UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Hybrid artificial intelligence algorithms for short-term load and price forecasting in competitive electric markets

Pedro Miguel Rocha Bento

Dissertação para obtenção do Grau de Mestre
Engenharia Eletrotécnica e de Computadores
(2º ciclo de estudos)

Orientador: Prof. Dr. Sílvio José Pinto Simões Mariano

Covilhã, Junho 2017

Aos meus pais
e
avós

Agradecimentos

Em primeiro lugar gostaria de agradecer ao Prof. Doutor Sílvio Mariano, pela dedicação, acompanhamento e auxílio prestados ao longo da dissertação, num percurso e interesse iniciado ainda nas cadeiras de Sistemas de Energia Elétrica e Métodos de Apoio à decisão.

Também uma palavra especial para o Mestre José Pombo por toda a sua disponibilidade e preciosa contribuição na elaboração desta dissertação.

Também não posso deixar de mencionar os meus pais por todo o apoio, essencial para a minha formação.

Um agradecimento aos Prof.s Doutores Rui Almeida e Rui Pacheco pelo auxílio prestado na compreensão de algumas matérias e nos recursos que disponibilizaram para a execução de simulações numéricas.

Por último um agradecimento a todos aqueles que, de uma forma ou de outra, contribuíram para a elaboração desta dissertação.

Resumo

O processo de liberalização e desregulação dos mercados de energia elétrica, obrigou os diversos participantes a acomodar uma série de desafios, entre os quais: a acumulação considerável de nova capacidade de geração proveniente de origem renovável (fundamentalmente energia eólica), a imprevisibilidade associada a estas novas formas de geração e novos padrões de consumo. Resultando num aumento da volatilidade associada aos preços de energia elétrica (como é exemplo o mercado ibérico).

Dado o quadro competitivo em que os agentes de mercado operam, a existência de técnicas computacionais de previsão eficientes, constituí um fator diferenciador. É com base nestas previsões que se definem estratégias de licitação e se efetua um planeamento da operação eficaz dos sistemas de geração que, em conjunto com um melhor aproveitamento da capacidade de transmissão instalada, permite maximizar os lucros, realizando ao mesmo tempo um melhor aproveitamento dos recursos energéticos.

Esta dissertação apresenta um novo método híbrido para a previsão da carga e dos preços da energia elétrica, para um horizonte temporal a 24 horas. O método baseia-se num esquema de otimização que reúne os esforços de diferentes técnicas, nomeadamente redes neuronais artificiais, diversos algoritmos de otimização e da transformada de *wavelet*. A validação do método foi feita em diferentes casos de estudo reais. A posterior comparação com resultados já publicados em revistas de referência, revelou um excelente desempenho do método hibrido proposto.

Palavras-chave

Algoritmos meta heurísticos, inteligência artificial, mercados elétricos, previsão da carga, previsão de preços da energia elétrica., programação evolucionária, redes neuronais artificiais, sistemas de energia elétrica, transformada de *wavelet*.

Resumo Alargado

Os sistemas de energia elétrica constituem "macro" entidades que agregam os mais diversos agentes, o espetro de operação destas entidades vai desde a produção, transmissão e distribuição e comercialização de energia elétrica. Particularizando, os mercados de energia elétrica, cuja função é a de explorar a eletricidade como bem económico, sofreram transformações abrangentes (como é exemplo o mercado ibérico), guiadas pelos conceitos de liberalização e desregulação, contrastando com estruturas até então, monopolistas e verticalizadas, na persecução de um melhor sistema.

Com esta mudança de paradigma, adveio um conjunto de normas regulatórias que introduziram novos desafios, entre os quais: mecanismos concorrenciais, gestão do *mix* de produção (acomodar nova capacidade de geração proveniente de origem renovável), imprevisibilidade associada a estas novas formas de geração e novos padrões de consumo (carga).

Neste contexto, os agentes de mercado assentam a sua tomada de decisão em função de modelos de previsão, em particular, a previsão da carga e dos preços de energia elétrica, sendo por isso um fator diferenciador. Neste quadro competitivo, há um forte interesse em técnicas computacionais de previsão que se revelem eficazes. No entanto as características inerentes às séries temporais da carga e dos preços (dada a sua volatilidade), resultam num acréscimo de complexidade, sendo por isso uma área de investigação muito ativa.

A literatura recente apresenta e valida a utilização de técnicas de inteligência artificial, como ferramentas de previsão eficazes, em concreto, métodos que combinem esforços de diferentes técnicas (métodos híbridos) revelam um desempenho superior. Como tal, é proposto foi um método híbrido, para a previsão da carga e dos preços da energia elétrica, para um horizonte temporal a 24 horas. O método baseia-se num esquema de otimização que reúne os esforços de diferentes técnicas, nomeadamente a transformada de *wavelet*, redes neuronais artificiais e diversos algoritmos de otimização, nomeadamente *Particle Swarm Optimization* e *Bat Algorithm*. No que diz respeito à utilização das redes neuronais artificiais (sua topologia) e na ausência de um paradigma ótimo, ou quase-ótimo, o método proposto determina uma topologia ótima, recorrendo a algoritmos de otimização.

Com o propósito de validar o método desenvolvido, o mesmo foi testado no mercado Português e no mercado regional de *New England*, no caso da previsão da carga, já na previsão dos preços o mesmo foi testado no mercado Espanhol e no mercado regional PJM. A consequente análise e comparação com resultados já publicados, em diferentes revistas de referência, permitiu validar o método hibrido desenvolvido.

Abstract

The liberalization and deregulation of electric markets forced the various participants to accommodate several challenges, including: a considerable accumulation of new generation capacity from renewable sources (fundamentally wind energy), the unpredictability associated with these new forms of generation and new consumption patterns, contributing to further electricity prices volatility (e.g. the Iberian market).

Given the competitive framework in which market participants operate, the existence of efficient computational forecasting techniques is a distinctive factor. Based on these forecasts a suitable bidding strategy and an effective generation systems operation planning is achieved, together with an improved installed transmission capacity exploitation, results in maximized profits, all this contributing to a better energy resources utilization.

This dissertation presents a new hybrid method for load and electricity prices forecasting, for one day ahead time horizon. The optimization scheme presented in this method, combines the efforts from different techniques, notably artificial neural networks, several optimization algorithms and wavelet transform. The method's validation was made using different real case studies. The subsequent comparison (accuracy wise) with published results, in reference journals, validated the proposed hybrid method suitability.

Keywords

Artificial intelligence, artificial neural networks, electric energy prices forecasting, electric markets, electric power systems, evolutionary programming, load forecasting, metaheuristic algorithms, wavelet transform.

Contents

List of Figures

List of Tables

List of Acronyms

EES	Electric Energy Systems
GDP	Gross Domestic Product
EIA	U.S Energy Information Administration
MIBEL	Mercado Ibérico de Energia Elétrica
STLF	Short-Term Load Forecast
ARMA	Autoregressive Moving Average
ARIMA	Autoregressive Integrated Moving Average
ARCH	Autoregressive Conditional Heteroscedasticity
AI	Artificial Intelligence
ANNs	Artificial Neural Networks
SVMs	Support Vector Machines
FIS	Fuzzy Inference Systems
WT	Wavelet Transform
ELM	Extreme Learning Machine
SARIMA	Seasonal ARIMA
ANFIS	Adaptive Neuro Fuzzy Inference System
MAPE	Mean Absolute Percentage Error
STPF	Short-Term Price Forecast
GARCH	Generalized Autoregressive Conditional Heteroskedastic
CNNs	Cascaded Neural Networks
GARCH	Short Time Fourier Transform
CNNs	Continuous Wavelet Transform
STFT	Short Time Fourier Transform
CWT	Continuous Wavelet Transform
DWT	Discrete Wavelet Transform
MRA	Multiresolution Analysis
Adaline	Adaptive linear neuron
Madaline	Multilayered Adaline
MSE	Mean Squared Error
MRE	Mean Relative Estimation Error
MAE	Mean Absolute Error
SSR	Sum of Squared Residuals
RMSE	Root Mean Square Error
LMS	Least Mean Squared
CG	Conjugate Gradient
BP	Back Propagation

LM	Levenberg-Marquardt
SCG	Scaled Conjugate Gradient
BFGS	Broyden–Fletcher–Goldfarb–Shanno
FFNN	Feed Forward Neural Network
MLP	Multi-Layer Perceptron
CN	Competitive Network
RN	Recurrent Network
ReLU	Rectified Linear Unit
PSO	Particle Swarm Optimization
BA	Bat Algorithm
CS	Cuckoo Search
MI	Mutual Information
CMI	Conditional Mutual Information
PAC	Partial Autocorrelation
EC	European Commission
REN	Rede Elétrica Nacional
ISONE	Independent System Operator New England
REE	Red Eléctrica de España
PJM	Pennsylvania-New Jersey-Maryland

Chapter I

1 Introduction

This chapter presents a short theoretical background about Electric Energy Systems (EES) key topics and a brief state of the art review in the load and electricity prices forecasting paradigm. EES nowadays are complex systems that cover vast geographical areas and must meet high quality of service and safety standards.
Furthermore, this chapter presents the motivation behind the dissertation, main tasks and objectives are also outlined.

1.1 Electric Energy Systems

The primary goal of EES is to monetize and explore electric energy conveniently. Its development is studied by electrical engineering branches, involving aspects that include generation, transmission, distribution and consumption technologies [1].

In contemporary societies, EES are cost intensive and probably constitute the vast man made system (thousands of km of overhead lines and underground cables, as well as an endless number of elements/infrastructures [2]). They must meet a set of requisites, such as: a continuously availability, independently of the location, electricity must be generated and transmitted as it is consumed (plus loses), which means that electric systems are highly

dynamic, minimizing the production costs as well as the environmental impact, all this fulfilling a set of constrained quality norms (fault tolerance).

1.2 Energy characteristics and challenges

The energy concept is abstract but usually is defined as the capacity to produce work, energy can be classified as primary, secondary, final commercial energy or useful energy depending on the level of processing/conversion it suffered, being primary energy a not converted natural resource (e.g. fossil fuels).

Since the distant days where wood and animal muscles were the primary source of energy for mankind activities, to the present times, where energy is a crucial element that we depend unceasingly to live well, and that is why countries with higher per capita energy consumption are usually those with higher GDP (gross domestic product) per capita. In fact energy demand is closely tied with population and economic growth and based on the projected evolution of these statistical parameters, EIA (U.S Energy Information Administration) estimates that world energy consumption will grow by 48% between 2012 and 2040 [3].

This energy demand growth translates into severe constraints affecting the already scarce and increasingly costly energy resources. The expected declining in a long-term perspective for the traditional coal, gas and oil reserves and a greater attention to its underlying carbon footprint, has led to an exhaustive search and implementation of clean, renewable but also viable sources of energy.

1.2.1 The rise of Electrical Energy

Electrical energy is the result of electric potential energy or kinetic energy conversion processes. Its relatively low entropy form, which can be transmitted at long distances (electric charges flow) and converted into other forms of energy such as motion, light or heat with high energy efficiency.

For these reasons, it's natural when analyzing final commercial energy data, that electrical energy turned into the main energy type at the consumer end. Greatly due to its controllability, instant availability and consumer end cleanliness, doubling its demand growth in comparison with other energy types (as can be seen in Figure 1.1 [4]).

Given this consumer end preference conventional coal, oil, gas, nuclear and other resources will be gradually less used as final commercial energy source, but rather a way to generate electrical energy (process known as electricity generation).

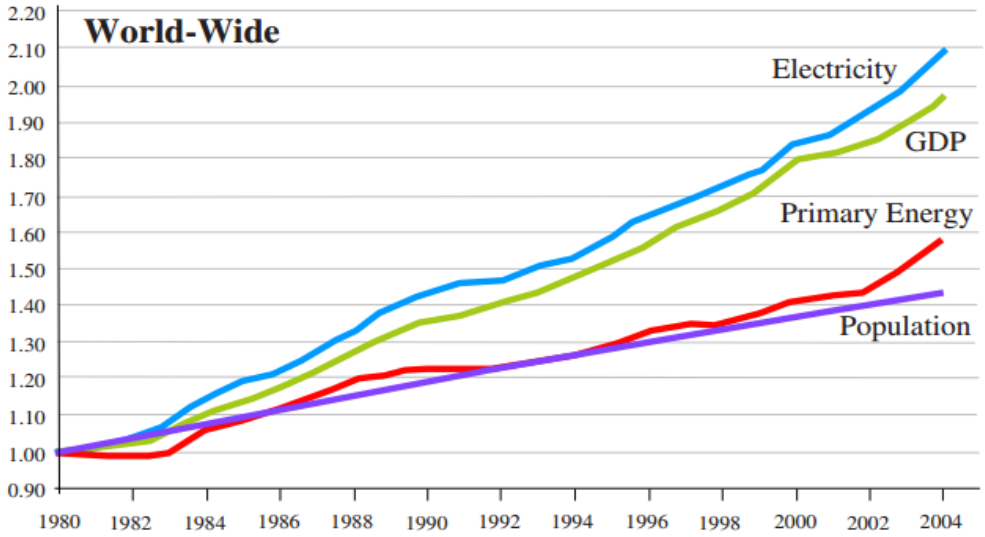


Figure 1.1 - World Growth Rate referred to 1980 value (evolution).

