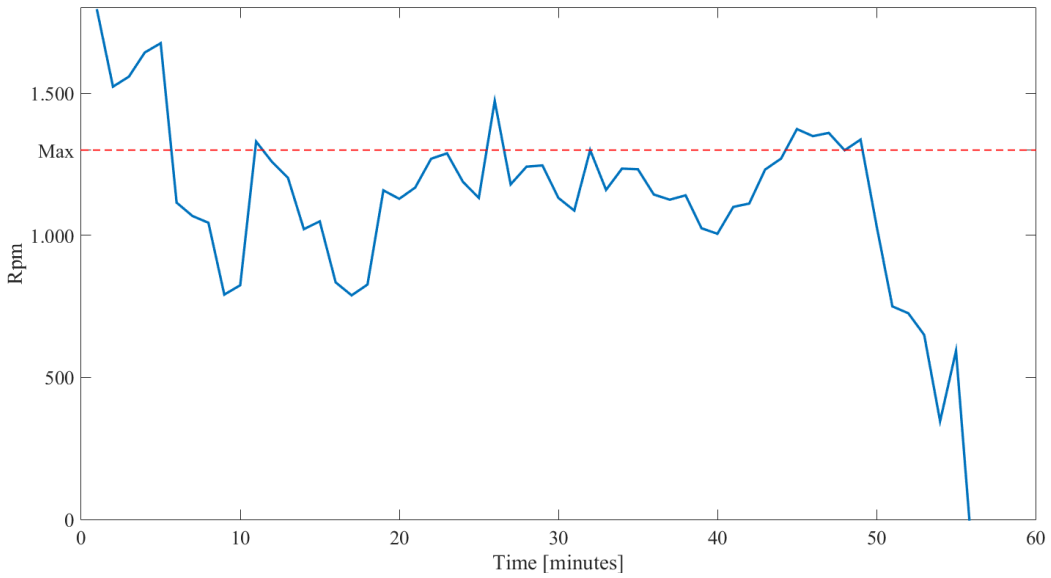


Figure 2.4: Rounds per minute of a cooling fan (with max threshold in red).

2.1.1 Basic properties and terminology

A time series can be continuous or discrete. A continuous time series arises when observations are made continuously through time (as for example in the occurrence of a certain event in time in Fig. 2.6).

A series is continuous also if its measured variables can only assume discrete values. Instead, a time series can be defined as discrete when the ob-

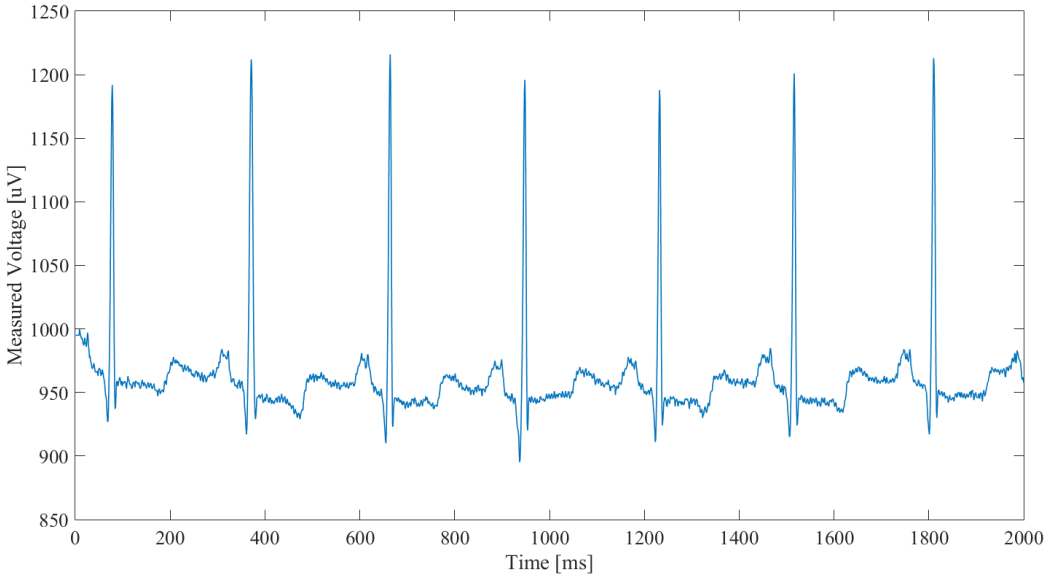


Figure 2.5: ECG time series

servations are taken only at certain specific times. Usually, the time interval between observations does not vary (equally spaced observations). In this work, all of the considered series are discrete and with equal intervals.

A discrete time series can arise in many ways. The most common is when a continuous time series is sampled at discrete points in time (sampled time series), uniformly or not [1]. Other discrete time series can be generated from aggregate values over given intervals of time (e.g. monthly rainfall) or can be intrinsically discrete (e.g. annual revenues). One of the most important and peculiar features of time series is that samples are usually dependent upon their time order. In other words, successive observations are usually *not* independent and when analysing them, this property must be taken into account. If a time series can be predicted exactly by using past observation, it can be defined as deterministic (I.e. the successive samples are dependant). On the contrary, most of the time series are stochastic, which means that can only partially be determined by past observations. In these cases exact predictions are not possible (it can be assumed that the future values have a probability distribution).

Depending on how many observed variables are changing over time, a time series can be said to be univariate or multivariate. If it is univariate,

[1]The choice of the sample interval is very important and has to be done to lose as less information as possible. If the goal is to reconstruct the original continuous signal, the sampling must be done bearing in mind the Nyquist-Shannon theorem

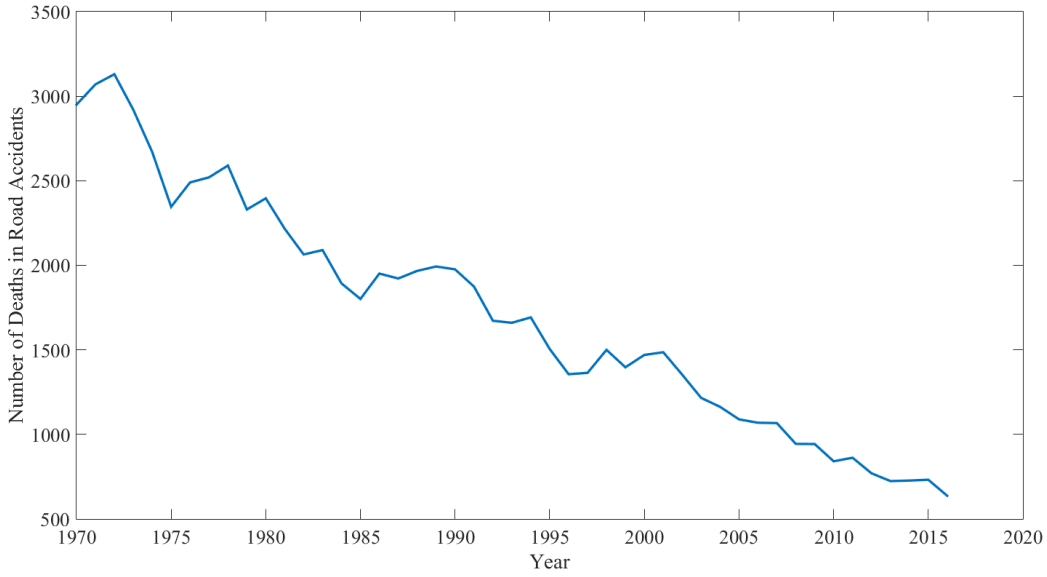


Figure 2.6: Number of deaths in road accidents in Belgium

only one variable is changing over time, for example data collected from a temperature sensor in a room. If it is multivariate, multiple variables are varying simultaneously over time, for example the three acceleration over three spatial axes collected by an accelerometer constitute a multivariate time series. In this work we will mainly focus on univariate time series.

2.1.2 Components

To further describe time series, here we introduce a straightforward approach for analysing them. In fact, it is possible to exploit some typical time series effects that can be found in most of them. In practice, a time series can be decomposed in the variations listed and explained below. All of these components can be present independently one from the other.

Seasonal variation

These are cyclic fluctuations that are related to the calendar. They can be found in a number of time series, mostly relative to demographic observations and environment quantities (as in Fig. 2.7). This seasonality is important to highlight because, if suitable for the analysis, it can be removed to give deseasonalized data that can be more effectively processed.

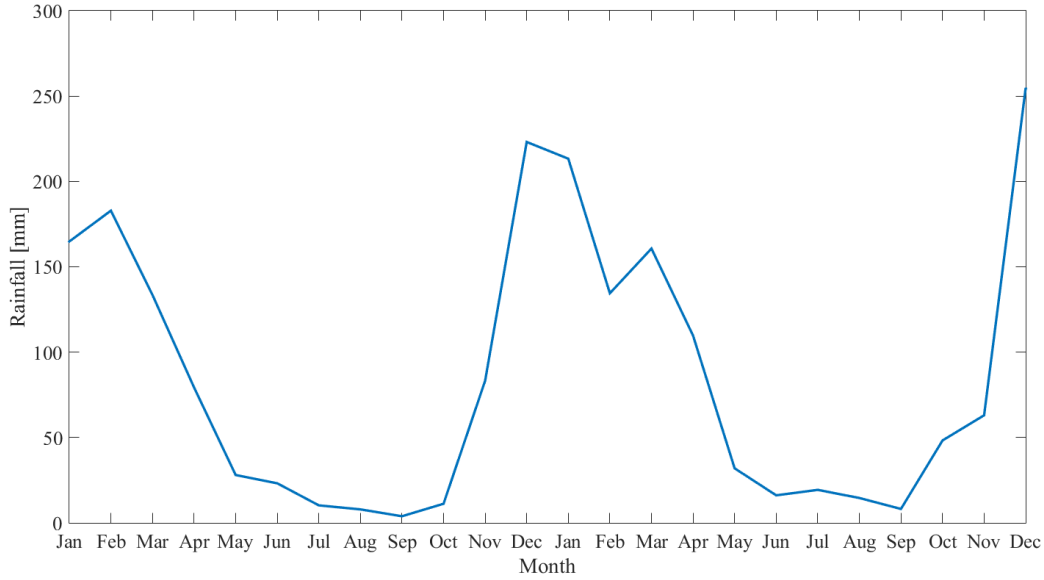


Figure 2.7: Monthhy rainfall in Mozambique for the year 1968

Other cyclic variation

There are some variations in a time series which are not related to the calendar, but that also have a fixed period. For example the daily consumption of energy (described later). There are also some series that exhibit oscillations that can be considered cyclic but with a more coarse period, such as business cycles. Cyclic variations in general are very useful for prediction purposes because, if properly accounted for, can enhance the accuracy of the prediction itself.

Trend

This variation can be defined as a long term movement in the mean. There is an issue with what can be considered 'long term' because, if there is a cyclic variation with a long period and if the number of observation available is not high enough, the complete cycle of the variation will not be available and could be confused with a trend. Thus, in general a trend can be assessed when there are sufficient observations to observe the process with a suitable time horizon. Trends can be linear or non-linear, and can in general be described by different simple mathematical models (e.g. fitted on a linear/quadratic/cubic polynomial, logarithmic, piecewise linear function).

Other irregular fluctuations

Once cyclic variations are removed, a time series can still exhibit some kind of fluctuations that may not be random. Those can also be called residuals.

2.1.3 Time Plot

Undoubtedly the first step for every kind of time series analysis is to plot it against time. This results in a graph called simply time plot, from which all its components can be easily and visually identified. The plotting operation may seem trivial but in real problems it is not as easy as it sounds. The visual inspection of a time series is always important, because can give us many information on how to analyse it. Thus, a lot of attention must be given to elements of the plot such as scales (relative to both time and other variables), size, title. Some operations can be done on the time series to be able to better visualize its components. These transformations are useful when there is the need to stabilize the variance, modify the seasonal effect (if directly proportional) or to make the data normally distributed.

2.1.4 Real Data

The time series that will be used in this work, like most of them stemming from observation of real-world phenomena, can contain oddities and impurities due to various reasons. This can include: missing samples, format errors, outliers and discontinuities. In fact, the preliminary part of the analysis consists in the examination of the data itself to assess its quality and, if necessary, modify it and clean it. This procedure is essential and tackles most of the aforementioned errors. Data cleaning could include identifying and correcting obvious errors, accounting outliers, and filling in any missing observations. Most of the times, all of these defects are evidently revealed by the time plot, hence its importance. Without a reliable procedure to clean real data, no precise analysis is possible, in particular the adequate treatment of oddities in relation to the context of operation.

2.2 Stationarity

A time series can be said to be stationary if a shift in time does'nt cause a change in the shape of the distribution. In other words, stationarity is present

when attributes like mean, variance and covariance are constant over time. Some definitions can be formalized in this regard:

- Strict stationarity: A time series is said to be strictly stationary if the joint distribution of $X(t_1), \ldots, X(t_k)$ is the same as the joint distribution $X(t_1 + \tau), \ldots, X(t_k + \tau)$, for every value of k and τ.
- First order stationarity: a time series has first order stationarity if its first order momentum (mean) does not change with time. Any other statistics (like variance) can change.
- Difference-stationarity: a time series which needs one or more differencings to become stationary.

2.3 Approaches to time series analysis

First, when working on time series analysis, we have to make sure if we understand the problem correctly in all its aspects. This is true for every experiment or scientific research, but in time series analysis the context of a problem is crucial to the objectives, in particular regarding the collection of data. This because if the observed time series is too short or has the wrong measured variables, it may be impossible to solve the given problem. Time series analysis can be used to reach several possible objectives that will be discussed in this section, in turn.

Describe

The most common objective of time series analysis is description of the time series itself. Usually, apart from the time plot, cited above, a time series is well described by some model with few parameters values. These parameters can reflect the salient features of the time series and give us a knowledge of the time series itself.

Predict

In many applications in various fields (e.g. economics, finance, industry, geoscience, etc.), it may be of interest to predict the future values of an observed time series. In many works, including this one, the terms 'prediction' and 'forecasting' are used interchangeably, but note that some authors do not.

The goal of the prediction could also be different from the accurate determination of the actual future single values of the observation. For example it would be possible to predict only one of the aforementioned variations, or classify the series into classes, for example trying to forecast if the variable will go up or down.

Transform

In some cases, it could be of interest to transform a time series into another (e.g. oil prices into interest rates), to be able to reach a deeper understanding of the mechanism that generated a given time series or to simply visualize it in a different way.

This work is primarily concerned with the prediction objective, but the description is often a prerequisite. In fact, all of the discussed analysis objectives are inherently bound and are routinely considered together.

Part II

Energy Time Series Prediction

Chapter 3

Time Series Prediction

3.1 Introduction

Good predictions are cardinal in numerous activities in diverse fields. For example, scientific research, industrial monitoring, commercial analysis, economic and financial business are all among the applications which benefit of a good forecasting. The goal of this chapter is to give an overview of the most commonly used methods for prediction, providing the reader with the intuitive ideas and basic theory of every presented method. The first and basic prediction method [1] 1^0 is called *judgemental forecasting* and is based on intuition, subjective judgement and information that can be gathered via the descriptive analysis explained in Chapter 2. Apart from that, prediction methods can be coarsely divided in two categories:

- Univariate methods: in this category, forecasting depends only on the values of the single time series being predicted.
- Multivariate methods: in this category, forecasting depends, to a certain degree, on values of more than one time series variables. This methods can depend on multivariate models.

In general, the former judgemental analysis is always present and taken into account to incorporate subjective information to the prediction.

[1] A *forecasting method* or *prediction method* is a procedure for computing future values of a time series. It may be just a rule and it can also not depend on a mathematical model. Thus, the meaning of the terms 'model' and 'method' must be kept diverse. Unfortunately, in literature the locution 'forecasting model' is often loosely used to address a forecasting method.

3.2 Univariate Prediction Models

In this section a variety of common forecasting models is introduced and discussed. These are the basis of many univariate prediction methods and their knowledge is mandatory to be able to follow further analysis carried on later in this work.

3.2.1 Autoregressive and Moving Average models

This class of models represents a fundamental prediction tool, and it is the basis of a lot of elementary ideas in time-series analysis. Most of the time, the acronym ARIMA (Autoregressive Integrated Moving Average) is used to address this category of models; it can be found in [2].

Autoregressive processes

An autoregressive (AR) process of order p (AR(p)) is defined as a time series $\{X_t\}$ that is a weighted linear sum of the past p values plus a random additive quantity:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + Z_t \tag{3.1}$$

where $\{Z\}$ is a purely random process with zero mean and ϕ_i are the parameters of the model. If we define an operator L such that $LX_t = X_{t-1}$, the model can be written compactly:

$$\phi(L)X_t = Z_t \tag{3.2}$$

where

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p \tag{3.3}$$

The term autoregression stems from the fact that value at time t depends linearly on the last p values, like a regression model. The trivial example of an AR process is the first order one:

$$X_t = \phi X_{t-1} + Z_t \tag{3.4}$$

The model properties depend on the characteristics of the coefficients ϕ. If $\phi = 1$ then the model is reduced to a random walk (when non-stationary). If $|\phi| > 1$ the series is explosive and thus non-stationary. If $|\phi| < 1$ the process is stationary [3] and its autocorrelation function decreases exponentially. A useful property of an AR(p) process is that the partial autocorrelation function is zero at all lags greater than p. This means that the sample partial autocorrelation function can be used to help determine the order of an AR process, which is usually unknown, by looking for the lag value at which the sample partial autocorrelation function 'cuts off' to zero.

Moving average processes

A moving average (MA) process of order q (MA(q)) is a time series that is a weighted linear sum of the last q random noise samples (called *shocks* in other works):

$$X_t = Z_t + \theta_1 Z_{t-1} + \cdots + \theta_q Z_{t-q} \tag{3.5}$$

where $\{\theta_j\}$ represents the parameters of the MA process. Thus the value at time t is a sort of moving average random noise samples, which are unobservable. It can be proven that the first order MA process (MA(1)) is stationary for all θ values. Similarly to the AR process, the MA can also be written in the form:

$$X_t = \theta(L) Z_t \tag{3.6}$$

Autoregressive moving average processes

A mixed process model with p autoregressive terms and q moving average terms (ARMA(p,q)) can be written as:

$$\phi(L) X_t = \theta(L) Z_t \tag{3.7}$$

The importance of ARMA processes is that many real data sets may be approximated in a more parsimonious way (meaning fewer parameters are needed) by a mixed ARMA model rather than by a pure AR or pure MA process.

Autoregressive integrated moving average processes

This more general class of models derive from practice, where most time series to which predictors are applied are non stationary. Thus, stationary AR, MA or ARMA processes cannot be applied directly. One possible way of handling non-stationary series is to apply differencing so as to make them stationary. The first differences, namely $X_t - X_{t-1} = (1 - L)X_t$ may may themselves be differenced to give second differences, and so on. The dth differences may be written as $(1 - B)^d X_t$. If the original data series is differenced d times before fitting an ARMA(p,q) process, then the model for the original series is defined as ARIMA(p,d,q), where the 'I' stands for integrated. This generalization can be expressed as:

$$\phi(B)(1 - B)^d X_t = \theta(B)Z_t \tag{3.8}$$

It can be noted that $\phi(B)$ and $\theta(B)$ are both just equal to unity (i.e. p and q are zero) and d equals one, then the model reduces to an ARIMA(0,1,0) model, given by:

$$X_t - X_{t-1} = Z_t \tag{3.9}$$

which is trivially the random walk model. When fitting the models in this section, the main difficulty does not lie in estimating the coefficients ϕ and θ but is instead found in valuating the order of the process (regarding p and q). With ARIMA models, there is an additional problem in choosing the required order of differencing d. Some formal procedures are available, but in many cases the method merely relies on differencing the series until the autocorrelation function comes down to zero in a decent time. First-order differencing is usually adequate for non-seasonal series, but second-order differencing is sometimes needed. Once the series has been made stationary, an ARMA model can be fitted to the differenced data in the usual way. More details on the fitting procedure can be found in [4], [5] and [6].

Other processes

Apart from the aforementioned AR, MA, ARMA and ARIMA processes, there are a number of variations to them that are suitable for prediction of different type of time series. For instance, if the series is seasonal, a seasonal ARIMA model (SARIMA) can be generalized from the ARIMA in 3.2.1. When fitting

SARIMA models, the seasonal and non-seasonal orders of differencing must be chosen beforehand, to remove most of the seasonality and make the series stationary. After that, an ARMA-type model is fitted to the differenced series (there may be AR and MA terms at lags which are a multiple of the seasonality period). A variant of the SARIMA model can be employed when if there is no need for the model coefficients to stay constant throughout the year. In these periodic autoregressive models the values of the autoregressive parameters are allowed to vary through the seasonal cycle. More generally periodic correlation arises when the size of autocorrelation coefficients depends, not only on the lag, but also on the position in the seasonal cycle. Another interesting variant of ARIMA models stems from allowing the d in the model to be non-integer (fractional differencing, thus ARFIMA or FARIMA). This technique entails an added difficulty in the computation and in the interpretation of the model itself but can be advantageous in the prediction of many time series which can be modeled with a so called long memory model in which deviations from the long-run mean decay more slowly than an exponential decay. Other processes based on ARIMA models are the one called ARIMAX, where explanatory foreign variables are added to the model to ensure better generalization capabilities.

3.2.2 Nonlinear models

Up to $10 - 15$ years ago, in literature, time series analysis and forecasting in particular, has focused on linear methods and models, essentially because of their mathematical convenience and intrinsic convenience. Despite their simplicity, linear univariate methods often deliver properly good results and adequate approximation for most of the practical tasks. However, process generated in real life may be (and most of them are!) not linear. Thus, many observed time series disclose non linear features which cannot be modeled via linear approaches. In fact, compelling forecasting of future states of a complex system from time series is still a challenge, mainly due to diverse combinations of the nonlinear (and also non-stationary) dynamic behaviors exhibited by these systems. The statistical characteristics which can be observed in a time series emerging from a non linear system are, among others: aperiodicity, non-normality, time irreversibility, non-normality, asymmetric cycles. In complex dynamic systems, time series data capture the dynamic behaviors and causalities of the underlying processes and provide a tractable means to

predict and monitor system state evolution. For most real-world systems, the vector field of state dynamics is a nonlinear function of the state variables; i.e., the relationship connecting intrinsic state variables with their autoregressive terms and exogenous variables is nonlinear. Statistically speaking, time series may be considered as emerging from a nonlinear dynamic system if it is marked by characteristics such as non-normality, aperiodicity, asymmetric cycles, multi-modality, nonlinear causal relationships among the lagged variables, variation of forecasting performance over the state space [7]. Time series emerging from such complex systems exhibit aperiodic (or chaotic) patterns even under steady state. Also, since real-world systems often evolve under transient conditions, the signals obtained therefrom tend to exhibit myriad forms of non-stationarity. For these reasons, conventional approaches do not adequately capture the system evolution (in particular regarding sensitivity to quantity and quality of a priori information) in these applications. From around the year 2000 and forth, the need for such nonlinear forecasting methods brought to the development of the nonlinear models described in this section.

In this section, various methods reported in the recent works are described and categorized based on their application to real-world data. Coarsely speaking, these forecasting methods may be classified based on the assumptions that: there is a known trend in the first moments, a piecewise stationarity of the signals is present, parameters vary progressively, the signal can be decomposed into linear segments. The focus is more acute on these models because their knowledge will introduce the reader to the approaches mainly used in the rest of the work.

Nonlinear AR models

As already discussed earlier in this work, AR, MA and their derivatives and combinations assume a linear relationship between the lagged variables. As most of the linear models, ARMA performance degrades considerably when time trends, seasonality and non-stationarity appear in the time series. In general, this category of models just gives a rough approximation of complex, real-world systems, failing to accurately approximate their evolution. For the sake of truth, however, it should also be pointed out that these methods can be used to remove or reduce non-stationarity, but limited only to the first order one. Apart from the obvious, merely formal, ways to generalize

the linear autoregressive model pf order p via a generic nonlinear function of the samples (NLAR(p)) or introducing a time-varying parameter, the first approach to nonlinearization of the autoregressive models is represented by the threshold autoregressive model (TAR). In this case, an AR model of order 1 is used, but the parameters depend on whether the past observations are above or under a certain threshold. In other words, TAR models assume a piecewise linearity, dividing the input space in terms of a threshold variable. Also, the threshold itself can be assumed to vary with time, allowing good flexibility of the model, especially regarding variations in model parameters through a regime-switching behavior. A variation of these models is the Smooth Transition AR (STAR) which can capture transition between different regimes with a less hard discontinuity [8], [9]. Other further variation include a design of time lagged state space (proposed in [10]).

Support Vector Machine models

SVM-based forecasting methods use a class of generalized regression models, such as Support Vector Regression (SVR) and Least-Squares Support Vector Machines (LS-SVMs in [11]). An SVM maps the inputs into a higher-dimensional feature space; only the estimate of the inner product of the mapped pattern is necessary. This estimate is expressed as a linear combination of specified kernel functions and thus the method can be classified by it as linear, Gaussian or RBF, polynomial, and multilayer perceptron). A linear regressor is then constructed by minimizing the structural risk minimization (the upper bound of the generalization error).

Neural networks

A completely different type of non-linear prediction model is provided by Neural Networks (NNs), which are designed with a structure that mimics the human brain to certain extent. Since the core of this work is focused on prediction via NN and Fuzzy-NN, we just give a brief general introduction here, remanding a more detailed description of the single application cases later on. NNs have been applied successfully to a very wide variety of scientific problems, notably to pattern recognition [12]. These models are vastly used in classification problems where a collection of features is presented to the network and the task is to assign the input feature to one or more classes. Another common field of use for NNs is constitued by regression problems

in which a smooth interpolation between points has to be found. In general, in pattern recognition problems all the relevant information is presented simultaneously to the network. In contrast, time series prediction involves processing of patterns that evolve over time. In this case, temporal information may be presented spatially to the network by a time-lagged vector. As stated, these models provide a very large set of solutions, which can be considered a subject on their own and in fact cover the vast majority of this work. Although they cannot be considered as conventional time series models in the sense that there is usually no attempt to model the 'error' component, they can be modeled and designed to result in approaches with a high forecasting ability. A NN can be described as a system connecting a set of inputs to a set of outputs in a possibly non linear way. Speaking about time series, its output could be the value of a time series to be predicted and the input could be lagged values of the series and of other explanatory variables. The connections between inputs and outputs are typically made via one or more hidden layers of neurons or nodes. The structure of an NN is usually called the architecture and choosing it includes determining the number of layers, the number of neurons in each layer, and how the inputs, hidden layers and output(s) are connected. It is necessary to highlight that each connection in a network has an associated weight which has to be estimated to reach best result. Also, the activation function of each neuron must be chosen. All of these analysis are carried out for each problem and described thoroughly for each application case later in this work. It can be said that in general, the weights to be used in the NN model are estimated from the data by some sort of optimization (for example minimizing the sum of squares of the one-step ahead errors) over a suitable portion of the data. This non-linear optimization problem is one of the core procedures of the forecasting methods with NN. It is sound practice to divide the data into two sections, to fit the NN model to the first section (training set) and use the remainder of the data (test set) to check the prediction performance. The weights to estimate are typically a large number and the procedures to fit them are complicated and time consuming. In the specialized literature, the iterative estimation procedure is a training algorithm, which often implies the widely known back propagation. Note that in some analyses, the fitting of NN is done to get the best prediction of the test set data, rather than the best fit to the training data. In this case the test set is no longer 'out-of-sample' in regard to model fitting and so a third section of data should be kept in reserve so that gen-

uine out-of-sample forecasts can be assessed [2]. Neural models are also often coupled with fuzzy logic to give fuzzy-neural predictors. These approaches are similar in the method but with different models, in which fuzzy rules are incorporated. Further details can be found in the remainder of this work.

3.2.3 Heteroschedastic Models

To overcome the limitation on the order of the non-stationarity that can be captured by ARIMA models, the work [13] introduced a class of heteroschedastic models (Autoregressive Conditionally Heteroscedastic models, ARCH) that can describe the dynamic changes in conditional variance as a (often quadratic) function of past observations. In this case, the objective is to give better estimates of the local variance, instead of improving directly forecasts of the observations. A generalization of ARCH models, called GARCH, has also been introduced in [14] where additional dependencies are permitted on lags of the conditional variance. This model has a representation that resembles the ARMA, so those two models share many properties. ARCH and GARCH approaches find profitable results mainly in financial time series prediction [15], [16]. Also, several extensions of the original GARCH model have been proposed in the past, by specifying different parameterizations to capture serial dependence on the conditional variance. For instance, some of them are integrated GARCH (IGARCH), exponential GARCH (EGARCH), threshold GARCH (TGARCH or GJR), GARCH-in-mean (GARCH-M), and so forth. All of them are able to observe some common characteristics of returns series, in particular volatility clustering, leverage effect, and heavier/fat tails, although they remain weak in capturing wild fluctuations and unanticipated events.

3.2.4 Nonparametric models

If the models are correctly specified, parametric models can provide accurate predictions but they tend to become suboptimal whenever the underlying dynamics are unknown or difficult to determine. Most real-world complex systems are inherently nonlinear (and non-stationary). In contrast, the class of nonparametric models is able to simplify the efforts of modeling because it does not impose any structural constraint. Consequently, the modeling

[2]For this reason, some works include a third chunk of data, called 'validation set', on which the model is appropriately verified before the actual tests take place

and forecasting accuracy for nonlinear and non-stationary time series is improved. One of the main drawbacks of this procedure is that, to glean underlying structures and relationship, nonparametric models usually require larger datasets than their parametric counterpart. These models for time series forecasting include: State space neighborhood and local topology-based models [17], bayesian inference models [18], and functional decomposition models [19]. Since it is out of the scope of this work, an exhaustive definition of each model is not included but can be found in the cited works.

3.3 Multivariate prediction models

All of the models considered before are univariate, they use a single time series to obtain the prediction. In some situations, different observations are taken simultaneously on two or more time series. For example, various measures of economic activity in a particular business sector can be collected through time (price, revenue, shares). Given such multivariate data, there could be the need to develop a multivariate model to describe the interrelationships among the series, and then to use this more complex model to make forecasts. The multivariate prediction models are only listed in this section (with sources), since they found rare applications outside economic analysis and are not used in the rest of the work: multiple regression, transfer function and distributed lag models, econometric models and multivariate versions of AR and ARMA models, including vector autoregressive (VAR) models. Since fitting a multivariate model is still a not easy task, computationally speaking, an useful remark must be done on the goodness of the results of multivariate prediction models. While multivariate models can usually be found which give a better fit than univariate models, there are several reasons why this do not lead to better forecasts: the computation of accurate forecasts of a dependent variable may require future values of explanatory variables; multivariate models are more complicated and thence more vulnerable to misspecifications; observed multivariate data may not necessarily be suitable for fitting a multivariate model and with more variables, errors and outliers are more common. A comparison and description of these models can be found in [20].

3.4 Error Measures

Since a lot of models and methods are presented in this work, an important remark must be done on how these solutions are evaluated in terms of forecasting performance. In general, the interest is to know how much accurate a prediction is, considering the best forecasting method as the most accurate one, in terms of some error metric. This section discusses ways of measuring prediction accuracy. The first thing to say is that the accuracy metric depends on the field of application of the prediction. In two diverse areas as, for example, finance and environment, the focus of the error will be so different that two distinct metrics must be used. The basic error metric is found by computing the residuals, namely the within-sample one-step ahead forecast error. Clearly, the less the error, the more accurate the forecast:

$$e_t = x_t - \hat{x}_{t-1} \tag{3.10}$$

where $\hat{x}$ denotes the predicted sample. In statistics, the accuracy is almost always measured by computing the mean square error (MSE) or its root (RMSE). The MSE is amenable to theoretical analysis but poses a problem of interpretation: it implies an underlying quadratic loss function that can cloud the understating (e.g. if the error doubles, is it two or four times as bad?). Instead, the RMSE is in the same units as the measured variable and so is often a better descriptive statistic. Also, normalization can be applied to the MSE to give a better accuracy metric, by dividing from the difference between the maximum and minimum value of the series (NMSE). Usually, the customary form of calculating the MSE is the prediction mean square error (PMSE):

$$PMSE = \sum_{t=N-m+1}^{N} e_t^2/m \tag{3.11}$$

where N is the number of samples, and m is the number of last observation on which it is computed. Sometimes, when the loss function is thought to be linear, the preferred estimate of accuracy is given by the mean absolute error (MAE):

$$MAE = \sum_{t=N-m+1}^{N} \|e_t\|/m \tag{3.12}$$

A problem of using the PMSE is when forecasts are required for more than one step ahead. In this case the question arises as to whether the formula in Eq. 3.11 must be modified to look at the step ahead errors or the within sample ones. Another type of scale-independent measure that is used is computed via the percentage errors, instead of the raw ones, and is called mean absolute prediction error (MAPE):

$$MAPE = \sum_{t=N-m+1}^{N} \|e_t/x_t\|/m \tag{3.13}$$

which, since it is based on percentages, must be used carefully when dealing with time series with zeros. The error can be also measured via the mean absolute range error (MARE), which is also normalized:

$$MARE = \sum_{t=N-m+1}^{N} \frac{\|x_t - \tilde{x}_t\|^2/m}{x_{\max} - x_{\min}} \tag{3.14}$$

This, compared to the RMSE, it is dimensionless and incorporates the inherent property of the data structure being analyzed. In some occasions, it would be convenient to express the error via logarithmic scale. By analogy with the telecommunication case, this can be expressed via the noise to signal ratio (NSR) in dB:

$$NSR_{dB} = -10log_{10}\frac{\sum_{t=N-m+1}^{N}(x_t - \tilde{x}_t)^2/m}{\sum_{t=N-m+1}^{N} x_t^2} \tag{3.15}$$

Chapter 4

Neural and Fuzzy Neural Networks for Energy-related Time Series Prediction

In this chapter the prediction of energy-related time series will be delineated and characterized with theoretical remarks on the algorithms used in various real-world application cases.

4.1 Introduction

In recent years, electrical power systems have progressively evolved from centralized bulk systems to decentralized systems in a distributed generation (DG) scenario characterized by smaller generating units connected directly to distribution networks near the consumer [21]. Several economic and environmental reasons, driven by government financial incentives [22], have considerably pushed the widespread adoption of DG in the energy marketplace, especially that from renewable resources (e.g., solar power and wind energy).

However, technical barriers often prevent the entrance of DG into the current distribution infrastructure with a significant penetration level due to the intermittent nature of the renewable energy resources that often do not match the energy load demands [23]. Indeed, the transition to an energy economy primarily founded on renewable resources depends on overcoming the difficulties associated with the variability and reliability of these non-dispatchable resources [24]. These issues give rise to substantial critical issues with respect to the usual working habits of the transmission (TSO) and distribution (DSO)

system operators, utility companies, as well as power producers: voltage and frequency regulation, islanding detection, harmonic distortion, distribution issues, as well as demand side management of prosumers. Certainly, energy storage systems (EES) could become a valuable response providing specific support services for renewable energy production in order to reduce short-term output fluctuations and, consequently, improve power quality [25].

In this framework, it is of paramount importance to forecast accurately the power output of plants over the next hours or days in order to integrate in an optimal way the DG from non-programmable renewable resources into power systems on a large scale [26]. The ability to forecast the amount of power fed by renewable energy plants into substations is a necessary requirement for the TSO in order to optimize short- and middle-term decisions, such as corrections to out-of-region trade and unit commitments [27]. Besides, accurate short-term and intra-hour forecasting are required for regulatory, dispatching and load-following issues [28], while power system operators are more sensitive to intra-day forecasts [29], especially when handling multiple load zones, since they avoid possible penalties that are incurred due to deviations between forecasted and produced energy. Moreover, power system operators are mostly interested in developing methods to be used in the framework of the daily session of the electricity market where the producers must present energy bid offers for the 24 h of the following day [30].

In general, forecasting problems can be broadly divided into four major categories [31], namely long-, medium-, short- and very short-term forecast, on the basis of the time horizon's scale, which ranges from several years in the long-term forecasting to hours in the medium-term forecasting, till arriving at minutes in the very short-term forecasting. When employed in photovoltaic (PV) power plants management, forecasting techniques can be applied directly to the produced power (direct methods) or indirectly to the solar irradiation, which is the main factor related to the output power that is calculated in a second successive step (indirect methods) [32]. Both cases share similar techniques that can be coarsely divided [31] into persistence methods, physical techniques, statistical techniques and hybrid models. Usually, physical methods try to obtain an accurate forecast using a white box approach, that is to say physical parametric equations. On the other hand, statistical or probabilistic methods try to predict the relevant quantities extracting dominant relations from past data that can be used to predict the future behavior. In addition, both methods can be properly mixed, originating the so-called

"gray box" approaches [33].

We already stated that in the prediction of time series, past observations are used to develop a method able to describe underlying relationships among data. Therefore, this can be viewed as extrapolating a mathematical model of the data, especially for those contexts where missing and incomplete data can limit the ability to gain knowledge about the data generated by the underlying, unknown process. As in the cases analysed here, complexity and dynamics or real-world problems require advanced methods and tools able to use past samples of the sequence in order to make an accurate prediction [34]. Additionally, the problem of forecasting future values of a time series in an energy context is often mandatory for the cost-effective management of available resources. These are well-known hard problems [26], given the non-stationarity and, often, the non-linearity of time series, which result in a complex dynamics that is hard to model adequately by using standard predictive models.

For many years in this application field, regressive statistical approaches [35] have been considered for prediction of energy-related time series; among them, the already cited moving average (MA), auto regressive (AR), auto regressive moving average (ARMA), auto regressive integrated moving average (ARIMA) and autoregressive integrated moving average with exogenous variables (ARIMAX) are typical examples of these approaches. Unfortunately, standard structural models provide a poor representation of actual data and therefore result in poor accuracy when used for forecasting. Consequently, many worldwide research activities intend to improve the accuracy of prediction models. In this regard, computational intelligence is considered as one of the most fruitful approaches for prediction [36]. Several forecasting methods, with different mathematical backgrounds, such as fuzzy predictors, artificial neural networks (ANN), evolutionary and genetic algorithms and support vector machines, have been considered [37]. Nevertheless, dealing with noisy and missing training examples and the lack of robustness against outliers are still open problems.

4.2 Models

In this section, some prediction models will be described which are all based on neural and fuzzy-neural approaches. These models are the core of prediction methods which constitute the central part of the present work. The goal of

this section is to delineate these methods in order to introduce a report and discussion of the employ of such methods in applications involving real-world time series.

4.2.1 Radial Basis Function

An RBF neural network is used to build up a function approximation model having the following structure:

$$f(\boldsymbol{x}) = \sum_{i=1}^{M} \lambda_i \phi(\|\boldsymbol{x} - \boldsymbol{c}_i\|), \tag{4.1}$$

where $\boldsymbol{x} \in \mathbb{R}^N$ is the input vector, $\phi(\cdot)$ is a radial basis function centered in $\boldsymbol{c}_i$ and weighted by an appropriate coefficient λ_i. The choice of $\phi(\cdot)$ and $\boldsymbol{c}_i$ must be considered for the ability of the network in its approximation capability. Commonly used types of radial basis functions include:

- *Gaussian:*

$$\phi(r) = e^{-(\epsilon r)^2};$$

- *Multiquadric:*

$$\phi(r) = \sqrt{1 + (\epsilon r)^2};$$

- *Inverse Quadratic*

$$\phi(r) = \frac{1}{1 + (\epsilon r)^2},$$

where ϵ is the so called *shape parameter* which regulates the 'flatness' of the RBF. Several methods can be used to minimize the error between desired output and model output and hence, to identify the parameters $\boldsymbol{c}_i$ and λ_i [38].

4.2.2 Gaussian Mixture Model

In the MoG neural network M different Gaussian components in the joint input-output space $\Re^N \times \Re$ are used in a mixture density model:

$$p(\boldsymbol{x}, y) = \sum_{j=1}^{M} \pi^{(j)} G^{(j)}_{\boldsymbol{x},y}(\boldsymbol{x}, y)\,, \tag{4.2}$$

where $\pi^{(j)}$ is the prior probability of the jth Gaussian component $G^{(j)}_{\boldsymbol{x},y}$, $j = 1 \dots M$. The mixture parameters are estimated by the Splitting Hierarchical Expectation Maximization (SHEM) technique [39].

The conditional density $p(y|\boldsymbol{x})$ is computed from (4.2):

$$p(y|\boldsymbol{x}) = \sum_{j=1}^{M} h_j(\boldsymbol{x}) G^{(j)}_{y|\boldsymbol{x}}(y)\,, \tag{4.3}$$

where $G^{(j)}_{y|\boldsymbol{x}}(y)$, $j = 1 \dots M$, is the conditional density of $G^{(j)}_{\boldsymbol{x},y}$, and $h_j(\boldsymbol{x})$ is the projection of $G^{(j)}_{\boldsymbol{x},y}$ into the input space. It can be evaluated by the marginal density $G^{(j)}_{\boldsymbol{x}}$ of $G^{(j)}_{\boldsymbol{x},y}$:

$$h_j(\boldsymbol{x}) = \frac{\pi^{(j)} G^{(j)}_{\boldsymbol{x}}(\boldsymbol{x})}{\sum_{k=1}^{M} \pi^{(k)} G^{(k)}_{\boldsymbol{x}}(\boldsymbol{x})}\,. \tag{4.4}$$

4.2.3 Adaptive Neuro Fuzzy Inference System

An ANFIS neural network implements a fuzzy inference system to approximate the function $y = f(\boldsymbol{x})$, $f : \mathbb{R}^N \to \mathbb{R}$. Fuzzy-based approaches such as the one detailed here are very flexible and act well when the mappings to be approximated are characterized by very irregular behaviors. An ANFIS network is composed of M rules of Sugeno first-order type, where the kth rule, $k = 1 \dots M$, is:

$$\text{If } \mathrm{x}_1 \text{ is } B^{(k)}_1, \dots, \text{ and } x_N \text{ is } B^{(k)}_N \text{ then } y^{(k)} = \sum_{j=1}^{N} a^{(k)}_j x_j + a^{(k)}_0\,, \tag{4.5}$$

where $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 \cdots x_N \end{bmatrix}$ is the input to the network and $y^{(k)}$ is the output of the rule. The antecedent part of the rule depends on the membership functions (MFs) $\mu_{B^{(k)}_j}(x_j)$ of the fuzzy input variables $B^{(k)}_j$, $j = 1 \dots N$; the consequent part is determined by the coefficients $a^{(k)}_j$, $j = 0 \dots N$, of the crisp output $y^{(k)}$. By using standard options for composing the input MFs

and combining the rule outputs [40], the output of the ANFIS network is represented by:

$$\widetilde{y} = \frac{\sum_{k=1}^{M} \mu_{\boldsymbol{B}^{(k)}}(\boldsymbol{x})\ y^{(k)}}{\sum_{k=1}^{M} \mu_{\boldsymbol{B}^{(k)}}(\boldsymbol{x})}\ , \tag{4.6}$$

where $\widetilde{y}$ is the estimation of y and $\mu_{\boldsymbol{B}^{(k)}}(\boldsymbol{x})$ is the composed MF of the kth rule.

When dealing with data driven estimation procedures the mapping $y = f(\boldsymbol{x})$ is known by numerical examples (i.e., by a training set of P input-output pairs $\{\boldsymbol{x}_i, y_i\}$, $i = 1 \dots P$). Clustering on the training set is thus a useful approach to the synthesis of ANFIS networks. Different types of clustering approaches can be used in this regard, followed by a suited classification for associating patterns a rules [41, 42]. For instance, clustering in the joint input-output space can overcome some drawbacks pertaining most of traditional approaches, where clustering is used only to determine the rule antecedents in the input space. In this work, an ANFIS synthesis procedure is used that is based on the joint input-output space approach. The first step is using a suitable clustering procedure to determine the coefficients of the Sugeno rule consequent. Then the rule antecedents are determined using a fuzzy classifier. Fuzzy Min-Max classifiers represent a valid solution in this regard because of their computational effectiveness. In particular, the use of Min-Max classifiers based on adaptive resolution mechanisms will be adopted. If the architecture consists of a suitable number of rules, the generalization capability of an ANFIS network can be sufficient. This is a crucial problem since in general neuro fuzzy networks can be affected by overfitting (for example in case of noisy or ill-conditioned data). To automatically determine the optimal number of rules, other techniques can be employed, based on well-known concepts of learning theory [43].

4.2.4 Higher Order Neuro Fuzzy Inference System

The HONFIS model adopted for data regression is a Takagi–Sugeno (TS) fuzzy inference system that is a generalization of ANFIS where the consequent part in Equation (4.21) is a polynomial of order greater than one combining the input values x_j, $j = 1 \dots N$. It is made of M different fuzzy rules, and

the coefficients are obtained through the use of a clustering procedure in the joint input-output space [42].

Let $\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_M\}$ be a set of M clusters (each associated with a rule output), and let every pattern of the training set be assigned randomly to one of these clusters. Then, the clustering procedure is fundamentally an alternating optimization technique that aims at identifying the M cluster prototypes [44]. At the end of the iterations, a label q, $1 \leq q \leq M$, is associated with each pattern, representing the rule it has been assigned during the last iterations. In this way, a classification model able to assign a fuzzy label $L(\boldsymbol{x})$ to any pattern $\boldsymbol{x}$ of the input space is obtained:

$$L(\boldsymbol{x}) = [\mu_{\boldsymbol{B}^{(1)}}(\boldsymbol{x}) \;\; \mu_{\boldsymbol{B}^{(2)}}(\boldsymbol{x}) \;\ldots\; \mu_{\boldsymbol{B}^{(M)}}(\boldsymbol{x})] \tag{4.7}$$

where the k-th element of $L(\boldsymbol{x})$ represents the fuzzy membership of the pattern to the k-th class, that is the firing strength $\mu_{\boldsymbol{B}^{(k)}}(\boldsymbol{x})$ in Equation (4.6).

In the following, we will adopt a K-nearest neighbor (K-NN) strategy for classification of the fuzzy label during the prediction tests. This is done by using the input value $\boldsymbol{x}$ only, and hence, the fuzzy label of $\boldsymbol{x}$ will be:

$$L(\boldsymbol{x}) = \frac{1}{K} \sum_{q=1}^{K} L(\boldsymbol{x}_{t_q}) \tag{4.8}$$

where $\boldsymbol{x}_{t_1}, \boldsymbol{x}_{t_2}, \ldots, \boldsymbol{x}_{t_K}$ are the K patterns of the training set that score the smallest Euclidean distance from $\boldsymbol{x}$, with their fuzzy labels determined during the training phase.

The output $\tilde{y}$ is calculated through Equation (4.6) by means of the firing strengths contained in the fuzzy label $L(\boldsymbol{x})$ and the output consequents whose parameters have been previously determined by clustering in the joint input-output space. For instance, in the case of a 3rd order polynomial as adopted in the following, the k-th rule's consequent, $k = 1 \ldots M$, is:

$$\begin{aligned} y^{(k)} = \;& a_{31}^{(k)} x_1^3 + \cdots + a_{3N}^{(k)} x_N^3 + \\ & a_{21}^{(k)} x_1^2 + \cdots + a_{2N}^{(k)} x_N^2 + \\ & a_{11}^{(k)} x_1 + \cdots + a_{1N}^{(k)} x_N + a_0^{(k)} \end{aligned} \tag{4.9}$$

Regarding the generalization capability, the optimal number of rules is automatically determined on the basis of learning theory [45]. The best model is chosen by evaluating a cost function based on network complexity and

approximation error [46]. As a measure of the network performance on the training set, the mean squared error (MSE) is adopted:

$$E = \frac{1}{N_T} \sum_{t=1}^{N_T} (y_t - \tilde{y}_t)^2 \tag{4.10}$$

where N_T is the number of samples in the training set to be predicted and $\tilde{y}_t$ is the output generated by HONFIS in correspondence with the t-th input pattern $\boldsymbol{x}_t$. The optimal network is selected by minimizing the following cost function that depends on the maximum allowed number of the HONFIS rules:

$$F(M) = (1 - \lambda) \frac{E(M) - E_{\min}}{E_{\max} - E_{\min}} + \lambda \frac{M}{N_T} \tag{4.11}$$

where $E_{\min}$ and $E_{\max}$ are the extreme values of the performance E that are encountered during the analysis of the different models, while λ is a non-critical weight, $0 \leq \lambda \leq 1$, set to 0.5 by default.

4.2.5 Echo State Network

For prediction problems in which real-world time series are involved, very useful models can be found in the framework of neural network techniques [47, 48]. Most of them can achieve good results when applied to time series with many outliers and missing samples. The approach proposed herein relies on the Echo State Network model (ESN), which is a class of recurrent neural network where the 'reservoir' (recurrent part of the NN) is separated from the 'readout' (the non-recurrent part of the NN). The recurrent part is fixed and the the non-recurrent one is solved as a standard linear regression over its weights. ESNs are used in different domains for their flexibility when applied to chaotic time series prediction [49], grammatical inference [50], stock price prediction [51] and acoustic modeling problems [52], between others.

The ESN was first proposed in [53] and it pertains to the ANN paradigm, in particular to the class of recurrent neural networks (RNNs). These kind of machine learning techniques are extensively used for prediction applications, being a reliable mean for solving non-linear time series analysis [54]. In the classical RNNs, all of the internal weights are updated during training via an iterating optimization method. When the problem requires less memory, the more simple approach of the ESN can be used with fruitful results. In fact, the recurrent part of an ESN is fixed: the weights of the recurrent *reservoir* are

randomly assigned. Conversely, the output layer (*readout*, the non recurrent part) is trained based on the available data. The main advantage of this approach, which makes the ESN efficient also with large datasets, stems from the easier training problem, which is economically reduced to a straight linear regression problem.

The ESN model can be split into three different components, as illustrated in Fig. 4.1, where the boxes enclose the different parts and the lines are dashed if the relative connection is random, solid if it is trainable.

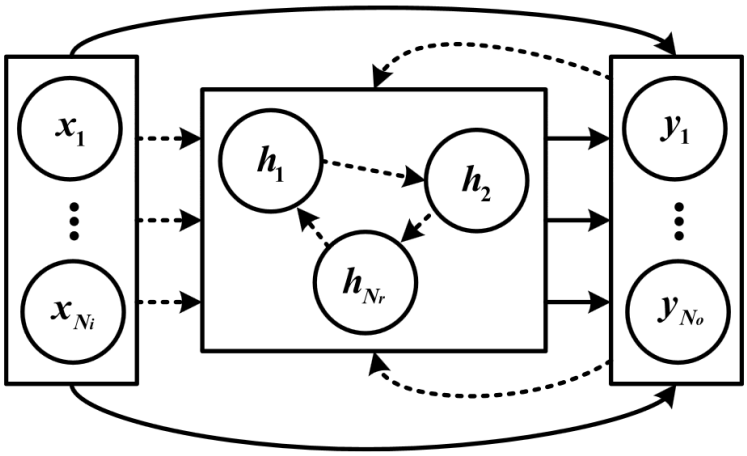


Figure 4.1: Functional scheme of an ESN with explicit different connections: dashed if they are random, solid if trainable.

The input of the network is an N_i dimensional vector $\boldsymbol{x} \in \mathbb{N}^{N_i}$ given to the reservoir with dimension of N_r. The reservoir internal state $\boldsymbol{h} \in \mathbb{N}^{N_r}$ is then updated using the following equation:

$$\boldsymbol{h}[n] = f_{\text{res}}(\boldsymbol{W}_i^r \boldsymbol{x}[n] + \boldsymbol{W}_r^r \boldsymbol{h}[n-1] + \boldsymbol{W}_o^r \boldsymbol{y}[n-1]) \,, \tag{4.12}$$

where $\boldsymbol{W}_i^r \in \mathbb{R}^{N_r \times N_i}$, $\boldsymbol{W}_r^r \in \mathbb{R}^{N_r \times N_r}$ and $\boldsymbol{W}_o^r \in \mathbb{R}^{N_r \times N_o}$ are matrices generated randomly, $f_{\text{res}}(\cdot)$ is a non-linear function defined suitably, and $\boldsymbol{y}[n-1] \in \mathbb{R}^{N_o}$ is the previous N_o-dimensional output of the network. If stability has to be increased, adding a small uniform noise to the update is a viable option. This is done before computing the non-linear transformation $f_{\text{res}}(\cdot)$ [53].