Given this consumer end preference conventional coal, oil, gas, nuclear and other resources will be gradually less used as final commercial energy source, but rather a way to generate electrical energy (process known as electricity generation).

1.3 Electrical Energy and Power

As seen previously EES provide electrical energy to consumers and ensure that contracted power is available so that they can satisfy their loads (consumption). The electrical power unit (SI) is the watt (W) or joule per second (J/s), analogously the energy is measured in joule (J) or watt-second $(W.s)$. Given the real magnitude of these parameters, it's common to use multiples like GW for power and $GW.h$ for energy.

Thus we can define the relationship between energy and power as:

$$P(t) = \frac{dW(t)}{dt} \tag{1.1}$$

$$E(t) = \int P(t)\, dt \tag{1.2}$$

where $E(t)$ denotes energy, $P(t)$ is power and t is the time instant.

1.3.1 Electrical power demand (Load)

As discussed in section 1.1, EES must match in real time the power generation P_G and electric power demand D (equation 1.3).

$$\sum P_{G_t} = D_t \text{ , for every time instant } t \tag{1.3}$$

As an example Spanish power demand daily evolution (also referred as daily load curve) on February 8, 2016, is represented graphically in Figure 1.2. As would be expected from equation 1.2, the area below the curve (gray) represents the total energy.

This daily load curve varies due to a set of factors, such as the time of day, the day of the week, temperature, seasons, latitude etc. [5].

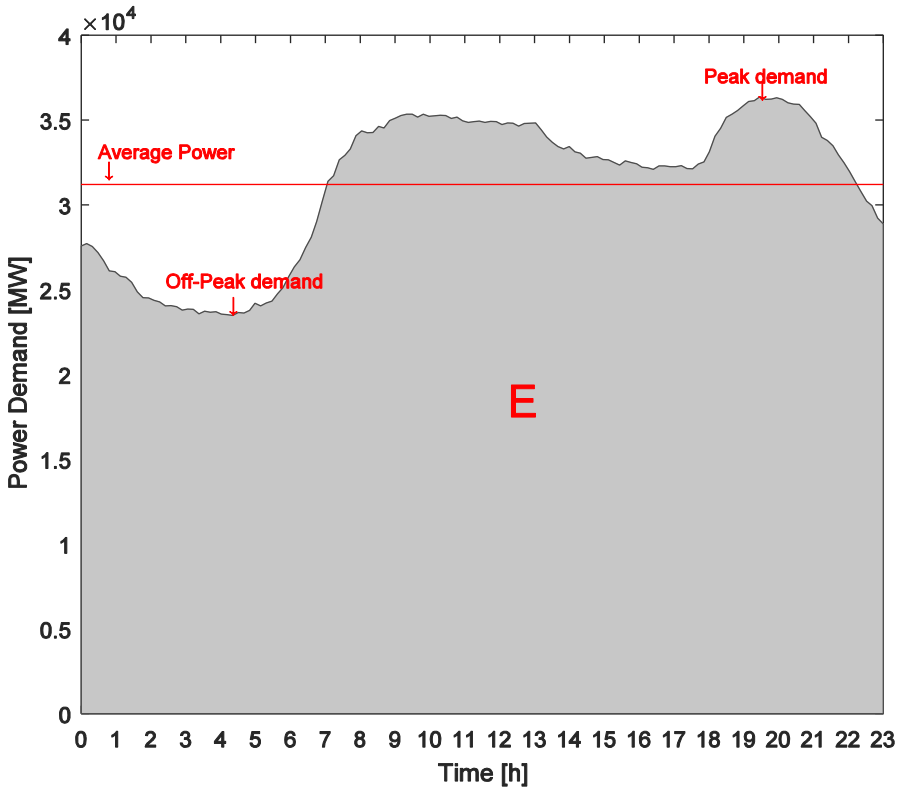


Figure 1.2 – Spanish daily load curve (example).

When analyzing a daily load curve we can highlight some points/regions of interest [5], [6], first the "peak demand" point (maximum power demand) which often occurs in early evening and typically with this increased "stress" for the generation operators it comes an high price. On the opposite way, when the minimum power demand is verified, it receives the designation of "off-peak demand" point. The period approximately between 6 and 9 AM is designated as "morning ramp", it's associated with the end of the night period, marking the relatively rapid transition from lower to higher power demands, and sometimes this "abrupt" transition can be stressful for EES, potentially reflecting in a greater price volatility. Another highlighted region is the red horizontal line indicating the average power for $T = 24h$, which is determined as follows:

$$E(t) = \frac{1}{T}\int_{0}^{T} P(t)\,dt \tag{1.4}$$

where T is the considered time period.

1.4 The electric market

In the last decades, electric markets undergone trough considerable changes guided by liberalization and deregulation concepts, these changes contrast with the previous reality, where the various EES participants were “heavily” regulated by governmental hassles (public service frameworks), with highly vertical business philosophies.

This deregulation process translates into a set of policies to promote free competition markets rather than traditional monopolistic, thereby relieving to a certain extent the government control. The idea behind deregulation is to establish a more proficient market, favoring energy exchanges between markets, increasing security of supply and the main goal, reduce the costs. As a consequence of this targeted policies, available power capacity can be used more efficiently in a large region compared to a small one, whereby integrated markets enhance productivity and improve efficiency.

The first experiences occurred in Chile (1982) following a comprehensive set of economic reforms that took place during this decade, which would be later known as the "Miracle of Chile”. Subsequently in the early 90s Nordic countries restructured their electric markets and established a common Nordic market (Nord Pool Spot market), which was the world's first multinational exchange for trading electric power and today operates in Norway, Denmark, Sweden, Finland, Estonia, Latvia, Lithuania, Germany and the UK. Certain regional US markets, as well as the Australia and Iberian (Spain and Portugal) undergone trough similar restructuring processes.

1.4.1 The Iberian electricity market

In order to liberalize the electricity market, promote price stability, a better generation capacity management, increase the number of participants (consumers and producers) as well as the competition, Portuguese and Spanish governments signed an agreement in 2004 to create the Iberian electricity market (MIBEL). Entering into live mode on July 2007, becoming then only the second European regional market to be formed. MIBEL was implemented as a mixed market (pool and bilateral contracts trading) [7]. Two market operators are responsible for both dairy and intra-diary short-term market management, where the prices are determined based on the supply and demand curves.

Another particular feature of the Iberian market is that by regulatory mandate it favored a paradigm change and investments on new generation technologies (renewable energies). As such the Portuguese scenario of high energetic dependence in 2007, especially from primary energy sources (with a fossil origin), reaching values close to 83%, contrasts 9 years later (2016) with a generation of 64% of the consumed electricity by renewable sources, with a special emphasis on wind energy.

This paradigm change added significant generation capacity but also an intrinsic uncertainty arising from the nature of the renewable energies. For this reason, market participants focus is divided both on the consumption (load) and generation predictability.

1.5 Time-series forecasting and (time horizons)

A time series is a sequence of equally spaced observations in time (sequence of discrete-time observations. Considering a time-series $[x_{t-n}, \dots, x_{t-2}, x_{t-1}, x_t]$, the forecasting (or prediction) task consists in determining the upcoming (unknown) time-series values ($[x_{t-n+1}, \dots, x_{t-2+1}, x_{t-1+1}, x_{t+1}]$), using and modeling the existing time-series. Examples of time-series are financial data (indices, rates etc.), weather data (temperature, wind speed, etc.), as are the load and electricity prices studied in this work.

When we speak about forecasting practices and how they are developed, one important factor is the considered temporal horizon. Consensually, forecast time horizons (for load and electricity prices) are divided into 3 large groups [8][9], to which can be added another for very short-term forecasts [10]:

- ↳ Very short-term: From scarce seconds to half an hour (one hour) prediction.
- ↳ Short-term: Typically, from one hour to one week.
- ↳ Medium-term: From one week to one year.
- ↳ Long-term: Longer than a year.

1.6 Load Forecast

As we saw the energy sector has evolved in the last years towards a wider, horizontal and more complex architecture, largely due to deregulation and search for efficient policies [11], [8], followed by emerging consumption patterns (with an overall growth tendency), a more deregulated supply, where the role of producer and consumer can intersect, etc. All of this has added non-linearity to the load profile [11], leading to poorly conditioned load curves, i.e. with fewer “smooth” regions, and making load forecasting more difficult, e.g. leading to worse correlations with some common exogenous variables, such as weather variables [12], which are frequently used as inputs in state-of-the-art methods [12], [13].

Short-term load forecast (STLF) is the prediction of electric load evolution, within a time horizon, with lead times extending from one hour to one week (168h) as stated in the previous section.

In such a competitive environment, STLF greatly influences decision-making on issues such as: generation scheduling (dispatch), price forecasting, need for maintenance and system liability assessment [12], [14], [15]. For these reasons researchers efforts typically focus in one day (24h) ahead load forecast. Therefore, the challenge and importance of having accurate forecast methods is even more important than before [12]. The existing literature presents a comprehensive spectrum of methodologies for STLF. To simplify they can be grouped into three different categories.

The first presents the statistical methods (time series based) and, as for other commodities, was the first conventional set of methods for STLF. This includes linear/multiple regression; the Box-Jenkins method based on time-series models, such as

autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA); the Holt-Winters exponential smoothing; and methods based on stochastic approaches, like autoregressive conditional heteroscedasticity (ARCH) [16]–[18].

Other common method is the persistence model also known as "Naïve Predictor", this assumes that load value in the instant $t + \Delta t$ is the same that took place in the instant t. This is a very accurate method in very short-term horizons and loses performance when we increase the horizon (Δt).

However these methods are best suited to deal with "smoother" time series, as they are not fully able to cope with the complex non-linear characteristics evidenced in the load time series [19].

To face this complex non-linearity the focus has changed to artificial intelligence (AI) methods [15], given their ability to learn nonlinear relationships between the load and exogenous variables. In this field, countless approaches have been proposed, such as: artificial neural networks (ANNs) [11], [14], [20]–[22]; random forest [19], [23]; support vector machines (SVMs) [24], [25]; fuzzy inference (expert) systems (FIS) [26], [27]; and data mining techniques [28], [29].

Despite the virtues of these AI methods, their complexity makes it hard to find the optimal parameter settings (e.g. decide ANN architecture). In addition to handicaps of each individual method, this has led us to the last category, hybrid methods. These methods take advantage of the best features of single methods. By combining them, it's possible to explore the powerful features of each one, resulting in new topologies.

A hybrid approach is presented in [30] where an ARIMA model – used to forecast the linear load segments – is combined with SVMs – used in the forecast of the non-linear sensitive load segments.

Signal processing techniques applied to the load time series, especially wavelet transform (WT), have also been extensively employed [12], [31], [32]. The results show that wavelet analysis can extract redundant information (e.g. high frequency changes) from the load time series, resulting in a filtering effect that improves forecast accuracy [15], [33].

The concept of extreme learning machine (ELM), linked with hybrid neural networks, has also aroused the interest of researchers [13], [34]. Alternatively, in [12], [35], authors proposed hybrid structures merging ANNs, WT and optimization algorithms.

Three different methodologies were compared in [36] for very short and short-term horizons. The results show that ANNs outperforms SARIMA models and the hybrid topology ANFIS, in terms of mean absolute percentage error (MAPE (%)).

To summarize, recent literature shows that hybrid methods that combine metaheuristics algorithms, AI methods and data processing techniques are among the most accurate, although they can be computationally intensive. An overall performance comparison of these methods reveals that the MAPE for a 24h time horizon is within a 1% to 3% interval [19].

1.7 Price Forecast

As a widely spread traded commodity electricity in the form of energy blocks is sold and bought by producers and consumers who submit their bids in a pool based market, these are analyzed by the market operator which then determines the clearing price (the energy price typically on an hourly basis). But unlike other commodities electricity cannot be queued and stored economically with the exception of pumped-storage hydro plants in certain conditions [37].

Therefore, the power industry operates in a very competitive framework, largely due to deregulation and the search for competition policies with the clear intent to reduce marginal costs (and consequently obtain lower consumer electricity prices at the end of the chain). This deregulated environment brings a degree of uncertainty to electricity prices (price volatility), with profit maximization being the major concern when addressing scheduling of energy generation [37].

As such forecast accuracy is a major subject for producers and consumers. Reliable price forecasting techniques are used by the market players to derive their pool bidding strategies and to optimally schedule energy resources [37], and for this reason can be a key factor between competitors, allowing producers to maximize their profits and consumers to maximize their utilities [33]. Due to this interest forecasting electricity demand and prices has emerged as one of the major research fields in electrical engineering [38].

A set of details from the price time-series makes the task of forecasting prices far from being trivial, these include high frequency, non-constant mean and variance, multiple seasonality, calendar effect, high level of volatility and high percentage of unusual price movements [38]. Another set of details affecting forecasting accuracy is the uncertainty related with fuel prices, future additions of generation and transmission capacity, regulatory structure and rules, future demand growth, plant operations and climate changes [39]. Presently, some markets with strong penetration of renewable energy sources, particularly wind power (as the Iberian market), have special regulatory regimes (feed-in-tariff scheme) imposing new challenges that need to be accommodated, mainly due to the fluctuating feed-in of wind power. For example, a day with prices close to zero can be followed by a day of maximum prices, which together with transmission congestion, contributes to price volatility.

A variety of methods have been developed for electricity price forecasting and most of them are also used load forecasting [38], the majority of these, as this work does, focus in lead times ranging from one hour to a week ahead forecasting, this time horizon constitutes the short-term price forecast (STPF) category.

Most methods are based in time series models, meaning that the scope is on the price time series past behavior, with addition it can be complemented with some exogenous variables. The first major approach are Parsimonious stochastic methods, for example ARIMA [40] and generalized autoregressive conditional heteroskedastic (GARCH) [41], [42] models are proposed, other time series models based on regression are presented in [43], [44]. A disadvantage of this methods is its high computational cost [37].

Wavelet Transform is a commonly used feature to better deal with the non-constant mean and variance and the significant number of outliers is the wavelet transform. The transformed time-series presents a better behavior (more stable variance and less outliers) than the original price series thus resulting in a better performance [33]. For exemple in [33], [45] the authors combine WT with the classical ARIMA and GARCH models.

Given the limitations presented by the first set of methods, authors focused their efforts on AI methods, thus constituting the second major group of methods, this are better suited to deal with hard non-linear relationships highlighted in the price time series, thus being computationally more efficient [37]. A large portion of this AI methods are centered in ANNs , therefore authors in [37], [46], [47] used feed forward, radial basis and Elman networks in classical approaches, in [48] authors propose an improved training process for the ANN.

In order to conjugate synergies from different features the current approaches focus on hybrid methods, an example is the use of WT and a fused version of neural networks and fuzzy logic [49], in [50] the authors used cascaded neural networks (CNNs) combined with the chemical reaction optimization algorithm, to properly train the CNN, other hybrid approach was followed in [51] where the fuzzy ARTMAP, wavelet transform and firefly algorithm were used to perform the STPF.

1.8 Motivation and Goals

The existing complexity both in load and price time series makes the forecasting task far from trivial, due to a set of characteristics such as multiple seasonality, not constant mean and variance, high volatility (significant number of spikes).

As we saw in the two previous sections, a diverse group of methods is applied both in load and price forecasting, and in a generic form, they can be grouped as illustrated in Figure 1.3.

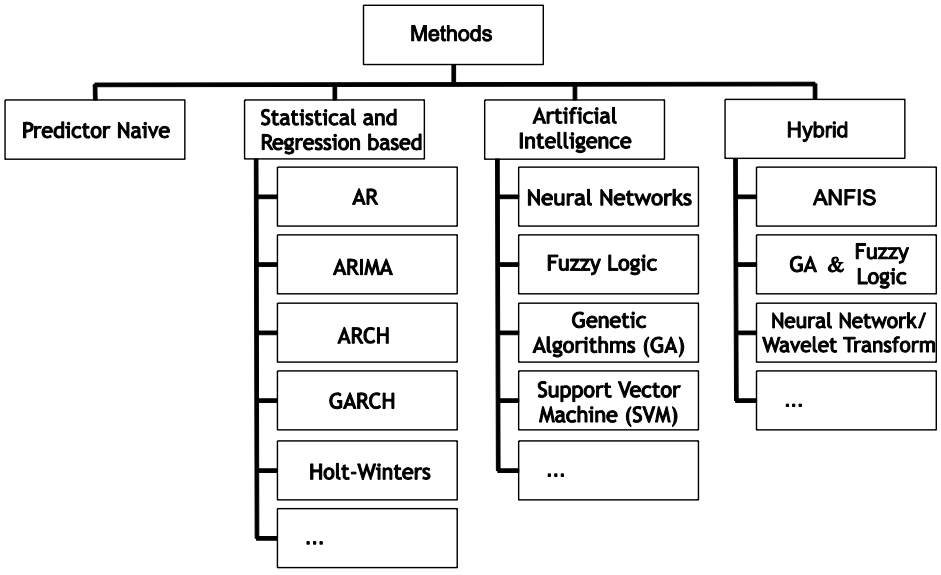


Figure 1.3 – Schematic illustration of the various methods applied in STLF and STPF.

Being a very active research area dealing with sensitive issues, and given the "rich" state of the art proposed so far, the task of exploring this diverse methods and techniques, as well in a subsequent stage, the pursuit for a contribution with a new approach, able to minimize the error compared to other reported methodologies, constitutes the major motivation behind this work. Thus the search for a STLF and STPF approach will guide this dissertation.

This new approach will fit in the hybrid methods classification (Figure 1.3), given the benefits revealed by literature review. From all the methods those who stand out and will be used are: ANN to perform the forecast, WT in a preemptive processing stage to better deal with the data series characteristics.

1.9 Dissertation structure

In this chapter a background review regarding key concepts, as well as the motivation and goals for this work dissertation were presented, the following chapters are organized in the following manner:

i. Chapter I presents a background review regarding key concepts, as well as the motivation and goals for this work
ii. Chapter II serves as a theoretical revision of the wavelet associated concepts
iii. Chapter III presents an extensive theoretical formulation of concepts associated with neural networks.
iv. Chapter IV introduces the optimization algorithms theme and the background theory behind the chosen algorithms employed in this work.
v. Chapter V and VI reveal the proposed methodology, the case studies where it was employed (STLF and STPF) and a general discussion about the achieved results.
vi. Chapter VII presents the main conclusions of this dissertation.

Chapter II

2 Wavelet Theory

Numerous works related with data series analysis and particularly the ones focused in forecasting models have implemented successfully wavelet analysis, as an effective form of signal representation. So this chapter presents a comprehensive explanation of the underlying wavelet theory, from its origins, its advantages, the way it's analytically implemented and other involving details.

2.1 Background

In 1909 with is theory of orthogonal systems, Alfred Haar was the first to conceive the idea behind what we call a "wavelet". This groundbreaking work resulted in the discovery of a set of rectangular basis functions (what constitutes the Haar wavelet family), which are the simplest family of wavelets (Figure 2.1).

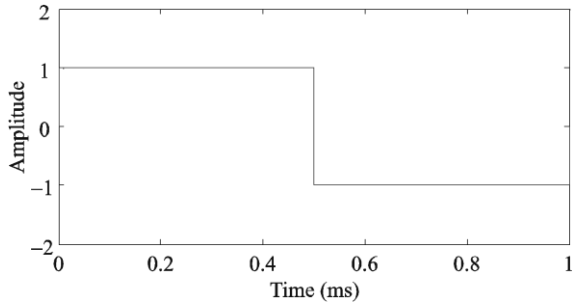


Figure 2.1 - Rectangular basis function example.

The first evident advantage is that *Haar* basis functions are scalable for different intervals, in contrast with Fourier basis functions which operate in the interval $[-\infty,+\infty]$.
The next major contributes arrived in 70's by the hands of *Jean Morlet* which successfully implemented a technique of scaling and shifting in an analysis window of a function. *Morlet* called the resulting waveform(s) from the analysis "wavelet(s)". Later in 1984 *Morlet* partner up with *Grossman* to prove that a signal could be transformed to a wavelet form and then reverse the process in a procedure without information loss.

So we can define a wavelet as a small and finite wave with an irregular oscillating pattern, with finite energy. So a wavelet transform is basically a process of repeated variations of scale, translations and convolution between 2 signals, allowing the extraction of the so called wavelet coefficients.

The small scales are used to properly depict the higher frequencies and larger scales correspond to low frequency components, Figure 2.2.

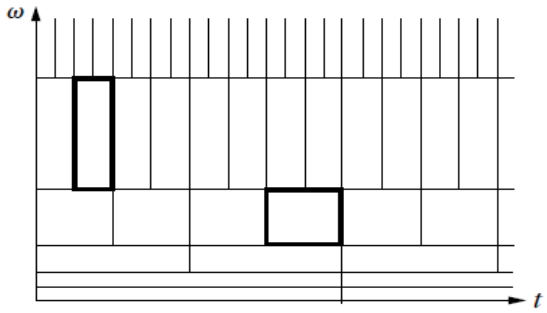


Figure 2.2 - Frequency vs Time resolution.

Characteristics viewed so far reveal some distinctive traits that arise with the use of wavelets, for example, unlike short time Fourier transform (STFT), which consists in performing multiple Fourier transforms over smaller windows translated in time, where the window size is important because if the window is to narrow it results in poorer frequency resolution. Conversely a longer time window improves frequency resolution while resulting in poorer time resolution because the Fourier transform loses all time resolution over the duration of the window [52], this effect is illustrated in Figure 2.3.

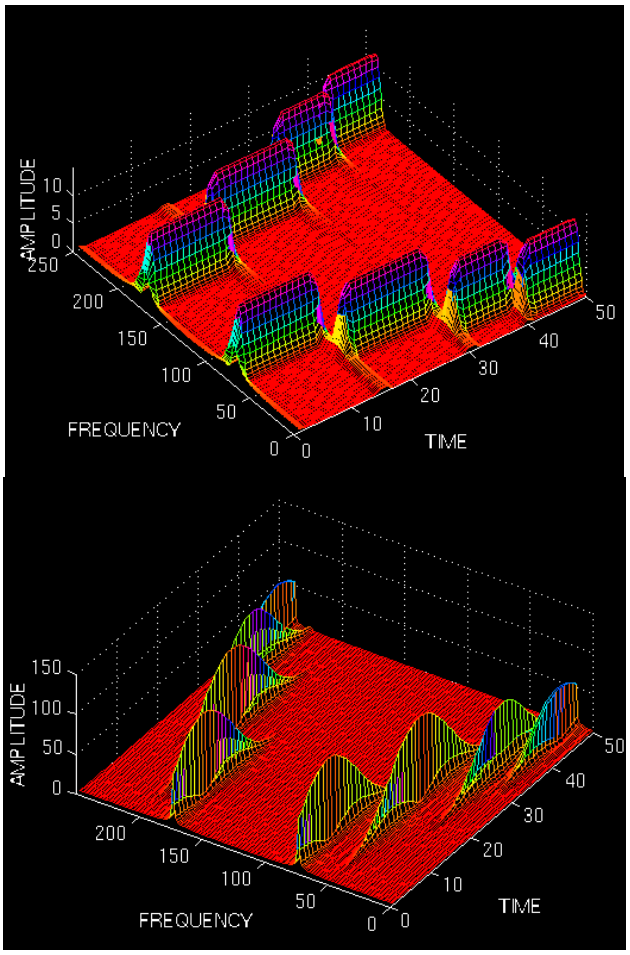


Figure 2.3 - Narrow to Wide Window effect in STFT [53].

Wavelet Transform enables variable-sized windows, allowing the use of long time intervals for more precise low-frequency information, and shorter windows for high-frequency information [52], thus better for revealing hidden features in the signal, such as trends, noise detection, discontinuities in higher derivatives and self-similarity [12]. Furthermore, wavelet analysis can often compress or de-noise a signal without appreciable degradation, i.e. it has a filtering effect [12]

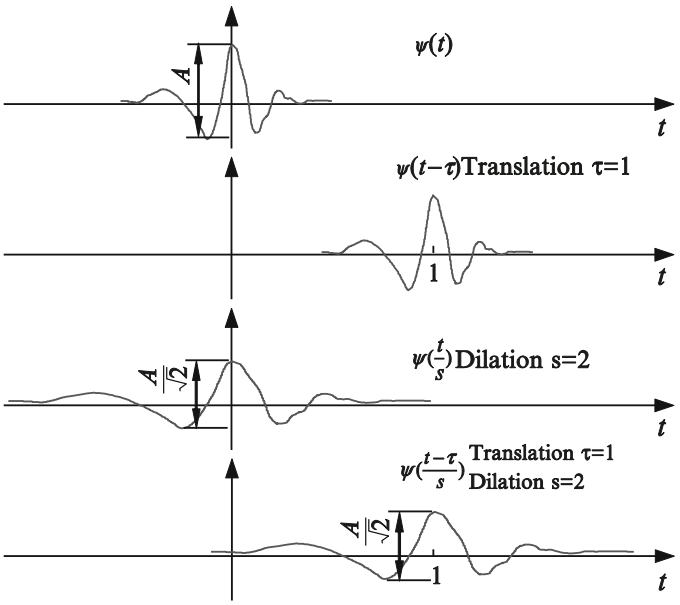


Figure 2.4 - Example of translation by a time constant τ and dilation by scaling factor s.

So WT decomposes the signal into a time-scale/frequency domain in a bi-dimensional analysis (time and Fourier space). As illustrated in Figure 2.4, the WT process consists in shifting a base wavelet along the signal (translation in time domain) and scale it (dilation or contraction) therefore changing the scale (which is inversely proportional to the frequency) and then, find resemblances interpreted in the form of coefficients. An overview of the complete process is illustrated in Figure 2.5, where we can see the operations both in the time and frequency domains.