The output is then computed as:

$$\boldsymbol{y}[n] = f_{\text{out}}(\boldsymbol{W}_i^o \boldsymbol{x}[n] + \boldsymbol{W}_r^o \boldsymbol{h}[n]) \,, \tag{4.13}$$

where $\boldsymbol{W}_i^o \in \mathbb{R}^{N_o \times N_i}, \boldsymbol{W}_r^o \in \mathbb{R}^{N_o \times N_r}$ are modified and adapted in accordance with the training data, and $f_{\text{out}}(\cdot)$ is an non-linear, invertible function.

Given a time series $S[n]$, $n > 0$, to be predicted, the input $\boldsymbol{x}$ to the ESN is a vector of N_i consecutive past samples of the time series while the desired output d is the sample to be predicted; more precisely:

$$\begin{aligned} &\boldsymbol{x}[n - N_i + 1] = \Big[S[n]\ S[n-1]\ \cdots\ S[n - N_i + 1]\Big]^T, \\ &d[n - N_i + 1] = S[n + m]\,, \\ &n = N_i,\ \ N_i + 1,\ \ \ldots,\ \ Q + N_i - 1\,, \end{aligned} \tag{4.14}$$

where $m > 0$ is the prediction distance, which is usually set to $m = 1$, and the available samples of the time series should range from $S[1]$ up to $S[Q + N_i + m - 1]$.

4.3 Embedding

In the following, the generalized embedding approach is presented for which, at the generic time n, the input to the predictor can be any combination of past available samples, which are used to predict the next one $S[n + m]$. As the relation between predicted and past samples usually is not given, in a prediction problem it is important to study which samples of the sequence are relevant to the prediction task. The naive approach resides in assuming that all the past samples of the sequence to be predicted are equally important, without finding an optimal subset. However, a more fruitful approach consists in selecting the samples via embedding of the sequence [55]. The sequence generated by a complex, often chaotic and noisy system like the PV under exam can be observed by its output only. So, in order to reconstruct the evolution of the system, the original sequence $S[n]$ must be embedded. This is done by determining two parameters [56]: the embedding dimension D and the time lag T. In such a way the reconstructed state at time n can be found as:

$$\overline{\boldsymbol{S}}[n] = \Big[S[n]\ S[n - T]\ \ldots\ S[n - T(D-1)]\Big]^T. \tag{4.15}$$

By setting $\boldsymbol{x}_n = \overline{\boldsymbol{S}}(n)$, it is evident that the regressor input is an estimate of the unfolded, reconstructed state of the unknown system that generates the observed time series. By the way, looking at (4.4.1), it becomes evident

that in the MA case it is implicitly assumed $T = 1$. By driving the function approximation model with embedding data, neural networks and fuzzy inference systems are nonlinear approximation models that extend the linear one adopted in (4.4.1) and result very suited to prediction tasks on real-world time series. In a more general approach, the embedding parameters can be estimated using the Average Mutual Information (AMI) procedure for T and the False Nearest Neighbors (FNN) procedure for D, respectively [57].

4.4 Applications

4.4.1 Energy market price prediction

Introduction

Over the last two decades, all the commodities that are traded internationally among many different stockholders have played an ever more pervasive role making several authors to envisage a future invasive commodified economy [58]. Generally, all the commodity prices are basically determined by supply and demand, but they can be strongly influenced by several external irregular events, e.g., stock levels, macroeconomic factors, political aspects, monetary policy, psychological expectations [59]. Forecasting commodity prices is an essential tool in modern competitive markets [60, 61] but it can be extremely challenging when dealing with commodities characterized by intrinsic difficulties and high volatility [62], which make the time series of past prices highly nonlinear and nonstationary.

Since the late Nineties, with the deregulation process of the energy market that has moved from a centralized operational approach to a competitive wholesale spot market [63], forecasting energy prices has become a valuable tool and emerged as a hot research field in electrical engineering [64, 65]. Nowadays, any company that trades energy, especially in the day-ahead market but also in the intra-day market [32], needs an accurate price prediction mainly for twofold reasons [66]: on the one hand, the forecast of the market clearing price (MCP) helps producers or consumers to prepare an effective offering or bidding strategy that maximize benefits; on the other hand, when recurring to bilateral contracts, a good knowledge of expected market pool prices helps to hedge against volatility.

A huge number of forecasting techniques have been proposed in literature

to predict electricity prices and extensive reviews are available in [67, 68]. It is quite impossible to cite all of the approaches that have been developed for analyzing and forecasting energy prices; further details in this regard can be found in [67]. Generally speaking, these approaches can be summarized in five different classes: multi-agent, fundamental, reduced-form, statistical, and computational intelligence models. All the methods present strengths and deficiencies that must be seriously considered when applied to the energy market that is affected by a high percentage of unusual prices as all of the non-perfect oligopolistic markets.

In this section, we analyze three techniques based on neural and fuzzy neural networks for forecasting electricity prices in the day-ahead market, namely: Radial Basis Functions (RBF) neural networks [69], Mixture of Gaussians (MoG) neural network [70] and Higher-Order Neuro-Fuzzy Inference System (HONFIS) [71]. The present approach leans on the possibility of considering a prediction problem as a data regression or classification one, which can be solved by using neural models and/or fuzzy inference systems [40, 72], by means of a suitable transformation of input data and, possibly, using ad-hoc input selection techniques [73]. Throughout several case studies, the illustration of their validity in terms of prediction performance and model accuracy is carried out and the applicability of these approaches to the forecasting of energy prices is assessed.

Regarding the forecasting in day-ahead markets, the auction system in place in Italy must be explained. The day-ahead auction for hourly delivery in the six Italian zones takes place every day at 12:00 noon. Until then, starting from 8:00 of the eighth day before, consumers and producers that participate to market make proposal to purchase (bids) and sell (offers) a certain amount of electric energy at some prices. Market participants can anonymously submit their bids/offers with a minimum volume of 1 kWh for individual hours and blocks and a minimum price change of 0.01 €/MWh. Moreover, prices for hourly contracts must remain within the range between ±3000 €/MWh. After all bids have been collected, based on a System Marginal Price model, the market clearing price, which applies to all transactions, is determined and published after 12:40 pm. The delivery takes place during the respective hour of the following day and hence, this is the reason of the term 'day-ahead'. Italy's national single price (PUN) is the average of the six different zonal prices in the day-ahead market, weighted for total purchases and net of purchases for pumped-storage units and by neighboring countries. In this

framework, the importance of an accurate forecasting of the PUN for properly tailor energy offers and bids is clear and evident.

Once the context is clear, we can now approach the forecasting technique used here. Given a time series $S(n)$, $n > 0$, the prediction of a future sample $S(n+m)$ at distance m is generally obtained by solving a function approximation problem, that is by estimating the parameters $\boldsymbol{\theta}$ of a regression model $y = f(\boldsymbol{x}; \boldsymbol{\theta})$, $f : \mathbb{R}^D \to \mathbb{R}$, where $\boldsymbol{x}$ is an input vector whose D elements are chosen among the previous samples of the time series observed so far.

For instance, a standard Dth-order moving average (MA) predictor is based on a linear regression model and each input is made of D subsequent samples of $S(n)$:

$$\begin{aligned} \boldsymbol{x}_n &= \left[S(n)\ S(n-1) \cdots S(n-D+1)\right], \\ y_n &= S(n+m)\ , \\ f_{\text{lin}}(\boldsymbol{x}_n; \boldsymbol{\theta}) &= \sum_{j=1}^{D} \theta_j x_{nj}\ , \end{aligned} \tag{4.16}$$

and hence, we have:

$$\widetilde{S}(n+m) = \sum_{j=1}^{D} \theta_j S(n-j+1)\ , \tag{4.17}$$

where $\widetilde{S}(n+m)$ is the estimate of the predicted sample and parameters $\boldsymbol{\theta}$ can be determined by standard estimation techniques, usually considering the statistical properties of the observed time series.

In the following, we propose to use a generalized approach for embedding the time series, considering the specific behavior that is relevant to the production of energy commodities and the related market prices [55]. A time series generated by a complex, often chaotic and noisy market system, like in the case of PUNs under exam, can be observed by its output only. So, in order to reconstruct the evolution of the system, the original sequence $S(n)$ must be embedded so as to reconstruct the evolution of the underlying, unknown market model. The technique used here is the same described in Sec. 4.3. In the following, we analyze the performance of three techniques based on neural and fuzzy neural networks, all already detailed in Sec. 4.2:

- RBF: a neural network that builds up a function approximation model with a usually multiquadratic radial basis function [69];

- MoG: a neural network model in which a density mixture of Gaussian components are used in the joint input-output space; the mixture parameters are estimated via the Expectation-Maximization algorithm and constructive learning as proposed in [70];
- HONFIS: a neuro-fuzzy inference system based on a set of 'higher-than-one order' rules of Takagi-Sugeno type [71].

Table 4.1: Prediction Results (NMSE) for July 1st

Prediction Model	Training Set (# days after the latest sample and before the test set)								
	1 day	2 days	3 days	4 days	5 days	6 days	7 days	8 days	9 days
LSE training	0.0201	0.0204	0.0198	0.0212	0.0213	0.0220	*0.0189*	0.0197	0.0241
LSE test	1.3229	1.2223	1.9396	1.7311	0.8341	1.5018	1.3010	2.3254	1.4010
MoG training	0.0153	0.0158	0.0160	0.0166	0.0158	0.0174	0.0163	0.0186	0.0252
MoG test	**0.5721**	1.4201	1.2939	3.2144	0.5231	6.1050	1.2001	2.1191	1.3034
RBF training	0.0472	0.0497	0.0493	0.0479	0.0470	0.0467	00.0479	0.0486	0.0508
RBF test	0.1955	**0.1444**	0.3041	0.3981	0.3132	0.5639	0.5434	0.8395	0.4653
HONFIS training	0.1001	0.1076	0.1119	0.0982	0.0889	0.0849	0.1135	0.1023	0.1172
HONFIS test	0.1934	0.3543	3.0712	0.9914	0.5827	0.7099	0.4809	0.7076	**0.1887**

Table 4.2: Prediction Results (NMSE) for December 1st

Prediction Model	Training Set (# days after the latest sample and before the test set)								
	1 day	2 days	3 days	4 days	5 days	6 days	7 days	8 days	9 days
LSE training	0.0355	0.0398	0.0271	0.0364	0.0584	0.0765	0.0134	0.0254	0.0369
LSE test	0.2001	0.3751	0.3642	1.6984	0.3224	0.2954	0.7700	2.3335	0.7465
MoG training	0.0048	0.0056	0.0046	0.0035	0.0037	0.0043	0.0043	0.0038	0.0038
MoG test	**0.3898**	2.4182	1.8551	1.7290	1.4141	2.2025	10.328	9.1297	1.3567
RBF training	0.0151	0.0144	0.0141	0.0119	0.0103	0.0116	0.0116	0.0129	0.0128
RBF test	0.1981	0.4543	0.9945	1.1542	**0.1915**	0.2860	0.6601	4.0883	0.6612
HONFIS training	0.0911	0.0898	0.0878	0.0851	0.0836	0.0833	0.0835	0.0871	0.0873
HONFIS test	**0.2239**	1.2202	2.3941	1.1652	0.7782	3.4035	11.632	7.5510	0.3277

To validate the described approach, some numerical experiments were investigated. The dataset used for these experiments is associated with a real PUN time series belonging to the year 2016. The original sequence is sampled every 15 minutes but it is downsampled at one sample per hour for reducing the computational cost. A sample interval of this time series is shown in Fig. 4.2.

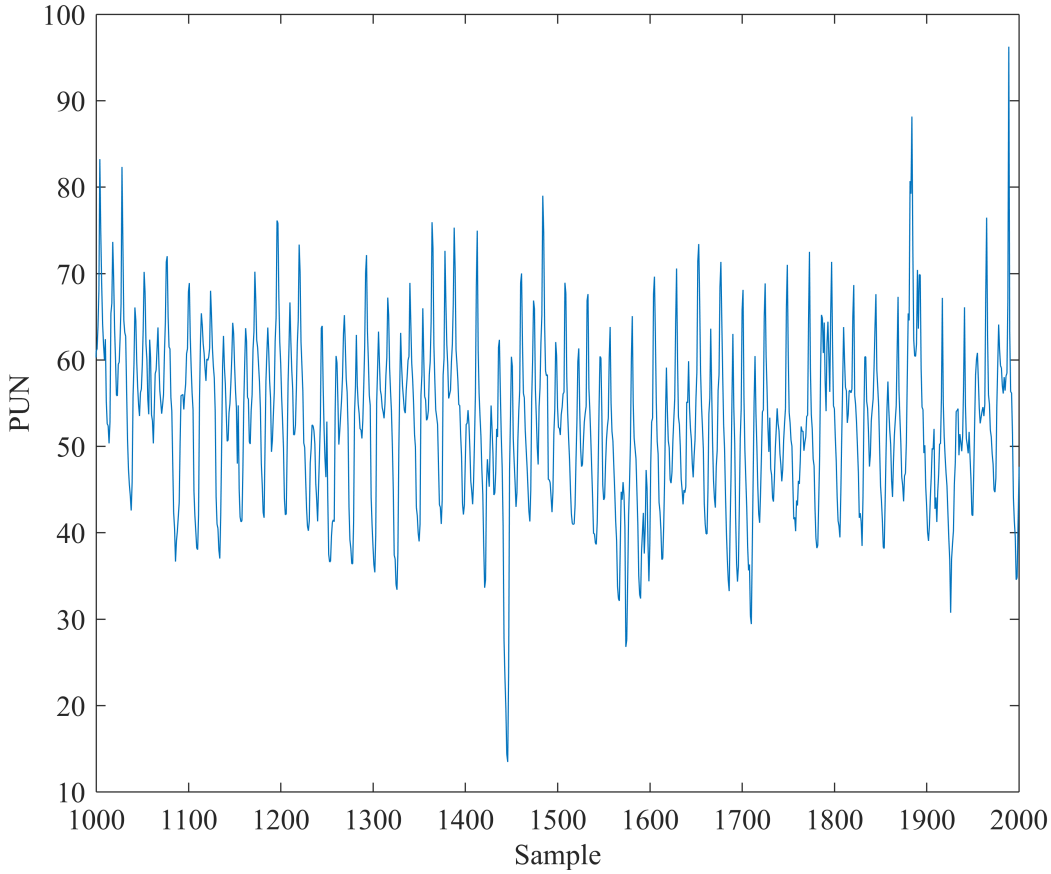


Figure 4.2: A sample of the considered PUN time series in 2016.

Two sets of experiments are considered, with two different days of prediction: the 1st of July and the 1st of December. These days are chosen because they are very representative in terms of price variation. The test set is therefore composed by the 24 samples (1 sample per hour) for either one of these two days. Nine different training sets are considered for a same test. Each training set is always composed by 720 samples (related to 30 days) but referring to 9 different intervals, that is the last day of the training set is 1 day up to 9 days before the day of the test set. This reflects the different cases of price prediction, one for every of the 9 days prior to the ending of the transactions. The embedding parameters are found by using the AMI and FNN algorithms: for the July case, we will adopt $T = 1$ and $D = 9$ for every training set; for the 1st of December, we will adopt $T = 1$ and $D = 27$ for every training set. This difference is due to the characteristics of the training sets related to different seasons and it should not be considered as an incoherence, as in the operational phase the embedding parameters can be estimated at any given time.

The performance index of prediction accuracy is the Normalized Mean Squared Error (NMSE), which is defined as:

$$\text{NMSE} = \frac{\sum_n \left[S(n) - \widetilde{S}(n)\right]^2}{\sum_n \left[S(n) - \bar{S}\right]^2}, \tag{4.18}$$

where $\bar{S}$ is the average of the PUN time series over the 24 samples of each test set. The numerical results are reported in Table 4.1 and Table 4.2 for July 1st and December 1st, respectively, considering the nine different training sets (i.e., 1 day to 9 days before) and they are compared with the performance of the naive LSE for benchmarking purposes. The best performance on the test set versus the different training condition is reported for each predictor in bold.

The best prediction model in terms of stable performance over the different trading conditions (i.e., different training sets) is evidently the HONFIS one. While MoG exhibits good performances but it is affected by large errors in some cases, likely due to the fast changing of the time series, the RBF is the less suitable one for prediction, as it achieves the highest NMSE values on the test set.

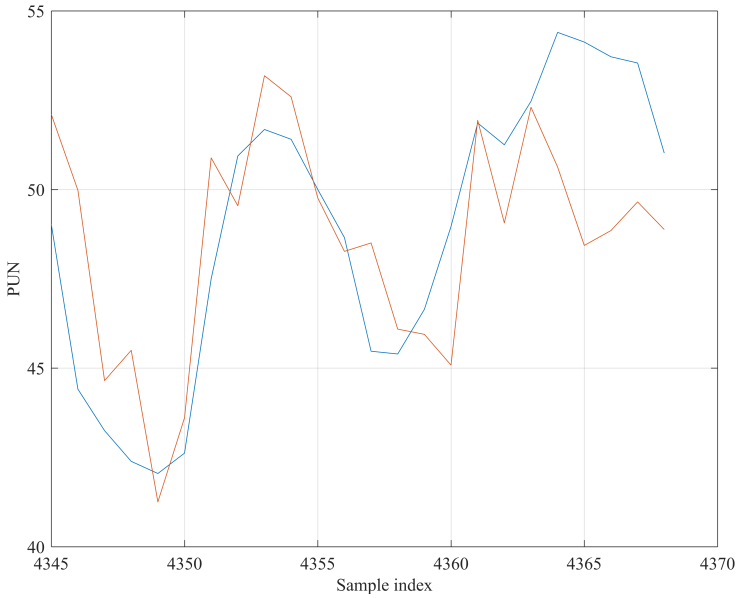


Figure 4.3: Best prediction on the 1st of July test set for the MoG predictor (1-day training set): actual time series (blue); predicted one (red).

It should be noted that the NMSE is a normalized L_2-norm, which is purely indicative of the general prediction performance. Further details on

the prediction capability can be captured by visual inspection looking at resulting plots. For the sake of illustration, the best predictions for the 1st of July are shown in Figs. 4.3-4.5 and the best ones for the 1st of December in Figs. 4.6-4.8, respectively.

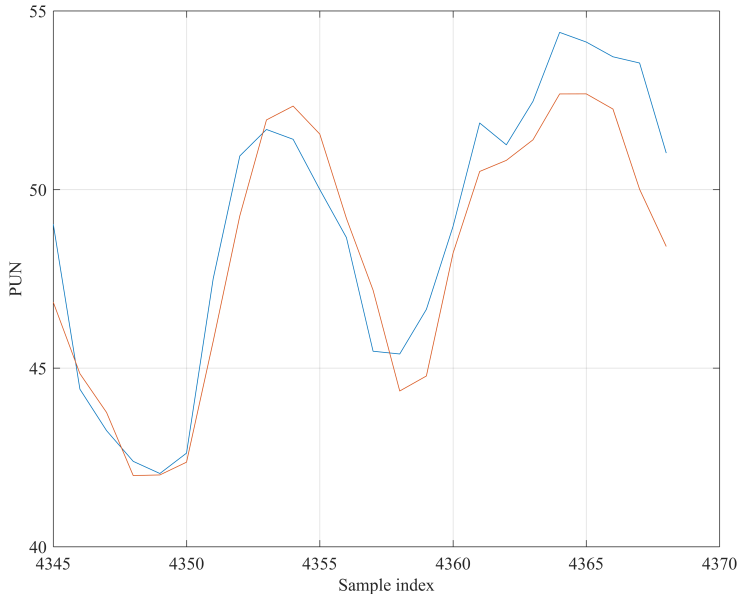


Figure 4.4: Best prediction on the 1st of July test set for the RBF predictor (1-day training set): actual time series (blue); predicted one (red).

From the graphical results, it is clear that the variation of the training set has a non negligible effect on the HONFIS model. Actually, this model is very sensitive to the setting of parameters of its learning algorithm and so, it should be re-learned at every iteration. It can also be noted that the performance is generally worst for December. This can be explained with a different volatility of prices in the related training sets.

4.4.2 Energy Production

This section presents embedding approaches based on neural and fuzzy neural networks that have been properly tailored to be efficiently applied to PV time series prediction. The models were applied to the time series of sampled output current of a test PV plant, predicting a single day of operation. The models were trained using different training sets, namely 1-day, 3-day, 7-day or 30-day. The obtained results demonstrate that in normal operation conditions, all of the solutions are suitable for the proposed goal, except for the

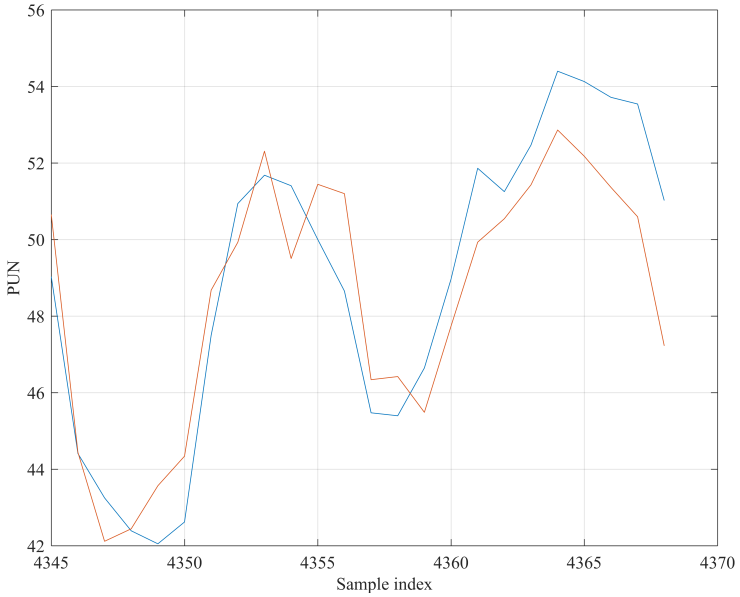


Figure 4.5: Best prediction on the 1st of July test set for the HONFIS predictor (1-day training set): actual time series (blue); predicted one (red).

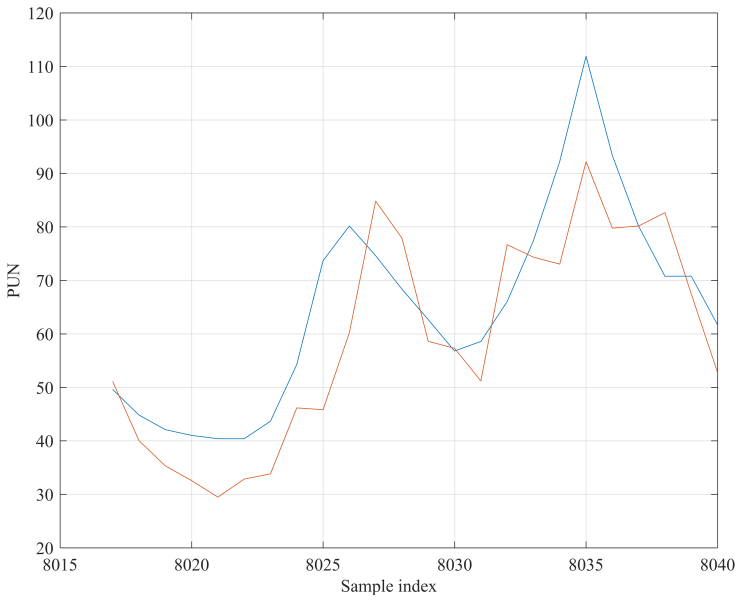


Figure 4.6: Best prediction on the 1st of December test set for the MoG predictor (1-day training set): actual time series (blue); predicted one (red).

shorter training sets, which result in less reliable performances. In particular, the approaches were shown to be very accurate when considering a seven-day training set to forecast the successive day, mostly because in a single-week window, the solar irradiation is more stable.

A novel approach is presented in this section, where fuzzy neural networks

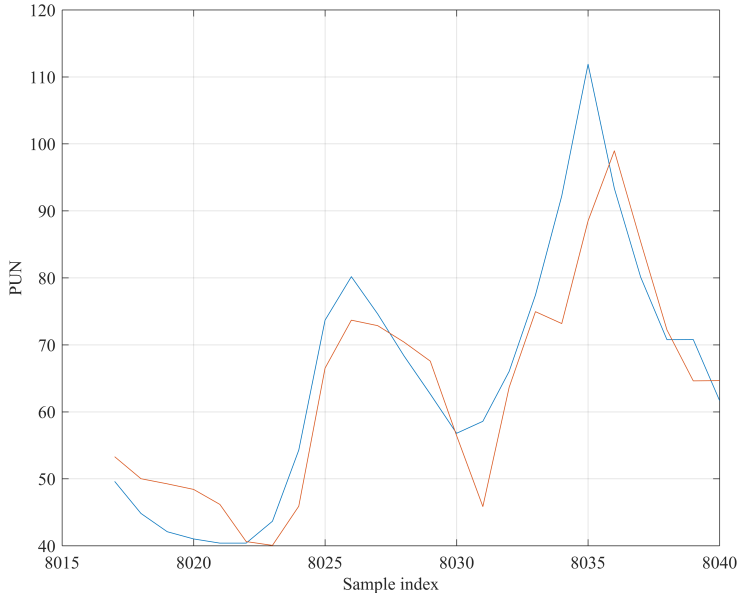


Figure 4.7: Best prediction on the 1st of December test set for the RBF predictor (1-day training set): actual time series (blue); predicted one (red).

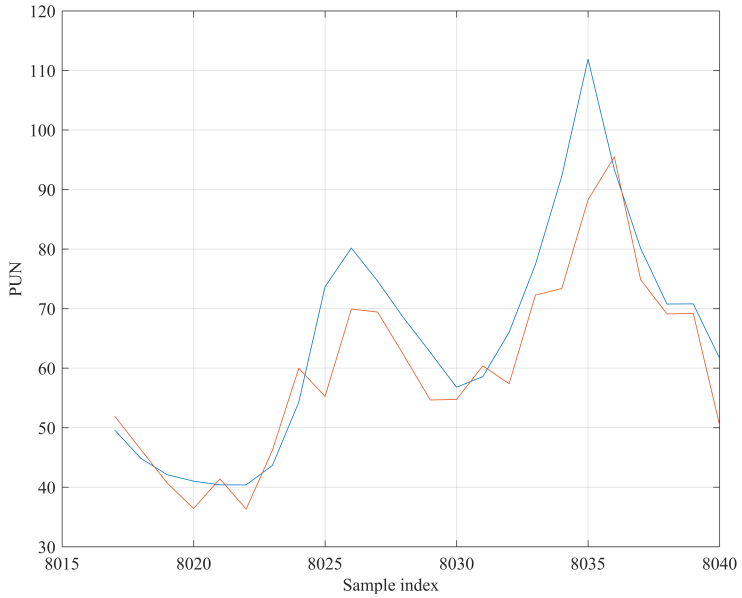


Figure 4.8: Best prediction on the 1st of December test set for the HONFIS predictor (1-day training set): actual time series (blue); predicted one (red).

are considered as a predictive tool in the strategic field of photovoltaic power plants prediction. The complexity and dynamics as a real-world problem require advanced methods and tools able to use past samples of the sequence in order to make an accurate prediction of the future ones. In this regard, computational intelligence is considered as one of the most fruitful approaches for prediction and for photovoltaic applications too [47]. In effect, several

forecasting methods have been proposed such as fuzzy predictors [74], artificial neural networks [48,75], evolutionary and genetic algorithms [76], and support vector machines [77]. Nevertheless, the lack of robustness against outliers and having to deal with noisy and missing training examples are still an open problem.

Here, a new learning approach of Takagi-Sugeno (TS) fuzzy inference systems is presented to solve the problem of voltage prediction in photovoltaic plants. The parameters of each rule are obtained through a clustering synthesis in the joint input-output space and a computationally efficient optimization is also reported. This optimization is based on a constructive procedure, by which the number of rules is progressively increased. The optimal one is then automatically determined on the basis of learning theory in order to maximize the generalization capability of the resulting neural network. The proposed approach is applied in the case of well-known photovoltaic time series. The application is ascertained by several benchmark results and extensive computer simulations, which prove the validity of the proposed learning procedure and show a favorable comparison with other well-known prediction techniques.

Complexity and dynamics of real-world problems require advanced methods and tools able to use past samples of the sequence in order to make an accurate prediction. Additionally, the problem of forecasting future values of a time series is often mandatory to the cost-effective management of available resources. These are well-known hard problems, given the non-stationary and non linear characteristics of the time series. Because of these characteristics, the resulting dynamics is hard to be adequately modeled by using standard predictive models. Unfortunately, standard structural models cited in Sec. 3.2 provide a poor representation of actual data and therefore result in a poor accuracy when they are used for forecasting. Consequently, many worldwide research activities are intended to improve the accuracy of prediction models. In this regard, computational intelligence is considered as one of the most fruitful approaches for prediction. Several forecasting methods, with different mathematical backgrounds, such as fuzzy predictors [74], artificial neural networks [75], evolutionary and genetic algorithms [76], and support vector machines [77] have been resorted. Nevertheless, the problem to deal with noisy and missing training examples, and the lack of robustness against outliers, are still open problems. For this reason, there are several approaches in the technical literature that consider Takagi-Sugeno (TS) fuzzy

models for time series prediction [78–82]. A huge amount of literature is available on short-term forecasting models applied to PV plants, even if most of them deal with irradiation forecasting models: while not claiming to be exhaustive, the proposed techniques include mainly statistical models [83], stochastic predictors [84], fuzzy systems [85], ANN [86], ANN in conjunction with statistical feature parameters [87] and artificial intelligence (AI)-based techniques [88]. Some papers present also the comparison between the predictions obtained though different models based on two or more forecasting techniques [89]. Despite the large amount of literature on forecasting techniques, only a few papers describe forecasting models to be used for directly predicting the daily energy production of the PV plant. These few papers use general techniques appropriately applied to the day-head horizon. Starting from the input of past power measurements and meteorological forecasts of solar irradiance, the power output is estimated by means of ANN [90] or soft computing [91], while in [92], the ANN is applied only on past energy production values. Additionally, the ANN can be used in conjunction with numerical weather prediction (NWP) models [93], which can be based on satellite and land-based sky imaging [94]. NWP models have been also used to build an outperforming multi-model ensemble (MME) for day-ahead prediction of PV power generation [95]. More complex schemes are proposed in [96], where an ANN is used to improve the performance of baseline prediction models, i.e., a physical deterministic model based on cloud tracking techniques, an ARMA model and a k-nearest neighbor (kNN) model, and in [97], where a Kalman filter is used in conjunction with a state-space model (SSM).

The use of inference systems for data regression based on neural and fuzzy neural networks is proposed here. The main distinguishing idea of this work is that a new learning approach is properly tailored to analyze data associated with renewable power sources starting from available techniques based on computational intelligence that have proven to produce very accurate predictions in several fields of application, like for example, risk management, biomedical engineering and financial forecasting [48], but also to analyze accurately the data associated with renewable power sources [98]. Actually, the stochastic and often chaotic behavior of time series in the prediction of output power of PV plants calls for ad hoc procedures to forecast power time series, which are here developed working only on the past measurements of electric quantities, as initially tested in our previous works [55].

Here, a prediction problem is brought in as a two-fold process. First,

the observed time series is embedded (see Sec.4.3) in order to select, among the available samples, the ones to be used to predict the next one. Then, such samples are used to feed different function approximation models and techniques based on neural and fuzzy neural networks, which are suited to predict data sequences that often show a chaotic behavior: radial basis function (RBF) [99]; Mixture of Gaussian NN; the adaptive neuro-fuzzy inference system (ANFIS) [100]; the higher-order neuro-fuzzy inference system (HONFIS) [101]. In fact, a predictor can be based on such function approximation models, whose parameters are estimated by data-driven learning procedures.

Takagi-Sugeno Fuzzy Systems Applied to Energy Time Series Prediction

To explain the technique used, let us consider a time series $S(n)$, the input vector $\mathbf{x}_n$ of a data regression model to be used for prediction is determined through the so-called 'embedding technique', which is based on past samples of the time series $S(n)$:

$$\mathbf{x}_n = [S(n)\, S(n-T)\, S(n-2T) \ldots S(n-(D-1)T)]\,, \tag{4.19}$$

where D is called the 'embedding dimension' and T is the 'time lag' between past samples of $S(n)$. The estimated sample $\widetilde{S}(n+m)$ predicted at a distance m will be:

$$\widetilde{S}(n+m) = f\left(\mathbf{x}_n\right)\,, \tag{4.20}$$

where $f(\cdot)$ is the regression model to be determined. In the following, we will consider the usual case of a 'one-step-ahead' prediction, that is $m = 1$.

In this regard, the model adopted in this section for data regression is a generalization of the ANFIS in Sec. 4.2.3 and it is based on a TS fuzzy system made of C different fuzzy rules [102], as in Sec. 4.2.4. Generally, the kth rule, $k = 1 \ldots C$, of a TS system has the following form:

$$\begin{aligned} &\text{If } x_1 \text{ is } B_1^{(k)} \text{ and } \ldots \text{ and } x_D \text{ is } B_D^{(k)} \text{ then} \\ &y^{(k)} = h\left(\mathbf{x}; \omega^{(k)}\right)\,, \end{aligned} \tag{4.21}$$

where $\mathbf{x} = [x_1\ x_2\ \ldots\ x_D]$ is a row vector (or pattern) in the D-dimensional (embedded) input space and $y^{(k)}$ is the scalar output associated with the rule.

The latter is characterized by the membership functions (MFs) $\mu_{B_j^{(k)}}(x_j)$ of the fuzzy input variables $B_j^{(k)}$, $j = 1 \dots D$, and the set of parameters $\omega^{(k)}$ of the related crisp output functions in the consequent parts.

Several alternatives are possible for the fuzzification of crisp inputs, the composition of input MFs, and the way rule outputs are combined [103]. Usually, the regression structure of the TS inference system is the following one:

$$\tilde{y} = f(\mathbf{x}) = \frac{\sum_{k=1}^{C} \mu_{\mathbf{B}^{(k)}}(\mathbf{x})\, y^{(k)}}{\sum_{k=1}^{C} \mu_{\mathbf{B}^{(k)}}(\mathbf{x})}\,, \tag{4.22}$$

where $\mathbf{B}^{(k)}$ is the overall fuzzy input variable, $\mu_{\mathbf{B}^{(k)}}(\mathbf{x})$ is the value of corresponding MF (i.e., the 'firing strength'), and $\tilde{y}$ is the output estimated for any input $\mathbf{x}$.

In this work, a training set of N_T input-output pairs $\{\mathbf{x}_t, y_t\}$, $t = 1 \dots N_T$, will be considered, where $\mathbf{x}_t$ is an embedded vector obtained as in (4.19) and y_t is the related sample to be predicted. The rule parameters of the TS system will be estimated through a data-driven learning approach; the main problem during this phase is to obtain a good generalization capability. In this case, the latter is maximized only if the TS system consists of a suitable number of rules. However, the determination of the optimal number is a very critical problem to be solved as the inference system might be easily overfitted in case of noisy or ill-conditioned data. Here, a constructive procedure for the automatic determination of rules is also reported. It aims at regularizing the network architecture by using learning theory and clustering in the joint input-output space [45, 104–106].

Subsequently, we explain here the procedure for determination of the rules, with preliminary tests on some energy dataset. We do this to ensure that the procedure is sound and reliable for further use on real-world applications such as the prediction problem analysed here. We propose to determine the parameters of the TS rules by using a clustering procedure in the joint input-output data space. The procedure is fundamentally an alternating optimization technique that aims at identifying the cluster prototypes [44]. Let $\Gamma = \{\Gamma_1, \Gamma_2, \dots, \Gamma_C\}$ be a set of C clusters (each associated with a rule output) and let every pattern of the training set be assigned randomly to one of these clusters. Then, the clustering procedure with C prototypes is based on

the following iterative steps:

- *Step 1*. The coefficients $\omega^{(k)}$, $k = 1 \dots C$, of each rule consequent are evaluated by solving a set of (generally) nonlinear equations; the generic equation is:

$$y_t = h\left(\mathbf{x}_t; \omega^{(k)}\right) , \tag{4.23}$$

  where y_t is the output associated with the input $\mathbf{x}_t$; index 't' spans only the pairs of the training set assigned to the kth cluster. In this regard, the set of nonlinear equations in (4.23) are solved using an iterative least squares estimation [107].

- *Step 2*. Each pair $(\mathbf{x}_t, y_t)$, $t = 1 \dots N_T$, of the training set is now assigned to the cluster Γ_q, $1 \leq q \leq C$, having the minimum distance from it:

$$\begin{aligned} d_t &= \left| y_t - h\left(\mathbf{x}_t; \omega^{(q)}\right)\right| = \\ &= \min_{k=1\dots C} \left| y_t - h\left(\mathbf{x}_t; \omega^{(k)}\right)\right| . \end{aligned} \tag{4.24}$$

- *Step 3*. For every cluster Γ_k, $k = 1 \dots C$, the *local* approximation error is evaluated:

$$D_k = \frac{1}{N_k} \sum\nolimits_t d_t \, , \tag{4.25}$$

  where index 't' spans only the N_k pairs of the training set assigned (in Step 2) to the kth cluster.

- *Step 4*. The convergence is based on the quantity:

$$\Theta = \frac{\left| D - D^{(old)} \right|}{D^{(old)}} \, , \tag{4.26}$$

  where D is the *global* approximation error over the whole dataset in the current iteration defined by:

$$D = \frac{1}{N_T} \sum_{t=1}^{N_T} d_i \, , \tag{4.27}$$

and $D^{(old)}$ is the global approximation error calculated in the previous iteration. The algorithm is stopped when Θ is less than a threshold θ (set by default to 0.01). If it is greater, it goes back to Step 1 by using the current updated association of patterns to clusters.

The proposed clustering procedure is able to provide the consequents only of the Sugeno-type fuzzy rules, but we need also the firing strengths of the rules' antecedents in order to obtain a complete structure of the TS system. In this regard, at the end of the iterations, a label q ($1 \leq q \leq C$) is associated with each pattern, representing the rule it has been assigned to during Step 2 of the last iteration. In this way, a classification model able to assign a fuzzy label $L(\boldsymbol{x})$ to any pattern $\mathbf{x}$ of the input space is obtained:

$$L(\mathbf{x}) = [\mu_{\mathbf{B}^{(1)}}(\mathbf{x}) \;\; \mu_{\mathbf{B}^{(2)}}(\mathbf{x}) \; \ldots \; \mu_{\mathbf{B}^{(M)}}(\mathbf{x})] \;, \tag{4.28}$$

where the kth element of $L(\mathbf{x})$ represents the fuzzy membership of the pattern to the kth class and hence, it can be assumed as the firing strength $\mu_{\mathbf{B}^{(k)}}(\mathbf{x})$ in (4.22) of the kth rule associated with the model $h\left(\cdot;\omega^{(k)}\right)$ corresponding to that class.

When the output $\tilde{y}$ must be estimated for any input $\mathbf{x}$ during the normal TS operation (i.e., testing), the classifier is used to determine the fuzzy label (4.28) by using the input value $\mathbf{x}$ only; then, the output $\tilde{y}$ is calculated through (4.22) by means of the firing strengths contained in the fuzzy label $L(\mathbf{x})$ and the output consequents in (4.21), whose parameters have been early determined by clustering in the joint input-output space. In the following, we will adopt a K-nearest neighbor (K-NN) strategy for classification. Namely, let $\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \ldots, \mathbf{x}_{t_K}$ be the K patterns of the training set that score the smallest Euclidean distance from $\mathbf{x}$; then, the fuzzy label of $\mathbf{x}$ will be:

$$L(\mathbf{x}) = \frac{1}{K} \sum_{q=1}^{K} L(\mathbf{x}_{t_q}) \,. \tag{4.29}$$

Preliminary tests on the approximation capability of different orders of the Takagi Sugeno system

One of the core issues of relying on the proposed HONFIS method, is the optimization of the generalization capability. The training of TS systems poses two problems: the right computation of the number of rules (C) and

the local convergence of the clustering algorithm for the synthesis of the rules. The former problem can be viewed as a 'structural optimization' problem and the underfitting or overfitting of the training set can cause degradation in the TS performance. The latter is linked to the goodness of the random initialization of the parameters of each rule[1]. Here, we propose to use different values of C and different initialization for every value of C. The best TS system is chosen by evaluating a cost function based on network complexity and approximation error [46, 103].

A set of experiments was put in place to preliminarly assess the performance of the described methods and to choose the best order for the HONFIS. The performance of the proposed approach is evaluated considering different orders for the polynomial function $h(\cdot)$ of the TS rule consequent. We studied five cases, that is from first-order to fifth-order functions. Tests were carried out using data from the De Nittis Power Plant in the Apulia Region, Italy (latitude $\phi = 41°26'16''$, longitude $\lambda = 15°45'47''$). Data is relative to a single photovoltaic plant, organized with eight cabins with two inverters each. The output current was used as the variable to be predicted. The value was sampled at the source with a 5-min sample interval, and it was collected from 6:00 a.m.–10:55 p.m., resulting in 204 samples per day. The used data stream comes from a single inverter of a single cabin, and it is relative to the year 2012. In order to provide a statistical characterization of the handled time series, whose samples are measured in Amperes, we have computed the first four statistical moments of the whole 2012 dataset, whose histogram is also reported in Figure 4.9: mean 85.1291; variance 1.0244×10^4; skewness 0.8151; kurtosis 2.1082. Successively, the whole time series has been normalized linearly between zero and one in order to cope with the numerical requirements of learning algorithms, especially for neural network models.

[1]In the present case, the initialization is carried out by assigning the patterns to clusters randomly before the clustering iteration starts.

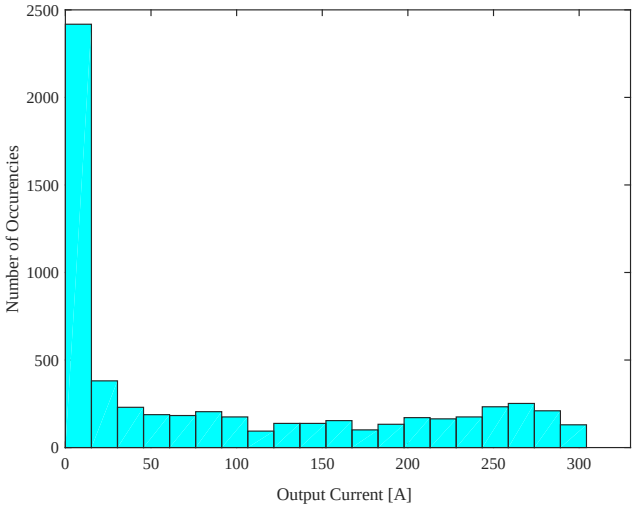


Figure 4.9: Histogram of the output current.

The results are reported in Table 4.3 in terms of NMSE only, as it is sufficient to determine the best order to be used for the next comparisons. Such results are relative to the test performed on the 30th of June as test set, with the dataset described earlier. We have chosen this day because of the stability of the sequence in terms of meteorological condition and irradiation. The best results are given by third-order and fourth-order functions for the 7-days and 30-days training set, respectively.

Table 4.3: Prediction Results (NMSE) for Different TS Orders of The Rule Consequent

TS Rule	7-days	30-days
First-order training	4.043	1.636
First-order test	0.058	0.019
Second-order training	0.596	2.019
Second-order test	0.053	0.018
Third-order training	0.611	1.591
Third-order test	**0.052**	0.017
Fourth-order training	0.639	1.514
Fourth-order test	0.055	**0.016**
Fifth-order training	0.819	1.513
Fifth-order test	0.097	0.016

NMSE values are scaled by 10^{-2}

We illustrate in Fig. 4.10 the worst performance for 7-days training set,

obtained by the fifth-order function, and the best performance obtained, as said, by the third-order function in Fig. 4.11. Similarly, we show in Fig. 4.12 the worst performance for 30-days training set, obtained by the first-order function, and the best performance obtained in this case by the fourth-order function in Fig. 4.13.

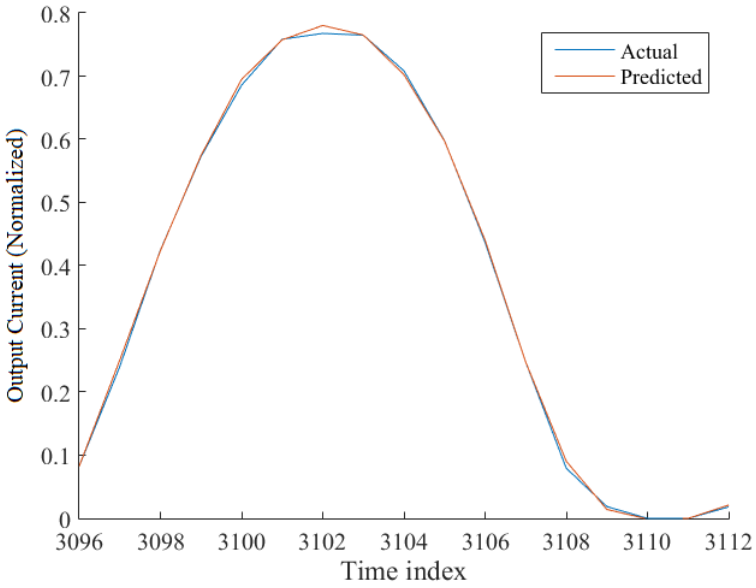


Figure 4.10: TS prediction on the 7-days training set: worst performance of fifth-order consequents.

Then, we compare the prediction performances of the proposed TS approach with respect to several well-known prediction models[2], that is:

- Linear (LSE) predictor, where the relationship between the value to be predicted and the current/previous ones is modeled by using a linear function, whose parameters are determined by least-squares techniques on training data;
- Radial Basis Function (RBF) neural network, trained as described in [69];
- Adaptive Neuro-Fuzzy Inference System (ANFIS) [102], which is trained by a subtractive clustering method for rule extraction [108]. The rule parameters are obtained by means of a standard least-squares method coupled with the back-propagation optimization [103].

[2]A cross-validation technique is adopted by using a suited early-stopping procedure in order to optimize the model complexity.

Experiments and performance comparison

Having assessed the performance and the order of the HONFIS in use, the successive experiments, herein described, will consider for testing one day for each month of 2012, and we have chosen the 15th of each month for the sake of uniformity (the last 15 samples of 2011 are also used for the 30-day training of 15 January). All of the computational models were trained using time series subsampled at an interval of 1 h, thus resulting in 17 samples per day. Four different kinds of training conditions were considered: 1-, 3-, 7- and 30-day, associated with training sets composed of 17, 51, 119 and 510 samples, respectively. Every computational model estimated by using one of these training sets was applied to test the day following the latest one in the training set, that is on a test set of 17 samples. After the preliminary analysis we carried out for determining the embedding parameters, as the latter did not show a considerable sensitivity to seasonality and to the length of the training set, which was relatively small with respect to the usual length of time series processed by the AMI and FNN methods, we had adopted as the optimal choice the values $T = 5$ and $D = 3$ for every training set and the related experiment. We remark that all of the test sets had different irradiation and meteorological conditions, and thus, they could be considered as a good

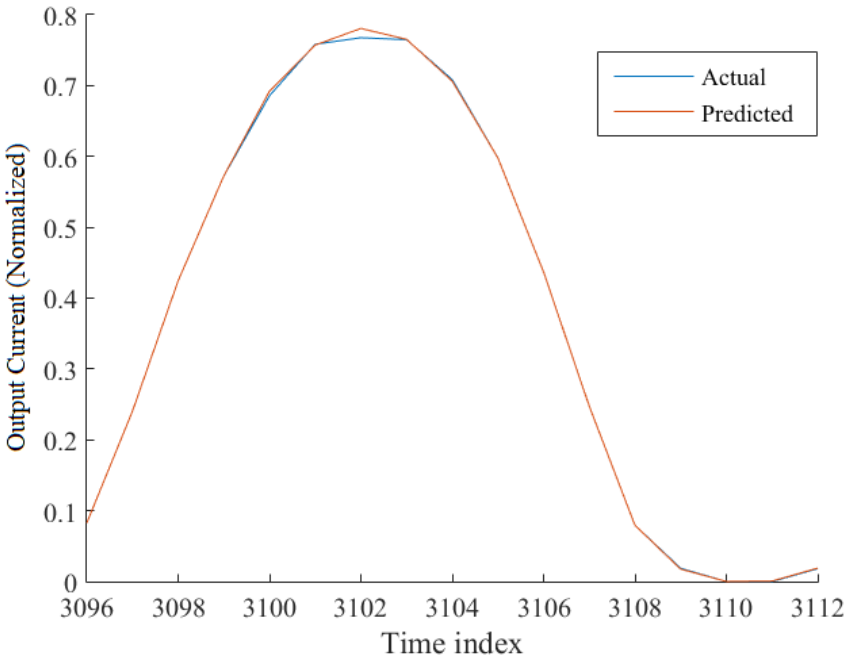


Figure 4.11: TS prediction on the 7-days training set: best performance of third-order functions.

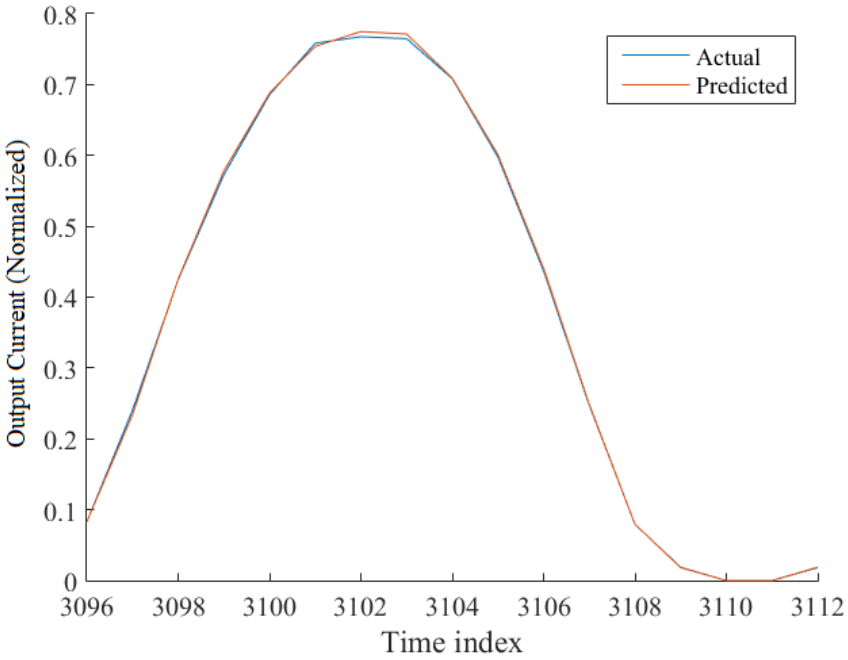


Figure 4.12: TS prediction on the 30-days training set: worst performance of first-order consequents.

ensemble for representing the typical behaviors that might be encountered.

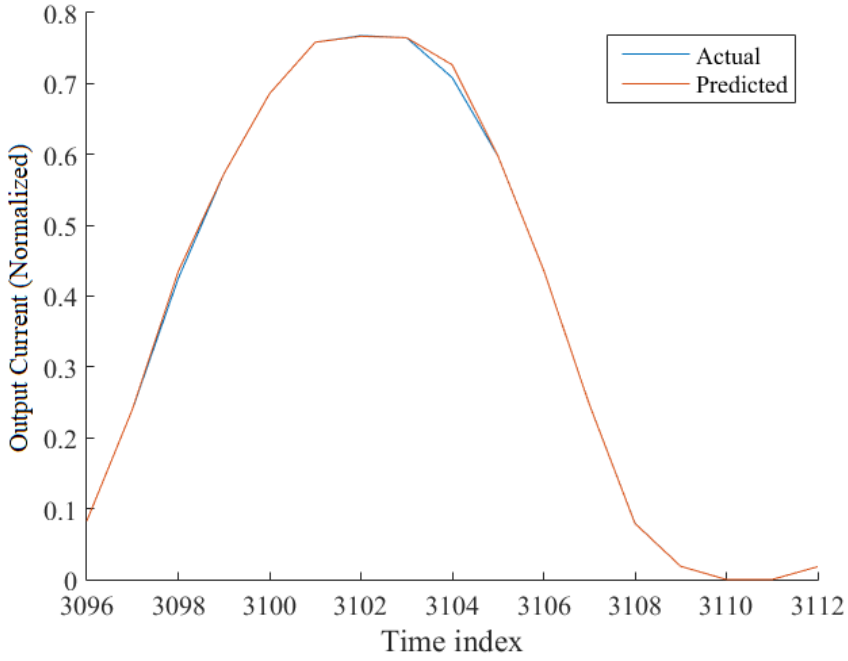


Figure 4.13: TS prediction on the 30-days training set: best performance of fifth-order functions.