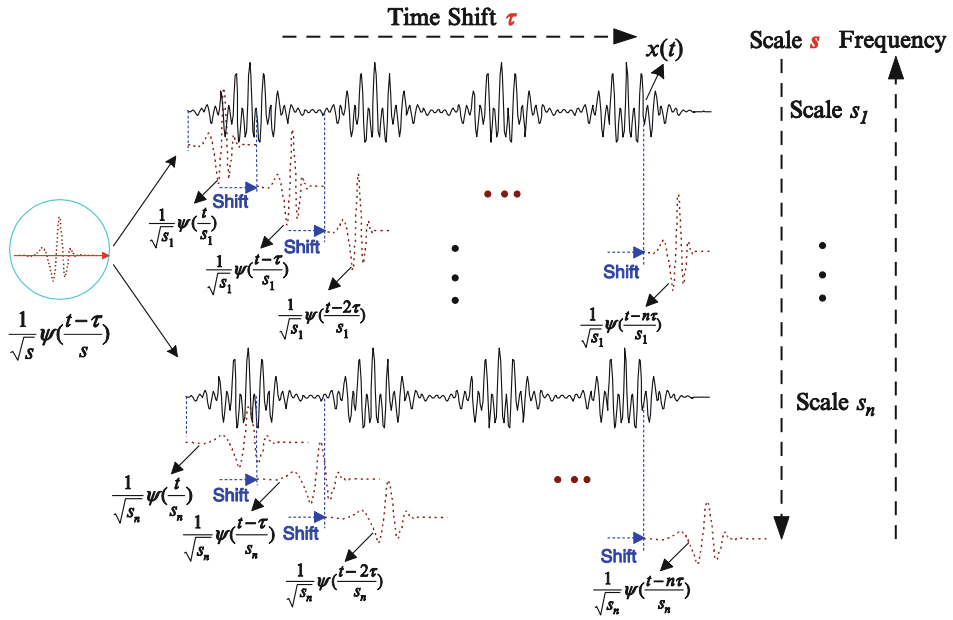


Figure 2.5 - Wavelet transform process overview.

Another important step was the introduction of multiresolution analysis presented in the 10 year period between 1989 and 1999 by *Mallat* and *Meyer*, which allowed investigators to build their own family of wavelets and facilitate the transformation process.

This theory led to innumerable research projects and applications, which emerged based on WT. From signal analysis, video coding [54] to image processing [55], and since is very good to extract key features from a signal it is used as an aiding mechanism in forecasting methods [34][56].

2.2 Continuous Wavelet Transform

To be considered as a wavelet, the candidate must satisfy the following admissibility condition:

$$C_\psi = \int_{-\infty}^{+\infty} \frac{|\Psi(f)|^2}{|f|} df < \infty \tag{2.1}$$

where $\Psi(f)$ is the Fourier transform (frequency domain) of the wavelet function $\psi(t)$. To guarantee that equation 2.1 is true, Fourier transform at frequency zero must be null: $|\Psi(0)|^2 = 0$. This implies that the average value of $\psi(t)$ in time domain is null, as illustrated in equation 2.2.

$$\int_{-\infty}^{+\infty} \psi(t)\, dt = 0 \tag{2.2}$$

A generic family of scaled and translated wavelets $\psi_{s,\tau}(t)$ is obtained in the following manner:

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-\tau}{s}\right), s > 0 \wedge \tau \in R \tag{2.3}$$

where τ is the time shifting parameter and dilation is achieved using the scale parameter *s*.

So the continuous wavelet transform (CWT) of a signal $x(t)$ is calculated as follows:

$$\begin{aligned} wt_x(s,\tau) = \langle x, \psi_{s,\tau} \rangle &= \int_{-\infty}^{+\infty} x(t) \times \psi^*_{s,\tau}(t) \\ &= \frac{1}{\sqrt{s}} \int_{-\infty}^{+\infty} x(t) \times \psi^*\left(\frac{t-\tau}{s}\right) dt \end{aligned} \tag{2.4}$$

In equation 2.4, $wt_x(s,\tau)$ are the wavelet coefficients, s is the scaling parameter which determines time and frequency resolution, τ is the shifting parameter. The variable ψ^* refers to the base wavelet complex conjugate of $\psi(t)$, $\langle x, \psi_{s,\tau} \rangle$ represents the inner product which generalizes the dot product to abstract vector spaces over a field of scalars. The factor $\frac{1}{\sqrt{s}}$ is applied to ensure energy preservation, by other words the energy of the base wavelet and its scaled and translated wavelet peer remains the same, $E_s = \delta$ (finite).

$$E_s = \langle \psi, \psi \rangle = \int_{-\infty}^{+\infty} |\psi(t)|^2 \, dt = \delta \tag{2.5}$$

The same can be demonstrated below with the following auxiliary calculations. First we make the variable change:

$$\frac{t-\tau}{s} = u \Leftrightarrow \frac{t}{s} = u - \frac{\tau}{s} \Rightarrow \frac{d}{dt}\left(\frac{t}{s}\right) = \frac{d}{dt}\left(u - \frac{\tau}{s}\right) \Leftrightarrow$$
$$\Leftrightarrow \frac{1}{s} = \frac{du}{dt} \Leftrightarrow dt = s \, du \tag{2.6}$$

Then resuming equation 2.3, we can compute the scaled and translated wavelet energy E_s, making the integration by substitution:

$$E_s = \langle \psi_{s,\tau}, \psi_{s,\tau} \rangle = \frac{1}{s} \int_{-\infty}^{+\infty} |\psi(u)|^2 \, s \, du = \int_{-\infty}^{+\infty} |\psi(u)|^2 \, du \overset{t}{\rightleftharpoons} \delta \tag{2.7}$$

Another important consideration is that when executing this transformation in real data (discrete signal), CWT also operates in discrete periods. Therefore, "continuity" refers to the fact that we can operate at any scale and the same is valid for the time shifts, both are made smoothly over the signal domain.

Properties of Continuous Wavelet Transform

i. Superposition

Having:

$x(t) \wedge y(t) \in S^2(\mathbb{R})$. Where $wt_x(s,\tau)$ is the CWT of $x(t)$ and $wt_y(s,\tau)$ is the CWT of $y(t)$.

k_1 and k_2 as constants, then:

If $z(t) = k_1 x(t) + k_2 y(t) \Rightarrow wt_z(s,\tau) = k_1 wt_x(s,\tau) + k_2 \, wt_y(s,\tau)$.

ii. Covariant (Translation)

If $wt_x(s,\tau)$ is the CWT of $x(t)$, then $wt_x(s,\tau - t_0)$ is the CWT of $x'(t) = x(t - t_0)$.

iii. Covariant (Change of Scale)

If $wt_x(s,\tau)$ is the CWT of $x(t)$, then $wt_x\left(\frac{s}{a}, \frac{\tau}{a}\right)$ is the CWT of $x'(t) = x\left(\frac{t}{a}\right)$.

2.3 Inverse Continuous Wavelet Transform

The correspondent inverse transformation enables perfect signal reconstruction from the wavelet coefficients $wt_x(s,\tau)$, and is defined as:

$$wt_{x'}(s,\tau) = \frac{1}{C_\psi} \int_0^{+\infty} \frac{ds}{s^2} \int_{-\infty}^{+\infty} wt_x(s,\tau) \times \psi_{s,\tau}(t) \, d\tau \tag{2.8}$$

To ensure that the inverse CWT exists, the chosen base wavelet needs to satisfy the admission condition in equation 2.1.

2.4 Discrete Wavelet Transform

When implementing the CWT, modifications of scale and translations are performed continuously, this introduces redundancy in the process.

To avoid this time consuming task and reduce the redundancy we can discretize the scale and translation parameters and obtain the Discrete Wavelet Transform (DWT). This sampled version of CWT is just as accurate and more efficient then CWT [57].

To discretize s and τ, the strategy is to use logarithmic (*dyadic)* discretization, this discretization ensures a good convergence rate instead of a polynomial one, and is expressed as follows:

$$\begin{cases} s = s_0^j \\ \tau = k\tau_0 {s_0}^j \end{cases}, s_0 < 1, \tau_0 \neq 0, j, k \in \mathbb{Z} \tag{2.9}$$

where j represents the level of decomposition. By convention we assume: $s_0 = 2$ and $\tau_0 = 1$, so as consequence equation 2.3 is rewritten as follows:

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}} \psi\left(\frac{t - k2^j}{2^j}\right), j, k \in \mathbb{Z} \tag{2.10}$$

So now the generic family of scaled and translated wavelets, allows scale variations of 2^j (sub bands), this wavelets constitute a orthonormal basis so they are called orthogonal wavelets [58].

$$wt_x(j,k) = \langle x, \psi_{j,k} \rangle = \frac{1}{\sqrt{2^j}} \int_{-\infty}^{+\infty} x(t) \times \psi^*\left(\frac{t - k2^j}{2^j}\right) dt \tag{2.11}$$

So for a discrete signal $x(n)$, the equation 2.11 is rewritten and DWT is computed as follows:

$$wt_x(j,k) = \frac{1}{\sqrt{2^j}} \sum_{t=0}^{T-1} x(n) \times \psi^*\left(\frac{n - k2^j}{2^j}\right) \tag{2.12}$$

where T is the length of the signal and n is the discrete time index. So DWT decomposes the signal into different frequency sub-bands. DWT is tied with MRA (multiresolution analysis) concept, where mother wavelets such as *Haar*, *Daubechies*, *Coiflets* are examples of orthogonal wavelets.

2.5 Inverse Discrete Wavelet Transform

To rebuild the original signal from the discrete wavelet coefficients, we must ensure compliance with the criterion known as wavelet frame, presented below:

$$A|x(t)|^2 \leq \left|\langle x, \psi_{j,k}\rangle\right|^2 \leq B|x(t)|^2$$
$$A \leq B, \forall\ A, B \in R^+ \tag{2.13}$$

So A and B define frame bounds, they depend on the scale and translation parameters chosen and the considered base (mother) wavelet. Signal reconstruction is achieved through the inverse discretized wavelet transform that is defined as follows:

$$x(t) = \frac{2}{A+B} \sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} wt_x(j,k)\, \psi_{j,k}(t) \tag{2.14}$$

If $A = B$ we are in the presence of *tight* frame, this implies that equation 2.14 can be simplified to the following form:

$$x(t) = \frac{1}{A} \sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} wt_x(j,k)\, \psi_{j,k}(t) \tag{2.15}$$

The discretization expressed in equation 2.9 requires the selection of orthogonal wavelets under a *tight* frame with $A = B = 1$.

2.6 Multiresolution Analysis

So, as we saw in the background section, *Mallat* was responsible for the MRA introduction. Supposing we have a space $S^2(\mathbb{R})$, MRA consists on finding successive approximation subspaces $\{V_j\}$, where $j \in \mathbb{Z}$ relates to the multiresolution in these subspaces. These satisfy a set of properties, with special emphasis for one that requires the existence of an orthogonal basis. This property basically states that there must be a function $\phi(t) \in V_0$ (scale function), whose closed subspaces $\{\phi(t-k)\}_{k\in\mathbb{Z}}$, (translated scale function) establish an orthogonal basis of the zero scale space V_0.

This reveals that all closed subspaces $\{V_j, j \in \mathbb{Z}\}$, are obtained through translations of the original scale function. To properly arrange an orthogonal basis of $S^2(\mathbb{R})$ space, then we need to define $W_j, j \in \mathbb{Z}$ as orthogonal complement of V_j in V_{j-1}, these subspaces constitute wavelet subspaces at scale j.

This results in the following properties:

$$V_{j-1} = V_j \oplus W_j \quad \wedge \quad W_j \perp W_{j'}, j \neq j' \tag{2.16}$$

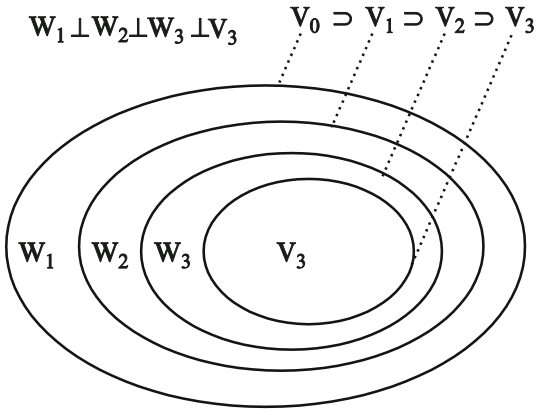


Figure 2.6 - Wavelet (closed) subspaces.

So we can express MRA with the following equation:

$$V_j = V_J \oplus \bigoplus_{k=0}^{J-j-1} W_{j-k} \tag{2.17}$$

If we apply this formula to the subspaces in Figure 2.6, the result is:

$$V_0 = V_1 \oplus W_1 = V_2 \oplus W_2 \oplus W_1 = V_3 \oplus W_3 \oplus W_2 \oplus W_1 \tag{2.18}$$

In the equations above (2.17 and 2.18), the symbol $\oplus$ is the summation operator, $\perp$ is the orthogonal operator and J is the highest predetermined scale. So we can conclude that all subspaces $W_j, j \in \mathbb{Z}$ are orthogonal.

Since W_j inherits the property of regular dilation from the space V_j:

$$x(t) \in V_j \Leftrightarrow x(2^j t) \in V_0 \tag{2.19}$$

Then it's also true that:

$$x(t) \in W_j \Leftrightarrow x(2^j t) \in W_j \tag{2.20}$$

So we conclude that when wavelet families $\psi_{j,k}$ with j and $k \in \mathbb{Z}$ form a set of orthogonal bases in $S^2(\mathbb{R})$ space, then they are designated orthogonal wavelets (like the *Daubechies* family).

2.6.1 Orthogonality

Two functions from a set of functions $\{f_1, f_2, \dots f_q\}$ in a closed space (interval [a, b]) are orthogonal (with respect to this inner product) if the integral equals zero:

$$\langle f_i, f_j \rangle = \int_a^b f_i(t) * f_j(t)\, dt = \sum_a^b f_i(t) * f_j(t) = \|f_i\|^2 \delta_{i,j} \tag{2.21}$$

where $\delta_{i,j}$ is the Kronecker delta function:

$$\delta_{i,j} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{2.22}$$

So for distinct functions (neither symmetrical), its inner product is null.

2.6.2 Orthonormality

Translations and scale changes of a wavelet need to be orthonormal to itself, to not have any affect in the coefficients, thus enabling the perfect signal reconstruction, thus respecting the next condition:

$$\langle f_i, f_j \rangle = \int_a^b f_i(t) * f_j(t)\, dt = \sum_a^b f_i(t) * f_j(t) = \delta_{i,j} \tag{2.23}$$

2.6.3 Orthogonal Wavelet Transform

From MRA theory another version of the discrete wavelet transform can be implemented. Suppose we have a signal $x(t) \in V_0$, where $j = 0$ represents the zero scale space, with that we can decompose this space into two subspaces using equation 2.16. By doing that we get W_1which contains the detailed information and V_1 the approximate information about $x(t)$. If we want to go further then we use V_1 to perform another decomposition, and we repeat this task until the desired scale j is achieved, this is what is called an orthogonal wavelet transform.

In this process when we project $x(t)$ into V_j we get approximation coefficients, the same procedure but in the W_j subspace allows the detailed coefficients extraction. For this purpose it's required a wavelet function (filter) and a scaling function (filter), then a sort of discrete convolution is performed:

$$\begin{cases} a_{j,k} = \langle x, \phi_{j,k} \rangle = \sum_{k=0}^{T-1} x(t) * \phi_{j,k}(t) \\ d_{j,k} = \langle x, \psi_{j,k} \rangle = \sum_{k=0}^{T-1} x(t) * \psi_{j,k}(t) \end{cases} \tag{2.24}$$

where $\phi(t)$ is the scale function and the wavelet function is represented as $\psi(t)$, $d_{j,k}$ are the detailed (wavelet) coefficients and represent the signal high frequency components, these are obtained applying the wavelet function to the sampled signal. The approximation or scaling coefficients $a_{j,k}$, decode the signal behavior at lower resolutions and are the result of the scale function application.

2.6.4 Inverse Orthogonal Wavelet Transform

With the approximated and detailed coefficients obtained, we can reconstruct the original signal, using the equation below:

$$x(t) = \sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} d_{j,k}\psi_{j,k}(t) + \sum_{k=-\infty}^{+\infty} a_{J,k}\phi_{j,k}\,(t) \tag{2.25}$$

where J is a predetermined scale. This equation is related with equation 2.14 and by using orthogonal wavelets we get $A = B = 1$.

2.6.5 Multiresolution Filters and *Mallat* Algorithm

From previous sections we verified the importance of two functions: the scale function $\phi(t)$ and the wavelet function $\psi(t)$. There is a relation between the two, but only among two consecutive scales, so they can be expressed as dual-scale equations as follows:

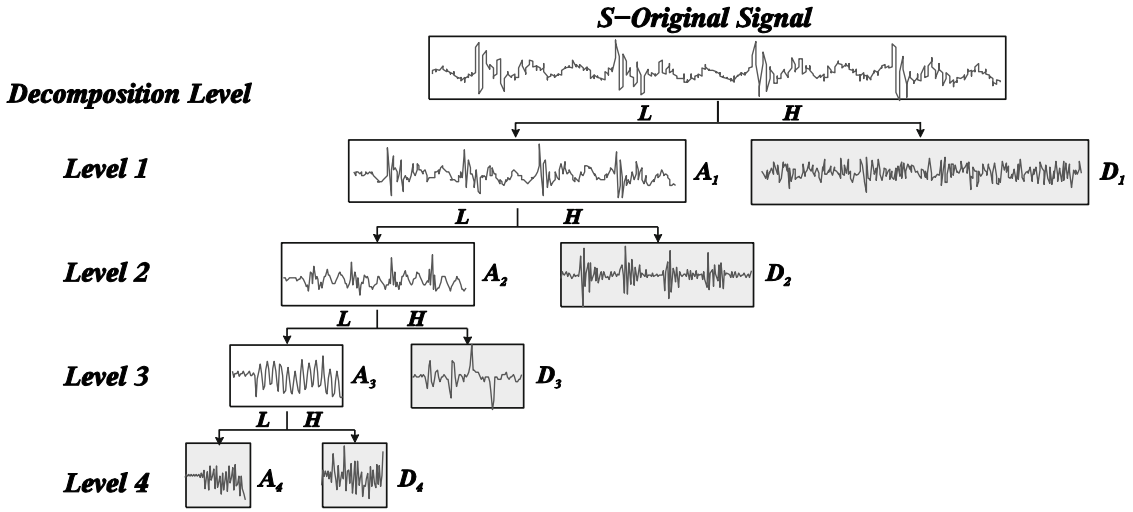
$$\begin{aligned} \phi(t) &= \sqrt{2}\sum_{n} l(n)\,\phi(2t-n) = \sqrt{2}\sum_{n}\langle\phi,\phi_{j-1,n}\rangle\,\phi(2t-n) \\ \psi(t) &= \sqrt{2}\sum_{n} h(n)\,\phi(2t-n) = \sqrt{2}\sum_{n}\langle\psi,\phi_{j-1,n}\rangle\,\phi(2t-n) \end{aligned} \tag{2.26}$$

where coefficients $l(n)$ and $h(n)$ refer to the pair of low pass and high pass wavelet filters respectively, and are used to perform the DWT.

So with this assumptions, *Mallat* developed a fast way [59], [60] to apply MRA, and therefore obtain low frequency components (approximation coefficients-A_i) and high frequency components (approximation coefficients-D_i) for each decomposition level. The same can be computed using the following equations system:

$$\begin{cases} a_{j,k} = \sqrt{2}\sum_{n} l(m-2k)\,a_{j-1,m} \\ d_{j,k} = \sqrt{2}\sum_{n} h(m-2k)\,a_{j-1,m} \end{cases} \tag{2.27}$$

The coefficients of the dual scale equation work like a filter, then *Mallat's* algorithm is practically a two-channel filter bank, with a down-sample at the output. In the sense that scale function and wavelet function work as low-pass filter and high-pass filter respectively, an illustration of this process can be seen in Figure 2.7.



Note: L−low pass filter; H−high pass filter; A_i−aproximation coefficient at level i; D_i−detailed coefficient at level i.

Figure 2.7 - Example of DWT filter bank decomposition (tree).

In the above figure and in equation 2.28 we can visualize signal decomposition in 4 consecutive levels, considering the original signal as our "first" set of approximation coefficients (A_0), clearly an iterative process occurs where the approximation coefficients are decomposed ("in filter banks") into new approximation coefficients and detailed ones. So for each new level, the approximation coefficients are the result of convolution between the approximation coefficients on the previous level with the coefficients of the low pass filter. A similar task occurs to obtain the detailed coefficients, however the convolution is made with the coefficients of the high pass filter and the previous approximation coefficients.

$$\begin{aligned} S &= A_1 + D_1 \Leftrightarrow \\ &= A_2 + D_2 + D_1 \Leftrightarrow \\ &= A_3 + D_3 + D_2 + D_1 \Leftrightarrow \\ &= A_4 + D_4 + D_3 + D_2 + D_1 \end{aligned} \tag{2.28}$$

Another reading is that the number of coefficients becomes less and less as we go through more and more levels. The 4 levels correspond to scales $2^1 = 2,\ 2^2 = 4,\ 2^3 = 8$ and $2^4 = 16$.

Although dyadic decimation in *Mallat* algorithm [59] guarantees a faster computational process, it can lead to some data loses, which is a very sensitive question when working with forecasts [61]. To overcome this obstacle we can use the DWT decomposition available in MATLAB® (Wavelet Toolbox), this one can ensure a redundant signal decomposition, where the extra storage needed is not excessive [62].

2.7 Base Wavelet (families)

A wide range of base (mother) wavelets exists and is widely described and used in the literature, some are fitted when performing CWT and others when doing DWT. The most

common are *Haar, Daubechies, Coiflets, Symlets, Meyer, Gaussian, Mexican hat, Morlet, Biorthogonal and Reverse Biorthogonal Wavelets.*

In particular those used in DWT don't have an explicit expression, so regarding base wavelets it's important to consider their different shapes and properties as shown in the next table [63]. The order of a wavelet is essentially a measure of its differentiability.

Table 2.1 - Base Wavelet properties.

Property	Haar	DbN	CoifN	SymN	Meyer	Mexh	Morlet	Bior/RevBior(Nr.Nd)
Infinitely regular					x	x	x	
Arbitrary regularity		x	x	x				x
Compactly supported orthogonal	x	x	x	x	x			
Compactly supported biorthogonal					x			x
Symmetry	x				x	x	x	x
Asymmetry		x						
Near symmetry			x	x				
Arbitrary number of vanishing moments		x	x	x				x
Vanishing moments for φ(t)			x					

2.8 Mother Wavelet Selection

One important aspect when working with wavelets, is how to select the best suited mother wavelet given our data (signal). Once more similar the mother wavelet is to our signal then higher the amplitude of the wavelet coefficients.

In the literature, criterions to make this optimum selection are divided into two groups [60], [64], the qualitative and quantitative criterions.

2.8.1 Qualitative Criterions

This type of selection is almost purely based on the properties examination of the different mother wavelets analysis, is complemented with a visual verification to find the mother wavelet whose resemblance the most to our signal. A summary of these selection procedures can be found in [60], [64].

2.8.2 Quantitative Measures/Criterions

Common qualitative criteria include the use of minimum description length (MDL), maximum cross correlation analysis, other tendency is to use information extraction criterions, a number of this measures exist in the literature, in this work we followed the approach proposed by *R.Yan* [60]:

The energy of a signal is a good measure to portray the information contained on it. When dealing with a discrete (sampled) signal $x(t)$, it's energy is computed the following way:

$$E_x = \langle x, x \rangle = \sum_{i=0}^{T-1} |x(i)|^2 \tag{2.29}$$

where T is the length of the signal and $x(i)$ is its amplitude for each sampling time. This same measure have been applied to the wavelet coefficients that represent the signal, when working with DWT this coefficients are given by the resultant detailed coefficients $d_{j,k}$.

$$E_{energy} = \sum_{s} \sum_{\tau} |wt(s,\tau)|^2 \tag{2.30}$$

This energy can be distributed for each scale, this is for each sub-band:

$$E_{energy} = \sum_{s} \sum_{\tau} |wt(s,\tau)|^2 \tag{2.31}$$

Therefore, the Maximum energy criterion states that: The base wavelet that extracts the largest amount of energy from the signal, represents the most appropriate wavelet for extracting features from it.

However this criterion doesn't take into account the spectral distribution of the energy, when this is important to ensure an effective features extraction. To tackle this limitation we can use Shannon entropy measure, as a good indicator of the uncertainty associated with the distribution of energy through the scales.

First, we need to find the energy probability distribution p_i, for each wavelet coefficient, which is defined as:

$$p_i = \frac{|wt(s,i)|^2}{E_{energy}(s)} \tag{2.32}$$

Ensuring a normalization by scale: $\sum_{i=1}^{T} p_i = 0$. Then we can compute the Shannon entropy as follows:

$$E_{entropy}(s) = -\sum_{i=1}^{T} p_i \log_2 p_i \tag{2.33}$$

The obtained Shannon entropy is bounded to the following limits:

$$0 \leq E_{entropy}(s) \leq log_2\, T \tag{2.34}$$

So the Minimum Shannon entropy criterion states that: The base wavelet that minimizes Shannon entropy of the wavelet coefficients represents the most appropriate wavelet.

This two previous metrics can be combined into one, constituting the energy-to-Shannon entropy ratio, which is the fraction between the energy of the signal and its entropy, and is calculated as follows:

$$R(s) = \frac{E_{energy}(s)}{E_{entropy}(s)} \tag{2.35}$$

Energy-to-Shannon entropy ratio criterion states that: The base wavelet that has produced the maximum energy-to-Shannon entropy ratio should be chosen as the most appropriate wavelet. Additional quantitative measures are presented in [60], [64].