All of the experiments had been performed using MATLAB® 2016b, running on a 3.1-GHz Intel Core i7 platform equipped with 16 GB of memory. In addition to the three proposed neural and fuzzy neural models, the linear (LSE), RBF and ARIMA predictors were adopted for benchmarking. LSE did not have parameters to be set in advance, while for ARIMA, RBF and ANFIS, we had adopted the default options provided by the software platform for training and model regularization (ARIMA, RBF and ANFIS models were trained by using the supported functions in the econometrics toolbox, neural network toolbox and fuzzy logic toolbox of MATLAB, respectively). In the following, the results of the third-order model are reported, as it was able to obtain the best performance in almost all of the considered cases (as per evidence previously presented). The input space classification was obtained by a three-NN classifier. The optimal number of rules was determined by Equation (4.11), using $\lambda = 0.5$ and varying M from 1–50% of N_T. The prediction performances were measured by two metrics commonly adopted for energy time series, NMSE and MARE, as cited in Sec.3.4. The numerical results for each tested day are reported in Tables 4.4–4.15, where the performance of the best model is marked in bold font. As per the following discussion, being HONFIS the model that assures the best performance in most of the situations, the graphical illustration of the actual time series (blue line) and the one predicted by HONFIS (red line), over the four training conditions, is reported in Figures 4.14–4.25, respectively. In the x-axis is reported the cumulative index of samples of the considered day, starting from Index 1 for the first sample of 1 January 2012 (which is a leap year) and considering 17 samples per day. The output current reported in the y-axis is a dimensionless value between zero and one, as the whole dataset has been normalized before the model processing. Some negative values may occur because of possible numerical issues of a trained model when its performance is inadequate.

Table 4.4: Prediction results for 15 January.

Prediction Model	**NMSE**				**MARE**			
	1-Day	**3-Day**	**7-Day**	**30-Day**	**1-Day**	**3-Day**	**7-Day**	**30-Day**
LSE training	0.4690	0.4429	0.7591	0.3523	0.0921	0.1312	0.0457	0.0884
LSE test	**0.8230**	0.4394	0.5463	0.4347	**0.1843**	0.1547	0.1542	0.1544
ARIMA training	1.9929	0.8843	0.6605	0.9852	0.3968	0.2910	0.2232	0.7345
ARIMA test	1.8852	0.5194	0.4838	0.2691	0.5822	0.2010	0.2222	0.1597
RBF training	0.1137	0.1393	0.1847	0.2901	0.0512	0.0687	0.0675	0.0801
RBF test	1.3472	0.2739	0.2959	0.4399	0.9586	0.1146	0.1134	0.1461
ANFIS training	0.0745	0.0005	0.2698	0.3383	0.0522	0.0040	0.0888	0.0910
ANFIS test	2.2393	**0.0004**	0.4415	0.5227	0.3895	0.0046	0.1597	0.1723
HONFIS (3rd ord.) training	0.0137	0.0005	0.0032	0.0650	0.0181	0.0027	0.0040	0.0174
HONFIS (3rd ord.) test	1.4820	0.0008	**0.0009**	**0.0511**	0.2373	**0.0041**	**0.0044**	**0.0212**

Table 4.5: Prediction results for 15 February.

Prediction Model	**NMSE**				**MARE**			
	1-Day	**3-Day**	**7-Day**	**30-Day**	**1-Day**	**3-Day**	**7-Day**	**30-Day**
LSE training	0.0978	0.2167	0.3480	0.2631	0.0906	0.1123	0.1355	0.1213
LSE test	0.4072	0.3690	0.3653	0.3786	0.1681	0.1411	0.1426	0.1527
ARIMA training	0.3354	0.7523	1.0269	1.2555	0.5194	0.2254	0.1786	0.1139
ARIMA test	0.4480	0.6653	1.4407	0.5348	0.2117	0.2257	0.3612	0.2266
RBF training	0.0060	0.0147	0.1239	0.1479	0.0253	0.0285	0.0784	0.0768
RBF test	**0.3212**	0.0056	0.0860	0.2660	0.1508	0.0219	0.0797	0.1129
ANFIS training	0.0001	0.0067	0.0894	0.0156	0.0003	0.0097	0.0581	0.0786
ANFIS test	0.5220	0.0071	0.0295	0.2602	**0.1480**	0.0113	0.0472	0.1147
HONFIS (3rd ord.) training	0.0081	0.0081	0.0067	0.0269	0.0002	0.0134	0.0065	0.0157
HONFIS (3rd ord.) test	0.7961	**0.0052**	**0.0068**	**0.1235**	0.1733	**0.0094**	**0.0106**	**0.0644**

It is important to remark that the number of samples that should be predicted in a test set is strictly related to the computational cost of the learning procedure, as well as to the desired horizon of predictability, for which it is necessary to follow a given strategy for the energy management and to broadcast the relevant information to the involved operators. For instance, as the learning times of the previous models were in the order of some minutes and the considered time series are subsampled at 1 h, any model could be re-trained on a hourly basis, including a new sample to the training set and using using a test set with one sample only. This will thus incorporate the new information given by the the latest available sample that was measured.

Table 4.6: Prediction results for 15 March.

Prediction Model	NMSE				MARE			
	1-Day	3-Day	7-Day	30-Day	1-Day	3-Day	7-Day	30-Day
LSE training	0.0399	0.0428	0.1248	0.2093	0.0663	0.0682	0.0946	0.1061
LSE test	0.0393	0.0388	0.0412	0.0463	0.0657	0.0651	0.0673	0.0720
ARIMA training	0.6048	0.3498	2.3712	0.4085	0.2417	0.1752	0.0692	0.0688
ARIMA test	0.0091	0.0190	0.6537	0.5065	0.0292	0.0437	0.0426	0.0687
RBF training	0.0041	0.0002	0.0323	0.1225	0.0168	0.0037	0.0476	0.0715
RBF test	0.0063	0.0002	0.0153	0.0128	0.1508	0.0040	0.0355	0.0310
ANFIS training	0.0001	0.0005	0.0134	0.1268	0.0002	0.0011	0.0269	0.0720
ANFIS test	**0.0021**	0.0003	0.0086	0.0216	**0.0120**	**0.0010**	0.0208	0.0399
HONFIS (3rd ord.) training	0.0001	0.0001	0.0039	0.0012	0.0002	0.0012	0.0090	0.0035
HONFIS (3rd ord.) test	0.9437	**0.0002**	**0.0062**	**0.0003**	0.1985	0.0013	**0.0129**	**0.0035**

Table 4.7: Prediction results for 15 April.

Prediction Model	NMSE				MARE			
	1-Day	3-Day	7-Day	30-Day	1-Day	3-Day	7-Day	30-Day
LSE training	0.4307	0.4117	0.4059	0.2502	0.1193	0.1482	0.1322	0.1126
LSE test	**0.4356**	0.4205	0.4115	0.4252	0.1882	0.1851	0.1916	0.1981
ARIMA training	1.2368	3.8105	0.8170	0.6961	0.4845	0.3825	0.1961	0.2525
ARIMA test	1.2658	0.9881	0.2392	0.1691	0.3113	0.2767	0.1543	0.1216
RBF training	0.1533	0.1309	0.1145	0.1225	0.0900	0.0828	0.0647	0.0763
RBF test	1.2685	0.0754	0.1500	0.2697	0.2595	0.0720	0.1018	0.1506
ANFIS training	0.0002	0.0066	0.1469	0.1444	0.0184	0.0011	0.0803	0.0801
ANFIS test	2.3168	0.0036	0.2457	0.1991	**0.0120**	0.0171	0.1437	0.1252
HONFIS (3rd ord.) training	0.0025	0.0006	0.0039	0.0003	0.0096	0.0039	0.0068	0.0030
HONFIS (3rd ord.) test	1.6248	**0.0004**	**0.0203**	**0.0003**	0.3250	**0.0037**	**0.0172**	**0.0016**

Table 4.8: Prediction results for 15 May.

Prediction Model	NMSE				MARE			
	1-Day	3-Day	7-Day	30-Day	1-Day	3-Day	7-Day	30-Day
LSE training	0.7847	0.5205	0.3469	0.2998	0.1499	0.1671	0.1442	0.1290
LSE test	**0.7678**	0.5437	0.5586	0.5714	**0.2479**	0.1896	0.2006	0.2058
ARIMA training	6.9047	1.2811	0.4089	0.9808	0.5694	0.2333	0.1765	0.2757
ARIMA test	1.9263	1.5920	1.2429	0.3656	0.3716	0.3131	0.2694	0.2795
RBF training	0.2155	0.1940	0.1347	0.1404	0.0728	0.1066	0.0946	0.0854
RBF test	7.9004	0.1939	0.2250	0.2900	2.0610	0.1077	0.1327	0.1386
ANFIS training	0.0001	0.0080	0.0762	0.1718	0.0004	0.0098	0.0564	0.0972
ANFIS test	1.1388	0.0225	0.0957	0.3253	0.3458	0.0181	0.0605	0.1483
HONFIS (3rd ord.) training	0.0119	0.0218	0.0017	0.0005	0.0123	0.0041	0.0030	0.0027
HONFIS (3rd ord.) test	1.8555	**0.0002**	**0.0003**	**0.0002**	0.3427	**0.0009**	**0.0016**	**0.0029**

Table 4.9: Prediction results for 15 June.

Prediction Model	**NMSE**				**MARE**			
	1-Day	**3-Day**	**7-Day**	**30-Day**	**1-Day**	**3-Day**	**7-Day**	**30-Day**
LSE training	0.0108	0.0121	0.0096	0.0957	0.0363	0.0383	0.0318	0.0579
LSE test	0.0119	0.0118	0.0118	0.0137	0.0376	0.0375	0.0372	0.0361
ARIMA training	0.7354	1.3923	0.3083	0.3240	0.2934	0.3922	0.1314	0.1667
ARIMA test	0.8885	0.0115	0.0670	0.0205	0.3397	0.0327	0.0831	0.0473
RBF training	0.0003	0.0002	0.0002	0.0367	0.0054	0.0049	0.0046	0.0326
RBF test	**0.0005**	**0.0002**	**0.0002**	0.0075	**0.0073**	0.0049	0.0048	0.0266
ANFIS training	0.0001	0.0005	0.0003	0.0423	0.0005	0.0017	0.0053	0.0353
ANFIS test	0.0010	0.0006	0.0004	0.0118	0.0099	0.0019	0.0064	0.0303
HONFIS (3rd ord.) training	0.0001	0.0006	0.0001	0.0053	0.0001	0.0010	0.0022	0.0068
HONFIS (3rd ord.) test	0.1763	0.0004	0.0006	**0.0004**	0.0482	**0.0008**	**0.0012**	**0.0045**

Table 4.10: Prediction results for 15 July.

Prediction Model	**NMSE**				**MARE**			
	1-Day	**3-Day**	**7-Day**	**30-Day**	**1-Day**	**3-Day**	**7-Day**	**30-Day**
LSE training	0.0112	0.4319	0.1981	0.1226	0.0340	0.1576	0.0949	0.0700
LSE test	0.2471	0.3071	0.2432	0.2399	0.0905	0.1597	0.1117	0.0960
ARIMA training	1.0052	2.1191	0.2111	0.8671	0.5624	0.4436	0.1372	0.2310
ARIMA test	0.9954	0.1042	0.1541	0.1041	0.5820	0.0786	0.1170	0.0788
RBF training	0.0007	0.1733	0.0471	0.0538	0.0080	0.0848	0.0328	0.0438
RBF test	**0.1519**	0.1249	0.0698	0.0840	0.0909	0.0860	0.0452	0.0690
ANFIS training	0.0001	0.0004	0.0214	0.0707	0.0003	0.0315	0.0037	0.0501
ANFIS test	0.2438	0.0005	0.0397	0.0891	**0.0002**	0.0399	0.0054	0.0621
HONFIS (3rd ord.) training	0.0001	0.0008	0.0005	0.0023	0.0009	0.0053	0.0037	0.0044
HONFIS (3rd ord.) test	2.3358	**0.0005**	**0.0011**	**0.0004**	0.4013	**0.0059**	**0.0054**	**0.0035**

Table 4.11: Prediction results for 15 August.

Prediction Model	**NMSE**				**MARE**			
	1-Day	**3-Day**	**7-Day**	**30-Day**	**1-Day**	**3-Day**	**7-Day**	**30-Day**
LSE training	0.0295	0.0861	0.0406	0.1226	0.0517	0.0614	0.0442	0.0853
LSE test	0.1973	0.1941	0.1963	0.1927	0.0942	0.0868	0.0846	0.0984
ARIMA training	0.8843	0.2923	0.2207	0.2990	0.6698	0.1163	0.1179	0.1566
ARIMA test	0.9124	0.2968	0.1316	0.1081	0.7546	0.1786	0.1155	0.0747
RBF training	0.0020	0.0117	0.0071	0.0758	0.0143	0.0244	0.0122	0.0529
RBF test	0.1174	0.1249	**0.0157**	0.0841	0.0613	0.0297	0.0452	0.0614
ANFIS training	0.0001	0.0004	0.0214	0.0707	0.0003	0.0315	0.0037	0.0501
ANFIS test	0.2438	0.0005	0.0307	0.0891	**0.0002**	0.0399	**0.0213**	0.0621
HONFIS (3rd ord.) training	0.0001	0.0085	0.0075	0.0929	0.0147	0.0053	0.0102	0.0525
HONFIS (3rd ord.) test	**0.0107**	**0.0005**	0.0350	**0.0701**	0.0177	**0.0059**	0.0236	**0.0448**

Table 4.12: Prediction results for 15 September.

Prediction Model	NMSE				MARE			
	1-Day	3-Day	7-Day	30-Day	1-Day	3-Day	7-Day	30-Day
LSE training	0.7511	0.3141	0.2900	0.2971	0.1845	0.1280	0.1281	0.1192
LSE test	0.7526	0.7263	0.7265	0.7262	0.2181	0.2088	0.2099	0.2114
ARIMA training	1.3369	1.3875	0.9725	0.3571	0.4424	0.2483	0.2423	0.1741
ARIMA test	**0.4237**	0.6639	0.5273	0.3427	0.3354	0.2334	0.1888	0.1604
RBF training	0.2025	0.1243	0.1076	0.1840	0.1016	0.0855	0.0708	0.0883
RBF test	1.2412	0.2118	0.3201	0.3445	0.3014	0.1155	0.1379	0.1362
ANFIS training	0.0001	0.0001	0.0498	0.1953	0.0001	0.0001	0.0467	0.0903
ANFIS test	0.4766	**0.0001**	0.0633	0.4668	**0.1738**	**0.0008**	0.0529	0.1582
HONFIS (3rd ord.) training	0.0001	0.0001	0.0122	0.0032	0.0001	0.0010	0.0138	0.0050
HONFIS (3rd ord.) test	1.7032	0.0002	**0.0068**	**0.0092**	0.3168	0.0011	**0.0136**	**0.0125**

Table 4.13: Prediction results for 15 October.

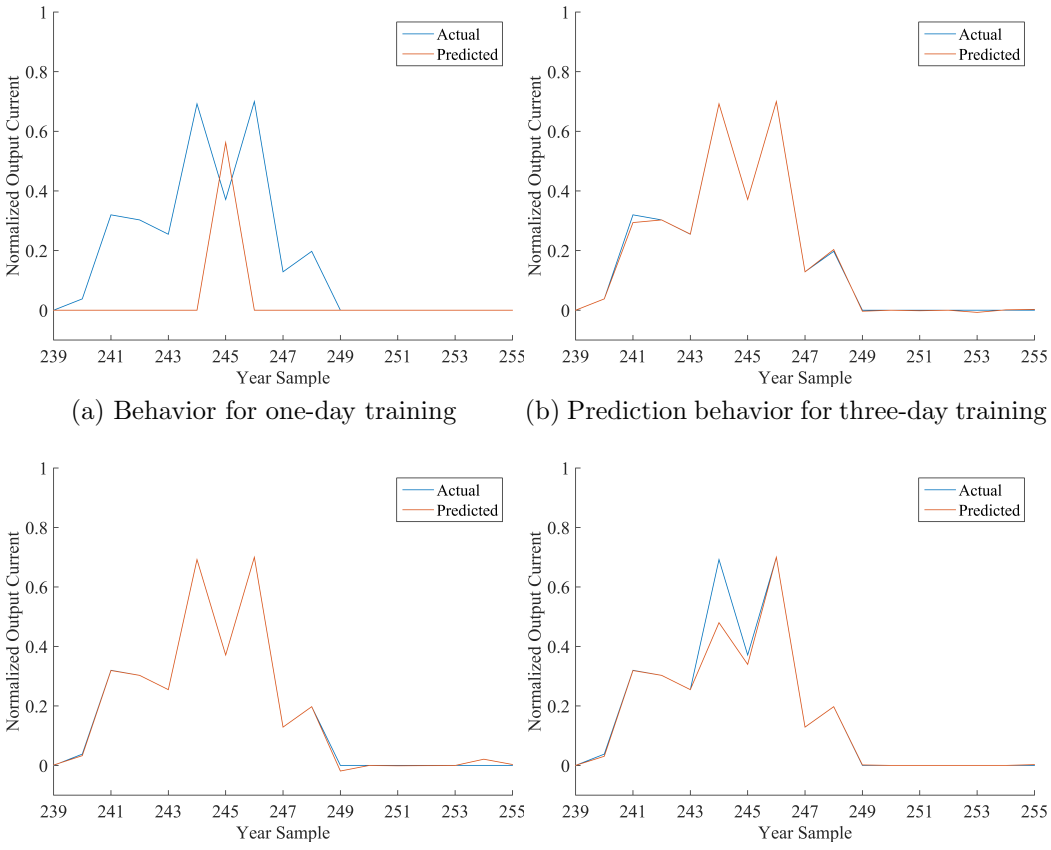
Prediction Model	NMSE				MARE			
	1-Day	3-Day	7-Day	30-Day	1-Day	3-Day	7-Day	30-Day
LSE training	0.1889	0.1198	0.0924	0.2720	0.0947	0.0983	0.0942	0.1047
LSE test	**0.2071**	0.1901	0.1899	0.1986	**0.1108**	0.1180	0.1194	0.1121
ARIMA training	1.5475	1.0569	0.5870	0.6301	0.3616	0.2311	0.1772	0.1965
ARIMA test	0.8957	0.6734	1.3392	0.3758	0.2439	0.2097	0.3119	0.1545
RBF training	0.0182	0.0231	0.0057	0.1583	0.0291	0.0426	0.0184	0.0677
RBF test	0.5444	0.0333	0.0126	0.1006	0.1772	0.0491	0.0273	0.0862
ANFIS training	0.0002	0.0001	0.0048	0.1692	0.0002	0.0003	0.0160	0.0702
ANFIS test	1.0372	**0.0003**	0.0086	0.1574	0.2217	**0.0004**	0.0188	0.1019
HONFIS (3rd ord.) training	0.0029	0.0036	0.0010	0.0145	0.0077	0.0078	0.0048	0.0112
HONFIS (3rd ord.) test	1.6475	0.0065	**0.0004**	**0.0008**	0.3124	0.0132	**0.0030**	**0.0036**

Table 4.14: Prediction results for 15 November.

Prediction Model	NMSE				MARE			
	1-Day	3-Day	7-Day	30-Day	1-Day	3-Day	7-Day	30-Day
LSE training	0.8224	0.4429	0.7591	0.3523	0.1139	0.1312	0.0457	0.0884
LSE test	**0.5896**	0.4394	0.5463	0.4347	**0.1575**	0.1547	0.1542	0.1544
ARIMA training	1.0569	0.7870	0.6644	1.1759	0.2311	0.2698	0.1937	0.2855
ARIMA test	0.6734	0.9403	4.4649	0.9554	0.2097	0.2776	1.2247	0.2504
RBF training	0.0725	0.1662	0.4894	0.2160	0.0335	0.0868	0.0379	0.0638
RBF test	8.5147	0.1844	0.9832	3.1527	0.6348	0.0913	0.2168	0.3780
ANFIS training	0.0001	0.0247	0.0480	0.2363	0.0001	0.0323	0.0650	0.0687
ANFIS test	5.5850	**0.0197**	**0.0599**	2.6863	0.4980	**0.0294**	**0.0688**	0.3969
HONFIS (3rd ord.) training	0.0009	0.0566	0.1659	0.0911	0.0007	0.0219	0.0048	0.0279
HONFIS (3rd ord.) test	2.3314	0.0433	0.6480	**0.3782**	0.5552	0.0349	0.1045	**0.1206**

Table 4.15: Prediction results for 15 December.

Prediction Model	NMSE				MARE			
	1-Day	3-Day	7-Day	30-Day	1-Day	3-Day	7-Day	30-Day
LSE training	0.6335	0.5107	0.5513	0.3814	0.1109	0.1110	0.0970	0.0984
LSE test	**0.4326**	0.3922	0.4051	0.4512	**0.1436**	0.1552	0.1526	0.1518
ARIMA training	1.0407	1.1536	1.9333	1.1766	0.3000	0.2067	0.2194	0.2016
ARIMA test	0.9998	0.8886	0.5563	0.4208	0.2272	0.2090	0.1557	0.1769
RBF training	0.0754	0.0379	0.0540	0.2106	0.0412	0.0481	0.0383	0.0665
RBF test	0.9442	0.0793	0.1061	0.1270	0.2207	0.0768	0.0627	0.0684
ANFIS training	0.0609	0.0043	0.0556	0.2272	0.0401	0.0064	0.0410	0.0719
ANFIS test	1.2890	**0.0057**	0.1005	0.1443	0.2599	**0.0074**	0.0639	0.0643
HONFIS (3rd ord.) training	0.0060	0.0039	0.0198	0.1012	0.0083	0.0069	0.0094	0.0316
HONFIS (3rd ord.) test	1.1232	0.0079	**0.0544**	**0.1017**	0.1951	0.0083	**0.0298**	**0.0416**



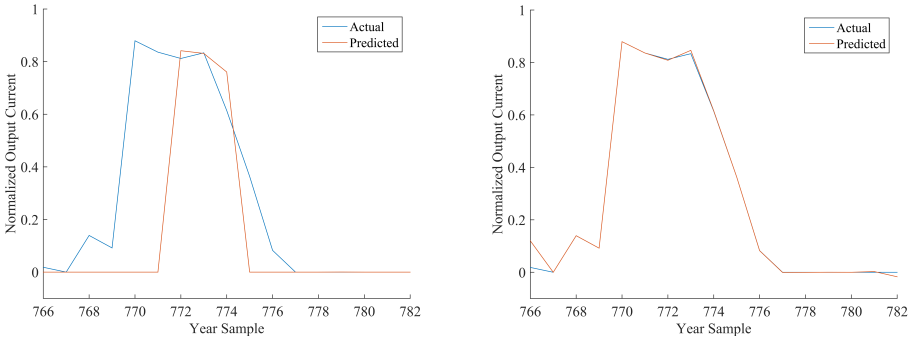
(a) Behavior for one-day training

(b) Prediction behavior for three-day training

(c) Prediction behavior for seven-day training

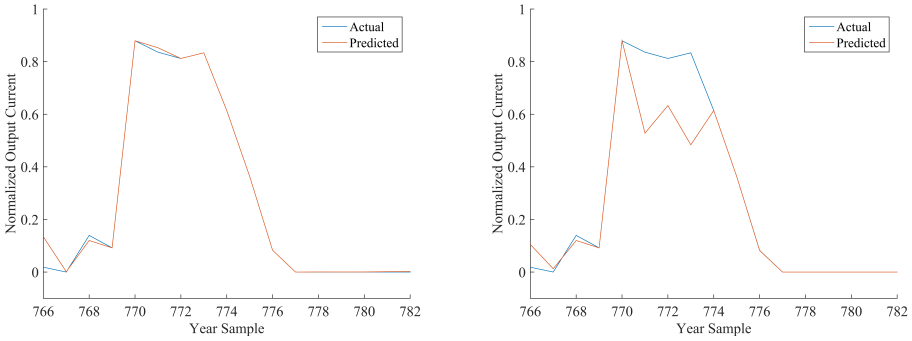
(d) Prediction behavior for 30-day training

Figure 4.14: Prediction using the best model (HONFIS) for 15 January.



(a) Behavior for one-day training

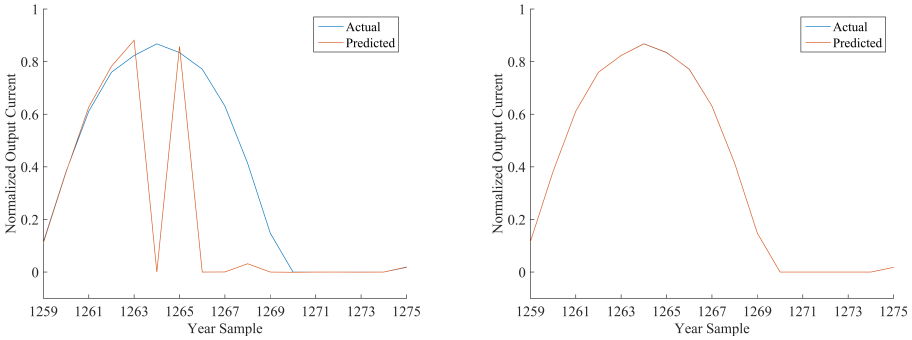
(b) Behavior for three-day training



(c) Behavior for seven-day training

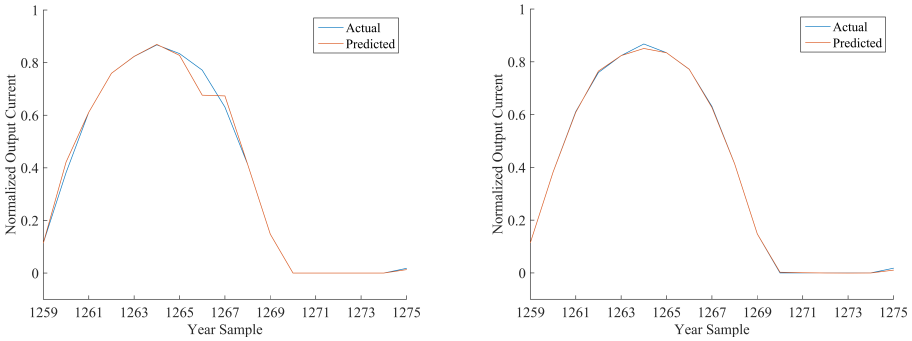
(d) Behavior for 30-day training

Figure 4.15: Prediction using the best model (HONFIS) for 15 February.



(a) Behavior for one-day training

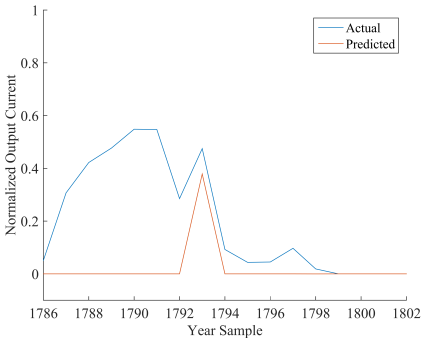
(b) Behavior for three-day training



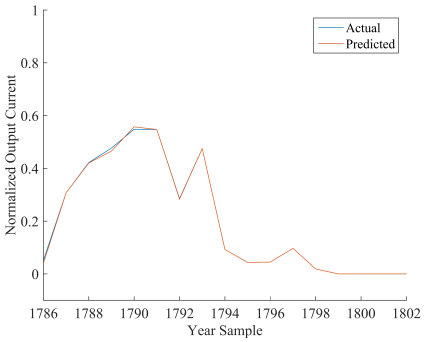
(c) Behavior for seven-day training

(d) Behavior for 30-day training

Figure 4.16: Prediction using the best model (HONFIS) for 15 March.

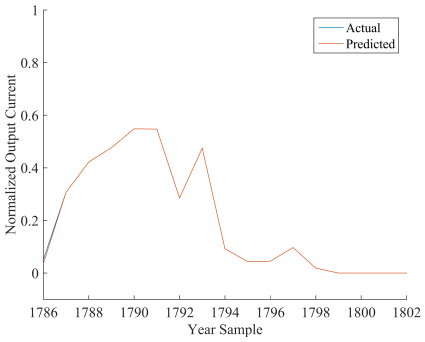


(a) Behavior for one-day training



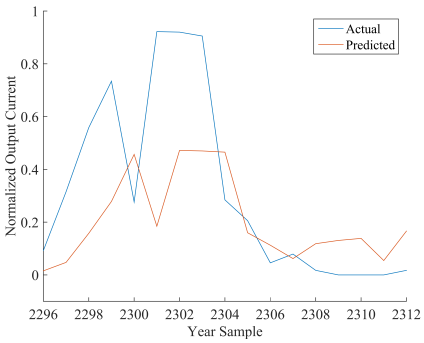
(b) Behavior for three-day training

(c) Behavior for seven-day training

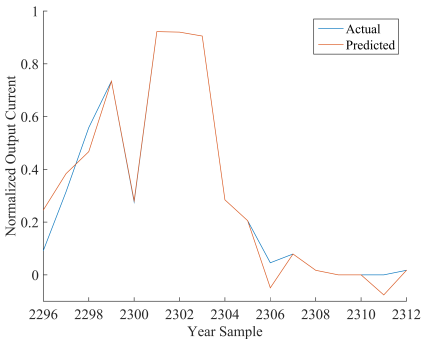


(d) Behavior for 30-day training

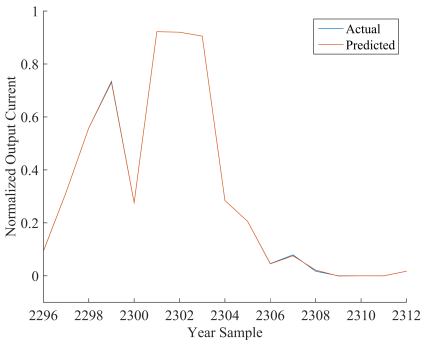
Figure 4.17: Prediction using the best model (HONFIS) for 15 April.



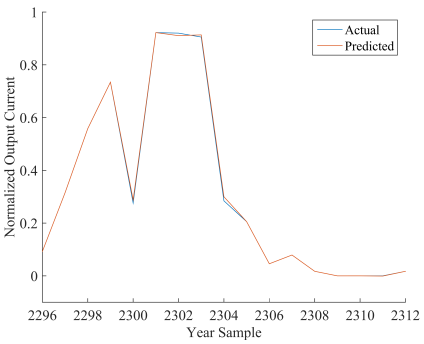
(a) Behavior for one-day training



(b) Behavior for three-day training

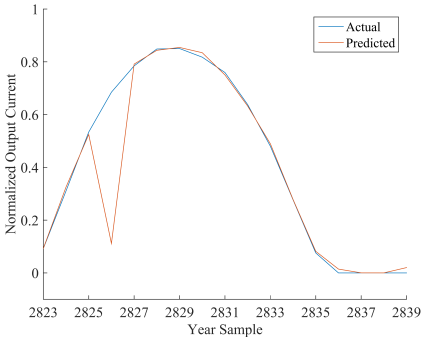


(c) Behavior for seven-day training

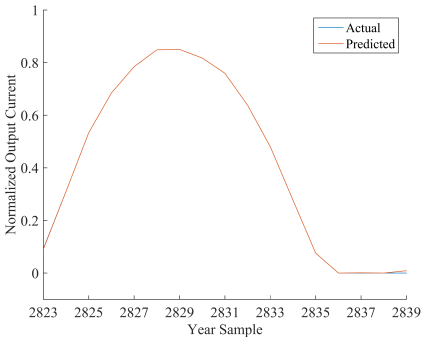


(d) Behavior for 30-day training

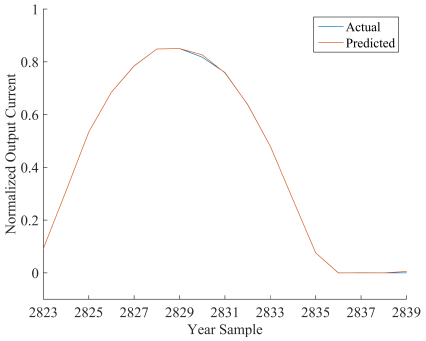
Figure 4.18: Prediction using the best model (HONFIS) for 15 May.



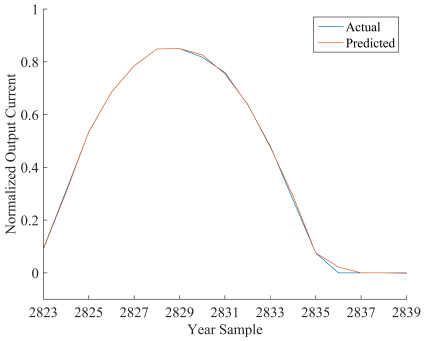
(a) Behavior for one-day training



(b) Behavior for three-day training

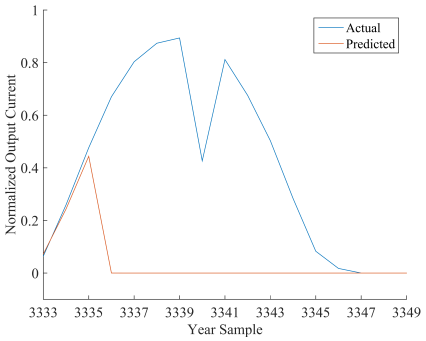


(c) Behavior for seven-day training

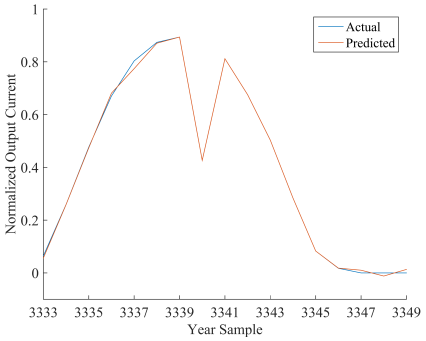


(d) Behavior for 30-day training

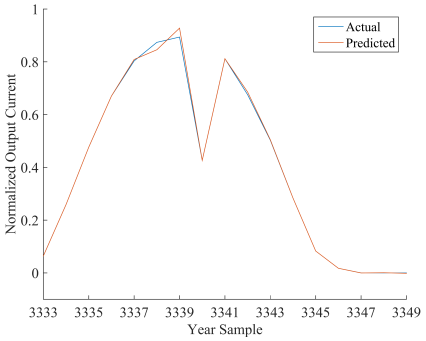
Figure 4.19: Prediction using the best model (HONFIS) for 15 June.



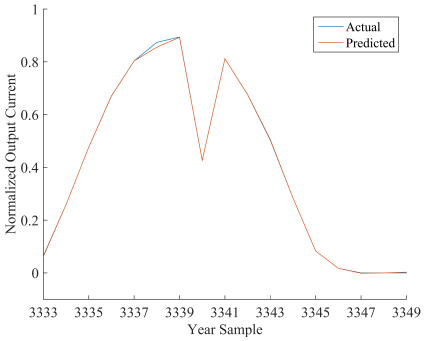
(a) Behavior for one-day training



(b) Behavior for three-day training

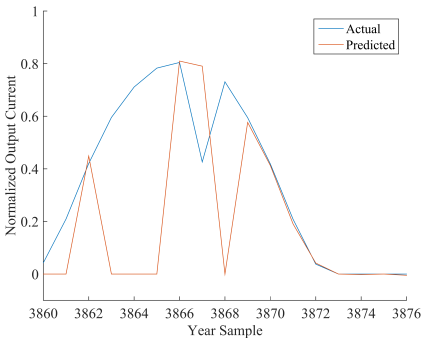


(c) Behavior for seven-day training

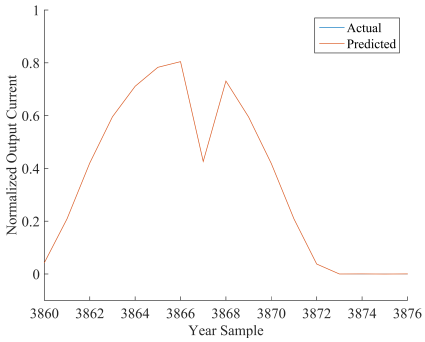


(d) Behavior for 30-day training

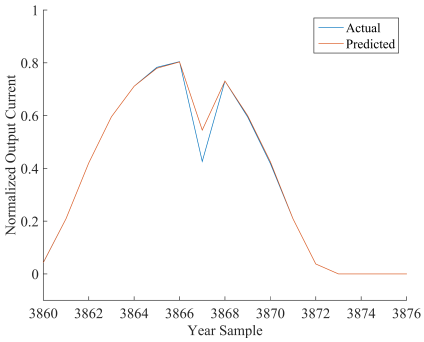
Figure 4.20: Prediction using the best model (HONFIS) for 15 July.



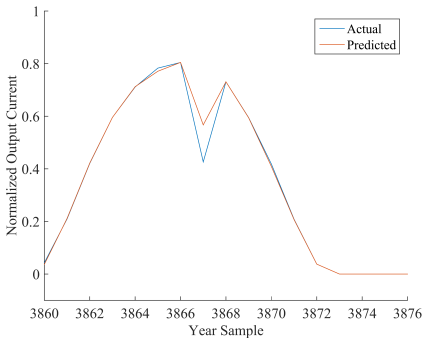
(a) Behavior for one-day training



(b) Behavior for three-day training

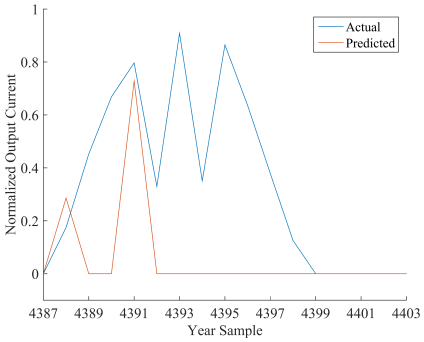


(c) Behavior for seven-day training

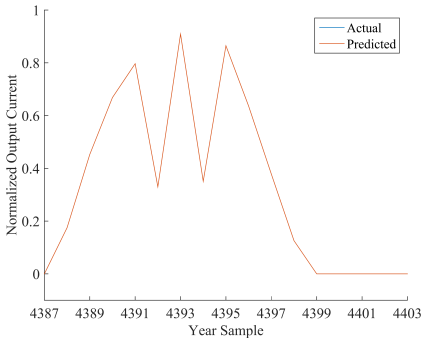


(d) Behavior for 30-day training

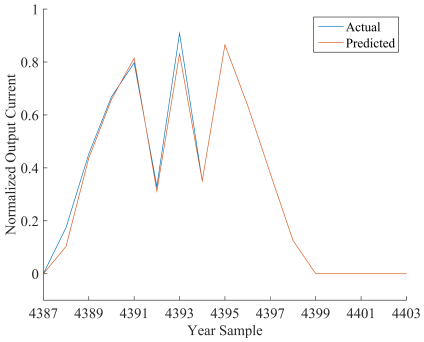
Figure 4.21: Prediction using the best model (HONFIS) for 15 August.



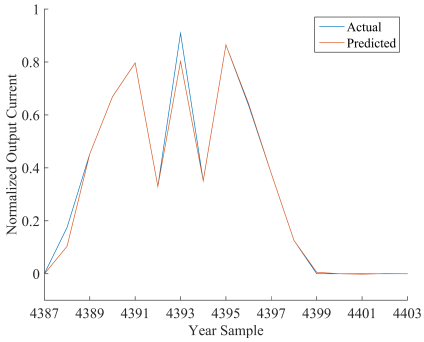
(a) Behavior for one-day training



(b) Behavior for three-day training

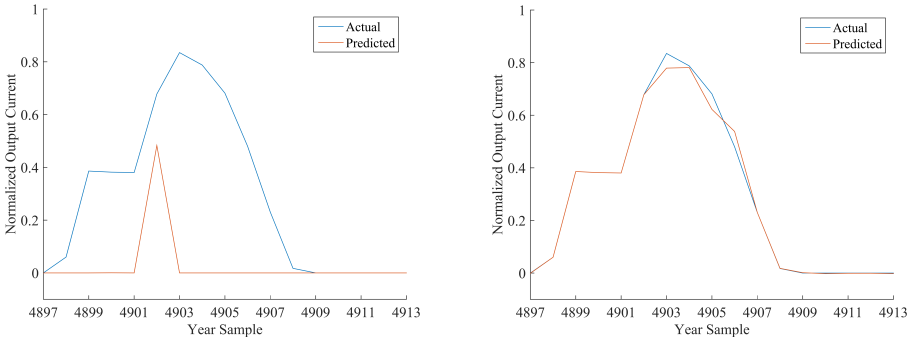


(c) Behavior for seven-day training



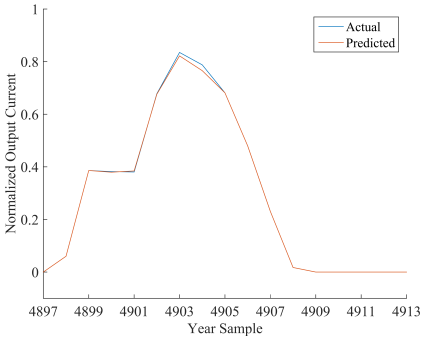
(d) Behavior for 30-day training

Figure 4.22: Prediction using the best model (HONFIS) for 15 September.

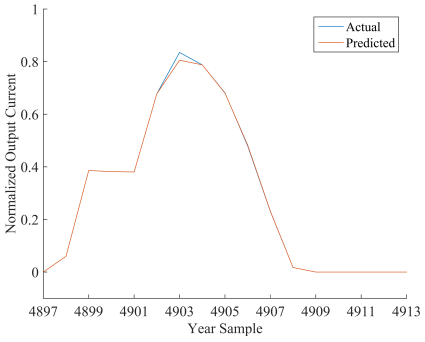


(a) Behavior for one-day training

(b) Behavior for three-day training

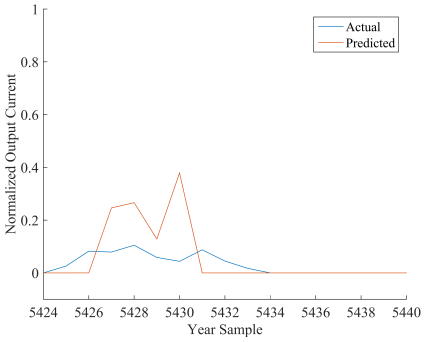


(c) Behavior for seven-day training

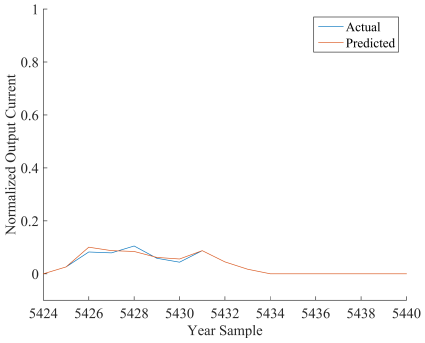


(d) Behavior for 30-day training

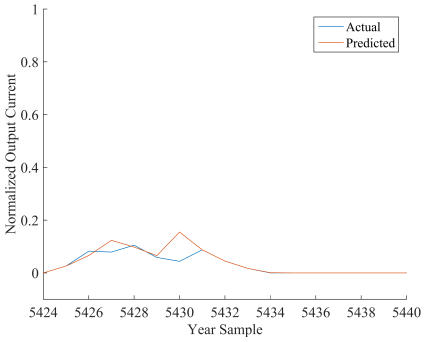
Figure 4.23: Prediction using the best model (HONFIS) for 15 October.



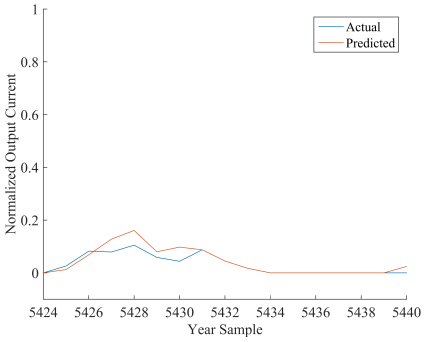
(a) Behavior for one-day training



(b) Behavior for three-day training

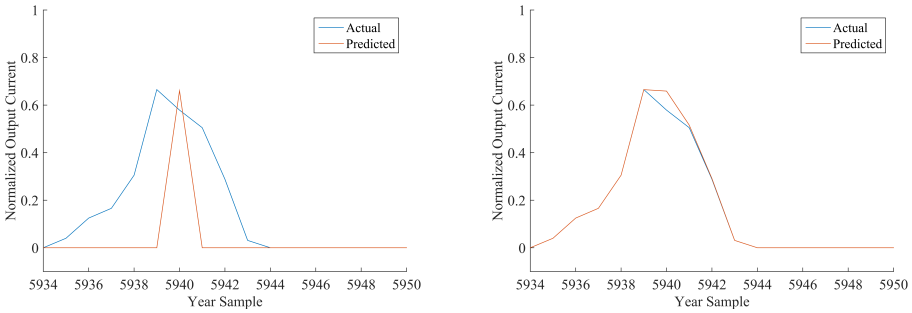


(c) Behavior for seven-day training



(d) Behavior for 30-day training

Figure 4.24: Prediction using the best model (HONFIS) for 15 November.



(a) Behavior for one-day training

(b) Prediction behavior for three-day training

(c) Prediction behavior for seven-day training

(d) Prediction behavior for 30-day training

Figure 4.25: Prediction using the best model (HONFIS) for 15 December.

Table 4.16: Prediction results from 15–21 January.

Prediction Model	**NMSE**		**MARE**	
	7-Day	**30-Day**	**7-Day**	**30-Day**
LSE test	**0.6304**	0.6464	0.0720	0.0739
ARIMA test	1.2130	1.2252	**0.1192**	0.0957
RBF test	5.9196	0.5863	0.1500	0.0734
ANFIS test	1.0452	0.5863	0.1239	0.0800
HONFIS (3rd ord.) test	3.4869	**0.2296**	0.1651	**0.0185**

Table 4.17: Prediction results from 15–21 February.

Prediction Model	**NMSE**		**MARE**	
	7-Day	**30-Day**	**7-Day**	**30-Day**
LSE test	**0.2490**	0.2390	**0.1242**	0.1239
ARIMA test	1.4324	1.0998	0.3048	0.2697
RBF test	1.4771	0.1224	0.2359	0.0758
ANFIS test	0.6664	0.1281	0.1895	0.0755
HONFIS (3rd ord.) test	1.0124	**0.0197**	0.2236	**0.0147**

However, in order to ensure a wide range of action, it is worth forecasting more than one sample at a time, as previously done considering an entire day. Although longer test sets might not be so useful and accurate, we consider in the following also weekly test sets, so as to report the prediction results for more days in a year and to evaluate the sensitivity of the proposed approach not only to different training, but also to different test conditions. The numerical results for each tested week are reported in Tables 4.16–4.27. In such a case, we have considered seven-day or 30-day training sets only, that is longer than the test set or of the same length. As the starting day in the test set is still the 15th of each month, the training results are the same as the ones reported in Tables 4.4–4.15, respectively.

Table 4.18: Prediction results from 15–21 March.

Prediction Model	**NMSE**		**MARE**	
	7-Day	**30-Day**	**7-Day**	**30-Day**
LSE test	**0.1402**	0.1414	0.0998	0.1007
ARIMA test	0.4567	0.2935	0.1934	0.1669
RBF test	0.2806	0.0801	0.1298	0.0657
ANFIS test	0.2732	0.0945	**0.1091**	0.0667
HONFIS (3rd ord.) test	3.4978	**0.0008**	0.3570	**0.0039**

Table 4.19: Prediction results from 15–21 April.

Prediction Model	**NMSE**		**MARE**	
	7-Day	**30-Day**	**7-Day**	**30-Day**
LSE test	0.4062	0.4418	**0.1493**	0.1627
ARIMA test	0.7751	0.6039	0.2667	0.2392
RBF test	1.1634	0.1780	0.1938	0.1004
ANFIS test	**0.2311**	0.1044	0.3102	0.1223
HONFIS (3rd ord.) test	1.8455	**0.0004**	0.3343	**0.0019**

Table 4.20: Prediction results from 15–21 May.

Prediction Model	**NMSE**		**MARE**	
	7-Day	**30-Day**	**7-Day**	**30-Day**
LSE test	0.4062	0.3882	0.1493	0.1299
ARIMA test	0.6885	0.4344	0.2492	0.1996
RBF test	4.4347	0.1571	0.1298	0.0773
ANFIS test	**0.2962**	0.1044	**0.1109**	0.0778
HONFIS (3rd ord.) test	1.8219	**0.0003**	0.3592	**0.0033**

Table 4.21: Prediction results from 15–21 June.

Prediction Model	**NMSE**		**MARE**	
	7-Day	**30-Day**	**7-Day**	**30-Day**
LSE test	**0.2334**	0.2295	**0.1031**	0.0981
ARIMA test	0.3503	0.3825	0.1124	0.1502
RBF test	0.9345	0.1238	0.2062	0.0688
ANFIS test	0.5972	0.1655	0.1650	0.0818
HONFIS (3rd ord.) test	0.9560	**0.01213**	0.1935	**0.0108**

Table 4.22: Prediction results from 15–21 July.

Prediction Model	**NMSE**		**MARE**	
	7-Day	**30-Day**	**7-Day**	**30-Day**
LSE test	**0.1932**	0.1886	**0.0712**	0.0725
ARIMA test	0.4167	0.4025	0.1918	0.1839
RBF test	9.8115	0.0695	0.4956	0.0400
ANFIS test	0.4592	0.0854	0.1229	0.045
HONFIS (3rd ord.) test	1.0701	**0.0057**	0.1905	**0.0053**

Table 4.23: Prediction results from 15–21 August.

Prediction Model	**NMSE**		**MARE**	
	7-Day	**30-Day**	**7-Day**	**30-Day**
LSE test	**0.1445**	0.1408	**0.0750**	0.0815
ARIMA test	0.1283	0.1487	0.0620	0.0763
RBF test	6.2338	0.0521	1.2980	0.0473
ANFIS test	0.1523	0.0849	0.0691	0.0478
HONFIS (3rd ord.) test	0.9875	**0.0045**	0.1884	**0.0065**

Table 4.24: Prediction results from 15–21 September.

Prediction model	NMSE		MARE	
	7-Day	**30-Day**	**7-Day**	**30-Day**
LSE test	**0.1803**	0.1806	**0.1032**	0.1023
ARIMA test	0.7250	0.3781	0.2216	0.1668
RBF test	0.4751	0.0898	0.1671	0.0670
ANFIS test	1.7361	0.1031	0.2823	0.0678
HONFIS (3rd ord.) test	1.8185	**0.0017**	0.3105	**0.0048**

Table 4.25: Prediction results from 15–21 October.

Prediction Model	**NMSE**		**MARE**	
	7-Day	**30-Day**	**7-Day**	**30-Day**
LSE test	**0.2266**	0.2138	**0.1169**	0.1098
ARIMA test	0.8376	0.3808	0.2021	0.1626
RBF test	1.6818	0.0898	0.2237	0.0670
ANFIS test	1.0212	0.1225	0.1782	0.0650
HONFIS (3rd ord.) test	0.8892	**0.0179**	0.1884	**0.0169**

Table 4.26: Prediction results from 15–21 November.

Prediction Model	**NMSE**		**MARE**	
	7-Day	**30-Day**	**7-Day**	**30-Day**
LSE test	**0.4126**	0.3083	**0.1274**	0.1166
ARIMA test	1.2414	1.3639	0.2193	0.2285
RBF test	2.6564	0.1509	0.2237	0.0686
ANFIS test	1.3567	0.1445	0.1766	0.1630
HONFIS (3rd ord.) test	1.6437	**0.1159**	0.2320	**0.0347**

Table 4.27: Prediction results from 15–21 December.

Prediction Model	**NMSE**		**MARE**	
	7-Day	**30-Day**	**7-Day**	**30-Day**
LSE test	**0.3800**	0.3790	0.1347	0.1348
ARIMA test	1.3209	1.1432	0.2322	0.2059
RBF test	3.2432	0.2143	0.2649	0.0823
ANFIS test	0.4764	0.2168	**0.1253**	0.0886
HONFIS (3rd ord.) test	3.6065	**0.0684**	0.2519	**0.0364**

Regarding the experimental results reported above, it can be noted that the numerical results in terms of either NMSE or MARE basically agree, and they are coherent with the graphical behaviors, reported in the figures, between actual and predicted time series. Note that a perfect match is obtained when the NMSE is smaller than about 5×10^{-4}.

In the days with the most stable meteorological conditions (i.e., 15 March or 15 June), the prediction results for daily test sets are better than the others. Furthermore, there are differences among the results associated with diverse training procedures. The tests with one-day training sets have a high variability for all of the algorithms because the prediction is highly affected by the difference in irradiation between the day of the training and the day of the test. In fact, when the meteorological condition varies greatly from the training day to the sequent test day, the algorithms are trained on a set that is almost not correlated with the test set. It can also be noted that when the irradiation is very similar (no clouds, good weather), the performance is very good also for the one-day training set.

A similar discussion can be made, with smaller variability, for the three-day training sets. It has to be noted that we have inserted the one-day and three-day training sets mainly to show the sensitivity of the models to different training conditions. The results for seven-day training sets are much more stable for all of the algorithms and the days. The results for 30-day training sets are stable as well, but the sequences are much longer; thus, if the days are variable in terms of meteorological conditions, the model has a worse performance because the training process is more difficult. Training sets longer than 30 days were also considered in some preliminary tests, and the results confirm such a trend for which the performance decreases progressively as much as the length of the training set increases.

4.4.3 Ponza Island Case Study

Increasing the use of renewable energy sources (RESs) for power generation is at the heart of European energy policy [109]. Achieve a large scale penetration of RESs into the conventional electrical power systems [110, 111] is still challenging for several reasons: on the one hand we observe high costs for installation, operation and maintenance in comparison to conventional sources [112]; on the other hand, RESs are characterized by a high intermittent power production [113]. Consequently, storage capacity is usually

necessary to make RESs dispatchable [25], combined with accurate forecasting techniques to predict demand and generation profiles [114].

In this scenario, the integration of RESs is much more challenging in the power systems of small islands that, being not connected to the national grid and controlled by one stakeholder only, lack the necessary flexibility in demand, generation, transmission, and pricing. It is consequently difficult to support modern mechanisms of price responsive demand [115], to encourage consumers' participation to the electricity market and foster elasticity in the load demand.

In the work presented in this section, the goal is to apply innovative forecasting techniques [55, 71] to predict both renewable energy (RE) production and load energy demand in the small Italian island of Ponza, in order to optimize the management of a centralized battery energy storage system (BESS) [116] necessary to maximize the penetration of RESs into the existing medium voltage grid. The prediction itself is carried on using neural networks, specifically the Echo State Network (ESN) paradigm. The studied scenario is the one considered by the Decree of the Italian Ministry of Economic Development adopted on February 14th, 2017 which prescribes the implementation of 2.16 MW RESs by 2030.

Broadly speaking, regarding the problem statement, electrification in small islands without connection to the national transmission grid is one of the most challenging issues. The electric power generation is usually carried out by autonomous diesel-fueled power stations whose operation brings a number of environmental and economic drawbacks [117]. Ponza island does not make an exception: the annual energy consumption – which ranges around 11.8 GWh with a power annual peak of 4.4 MW measured during summer months – is today supplied at 98.7% by six diesel generators (DGs), whose nominal rated power ranges from 1020 kW to 1600 kW. In order to ensure stability for both N and $N-1$ conditions and keep generators in a safety range, the generators are properly managed by the local electric operator who ensures always a minimum power reserve equal to the load power consumption by taking online at least two parallel DGs. Consequently, since the generators work at low efficiency with a low loading rate, the annual consumption of diesel equals 3.2 millions of liters with an overall average efficiency of the generators around 35%.

Being located in area with an abundance of wind, sunshine and water, and being relatively small, Ponza island has geographically ideal conditions

for almost all forms of RESs. In a stand-alone micro grid, the highest priority is to keep a reliable power supply to customers, trying meanwhile to minimize operation costs by using totally wind and solar generation. The scenario studied in this application and prescribed for 2030 by the Decree of the Italian Ministry of Economic Development adopted on February 14th, 2017, considers the integration of 2.16 MW RESs, which are split into 1.16 MW photovoltaic power plants, 0.9 MW wind plants, and 0.1 MW Organic Rankine Cycle (ORC). In this scenario, DGs, due to their reliable power supply capability, remain in order to control the system voltage and frequency and to balance the power generation and consumption (primary, secondary and tertiary control). In addition, the installation of a centralized BESS is planned, with rated power equal to 2.7 MW and energy storage capacity equal to 1.35 MWh (a rate of $2C$ is supposed) in order to provide power support, spinning reserve, peak-shaving, and regularize intermittent RE outputs.

In this framework, forecasting techniques that are able to predict demand and generation profiles over the next week become a paramount factor for implementing an effective control strategy of the DGs and BESS, while ensuring a safe and conservative energy management for the stand-alone microgrid. The control strategy can be summarized as follows.

1. Diesel generation P_{DG} is always maintained between a minimum $P_{\mathrm{DG}}^{\mathrm{min}}$ and a maximum $P_{\mathrm{DG}}^{\mathrm{max}}$ value to avoid light-load conditions and provide reserve capacity margin.

2. The BESS provides mainly spinning reserve to ensure stability under $N-1$ conditions, thus allowing to take online a single DG when possible and generally increase the load rate of the DGs. Consequently the State of Charge (SOC) level is never lowered under a proper minimum threshold $\mathrm{SOC}^{\mathrm{min}}$ that ensures the energy reserve necessary to supply the load for the time necessary to turn on a DG under fault conditions. Assuming the worst case of the shedding of one of the largest 1600 kW DG, which requires around 10 minutes to start and take the load, the minimum energy $E_{\mathrm{BESS}}^{\mathrm{min}}$ always required in the BESS is 267 kW, rounded to 500 kW to avoid deep discharges.

3. The BESS provides smoothing of short-time RE output fluctuations since the power regulation capability of DGs is limited by ramp constraint. This should require a short time scale forecasting. Anyway, it

is not necessary in the present case since energy storage is large enough to handle any practical expected fluctuation (even the shedding of all the RESs when the BESS is completely full).

4. The forecasting is performed up to three days ahead with one hour time step on the load P_{LOAD} and RES generation P_{RES} profiles. Due to the equipment rated powers of the present study and being Ponza a summer holiday destination, we observe that the generated P_{RES} can be larger than P_{LOAD} only in the central hours of the day during spring and fall months when the PV production is high.

5. In order to reduce the number of cycles of the BESS, and extend its lifetime, we avoid deep charging and discharging frequently; consequently, when $P_{\mathrm{LOAD}} > P_{\mathrm{RES}}$, the BESS is discharged (remaining always above $\mathrm{SOC}^{\mathrm{min}}$ level) only when it is necessary to free the necessary capacity to store the predicted surplus of production over the three days ahead that reveled to be a suitable long period in the present case study. The discharge is preferably performed at the load peaks where it is easier to reduce DGs outputs remaining always above the $P_{\mathrm{DG}}^{\mathrm{min}}$. In this manner, we try to ensure that when $P_{\mathrm{RES}} > P_{\mathrm{LOAD}}$ the BESS is always able to store the surplus production without the need to dump the RE output, since there is no possibility to manage the demand side energy consumption in our scenario. The SOC level of the BESS is allowed to range between a lower $\mathrm{SOC}^{\mathrm{low}} > \mathrm{SOC}^{\mathrm{min}}$ and an upper $\mathrm{SOC}^{\mathrm{up}} < 100\%$ level in order to provide reserve capacity for forecasting errors.

For the described purpose, the ESN model is used as detailed in Sect. 4.2.5. In the following, we propose a generalized approach for which, at the generic time n, the input to the ESN reservoir can be any combination of past available samples, which are used to predict the next one $S[n+m]$. As the relation between predicted and past samples usually is not given, in a prediction problem it is important to study which samples of the sequence are relevant to the prediction task. The naive approach described in (4.14) resides in assuming that all the past samples of the sequence to be predicted are equally important, without finding an optimal subset. However, a more fruitful approach consists in selecting the samples via embedding of the sequence [55]. The sequence generated by a complex, often chaotic and noisy system like the PV under exam can be observed by its output only. So, in order to reconstruct the evolution of the system, the original sequence $S[n]$

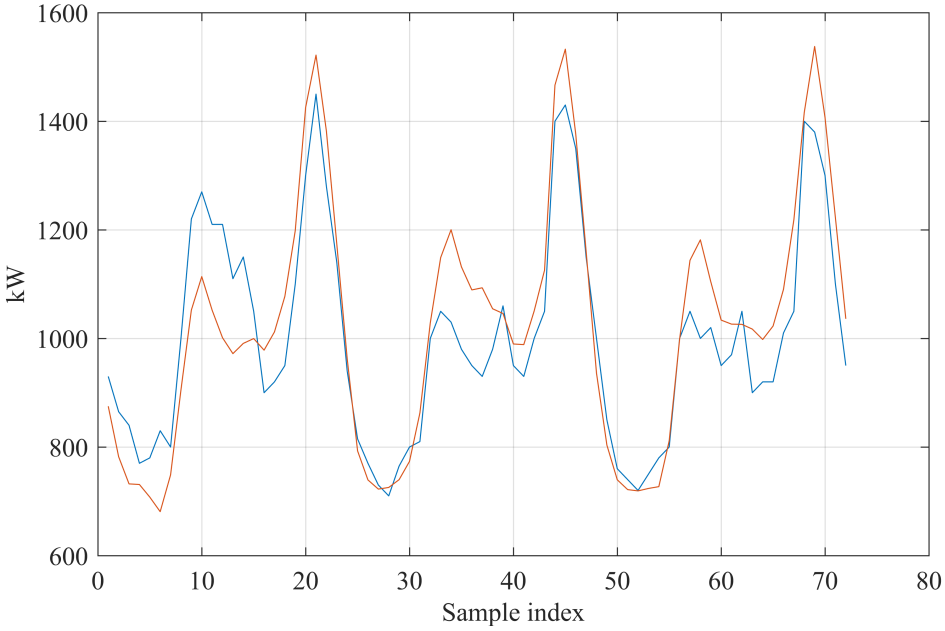


Figure 4.26: Actual (blue) and predicted (red) load for the test samples.

must be embedded. This is done by determining two parameters [56]: the embedding dimension D and the time lag T, in such a way the reconstructed state at time n can be found as:

$$\overline{\boldsymbol{S}}[n] = \Big[S[n]\ S[n-T]\ \ldots\ S[n-T(D-1)]\Big]^T. \tag{4.30}$$

Looking at (4.14), it is evident that the ESN input is an estimate of the unfolded, reconstructed state of the unknown system that generates the observed time series, having assumed $T = 1$ and $D = N_i$. These parameters can be estimated by using suited techniques, in the following we will adopt a rule-of-thumb evaluation based on the processed time series, as in [118].

We carried out several experiments for assessing the proposed approach. In the following, we introduce the results of a representative case study, which is relative to days in the middle of May 2016. In the operation phase, a new prediction is computed every 3 days by using the previous 30 days for training the ESN, in order to use the predicted information for efficiently managing the BESS. All the time series are sampled at a rate of one sample per hour and hence, the training set is made of $24 \cdot 30 = 720$ samples, while the test set is made of $24 \cdot 3 = 72$ samples. As we have to predict three days at once, the prediction distance is set to $m = 72$.

Firstly, we illustrate the prediction of the load in Fig. 4.26 and the one of solar production in Fig. 4.27. Both of them have high variability but the

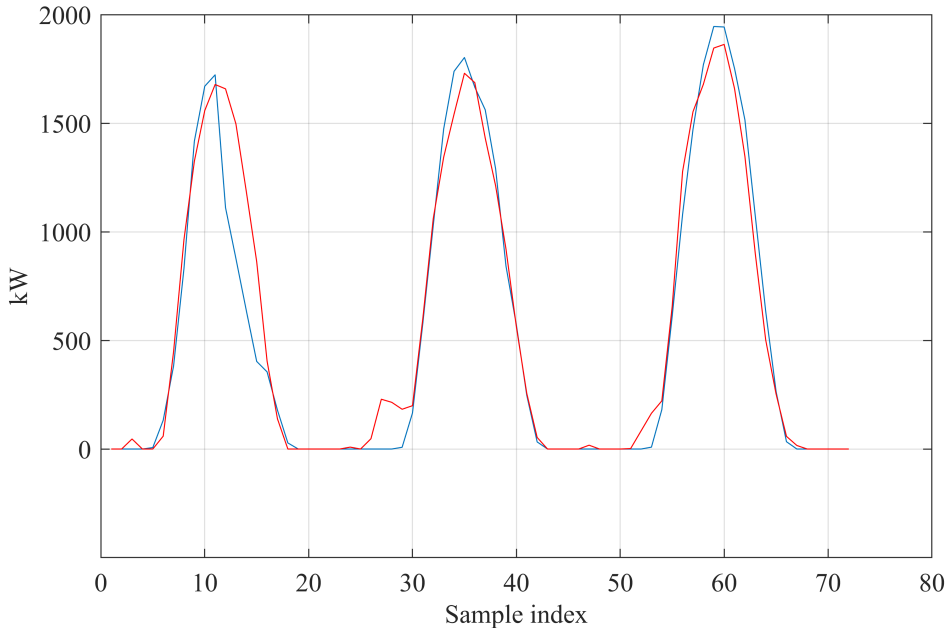


Figure 4.27: Actual (blue) and predicted (red) solar production for the test samples.

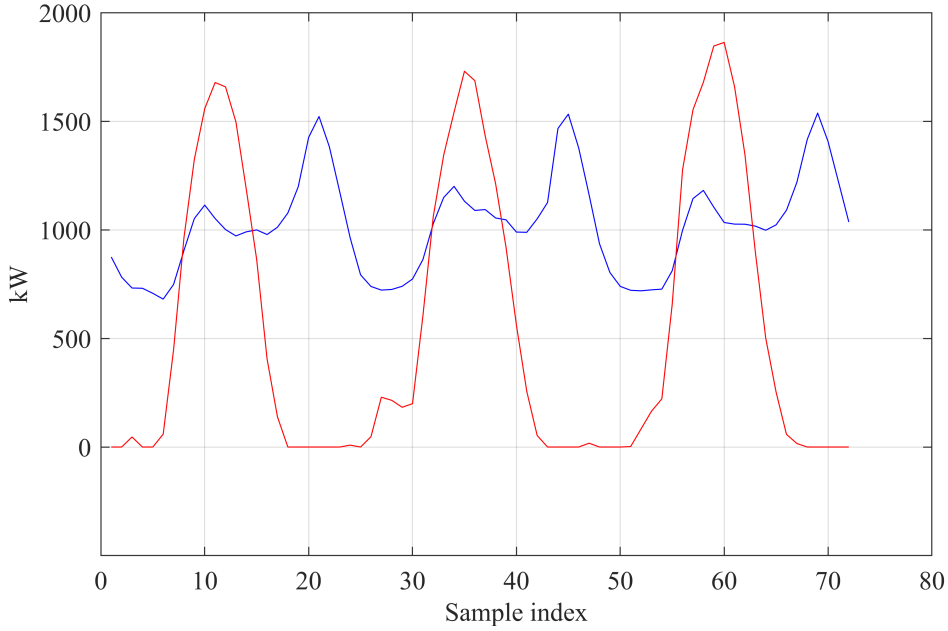


Figure 4.28: Estimated load (blue) and predicted solar production (red) for the test.

ESN is capable of giving a robust prediction using the embedded time series, with $T = 1$ and $D = 7$ for the load and to $T = 1$ and $D = 17$ for the solar production. We outline that it is necessary to use a very high prediction distance, which is usually set to one in other applications. Thus, the results

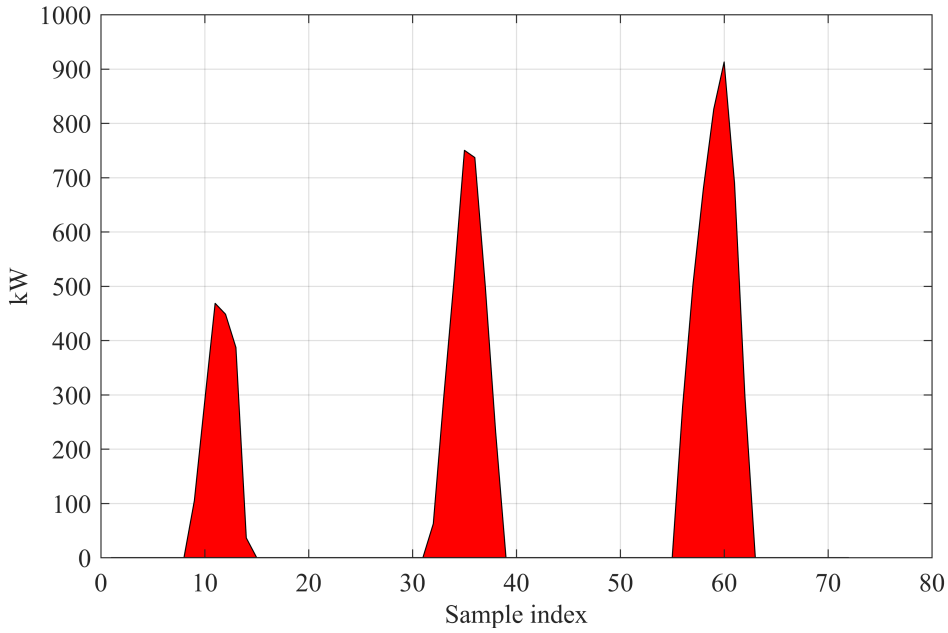


Figure 4.29: Estimated load power surplus.

are even more valuable in the present case.

At every iteration, the algorithm computes the difference between the predicted solar generation and the estimated load, which are shown together in Fig. 4.28. Defining the surplus as the positive values of the difference between the predicted solar generation and estimated load, we can see that in the first day of prediction there is a surplus of energy, as illustrated in Fig. 4.29. This information is useful for the BESS, which has to present itself at that day with the right SOC to receive the surplus. The surplus itself is indeed managed by looking at the successive prediction days, in which there will be no sufficient generation.

We also report in Fig. 4.30 the surplus computation for the actual load and real solar production for the same three days, which does not differs significantly, being smaller than the predicted one. This is a desirable property as, conversely, if the actual surplus were greater than the predicted one it would result in an unmanageable excess of power in the grid. Comments similar to the present ones can be drawn from any other time period and the related prediction of surplus.

4.4.4 Distributed prediction

As already introduced, the forecasting of power output is particularly crucial for producers, grid operators and market players when dealing with renewable

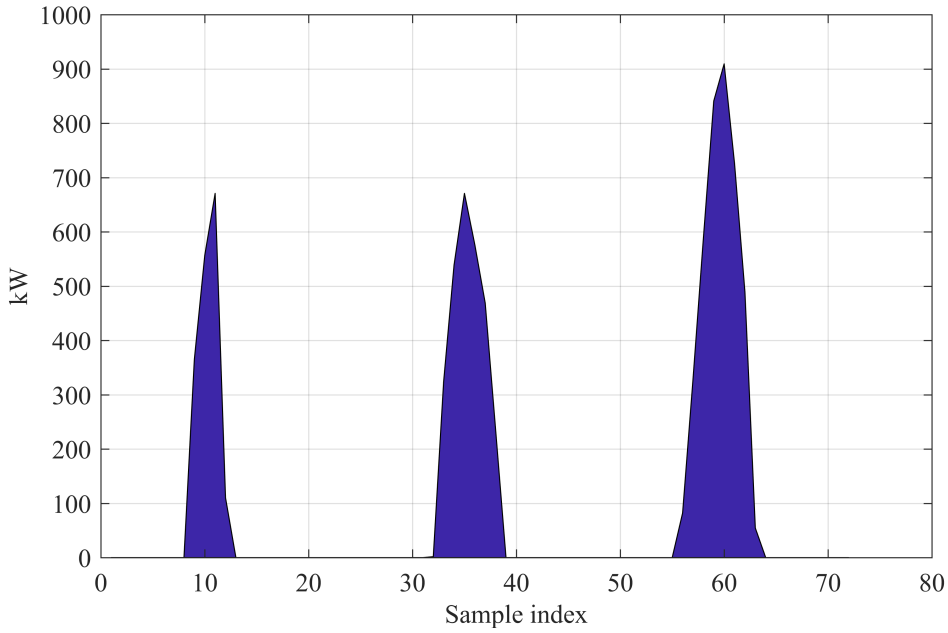


Figure 4.30: Real load power surplus.

energy sources (RES) whose expected power production is inherently intermittent, as is the case with photovoltaic (PV) [119,120] and wind power [121] sources.

In the past few years, a huge amount of literature has been published about power forecasting models related to PV plants [26, 31, 32], focused either on the prediction of the primary source (i.e., irradiance or irradiation) [87, 122] or on the produced output power [29, 123]. The existing solutions can be classified into four major categories, namely, long, medium, short and very short term forecast, and, despite the time horizon's scale, all the proposed solutions share similar techniques that can be divided in physical, statistical, artificial intelligence (AI) and hybrid methods.

Concerning physical models, in [124], three electrical models were compared in terms of energy production in six different days determining the cell temperature through two thermal models and calibrating the I–V curve on manufacturer and measured data. In general, these methods require a detailed set of information for their calibration that is not always easily available. In [125], physical models were compared to hybrid methods based on artificial neural networks combined with clear sky solar radiation data, showing that hybrid models trained on datasets from 10 to 50 days provide comparable results.

Regarding statistical approaches, research activities mainly aims at im-

proving the accuracy of classical regressive (e.g., ARMA, ARIMA, ARMAX, ARIMAX [35, 126]) methods that generally result in a poor accuracy when used for forecasting. Recently, ARMAX models have been applied in [127] showing improved performance. In [128], authors proposed a multiple time-scale data-driven forecasting model for solar irradiance, which is based on a spatio-temporal (ST) and autoregressive with exogenous input (ARX) prediction. It is interesting to observe that the model leverages both spatial and temporal correlations among neighboring solar sites.

AI methods relies on Artificial Neural Networks (ANN) [129], fuzzy predictors [114], evolutionary algorithms [76], and other general machine learning models as Support Vector Machines (SVM) [130]. Comprehensive reviews on the matter are available in literature [131, 132]. In [133], several ANN and Deep Neural Networks (DNN) architectures are investigated and powerful Deep Learning algorithms are introduced in the field of solar power forecast. In [134], a non-parametric machine learning approach based on the Gradient Boosted Regression Tree (GBRT) model is used for multi-site prediction of solar power generation on a very short time horizon ranging from one to six hours. The multi-site framework is adopted to improve the predictions for a given PV plant that will be based on historical data from other PV plants with similar patterns.

Finally, hybrid models are a combination of the previous three approaches [26,31,32]. Recently, the Physical Hybrid Artificial Neural Network (PHANN) [125] has been applied using several datasets and training methods and showing an improved accuracy with respect to classical AI approaches.

The main feature of the majority of the approaches available in literature is their centralized nature [135]. Prediction of PV plants production is usually limited to single plants: local data are processed in a centralized way by a single agent and the prediction is relative to the single plant production. Even when more enhanced algorithms [136–138] are used, resorting as input variables not only to the past values of the produced power but also to several others parameters such as ambient and/or module temperature, solar irradiation, weather classification, the algorithms work always on local data that refer only to the site where the PV plant is installed. In addition, these algorithms usually need complex instrumentation or public/private data sources that are not always available and that limit their applicability. The centralized solution is not feasible in various contexts. For example, in very large plants, different areas of the plant (with different control units) could have

different production, given different conditions (e.g., irradiation, weather differences, faults). Another example is a group of plants operated by a single management and located in different positions in the same region: not every plant will have the same production and data could be not consistent from one another.

Only very recently, more advanced forecasting models have been proposed using data related to multiple plants [128,134,139,140]. These methods try to leverage the spatio-temporal autocorrelation that characterizes different geophysical data, such as weather conditions, to make more accurate predictions. In [139], particular attention is devoted to the correct choice of spatial and temporal statistics for building up an optimum cross-regressive model between PV plants and, in [140], vector autoregression (VAR) and gradient boosting (GB) frameworks are combined to share information among the distributed PV plants and produce an optimal forecast.

In this framework, the idea of this work is to present a novel distributed decentralized prediction technique for the forecasting of the PV power output that relies on the easy share of historical data among different neighbouring PV plants. The technique is mainly intended for asset owners that hold a wide portfolio of PV systems that are usually geographically spread out over large areas. Consequently, the underlying innovative idea is to keep the algorithm manageable for the single agent, which processes only the generated power data that are always available also due to fiscal laws, and to use more agents located in different plants that share only relevant information to minimize communication among the plants. Our method requires only a reliable communication channel that is always present nowadays (i.e., radio communication, telephone line, internet broadband, wireless communication, fiber optic, satellite communication) for operation and maintenance (O&M) purposes.

In our innovative decentralized solution, information from different agents (i.e., different plants or different control units in large plants) is shared between the agents themselves through a connectivity matrix that reflects the capabilities of the real network infrastructure with a prescribed connectivity level; distributed information is used to strengthen the prediction accordingly [141]. Atop of that, since this kind of data tends to be very large, in many occasions is too onerous to move all data from different agents (or collection sites) to a single processing unit. In this context, our novel distributed approach is a suitable solution for the PV ouput prediction problem in large

environments.

Therefore, in the present section we propose the following original contributions:

- We use Echo State Networks (ESNs) for data prediction, which are a class of recurrent neural networks where the central layer is generated in a stochastic fashion but with recurrent connections [53].

- The distributed learning algorithm is based on the Alternating Direction Method of Multiplier (ADMM) optimization procedure [142]. The agents in the prediction network can exchange only information about the data (or its representative) with their neighbors.

- The local agents interact without the need of a coordinator, using the Distributed Average Consensus protocol (DAC) [143].

To show the suitability and asses the performance of the proposed approach, we apply the distributed algorithm to the grid delivered active power data of five different plants located in the Abruzzo region, Italy. The production will be predicted by using different training and consensus conditions and by varying the time horizon from one day to an entire week. Performances are compared with respect to the ones obtained by the centralized version of the ESN algorithm and with the standard local prediction made independently in each plant.

Regarding the theoretic formulation, we present now the elemental theory of the distributed ESN prediction is presented. The DAC and ADMM techniques are explained, and the proposed distributed ESN model is introduced.

In 4.2.5, a detailed description of the ESN is already given, and thus it is not repeated in this section. Instead, in the following, we will extend this approach in such a way that, at the generic time n, the input to the ESN reservoir can be a generalized combination of available past samples that are used to predict the future ones. By the way, the good performances of ESN compared to other benchmarking prediction models on forecasting time series related to the management of energy sources have been recently proved in [144].

For learning applications, it is fundamental for the reservoir to satisfy the 'echo state property' (ESP) [145]. In other words, given an input, its effect on the reservoir state must disappear in a finite number of time steps. Usually,

by rule of thumb, the matrix $\boldsymbol{W}_r^r$ is rescaled, resulting in $\rho(\boldsymbol{W}_r^r) < 1$, where $\rho(\cdot)$ denotes the spectral radius operator.

In the general learning problem, the ESN is trained by feeding the reservoir a sequence of Q desired input-outputs pairs $(\boldsymbol{x}[1], d[1]), \ldots, (\boldsymbol{x}[Q], d[Q])$. The states are then 'gathered' as a sequence $\boldsymbol{h}[1], \ldots, \boldsymbol{h}[Q]$. In this phase, the desired output is used for feedback because the ESN output is not yet available. Defining the hidden matrix $\boldsymbol{H}$ and output vector $\boldsymbol{d}$ as:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{x}^T[1] \; \boldsymbol{h}^T[1] \\ \vdots \\ \boldsymbol{x}^T[1] \; \boldsymbol{h}^T[Q] \end{bmatrix} \tag{4.31}$$

$$\boldsymbol{d} = \begin{bmatrix} f_{\text{out}}^{-1}(d[1]) \\ \vdots \\ f_{\text{out}}^{-1}(d[Q]) \end{bmatrix} \tag{4.32}$$

The output vector is obtained by solving the regularized least-squares problem, resulting in the following formulation for the weight vector:

$$\boldsymbol{w}^* = \arg \min_{\boldsymbol{w} \in \mathbb{R}^{N_i + N_r}} \frac{1}{2} ||\boldsymbol{H}\boldsymbol{w} - \boldsymbol{d}||_2^2 + \frac{\lambda}{2} ||\boldsymbol{w}||_2^2 \,, \tag{4.33}$$

where $\boldsymbol{w} = [\boldsymbol{W}_i^o \, \boldsymbol{W}_r^o]^T$ and $\lambda \in \mathbb{R}^+$ is called the *regularization factor*. A close form solution of problem in (4.33) can be obtained as:

$$\boldsymbol{w}^* = \left(\boldsymbol{H}^T\boldsymbol{H} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{H}^T\boldsymbol{d} \,. \tag{4.34}$$

Given their transient state, in practice it is possible to remove the initial elements from the sequence with which the least-squares problem is solved. The number of these 'dropout' elements is fixed a priori, as discussed in the following.

To be able to implement the ESN in a distributed way, it is necessary to introduce two techniques that can be used for the distribution of information between agents. The problem here is twofold: the agents in the network must exchange information of some sort, sharing it with the connected nodes via a regulated process, and they must reach a global optimum.

If we consider a network of a number L of agents, we can characterize it by means of a connectivity matrix $\boldsymbol{C}$ which describes the fixed a priori connection between the agents. It is a square matrix of dimension L and each element

$C_{ij} \neq 0$ if and only if the two nodes i and j are connected. In this work, the network is supposed to be connected and undirected. Also, we assume that every node has a measurement vector denoted by $\boldsymbol{\theta}_k[0]$, $k = 1 \dots L$.

To address the regulation of information between agents, the DAC network protocol is introduced. It relies solely on local communication between the agents and it is used to calculate the global average based only on the local measures [146–148]. The local update of the protocol, for the iteration n, is:

$$\boldsymbol{\theta}_k[n] = \sum_{j=1}^{L} C_{kj} \boldsymbol{\theta}_j[n-1] \,. \tag{4.35}$$

The whole process is reduced to the global average when the single elements of the $\boldsymbol{C}$ matrix are correctly selected:

$$\lim_{n \to +\infty} \boldsymbol{\theta}_k[n] = \frac{1}{L} \sum_{k=1}^{L} \boldsymbol{\theta}_k[0], \, \forall k \in \{1, 2, \dots, L\} \,. \tag{4.36}$$

The DAC can be halted either when a threshold is reached for the δ or when the maximum number of iterations is completed. For the convergence, the 'max-degree' weights [148] can be chosen:

$$C_{kj} = \begin{cases} \frac{1}{d+1} & \text{if } k \text{ is connected to } j \\ 1 - \frac{d_k}{d+1} & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \,, \tag{4.37}$$

where d_k is the degree of node k, and d is the maximum degree of the network.

To ensure the reaching of an optimum for all the agents, the ADMM is brought in. It is a procedure used for optimization problems which can be posed as [149]:

$$\begin{aligned} &\underset{\boldsymbol{s} \in \mathbb{R}^d, \boldsymbol{z} \in \mathbb{R}^l}{\text{minimize}} \;\; f(\boldsymbol{s}) + g(\boldsymbol{z}) \\ &\text{subject to} \;\; \boldsymbol{A}\boldsymbol{s} + \boldsymbol{B}\boldsymbol{z} + \boldsymbol{c} = 0, \end{aligned} \tag{4.38}$$

where $\boldsymbol{A} \in \mathbb{R}^{p \times d}$, $\boldsymbol{B} \in \mathbb{R}^{p \times l}$ and $\boldsymbol{c} \in \mathbb{R}^p$. To solve it, we can rely on the

augmented Lagrangian given by:

$$\mathcal{L}_\rho(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{t}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{t}^T (\boldsymbol{A}\boldsymbol{s} + \boldsymbol{B}\boldsymbol{z} + \boldsymbol{c}) + \\ + \frac{\rho}{2} \|\boldsymbol{A}\boldsymbol{s} + \boldsymbol{B}\boldsymbol{z} + \boldsymbol{c}\|^2, \tag{4.39}$$

where $\boldsymbol{t} \in \mathbb{R}^p$ is the vector of Lagrange multipliers, ρ is a scalar, and the last term is added to ensure differentiability and convergence [149]. The solution of the problem in (4.38) can be achieved by iterating these steps:

$$\boldsymbol{s}[n+1] = \arg\min_{\boldsymbol{s}} \{\mathcal{L}_\rho(\boldsymbol{s}, \boldsymbol{z}[n], \boldsymbol{t}[n])\} \tag{4.40}$$

$$\boldsymbol{z}[n+1] = \arg\min_{\boldsymbol{z}} \{\mathcal{L}_\rho(\boldsymbol{s}[n+1], \boldsymbol{z}, \boldsymbol{t}[n])\} \tag{4.41}$$

$$\boldsymbol{t}[n+1] = \boldsymbol{t}[n] + \rho (\boldsymbol{A}\boldsymbol{s}[n+1] + \boldsymbol{B}\boldsymbol{z}[n+1] + \boldsymbol{c}) \tag{4.42}$$

Results on the convergence of ADMM are in [149, Sect. 3.2]. It is common practice to stop ADMM after a maximum number of iterations, or when the two residuals are found under given thresholds.

Now, we describe the proper distributed algorithm for the ESN. In general terms, where a pure decentralized algorithm is necessary, we suppose to work with a dataset S which is distributed among the agents of the network. As said earlier, we are working here with multiple sets of data, all pertaining to time series of the same physical quantity taken at different sites. Each plant is considered as a single agent of the network. We suppose that every agent sets the choice of the matrices $\boldsymbol{W}_i^r$, $\boldsymbol{W}_r^r$ and $\boldsymbol{W}_o^r$.

Let $\boldsymbol{H}_k$ and $\boldsymbol{d}_k$ be the hidden matrices and output vectors, which are computed at the kth node according to (4.31)-(4.32) with its local dataset. In this case, extending the result in (4.33), the global optimization problem can be stated as:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w} \in \mathbb{R}^{N_i + N_r}}{\arg\min} \frac{1}{2} \left(\sum_{k=1}^{L} \|\boldsymbol{H}_k \boldsymbol{w} - \boldsymbol{d}_k\|^2 \right) + \frac{\lambda}{2} \|\boldsymbol{w}\|^2 . \tag{4.43}$$

In our case, the reservoir resulting from the solution of the distributed least-squares problem is large in size. Thus, the communication of the matrices $\boldsymbol{H}_k^T \boldsymbol{H}_k$ is impractical and a probable network bottleneck. The problem can be reformulated and then solved with the efficient use of the ADMM, introducing local variables $\boldsymbol{w}_k$ for every agent and forcing them to converge to the same value:

$$\underset{\boldsymbol{z},\boldsymbol{w}_1,\ldots,\boldsymbol{w}_L\in\mathbb{R}^{N_i+N_r}}{\text{minimize}} \quad \frac{1}{2}\left(\sum_{k=1}^{L}\|\boldsymbol{H}_k\boldsymbol{w}_k-\boldsymbol{d}_k\|^2\right)+\frac{\lambda}{2}\|\boldsymbol{z}\|^2 \tag{4.44}$$

$$\text{subject to} \;\; \boldsymbol{w}_k=\boldsymbol{z},\; k=1\ldots L\,. \tag{4.45}$$

The augmented Lagrangian of problem in (4.45) is given by:

$$\begin{aligned}\mathcal{L}_\rho(\cdot)=\frac{1}{2}\left(\sum_{k=1}^{L}\|\boldsymbol{H}_k\boldsymbol{w}_k-\boldsymbol{d}_k\|^2\right)+\frac{\lambda}{2}\|\boldsymbol{z}\|^2+\\ +\sum_{k=1}^{L}\boldsymbol{t}_k^T(\boldsymbol{w}_k-\boldsymbol{z})+\frac{\gamma}{2}\sum_{k=1}^{L}\|\boldsymbol{w}_k-\boldsymbol{z}\|^2\,,\end{aligned} \tag{4.46}$$

The updates for $\boldsymbol{w}_k[n+1]$ and $\boldsymbol{z}[n+1]$ can be computed in closed form as:

$$\boldsymbol{w}_k[n+1]=\left(\boldsymbol{H}_k^T\boldsymbol{H}_k+\gamma\boldsymbol{I}\right)^{-1}\left(\boldsymbol{H}_k^T\boldsymbol{d}_k-\boldsymbol{t}_k[n]+\gamma\boldsymbol{z}[n]\right)\,, \tag{4.47}$$

$$\boldsymbol{z}[n+1]=\frac{\gamma\hat{\boldsymbol{w}}+\hat{\boldsymbol{t}}}{\lambda/L+\gamma}\,, \tag{4.48}$$

where we introduced the averages $\hat{\boldsymbol{w}} = \frac{1}{L}\sum_{k=1}^{L}\boldsymbol{w}_k[n+1]$ and $\hat{\boldsymbol{t}} = \frac{1}{L}\sum_{k=1}^{L}\boldsymbol{t}_k[n]$. These averages can be computed in a decentralized fashion using a DAC step. The formula in (4.42) instead simplifies to:

$$\boldsymbol{t}_k[n+1]=\boldsymbol{t}_k[n]+\gamma\left(\boldsymbol{w}_k[n+1]-\boldsymbol{z}[n+1]\right)\,. \tag{4.49}$$

Equations (4.47) and (4.49) can be computed locally at every node, leading to a purely distributed implementation of the general algorithm. In this framework the communication is carried on solely with the use of the DAC protocol.

Hence, the overall algorithm can be implemented in a purely decentralized fashion, where communication is restricted to the use of the DAC protocol. In cases where, on a node, the number of training samples is lower than N_r+N_i, we can exploit the matrix inversion lemma to obtain a more convenient matrix inversion step [150]:

$$\left(\boldsymbol{H}_k^T\boldsymbol{H}_k+\gamma\boldsymbol{I}\right)^{-1}=\gamma^{-1}\left[\boldsymbol{I}-\boldsymbol{H}_k^T\left(\gamma\boldsymbol{I}+\boldsymbol{H}_k\boldsymbol{H}_k^T\right)\boldsymbol{H}_k\right]\,. \tag{4.50}$$

The pseudocode of the proposed algorithm described so far, for training a

distributed ESN with DAC and ADMM procedures, is reported in Algorithm 1, with the details on all the implementation steps.

Algorithm 1 Pseudocode of the Distributed ESN Algorithm with DAC and ADMM Procedures

Input Given a network of L agents and a training set of L different time series with length N. Given numerical parameters of the algorithm (λ, γ, ρ, etc.) and the maximum number of iterations N_{iter}.

Initialize $\boldsymbol{t}_k[0]$ and $\boldsymbol{z}[0]$.

Initialize local ESN for each node as in (4.31)-(4.34).

Compute the inverted matrix $\left(\boldsymbol{H}_k^T \boldsymbol{H}_k + \gamma \boldsymbol{I}\right)^{-1}$ using (4.50) when necessary.

for $n = 1$ to N_{iter} **do**

Update Compute current weights $\boldsymbol{w}_k$ for every node using (4.47).

Consensus step. Run the DAC protocol as in (4.35)-(4.37) and compute the averages $\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{t}}$.

Update. Compute current $\boldsymbol{z}$ using (4.48).

Update. Compute current new state $\boldsymbol{t}_k$ for every node using (4.49).

Compute the ADMM residuals and **stop** if any convergence criterion is satisfied.

Return as the final readout parameters on each node the latest estimation of $\hat{\boldsymbol{w}}$.

The proposed cooperative prediction approach will be applied to five PV plants of equal rated nominal power of 998.20 kWp. All of them are located in the Italian region named 'Abruzzo', in the center of Italy, on the east coast. The geographic position of the PV plants is shown in Fig. 4.31 while the geographic coordinates are reported in Table 4.28. The PV plants are located along a line that extends from North ('Arielli') to South ('Dogliola') for around 50 km and they are approximately equidistant from each other.

Each time series is the output power from one of the five solar plants in the year 2015, with a sample interval of 15 minutes (quarter-hour production curves). Before applying the learning procedures, the samples of time series are normalized by using a linear mapping in the range between -1 and 1 for regularization purpose, assuming as extremes for normalization the physical operation of plants; i.e., -1 will correspond to 0 kW and +1 to 1000 kW.

We used a training set of 30 consecutive days for every test; the training set contains the known samples that are used to forecast the future ones. The

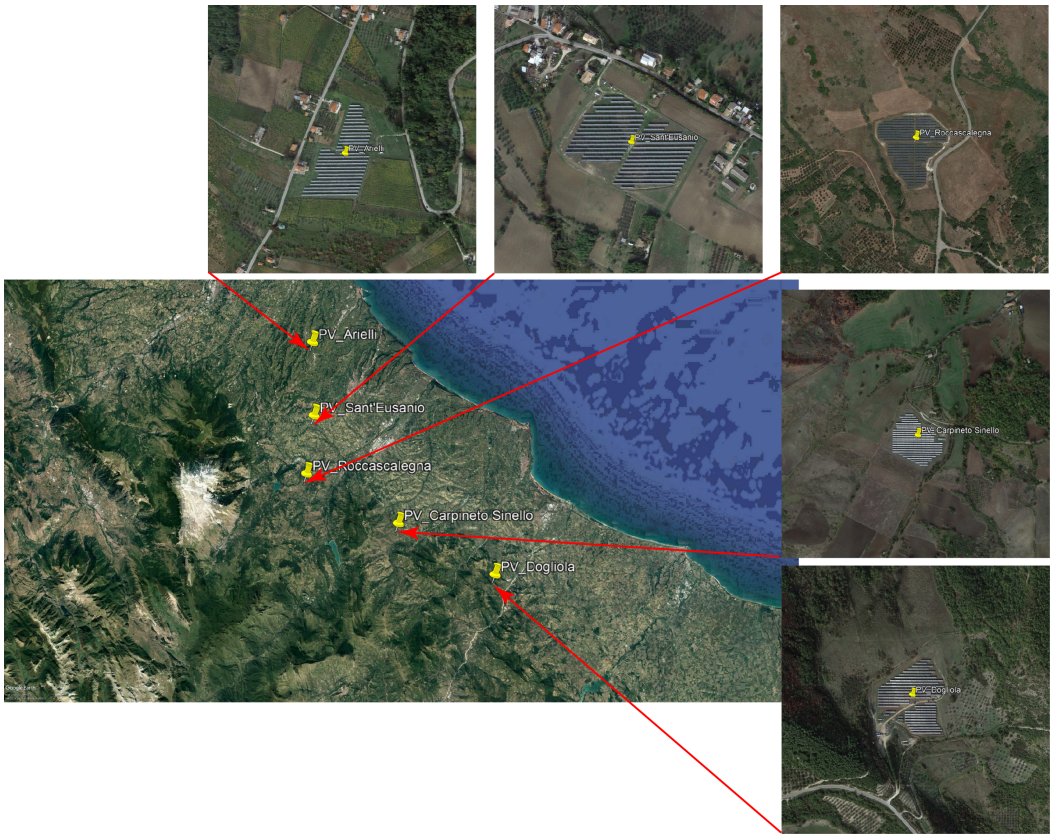


Figure 4.31: Aerophoto showing the locations of the five PV plants.

Table 4.28: List of PV Plants with Their Geographic Coordinates

PV Plant Name	**Latitude**	**Longitude**
Arielli	42°16′22.69″ N	14°18′49.23″ E
Sant'Eusanio del Sangro	42°10′24.09″ N	14°18′59.16″ E
Roccascalegna	42°5′40.05″ N	14°18′9.84″ E
Carpineto Sinello	42°1′37.06″ N	14°28′13.19″ E
Dogliola	41°57′25.67″ N	14°38′47.71″ E

latter are associated with test sets having three different lengths: 1-day, 3-days, and 7-days after the last available sample of the training set. Actually, finding the optimal size of training data is an open issue that is crucial in time series forecasting. A theoretical trend would lead to extend the size as much as possible in order to exploit all the available information. However, in this case one has to deal with the increasing computational cost related to

very large training sets as well as with the non-stationarity and seasonality of observed data, for which to limit the temporal extension in the past of the used samples can improve the performances. It should be observed that, due to the high memory capacity of data loggers of the modern PV monitoring systems, there is practically no limit to the length of the training set that can contain all the available registered data. Looking for a good trade-off in this regard, we found the use of 30 days before the samples to be tested; more investigations on this specific problem could be faced in future research works.

In order to solve the problem of zero solar irradiation, we have considered the geographical data for each plant computing the sunrise and sunset times [151] and therefore the ESN is forced to zero output during the night period.

All of the experiments described in the following were carried out using MATLAB R2017a on a machine equipped with an Intel® Core™ i7-2600 64-bit CPU at 3.40 GHZ and with 16 GB of RAM.

Adopted Algorithms

The tests are carried out considering a network of $L = 5$ agents, each corresponding to a single plant. The data communication network among plants is chosen randomly with a 0.75% degree of connectivity, reflecting the tentative capabilities of the network infrastructure. The resulting graph, where each plant is a node, is therefore connected but not complete. We compared three different algorithms:

- Centralized ESN (C-ESN): this option simulates the case where all data is gathered at a single location and the straightforward ESN prediction is applied by solving (4.33). It should be observed that the C-ESN has only a theoretical benchmark purpose, since it is unfeasible from a practical point of view, requiring the transmission of all the data collected in the peripheral PV sites to a central location.

- Local ESN (L-ESN): this is the case where data is indeed distributed but there is no communication in the network, so every agent trains a single ESN from its subset of data. This corresponds to predictions made in each plant independently of each other.

- Distributed ESN (D-ESN): This is the actual distributed ESN described in earlier. We set $\rho = 0.001$, a maximum number of 500 iterations, and

$\epsilon_{\text{abs}} = \epsilon_{\text{rel}} = 10^{-4}$.

For the numerical evaluation of performances, we used the common Root-Mean-Square error (RMSE) measure, which is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum\nolimits_n \Big(y[n] - d[n]\Big)^2}, \tag{4.51}$$

where n spans over the T samples of the test set. The first 100 predicted samples were dropped out as they contain transient effects related to the dynamical recurrent ESN model.

ESNs are proved to be very robust to overfitting phenomena. However, overfitting is possible for two main reasons: the size of reservoir N_r; the LSE regularization factor λ. In this regard, a grid search procedure can be adopted by using data in the training set only (i.e., known samples of the time series only), so as to set in advance the best size of the reservoir and the best regularization factor. We preliminary tested different sizes of the reservoir in the range $N_r = \{50, 100, 150, \ldots, 500\}$ and different values of $\lambda = 2^j$, $j = \{-10, -9, \ldots, -1, 0, 1, \ldots, 9, 10\}$. There were some slight changes in the results considering different data on different plants, however we adopted a same choice for each and every plant and prediction test, in order to prove that overfitting is prevented by using an objective and replicable criterion that can be set in advance, without relying on specific skills or tricking for each time series. The adopted values are $N_r = 50$, with reservoir matrices randomly initialized, and $\lambda = 2^{-9}$.

Each ADMM iterates a maximum number of 500 times as for the DAC procedure, whose termination threshold was set to 10^{-6}. Consequently, each test was composed by 10 runs, each corresponding to a different random initialization of the ESN weights and topology. Both average and standard deviation of RMSE are reported accordingly.

As mentioned earlier, we can generalize the combination of past samples that are chosen to feed the reservoir of the ESN. In order to cope with the intrinsic periodicity of PV time series, we propose to rewrite the generic input vector in (4.14) as:

$$\boldsymbol{x}[n - N_i + 1] = \Big[S[n]\ S[n-T]\ \ldots\ S[n-T(D-1)]\Big]^T. \tag{4.52}$$

This approach is novel for ESN and relies on the assumption that not all of the past samples of the time series carry information for predicting the future

ones. Since the relation between predicted and past samples is not given, we can try to select the relevant samples by embedding the time series [55]. This is done by choosing two parameters, the embedding dimension D and the time lag T, in such a way the reconstructed state of the underlying unknown system, observable only through its output $S[n]$, can be estimated at time n as in (4.52). At the present stage no attempt is done to catch the cyclicity in the PV output data. Future research activities will focus on tailoring appropriate detrending strategies to remove the non-stationary trend lying in the PV output time series. Anyway, it has been demonstrated that ESN usually performs better than other approaches even for non-stationary data.

Looking at (4.14), it is evident that the classical ESN input is an estimate of the unfolded, reconstructed state of the unknown system that generates the observed time series, having assumed $T = 1$ and $D = N_i$. In this more general context, these parameters can be estimated in an optimal way by using the Average Mutual Information (AMI) procedure for T and the False Nearest Neighbors (FNN) procedure for D [56].

Although numerical experiments had been carried out extensively for several periods of 2015, for the sake of illustration we report in the following a case study that is representative of the general behavior obtained by using the proposed approach. The tests are relative to six days of 2015, which were chosen for showing a set of days with variable weather conditions.

The test set always starts in the mid of February, April, June, August, October, and December 2015. It is composed by the 15th day of the month and, possibly, by the successive 2 and 6 days after it thus resulting in the 1-day, 3-days, and 7-days test set, respectively. The prediction distance is set accordingly, that is $m = 4 \cdot 24 = 96$ for the 1-day test set, $m = 4 \cdot 24 \cdot 3 = 288$ for the 3-days test set, $m = 4 \cdot 24 \cdot 7 = 672$ for the 7-days test set.

The training set is always composed by the 30 days prior to the 15th day of the considered month, plus further samples before due to the adjustments for using a different prediction distance. The embedding algorithms AMI and FNN applied to the samples of the training set, as previously discussed, yielded coherent results. In fact, we obtained $T = 4$ and hence, the time series is inherently subsampled, as a sample every 1 hour is enough to carry on the sufficient information. Also, we obtained as optimal embedding dimension $D = 24$, that is the ESN is fed by the inputs of one entire day. This result seems to be coherent with the cyclicity of the observed time series, which is basically due to the daily periodicity of sun irradiation that is caught by using

24 samples for an entire day.

The RMSE results for the three algorithms applied to the 1-day test set are reported in Table 4.29. On each run, once the ESN has been trained by using one of the considered algorithms, the network is used to forecast the time series of the output power of each plant with a related error. The total error over all of the plants in also computed and reported in this Table and in the following ones. In the Table there are reported both mean and standard deviation of the previous values over the 10 runs carried out after different (random) weight initializations and network topologies.

The global performance of D-ESN is always better than the classical L-ESN, with a gain in terms of reduced RMSE up to 4% in some days of the year. It is important to outline that the performance of D-ESN on each plant is sometimes equivalent to L-ESN but, most of the times, the proposed distributed approach outperforms considerably the local one, with a reduced RMSE up to 15% in some plants.

A visual examination of the three predictions can be done for a single example related to Plant 3 in the month of December 2015, as reported in Fig. 4.32, Fig. 4.33, and Fig. 4.34, respectively, conveying the same conclusions.

The numerical results of the 3-days test set are reported in Table 4.30 and they confirm the previous behavior, with D-ESN that performs better than L-ESN and it achieves RMSE values that are comparable with the ones obtained by the centralized approach. This is also confirmed by analyzing the plots in Fig. 4.35, Fig. 4.36, and Fig. 4.37, which are still related to Plant 3 in the month of December 2015.

Very similar results are obtained accordingly also for the 7-days test set, considering the numerical performances on Table 4.31 and the graphical results in Fig. 4.38, Fig. 4.39, and Fig. 4.40 for Plant 3 in December 2015. We note that the performance on each plant, as well as the global one, remains stable although the prediction horizon has been increased to 7 days, that is for much more samples than in the previous tests.