Chapter III

3 Neural Networks

When studying artificial intelligence techniques, a mandatory subject to learn about are neural networks, these have been efficaciously applied into several research areas and problems. Designed to emulate some aspects of the human brain functioning, they are able to perform model tasks by considering examples, and specifically in this work to serve as forecasting tools, using available historical data. So this chapter addresses the underlying theory behind ANNs.

3.1 Background

Neural networks history dates back to 1943 when the neurophysiologists McCulloch and Pitts tried to replicate the model of a biological neuron. In a simplified view we can say that dendrites accept inputs (electric signals), these are processed in soma, and then the axon acts to transform this signals into outputs that are propagated to other neurons due to synapses.

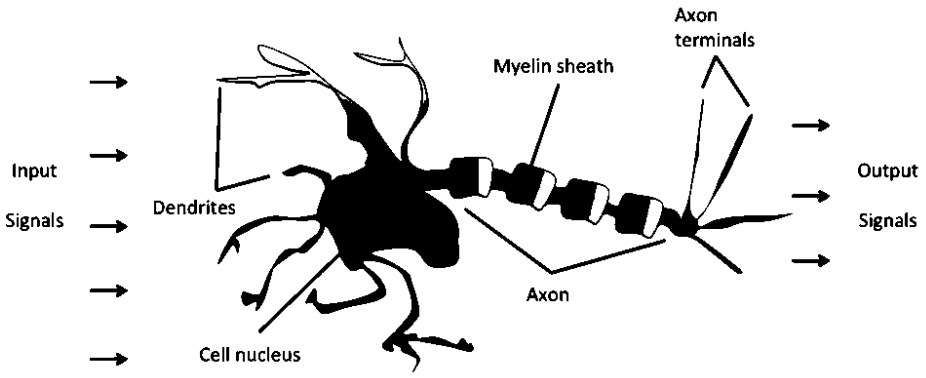


Figure 3.1 - Biological neuron structure (behavior).

Inspired by the real behavior they've built an artificial neuron replica as a weighted sum of binary inputs, being the outcome accepted when it reaches a certain threshold defined by the activation function $f(\omega_i, x_i)$. This function evaluates the weighted sum according to the following equation:

$$y = f\left(\sum_{i=1}^{n} \omega_i \times x_i\right) \tag{3.1}$$

where y symbolizes the artificial neuron output, x_i represents the entries to the artificial neuron and ω_i is the normalized weight of the established connection. In *McCulloch-Pitts* model (Figure 3.2) the *Heaviside* step function work as activation function, so only binary results are produced.

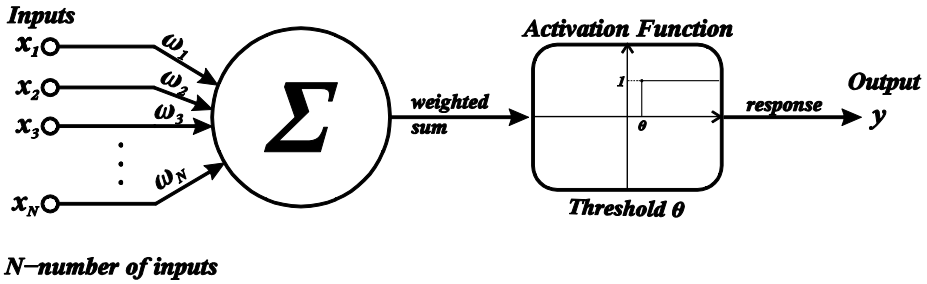


Figure 3.2 - McCulloch-Pitts neuron model.

Later in 1958 *Rosenblatt* proposed the "*Perceptron*", and it can be described as one of the first neural networks (Figure 3.3), single nodded and basically an enhanced version of the *McCulloch-Pitts* binary classifier. The first enhancement came from the addition of an extra input known as bias (b), i.e. translation is performed on the weighted sum defined in equation 3.1 which becomes:

$$y = f\left(\sum_{i=1}^{n} \omega_i \times x_i + b\right) \tag{3.2}$$

With $b = 1$. The second improvement came with the use of *Perceptron* learning rule (equations 3.3 and 3.4) to update the weights that were fixed (constant) in *McCulloch-Pitts* model.

$$\omega_i = \omega_i + \Delta\omega_i \tag{3.3}$$

$$\Delta\omega_i = \eta(t_i - y_i) \tag{3.4}$$

where η is the learning factor (a constant between 0 and 1), which is then multiplied by the difference between the target output and the obtained (predicted) one.

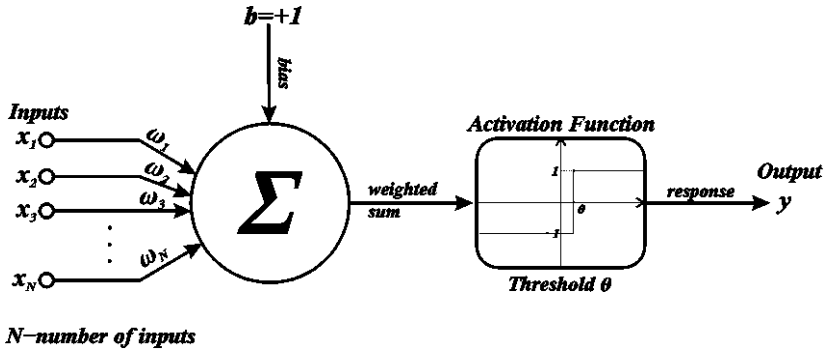


Figure 3.3 - Perceptron network model.

So now generalizing equation 3.2 for an artificial neural network with several consecutive nodes (neurons), where the output of each artificial neuron (node) is evaluated by the chosen activation function f:

$$y_j = f_j\left(\sum_{i=1}^{n} \omega_{ij} \times x_i + b_j\right), j \in [0, m] \tag{3.5}$$

where n is the number of neurons in layer $k-1$ and m is the number of neurons in layer k; y_j is the output for neuron j, x_i and b_j are input signals; and finally ω_{ij} is the weight of the synaptic connection between neurons i and j.

Another milestone in ANN history (1960) was the introduction of Adaline (adaptive linear neuron), illustrated in Figure 3.4, and Madaline (Multilayered Adaline) networks by *Widrow* and *Hoff* which were designed based on the delta rule.

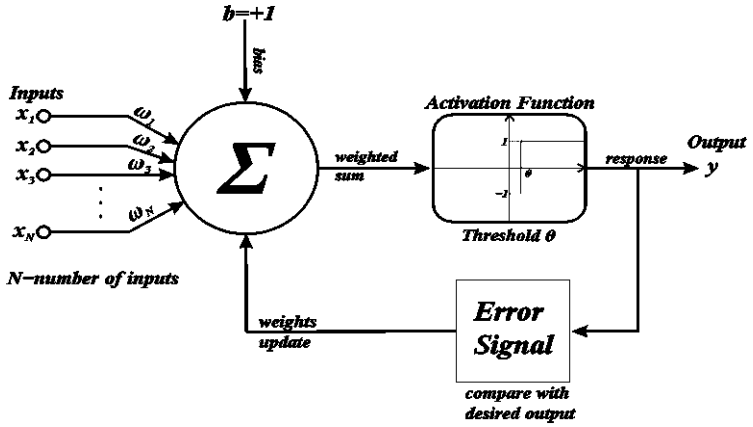


Figure 3.4 - Adaline network model.

Madaline network (Figure 3.5) is just a combination of Adalines in a multi-layer configuration, where the outputs of some become the inputs for others.

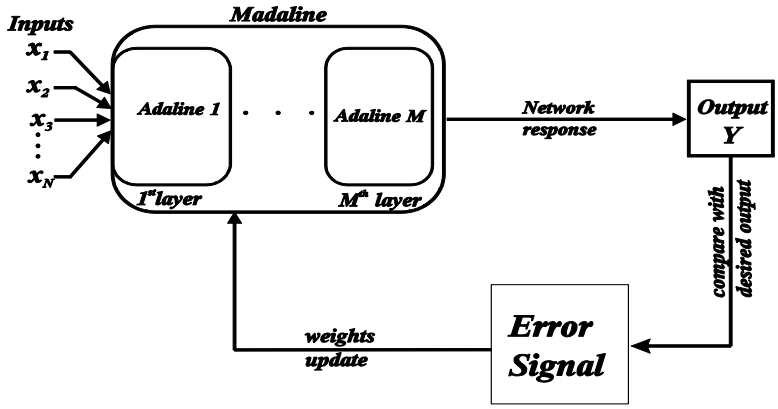


Figure 3.5 - Madaline network model.

So the key building blocks in a ANN are its architecture (neurons arrangement), learning process and its activation function, as illustrated in Figure 3.6.

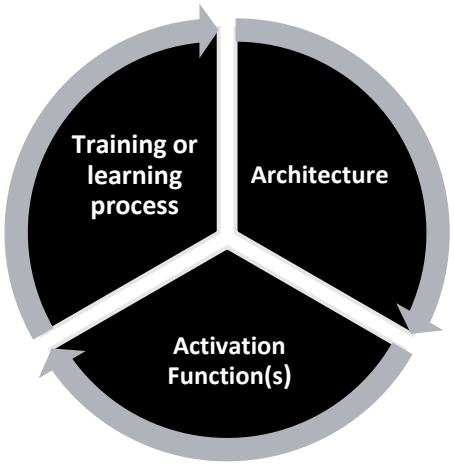


Figure 3.6 -ANN key building blocks.

3.2 Neural Network learning process (algorithms)

A crucial aspect in a neural networks is his ability to "learn", the same is accomplished with an iterative learning/training process, making adjustments on the network weights. The process is interrupted when the network reaches an acceptable solution. We can divide network training approaches in 3 types:

Supervised training/learning: This type of assisted learning (with a "teacher") tests the network with a series of sample inputs and then compares the output of the network with the desired response (target), this learning scheme is illustrated in Figure 3.7.

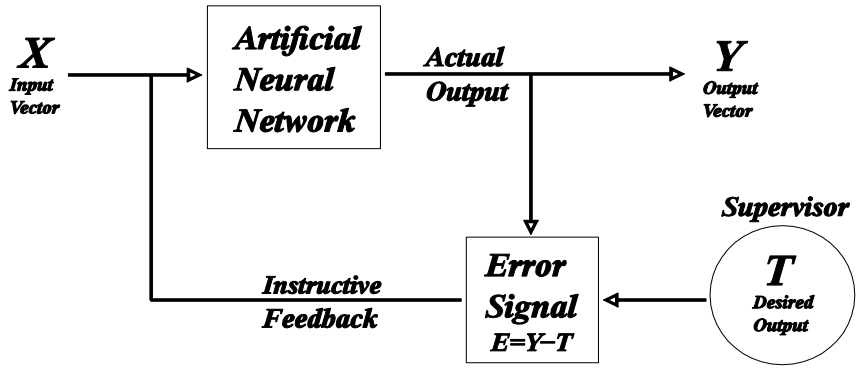


Figure 3.7 - Supervised Training scheme.

So when implementing this method a training pair most be formed with the input vector along with is corresponding target vector (desired output), then according to a learning rule, weights are updated (error dependent). This functionalities makes it ideal for pattern association or classifier (labeling) tasks. Regarding supervised training another distinction can be made:

i. Offline/Batch learning The ANN is exposed to the all set of input patterns at once (in one step), therefore the weights are updated at once too. Input set is static, so a faster learning period is achieved, but with costs to the network which is usually less prepared when confronted with a more diverse "environment".

ii. Online learning: Following this learning approach a variety of input samples is feed to the network continuously (online) and the network weights are changed for every "new" training pattern presented. Despite a larger learning period, a more suitable and embracing ANN is obtained.

Unsupervised training/learning: unlike the previous type the "teacher" figure doesn't exist, so no need to know beforehand the desired network response, the weights are modified to resemble the input sets. This learning approach is more complex, involves architectures with feedback and self-organizing structures, making it a good option for clustering tasks. (Competitive learning rule is an example of unsupervised learning)

Reinforcement training/learning: Essentially a hybrid version combining the 2 approaches presented above. A "teacher" performs a kind of indirect supervision. The network only receives a feedback suggestion ("reinforcement signal") whether the response is

the desired or not, then it's up to the network itself to improve its performance. It's a good option if insufficient information is available about the target output and therefore not possible to use supervised learning, this learning scheme is illustrated in Figure 3.8.

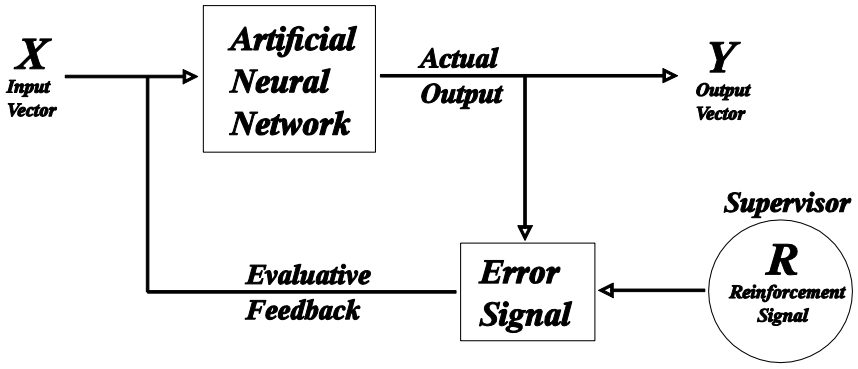


Figure 3.8 - Reinforcement training scheme.