In the light of these results, it is worthy to observe that the proposed technique gives the best results for any range from 1 to 7 days ahead, so as to become a valuable tool for companies that trade energy, especially in the day-ahead market. From a practical point of view, the proposed D-ESN can achieve a better prediction accuracy, with an RMSE around 4% and up to 15% lower than L-ESN and even comparable with C-ESN, stating that

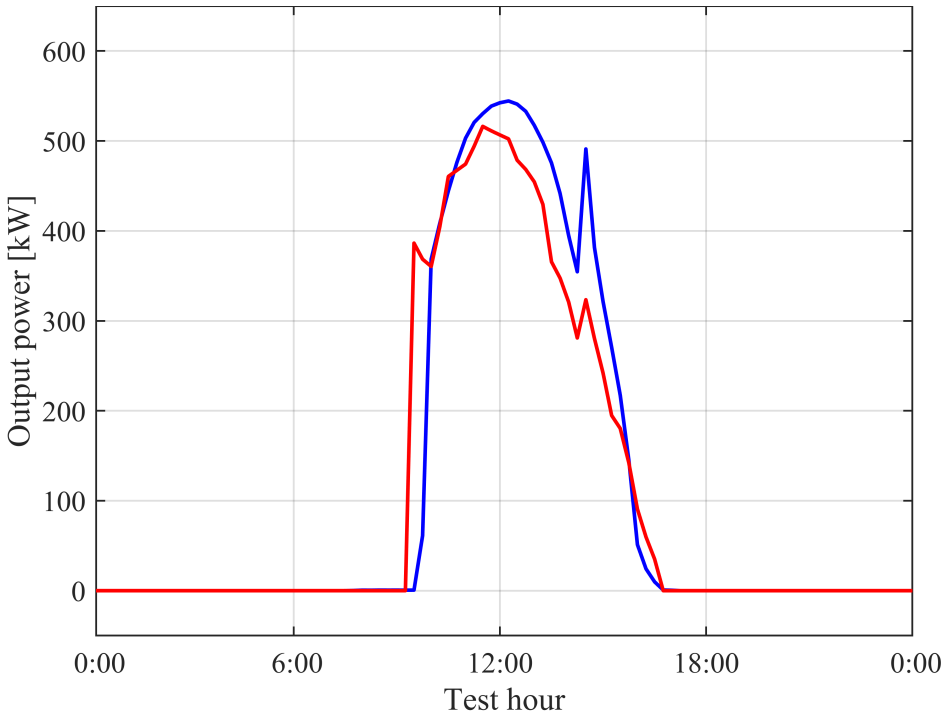


Figure 4.32: Predicted (red) and real (blue) value of the time series at Plant 3 in the mid of December 2015, by using C-ESN and 1-day test set.

the proposed approach produces very good forecasts in the long-term. It is important to remark that all the previous results are obtained by using a very high prediction step m, which is unusual in other forecasting applications but necessary in this case, a fact that makes more valuable the good results obtained anyway.

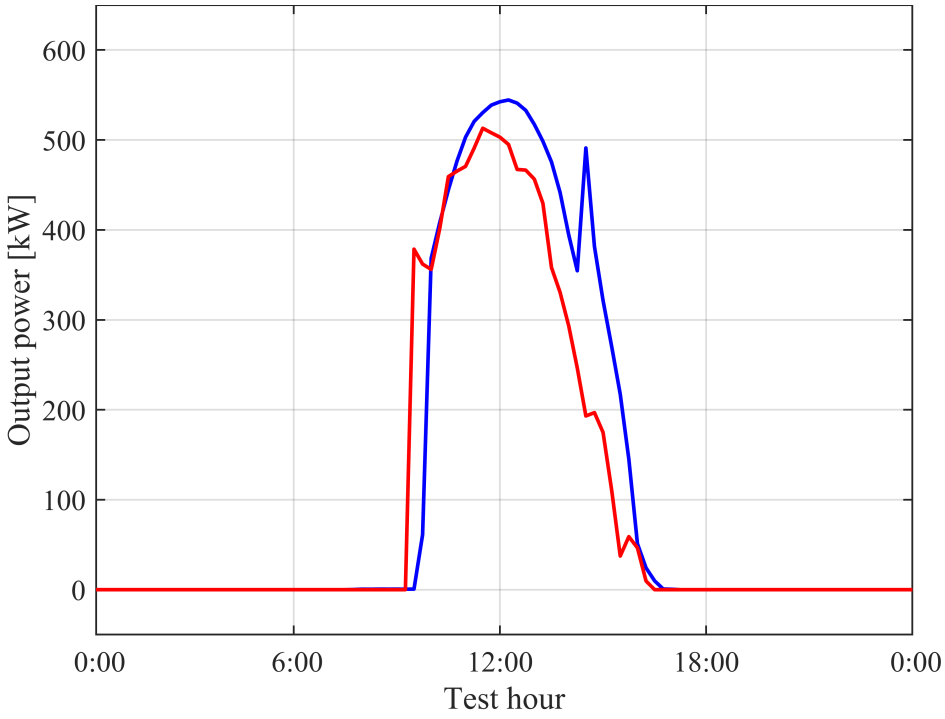


Figure 4.33: Predicted (red) and real (blue) value of the time series at Plant 3 in the mid of December 2015, by using L-ESN and 1-day test set.

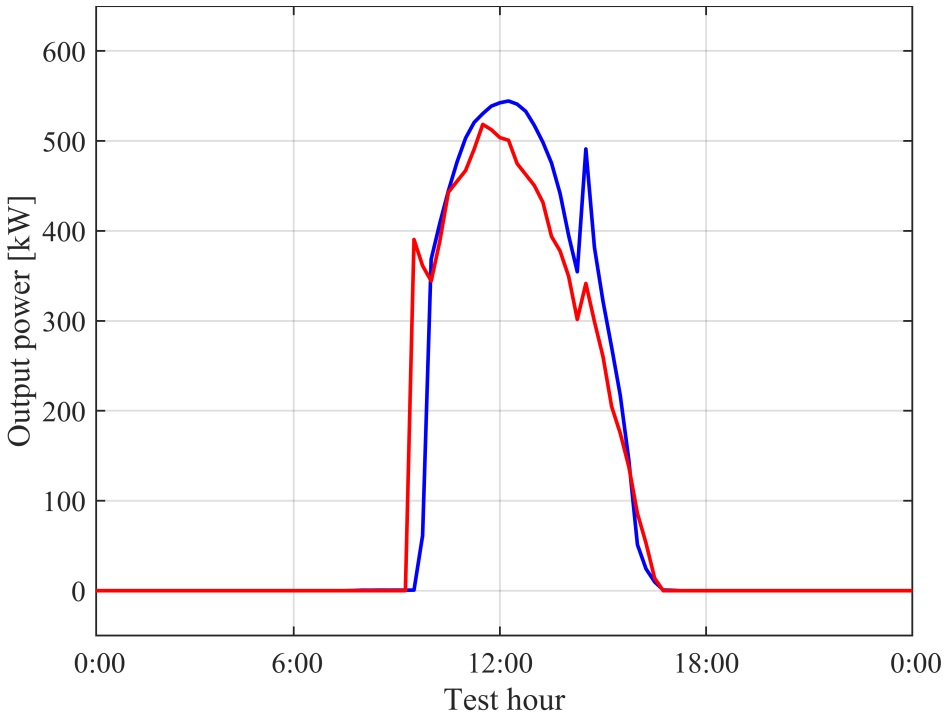


Figure 4.34: Predicted (red) and real (blue) value of the time series at Plant 3 in the mid of December 2015, by using D-ESN and 1-day test set.

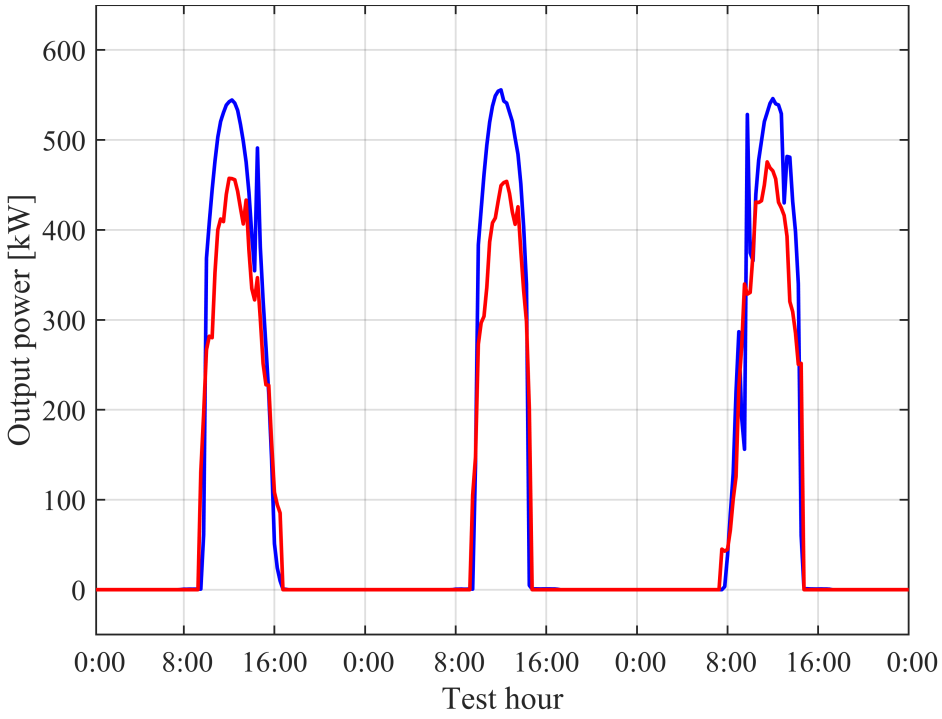


Figure 4.35: Predicted (red) and real (blue) value of the time series at Plant 3 in the mid of December 2015, by using C-ESN and 3-days test set.

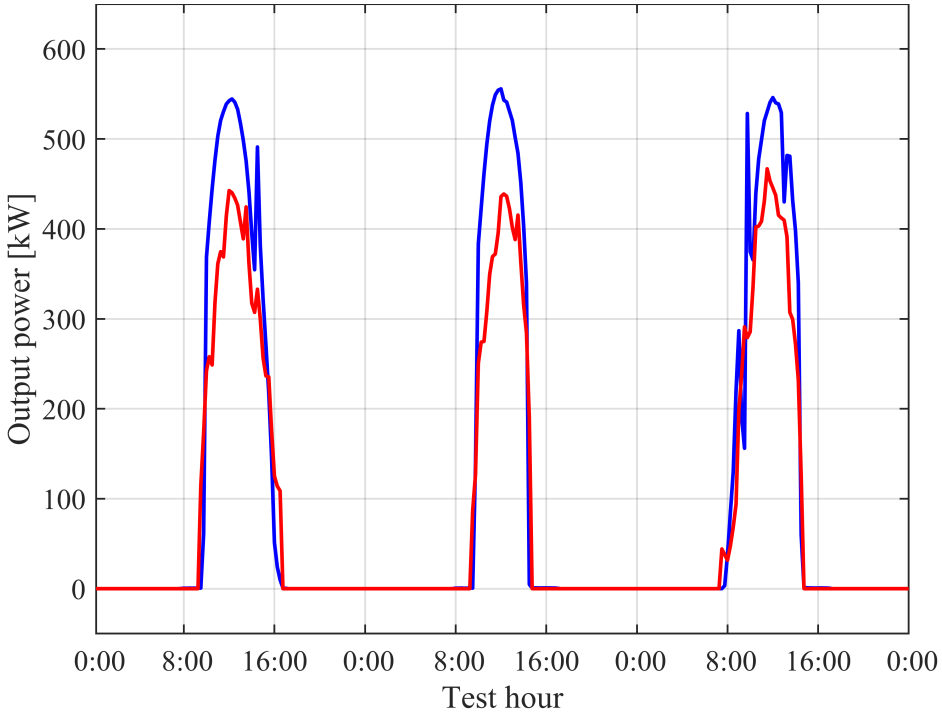


Figure 4.36: Predicted (red) and real (blue) value of the time series at Plant 3 in the mid of December 2015, by using L-ESN and 3-days test set.

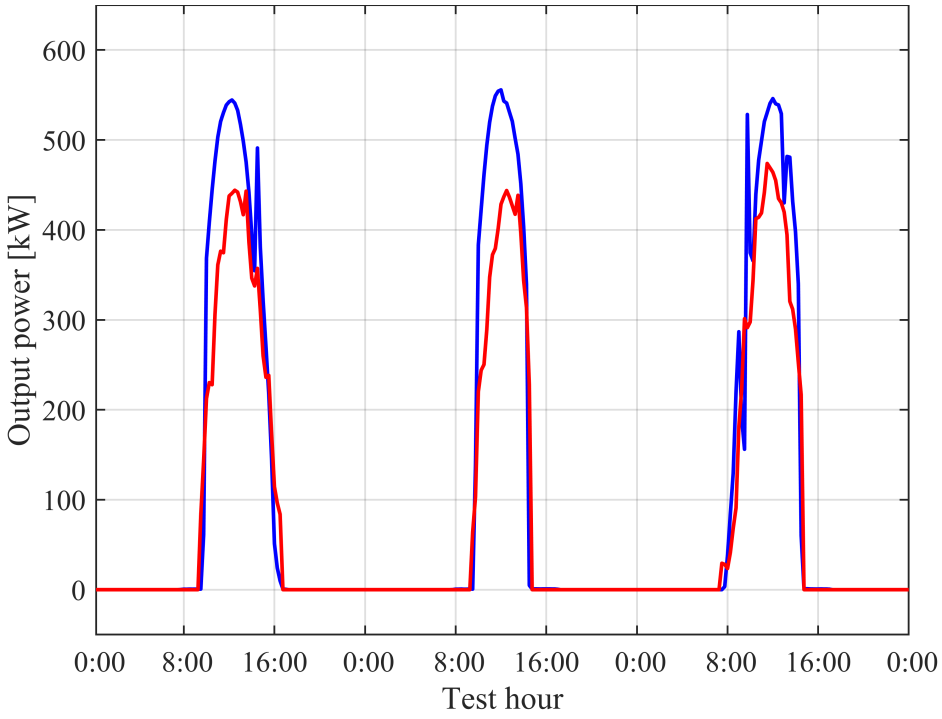


Figure 4.37: Predicted (red) and real (blue) value of the time series at Plant 3 in the mid of December 2015, by using D-ESN and 3-days test set.

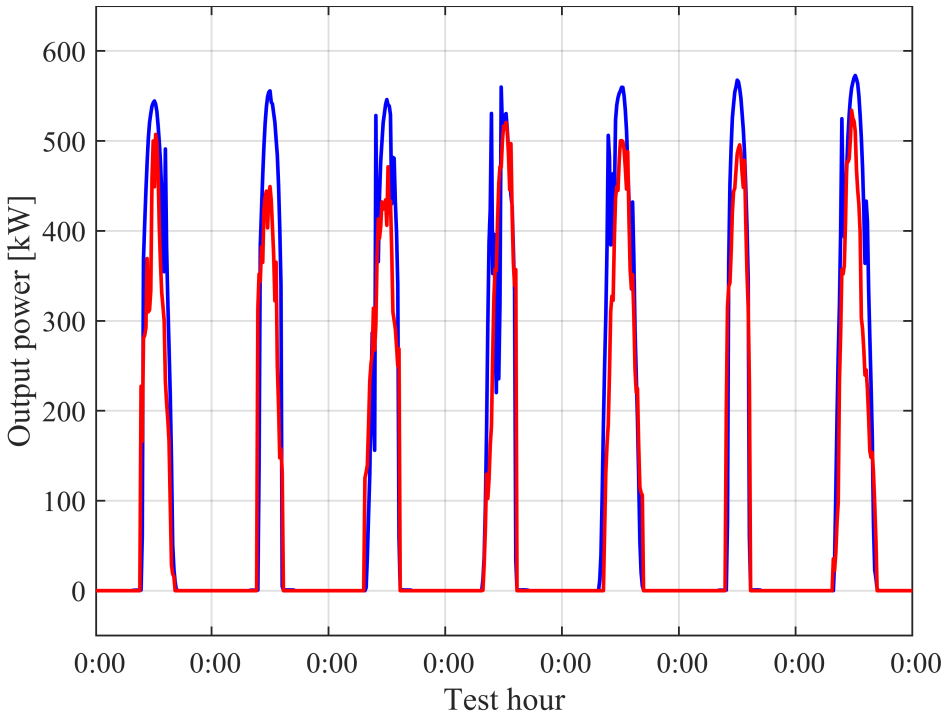


Figure 4.38: Predicted (red) and real (blue) value of the time series at Plant 3 in the mid of December 2015 by using C-ESN and 7-days test set.

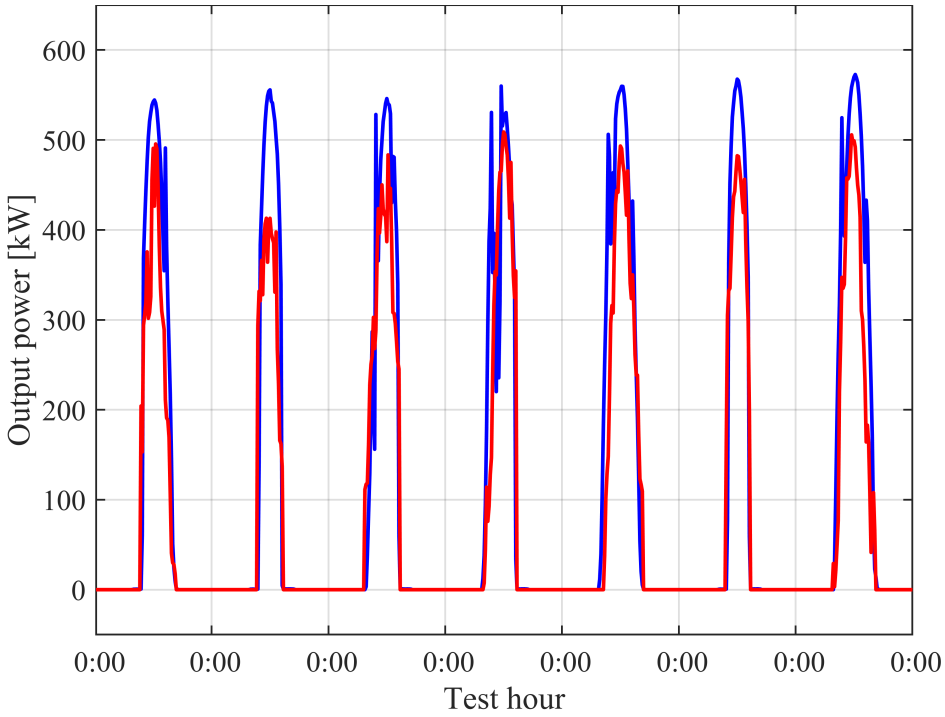


Figure 4.39: Predicted (red) and real (blue) value of the time series at Plant 3 in the mid of December 2015 by using L-ESN and 7-days test set.

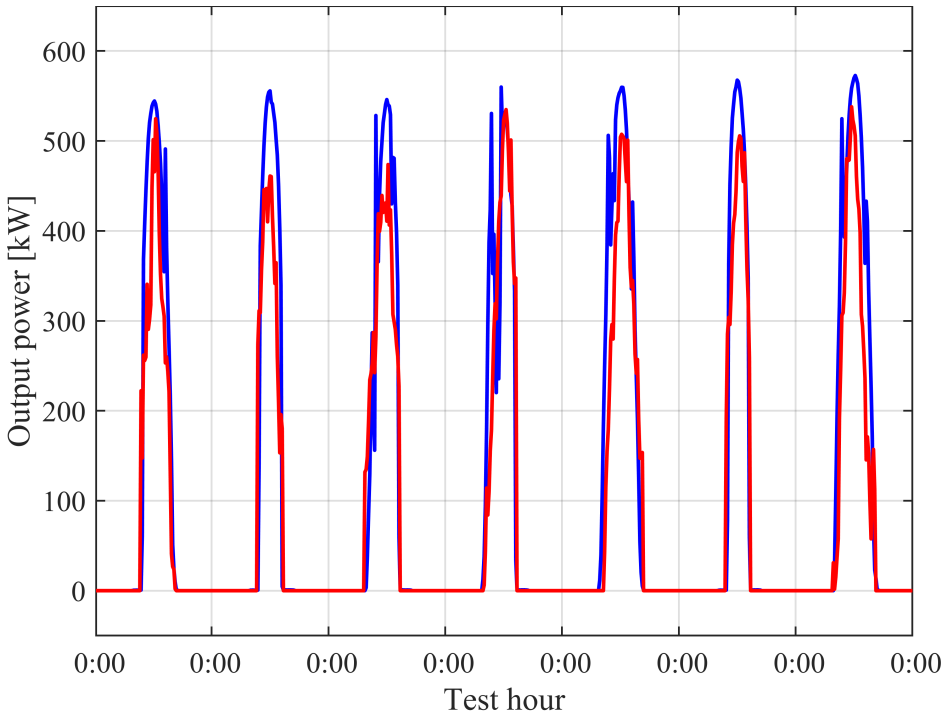


Figure 4.40: Predicted (red) and real (blue) value of the time series at Plant 3 in the mid of December 2015 by using D-ESN and 7-days test set.

Table 4.29: RMSE of Each Plant and Average Result for 1-day Test Set and Different Algorithms

Month	**Algorithm**	**Plant 1**	**Plant 2**	**Plant 3**	**Plant 4**	**Plant 5**	**Total**
February	C-ESN	0.319 ± 0.086	0.284 ± 0.115	0.247 ± 0.078	0.300 ± 0.128	0.248 ± 0.077	0.280 ± 0.097
	L-ESN	0.347 ± 0.120	0.268 ± 0.109	0.266 ± 0.102	0.301 ± 0.145	0.243 ± 0.045	0.285 ± 0.103
	D-ESN	0.311 ± 0.053	0.267 ± 0.066	0.243 ± 0.047	0.290 ± 0.074	0.243 ± 0.045	0.273 ± 0.056
April	C-ESN	0.180 ± 0.009	0.324 ± 0.005	0.213 ± 0.007	0.218 ± 0.007	0.165 ± 0.009	0.220 ± 0.002
	L-ESN	0.198 ± 0.009	0.327 ± 0.006	0.226 ± 0.010	0.221 ± 0.008	0.155 ± 0.007	0.225 ± 0.003
	D-ESN	0.174 ± 0.009	0.319 ± 0.012	0.227 ± 0.007	0.229 ± 0.009	0.155 ± 0.007	0.221 ± 0.003
June	C-ESN	0.306 ± 0.009	0.291 ± 0.006	0.216 ± 0.009	0.233 ± 0.007	0.244 ± 0.010	0.258 ± 0.006
	L-ESN	0.318 ± 0.011	0.298 ± 0.008	0.245 ± 0.007	0.223 ± 0.009	0.242 ± 0.011	0.265 ± 0.006
	D-ESN	0.301 ± 0.013	0.280 ± 0.009	0.201 ± 0.013	0.222 ± 0.011	0.242 ± 0.011	0.249 ± 0.009
August	C-ESN	0.237 ± 0.008	0.218 ± 0.011	0.201 ± 0.006	0.187 ± 0.011	0.180 ± 0.005	0.205 ± 0.004
	L-ESN	0.282 ± 0.010	0.214 ± 0.012	0.212 ± 0.010	0.185 ± 0.010	0.189 ± 0.005	0.216 ± 0.005
	D-ESN	0.219 ± 0.004	0.215 ± 0.009	0.214 ± 0.006	0.193 ± 0.011	0.189 ± 0.005	0.208 ± 0.002
October	C-ESN	0.277 ± 0.007	0.175 ± 0.007	0.236 ± 0.007	0.227 ± 0.009	0.195 ± 0.007	0.222 ± 0.005
	L-ESN	0.273 ± 0.011	0.200 ± 0.008	0.249 ± 0.009	0.249 ± 0.006	0.210 ± 0.007	0.236 ± 0.005
	D-ESN	0.277 ± 0.007	0.173 ± 0.007	0.246 ± 0.006	0.244 ± 0.009	0.210 ± 0.007	0.234 ± 0.005
December	C-ESN	0.106 ± 0.004	0.142 ± 0.004	0.141 ± 0.005	0.100 ± 0.003	0.121 ± 0.002	0.122 ± 0.002
	L-ESN	0.118 ± 0.008	0.140 ± 0.002	0.148 ± 0.006	0.106 ± 0.005	0.121 ± 0.003	0.127 ± 0.002
	D-ESN	0.101 ± 0.005	0.147 ± 0.006	0.149 ± 0.006	0.105 ± 0.005	0.121 ± 0.003	0.125 ± 0.003

Table 4.30: RMSE of Each Plant and Average Result for 3-days Test Set and Different Algorithms

Month	**Algorithm**	**Plant 1**	**Plant 2**	**Plant 3**	**Plant 4**	**Plant 5**	**Total**
February	C-ESN	0.310 ± 0.008	0.270 ± 0.006	0.304 ± 0.009	0.323 ± 0.008	0.270 ± 0.006	0.296 ± 0.007
	L-ESN	0.317 ± 0.008	0.270 ± 0.005	0.312 ± 0.012	0.320 ± 0.009	0.269 ± 0.004	0.298 ± 0.007
	D-ESN	0.309 ± 0.007	0.270 ± 0.004	0.301 ± 0.008	0.320 ± 0.006	0.269 ± 0.004	0.294 ± 0.005
April	C-ESN	0.230 ± 0.020	0.196 ± 0.025	0.178 ± 0.023	0.256 ± 0.008	0.177 ± 0.017	0.208 ± 0.008
	L-ESN	0.238 ± 0.041	0.195 ± 0.018	0.198 ± 0.042	0.262 ± 0.010	0.183 ± 0.006	0.215 ± 0.015
	D-ESN	0.236 ± 0.011	0.190 ± 0.020	0.185 ± 0.008	0.257 ± 0.014	0.183 ± 0.006	0.210 ± 0.006
June	C-ESN	0.214 ± 0.005	0.199 ± 0.003	0.231 ± 0.004	0.216 ± 0.004	0.236 ± 0.004	0.219 ± 0.003
	L-ESN	0.223 ± 0.006	0.210 ± 0.003	0.242 ± 0.005	0.206 ± 0.004	0.231 ± 0.005	0.222 ± 0.003
	D-ESN	0.212 ± 0.007	0.194 ± 0.004	0.224 ± 0.007	0.208 ± 0.007	0.231 ± 0.005	0.214 ± 0.004
August	C-ESN	0.219 ± 0.003	0.214 ± 0.005	0.246 ± 0.005	0.227 ± 0.004	0.241 ± 0.009	0.229 ± 0.004
	L-ESN	0.224 ± 0.004	0.229 ± 0.006	0.260 ± 0.008	0.223 ± 0.006	0.251 ± 0.008	0.237 ± 0.004
	D-ESN	0.225 ± 0.003	0.215 ± 0.005	0.246 ± 0.003	0.233 ± 0.004	0.251 ± 0.008	0.234 ± 0.003
October	C-ESN	0.148 ± 0.005	0.240 ± 0.006	0.345 ± 0.003	0.354 ± 0.004	0.311 ± 0.005	0.279 ± 0.002
	L-ESN	0.160 ± 0.005	0.240 ± 0.008	0.341 ± 0.002	0.360 ± 0.005	0.312 ± 0.005	0.283 ± 0.002
	D-ESN	0.154 ± 0.006	0.239 ± 0.005	0.349 ± 0.003	0.359 ± 0.005	0.312 ± 0.005	0.283 ± 0.002
December	C-ESN	0.228 ± 0.001	0.220 ± 0.001	0.219 ± 0.001	0.190 ± 0.001	0.183 ± 0.002	0.208 ± 0.001
	L-ESN	0.234 ± 0.003	0.221 ± 0.001	0.238 ± 0.002	0.198 ± 0.003	0.181 ± 0.003	0.214 ± 0.001
	D-ESN	0.229 ± 0.004	0.215 ± 0.002	0.213 ± 0.003	0.185 ± 0.003	0.181 ± 0.003	0.205 ± 0.003

Table 4.31: RMSE of Each Plant and Average Result for 7-days Test Set and Different Algorithms

Month	**Algorithm**	**Plant 1**	**Plant 2**	**Plant 3**	**Plant 4**	**Plant 5**	**Total**
February	C-ESN	0.284 ± 0.006	0.287 ± 0.004	0.263 ± 0.004	0.302 ± 0.006	0.230 ± 0.006	0.273 ± 0.005
	L-ESN	0.291 ± 0.008	0.286 ± 0.004	0.270 ± 0.006	0.302 ± 0.007	0.229 ± 0.003	0.276 ± 0.004
	D-ESN	0.280 ± 0.002	0.291 ± 0.005	0.262 ± 0.003	0.296 ± 0.004	0.229 ± 0.003	0.272 ± 0.002
April	C-ESN	0.267 ± 0.011	0.289 ± 0.009	0.257 ± 0.009	0.338 ± 0.012	0.295 ± 0.008	0.289 ± 0.010
	L-ESN	0.273 ± 0.015	0.294 ± 0.010	0.271 ± 0.010	0.332 ± 0.016	0.287 ± 0.013	0.291 ± 0.008
	D-ESN	0.259 ± 0.013	0.282 ± 0.010	0.251 ± 0.013	0.327 ± 0.019	0.286 ± 0.013	0.281 ± 0.013
June	C-ESN	0.231 ± 0.003	0.257 ± 0.004	0.279 ± 0.004	0.257 ± 0.003	0.214 ± 0.003	0.248 ± 0.002
	L-ESN	0.237 ± 0.004	0.259 ± 0.004	0.295 ± 0.004	0.263 ± 0.005	0.221 ± 0.004	0.255 ± 0.002
	D-ESN	0.231 ± 0.003	0.254 ± 0.004	0.269 ± 0.005	0.260 ± 0.003	0.221 ± 0.004	0.247 ± 0.002
August	C-ESN	0.254 ± 0.004	0.234 ± 0.003	0.252 ± 0.003	0.259 ± 0.003	0.230 ± 0.004	0.246 ± 0.002
	L-ESN	0.263 ± 0.004	0.244 ± 0.003	0.258 ± 0.005	0.262 ± 0.005	0.231 ± 0.005	0.252 ± 0.003
	D-ESN	0.258 ± 0.003	0.235 ± 0.003	0.252 ± 0.004	0.265 ± 0.003	0.231 ± 0.005	0.248 ± 0.003
October	C-ESN	0.305 ± 0.002	0.230 ± 0.002	0.219 ± 0.002	0.254 ± 0.002	0.236 ± 0.002	0.249 ± 0.002
	L-ESN	0.320 ± 0.002	0.229 ± 0.001	0.221 ± 0.005	0.260 ± 0.002	0.244 ± 0.002	0.255 ± 0.002
	D-ESN	0.307 ± 0.002	0.229 ± 0.002	0.216 ± 0.003	0.262 ± 0.002	0.244 ± 0.002	0.254 ± 0.002
December	C-ESN	0.190 ± 0.003	0.177 ± 0.004	0.203 ± 0.003	0.223 ± 0.003	0.193 ± 0.003	0.197 ± 0.003
	L-ESN	0.182 ± 0.004	0.173 ± 0.004	0.219 ± 0.003	0.231 ± 0.004	0.192 ± 0.003	0.199 ± 0.003
	D-ESN	0.189 ± 0.002	0.178 ± 0.003	0.202 ± 0.003	0.222 ± 0.003	0.192 ± 0.003	0.197 ± 0.002

Chapter 5

Conclusive remarks and discussion

In this chapter, some conclusion will be drawn from the presented applications, discussing the reliability performance of the proposed methods.

First, several neural and fuzzy neural system approaches suitable for electric price prediction have been presented and analyzed. We have illustrated the validity of these approaches in terms of prediction performance and stability over different training/operative conditions. The accuracy of forecasting makes the proposed approach a very valuable path for optimizing energy sales by aiming at knowing beforehand what the optimal price would be for the selling day, many days in advance. All of the proposed models are suitable for this purpose, with the HONFIS one outperforming RBF and MoG. Future extensions of this research could consider an on-line prediction in which the parameters are re-learned at each new sample collected and available for training. The forecasting would thus be more strongly resilient to sudden changes of the time series. Also, incorporating heuristics on other statistical features of the time series, by using for instance ensemble techniques, could improve the prediction performances.

It is worth discussing the variation of accuracy related to the different time-horizon considered, especially for the energy prediction application case. A one-day training is based on a very small number of samples in the training set, which is interesting for the computational cost of the learning procedure but it is an issue when many model parameters must be estimated, which is the well-known curse of dimensionality for neural networks. In fact, the third-order polynomial adopted for HONFIS makes it the most complex model

among the ones considered for this prediction problem, and its performances improve with respect to the other models as much as the the number of samples increases in the training set, although a larger training set may not be the optimal choice.

Although of relative usefulness, the numerical results for weekly test sets are similar to the previous ones. In the case of seven-day training sets, all of the neural models suffer from the curse of dimensionality, as the information given by a training set of the same length of the test set is too small to ensure a robust estimation of the model parameters. In such cases, the LSE predictor yields better performances than the others models albeit too shallow in absolute terms. In the case of 30-day training sets, HONFIS is able to obtain the best results for weekly test sets, with a numerical score in terms of both NMSE and MARE that is stable enough with respect to shorter test sets of one day only.

The performance of the prediction is highly affected by the intrinsic seasonality of the time series considered. Anyway, HONFIS achieves the best results for almost all of the training sets with respect to the other proposed neural models, and all of them outperform both the LSE benchmark and the ARIMA approach, which is not a feasible solution in this case mainly for the intrinsic chaotic properties of the sequence. This reinforces the fact that the prediction of photovoltaic production is a promising field for the application of neural and fuzzy neural approaches along with the use of a suitable embedding procedure.

The results are very promising and suggest several opportunities for future work. We could make use of detrending techniques, such as mean-reverting approaches, in order to remove seasonal differences and spike outliers, as well as to improve the training accuracy. Additionally, it could be useful to test distributed learning approaches [147], by which the results could be improved sharing the data from different cabins of the same plant or from different nearby plants. Instead, regarding the ESN, we first tested it in a peculiar isolated grid case. The optimal management of a BESS, with prediction of PB production and load forecasting, was chosen as a problem, showing the versatility of the suited ESN method. In fact, the tests show that it is feasible and can be used effectively in the proposed context. Future works could consider different time horizons for the prediction and incorporating other information to strengthen the prediction, particularly the wind speed, in a learning scenario on distributed multiple sources of data [152, 153].

We then extended the ESN model to work with a novel, distributed approach for the prediction of time series in a network of multiple power plants. We relied on the ESN paradigm and direct application of ADMM and DAC protocols in order to accurately handle the communication and information sharing among the PV plants in the network. By testing it on real-world data, we demonstrated the efficiency of the described method, comparing it with respect to the centralized and local versions of the same algorithm. While having good performances on every analyzed time horizon, the distributed algorithm improves the performance for 1-day, 3-days and even for a 7-days long prediction. The proposed approach demonstrates to be a valuable tool for distribution companies that have interest to make dispatchable the intermittent energy produced by RESs, for traders that have interest to predict energy productions on the next 3-7 days in order to submit effective bids in the day-head energy market, and for O&M companies interested to properly scheduling the programmed maintenance operations.

Part III

Other Contributions

Chapter 6

Validated Distributed Ensemble Clustering

6.1 Introduction

In addition to the supervised problems discussed so ar, another interesting field of application of distributed computing paradigms is given by unsupervised learning problems, clustering in particular.

With the advent of big data, cloud storage, social networks and so on, it is necessary to deal with a huge amount of information and data that cannot be stored, as in the past, in a central database or computing node. Thus, new techniques are mandatory for modeling and managing distributed data sources. In this chapter, an unsupervised learning algorithm is presented that is suitable for clustering data in a distributed environment.

As explained in [154], there are some aspects that need a careful attention when dealing with big data; the algorithms must be able to handle different kind of data (numerical, categorical or hierarchical), working at a reasonable speed even when the volume of the data increases. In astronomy, for instance, several telescopes spread all over the world are able to collect up to 1 GB of data per hour. This amount of data may be difficult to transmit to a central node for analysis and processing of the whole dataset [155]. Technologies such as Sensor Networks, Cloud Storage and Social Networks reshape the ICT world with big data [156, 157], while pervasive computing and the Internet of Things make more convenient to process data in a decentralized manner when data are collected across the network [158, 159].

For these reasons, latest researches are moving towards distributed tech-

niques. The information is exchanged only thorough the neighbors, each node having a visibility on a partial (local) dataset only, thus avoiding a central node that detains all data as depicted in Fig. 6.1. There is a network of agents that are linked together and that acquire data autonomously. We suppose that the connectivity is known *a priori* and fixed. For the sake of simplicity, we will assume that the network is fully connected (every node can be reached from another node), and undirected (the adjacency matrix is symmetric). This framework solves the problems of security, processing, transferability and privacy. Additionally, the computational cost at each node is generally less than the cost of communication between agents, without considering that sometimes the transmission from a node to the central one is almost impossible, making the centralized solution expensive and/or infeasible.

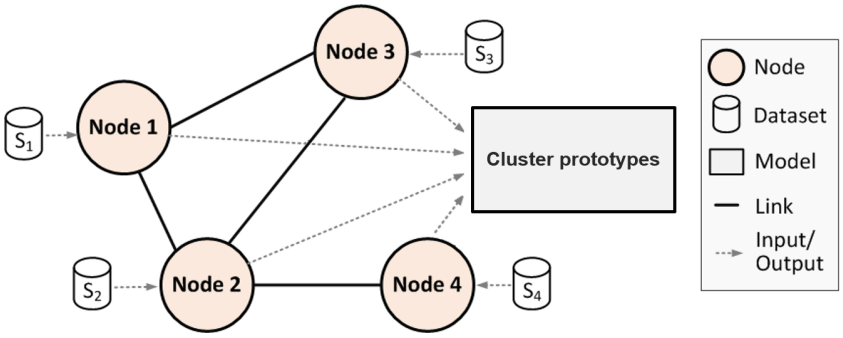


Figure 6.1: A distributed scenario for clustering.

Several authors have treated the unsupervised distributed learning problem. In order to reduce the computational cost while preserving the serialization of the algorithm, in [160–162] several methods based on Gaussian mixture components are presented. In these methods, the authors assume that each node retrieves data from an environment that can be described by a probability density function as a mixture of elementary conditions. The mixture parameters are commonly estimated in an iterative way through the Expectation-Maximization (EM) algorithm, performing the E-step locally at each node and reaching the consensus in the M-step by local exchange of information among neighbors. Although the algorithms are very scalable and robust, they need bridge sensors to reach consensus and furthermore the number of Gaussian components must be known beforehand.

In [163, 164] the idea is to split the original problem into different subproblems that are solved using the K-Means algorithm at the singles nodes.

Then, local results are processed together in a central node to find a common structure through the merging of the different solutions. To reduce the amount of data to be exchanged in the communication process, in [165] a Principal Component Analysis (PCA) control is applied to extract relevant features before applying the K-Means algorithm. A similar approach is the one presented in [166, 167] where a master-slave architecture is proposed and a central node manages the local results to find the optimal partition. An technique that can be efficiently implemented in a purely distributed fashion is based on the well-known Fuzzy C-Means (FCM) algorithm.

With particular focus on the image segmentation, in [168] an approach suitable for large size image processing is presented. This algorithm divides the computation among the processors minimizing the need of accessing the secondary storage. In [169, 170] a FCM algorithm is applied in a context where sites receive data from multiple sources. Consensus criteria and a collaborative approach are used to find out an agreement among the nodes. Finally, particular attention should be devoted to the recent Ensemble Clustering (EC) techniques [171] that are characterized by two main phases: (i) generation of a number of local clustering solutions; (ii) determination of global consensus solution by merging the local ones. When the dataset is very large, EC techniques represent a proper approach to obtain a common structure of partitions through a process of collaboration and communication among the agents ([172, 173]). In [174] the authors proposed an EC approach through decentralized observations. It is an on-line algorithm that is basically different from the approach proposed in this chapter. Also, it relies on exchanging patterns among agents and it is not oriented to the privacy-preserving of data.

Given the unsupervised nature of the problem, it is very difficult to determine the best solution among a set generated ones, which are obtained by using the chosen clustering procedure. Moreover, the problem is much harder in a distributed scenario, where voting procedures and consensus strategies are mandatory to find out the result that best fits the actual structure of local data with respect to a pre-defined metric. Leveraging consensus across multiple clustering results provides more accurate and stable solutions when compared to traditional (centralized) clustering techniques. Therefore, the problem to be faced is twofold: to deploy a centralized intelligence, for using consensus or other similar cooperative techniques, and to solve a cluster validity problem, in order to estimate the right number of clusters.

The main scope of this contribution is to discuss and propose a new solution to such problems, when clustering is applied to multiple sources of data within the distributed environment described so far. In the framework of the EC approach presented in [175, 176], we will introduce the Validated Distributed Ensemble Clustering (V-DEC) algorithm. The proposed approach is based on two main concepts: in order to reduce the amount of exchanged information, only the prototypes of local clusters are communicated among neighbors; also, clustering validity indexes are adopted to avoid conflicts among nodes and to converge to a coherent data partition at the end of the communication process.

The rest of the chapter is organized as follows. We introduce in Section 6.2 the details of the proposed V-DEC algorithm and in Section 6.3 the use of cluster validity indexes to overcome some clustering problems in a distributed learning environment. The experimental results are illustrated in Section 6.4, based on numerical simulations applied to well-known datasets and clustering benchmarks, while in Section 6.4.1 we draw our conclusions with some final remarks.

6.2 The Proposed Clustering Algorithm

The proposed solution is illustrated in the following, where a toy problem consisting of four Gaussian sources in a 2-D data space is used to better explain each step of the algorithm.

6.2.1 Initial clustering

In the first iteration of the algorithm, each node has a partial vision of the entire dataset. Autonomously, the agents try to find out the local partition of their own data applying a clustering algorithm. In particular, the different local results can be obtained by employing different algorithms such as K-means, fuzzy C-means, spectral clustering, Expectation-Maximization (EM), etc., by varying their parameters using different metrics or dissimilarity measures, number of clusters, initial random centers, and so on.

One of the novelties of the V-DEC approach is the capability to work with different datasets without the requirement that each node starts with the same local dataset to reach a final consensus. For instance, in a sensor network that is measuring the level of air pollution in a physical environment

Algorithm 2 V-DEC Algorithm

Initial clustering

Let $R = \{R^{ik}\}_{1 \leq i \leq m}$ *as the intial set of clustering.*

Collaboration phase

Conflict detection

$k = conflicts(R)\ with K_k^{i,j} = 1 - S(C_i^k, CC(C_i^k, R_j))$

where $S(C_i^k, C_j^l) = max\{S(C_i^k, C_j^m), \forall m \subseteq [1, n_j]$

Conflict Resolution

$K_k^{i,j} = argmax_{K_l^{r,s}}\ Cl(K_l^{r,s})$ and $CC(C_i^k, R_j) = C_j^l$

The local resolution of conflicts is obtained by:

if $k <$ *thereshold then:*

$R^{'} = R^{'} \setminus k \cup merge(k, R^j)$

else

$R^{'} = R^{'}\{C_k^j \cup split(C_k^j, |k|\ )$

if $\{$*Validity index*$(R^{'}) >$ *Validity index*$(R)\}$ *then:*

$R = R^{'}$ *end*

end

Additional refinements

Consensus computation.

using sensors spread all over the land, it is not realistic to assume that each node starts with the same concentration of polluting agents.

We will consider in the following the K-means algorithm as the reference clustering procedure, assuming to initialize every agent with the same number of clusters (so as to prevent an undetermined number of clusters in every local model) but with different centroids. The initial clustering applied to the chosen toy problem is shown in Fig. 6.2. In this case, each agent has access to a different dataset and, based on a different initialization of centroids, different results are obtained at each node (N1 to N4) with several errors: some clusters are split and others are grouped as a unique one.

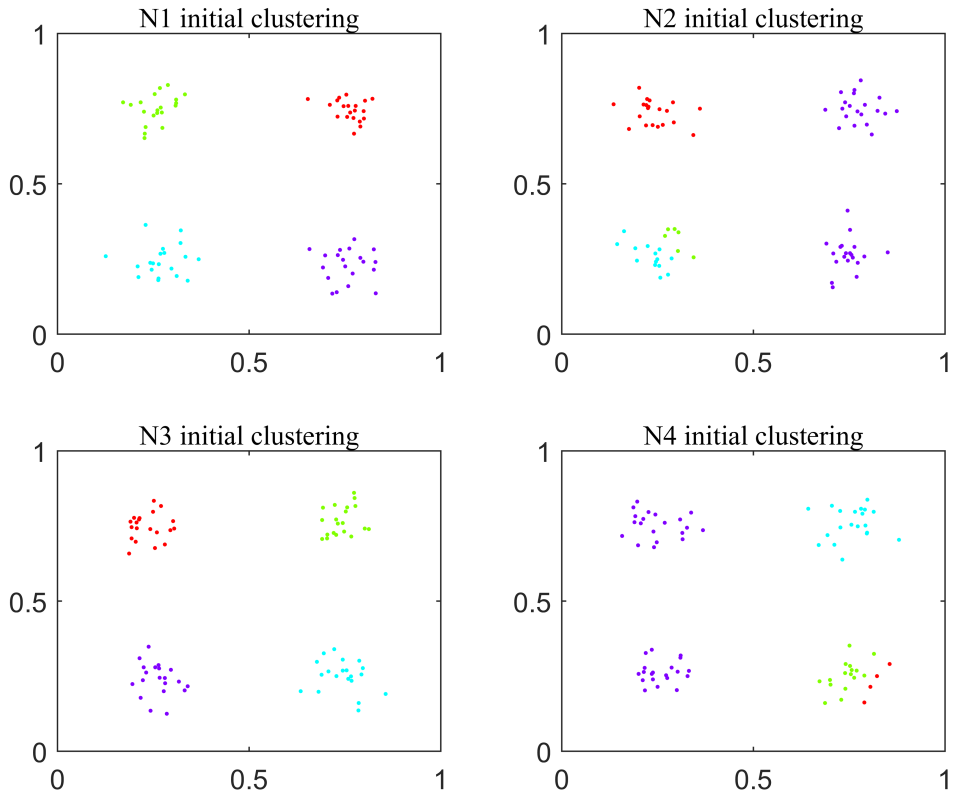


Figure 6.2: Initial clustering on a 2-D toy problem at four nodes: each color represents a different cluster a pattern is assigned.

6.2.2 Collaboration phase

The collaboration phase starts with the conflict detection achieved thanks to the local exchange of the centroid information or, equivalently, if another algorithm is used in place of K-means, with the exchange of the suited local representative . Let $\sigma(C_k, C_h)$ be the similarity measure between two clusters C_k and C_h, respectively; such a measure will be defined and discussed successively, but it is always assumed to be normalized in the range from 0 to 1.

The similarity measure σ between two clusters of two different agents is defined as the percentage of elements of a cluster that are closest to the cluster in the other agent, similar to a KNN measure. To compute it for a cluster in a node with a cluster of another node, the agents need to previously share all the centroids information. Once the centroids are shared, the agent can compute the distance of every single element in the cluster with respect to all of the centroids of another node. Firsly, we define the distance between a single point of a cluster and the centroid of another cluster in another node.

Given the m-th cluster in the node i, C_m^i, with L data points p_l^m, the distance between a data point and the n-th cluster in node j, C_n^j , whose centroid is c_n^j, is:

$$d_{l,n} = \mathbf{d}(p_l^m, c_n^j), \tag{6.1}$$

where $\mathbf{d}$ is the euclidean distance. For each data point p_l^m in the first node i, we compute the minimum distance respect to every cluster C_n^j in the other node j:

$$\hat{k} = \underset{n=1,\dots,N}{\arg\min} \{d_n\}, \tag{6.2}$$

where N is the number of clusters in node j. Then we define K_n as the number of elements in C_m^i for which $\hat{k} = n$; here we are simply counting how many elements in the cluster m of the first node i have found to have the minimum distance with the centroid of cluster n of the second node j. Given that, the distance between C_m^i and C_n^j can be defined as the percentage of such elements respect to all of the elements in the cluster:

$$\sigma(C_m^i, C_n^j) = \frac{K_n}{L}. \tag{6.3}$$

When clustering is performed independently at each node, referring to different instances of a same data source, there might be used a different label or a different index order to represent clusters that correspond to a same local distribution. Consequently, it is important to set a function φ able to determine which cluster in a node corresponds to a same cluster in another node. Let C_k^i be the k-th cluster found at node i, then $C_h^j = \varphi(C_k^i, j)$ will find the cluster C_h^j at node j as the one corresponding to the same local distribution of data points as cluster C_k^i at node i. Such a correspondence is based on the similarity measure between clusters and hence, referring to the previous example where index h is the essential outcome of the function φ:

$$h = \underset{m}{\arg\max} \left\{\sigma(C_k^i, C_m^j)\right\}. \tag{6.4}$$

While finding a correspondence, some conflicts may occur. We can define two type of conflicts, namely *type-I*, defined between a cluster in a node and its corresponding cluster in another node, and *type-II*, defined by two clusters in the same node. Usually, each cluster is found to be corresponding to

a cluster in another node with similarity equal or nearly equal to 1. If so, we can be confidently sure that the cluster is in fact the same cluster (from the same local distribution) in the other node. There may be the occurrence in which a cluster is found to be corresponding to a cluster with an unsatisfactory similarity. This means that, the elements of the cluster are labeled as pertaining to two or more different clusters of another node, removing the possibility of identifying a unique corresponding cluster with sufficient confidence. If we define a similarity threshold σ_t, we can formalize the definition of the *type-I* conflict:
A type-I conflict between the k-th cluster at node i and its corresponding cluster at node j occurs when the similarity between them is below the satisfactory threshold:

$$\sigma(C_k^i, \varphi(C_k^i, j)) < \sigma_t. \tag{6.5}$$

(Another possible way of defining the conflict would be to evaluate the difference in similarity between the one computed with the corresponding cluster and the one computed with the other cluster for which the similarity is not 0). This means that when *type-I* a conflict occurs, there are more than one cluster at node j that could be considered as the possible corresponding cluster; in other words, the elements of the cluster at node i are found to be spread in two or more clusters at node j, without one of them having a sufficient portion of elements of the cluster at node i uniquely labeled (uniquely) as its. Instead, the *type-II* conflict arises when more than one cluster in a node finds the same corresponding cluster in another node:
A type-II conflict between two clusters at the same node j occurs when they have a similarity equal to 1 with respect to a same cluster in a node i:

$$\varphi(C_m^j) = \varphi(C_n^j) = C_k^i \tag{6.6}$$

with

$$\sigma(C_k^i, C_m^j) = \sigma(C_k^i, C_n^j) = 1 \tag{6.7}$$

The mismatch degree of a correspondence can be measured still relying on the similarity between clusters. Namely, we can define a bounded function μ in the range 0 to 1, which measures the complement to 1 of the similarity

between the original cluster and the one chosen as corresponding in the other node:

$$\mu(C_k^i, j) = 1 - \sigma(C_k^i, \varphi(C_k^i, j)) \,, \tag{6.8}$$

where μ is 0 in case of perfect correspondence (perfect similarity). Evidently, function μ can be used also to estimate the relevance of a possible *type-II* conflict as, in the case of a cluster at node j corresponding to more than one cluster at node i, the related similarities will be different (in general) and hence, the two correspondences will have a degree of mismatch greater than 0.

Once conflicts are determined, we execute two operations to improve the cluster similarities: 'merging' and 'splitting'. In the cluster ensemble approaches previously cited, the operations involved in the resolution of the conflicts are three: merge, split and recluster. In particular the authors apply there merging and splitting together (when a cluster is merged the other involved in the conflict is split) while the recluster is used if the initial clustering result is under a certain goodness threshold. We revised this version to make it more suited for purely distributed environment, changing the operations involved. We start by solving the *type-I* conflicts by applying the merge operator to the clusters involved. The strategy is to solve the *type-I* by merging the clusters from which the uncertainty arises (all the clusters of the same node whose correspondence is in conflict), so to have only corresponding clusters with $\sigma = 1$.

In our algorithm the merging operation is simply done by reassigning the labels coherently after selecting and storing the elements which generate conflicts. We can see in Fig. 6.3 when two clusters are merged together (in the toy problem example).
There are two possible outcomes for every *type-I* conflict after merging:

- the conflict is solved: the merged clusters come indeed from the same local distribution and are now labeled as such.
- the conflict is transformed in a *type-II* conflict: two clusters of another node find the same corresponding cluster in the new, merged one that comes from two different local distributions.

Consequently, after the merging step, we will have only *type-II* conflicts. We will address these ones by using a 'splitting' step. The K-Means algorithm

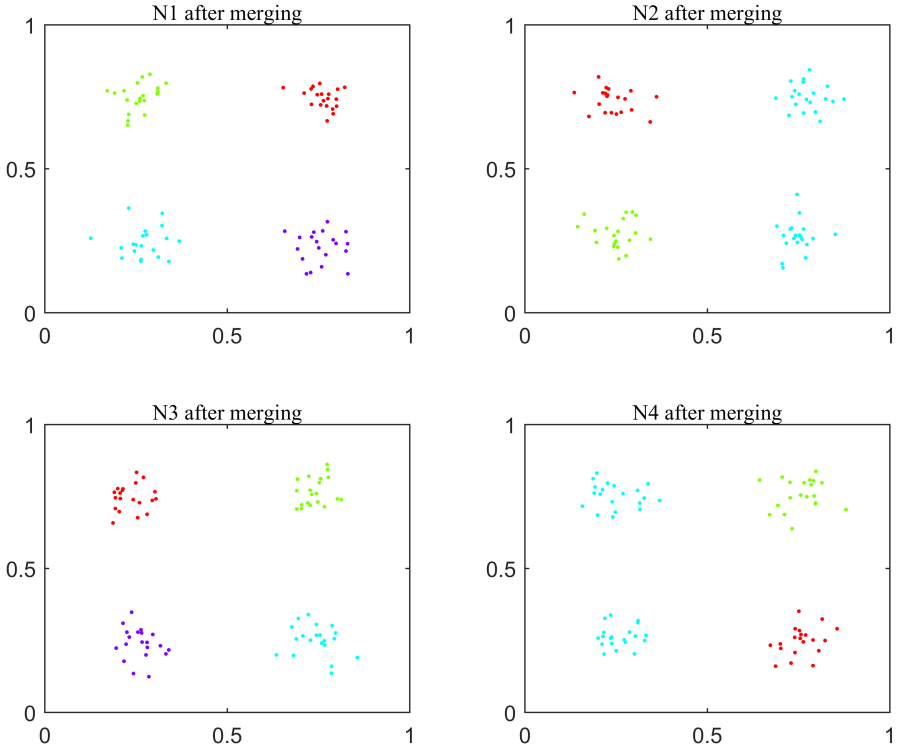


Figure 6.3: Toy problem after merging

(or any other appropriate clustering algorithm) is applied to the conflict's data points with a suited cluster number initialization. This way each cluster involved in the conflict is split in different clusters coherently. As we can see from Fig. 6.4 the split operators is able to divide clusters erroneously classified by the initial clustering.

To ensure the accuracy of the merging and splitting steps, we have inserted validation indexes to check if the new clustering result is effectively better than the previous. These indexes are detailed in sec.6.3. At the start of the collaboration phase, these indexes are computed for every agent. At each new result of the conflict resolution step, these indexes are recomputed and compared with the old ones. The modification of the result is then carried on only if the validity indexes of the new cluster result are better that the old ones, otherwise the changes on the labels are discarded.
After the conflict resolution, detection of the conflict must be reiterated and cluster labels must be normalized to ensure that each result has a complete set of labels (if there are N clusters in a results they should be labeled continuously from 1 to N, without gaps to avoid computational problems). The centroids are also locally recomputed after each modification of the results.

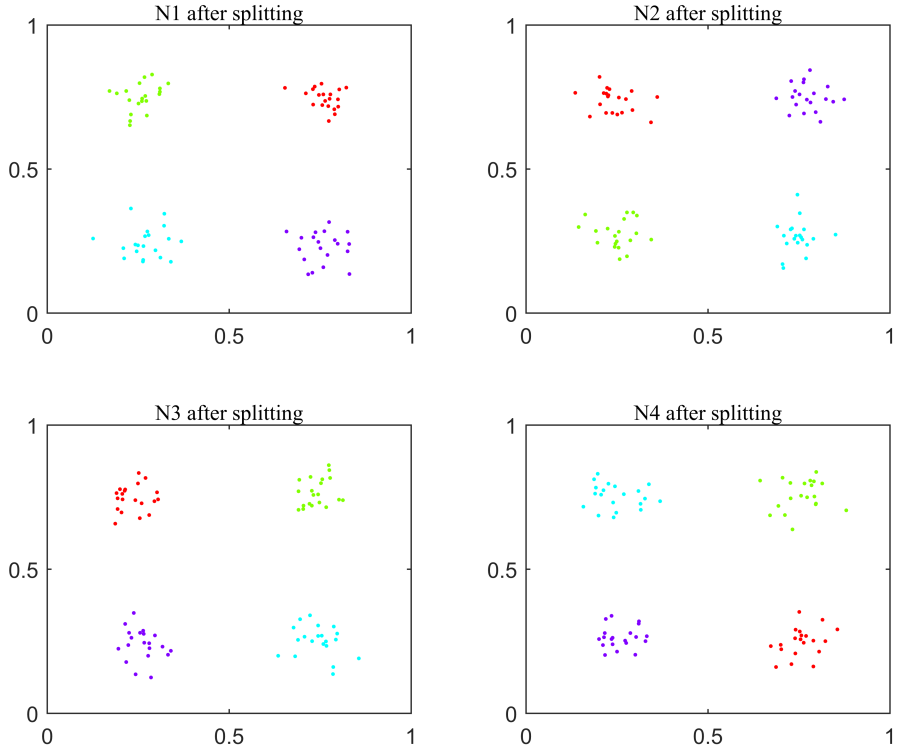


Figure 6.4: Toy problem after splitting

6.2.3 Consensus computation

In the last step of the V-DEC algorithm, it produces different partitions that represent only permutation of the original ones. In these results all of the conflicts are solved. In the state of the art clustering ensemble techniques a consensus operation is in place and works exchanging all the datapoints between every learner.

Considering that all the partitions that differ only in cluster labeling can be considered identical in terms of clustering accuracy, a relabeling operation is necessary to find a common agreement. Since every agent has already evaluated all the centroids there is no need to exchange other information, because the centroids have been exchanged in the collaboration phase. Thus, each agent reaches the consensus agreement by applying the initial clustering algorithm (e.g. K-means) to the complete set of centroids. In this way, each agent is able to relabel its own clusters to reach a global agreement. After all, given the shared centroids, this consensus resembles a global clustering for label reassignment.

To better converge to an unified result, for result evaluation purpose only, the relabeling is done incorporating the real labels of the clusters inside the anal-

ysis. This is further explained in the results section. A visual representation should be seen in Fig. 6.5 where all the clusters are labeled correctly, making this procedure suitable for purely distributed environment where each agent has to compute the labels independently.

Every node has all the centroids, so it can recluster them and assign its own node to the right cluster of nodes (label)

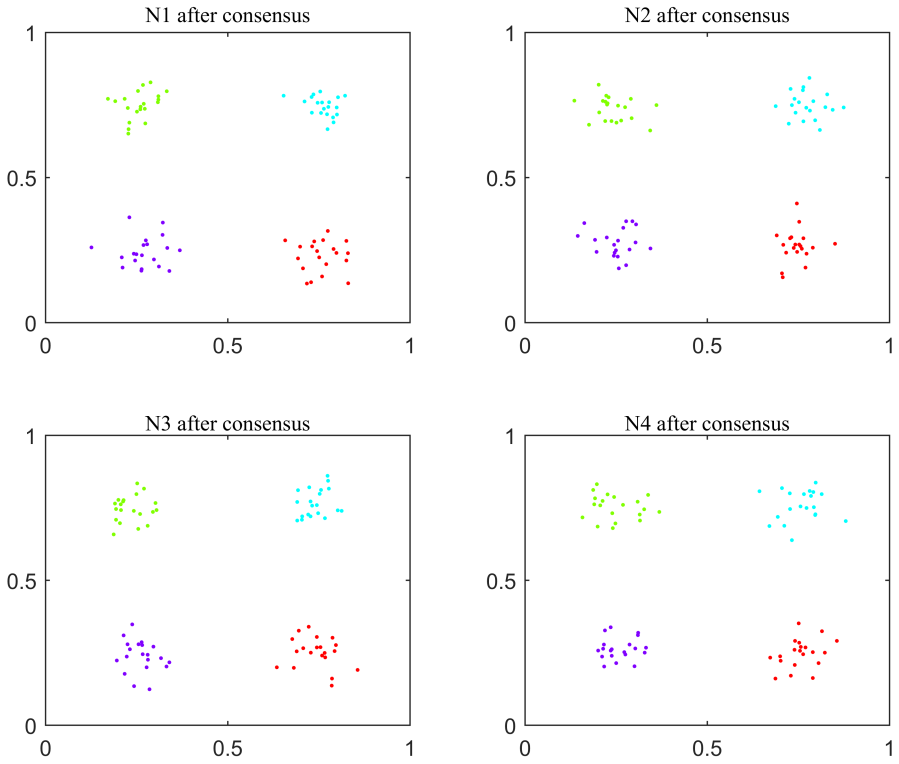


Figure 6.5: Toy problem after consensus

6.3 Cluster Validity in a Distributed Scenario

In this paper, we apply an approach based on an ensemble clustering technique, where the collaboration process in a network as in Fig. 6.1 is carried out by the exchange of information between agents and by a consensus voting algorithm. The novelty is the capability to work in a purely distributed environment where each node has a different vision of the global data (i.e., it has an own local dataset).

By exploiting the local datasets and the communication among neighbors, the agents must be able to reach an agreement on the global results. Namely, the obtained results should approximate sufficiently well the situation where

a same clustering strategy were executed on a centralized node that collects all the local datasets. In order to achieve a data structure that is common and globally coherent with all data, as well as to detect a possible disagreement among local partitions, we propose the use of cluster validation indexes.

In fact, as it is difficult to evaluate the correctness of a partition given the unsupervised nature of the clustering problem, several indexes have been proposed in the literature that can be used to assess the quality of a clustering result. As described in the following, in the proposed V-DEC algorithm the agents communicate with each other to detect main conflicts and to solve them by merging or splitting clusters. Therefore, a suitable cluster validation index could be adopted to verify if such operations yield either an improvement or a worsening of global clustering results. Such a management of local changes of a clustering result is one of the novelties introduced by the V-DEC algorithm.

Some clustering ensemble techniques [176] are based on a combination of the intercluster similarity, some cluster quality criterion and user-defined parameters. One of the main issues in these kind of problem is to end up with trivial solutions (only one clusters with all the elements or many clusters with only one element). To avoid this, δ^i and δ^j are introduced in [176]. These are quality criteria that incorporate the external knowledge in the quality measure (e.i. estimation of number of clusters, a-priori-labeled samples, constraints). These criteria are useful if there is in fact some a priori knowledge to incorporate in the analysis. We decided to not use them because we want to be able to solve the clustering problem without external knowledge.

Once the local similarity criterion γ is computed, a global agreement coefficient Γ is evaluated for the management of the global results.

$$\Gamma = \frac{1}{m} \sum_{i=1}^{m} \Gamma^i \tag{6.9}$$

where

$$\Gamma^i = \frac{1}{m-1} \sum_{j=1, j \neq i}^{m} \gamma^{i,j} \tag{6.10}$$

These two steps are complex and onerous in terms of time and resources (and do not guarantee privacy). Instead, in V-DEC we introduce validity indexes to evaluate the local changes based on the global results for better performance.

We use three different indexes that are tested with several datasets:

- *Davies-Bouldin Index* ([177]): this is an internal evaluation index, where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset.
- *Dunn Index* ([178]): it is of the same group as the Davies-Bouldin index, in that it is an internal evaluation scheme, where the result is based on the clustered data itself. The aim is to identify sets of clusters that are compact, with a small variance between members of the cluster, and well separated, where the means of different clusters are sufficiently far apart, as compared to the within cluster variance.
- *DW-DB Index* ([179]): The double weighted Davies-Bouldin index is a modified version of the DB Index. It is a double weighted index that avoids to fall into some local minimums that affect the standard Davies-Bouldin Index.

Whenever a modification takes place in a local result, the agent computes the index taking into account the other results. If the modified result obtains better indexes the changes are accepted and the centroids recomputed, otherwise the conflict is eliminated and the algorithm iterates. This check is necessary to avoid to fall into a local minimum or maximum, in fact the multiple iterations of split and merge without any check could produce the situation in which there is one cluster for all the objects or one cluster for each object. There are differences in performance depending on which index is used. The DB index performs well in highly differentiated datasets but is too inaccurate when the clusters are not well separated. The Dunn index is useful when there are a lot of attributes but it is harder to compute and tends to fall in a situation where only one global cluster is found. The DW-DB index performs better than the others in both separable or non-separable classes and different datasets.

6.4 Experimental Results

We tested the V-DEC algorithm on four different available datasets of the UCI repository and one toy problem built ad hoc to test it in different distributed scenario. A schematic description is given in Tab.6.1 where we present some additional information on them.

Table 6.1: Description of the datasets

Dataset	Features	Instances	Classes	Task
Iris Data Set	4	150	3	Classification
Wine Data Set	13	178	3	Classification
Ionosphere Data Set	34	351	2	Classification

- *Iris Data Set* is a classification dataset, composed by 150 instances for each of the three classes. The input is given by 4 features.
- *Wine Data Set* where the task is to identify the correct type of wine among 3 classes thanks to 178 instances composed by 13 features.
- *Ionosphere Data Set* The Ionosphere dataset consists of 16 high-frequency antennas and 17 pulse numbers for the Goose Bay system. Instances are described by 2 attributed per pulse number. It is a binary classification dataset, where each learner get 87 datapoints with 10 attributes each.
- *Toy Problem* In addition to the mentioned UCI datasets we used also a toy problem dataset composed by four classes that could be arranged and distributed very easily to preliminary test the V-DEC algorithm.

In all cases, input variables are normalized between 0 and 1, and missing values are replaced with the average computed over the rest of the dataset. We compare the following techniques, with the same set of runs and parameters:

- **Centralized Algorithm**
- **Samarah Algorithm** that is part of the Cluster Ensemble techniques presented in [176]
- **V-DEC**: Validated Distributed Ensemble Clustering.

The initial clustering algorithm used is the K-Means with a random initialization. We also tested the E-M algorithm to compare the results and to see how much the final result is affected by using different local clustering algorithms. After a grid search procedure, we fixed the threshold for the merge criterion (σ_t) by setting it to 0.4. This is found to be the safest value because it avoids unnecessary merging of clusters that become hard to split in sequent phase without skipping the merge when it is needed.

For each test we performed 100 runs and taking the average value. All experiments are carried on using MATLAB R2013b on a on a machine with Intel Core i5 processor with a CPU @ 3.00 GHz with 16 GB of RAM.

As stated in the Consensus section, to be able to evaluate the quality of the final clustering results it is necessary that the labels of each learner are all coherent. To achieve that, a data-driven clustering is applied to all the centroids. The results' centroids are gathered in a matrix, clustered via K-Means and, taking in account the real clusters' centroids, relabeled. This way it is possible to compute quality indexes to evaluate the performance.

As explained before we partition all the datasets to test the algorithm in a purely distributed scenario, differentiating from [176] where all the dataset is given to each node. The number of attributes is different from dataset to dataset, but is always greater than three.

In order to evaluate the performances we use four quality indexes to compare the results with those obtained in [176]:

- Rand Index ([180])
- Falks-Mallows ([181])
- F-measure ([181])
- K-Index ([182])

In Tab.6.2 are reported the results for all of the similarity indexes presented below, in order to compare how the results change when we increase the number of agents in the network, as well as change the initial classification algorithm. All the indexes are normalized between 0 and 1 (0 being 'worst result' and 1 being 'best result'), for the sake interpretability Best results for each initial configuration are highlighted in bold. As we can see results are quite insensitive to the increasing of the network, since the initial algorithm should has more impact in some dataset.

In Tab.6.3 the V-DEC approach is compared with the cluster ensemble techniques and the centralized approach in terms of the quality indexes. Best results for each dataset are reported in bold. As we can see the performance are slightly less good when we compared with the ensemble clustering technique. This is coherent with our analysis, because they use the entire dataset at each node, while we are tested our algorithm in a purely distributed context, forcing each agent to have partial and different vision for each dataset.

Dataset	Algorithm	N learners	FM	RI	KI
Iris	K-Means	1	0.786 ± 0.065	0.845 ± 0.066	**0.737 ± 0.196**
	EM	1	**0.827 ± 0.110**	**0.859 ± 0.101**	0.711 ± 0.260
	K-Means	4	**0.761 ± 0.055**	**0.828 ± 0.062**	**0.661 ± 0.020**
	EM	4	0.676 ± 0.069	0.758 ± 0.050	0.585 ± 0.121
Wine	K-Means	1	0.429 ± 0.007	0.594 ± 0.001	0.226 ± 0.003
	EM	1	**0.534 ± 0.062**	**0.629 ± 0.073**	**0.352 ± 0.119**
	K-Means	4	**0.453 ± 0.037**	**0.570 ± 0.050**	0.231 ± 0.062
	EM	4	0.441 ± 0.036	0.570 ± 0.033	**0.236 ± 0.063**
Vehicle	K-Means	1	0.367 ± 0.018	0.637 ± 0.021	**0.262 ± 0.012**
	EM	1	**0.377 ± 0.025**	**0.660 ± 0.020**	0.245 ± 0.040
	K-Means	4	**0.369 ± 0.025**	**0.616 ± 0.039**	**0.234 ± 0.033**
	EM	4	0.356 ± 0.027	0.615 ± 0.0302	0.199 ± 0.032
	K-Means	10	**0.366 ± 0.025**	0.617 ± 0.031	**0.226 ± 0.027**
	EM	10	0.342 ± 0.024	**0.624 ± 0.022**	0.193 ± 0.028
Ionosphere	EM	1	**0.685 ± 0.059**	**0.613 ± 0.090**	0.319 ± 0.305
	K-Means	1	0.605 ± 0.000	0.589 ± 0.000	**0.411 ± 0.000**
	EM	4	**0.620 ± 0.067**	0.533 ± 0.021	0.089 ± 0.114
	K-Means	4	0.599 ± 0.211	**0.581 ± 0.020**	**0.389 ± 0.470**
	EM	10	**0.612 ± 0.060**	0.532 ± 0.018	0.107 ± 0.093
	K-Means	10	0.596 ± 0.296	**0.573 ± 0.024**	**0.342 ± 0.091**

Table 6.2: Cluster quality indexes for K-Means and EM initialization over different initial network configurations.

It is important to underline, that despite of this constraint we are also able to obtain comparable performance with the centralized approach.

Dataset	Algorithm	Rand Index	F-Measure	F-M Index
Iris	C	0.73 (± 0.03)	0.64 (± 0.02)	0.60 (± 0.01)
	S	**0.85 (± 0.00)**	**0.78 (± 0.00)**	**0.78 (± 0.01)**
	V-DEC	0.76 (± 0.05)	0.65 (± 0.07)	0.59 (± 0.02)
Wine	C	0.68 (± 0.09)	0.66 (± 0.03)	0.62 (± 0.04)
	S	**0.88 (± 0.04)**	**0.83 (± 0.06)**	**0.83 (± 0.06)**
	V-DEC	0.75 (± 0.06)	0.71 (± 0.08)	0.70 (± 0.05)
Ionosphere	C	0.56 (± 0.01)	0.53 (± 0.03)	0.50 (± 0.04)
	S	0.59 (± 0.03)	0.53 (±0.07)	**0.50 (± 0.09)**
	V-DEC	**0.71 (± 0.02)**	**0.58 (± 0.04)**	0.49 (± 0.05)

Table 6.3: Cluster quality Indexes for the Centralized, the ensemble clustering approach and the V-DEC algorithm.

A visual representation of the results is given in Fig. 6.6, 6.7, 6.8:

6.4.1 Conclusion

In this chapter, we have introduced a distributed algorithm able to work in an environment where data are collected among a network of agents. In par-

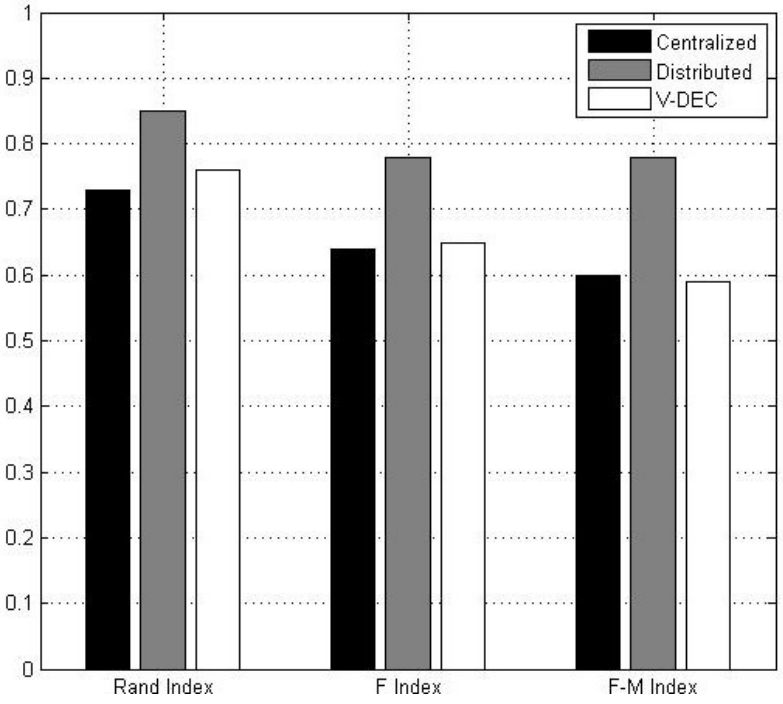


Figure 6.6: Quality Indexes comparison for Iris Dataset

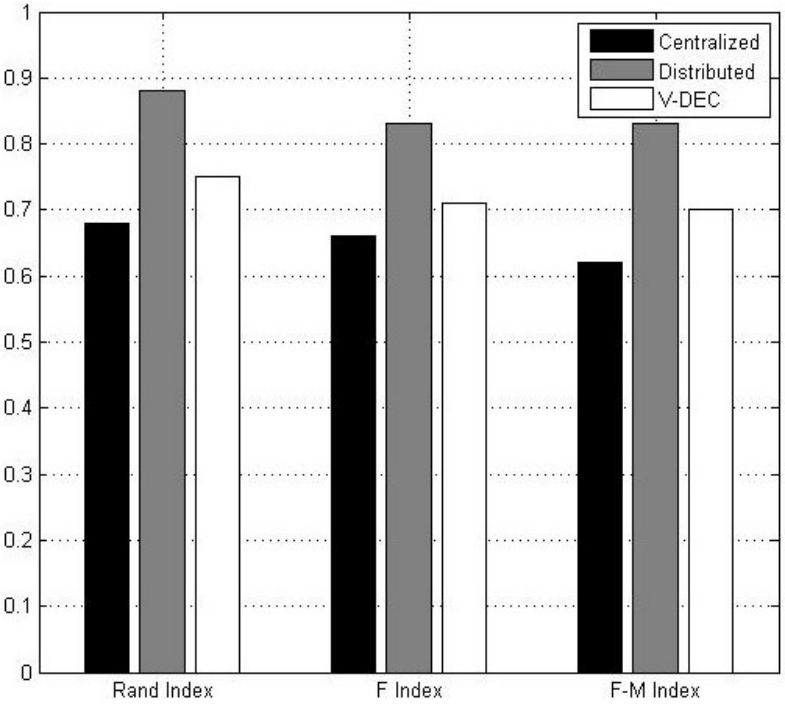


Figure 6.7: Quality Indexes comparison for Wine Dataset

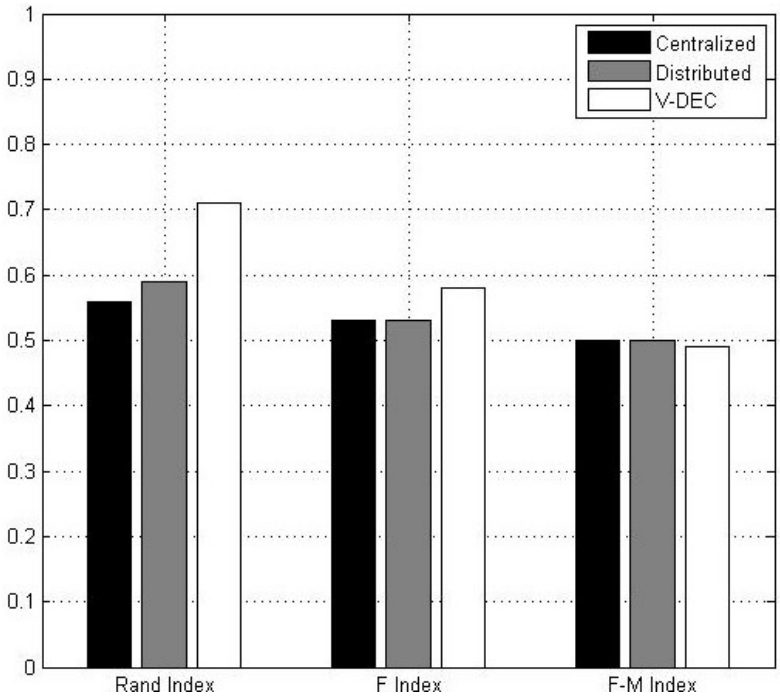


Figure 6.8: Quality Indexes comparison for Ionosphere Dataset

ticular, we consider the traditional cluster ensemble techniques and extend them. Since the traditional approaches have the requirement that each node must have the same (complete) dataset, we allows each agent to work with a different and partial dataset. For this reason the algorithm is slightly less good of the traditional cluster ensemble techniques. However, experimental results suggest that the procedure is able to efficiently match a fully centralized implementation, using only in-network implementation. Additionally our approach require the exchange of only the representatives, making it very efficient in computation.

Chapter 7

Finite precision Random Vector Functional Link Network

7.1 Introduction

With the deep penetration of sensor network architectures in several application contexts, many issues are raised regarding the implementation and development of machine learning algorithms for low computational power devices in distributed systems.

Internet of Things (IoT), cloud computing, pervasive computing and so on, have revolutionized the way signals are processed and information is managed. To infer knowledge from big data partitioned over geographically distinct locations is considered a fundamental problem now in many scientific fields [183, 184], including sensor networks [185], smart grids [186], medical applications [187] and many others. In this kind of scenario, it is often forbidden to transmit all data to a centralized authority, for reasons of security, privacy, computational efficiency and economical costs.

Neural networks are usually adopted to solve most of the above mentioned machine learning problems but, unfortunately, these approaches are often computationally intensive and memory demanding. This makes difficult to deploy them on distributed systems where additional constraints might be imposed, such as the low computational power of a simple and cheap hardware. Thus, more efficient approaches become necessary.

Usually, neural network parameters are estimated via learning algorithms running on standard computers with double precision floating point arithmetic due to the associated high computational demand. However, to up-

loading the network model on a digital, possibly distributed, architecture with finite precision arithmetic, after a direct quantization of coefficients, leads to unsatisfactory results due to the non linear nature of the network [188, 189]. Nonetheless, many real-time applications need an adaptive learning, as in the case of the well-known consensus strategy, where the model must be dynamically adapted to new observations even after the hardware implementation.

There have been various proposals to make inference in contexts with limited hardware resources. A learning procedure for generating multilayer integer-weight neural networks has been presented in [190]. Differential evolution strategies have been also applied to train neural networks with small integer constraints [191, 192]. In [193], a training mechanism with quantized weights that reduces the cost of hardware implementation has been presented and in [194] both approximation and quantization techniques are used for network pruning. However, all of such models are not suited to deal with distributed learning and data processing.

Random Vector Functional-Link (RVFL) can be viewed as a feed-forward neural network with a single hidden layer, resulting in a linear combination of a fixed number of non-linear expansions of the original input [195]. By solving the general problem of data regression, RVFLs can be applied to signal processing applications where function approximation, classification or time series forecasting is required. In particular, this technique has been used to implement distributed learning systems where training data is diffused under a decentralized information structure [196, 197], or to solve specific classification problem [198].

So far, only preliminary studies have been proposed for RVFL networks with limited hardware resources [199]; for instance, in [200] the algebraic properties of the mathematical model are investigated, considering the implementation on FPGA architectures with a relatively high number of bits (i.e., more than 16). In this work, we introduce a methodology to address the challenging problem of hardware implementation with finite precision arithmetic. Also, a novel optimization strategy based on a Genetic Algorithm (GA) is introduced, in order to estimate the inner parameters of the network under the constraint of finite precision arithmetic. The numerical simulations reported herein prove that the overall approach is able to simultaneously speed up the computation in the testing (operational) phase and to reduce the memory overhead on digital architectures based, for example, on a simple microcontroller using a very low number of bits.

7.2 RVFL Architecture

The peculiarity of a RVFL feed-forward neural network is the inner layer fixed a priori with a predefined set of nodes. Given a d-dimensional input $\boldsymbol{x} = [x_1 \ \dots \ x_d]^T$, the RVFL aims at estimating a scalar output $y \in \mathbb{R}$. The traditional formulation uses a weighted sum of C non-linear transformations of the input:

$$f(\boldsymbol{x}) = \sum_{m=1}^{C} \beta_m h_m(\boldsymbol{x}; \boldsymbol{w}_m) \,. \tag{7.1}$$

Each $h : \mathcal{X} \to \mathbb{R}$ is a *base, hidden* function, or *functional link*. In the following, a sigmoid basis functions is adopted:

$$h(\boldsymbol{x}; \boldsymbol{w}, b) = \frac{1}{1 + \exp\{-\boldsymbol{w}^T \boldsymbol{x} + b\}} \,. \tag{7.2}$$

The resulting model is linear in the β parameters and the parameters $\boldsymbol{w}_1, ..., \boldsymbol{w}_C$ are initially assigned randomly before the training process, according to an uniform probability distribution. Under few assumptions on the smoothness of the underlying function, the RVFL has universal approximation capability if a large number of hidden functions is provided, that is forcing C to be large enough [201].

The problem of learning with an RVFL model of the form (7.1) should be reorganized as a linear regression over the coefficients $\boldsymbol{\beta} = [\beta_1 \ \dots \ \beta_C]^T$. Considering a training set of N input-output pairs $\{\boldsymbol{x}_i, y_i\}$, $i = 1 \dots N$, a hidden matrix $\boldsymbol{H}$ can be organized in the following way:

$$\boldsymbol{H} = \begin{bmatrix} h_1(\boldsymbol{x}_1) & \cdots & h_C(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\boldsymbol{x}_N) & \cdots & h_C(\boldsymbol{x}_N) \end{bmatrix}, \tag{7.3}$$

while the output vector is $\boldsymbol{y} = [y_1 \ \dots \ y_N]^T$.

The optimization procedure can be expressed as a standard regularized least-squares (RLS) problem where the optimal $\boldsymbol{\beta}$ for training a RVFL is found by minimizing the following objective function:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^C} \frac{1}{2} \left\| \boldsymbol{y} - \boldsymbol{H}\boldsymbol{\beta} \right\|^2 + \frac{\lambda}{2} \left\| \boldsymbol{\beta} \right\|^2 , \tag{7.4}$$

where $\lambda > 0$ is the *regularization factor*. The problem (7.4) is strictly convex, so the solution can be found by setting the gradient equal to 0, from which we obtaining the well-known solution:

$$\boldsymbol{\beta}^* = \left(\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{H}^{\mathrm{T}}\boldsymbol{y}\,. \tag{7.5}$$

7.3 A Finite Precision Model of RVFL Networks

In this section, it is firstly tested what happens when all the introduced parameters, input and output values, and the computation of hidden functions are subject to a uniform quantization. Successively, a non-uniform distribution of the input interval is also investigated to underline how the results change. Namely, it is proposed a non linear A/D conversion of input signals by considering the actual structure of data to be processed. In both cases, the genetic optimization is used to tune their implementation on hardware architectures based on a finite precision arithmetic.

7.3.1 Uniform Quantization

In a finite precision implementation of a RVFL network all the introduced parameters, input and output values, and the computation of hidden functions are subject to a uniform quantization. Let $q_n(\cdot)$ be a uniform quantizer in the respective range of the input variable, with a two's complement binary representation using n bits. It is applied element-wise if the input is either a vector or a matrix. With respect to the previous processing, given a same generation of random of weights, a quantized hidden matrix $\boldsymbol{H}^{(n)}$ will be arranged on the basis of the following hidden functions:

$$h^{(n)}(\boldsymbol{x}; \boldsymbol{w}, b) = q_n\left(\frac{1}{1 + \exp\left\{-q_n(\boldsymbol{w}^T)q_n(\boldsymbol{x}) + q_n(b)\right\}}\right)\,. \tag{7.6}$$

Then, the generic output for an n-bit finite precision implementation of a RVFL will be:

$$f^{(n)}(\boldsymbol{x}) = q_n\left(\sum_{m=1}^{C} \beta_m^{(n)} h_m^{(n)}(\boldsymbol{x}; \boldsymbol{w}_m)\right)\,. \tag{7.7}$$

Since the weights are extracted from a uniform distribution, the parameters $\boldsymbol{w}_1, ..., \boldsymbol{w}_C$ can be considered as forced to be quantized, as well as the results of the hidden functions, in such a way that the intrinsic randomness of the RVFL input and inner layers is still preserved. However, a main problem arises in the computation of finite precision $\boldsymbol{\beta}^{(n)}$ parameters in (7.12), which can be reformulated as an Integer Least Squares (ILS) problem:

$$\begin{aligned} \min_{\boldsymbol{\beta}^{(n)} \in \mathbb{R}^C} & \frac{1}{2} \left\| \boldsymbol{y} - q_n\Big(\boldsymbol{H}^{(n)} \boldsymbol{\beta}^{(n)}\Big) \right\|^2 + \frac{\lambda}{2} \left\| \boldsymbol{\beta}^{(n)} \right\|^2 \\ \text{s.t.} \quad & \boldsymbol{\beta}^{(n)} \in \mathbb{Z}^{(n)} , \end{aligned} \tag{7.8}$$

where $\mathbb{Z}^{(n)}$ represents here a generic set of integers to which quantized values can be assimilated. Also, due to the nonlinear nature of the output quantization, the finite precision RVFL in (7.12) is no longer linear in the $\beta^{(n)}$ parameters.

ILS is a common problem in many fields of signal processing as, for example, channel coding, cryptography, radar imaging and global positioning [202, 203]. It has been shown that ILS is an NP-hard problem and the algorithms for solving it have exponential complexity [204, 205]. Computationally, an ILS problem is equivalent to a LS where there is a constraint on the quantization of the result. In fact, a constraint on the bounded representation of the weights can be treated as a box constraint and solved similarly to an ILS problem regardless of the bounds [206].

A straightforward approximation of the ILS solution, which will be the baseline for the successive experiments, is to quantize with a two's complement representation using n bits the result obtained from (7.5):

$$\boldsymbol{\beta}_{\text{rnd}}^{(n)} = q_n \left(\left(\boldsymbol{H}^{(n)T} \boldsymbol{H}^{(n)} + \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{H}^{(n)T} q_n(\boldsymbol{y}) \right) , \tag{7.9}$$

where outputs $\boldsymbol{y}$ are in general quantized, as this learning step could be performed on a fixed point hardware as well.

As previously said, such an approach is only an approximation of the correct ILS solution and the performance of the resulting RVFL may be worse due to the nonlinear nature of the overall processing system. For such a reason, we propose in the following an optimization of the NP-hard nonlinear ILS problem, which is based on a GA approach in order to obtain an optimal set of quantized $\beta^{(n)}$ parameters.

Genetic Optimization

GAs are adaptive or meta-heuristic search algorithms used for solving both constrained and unconstrained optimization problems. They are inspired by the process of natural selection that belongs to the biological evolution [207]. Any GA starts from a population of candidate solutions, called individuals, and repeatedly modifies them using different operators (selection, mutation and crossover) in an iterative process obtaining a succession of sets of individuals (i.e., the generations). The population 'evolves' over successive generations, toward an optimal solution obtained by the improvement of the fitness of the best individual.

The GA starts from a generation of completely random individuals and the successive generation is obtained through the application of selection, mutation and crossover operators. In other words, at each generation the fitness of each individual is evaluated and a new solution is randomly created from the current one considering the fitness values. Successively, the solution is modified (mutated or recombined) to form the new generation.

The quantized parameters $\beta^{(n)}$ we are searching for are binary coded in a finite precision arithmetic. These are combined to form the so called chromosome, which is a string of bits each one treated as a gene. In the proposed approach, the genome of each individual is represented by a binary string of nC bits as illustrated in Fig. 7.4.

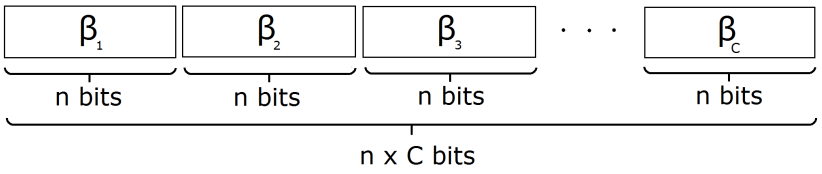


Figure 7.1: Visual scheme of the binary organization of a $\boldsymbol{\beta}^{(n)}$ solution.

Given a dataset on which the GA optimization is being performed, the quantized weights and the related quantized hidden functions $h_m^{(n)}$, $m = 1 \dots C$, are computed once for all at the beginning of the process. Then, for each individual and in every generation, the output (7.12) is computed for all inputs of the dataset by using the particular instance of $\boldsymbol{\beta}^{(n)}$ associated with the chromosome of the individual. The fitness of the individual is the

Noise-to-Signal Ratio (NSR):

$$\mathrm{NSR}_{\mathrm{dB}} = 10 \log_{10} \frac{\sum_{i=1}^{N} \left(y_i - f^{(n)}(\boldsymbol{x}_i)\right)^2}{\sum_{i=1}^{N} \left(y_i - \bar{y}\right)^2}, \tag{7.10}$$

where $\bar{y}$ is the average value of the output values y_i on the dataset. Thus, the lower is the NSR the better is the fitness.

The general steps of the GA can be summarized as:

1. A population of chromosomes G_0 with P individuals is created and set as the current generation.

2. The chromosomes are evaluated by a defined fitness function and sorted by ascending values of it.

3. Some chromosomes are selected from the current generation for performing genetic operations. Cloning, mutations and crossover are applied and the produced offspring replaces their parents in the next generation.

4. The next generation becomes the current one.

The steps 2-4 are repeated for a predefined fixed number $M_{\mathbf{gen}}$ of generations. The goodness of the performance of the algorithm relies on the values of P and $M_{\mathbf{gen}}$ as well as on the mutation rate (M_r) and on the crossover rate (C_r). The last one are used to control the rate at which mutation and crossover occur in terms of probability thresholds. The next generation G_{k+1} is produced from the current one G_k in the Step 3 of the previous algorithm as in the following:

1. The last two individuals of G_k are deleted.

2. The best individual of G_k is guaranteed to survive to the next generation, for that it is cloned and put in G_{k+1} (elitism).

3. The algorithm selects a fraction equal to M_r of the mutation rate to update the second individual of G_k by using a 'Uniform' function, then it is put in G_{k+1}.

4. A "Roulette Wheel" procedure is used to randomly select a pair of parents. A probability threshold C_r is used to produce the offspring of two parents by means of a "two-point" crossover. Successively, each of the two resulting individuals is mutated with a probability equal to

Table 7.1: Detailed Description of Datasets

Dataset	Features	Instances	Desired output
Airfoil	6	1503	Pressure level
Concrete	9	1030	Compressive strength
Energy	8	768	Heating Load
Istanbul	8	536	Stock exchange returns

M_r and placed in G_{k+1}. This step is repeated until the next generation contains exactly P individuals.

The successive simulations will be performed by fixing the following parameters: $P = 100$, $M_{\mathbf{gen}} = 100$, $C_r = 0.8$, and $M_r = 0.01$, while the evolutionary process is stopped if the best fitness stalls for 5 consecutive generations within a relative interval of $\pm 0.01\%$. The optimal solution found at the end of the GA optimization will be denoted as $\boldsymbol{\beta}_{\text{gen}}^{(n)}$.

Description of the Experiments

Here, we evaluate the performances of the proposed approach on four public datasets summarized in Table 7.6, which are available on the UCI repository (https://archive.ics.uci.edu/ml/datasets.html):

- *Airfoil* is a NASA dataset where data of two or three dimensional airfoil blade subsections conducted in an anechoic wind tunnel are acquired at various tunnel speeds and angles of attack. The aim is to predict the scaled sound pressure level [208].

- *Concrete* is considered in order to predict the compressive strength, which is the most important material in civil engineering, several quantitative attributes of the material are described [209].

- *Energy* contains an energy analysis using 12 different building shapes. Varying several characteristics, like the glazing area distribution, the orientation and so on, the aim of the dataset is to assess the heating load requirement of buildings [210].

- *Istanbul* includes returns of Istanbul Stock Exchange with seven other international indexes [211].

Table 7.2: Optimal Number of Hidden Nodes (C) Found by The Inner-fold Cross-validation

Dataset	n=4	n=8	n=12	n=16	float-64
Airfoil	200	200	300	300	500
Concrete	200	200	200	500	400
Energy	200	200	500	500	500
Istanbul	300	300	300	300	300

The input and output values of datasets are normalized before training in order to accommodate every feature in the range between 0 and 1. The weights $\mathbf{w}$ of the sigmoid basis functions are extracted randomly from a uniform distribution over the interval $[-1, +1]$, while the scalars b are extracted randomly from a uniform distribution over the interval $[-0.1, +0.1]$. The performance is obtained through a 10-fold cross validation, where the overall NSR is obtained comparing actual values of outputs with the predicted ones in every fold. The tests were carried on on different machines resulting in different training times (15 to 30 minutes).

First of all, we consider the baseline solution standard rounding $\boldsymbol{\beta}_{\text{rnd}}^{(n)}$ parameters obtained from (7.9). Four different number of bits for the weight quantization n are considered: 4, 8, 12, and 16. The performance of such solutions is compared to the one obtained by a software implementation on standard computers using 64-bit floating point arithmetic, which will be assimilated to the analog result $\boldsymbol{\beta}^*$ in (7.5).

In order to compute the optimal number of hidden nodes C and the regularization factor λ, we executed an inner-fold cross-validation of the training data. In particular, we searched in a grid-search procedure the set $\{200, 300, 400, 500\}$ for the optimum number of hidden nodes C and the set $\{2^{-10}, 2^{-9}, \ldots, 2^{9}, 2^{10}\}$ for λ.

The optimum C and λ for every number of bits is reported in Table 7.2 and Table 7.3. It can be noted that the optimum value of C for Airfoil, Concrete and Energy datasets is low for $n = 4$ and $n = 8$ and it increases as the precision increases. Instead, for the Istanbul dataset it is constant (i.e., $C = 300$). The regularization factor has a different trend; as the number of bits varies, the behavior is almost the same for every dataset. In fact, it is as high as possible (i.e., $\lambda = 2^{10}$) for $n = 4$ and then decreases.

Table 7.3: Optimal λ Found by Inner-fold Cross-validation

Dataset	n=4	n=8	n=12	n=16	float-64
Airfoil	2^{10}	2^{8}	2^{-4}	2^{-10}	2^{-10}
Concrete	2^{10}	2^{2}	2^{-6}	2^{-10}	2^{-10}
Energy	2^{10}	2^{2}	2^{-5}	2^{-10}	2^{-10}
Istanbul	2^{10}	2^{2}	2^{-4}	2^{-5}	2^{-5}

Table 7.4: Performance (NSR) of Basic Rounding Vs. Bit Precision

Dataset	n=4	n=8	n=12	n=16	float-64
Airfoil	-23.816	-26.604	-28.858	-30.175	-30.237
Concrete	-6.644	-11.233	-13.181	-15.038	-15.150
Energy	-8.386	-18.025	-19.909	-23.738	-24.122
Istanbul	1.784	-5.420	-6.674	-6.722	-6.723

Taking the optimal choices of C and λ, the baseline performance in terms of NSR versus the number of bits is shown in Table 7.4. For every dataset, the NSR for values of n greater than 8 is similar to the 64-bit floating point precision, with differences as low as 0.1 dB. Therefore, it is worth considering optimized implementations where a number of bits equal to or lower than 8 is considered.

The results obtained by using the GA optimization, keeping the same choices of C and λ, are reported in Table 7.5. We outline that GA optimization improves the performance for $n \leq 12$, in the majority of cases. This can be explained taking into account that, for a higher number of bits, the huge cardinality of the search space makes more sparse and evanescent the GA approach in a relatively limited time.

Finally, for the sake of illustration, we report in Fig. 7.2 the comparison of actual and predicted values of Energy dataset using a software implementation with 64-bit floating point precision, while in Fig. 7.3 there is shown the GA optimization using 8 bits. Evidently, the behavior is comparable even with a numerical difference of about 5.5 dB.

Table 7.5: Performance (NSR) of Genetic Optimizer Vs. Bit Precision

Dataset	n=4	n=8	n=12	n=16
Airfoil	-25.784	-26.484	-29.067	-30.082
Concrete	-7.982	-11.537	-13.386	-14.673
Energy	-10.779	-18.595	-20.437	-23.137
Istanbul	1.531	-6.026	-6.619	-6.608

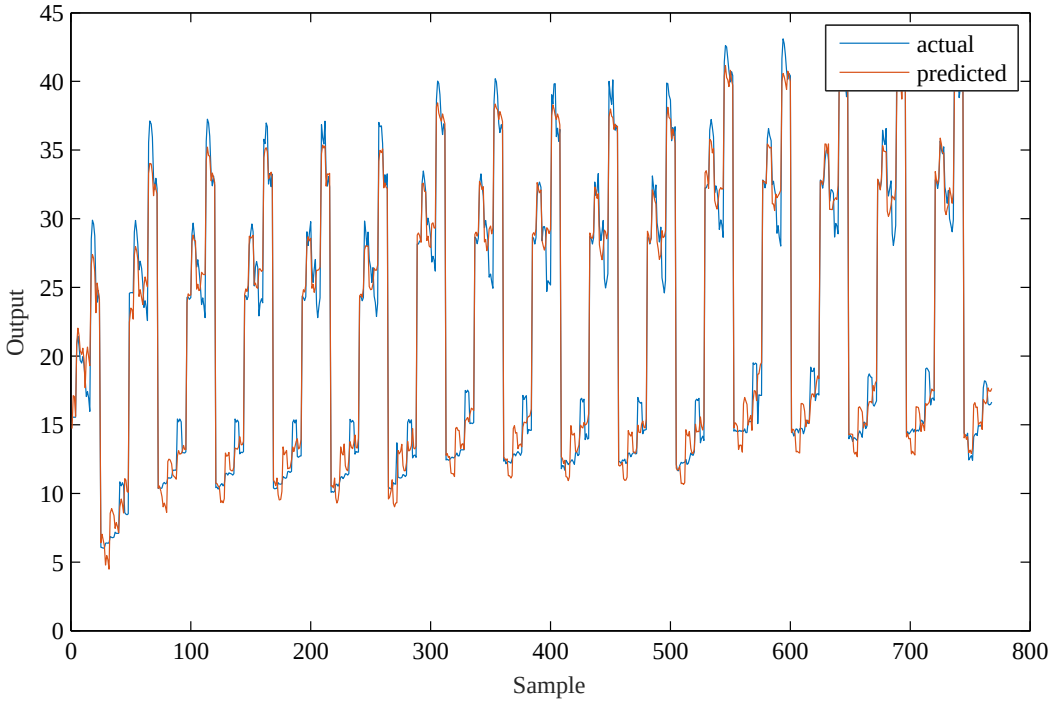


Figure 7.2: Output on Energy dataset using 64-bit floating point precision.

7.3.2 Nonuniform Quantization

In this section, we propose a nonuniform quantizer, in addition to the uniform quantization over all of the parameters of the network. This is done in order to cope with the actual structure of datasets and also to compensate the rounding of internal parameters of the neural model. In effect, it is well-known that, except in the case of the uniform distribution, the non-uniform quantization is superior in the sense that it results in a smaller average quantization error. For these reasons, it is assumed for the rest of the chapter, a nonuniform quantization of input values and a uniform quantization elsewhere.

Let $u_n(\cdot;\boldsymbol{\theta})$ be a nonuniform quantizer using n bits for representing the

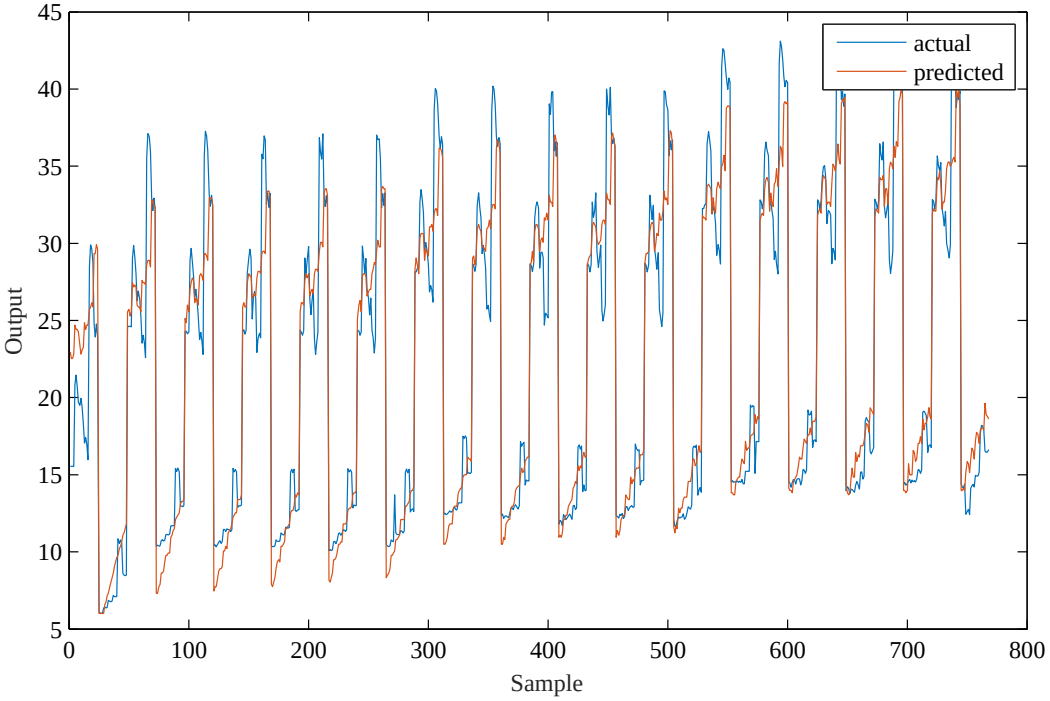


Figure 7.3: Output on Energy dataset using 8-bit precision optimized by GA.

quantization values. Since $\boldsymbol{x}$ is a d-dimensional array, the nonuniform quantizer is applied with different quantization values on each dimension. Thus, $\boldsymbol{\theta}$ is a $2^n \times d$ matrix of real numbers and the output $u_n(\boldsymbol{x}; \boldsymbol{\theta})$ of the quantizer will be a d-dimensional array where the jth element is $u_{n,j} = \theta_{rj}$, $j = 1 \ldots d$, being $\theta_{rj} \in \boldsymbol{\theta}$ the largest value no greater than x_j in the jth column of $\boldsymbol{\theta}$, for any $1 \leq r \leq 2^n$.

Consequently, given a same generation of random of weights, a quantized hidden matrix $\boldsymbol{H}^{(n)}$ will be arranged on the basis of the following hidden functions:

$$h^{(n)}(\boldsymbol{x}; \boldsymbol{w}, b, \boldsymbol{\theta}) = q_n \left(\frac{1}{1 + \exp\left\{ -q_n(\boldsymbol{w}^T) u_n(\boldsymbol{x}; \boldsymbol{\theta}) + q_n(b) \right\}} \right) . \tag{7.11}$$

Then, the generic output for an n-bit finite precision implementation of a RVFL will be:

$$f^{(n)}(\boldsymbol{x}) = q_n \left(\sum_{m=1}^{C} \beta_m^{(n)} h_m^{(n)}(\boldsymbol{x}; \boldsymbol{w}_m, \boldsymbol{\theta}) \right) . \tag{7.12}$$

Since the RVFL weights are extracted from a uniform distribution, the parameters $\boldsymbol{w}_1, ..., \boldsymbol{w}_C$ can be considered as forced to be quantized as well as

the results of the hidden functions, in such a way that the intrinsic randomness of the RVFL's inner layer is still preserved. By the way, the nonuniform quantization of input values should also improve the numerical accuracy in representing the quantized outputs of hidden functions, which undergo strong saturation effects.