To evaluate the error during network training, several metrics can be employed: mean squared error (MSE), mean relative estimation error (MRE), mean absolute error (MAE), sum of squared residuals (SSR), root mean square error (RMSE).

$$\mathrm{MSE} = \sum_{i=1}^{N} \frac{(a_i - d_i)^2}{N} \tag{3.6}$$

$$\mathrm{MRE} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{(a_i - d_i)}{t_i} \right| \tag{3.7}$$

$$\mathrm{MAE} = \frac{1}{N} \sum_{i=1}^{N} (a_i - d_i) \tag{3.8}$$

$$\mathrm{SSR} = E_T = \frac{1}{2} \sum_{i} (a_i - d_i)^2 \tag{3.9}$$

$$\mathrm{RMSE} = \sqrt[2]{\sum_{i=1}^{N} \frac{(a_i - d_i)^2}{N}} \tag{3.10}$$

where *N* is the number of training samples, a_i is the actual output response (activation value) and d_i the desired output response.

3.2.1 A review on the classical learning rules

One of the oldest and most emblematic learning rules derives from the *Hebb* Theory [65]. From the standpoint of ANNs, *Hebbian* learning states that the weight of a synaptic connection must be adjusted according to the level of synchronization between inputs and outputs, through the expression:

$$\Delta\omega_i = \eta(x_i y) \tag{3.11}$$

As we saw earlier *Rosenblatt* through a sort of trial and error method was making adjustments in the network parameters, this method was upgraded by *Selfridge* (1958) by choosing randomly a direction vector, then adjusted the weights, if the performance didn't improve another direction vector is assigned in a process known as "*Climbing the Mountain*" [65], and is illustrated in Figure 3.9.

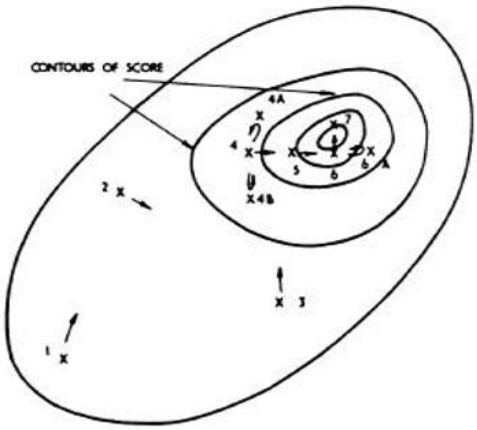


Figure 3.9 -Climbing the Mountain.

In 1960 *Widrow* and *Hoff* assuming that a desired response exists, proposed a gradient search based method (Delta Rule). The network learning is achieved with an iterative process where the gradient of an error function, least mean squared error (LMS), with respect to all the weights is computed and used to adjust the weights. Due to the promising results of the gradient descent methods they were widely spread. Today this learning gradient descent methods are referred as LMS methods and many variations exist.

In 1974 *Werbos* developed the first backpropagation algorithm basically a generalized version of the *Widrow-Hoff* LMS algorithm, allowing a multi-layer configuration, the learning is achieved through the backward propagation of errors, this procedure is combined with LMS method. This work was latter continued by *Widrow*, *Winter* and *Banter* (1987) with an effective algorithm to update all the weights in a network

Another approach is Competitive Learning rule, being a variant of Hebbian learning proposes, it states that: In the presence of similar neurons that differ in the synaptic connections (weights), the rule establishes a competition mechanism that gives the "right" to a neuron (or one per group) to respond to a set of inputs, becoming the output neuron, meaning it is activated, in a competition where the "winner takes all".

This learning process allows that similar patterns are grouped together in clusters, if a matching input had not been found, then a new cluster emerges and makes use of *Kohonen* learning rule to update the weights:

$$\Delta w_{ij} = \begin{cases} \propto (x_j - w_{ij}) & if\ neuron\ i\ wins\ the\ competition \\ 0 & if\ neuron\ i\ loses\ the\ competition \end{cases} \quad (3.12)$$

Grossberg or *Out Star* learning rule has some similarities with Competitive Learning rule. It's a supervised learning technique, where weights are updated as follows:

$$\Delta w_{ij} = \begin{cases} \propto (y_j - w_{ij}) \;\; if\ neuron\ i\ wins\ the\ competition \\ 0 \;\; if\ neuron\ i\ loses\ the\ competition \end{cases} \tag{3.13}$$

In addition to those referred above, we could mention the ones based on stochastic learning, like *Boltzmann* Learning or the ones that use the concept of Memory based Learning.

3.2.2 Generalized Delta Rule (explained)

The chain rule is way to compute the derivative of the composition of two or more functions, and for two generic functions y and u ,can be expressed as:

$$y = f(u)\ and\ u = g(x) \tag{3.14}$$

$$(f \circ g)' = (f' \circ g)g' \tag{3.15}$$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx} \tag{3.16}$$

Then knowing that the neuron i output, is given by its activation value $a_i = f(u_i)$, resuming equation 3.5:

$$u_i(x_i, w_{ij}, b_i) = Node\ response = b_i + \sum_i x_i * w_{ij} \tag{3.17}$$

And the typical activation function used:

$$a_i(u_i) = y_i(Node\ response) = \frac{1}{1 + e^{-u_i}}\ (binary\ sigmoidal) \tag{3.18}$$

To measure the error (E_T) we will use the SSR metric (equation 3.9) once it's important the magnitude of the error and not the signal. For this purpose, the partial derivative of the error as a function of the weights is used:

$$\begin{gathered} \Delta w_{ij} = -\eta \nabla E_T(W_n) = -\eta \left(\frac{\partial E_T}{\partial w_{ij}}\right) = -\eta \left(\frac{\partial E_T}{\partial a_i}\right) \cdot \left(\frac{\partial a_i}{\partial w_{ij}}\right) \Leftrightarrow \\ \Leftrightarrow \Delta w_{ij} = -\eta \left(\frac{\partial E_T}{\partial a_i}\right) \cdot \left(\frac{\partial a_i}{\partial u_i}\right) \cdot \left(\frac{\partial u_i}{\partial w_{ij}}\right) \Leftrightarrow \end{gathered} \tag{3.19}$$

whereas η represents the learning rate, our goal is to minimize the error, then search the weights that converge to this purpose.

Auxiliary Calculations:

$$\begin{gathered} \frac{\partial E_T}{\partial a_i} = -(a_i - t_i) \\ \frac{\partial a_i}{\partial u_i} = -1 * (1 + e^{-u_i})^{-2} * e^{-u_i} * -1 \Leftrightarrow \\ \Leftrightarrow \frac{\partial a_i}{\partial u_i} = \frac{e^{-u_i}}{(1 + e^{-u_i})^2} \Leftrightarrow \frac{\partial a_i}{\partial u_i} = \frac{1}{1 + e^{-u_i}} * \frac{e^{-u_i}}{1 + e^{-u_i}} \Leftrightarrow \end{gathered} \tag{3.20}$$

$$\Leftrightarrow \frac{\partial a_i}{\partial u_i} = \frac{1}{1+e^{-u_i}} * \frac{(1+e^{-u_i})-1}{1+e^{-u_i}} \Leftrightarrow$$

$$\Leftrightarrow \frac{\partial a_i}{\partial u_i} = \frac{1}{1+e^{-u_i}} * \left(1 - \frac{1}{1+e^{-u_i}}\right) \Leftrightarrow$$

$$\Leftrightarrow \frac{\partial a_i}{\partial u_i} = a_i(1-a_i)$$

$$\frac{\partial u_i}{\partial w_{ij}} = \sum_i x_i$$

Resuming equation 3.19, we now can determine the correspondent weight update:

$$\Leftrightarrow \Delta w_{ij} = -\eta \left[-(a_i - t_i) * a_i(1-a_i) * \sum_i x_i \right] \Leftrightarrow$$

$$\Leftrightarrow \Delta w_{ij} = -\eta \left(\delta_i * \sum_i x_i \right) \tag{3.21}$$

$$\Delta w_{ij}(n+1) = -\eta \left(\delta_i * \sum_i x_i \right) + \varphi \Delta w_{ij}(n)$$

where $\delta_i = -(a_i - t_i) * a_i(1-a_i)$.

3.2.3 Other approaches

Some of this traditional methods can behave poorly in large-scale problems, and are sensitive to the choice of some parameters, which can lead to a poor convergence rate [66]. The majority of these methods use conjugate gradient (CG) algorithms, which work based on back propagation algorithm (BP). However, unlike back BP algorithms that adjust weights in the steepest descent direction, CG focuses in the direction that produces a faster convergence, while still minimizing the error (conjugate direction).

So in the BP algorithm, the network weights update mechanism (as expressed in equation 3.19) evaluates the gradient of the total error function and the $\eta > 0$ is a user-selected learning rate parameter which can be constant or variable at each iteration, this mechanism translates into an undesired "zig-zag" approach to the minimum point [67]. To avoid that CG algorithms integrated a direction term D_n, which means that the previous weights adjustment will impact the current one, and can be written as:

$$\Delta W_n = \eta_n D_n \tag{3.22}$$

$$D_{n+1} = -\nabla E_T(W_{n+1}) + \varphi_n D_n \tag{3.23}$$

where φ_n is an iteration-varying parameter and for the first iteration, $n = 0$, the direction $D_0 = -\nabla E_T(W_0)$. The best η_n, has to be found in each iteration (learning epoch) to rapidly minimize the objective function (total error), the same is achieved through a process called "line search" (direction) which is an expansive iterative process, in practice this means that the optimum size step is unreachable and so after few repetitions the size step is determined,

in what is designated as “inexact line search” [67]. Another shortage in CG algorithms is that they often convergence to non-stationary and relatively distant points from the desired minimum, this happens due to the bad approximation capacity especially when weights are far-off the desired minimum and also the incapacity to deal with non-positive definite Hessian matrices.

These shortages lead to the appearance of algorithms such as: Levenberg-Marquardt (LM), scaled conjugate gradient (SCG) or the quasi-Newton BFGS, which are powerful optimization algorithms, with good convergence rates [68].

3.2.4 Scaled Conjugate Gradient Algorithm

SCG is better suited to train large networks (large weights vector), due to its step size scaling mechanism. SCG avoids a time consuming line-search per learning epoch and eliminates the dependence on critical user-defined parameters, requiring less memory usage, making the algorithm faster than others second order algorithms. SCG can serve as learning algorithm in any network as long its weights, net input and transfer functions are differentiable. This algorithm combines the model-trust region approach from LM algorithm with the CG approach [66].

SCG algorithm uses a factor ρ that increases or decreases in each iteration depending on whether the Hessian matrix is positive definite or not, represented by the quantity θ_n. The method is restarted to change the direction vector properly D_n, when a number of repetitions since the last restart (or the beginning) gets equal to the number of free parameters (number of network weights).

The full method implementation can be found here [66], the following steps describe shortly the SCG algorithm implementation [67]:

1. Initialization:
    a. For the first iteration ($n = 0$), define the initial weights W_0., and the scalars $0 < \sigma \leq 10^{-4}, 0 < \rho_0 \leq 10^{-6}$ and $\overline{\rho_0} = 0$;
    b. Set the boolean variable *success* = true;
    c. Set the initial direction vector $D_0 = -\nabla E_T(W_0)$;
2. If *success* = true then calculate second order information:
    a. $\sigma_n = \frac{\sigma}{|D_n|}$;
    b. $S_n = \left(\nabla E_T(W_n + \sigma_n D_n) - \nabla E_T(W_n)\right)/\sigma_n$;
    c. $\theta_n = D_n^T S_n$;
3. Quantity θ_n:
    a. $\theta_n = \theta_n + (\rho_n - \bar{\rho}_n)|D_n|^2$;
4. If $\theta_n \leq 0$ then make the Hessian positive definite definite:
    a. $\bar{\rho}_n = 2\frac{\rho_n - \theta_n}{|D_n|^2}$;
    b. $\theta_n = -\theta_n + \rho_n|D_n|^2$;
    c. $\rho_n = \bar{\rho}_n$;

5. Calculate the step size:
    a. $\xi_n = -D_n^T \nabla E_T(W_n) = D_n^T G_n$;
    b. $\eta_n = \frac{\xi_n}{\theta_n}$;
6. Calculate the comparison parameter, c_n:
    a. $c_n = 2\theta_n(E_T(W_n) - E_T(W_n + \eta_n D_n)/\xi_n^2$;
7. Weight and direction update:
    a. If $c_n \geq 0$ then a successful update can be made:
        i. $W_{n+1} = W_n + \eta_n D_n$;
        ii. $G_{n+1} = -\nabla E_T(W_n)$;
        iii. $\bar{\rho} = 0$;
        iv. *success* = true;
        v. If number of repetitions is achieved then the algorithm is restarted with $D_{n+1} = G_{n+1}$ and $n = 0$;
        vi. Else:
            1. $\varphi_n = (|G_{n+1}|^2 - G_{n+1}^T G_n)/\xi_n$;
            2. $D_{n+1} = G_{n+1} + \varphi_n D_n$;
        vii. If $c_n \geq 0.75$, then reduce the scale parameter:
            1. $\rho_n = 0.25\rho_n$;
        viii. Else
            1. $\rho_n = \bar{\rho}_n$, *success* = false;
8. If $c_k \leq 0.25$, then increase the scale parameter:
    a. $\rho_n = \rho_n \theta_n (1 - c_n)/|D_n|^2$;
9. Repitition
    a. If the steepest descent direction $G_n \neq 0$, set $n = n + 1$ and go back to step 2;
    b. Else terminate and return W_{n+1} as the desired minimum.