However, a main problem arises in the computation of finite precision $\boldsymbol{\beta}^{(n)}$ parameters in (7.12), which can be reformulated as an Integer Least Squares (ILS) problem:

$$\begin{aligned} \min_{\boldsymbol{\beta}^{(n)} \in \mathbb{R}^C} & \frac{1}{2} \left\| \boldsymbol{y} - q_n\Big(\boldsymbol{H}^{(n)}\boldsymbol{\beta}^{(n)}\Big) \right\|^2 + \frac{\lambda}{2} \left\|\boldsymbol{\beta}^{(n)}\right\|^2 \\ \text{s.t.} \quad & \boldsymbol{\beta}^{(n)} \in \mathbb{Z}^{(n)} , \end{aligned} \tag{7.13}$$

where $\mathbb{Z}^{(n)}$ refers here as a generic set of integers to which quantized values can be assimilated. It has been shown that ILS is an NP-hard problem and the algorithms for solving it have exponential complexity [206]. Also, the ILS problem in (7.13) becomes nonlinear in the $\boldsymbol{\beta}^{(n)}$ parameters. The underlying idea of this paper is that an optimized nonlinear quantization might also compensate, and possibly regularize, the approximate solution of the said ILS problem. Therefore, we will consider in the following a straightforward approximation of the result obtained in (7.5), by using a finite precision representation:

$$\boldsymbol{\beta}\text{rnd}^{(n)} = q_n\left(\left(\boldsymbol{H}^{(n)T}\boldsymbol{H}^{(n)} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{H}^{(n)T} q_n(\boldsymbol{y})\right) , \tag{7.14}$$

where output values $\boldsymbol{y}$ are in general quantized, as they can be obtained from measures by digital equipments.

GA Optimization of Nonuniform Quantizer

Differently from what is proposed in the uninform quantization, here the GA is applied to the $\boldsymbol{\theta}$ parameters of the nonuniform quantizer and hence, the chromosome of each individual is represented by an array of $d{\cdot}2^n$ real numbers (i.e., quantization values) as illustrated in Fig. 7.4. It is a possible solution that implements a nonuniform quantizer.

Given a dataset on which the GA optimization is being performed, the quantized weights $\boldsymbol{w}_m$ are computed once for all at the beginning of the process. Then, for each individual and in every generation, the RVFL is trained using (7.14) and the hidden matrix $\boldsymbol{H}^{(n)}$, which is obtained through

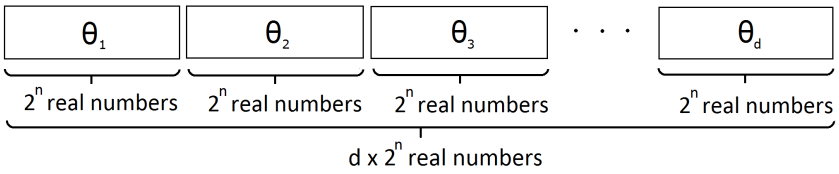


Figure 7.4: A chromosome defining the nonuniform quantizer, where $\boldsymbol{\theta}_j^T$, $j = 1 \dots d$, is the jth column of $\boldsymbol{\theta}$.

(7.11) using the particular instance of $\boldsymbol{\theta}$ associated with the chromosome of the individual. Successively, the output on a test set is computed using (7.12). The fitness of the individual is the Noise-to-Signal Ratio (NSR) defined as in (7.10).

The steps of the GA are the same as in the uniform quantization procedure.

The goodness of the performance of the algorithm relies on the values of P and $M_{\mathbf{gen}}$ as well as on the mutation rate (M_r) and on the crossover rate (C_r). The successive simulations will be performed by fixing the following parameters: $P = 100$, $M_{\mathbf{gen}} = 100$, $C_r = 0.8$, and $M_r = 0.01$, while the evolutionary process is stopped if the best fitness stalls for 5 consecutive generations within a relative interval of $\pm 0.01\%$.

Description of the Experiments

In this section, we evaluate the performances of the proposed approach on the three different and well-known and commonly consolidated datasets summarized in Table 7.6, which are available on the UCI repository (https://archive.ics.uci.edu/ml/datasets.html):

- *Airfoil*. The aim is to predict the scaled sound pressure level when different tunnel speed or angles of attack are applied on airfoil blade section.

- *Concrete*. Several quantitative attributes of civil materials are used to predict the compressive strength, which is the most important one.

- *Energy*. Twelve different building shapes, with several characteristics, are used to assess the heating load requirement of buildings.

Both input and output values of datasets are normalized before training in order to accommodate every feature in the range between 0 and 1. The

Table 7.6: Description of The Adopted Datasets

Dataset	Features	Instances	Desired output
Airfoil	6	1503	Pressure level
Concrete	9	1030	Compressive strength
Energy	8	768	Heating Load

weights $\mathbf{w}$ of the sigmoid basis functions are extracted randomly from a uniform distribution over the interval $[-1, +1]$, while the scalars b are extracted randomly from a uniform distribution over the interval $[-0.1, +0.1]$. Every test result of RVFL is obtained through a 10-fold cross validation, where the overall NSR is obtained comparing actual values of outputs with the predicted ones in every fold. Nonetheless, given the stochastic nature of the RVFL initialization, as well as for GAs, all the performances reported in the following are an average of the results obtained by repeating the same experiment over 10 independent runs.

First of all, we consider the baseline solution where a finite precision RVFL is trained and tested using the above steps and a uniform quantization of input values. Four different values of n (i.e., bits for quantization) are considered: 4, 8, 10, and 12. The performances are compared with respect to the one obtained by a software implementation on a standard computer or DSP using 64-bit floating point arithmetic. Higher values of n are not considered, because in such cases the hardware complexity starts to be considerable and the performances are very close to the one obtained using a floating point arithmetic.

In order to compute the optimal number of hidden nodes C and the regularization factor λ, we executed an inner-fold cross-validation of the training data. In particular, we performed a grid-search procedure where C was varied from 50 to 500 in a step of 50 and λ was in the set $\{2^{-10}, 2^{-9}, \ldots, 2^{9}, 2^{10}\}$. The optimum values of C and λ for every number of bits is reported in Table 7.7 and Table 7.8, respectively. It can be noted that the optimum value of C tends to increase as the precision increases, whilst λ decreases.

Taking the optimal choices of C and λ, the performances in terms of NSR versus the number of bits are shown in Table 7.9. As expected, it is worth considering optimized implementations of finite precision RVFLs using 12 or a lower number of bits, as the related performances are different from the one using a floating point architecture.

Table 7.7: Optimal Hidden Nodes (C) Found by Cross-validation

Dataset	n=4	n=8	n=10	n=12	float-64
Airfoil	50	50	50	50	500
Concrete	50	100	50	150	500
Energy	150	50	50	250	500

Table 7.8: Optimal Regularization Factor (λ) Found Cross-validation

Dataset	n=4	n=8	n=10	n=12	float-64
Airfoil	2^{9}	2^{0}	2^{-3}	2^{-7}	2^{-10}
Concrete	2^{8}	2^{2}	2^{-4}	2^{-6}	2^{-10}
Energy	2^{9}	2^{0}	2^{-4}	2^{-5}	2^{-10}

Table 7.9: Performance (NSR) using a Uniform Quantizer of RVFL Inputs

Dataset	n=4	n=8	n=10	n=12	float-64
Airfoil	-24.358	-28.055	-28.339	-28.890	-30.369
Concrete	-6.635	-11.233	-11.883	-12.997	-15.326
Energy	-8.578	-18.061	-18.920	-19.862	-24.433

Table 7.10: Performance (NSR) using a GA-optimized Nonuniform Quantizer of RVFL Inputs

Dataset	n=4	n=8	n=10	n=12
Airfoil	-24.453	-28.165	-28.362	-28.851
Concrete	-7.105	-11.573	-11.855	-13.202
Energy	-8.629	-19.424	-21.682	-24.262

The results obtained by using the GA optimization, keeping the same choices of C and λ, is reported in Table 7.10. While the improvements on Airfoil are not relevant, as it probably contains noisy and spread data, for Concrete and Energy the NSR decreases of 1 or 2 dB, more significantly with a higher number of bits as the space of solution enlarges and there is more margin to found a better local optimum.

To confirm the efficacy of the proposed approach, we report in Fig. 7.5

the quantization levels obtained in the case of a 10-bit finite precision implementation of a RVFL trained on the Energy dataset. The optimal solution found by the GA determines an evident nonuniform distribution of the quantization levels, which is also different for each dimension. Finally, we report

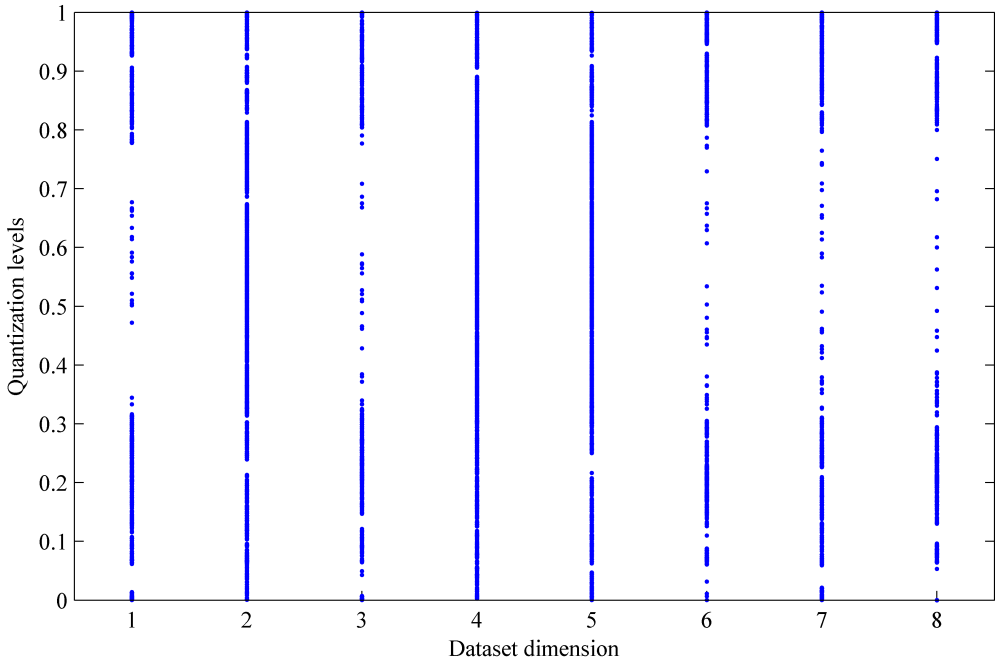


Figure 7.5: Quantization levels of a 10-bit nonuniform quantizer optimized by GA on the Energy dataset.

in Fig. 7.6 the comparison of actual and predicted values of Energy dataset using a 8-bit nonuniform quantizer optimized by GA. Evidently, the behavior is comparable with a numerical difference of about 1.4 dB and for NSR values around 19 dB.

7.4 Conclusion

In this work, a method for implementing of RVFL networks on finite precision hardware architectures is proposed. A genetic algorithm is adopted for optimizing the performances of a uniform quantizer applied to all the parameters in the algorithm and a nonuniform quantizer is applied to input values only, to optimize the performances even when a low number of bits is adopted. For the uniform quantization, the performance gap between the basic rounding and the GA optimization shrinks as the number of bits increases, while the GA procedure obtains better performance using even 4 and 8 bits. Also, the

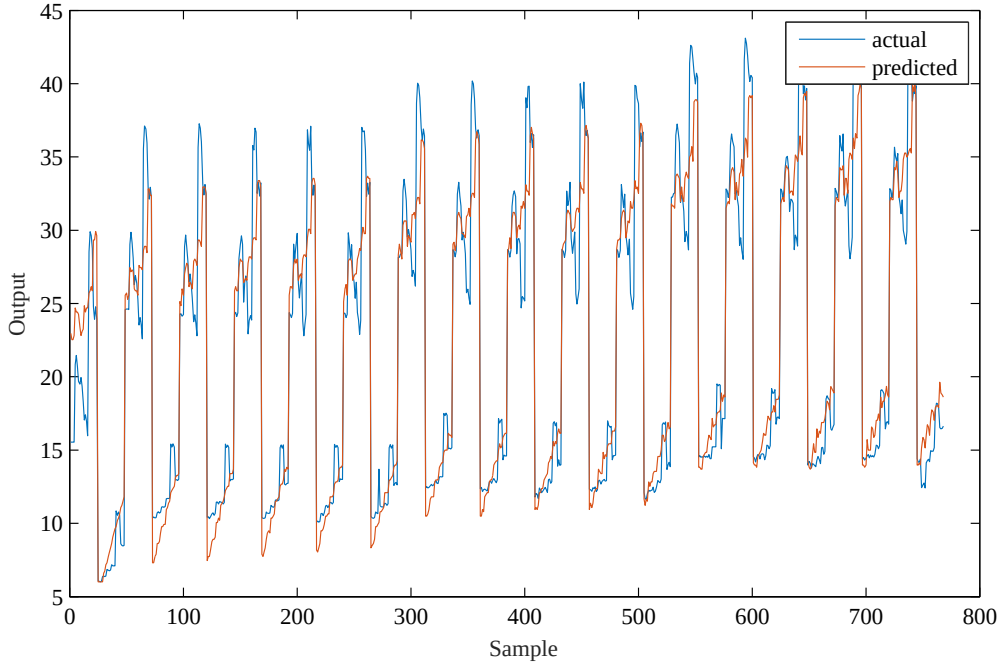


Figure 7.6: Output on Energy dataset using a 8-bit precision and GA optimization.

numerical results obtained with the nonuniform quantizatin show that the proposed approach is effective even using a small number of bits and it compensates the effects of rounding in the successive layers of the neural network. Future works could focus on other nature-inspired optimization process, like for example particle swarms, or might consider hardware architectures and finite precision adaptive, online strategies reflecting the numerical results herein obtained.

Chapter 8

Remote Water Quality Prediction Monitoring

8.1 Motivation

A novel context where distributed computing is receiving an increasing interest is the forecasting and monitoring of physico-chemical parameters in water reservoirs using satellite images. This class of remote sensing applications implies fast, precise and accurate predictions that can avoid the utilization of much more complicated in situ analysis methodologies.

Freshwater ecosystems are an important natural resource, essential for multiple purposes such as drinking, domestic use, industrial cooling, power generation, agriculture, waste disposal, and transportation routes [212]. Therefore, many scientists have studied and published papers on the influence of freshwater physico-chemical-parameters to the changing ecosystems. The prediction of such water quality variables is a very important aspect for the analysis of any aquatic system. The time evolution of physico-chemical variables of aquatic ecosystems is very complex and nonlinear. Therefore, the research on the various methods for prediction of water quality reservoir has important theoretical value and practical significance.

One of the main processes leading to water deterioration in lentic environments is the eutrophication, which is mostly caused by anthropogenic activities, as well as the releases of industrial and domestic effluents into water bodies. Physico-chemical water parameters typically related to eutrophication are, among others, Chlorophyll-a (C), Turbidity (T) and Suspended Solids (SS), and can be used to assess the eutrophic state of wa-

ter bodies [213, 214]. Generally, these parameters are analyzed through a manual measurement after taking water samples from laboratory to analyze their concentrations. These processes are time-consuming and require trained personnel. There already exist numerous water quality prediction methods [215–217], mainly monitoring the water quality of reservoirs using artificial neural network (ANN) [218], fuzzy neural network [219], series prediction model of water quality fused with ARIMA and neural network [220], regression analysis [221], gray theory [222], time series analysis [220], fuzzy reasoning [223], satellite remote sensing [224], wavelet transform model [225] and Adaptive Neuro Fuzzy Inference System (ANFIS) [226]. These techniques were applied alone or together with other computational techniques to support the evaluation of spatial and seasonal variations in water quality.

Artificial neural networks (ANNs) with cross-validation have recently gained attention in the literature of water quality monitoring [214, 226–232]. Moreover, an information-theoretic (IT) approach to model-based inference has been used to simulate the changing ecosystems [233–235].

The key physico-chemical parameters that can influence water ecosystems have been found to be: Chlorophyll-a Levels (C); Total Suspended Solids (TSS); Transparency/Turbidity (T). They have been found to be important factors in the water quality monitoring [215, 236–238]. The methods currently used for water analysis are time-consuming and extremely costly, because they require sample collection, trained people and specialized laboratories. We propose to analyze water quality by ANNs models and remote sensing technology. The algorithm to retrieve water parameters from satellite images has been developed by using 'in situ measurements' and a series of Landsat images. In this work an IT method is proposed to predict C, TSS and T parameters, focusing on ANN and Leave-One-Out (LOO) cross-validation. However, to focus the scope of the analysis and further pursue the goal of this work, some remarks should be drawn regarding the physical properties and interactions between sunlight and water bodies. To be able to discriminate parameters such as TSS from the water reflectance, it is very important to take into account the absorption and scattering properties of the water bodies. These properties depend on the wavelengths of the electromagnetic spectrum. In the whole 400-1300 nm region of the spectrum, the scattering is due mainly to solid sediments. Instead, the absorption for the same spectrum region is regulated mainly by Chlorophyll-a and other organic materials. We note that these properties are much more evident when considering radiation with

wavelength under 500 nm and are almost negligible otherwise.

In this analysis, it is also very important to be able to discriminate between the water radiance and the radiance of the light scattered by the bottom surface of the water body. Fortunately, the wavelengths in which we are interested are affected only lightly by this confusion. In particular, the spectral and spatial resolution must be adequate in order to analyze the water parameters. For instance, Thematic Mapper (TM) is appropriate for the study of water bodies that are not too large, but its spectral resolution is not enough to determine some water quality variables. Other sensors have better spectral resolution but a too coarse spatial resolution. In this work, the Enhanced TM (ETM) is chosen and the analysis is done combining multiple bands to balance out the poor spectral resolution. This is the reason why we used different images and different spectral bands, to be able to reconstruct and wholly analyze the chosen water bodies.

8.2 Methodologies

The proposed approach for monitoring water quality parameters is based mostly on Neural Networks and satellite images. First of all the images from Landsat 7 sensor ETM+ and Landsat 8 were acquired. Typically, in order to obtain and predict the C, TSS, and T concentration from the reflectance of the satellite images, all the satellite image bands from visible and NIR were first calibrated for radiance values and, subsequently, for reflectance values. After that, they are given in input to an Artificial Neural Network (ANN) for the actual analysis.

Image-based methods for atmospheric correction can estimate path radiance without using atmospheric properties, their accuracy is highly dependent on what is captured in a scene, as described in many papers [239–243]. The characteristics of the analyzed bands are reported in Table 8.1.

Table 8.1: Characteristics of visible and NIR bands of the analyzed sensors.

Satellite	Bands	Spectral Resolution [nm]	Spatial Resolution [m]	Temporal Resolution [days]
Landsat7(ETM+)	TM1	450-520	30	16
Landsat7(ETM+)	TM2	520-600	30	16
Landsat7(ETM+)	TM3	630-690	30	16
Landsat7(ETM+)	TM4	760-900	30	16

The multitemporal Landsat images and the linear mixed models were also used and applied for the determination and quantification of spectral reflectance optical factors of dispersed elements in the water reservoir, drinking water management and several other fields [229]. However, there are scant reports analyzing seasonal cycles and recent satellite images as Landsat-8. One of the problems with the traditional neural network model is associated with the choice of the network structure. The training parameters can easily fall into local minima, seriously affecting the precision and reliability of the model [225]. Specifically, when these methods are directly applied to physico-chemical parameters prediction, the results are often unsatisfactory.

Based on the previous applications cited above, in this work we propose a method for predicting C, T, and SS in the water reservoir. By applying Wavelet Artifical Neural Networks (WANN) and remote sensing techniques to assess water quality, with respect to the seasonal cycle, we implement a simple and operative method that can be available for the monitoring of multiple variables related to the ecological systems with precise and less expensive sampling than the methods currently used for analysis of water in reservoirs. The proposed method consider the seasonal characteristics of a region, analyzing full cycle (March to May), emptying cycle (June to August), dry cycle (September to November) and a filling cycle (December to February) of the water reservoir; the neural network is trained by hydrological cycle. Four images are collected per year, corresponding to each hydrological cycle of each collection water body.

8.2.1 Landsat 7, ANN and LOO

The chosen Landsat 7 ETM+ satellite has a spatial resolution of 30 m for the six reflective bands, 60 m for the thermal band, and includes a panchromatic (pan) band with a 15 m resolution. L7 has a 378 Gigabit (Gb) Solid State Recorder (SSR) that can hold 42 min (approximately 100 scenes) of sensor data and 29 h of housekeeping telemetry concurrently (L7 Science Data User's Handbook, available at `http://landsathandbook.gsfc.nasa.gov/handbook.html`).

The proposed method for this satellite corrects a path radiance spectrum estimated by the dark object subtraction (DOS) method so that the spectrum meets general spectral characteristics of path radiance. The atmospheric effects that influence the signal registered by remote sensors might be mini-

mized in order to provide reliable spectral information. In aquatic systems, the application of atmospheric correction avoids the under or overestimation of remote sensing reflectance (Rrs). An accurate Rrs provides better information about the state of aquatic system, establishing more precisely the concentration of aquatic compounds [244]. In this part of the study, the DOS method with semi-automatic classification plug-in is used, as described elsewhere [245, 246]. Afterward, a relative scattering model is chosen based on the atmospheric conditions of the image at the acquisition time and the initial haze value for the other spectral bands is then calculated. Equations and parameters to convert calibrated Digital Numbers (DNs) to physical units, such as at-sensor radiance and reflectance, have been presented in a 'sensor-specific' manner elsewhere [247]. These values are calculated for each band after the normalization for the visible and NIR bands, by using the following formulas:

$$\mathrm{L}_\lambda = \mathrm{LMIN}_\lambda + (Q_{\mathrm{cal}} - Q_{\mathrm{calmin}}) \left(\frac{\mathrm{LMAX}_\lambda - \mathrm{LMIN}_\lambda}{Q_{\mathrm{calmax}} - Q_{\mathrm{calmin}}} \right), \tag{8.1}$$

or

$$\mathrm{L}_\lambda = Q_{\mathrm{cal}} \times \mathrm{GAIN\_BAND} + \mathrm{OFFSET\_BAND}, \tag{8.2}$$

where:
L_λ: Spectral radiance at the sensor's aperture [W/(m^2 × sr × μm)],
$\mathrm{Q}_{\mathrm{cal}}$: Quantized calibrated pixel value [DN],
$\mathrm{Q}_{\mathrm{calmin}}$: Minimum quantized calibrated pixel value corresponding to LMIN_λ [DN],
$\mathrm{Q}_{\mathrm{calmax}}$: Maximum quantized calibrated pixel value corresponding to LMAX_λ [DN],
LMIN_λ: Spectral at-sensor radiance that is scaled to $\mathrm{Q}_{\mathrm{calmin}}$ [W/(m^2 × sr × μm)],
LMAX_λ: Spectral at-sensor radiance that is scaled to $\mathrm{Q}_{\mathrm{calmax}}$ [W/(m^2 × sr × μm)],
GAIN_BAND: Band-specific rescaling gain factor [(W/(m^2 × sr × μm))/DN].

An ANN is a parallel-distributed processor that resembles the human brain by acquiring knowledge through a learning process and then stores the knowledge in the connection strength between computational units called

neurons [248]. We used for this purpose a simple feed-forward neural network in which the information moves in only one direction, from the input to the output nodes. The preprocessed data is given as the input of the ANN, and the outputs are: C, TSS, and T. In Fig. 8.1 there are shown the inputs of the neural network considering the processed image of Landsat satellite, sensor ETM+, band 1, band 2, band 3 and band 4. The process is repeated for each year from 2007 to 2014. As a result, estimates of C, T, and TSS are obtained at the output.

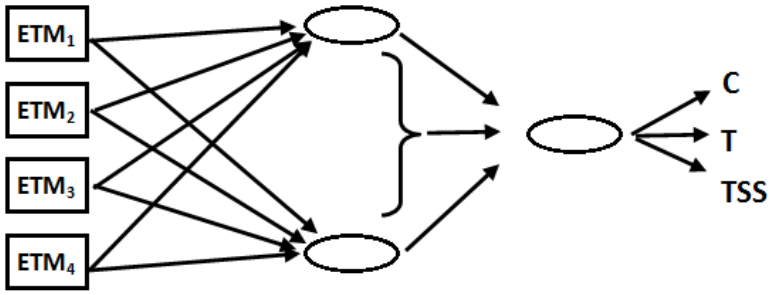


Figure 8.1: Diagram of the adopted ANN.

We used the LOO method to partition the dataset into training set (Tr) and test set (Ts). We split the data set $\mathcal{D}$ of size N into N partitions of size 1 such that:

$$\mathcal{D} = \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup ... \cup \mathcal{Q}_{N-1} \cup \mathcal{Q}_N \tag{8.3}$$

with $\mathcal{Q}_i \cap \mathcal{Q}_j = 0$ for every $i \neq j$. Each partition $\mathcal{Q}_i$ is used systematically for testing exactly once whereas the remaining partitions are used for training. Let $\mathcal{P}_i = \mathcal{D} - \mathcal{Q}_i$ be the training set with respect to the test partition $\mathcal{Q}_i$, $i = 1 \ldots N$; then, we can compute the error for each test partition considering such a trained model. The error over all partitions is considered as the LOO performance (error) [37], [249].

8.2.2 Landsat 8 and WANN

Landsat-8 data are obtained from the U.S. Geological Survey (USGS), they include spectral band 2 (Blue), band 3 (Green) and band 4 (Red) (available at https://espa.cr.usgs.gov/). All satellite images used were calibrated for radiance values and, subsequently, for reflectance values, as described in many papers [239–242].

In this case, atmospheric interference calculation and correction in the satellite images were done using an improved Dark Object Subtraction (DOS) method for Landsat-8 multispectral satellite image [250]. DOS is perhaps the simplest yet most widely used image-based absolute atmospheric correction approach for classification and change detection applications [250–252]. DOS approach assumes the existence of dark objects (zero or small surface reflectance) throughout a Landsat scene and a horizontally homogeneous atmosphere. The minimum digital number (DN) value in the histogram from the entire scene is thus attributed to the effect of the atmosphere and is subtracted from all the pixels as described in [253]. This method was chosen so the spectrum meets general spectral characteristics of path radiance. These include bad weather conditions, such as clouds, that affect both the amount of incoming solar radiation reaching the water surface and the fraction of light leaving the water surface and reaching the satellite sensor [254]. DOS method with semi-automatic classification plugin was used, as described elsewhere [246].

The reflectance calibration of the Operational Land Imager (OLI) was performed using methods based on atmospheric modeling as showed in (8.4). Various models exist with different degrees of spectral resolution, number of atmospheric constituents, cloud management, etc.. These models are generally based on the radiative transfer equation, which is given below in its generalized form.

$$L_{sensor} = \underbrace{\frac{E_o * cos(\theta) * \tau * \rho}{\pi} + \frac{E_d * \tau * \rho}{\pi}}_{\text{Ground\_reflected}} + L_{path}, \tag{8.4}$$

where:

L_{sensor}: Radiance received at the sensor.

E_o: Solar radiance at the top of the atmosphere.

θ: Incidence angle of solar radiance on the surface (0 for vertical, 90 for horizontal).

τ: Transmittance factor .

ρ: Reflectance factor.

E_d: Scattered background radiation (of the sky).

L_{path}: Path scattered radiation reaching the sensor.

This equation describes the core process of many models. Atmospheric models need to take into account the complex time and space varying composition of the atmosphere, as well as the wavelength dependent interactions caused by the radiance reaching the sensor. The primary characteristics of the OLI instrument relevant to this paper are presented in table 8.2, reporting characteristics of the analyzed bands: band 2, band 3 and band 4. From here, the spectral bands will be referred to by their band number, name or center wavelength.

Table 8.2: Landsat-8 Operational Land Imager (OLI).

Landsat-8			
Band	***Band Name***	***Center Wavelength(nm)***	***Bandwidth(nm)***
2	Blue	482	60
3	Green	561	57
4	Red	655	38

Wavelet artificial neural network (WANN) is a mathematical modeling method combining wavelet transform with an artificial neural network; it has emerged as an effective tool to simplify the non-stationarity in the dataset and has been widely applied by coupling it with neural networks for water resource variables forecasting [255–257]. The learning associated to the weights is easier than in conventional neural networks. The error function is a convex function and has a nonlinear approximation, strong fault tolerance, fast convergence speed and good prediction results. WANN has been effectively applied in signal processing, predictive control, fault diagnosis, and pattern recognition [258–260]. There are, however, scarce reports for analysis of the Landsat-8 surface reflectance and image processing.

Therefore, this study aids to analyze water quality using a hybrid system, WANN and remote sensing techniques to estimate and predict water parameters from satellite images. In order to develop the WANN model, wavelet sub time series were generated by using discrete wavelet transformation (DWT) and applying Haar Wavelet Transformation with one decomposition Level. These sequences are used as inputs to the WANN model. Haar Wavelet Transformation is a simple form of compression which involves averaging and differencing terms, storing detail coefficients, eliminating data

and reconstructing the matrix such that the resulting matrix is similar to the initial one [261, 262].

This approach is based on neural networks and fuzzy logic, and the inference system is treated as a function approximation problem. An ANFIS neural network implements a fuzzy inference system as in previous chapters.

The wavelet transform is an integral transform whose kernel is a class of special functions, called wavelets [263]. The main advantage of this method, compared to other methods, is in its spectral location capability in space and frequency, which allows the analysis of non-stationary signals in their various scales [264] [265]. WANN was trained by season, four images are collected per year, corresponding to four different cycles of the reservoir.

Its essence is to express a function through the telescopic or translation of basic *mother* wavelet function $\psi(t)$.

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) \tag{8.5}$$

$$y^{(k)} = \sum_{j=1}^{N} \omega_i \psi_j \left(\sum_{k=1}^{M} \frac{x_k * U_k j}{a_j}\right) , \tag{8.6}$$

where $\underline{x} = \begin{bmatrix} x_1 \; x_2 \cdots x_N \end{bmatrix}$ is the input, composed by the pixels of satellite images, and $y^{(k)}$ is the predicted output value (C, T, SS). U_{kj} is the connection weight from the kth node (input layer) to the jth node (hidden layer); ψ_j is the activation function of the jth neuron (hidden layer); ω_j is the layer weight from the jth nodes (hidden layer) to the output; a_j is the expansion parameter of wavelet function; and b_j is the translation parameter of wavelet function.

WANN is based on the topology and structure of the back propagation neural network. The wavelet basis function is the incentive function of the neurons. By using the advantage of the wavelet signal analysis and combining the function of the neural network training and prediction, the signal is transmitted forward and the error is propagated backward, so as to achieve a more accurate predictive value signal.

This study adopted a three-layer network structure: the input with m nodes, the hidden layer with n nodes, and the output with a single node [225]. The WANN network structure is shown in Fig. 8.2.

The validation process of the WANN was conducted with images from the year 2017. Fig. 8.2 shows the architecture of the ANN with the column vector

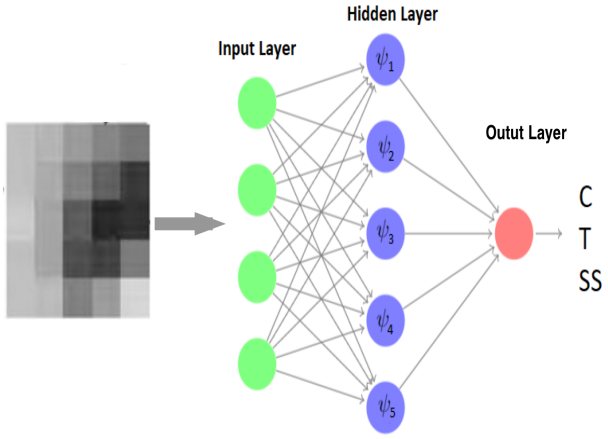


Figure 8.2: WANN

$P_i, (i = 1, 2, .., 768)$ as input. The WANN was trained with the following parameters: Learning rate 0.01, *tansig* transference function in all the neurons of the hidden layer and *purelin* in the output layer. The network training function is a gradient descent with momentum and adaptive learning rate. The wavelet transform used in the present study is the discrete transform, which allows for the multi-resolution analysis of a signal, decomposing said signal into approximations and details.

The approximations are high ranges, i.e., low-frequency signal components. The details are the low ranges, i.e., high frequency components [266] [267]. One sampling station was initially chosen for analysis and a geographic image of the water sampling station, of 32x32 pixels, was cropped, corresponding to an array containing 1024 pixels. Each digital pixel value corresponds to an average of radiance values, emittance or backscatter of the different targets that can be contained in the pixel from the vicinity of the water sampling station, as displayed in an example in Fig. 8.3.

Subsequently, the Haar wavelet transform was applied, with only one level of decomposition, resulting in a matrix array of 16x16 pixels for each of the following three components: Horizontal (H), vertical (V) and diagonal (D). The conversion of the arrays of the H, V and D components to their respective column-matrices was performed, and subsequently, a concatenation of the three arrays (each containing 256 pixels) was executed, generating a vector with 768 column size (256 x 3).

The images of the geographical area containing the water sampling collection point, decomposed via wavelet into its three wavelet components was used as the input for WANN. Preliminary tests were conducted considering

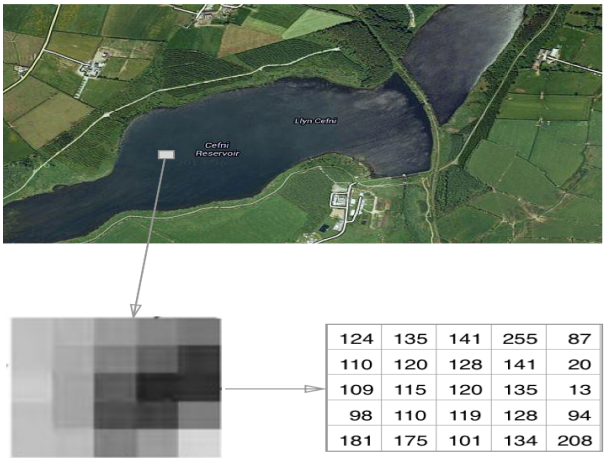


Figure 8.3: Pre Processing Images: Cefni reservoir with group of pixels area in gray values and corresponding digital numbers (DN)

the image representations isolated for each wavelet component, with satisfactory results. However, when the input data of the three wavelet components was considered, the approximations were even better, which motivated the choice of this arrangement in the proposed solution.

8.3 Case Studies

In this section, two case studies for the two presented methodlogies are reported.

8.3.1 Tucurui plant

Study Area

A water reservoir in northern Brazil was chosen as the first study area. It is considered as a deep reservoir, with a maximum depth of 77 m and an average depth of 19.8 m. The reservoir is located at coordinates: latitude 03^o 45' 03" S, longitude 49^o 40' 03" W. The power plant reservoir was built in Tocantins river, about 7 km from the city of Tucurui. The reservoir has a total flooded area of approximately 2850km^2, with approximately 50.8 million m^3 of water. In the Brazilian Amazonian region, there are 5 reservoirs in operation: Couracy Nunes, Curua Una, Tucurui, Balbina, and Samuel. The UHE Tucurui plant is a large-scale hydroelectric power plant that is located in the state of Para on the Tocantins River [268]. As previously stated, we propose remote monitoring of the reservoir using Landsat 7 in order to predict

the physico-chemical parameters of the water in seven points, as shown in Fig. 8.4. Water samplings were conducted at 7 sampling stations: Caraipu 1 (C1), Caraipu 2 (C2), Breu Branco (MBB), Jacundá Velho (MJV), Upstrem 1 (M1), Upstrem 3 (M3), Ipixuna (MIP). The images were collected for the years from 2007 up to 2014 and then, they were classified and converted to vector format; the output of the neural network has been validated with values observed in laboratory. The selected sampling stations are named: C1, C2, M1, M3, MBB, MJV, MIP.

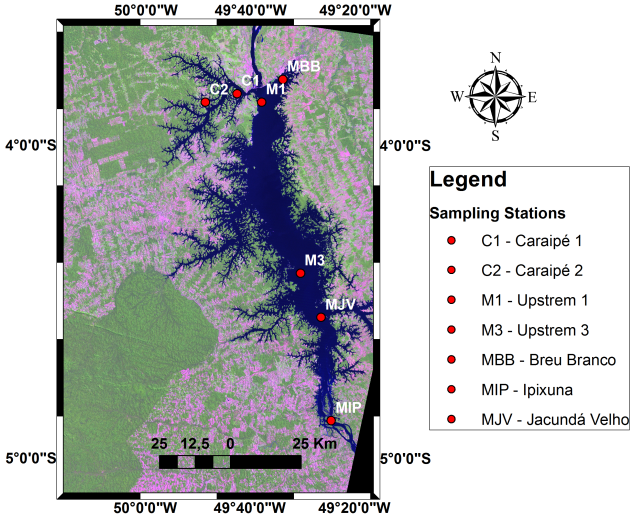


Figure 8.4: The area considered in this study.

The collection of water in these points were done periodically 4 times a year corresponding to the hydrological cycle: full, emptying, dry and filling. The different cycles are a consequence of the differences in rainfall during the year and they influence a lot the water monitoring. Hydroelectric plants currently constitute an indispensable component for supplying renewable energy. However, this reservoir, as the others in the Amazon region, has had several impacts on the ecosystem: loss of biodiversity of terrestrial and aquatic fauna and flora, high concentration of organic matter in the water bottom due to vegetation inundation, chemical changes in the water downstream, large volume of anoxic water in the reservoir and downstream, loss of water quality (low dissolved oxygen, high conductivity, low pH, high content of dissolved and particulate, organic matter), high concentration of aquatic macrophytes and reduction of fisheries downstream. The reservoir has impacted also on the human settlements in the area by weakening physical infrastructure, decreasing efficiency in land use, creating resettlement problems and influencing

mining operations on the reservoir itself [269]. For these reasons, the area is of high interest and water monitoring is one of the most important things that have to be in place to ensure the sustainability of the reservoir.

Experimental Results

The ANN results for each sampling station are shown in Table 8.3 and Table 8.4 for training and test sets, respectively. The values are considered in terms of mean squared error (MSE), considering the parameters to be estimated. The relative error E_r was calculated by the following formula and showed in Table 8.5:

$$E_r = \frac{|X_e - X_o|}{X_o}, \tag{8.7}$$

where X_e is the estimated value and X_o the observed value.

Table 8.3: MSE in the ANN training.

MSE					
Parameters	Station	Full	Emptying	Dry	Filling
C	C1	1.25×10^{-23}	3.70×10^{-07}	1.35×10^{-22}	6.68×10^{-24}
	C2	4.44×10^{-22}	3.97×10^{-24}	5.61×10^{-22}	2.73×10^{-07}
	MBB	1.27×10^{-22}	5.37×10^{-09}	1.33×10^{-23}	2.04×10^{-22}
T	MJV	3.78×10^{-06}	3.72×10^{-08}	2.43×10^{-22}	1.11×10^{-22}
	M1	5.39×10^{-08}	2.78×10^{-07}	5.83×10^{-07}	1.88×10^{-10}
	M3	1.14×10^{-08}	9.67×10^{-09}	3.15×10^{-22}	4.67×10^{-21}
TSS	M3	4.63×10^{-07}	2.43×10^{-08}	9.88×10^{-23}	3.83×10^{-23}
	MJV	3.10×10^{-05}	7.70×10^{-09}	1.31×10^{-21}	3.80×10^{-23}
	MIP	4.89×10^{-24}	5.06×10^{-09}	3.01×10^{-04}	1.66×10^{-22}

Table 8.4: MSE in the ANN test (2014).

MSE Validation by Cycle					
Parameters	Station	Full	Emptying	Dry	Filling
C	C1	1.1593	17.1529	0.2679	5.4940
	C2	0.1555	4.3905	0.4366	0.0889
	MBB	0.3346	0.0070	0.8778	0.1564
T	MJV	0.1789	0.0736	0.0156	0.0828
	M1	0.0010	0.4436	0.4957	0.0272
	M3	0.0966	0.2470	0.2272	0.2318
TSS	M3	0.0444	1.1343	0.0006	1.6069
	MJV	0.0106	1.1881	0.0471	0.2483
	MIP	1.1363	0.0135	11.3284	0.7024

Table 8.5: Relative Error (E_r) in the ANN test (2014).

Relative Error					
Parameters	Station	Full	Emptying	Dry	Filling
C	C1	0.0800	0.5791	0.1392	0.3128
	C2	0.0421	0.6663	0.0751	0.0479
	MBB	0.1106	0.0229	0.2067	0.0886
T	MJV	0.4824	0.0979	0.0793	0.2423
	M1	0.0180	0.1885	0.1416	0.0307
	M3	0.3495	0.1421	0.1479	0.1731
TSS	M3	0.0440	0.4702	0.0303	0.1960
	MJV	0.0229	1.1978	0.0682	0.0647
	MIP	0.0425	0.0174	0.4975	0.0328

The validation results of 2014 for the different sampling stations are reported in Figs. 8.5-8.7, comparing the observed values of laboratory results with the ones estimated by wavelet transformation of the remote sensing images and subsequent analysis by ANN, as proposed in this paper. The X-axis of each figure represents the hydrological cycle; the Y-axis represents the quantitative value of the analyzed parameter in that hydrological cycle. Coherently with the numerical results given in Table 8.4 and Table 8.5, the errors between expected and observed values are quite low. In particular, the bests results are obtained during the dry season cycle 3 (September-October-November) for TSS. This period corresponds to a less cloudy period in the Region. This facilitates the analysis based on satellite images and allows us to obtain more accurate results.

Overall, the obtained errors are quite low and the ANN performance shows good results in the evaluation of physico-chemical parameters that, in turn, allows the identification of possible anthropogenic impacts that are relevant in environmental management and in political decision-making processes.

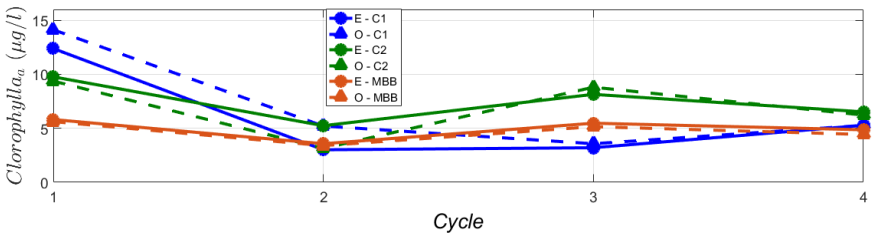


Figure 8.5: C levels in hydroelectric power plant reservoir for sample station: C1 - Caraipu 1, C2 - Caraipu 2, MBB - Breu Branco; E = Estimated; O = Observed-

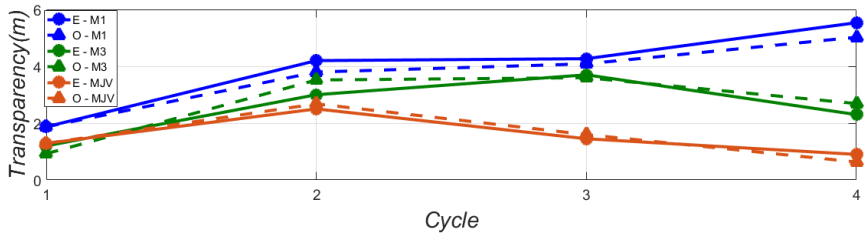


Figure 8.6: T levels in hydroelectric power plant reservoir for sample station: M1 - Upstrem 1, M3 - Upstrem 3 , MJV - Jacunda Velho; E = Estimated; O = Observed.

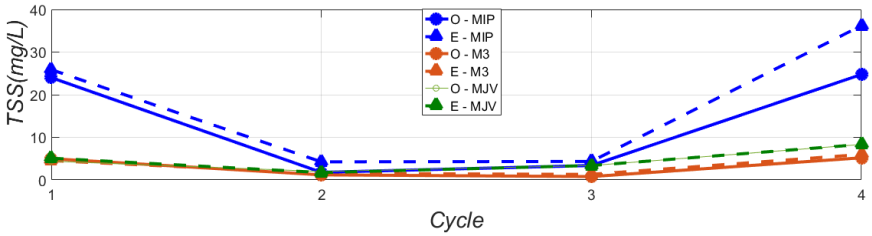


Figure 8.7: TSS levels in hydroelectric power plant reservoir for sample station: MIP - Ipixuna, M3 - Upstrem 3 , MJV - Jacunda Velho; E = Estimated; O = Observed.

8.3.2 Cefni Reservoir

Study Area

The Cefni reservoir (also called "Queen Elizabeth") is located in the center of Anglesey, Wales, UK, and it is managed by Welsh Water and Hamdden Ltd, while the associated fishery is managed by the Cefni Angling Association. The associated area is 860000 squared meters (86 ha), with a length of 2.3 km (Fig. 8.8).

Cefni Reservoir on the Isle of Anglesey was overflown as part of the UK Natural Environment Research Council (NERC). The lake is shallow, with a maximum depth of approximately 4 m and contains beds of submersed, floating-leaved and emergent aquatic macrophyte species. It is also known to support dense growths of toxic blue-green algae during summer.

The reservoir is surrounded by an approximately 100-m wide plantation of coniferous trees with agricultural fields beyond [270]. The collection of water was done periodically in this reservoir.

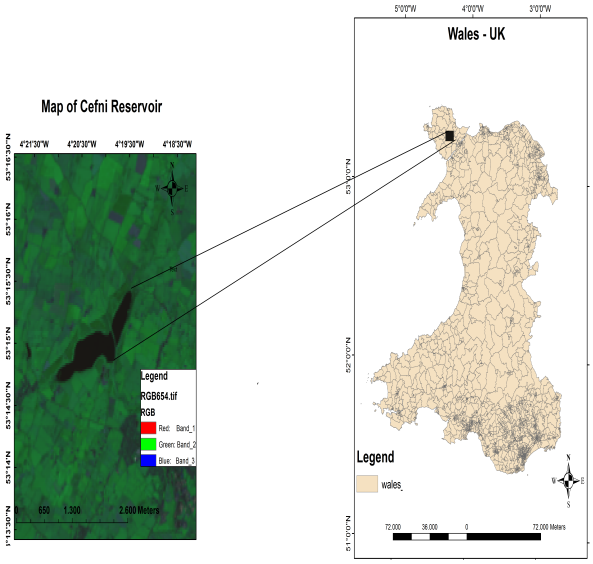


Figure 8.8: Cefni Reservoir, Anglesey, UK.

Water samplings for physico-chemical water quality evaluations are performed periodically at Cefni and the associated data were provided by the Welsh Water and Hamdden Ltd Company. These data were collected from January 2013 to December 2017.

The relationship between Chlorophyll-a Levels, Suspended Solids, Turbidity and spectral response of the reservoir was determined using the physical water samples collected.

These data have been extracted from the samples and analyzed. We compared it to the proposed parameters level extracted from the remote sensing images, analyzed with a WANN method, as described above.

Experimental Results

In the following, the results of the prediction are shown. The table 8.6 shows the WANN training results for Cefni. The mean square errors (MSE) for neural network training are shown to be low for all the Chlorophyll_a, Turbidity and Suspended Solids.

The relative errors were calculated by the following formula in (8.8) and showed in 8.6:

Table 8.6: Mean square errors in the WANN training conducted in the present study

MSE				
Var	***Summer***	***Autumn***	***Winter***	***Winter***
C	9.1×10^{-22}	3.5×10^{-09}	4.1×10^{-22}	5.7×10^{-22}
T	1.1×10^{-06}	8.3×10^{-06}	2.2×10^{-08}	3.4×10^{-09}
SS	2.2×10^{-24}	3.5×10^{-08}	2.9×10^{-05}	4.6×10^{-24}

$$RelativeError(E_r) = \frac{|\chi_e - \chi_o|}{\chi_o} \tag{8.8}$$

where:

χ_e: Estimated Value

χ_o: Observed Value

Table 8.7: Relative Error (Er) in the sampling station, evaluated parameter and season cycle

Relative Error				
Var	***Summer***	***Autumn***	***Winter***	***Winter***
C	0.0001	0.1402	0.1301	0.0412
T	0.2027	0.0881	0.0257	0.0186
SS	0.0201	0.0305	0.2321	0.0514

Validation results for 2017 are shown in the Figures 8.9, 8.10, 8.11. The laboratory results correspond to the *observed values* and the analysis by ANN correspond to the *estimated values* (namely C, T and SS).

The X-axes of figures represent the season cycles (1, 2, 3 and 4), respectively, summer, autumn, winter, and spring. The Y-axis represents the quantitative value of the analyzed parameters.

Tables 8.7 and 8.8 show the errors between expected and observed values. Best results were obtained for Chlorophyll_a in the summer season, cycle 1. This period corresponds to the less cloudy period in the region, being the best estimates by WANN and satellite images from Landsat-8. The results are satisfactory also for the other cycles and the other parameters

The results estimated in the present study when compared with those observed in the laboratory proved extremely close to each other, demonstrating

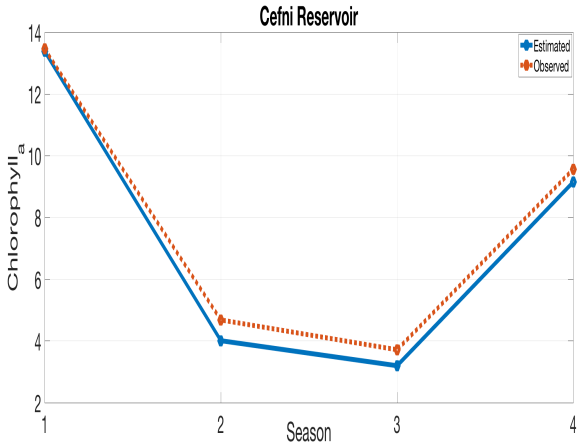


Figure 8.9: Predicting Chlorophyll_a Levels in the Cefni reservoir by WANN and satellite images.

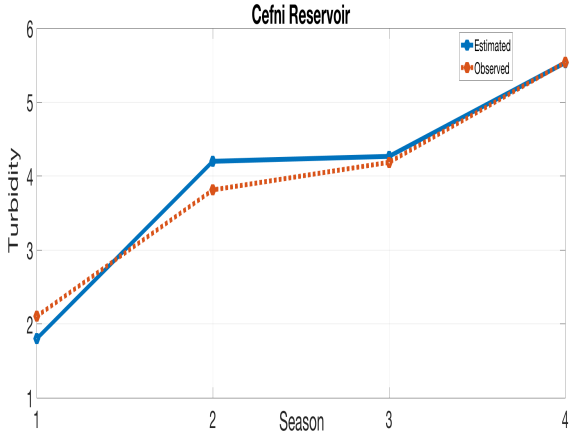


Figure 8.10: Predicting Turbidity in the Cefni reservoir by WANN and satellite images.

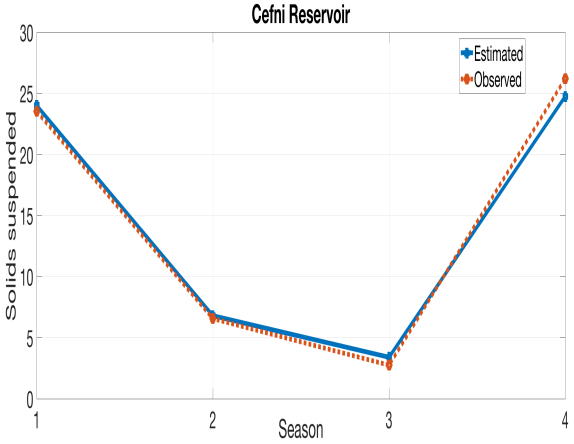


Figure 8.11: Predicting Solids Suspended in the Cefni reservoir by WANN and satellite images.

Table 8.8: Approximation errors of the proposed method for 2017 in the sampling station, evaluated parameter and seasonal cycle.

MSE Validation				
Var	***Summer***	***Autumn***	***Winter***	***Winter***
C	0.0001	0.4522	0.2603	0.1700
T	0.2094	0.1156	0.0115	0.0103
SS	0.2300	0.0501	0.4101	2.1121

the adequate efficiency of the proposed method, WANN showed good results in the evaluation of physico-chemical parameters.

Bibliography

[1] C. Chatfield, *Time-Series Forecasting*. CRC Press, 2000. [Online]. Available: https://books.google.it/books?id=fLlGsTFb21EC

[2] J. G. Box, G., "Time series analysis: forecasting and control," 1970.

[3] D. R. Brockwell, P., *Time Series: Theory and Methods*. Springer, 1991.

[4] P. Newbold, C. Agiakloglou, and J. Miller, "Adventures with arima software," *International Journal of Forecasting*, vol. 10, no. 4, pp. 573 – 581, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0169207094900256

[5] S. L. Ho and M. Xie, "The use of arima models for reliability forecasting and analysis," in *Proceedings of the 23rd International Conference on on Computers and Industrial Engineering*. Elmsford, NY, USA: Pergamon Press, Inc., 1998, pp. 213–216. [Online]. Available: http://dl.acm.org/citation.cfm?id=303088.303181

[6] J. H. Kim, "Forecasting autoregressive time series with bias-corrected parameter estimators," *International Journal of Forecasting*, vol. 19, no. 3, pp. 493 – 502, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207002000626

[7] Y. Q. Fan, J., *Nonlinear Time Series*. Springer, 2003.

[8] T. Teräsvirta, "Specification, estimation, and evaluation of smooth transition autoregressive models," *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 208–218, 1994. [Online]. Available: https://doi.org/10.1080/01621459.1994.10476462

[9] T. Schreiber and A. Schmitz, "Surrogate time series," *Phys. D*, vol. 142, no. 3-4, pp. 346–382, Aug. 2000. [Online]. Available: http://dx.doi.org/10.1016/S0167-2789(00)00043-9

[10] S. Lundbergh, T. Teräsvirta, and D. Van Dijk, "Time-varying smooth transition autoregressive models," *Journal of Business & Economic Statistics*, vol. 21, no. 1, pp. 104–121, 2003.

[11] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[12] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[13] R. F. Engle, "Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation," *Econometrica: Journal of the Econometric Society*, pp. 987–1007, 1982.

[14] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of econometrics*, vol. 31, no. 3, pp. 307–327, 1986.

[15] T. Bollerslev, R. Y. Chou, and K. F. Kroner, "Arch modeling in finance: A review of the theory and empirical evidence," *Journal of econometrics*, vol. 52, no. 1-2, pp. 5–59, 1992.

[16] M. Panella, F. Barcellona, and R. L. D'ecclesia, "Forecasting energy commodity prices using neural networks," *Advances in Decision Sciences*, vol. 2012, 2012.

[17] R. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

[18] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[19] M. G. Frei and I. Osorio, "Intrinsic time-scale decomposition: time–frequency–energy analysis and real-time filtering of non-stationary signals," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2078. The Royal Society, 2007, pp. 321–342.

[20] J. L. Kling and D. A. Bessler, "A comparison of multivariate forecasting procedures for economic time series," *International Journal of Forecasting*, vol. 1, no. 1, pp. 5–24, 1985.

[21] J. P. Lopes, N. Hatziargyriou, J. Mutale, P. Djapic, and N. Jenkins, "Integrating distributed generation into electric power systems: A review of drivers, challenges and opportunities," *Electr. Power Syst. Res.*, vol. 77, no. 9, pp. 1189 – 1203, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378779606001908

[22] V. D. Dio, S. Favuzza, D. L. Cascia, F. Massaro, and G. Zizzo, "Critical assessment of support for the evolution of photovoltaics and feed-in tariff(s) in Italy," *Sustainable Energy Technol. Assess.*, vol. 9, pp. 95 – 104, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2213138814001015

[23] Y. Du, D. D.-C. Lu, G. James, and D. J. Cornforth, "Modeling and analysis of current harmonic distortion from grid connected PV inverters under different operating conditions," *Sol. Energy*, vol. 94, pp. 182 – 194, 2013.

[24] A. Samadi, R. Eriksson, L. S. o. der, B. G. Rawn, and J. C. Boemer, "Coordinated active power-dependent voltage regulation in distribution grids with PV systems," vol. 29, no. 3, pp. 1454–1464, June 2014.

[25] X. Han, S. Liao, X. Ai, W. Yao, and J. Wen, "Determining the minimal power capacity of energy storage to accommodate renewable generation," *Energies*, vol. 10, no. 4, pp. 1996–1073, 2017.

[26] R. H. Inman, H. T. C. Pedro, and C. F. M. Coimbra, "Solar forecasting methods for renewable energy integration," *Prog. Energy Combust. Sci.*, vol. 39, no. 6, pp. 535 – 576, 2013.

[27] E. M. Garrigle and P. Leahy, "Quantifying the value of improved wind energy forecasts in a pool-based electricity market," *Renew. Energy*, vol. 80, pp. 517 – 524, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148115001135

[28] N. Liu, Q. Tang, J. Zhang, W. Fan, and J. Liu, "A hybrid forecasting model with parameter optimization for short-term load forecasting of micro-grids," *Appl. Energy*, vol. 129, pp. 336 – 345, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261914005182

[29] E. G. Kardakos, M. C. Alexiadis, S. I. Vagropoulos, C. K. Simoglou, P. N. Biskas, and A. G. Bakirtzis, "Application of time series and artificial neural network models in short-term forecasting of pv power generation," in *Power Engineering Conference (UPEC)*, Sept 2013, pp. 1–6.

[30] A. Clements, A. Hurn, and Z. Li, "Strategic bidding and rebidding in electricity markets," *Energy Economics*, vol. 59, pp. 24 – 36, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140988316301839

[31] M. Q. Raza, M. Nadarajah, and C. Ekanayake, "On recent advances in PV output power forecast," *Solar Energy*, vol. 136, pp. 125 – 144, 2016.

[32] J. Antonanzas, N. Osorio, R. Escobar, R. Urraca, F. J. M. de Pison, and F. Antonanzas-Torres, "Review of photovoltaic power forecasting," *Solar Energy*, vol. 136, pp. 78 – 111, 2016.

[33] N. Al-Messabi, C. Goh, and Y. Li, "Heuristic grey-box modelling for photovoltaic power systems," *Systems Science & Control Engineering*, vol. 4, no. 1, pp. 235–246, 2016.

[34] D. Coyle, G. Prasad, and T. M. McGinnity, "A time-series prediction approach for feature extraction in a brain-computer interface," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 13, no. 4, pp. 461–467, 2005.

[35] C. Bennett, R. A. Stewart, and J. Lu, "autoregressive with exogenous variables and Neural Network short-term load forecast models for residential low voltage distribution networks," *Energies*, vol. 7, no. 5, pp. 2938–2960, 2014. [Online]. Available: http://www.mdpi.com/1996-1073/7/5/2938

[36] S. Tzafestas and E. Tzafestas, "Computational intelligence techniques for short-term electric load forecasting," *J. Intell. Robotics Syst.*, vol. 31, no. 1-3, pp. 7–68, May 2001.

[37] A. Proietti, M. Panella, F. Leccese, and E. Svezia, "Dust detection and analysis in museum environment based on pattern recognition," *Measurement*, vol. 66, pp. 62 – 72, 2015.

[38] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems 2*, pp. 321–355, 1988.

[39] M. Panella, A. Rizzi, and G. Martinelli, "Refining accuracy of environmental data prediction by MoG neural networks," *Neurocomputing*, vol. 55, no. 3-4, pp. 521–549, 10 2003.

[40] J. Jang, C. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence*. Prentice-Hall, Upper Saddle River, NJ, 1997.

[41] A. Rizzi, M. Panella, F. F. Mascioli, and G. Martinelli, "A recursive algorithm for fuzzy Min-Max networks," in *Proceedings of International Joint Conference on Neural Networks (IJCNN 2000)*, vol. 6. IEEE, July 2000, pp. 541–546.

[42] M. Panella, "A hierarchical procedure for the synthesis of ANFIS networks," *Advances in Fuzzy Systems*, vol. 2012, pp. 1–12, 2012. [Online]. Available: http://www.hindawi.com/journals/afs/2012/491237

[43] M. Panella, A. Rizzi, F. M. F. Mascioli, and G. Martinelli, "Anfis synthesis by hyperplane clustering," in *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference (Cat. No. 01TH8569)*, vol. 1, July 2001, pp. 340–345 vol.1.

[44] M. Panella, L. Liparulo, and A. Proietti, "A higher-order fuzzy neural network for modeling financial time series," in *Proc. of IEEE International Joint Conference on Neural Networks (IJCNN 2014)*, Beijing, China, 2014, pp. 3066–3073.

[45] A. Sharma, R. Podolsky, J. Zhao, and R. A. McIndoe, "A modified hyperplane clustering algorithm allows for efficient and accurate clustering of extremely large datasets," *Bioinformatics*, vol. 25, no. 9, pp. 1152–1157, 2009.

[46] S. Haykin, *Neural Networks, a Comprehensive Foundation, 2nd Edition.* Englewood Cliffs, NJ, USA: Prentice-Hall, 1999.

[47] A. Mellit and S. A. Kalogirou, "Artificial intelligence techniques for photovoltaic applications: A review," *Progress in Energy and Combustion Science*, vol. 34, no. 5, pp. 574 – 632, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360128508000026

[48] M. Panella, "Advances in biological time series prediction by neural networks," *Biomedical Signal Processing and Control*, vol. 6, no. 2, pp. 112–120, 2011.

[49] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[50] M. H. Tong, A. D. Bickett, E. M. Christiansen, and G. W. Cottrell, "Learning grammatical structure with echo state networks," *Neural Networks*, vol. 20, no. 3, pp. 424 – 432, 2007, echo State Networks and Liquid State Machines.

[51] A. H. Moghaddam, M. H. Moghaddam, and M. Esfandyari, "Stock market index prediction using artificial neural network," *J. of Economics, Finance and Administrative Science*, vol. 21, no. 41, pp. 89 – 93, 2016.

[52] F. Triefenbach, A. Jalalvand, K. Demuynck, and J.-P. Martens, "Acoustic modeling with hierarchical reservoirs," *IEEE Trans. on Audio, Speech, and Language Proc.*, vol. 21, no. 11, pp. 2439–2450, Nov. 2013.

[53] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Adv. in Neural Inf. Proc. Syst.*, 2002, pp. 593–600.

[54] S. Scardapane, D. Wang, and M. Panella, "A decentralized training algorithm for echo state networks in distributed big data applications," *Neural Networks*, vol. 78, pp. 65 – 74, 2016.

[55] A. Rosato, R. Altilio, R. Araneo, and M. Panella, "Embedding of time series for the prediction in photovoltaic power plants," in *2016 IEEE 16th International Conference on Environment and Electrical Engineering*, June 2016, pp. 1–4.

[56] H. Abarbanel, *Analysis of Observed Chaotic Data.* Springer, New York, 1996.

[57] M. Panella, A. Rizzi, F. M. F. Mascioli, and G. Martinelli, "ANFIS synthesis by hyperplane clustering," in *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, vol. 1, July 2001, pp. 340–345 vol.1.

[58] C. C. Williams, "Evaluating the penetration of the commodity economy," *Futures*, vol. 35, no. 8, pp. 857 – 868, 2003.

[59] T. O. Awokuse and J. Yang, "The informational role of commodity prices in formulating monetary policy: a reexamination," *Economics Letters*, vol. 79, no. 2, pp. 219 – 224, 2003.

[60] S. Shafiee and E. Topal, "An overview of global gold market and gold price forecasting," *Resources Policy*, vol. 35, no. 3, pp. 178 – 189, 2010.

[61] C. Baumeister and L. Kilian, "Forty years of oil price fluctuations: Why the price of oil may still surprise us," *J. Econ. Perspect.*, vol. 30, no. 1, pp. 139–60, February 2016.

[62] L. Yu, S. Wang, and K. K. Lai, "Forecasting crude oil price with an emd-based neural network ensemble learning paradigm," *Energy Economics*, vol. 30, no. 5, pp. 2623 – 2635, 2008.

[63] J. Bastian, J. Zhu, V. Banunarayanan, and R. Mukerji, "Forecasting energy prices in a competitive market," *IEEE Comput. Appl. Power*, vol. 12, no. 3, pp. 40–45, Jul. 1999.

[64] C. P. Rodriguez and G. J. Anders, "Energy price forecasting in the Ontario competitive power system market," vol. 19, no. 1, pp. 366–374, Feb. 2004.

[65] N. M. Pindoriya, S. N. Singh, and S. K. Singh, "An adaptive wavelet neural network-based energy price forecasting in electricity markets," vol. 23, no. 3, pp. 1423–1432, Aug. 2008.

[66] N. Amjady and M. Hemmati, "Energy price forecasting - problems and proposals for such predictions," *IEEE Power Energy Mag.*, vol. 4, no. 2, pp. 20–29, Mar. 2006.

[67] R. Weron, "Electricity price forecasting: A review of the state-of-the-art with a look into the future," *International Journal of Forecasting*, vol. 30, no. 4, pp. 1030 – 1081, 2014.

[68] S. K. Aggarwal, L. M. Saini, and A. Kumar, "Electricity price forecasting in deregulated markets: A review and evaluation," *Int. J. Electr. Power Energy Syst.*, vol. 31, no. 1, pp. 13 – 22, 2009.

[69] A. G. Bors and I. Pitas, "Median radial basis function neural network," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1351–1364, 1996.

[70] M. Panella, A. Rizzi, and G. Martinelli, "Refining accuracy of environmental data prediction by MoG neural networks," *Neurocomputing*, vol. 55, pp. 521–549, 2003.

[71] A. Rosato, R. Altilio, R. Araneo, and M. Panella, "Takagi-Sugeno fuzzy systems applied to voltage prediction of photovoltaic plants," in *17th Int. Conf. Environ. and Electr. Eng. and 1st Ind. and Com. Power Systems Europe*, June 2017, pp. 1–6.

[72] S. Haykin, *Neural Networks, A Comprehensive Foundation*. Prentice-Hall, Englewood Cliffs, NJ, 1999.

[73] A. Rizzi, M. Buccino, M. Panella, and A. Uncini, "Genre classification of compressed audio data," in *Proceedings of IEEE Workshop on Multimedia Signal Processing (MMSP 2008)*. IEEE, October 2008, pp. 654–659.

[74] N. N. Karnik and J. M. Mendel, "Applications of type-2 fuzzy logic systems to forecasting of time-series," *Information sciences*, vol. 120, no. 1, pp. 89–111, 1999.

[75] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International journal of forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[76] G. G. Szpiro, "Forecasting chaotic time series with genetic algorithms," *Physical Review E*, vol. 55, no. 3, p. 2557, 1997.

[77] F. E. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, no. 4, pp. 309–317, 2001.

[78] A. K. Palit and R. Babuska, "Efficient training algorithm for takagi-sugeno type neuro-fuzzy network," in *IEEE International Conference on Fuzzy Systems*, vol. 3, 2001, pp. 1367–1371.

[79] N. K. Kasabov and Q. Song, "Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 144–154, 2002.

[80] E. D. Lughofer, "Flexfis: A robust incremental learning approach for evolving takagi-sugeno fuzzy models," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1393–1410, 2008.

[81] J. Andreu and P. Angelov, "Forecasting time-series for nn gc1 using evolving takagi-sugeno (ets) fuzzy systems with on-line inputs selection," in *International Conference on Fuzzy Systems*, 2010, pp. 1–5.

[82] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, and J. Lu, "Fuzzy regression transfer learning in takagi-sugeno fuzzy models," *IEEE Transactions on Fuzzy Systems*, vol. PP, no. 99, pp. 1–1, 2016.

[83] S. Safi, A. Zeroual, and M. Hassani, "Prediction of global daily solar radiation using higher order statistics," *Renewable Energy*, vol. 27, no. 4, pp. 647 – 666, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148101001537

[84] S. Kaplanis and E. Kaplani, "Stochastic prediction of hourly global solar radiation for Patra, Greece," *Applied Energy*, vol. 87, no. 12, pp. 3748 – 3758, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261910002205

[85] R. Iqdour and A. Zeroual, "Prediction of daily global solar radiation using fuzzy systems," *Int. J. Sustainable Energy*, vol. 26, no. 1, pp. 19–29, 2007.

[86] C. Voyant, M. Muselli, C. Paoli, and M.-L. Nivet, "Optimization of an artificial neural network dedicated to the multivariate forecasting of daily global radiation," *Energy*, vol. 36, no. 1, pp. 348 – 359, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360544210005955

[87] F. Wang, Z. Mi, S. Su, and H. Zhao, "Short-term solar irradiance forecasting model based on artificial Neural Network using statistical feature parameters," *Energies*, vol. 5, no. 5, pp. 1355–1370, 2012. [Online]. Available: http://www.mdpi.com/1996-1073/5/5/1355

[88] A. Mellit and S. A. Kalogirou, "Artificial intelligence techniques for photovoltaic applications: A review," *Prog. Energy Combust. Sci.*, vol. 34, no. 5, pp. 574 – 632, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360128508000026

[89] F. O. Hocaoğlu, O. N. Gerek, and M. Kurban, "Hourly solar radiation forecasting using optimal coefficient 2-D linear filters and feed-forward neural networks," *Sol. Energy*, vol. 82, pp. 714–726, 2008.

[90] C. Chen, S. Duan, T. Cai, and B. Liu, "Online 24-h solar power forecasting based on weather type classification using artificial neural network," *Solar Energy*, vol. 85, no. 11, pp. 2856 – 2870, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038092X11003008

[91] C. Monteiro, L. A. Fernandez-Jimenez, I. J. Ramirez-Rosado, A. Muñoz-Jimenez, and P. M. Lara-Santillan, "Short-term forecasting models for photovoltaic plants: analytical versus soft-computing techniques," *Mathematical Problems in Engineering*, vol. 2013, no. ID 767284.

[92] E. Izgi, A. ´Oztopal, B. Yerli, M. K. Kaymak, and A. D. Şahin, "Short-mid-term solar power prediction by using artificial neural networks," *Solar Energy*, vol. 86, no. 2, pp. 725 – 733, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038092X11004245

[93] L. A. Fernandez-Jimenez, A. Munoz-Jimenez, A. Falces, M. Mendoza-Villena, E. Garcia-Garrido, P. M. Lara-Santillan, E. Zorzano-Alba, and P. J. Zorzano-Santamaria, "Short-term power forecasting system for photovoltaic plants," *Renewable Energy*, vol. 44, pp. 311 – 317, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148112001516

[94] F. Barbieri, S. Rajakaruna, and A. Ghosh, "Very short-term photovoltaic power forecasting with cloud modeling: A review," *Renewable Sustainable Energy Rev.*, vol. 75, pp. 242 – 263, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S136403211630733X

[95] M. Pierro, F. Bucci, M. D. Felice, E. Maggioni, D. Moser, A. Perotto, F. Spada, and C. Cornaro, "Multi-model Ensemble for day ahead prediction of photovoltaic power generation," *Solar Energy*, vol. 134, pp. 132 – 146, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038092X16300731

[96] Y. Chu, B. Urquhart, S. M. Gohari, H. T. Pedro, J. Kleissl, and C. F. Coimbra, "Short-term reforecasting of power output from a 48 MWe solar PV plant," *Solar Energy*, vol. 112, pp. 68 – 77, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038092X14005611

[97] H. Takeda, "Short-term ensemble forecast for purchased photovoltaic generation," *Solar Energy*, vol. 149, pp. 176 – 187, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038092X17302785

[98] P. Mirowski, S. Chen, T. K. Ho, and C. N. Yu, "Demand forecasting in smart grids," *Bell Labs Technical Journal*, vol. 18, no. 4, pp. 135–158, March 2014.

[99] C.-N. Ko and C.-M. Lee, "Short-term load forecasting using SVR (support vector regression)-based radial basis function neural network with dual extended Kalman filter," *Energy*, vol. 49, pp. 413 – 422, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360544212008766

[100] G. Osório, J. Matias, and J. C. ao, "Short-term wind power forecasting using adaptive neuro-fuzzy inference system combined with evolutionary particle swarm optimization, wavelet transform and mutual information," *Renewable Energy*, vol. 75, pp. 301 – 307, 2015.

[101] J. R. Castro, O. Castillo, M. A. Sanchez, O. Mendoza, A. Rodríguez-Diaz, and P. Melin, "Method for higher order polynomial Sugeno Fuzzy inference systems," *Information Sciences*, vol. 351, pp. 76 – 89, 2016.

[102] J.-S. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[103] J.-S. Jang, C. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence*. Upper Saddle River, NJ, USA: Prentice Hall, 1997.

[104] M. Panella, A. Rizzi, F.M. Frattale Mascioli, and G. Martinelli, "ANFIS synthesis by hyperplane clustering," in *Proc. of IFSA/NAFIPS*, vol. 1, Vancouver, Canada, 2001, pp. 340–345.

[105] M. Panella and A. S. Gallo, "An input-output clustering approach to the synthesis of ANFIS networks," *IEEE Transactions on fuzzy systems*, vol. 13, no. 1, pp. 69–81, 2005.

[106] Z. He and A. Cichocki, "An efficient K-hyperplane clustering algorithm and its application to sparse component analysis," in *Lecture Notes in Computer Science*, D. L. et al., Ed. Berlin Heidelberg, Germany: Springer-Verlag, 2007, vol. 4492, pp. 1032–1041.

[107] G. Seber and C. Wild, *Nonlinear Regression*. NJ: Wiley-Interscience: Hoboken, 2003.

[108] S. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of Intelligent & Fuzzy Systems*, vol. 2, pp. 267–278, 1994.

[109] P. del Río, G. Resch, A. Ortner, L. Liebmann, S. Busch, and C. Panzer, "A techno-economic analysis of EU renewable electricity policy pathways in 2030," *Energy Policy*, vol. 104, pp. 484 – 493, 2017.

[110] R. Araneo, S. Celozzi, and C. Vergine, "Eco-sustainable routing of power lines for the connection of renewable energy plants to the Italian high-voltage grid," *Int. J. Energy Environ. Eng.*, vol. 6, no. 1, pp. 9–19, 2014.

[111] E. Lucchetti, J. Barbier, and R. Araneo, "Assessment of the technical usable potential of the TUM Shaft Hydro Power plant on the Aurino River, Italy," *Renewable Energy*, vol. 60, pp. 648–654, Dec. 2013.

[112] A. Jäger-Waldau, M. Szabó, N. Scarlat, and F. Monforti-Ferrario, "Renewable electricity in Europe," *Renew. Sustain. Energy Rev.*, vol. 15, no. 8, pp. 3703–3716, Oct. 2011.

[113] F. Chen, D. Liu, and X. Xiong, "Research on stochastic optimal operation strategy of active distribution network considering intermittent energy," *Energies*, vol. 10, no. 4, pp. 1996–1073, 2017.

[114] A. Rosato, R. Altilio, R. Araneo, and M. Panella, "Prediction in photovoltaic power by neural network," *Energies*, vol. 10, no. 7, 2017.

[115] C. Cecati, C. Citro, and P. Siano, "Combined operations of renewable energy systems and responsive demand in a smart grid," *IEEE Trans. Sustain. Energy*, vol. 2, no. 4, pp. 468–476, Oct. 2011.

[116] R. Araneo and M. C. Falvo, "Simulation of a ESS in a prosumer powerplant with a PV system and an EV charging station," in *16th Int. Conf. Environ. and Electr. Eng.*, June 2016, pp. 1–5.

[117] G. Singh, P. Baredar, A. Singh, and D. Kurup, "Optimal sizing and location of pv, wind and battery storage for electrification to an island: A case study of kavaratti, lakshadweep," *Journal of Energy Storage*, vol. 12, pp. 78 – 86, 2017.

[118] R. Altilio, A. Rosato, and M. Panella, "A new learning approach for takagi-sugeno fuzzy systems applied to time series prediction," in *Proc. of IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE 2017)*, 2017, pp. 1–6.

[119] C. B. Martinez-Anido, B. Botor, A. R. Florita, C. Draxl, S. Lu, H. F. Hamann, and B.-M. Hodge, "The value of day-ahead solar power forecasting improvement," *Solar Energy*, vol. 129, pp. 192 – 203, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038092X16000736

[120] D. P. Larson, L. Nonnenmacher, and C. F. Coimbra, "Day-ahead forecasting of solar power output from photovoltaic plants in the american southwest," *Renew. Energy*, vol. 91, pp. 11 – 20, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148116300398

[121] R. Rodrigues, V. Mendes, and J. Catalão, "Protection of wind energy systems against the indirect effects of lightning," *Renew. Energy*, vol. 36, no. 11, pp. 2888–2896, Nov. 2011.

[122] A. Tuohy, J. Zack, S. E. Haupt, J. Sharp, M. Ahlstrom, S. Dise, E. Grimit, C. Mohrlen, M. Lange, M. G. Casado, J. Black, M. Marquis, and C. Collier, "Solar forecasting: Methods, challenges, and performance," *IEEE Power Energy Mag*, vol. 13, no. 6, pp. 50–59, Nov. 2015.

[123] L. Cavalcante and R. J. Bessa, "Solar power forecasting with sparse vector autoregression structures," in *2017 IEEE Manchester PowerTech*, June 2017, pp. 1–6.

[124] A. Dolara, S. Leva, and G. Manzolini, "Comparison of different physical models for PV power output prediction," *Solar Energy*, vol. 119, pp. 83 – 99, 2015.

[125] E. Ogliari, A. Dolara, G. Manzolini, and S. Leva, "Physical and hybrid methods comparison for the day ahead pv output power forecast," *Renewable Energy*, vol. 113, pp. 11 – 21, 2017.

[126] C. Wan, J. Zhao, Y. Song, Z. Xu, J. Lin, and Z. Hu, "Photovoltaic and solar power forecasting for smart grid energy management," *CSEE Journal of Power and Energy Systems*, vol. 1, no. 4, pp. 38–46, Dec. 2015.

[127] Y. Li, Y. Su, and L. Shu, "An armax model for forecasting the power output of a grid connected photovoltaic system," *Renewable Energy*, vol. 66, pp. 78 – 89, 2014.

[128] C. Yang, A. A. Thatte, and L. Xie, "Multitime-scale data-driven spatio-temporal forecast of photovoltaic generation," *IEEE Transactions on Sustainable Energy*, vol. 6, no. 1, pp. 104–112, Jan. 2015.

[129] W. Yan, "Toward automatic time-series forecasting using neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 7, pp. 1028–1039, July 2012.

[130] R. Samsudin, A. Shabri, and P. Saad, "A comparison of time series forecasting using support vector machine and artificial neural network model," *Journal of Applied Sciences*, vol. 10, pp. 950–958, 2010.

[131] C. Voyant, G. Notton, S. Kalogirou, M.-L. Nivet, C. Paoli, F. Motte, and A. Fouilloy, "Machine learning methods for solar radiation forecasting: A review," *Renewable Energy*, vol. 105, pp. 569 – 582, 2017.

[132] A. K. Yadav and S. S. Chandel, "Solar radiation prediction using Artificial Neural Network techniques: A review," *Renewable and Sustainable Energy Reviews*, vol. 33, pp. 772 – 781, 2014.

[133] A. Gensler, J. Henze, B. Sick, and N. Raabe, "Deep learning for solar power forecasting – An approach using AutoEncoder and LSTM neural networks," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 002 858–002 865.

[134] C. Persson, P. Bacher, T. Shiga, and H. Madsen, "Multi-site solar power forecasting using gradient boosted regression trees," *Solar Energy*, vol. 150, pp. 423 – 436, 2017.

[135] A. Hamlyn, H. Cheung, L. Wang, C. Yang, and R. Cheung, "Distributed monitoring and centralized forecasting network for dg-connected distribution systems," in *2008 IEEE Power and Energy Society General Meeting*, July 2008, pp. 1–7.

[136] A. Nguyen, M. Velay, J. Schoene, V. Zheglov, B. Kurtz, K. Murray, B. Torre, and J. Kleissl, "High PV penetration impacts on five local distribution networks using high resolution solar resource assessment with sky imager and quasi-steady state distribution system simulations," *Solar Energy*, vol. 132, pp. 221 – 235, 2016.

[137] A. Dolara, F. Grimaccia, S. Leva, M. Mussetta, and E. Ogliari, "A physical hybrid artificial neural network for short term forecasting of PV plant power output," *Energies*, vol. 8, no. 2, pp. 1138–1153, 2015. [Online]. Available: http://www.mdpi.com/1996-1073/8/2/1138

[138] J. R. Trapero, N. Kourentzes, and A. Martin, "Short-term solar irradiation forecasting based on dynamic harmonic regression," *Energy*, vol. 84, pp. 289 – 295, 2015.

[139] M. Ceci, R. Corizzo, F. Fumarola, D. Malerba, and A. Rashkovska, "Predictive modeling of PV energy production: How to set up the learning task for a better prediction?" *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 956–966, Jun. 2017.

[140] R. J. Bessa, A. Trindade, and V. Miranda, "Spatial-temporal solar power forecasting for smart grids," *IEEE Trans. Ind. Informat.*, vol. 11, no. 1, pp. 232–241, Feb. 2015.

[141] S. Scardapane, R. Fierimonte, P. D. Lorenzo, M. Panella, and A. Uncini, "Distributed semi-supervised support vector machines," *Neural Networks*, vol. 80, pp. 43 – 52, 2016.

[142] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems," vol. 60, no. 3, pp. 644–658, Mar. 2015.

[143] T. Morstyn, B. Hredzak, G. D. Demetriades, and V. G. Agelidis, "Unified distributed control for dc microgrid operating modes," vol. 31, no. 1, pp. 802–812, Jan. 2016.

[144] A. Rosato, R. Altilio, R. Araneo, and M. Panella, "A Smart Grid in Ponza Island: Battery Energy Storage Management by Echo State Neural Network," in *Proc. of IEEE International Conference on Environment and Electrical Engineering (IEEE EEEIC 2018)*, 2018, pp. 1–4.

[145] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127 – 149, 2009.

[146] L. Georgopoulos and M. Hasler, "Distributed machine learning in networks by consensus," *Neurocomputing*, vol. 124, pp. 2 – 12, 2014.

[147] S. Scardapane, R. Fierimonte, D. Wang, M. Panella, and A. Uncini, "Distributed music classification using random vector functional-link nets," in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–8.

[148] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[149] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[150] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, Oct 2010.

[151] J. A. Duffie and W. A. Beckman, *Solar engineering of thermal processes*, 3rd ed. New York: Wiley, 2006.

[152] S. Scardapane, R. Fierimonte, P. D. Lorenzo, M. Panella, and A. Uncini, "Distributed semi-supervised support vector machines," *Neural Networks*, vol. 80, pp. 43 – 52, 2016.

[153] R. Fierimonte, R. Altilio, and M. Panella, "Distributed on-line learning for random-weight fuzzy neural networks," in *Proc. of IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE 2017)*, 2017, pp. 1–6.

[154] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Zomaya, S. Foufou, and A. Bouras, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE transactions on emerging topics in computing*, vol. 2, no. 3, pp. 267–279, 2014.

[155] R. Hanisch, "Distributed data systems and services for astronomy and the space sciences," in *Astronomical Data Analysis Software and Systems IX*, vol. 216, 2000, p. 201.

[156] C. Lynch, "Big data: How do your data grow?" *Nature*, vol. 455, no. 7209, pp. 28–29, 2008.

[157] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*. ACM, 2004, pp. 20–27.

[158] D. Saha and A. Mukherjee, "Pervasive computing: a paradigm for the 21st century," *Computer*, vol. 36, no. 3, pp. 25–31, 2003.

[159] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[160] R. D. Nowak, "Distributed em algorithms for density estimation and clustering in sensor networks," *Signal Processing, IEEE Trans. on*, vol. 51, no. 8, pp. 2245–2253, 2003.

[161] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed expectation-maximization algorithm for density estimation and classification using wireless sensor networks," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conf. on.* IEEE, 2008, pp. 1989–1992.

[162] D. Gu, "Distributed em algorithm for gaussian mixtures in sensor networks," *Neural Networks, IEEE Trans. on*, vol. 19, no. 7, pp. 1154–1166, 2008.

[163] S. Kantabutra and A. L. Couch, "Parallel k-means clustering algorithm on nows," *NECTEC Technical journal*, vol. 1, no. 6, pp. 243–247, 2000.

[164] X. Pan, J. E. Gonzalez, S. Jegelka, T. Broderick, and M. I. Jordan, "Optimistic concurrency control for distributed unsupervised learning," in *Advances in Neural Information Processing Systems*, 2013, pp. 1403–1411.

[165] Y. Liang, M.-F. Balcan, and V. Kanchanapally, "Distributed pca and k-means clustering," in *The Big Learning Workshop at NIPS*, 2013.

[166] E. Januzaj, H.-P. Kriegel, and M. Pfeifle, "Towards effective and efficient distributed clustering," in *Workshop on Clustering Large Data Sets (ICDM2003)*, 2003.

[167] X. Xu, J. Jäger, and H.-P. Kriegel, "A fast parallel clustering algorithm for large spatial databases," in *High Performance Data Mining.* Springer, 2002, pp. 263–290.

[168] S. Rahimi, M. Zargham, A. Thakre, and D. Chhillar, "A parallel fuzzy c-mean algorithm for image segmentation," in *Fuzzy Information, 2004. Processing NAFIPS'04. IEEE Annual Meeting of the*, vol. 1. IEEE, 2004, pp. 234–237.

[169] L. Vendramin, R. J. G. B. Campello, L. F. Coletta, and E. R. Hruschka, "Distributed fuzzy clustering with automatic detection of the number of clusters," in *International Symposium on Distributed Computing and Artificial Intelligence.* Springer, 2011, pp. 133–140.

[170] J. Zhou, C. Philip Chen, L. Chen, and H.-X. Li, "A collaborative fuzzy clustering algorithm in distributed network environments," *Fuzzy Systems, IEEE Trans. on*, vol. 22, no. 6, pp. 1443–1456, 2014.

[171] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *The Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2003.

[172] S. Zhang, H.-S. Wong, and Y. Shen, "Generalized adjusted rand indices for cluster ensembles," *Pattern Recognition*, vol. 45, no. 6, pp. 2214–2226, 2012.

[173] P. Hore, L. O. Hall, and D. B. Goldgof, "A scalable framework for cluster ensembles," *Pattern Recognition*, vol. 42, no. 5, pp. 676–688, 2009.

[174] D. Katselis, C. L. Beck, and M. van der Schaar, "Ensemble online clustering through decentralized observations," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 910–915.

[175] C. Wemmert, P. Gançarski, and J. J. Korczak, "A collaborative approach to combine multiple learning methods," *International Journal on Artificial Intelligence Tools*, vol. 9, no. 01, pp. 59–78, 2000.

[176] G. Forestier, P. Gancarski, and C. Wemmert, "Collaborative clustering with background knowledge," *Data & Knowledge Engineering*, vol. 69, no. 2, pp. 211–228, 2010.

[177] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[178] J. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, 1973.

[179] F. F. Mascioli, G. Risi, A. Rizzi, and G. Martinelli, "A nonexclusive classification system based on co-operative fuzzy clustering," in *Proc. EUSIPCO'98*, 1998, pp. 395–398.

[180] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[181] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, no. 2, pp. 107–145, 2001.

[182] T. Kvålseth, "A coefficient of agreement for nominal scales: An asymmetric version of kappa," *Educational and Psychological Measurement*, vol. 51, no. 1, pp. 95–101, 1991, cited By 7. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0011838503&partnerID=40&md5=bb15325339ed058b510853f6e08f85ae

[183] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEEE transactions on knowledge and data engineering*, vol. 26, no. 1, pp. 97–107, 2014.

[184] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.

[185] J. B. Predd, S. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56–69, 2006.

[186] M. Pipattanasomporn, H. Feroze, and S. Rahman, "Multi-agent systems in a distributed smart grid: Design and implementation," in *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES*. IEEE, 2009, pp. 1–8.

[187] S. Scardapane, R. Altilio, V. Ciccarelli, and M. Panella, "Privacy-preserving data mining for distributed medical scenarios," in *Advances in Neural Networks*. Springer, 2017.

[188] G. Magoulas, M. Vrahatis, T. Grapsa, and G. Androulakis, "A training method for discrete multilayer neural networks," in *Mathematics of Neural Networks*. Springer, 1997, pp. 250–254.

[189] E. M. Corwin, A. M. Logar, and W. J. Oldham, "An iterative method for training multilayer networks with threshold functions," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 507–508, 1994.

[190] A. Khan and E. Hines, "Integer-weight neural nets," *Electronics Letters*, vol. 30, no. 15, pp. 1237–1238, 1994.

[191] V. Plagianakos and M. Vrahatis, "Neural network training with constrained integer weights," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3. IEEE, 1999, pp. 2007–2013.

[192] V. P. Plagianakos and M. N. Vrahatis, "Training neural networks with threshold activation functions and constrained integer weights," in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 5. IEEE, 2000, pp. 161–166.

[193] S. Anwar, K. Hwang, and W. Sung, "Fixed point optimization of deep convolutional neural networks for object recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1131–1135.

[194] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[195] W. F. Schmidt, M. A. Kraaijveld, and R. P. Duin, "Feedforward neural networks with random weights," in *Pattern Recognition, 1992. Vol.*

II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on. IEEE, 1992, pp. 1–4.

[196] S. Scardapane, M. Panella, D. Comminiello, and A. Uncini, "Learning from distributed data sources using random vector functional-link networks," *Procedia Computer Science*, vol. 53, pp. 468–477, 2015.

[197] R. Fierimonte, R. Altilio, and M. Panella, "Distributed on-line learning for random-weight fuzzy neural networks," *Proc. of IEEE International Conference on Fuzzy Systems*, 2017.

[198] S. Scardapane, R. Fierimonte, D. Wang, M. Panella, and A. Uncini, "Distributed music classification using random vector functional-link nets," in *Neural Networks (IJCNN), 2015 International Joint Conference on.* IEEE, 2015, pp. 1–8.

[199] R. Altilio, A. Rosato, and M. Panella, "A nonuniform quantizer for hardware implementation of neural networks," in *Circuit Theory and Design (ECCTD), 2017 European Conference on*, 2017.

[200] J. M. Martínez-Villena, A. Rosado-Muñoz, and E. Soria-Olivas, "Hardware implementation methods in random vector functional-link networks," *Applied Intelligence*, vol. 41, no. 1, pp. 184–195, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10489-013-0501-1

[201] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1320–1329, 1995.

[202] P. J. Teunissen, "An optimality property of the integer least-squares estimator," *Journal of Geodesy*, vol. 73, no. 11, pp. 587–593, 1999.

[203] J. Goldberger and A. Leshem, "Iterative tomographic solution of integer least squares problems with applications to mimo detection," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 8, pp. 1486–1496, Dec 2011.

[204] P. van Emde Boas, *Another NP-complete partition problem and the complexity of computing short vectors in a lattice.* Universiteit van Amsterdam. Mathematisch Instituut, 1981.

[205] D. Micciancio, "The hardness of the closest vector problem with preprocessing," *IEEE Transactions on Information Theory*, vol. 47, no. 3, pp. 1212–1215, Mar 2001.

[206] X.-W. Chang and Q. Han, "Solving box-constrained integer least squares problems," *IEEE Trans. Wireless Communications*, vol. 7, pp. 277–287, 2008.

[207] J. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. Bradford Books, 1992.

[208] T. F. Brooks, D. S. Pope, and M. A. Marcolini. (1989) Airfoil self-noise and prediction. [Online]. Available: https://ntrs.nasa.gov/search.jsp?R=19890016302

[209] I.-C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks," *Cement and Concrete research*, vol. 28, no. 12, pp. 1797–1808, 1998.

[210] A. Tsanas and A. Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools," *Energy and Buildings*, vol. 49, pp. 560–567, 2012.

[211] O. Akbilgic, H. Bozdogan, and M. E. Balaban, "A novel hybrid rbf neural networks model as a forecaster," *Statistics and Computing*, vol. 24, no. 3, pp. 365–375, 2014.

[212] F. A. Khan and A. A. Ansari, "Eutrophication: an ecological vision," *The botanical review*, vol. 71, no. 4, pp. 449–482, 2005.

[213] D. G. F. Cunha, M. do Carmo Calijuri, and M. C. Lamparelli, "A trophic state index for tropical/subtropical reservoirs (tsi tsr)," *Ecological Engineering*, vol. 60, pp. 126–134, 2013.

[214] E. Tebbs, J. Remedios, and D. Harper, "Remote sensing of chlorophyll-a as a measure of cyanobacterial biomass in lake bogoria, a hypertrophic, saline–alkaline, flamingo lake, using landsat etm+," *Remote Sensing of Environment*, vol. 135, pp. 92–106, 2013.

[215] C. Dona, J. M. Sanchez, V. Caselles, J. A. Domínguez, and A. Camacho, "Empirical relationships for monitoring water quality of lakes and reservoirs through multispectral images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 5, pp. 1632–1641, 2014.

[216] C. F. Cerco and M. R. Noel, "Impact of reservoir sediment scour on water quality in a downstream estuary," *Journal of environmental quality*, vol. 45, no. 3, pp. 894–905, 2016.

[217] J. Tundisi, J. Goldemberg, T. Matsumura-Tundisi, and A. Saraiva, "How many more dams in the amazon?" *Energy Policy*, vol. 74, pp. 703–708, 2014.

[218] J. Eynard, S. Grieu, and M. Polit, "Wavelet-based multi-resolution analysis and artificial neural networks for forecasting temperature and thermal power consumption," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 3, pp. 501–516, 2011.

[219] L.-x. GUO and C.-h. DENG, "Prediction model for dissolved oxygen in fish pond based on fuzzy neural network [j]," *Journal of Fisheries of China*, vol. 30, no. 2, pp. 225–229, 2006.

[220] D. Ö. Faruk, "A hybrid neural network and arima model for water quality time series prediction," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp. 586–594, 2010.

[221] J. Abaurrea, J. Asín, A. C. Cebrián, and M. A. García-Vera, "Trend analysis of water quality series based on regression models with correlated errors," *Journal of Hydrology*, vol. 400, no. 3, pp. 341–352, 2011.

[222] X. Wang, X.-p. Zhao, Z. Liu, and S. Dong, "Research on lake eutrophication forecasting methods based on grey theory," *Computer Simulation*, vol. 1, pp. 17–19, 2011.

[223] S. Mahapatra, S. K. Nanda, and B. Panigrahy, "A cascaded fuzzy inference system for indian river water quality prediction," *Advances in Engineering Software*, vol. 42, no. 10, pp. 787–796, 2011.

[224] M. Awad, "Sea water chlorophyll-a estimation using hyperspectral images and supervised artificial neural network," *Ecological informatics*, vol. 24, pp. 60–68, 2014.

[225] L. Xu and S. Liu, "Study of short-term water quality prediction model based on wavelet neural network," *Mathematical and Computer Modelling*, vol. 58, no. 3, pp. 807–813, 2013.

[226] A. Najah, A. El-Shafie, O. A. Karim, and A. H. El-Shafie, "Performance of anfis versus mlp-nn dissolved oxygen prediction models in water quality monitoring," *Environmental Science and Pollution Research*, vol. 21, no. 3, pp. 1658–1670, 2014.

[227] F. Soltani, R. Kerachian, and E. Shirangi, "Developing operating rules for reservoirs considering the water quality issues: Application of anfis-based surrogate models," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6639–6645, 2010.

[228] A. A. Najah, A. El-Shafie, O. A. Karim, and O. Jaafar, "Water quality prediction model utilizing integrated wavelet-anfis model with cross-validation," *Neural Computing and Applications*, vol. 21, no. 5, pp. 833–841, 2012.

[229] M. Bonansea, M. C. Rodriguez, L. Pinotti, and S. Ferrero, "Using multi-temporal landsat imagery and linear mixed models for assessing water quality parameters in río tercero reservoir (argentina)," *Remote Sensing of Environment*, vol. 158, pp. 28–41, 2015.

[230] M. Bonansea and R. Fernandez, "Remote sensing of suspended solids concentration in a reservoir with frequent wildland fires on its watershed," *Water Science and Technology*, vol. 67, no. 1, pp. 217–223, 2012.

[231] N. Karakaya, F. Evrendilek, G. Aslan, K. Gungor, and D. Karakas, "Monitoring of lake water quality along with trophic gradient using landsat data," 2012.

[232] B. R. Neto, R. Hauser-Davis, T. Lobato, A. Saraiva, I. Brandão, T. Oliveira, and A. Silveira, "Estimating physicochemical parameters and metal concentrations in hydroelectric reservoirs by virtual sensors: A case study in the amazon region," *Computer Science and Engineering*, vol. 4, no. 2, pp. 43–53, 2014.

[233] C. J. Bradshaw and B. W. Brook, "The conservation biologist's toolbox–principles for the design and analysis of conservation studies," *Conservation Biology for All*, pp. 313–334, 2010.

[234] C. Grueber, S. Nakagawa, R. Laws, and I. Jamieson, "Multimodel inference in ecology and evolution: challenges and solutions," *Journal of evolutionary biology*, vol. 24, no. 4, pp. 699–711, 2011.

[235] X. Giam and J. D. Olden, "A new r 2-based metric to shed greater insight on variable importance in artificial neural networks," *Ecological Modelling*, vol. 313, pp. 307–313, 2015.

[236] E. Alcântara, N. Bernardo, F. Watanabe, T. Rodrigues, L. Rotta, A. Carmo, M. Shimabukuro, S. Gonçalves, and N. Imai, "Estimating the cdom absorption coefficient in tropical inland waters using oli/landsat-8 images," *Remote Sensing Letters*, vol. 7, no. 7, pp. 661–670, 2016.

[237] F. S. Y. Watanabe, E. Alcântara, T. W. P. Rodrigues, N. N. Imai, C. C. F. Barbosa, and L. H. d. S. Rotta, "Estimation of chlorophyll-a concentration and the trophic state of the barra bonita hydroelectric reservoir using oli/landsat-8 images," *International journal of environmental research and public health*, vol. 12, no. 9, pp. 10 391–10 417, 2015.

[238] S. Martins, N. Bernardo, I. Ogashawara, and E. Alcantara, "Support vector machine algorithm optimal parameterization for change detection mapping in funil hydroelectric reservoir (rio de janeiro state, brazil)," *Modeling Earth Systems and Environment*, vol. 2, no. 3, p. 138, 2016.

[239] E. Kaneko, H. Aoki, and M. Tsukada, "Image-based path radiance estimation guided by physical model," in *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*. IEEE, 2016, pp. 6942–6945.

[240] T. Blakey, A. Melesse, M. C. Sukop, G. Tachiev, D. Whitman, and F. Miralles-Wilhelm, "Developing benthic class specific, chlorophyll-a retrieving algorithms for optically-shallow water using seawifs," *Sensors*, vol. 16, no. 10, p. 1749, 2016.

[241] G. Lantzanakis, Z. Mitraka, and N. Chrysoulakis, "Comparison of physically and image based atmospheric correction methods for sentinel-2 satellite imagery," in *Perspectives on Atmospheric Sciences*. Springer, 2017, pp. 255–261.

[242] L. Shi, Z. Mao, P. Chen, S. Han, F. Gong, and Q. Zhu, "Comparison and evaluation of atmospheric correction algorithms of quac, dos and flaash for hico hyperspectral imagery," in *SPIE Remote Sensing*. International Society for Optics and Photonics, 2016, pp. 999 917–999 917.

[243] M. Nazeer and J. E. Nichol, "Selection of atmospheric correction method and estimation of chlorophyll-a (chl-a) in coastal waters of hong kong," in *Earth Observation and Remote Sensing Applications (EORSA), 2014 3rd International Workshop on*. IEEE, 2014, pp. 374–378.

[244] N. Bernardo, F. Watanabe, T. Rodrigues, and E. Alcântara, "Atmospheric correction issues for retrieving total suspended matter concentrations in inland waters using oli/landsat-8 image," *Advances in Space Research*, 2017.

[245] L. Congedo, "Semi-automatic classification plugin for qgis," 2013.

[246] L. Congedo, L. Sallustio, M. Munafò, M. Ottaviano, D. Tonti, and M. Marchetti, "Copernicus high-resolution layers for land cover classification in italy," *Journal of Maps*, vol. 12, no. 5, pp. 1195–1205, 2016.

[247] G. Chander, B. L. Markham, and D. L. Helder, "Summary of current radiometric calibration coefficients for landsat mss, tm, etm+, and eo-1 ali sensors," *Remote sensing of environment*, vol. 113, no. 5, pp. 893–903, 2009.

[248] A. C. Teodoro, F. Veloso-Gomes, and H. Goncalves, "Retrieving tsm concentration from multispectral satellite data by multiple regression and artificial neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 5, pp. 1342–1350, 2007.

[249] R. Altilio, L. Liparulo, A. Proietti, M. Paoloni, and M. Panella, "A genetic algorithm for feature selection in gait analysis," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, July 2016, pp. 4584–4591.

[250] S. Gilmore, A. Saleem, and A. Dewan, "Effectiveness of dos (dark-object subtraction) method and water index techniques to map wetlands in a rapidly urbanising megacity with landsat 8 data," in *Research@ Locate'15*. http: http://SunSITE. Informatik. RWTH-Aachen. DE/Publications/CEUR-WS/., 2015, pp. 100–108.

[251] M. A. Spanner, L. L. Pierce, D. L. Peterson, and S. W. Running, "Remote sensing of temperate coniferous forest leaf area index the influence of canopy closure, understory vegetation and background reflectance," *TitleREMOTE SENSING*, vol. 11, no. 1, pp. 95–111, 1990.

[252] M. E. Jakubauskas, "Thematic mapper characterization of lodgepole pine seral stages in yellowstone national park, usa," *Remote sensing of environment*, vol. 56, no. 2, pp. 118–132, 1996.

[253] P. S. Chavez Jr, "Radiometric calibration of landsat thematic mapper multispectral images," *Photogrammetric Engineering and Remote Sensing*, vol. 55, no. 9, pp. 1285–1294, 1989.

[254] P. Brezonik, K. D. Menken, and M. Bauer, "Landsat-based remote sensing of lake water quality characteristics, including chlorophyll and colored dissolved organic matter (cdom)," *Lake and Reservoir Management*, vol. 21, no. 4, pp. 373–382, 2005.

[255] J. J. Makwana and M. K. Tiwari, "Intermittent streamflow forecasting and extreme event modelling using wavelet based artificial neural networks," *Water resources management*, vol. 28, no. 13, pp. 4857–4873, 2014.

[256] R. Sahay and V. Sehgal, "Wavelet regression models for predicting flood stages in rivers: a case study in eastern india," *Journal of Flood Risk Management*, vol. 6, no. 2, pp. 146–155, 2013.

[257] R. R. Sahay and A. Srivastava, "Predicting monsoon floods in rivers embedding wavelet transform, genetic algorithm and neural network," *Water resources management*, vol. 28, no. 2, pp. 301–317, 2014.

[258] B. Tahani, B. Boumedyen, A. M. Naceur, O. P. Fogh, and A. Christophe, "Multiple fault detection based on wavelet denoising: Application on wind turbine system," in *Control and Automation (MED), 2017 25th Mediterranean Conference on*. IEEE, 2017, pp. 419–423.

[259] R. Barzegar, J. Adamowski, and A. A. Moghaddam, "Application of wavelet-artificial intelligence hybrid models for water quality prediction: a case study in aji-chay river, iran," *Stochastic environmental research and risk assessment*, vol. 30, no. 7, pp. 1797–1819, 2016.

[260] M. Ravansalar, T. Rajaee, and O. Kisi, "Wavelet-linear genetic programming: A new approach for modeling monthly streamflow," *Journal of Hydrology*, vol. 549, pp. 461–475, 2017.

[261] K. H. Talukder and K. Harada, "Haar wavelet based approach for image compression and quality assessment of compressed image," *arXiv preprint arXiv:1010.4084*, 2010.

[262] D. Gupta and S. Choubey, "Discrete wavelet transform for image processing," *International Journal of Emerging Technology and Advanced Engineering*, vol. 4, no. 3, pp. 598–602, 2015.

[263] D. Tomassi, D. Milone, and J. D. Nelson, "Wavelet shrinkage using adaptive structured sparsity constraints," *Signal Processing*, vol. 106, pp. 73–87, 2015.

[264] P. S. Addison, *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance*. CRC press, 2017.

[265] M. Shoaib, A. Y. Shamseldin, B. W. Melville, and M. M. Khan, "Hybrid wavelet neural network approach," in *Artificial Neural Network Modelling*. Springer, 2016, pp. 127–143.

[266] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[267] A. X. Patel, P. Kundu, M. Rubinov, P. S. Jones, P. E. Vértes, K. D. Ersche, J. Suckling, and E. T. Bullmore, "A wavelet method for modeling and despiking motion artifacts from resting-state fmri time series," *Neuroimage*, vol. 95, pp. 287–304, 2014.

[268] P. da Silva Holanda, C. J. C. Blanco, A. L. A. Mesquita, A. C. P. B. Junior, N. M. de Figueiredo, E. N. Macêdo, and Y. Secretan, "Assessment of hydrokinetic energy resources downstream of hydropower plants," *Renewable Energy*, vol. 101, pp. 1203–1214, 2017.

[269] J. G. Tundisi, M. A. Santos, and C. F. S. Menezes, "Tucuruí reservoir and hydroelectric power plant," *Sharing Experiences and Lessons Learned in Lake Basin Mangement, Burlington, Vermont. Management Experiences and Lessons Learned Brief*, vol. 1, pp. 1–20, 2003.

[270] T. Malthus and D. George, "Airborne remote sensing of macrophytes in cefni reservoir, anglesey, uk," *Aquatic Botany*, vol. 58, no. 3-4, pp. 317–332, 1997.