3.3 Neural Network Architectures (configurations)

ANNs can be arranged in several different ways and sizes. The most common assembly (network) is to connect neurons into layers and then connect layers itself. Commonly the architecture is used to designate the network. In particular what determines network architecture is the way neurons are distributed through the layers and the established connections between neurons inside/outside the layer. The first differentiation that can be done is between:

Single-Layer Network: Although it has 2 layers, the input layer (the first one) is only to pass and distribute network inputs to next layer. Thus, the only computation layer of neurons (nodes) is the output layer (last one). Each input is connected to every node in the output layer (fully connected net), in such a way that different connections (weights) are responsible for different outputs. The ANN may not have all possible connections (null connection weight), so we could have partially connected network.

Multi-Layer Networks: To achieve more advanced computational proficiencies, a larger and complex structure is required. With at least 1 hidden layer distinguishing itself from the single-layer network. In this multi-layer structure (Figure 3.10), the input nodes pass the

information to the units in the first hidden layer, then the outputs from the first hidden layer are passed to the next layer, and so on.

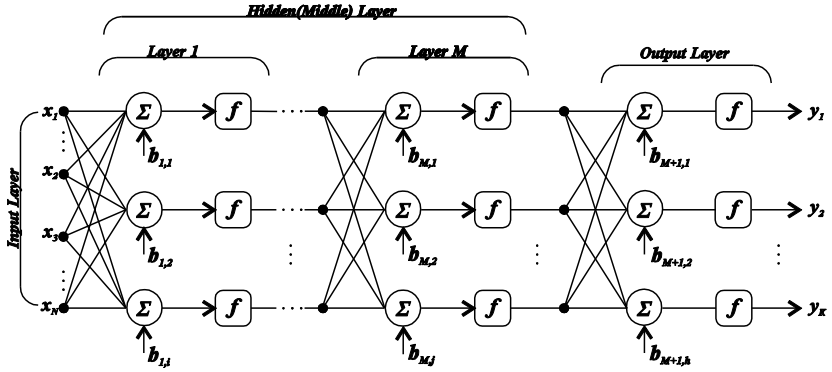


Figure 3.10 - Multiple Layer Neural Network.

Synthesizing the task of each layer showed in Figure 3.10:

i. Input layer: where the input sets are presented to the network.
ii. Hidden layer(s): basically the processing unit of an ANN, and where features of the input data are extracted and learned.
iii. Output layer: where the ANN response is completed and available.

The architectures examined so far, fall into the category of feed forward neural networks (FFNN). The distinctive feature is that signals flow through the layers from the input units to the output ones exclusively in a forward direction (without feedback). Often, we refer to these as multi-layer perceptron (MLP) networks.

Competitive Network (CN): Similar to a feed forward net but with connections between the nodes in the output layer (typically negative weights), triggering competition between them to represent properly the input patterns. The established connections can only be formed between neighboring (closest) nodes or extend to all nodes in the output layer (locally fully connected).

Recurrent Network (RN): Following another approach this architecture establishes connections between nodes to form loops, allowing "information" to persist in the network. Unlike FFN which are organized in layers and information flows unidirectionally trough them, in RN we may have undirected cycles therefore many different connectivity patterns, so the nodes are even allowed to be connected to themselves. This creates an internal state for the network ("limited form of memory"), tackling the problem of ANN learning sequences of information. This means that the response to the current input depends on the previous ones.

The simplest variation is perhaps the fully recurrent network where all nodes are connected to each other, and they all work as outputs and inputs.

Other characteristic examples of recurrent networks are Elman Networks where the feedback ("copy back" process) is established between the hidden layer and one additional layer, called the context layer. Jordan Networks are similar but the feedback is made from the output layer to the context layer, as represented in Figure 3.11.

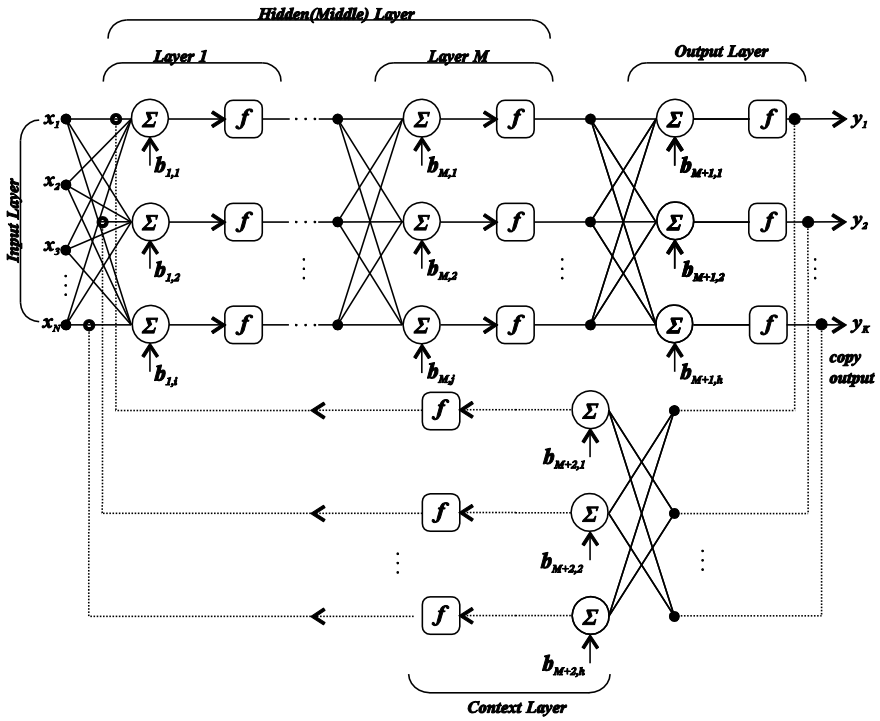


Figure 3.11 - Jordan (recurrent) Network.

Most common network architectures:

- ↳ Single/Multi-Layer Perceptron Network(FFNN);
- ↳ Associative Memory Network (FFNN);
- ↳ Competitive Network (CN);
- ↳ Fully Recurrent Network (RN);
- ↳ Simple Recurrent Network (RN);
- ↳ Jordan Network (RN);
- ↳ Elman Network (RN);
- ↳ Time Lagged Feedforward Network (FFNN);
- ↳ Radial Basis Function Network (FFNN);
- ↳ Hopfield Network (RN).

3.4 Activation Functions

Activation Functions are an important feature of an ANN and are used to compute the neuron response (activation value), the activation function receives as input, the shifted weighted sum and then scales the magnitude of the output in the range from of $[0,1]$ or alternately $[-1,1]$.

The behavior of the activation function describes the characteristics of an artificial neuron. In multi-layer networks, nodes in the same layer have the same activation function, typically non-linear ones, sometimes it's interesting to have hybrid structures selecting different types of activation functions for different layers.

3.4.1 Commonly used activation functions

Since the majority of the real models have non-linear characteristics therefore nonlinear transfer functions are the most used [69]. However in the literature linear activation functions are also mentioned.

Starting by the linear/piecewise linear functions, we can mention the Identity function, *Heaviside*/Binary step function (Figure 3.12), Bipolar step function, Rectified Linear Unit (ReLU) or Ramp Function, Absolute function.

$$f(x) = \begin{cases} 1 & if\ f(x) \geq \theta \\ 0 & if\ f(x) < \theta \end{cases} \tag{3.24}$$

where θ is the threshold value.

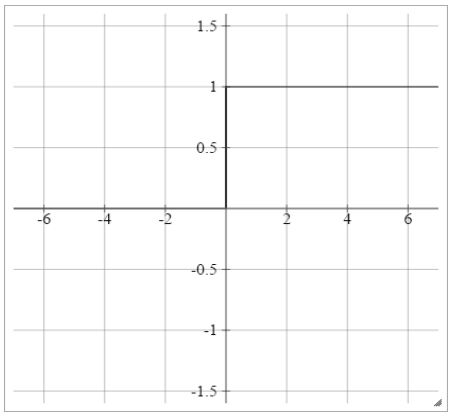


Figure 3.12 - Heaviside Step Function (σ=0).

Then in the nonlinear section, most of them are based in the hyperbolic and logistic functions. More often sigmoidal functions (binary and bipolar) are chosen. With an S shape they constitute a valid option and given their differentiability it allows the adoption of LMS based training algorithms (gradient descent).

Binary Sigmoidal/Sigmoid, also called Log-Sigmoid or Logistic Function (Figure 3.13), it's basically a nonlinear version of the binary step which translates into the addition of a region of uncertainty.

$$f(x) = \frac{1}{1 + \mathrm{e}^{-\sigma x}} \tag{3.25}$$

where σ is referred as steepness parameter, which as measure of how steep is the slope.

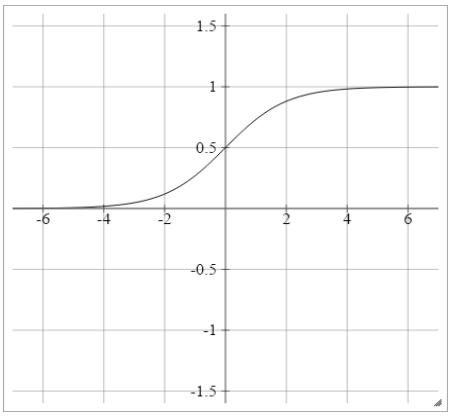


Figure 3.13 - Sigmoid function (σ=1)

Sigmoid function in the same way as the linear Heaviside function it has a bipolar version (Bipolar Sigmoidal Function), with a range extending from $[-1,1]$. This function is illustrated in Figure 3.14.

$$g(x) = \frac{1 - \mathrm{e}^{-\sigma x}}{1 + \mathrm{e}^{-\sigma x}} = \tanh\left({}^{\sigma x}/_{2}\right) \tag{3.26}$$

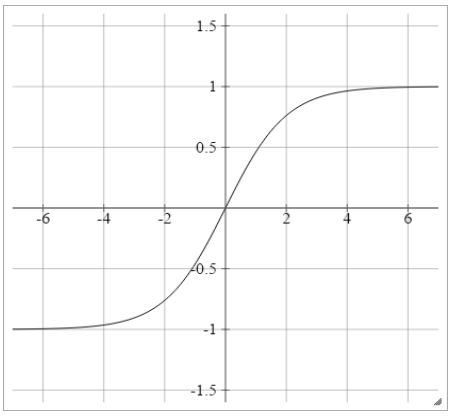


Figure 3.14 - Bipolar Sigmoid function (σ=1).

Other examples of non-linear activation functions are: Tan-Sigmoid, Complementary log-log, Smooth Rectifier, Probit, Logit, Softmax and Modified ReLu's functions, etc.

3.5 Other Considerations

Other important factors concerning neural network's performance, are a proper weights initialization and the choice of a suitable stopping criterion, both have significant impact in training time and results accuracy.

3.5.1 Weights initialization

With regard to weights initialization procedure, which have a great influence on whether the network reaches its global minima and if so how quickly it converges (error in supervised training). To ensure a proper initialization, the *Nguyen Widrow* algorithm [70] can be implemented, which is designed to improve the learning ability with a more reliable

initialization of weights, other alternatives exist as the purely random distribution of the weights. The *Nguyen Widrow* algorithm is implemented as follows:

First, we determine a scale factor, where n is the number of input units and p is the number of hidden units.

$$\beta = 0.7\sqrt[n]{p} \tag{3.27}$$

Then for all the connections between layers ($for\ i = 1{:}n\ \wedge\ j = 1{:}p$), the initial weights are randomly assigned $v_{ij}(old)$, according to the following equation:

$$v_{ij}(old) = \varepsilon, \text{ where } \varepsilon \in [-1,1] \tag{3.28}$$

And then we set the bias as random numbers between $-\beta$ and β. With these steps finished the weights are rewritten as:

$$v_{ij} = \frac{\beta v_{ij}(old)}{\left\| v_{ij}(old) \right\|} \tag{3.29}$$

3.5.2 Training Time

As far as the training (learning) time concerns, it's necessary to understand that the necessary number of epochs to complete the training is a major factor. This sensitive issue is very relevant, because if it is true that the network must train in order to effectively learn, it's also true that we don't want to extend training too much and fall in a situation of overfitting, where the network has no ability to generalize and as a consequence will try to always display the same outputs.

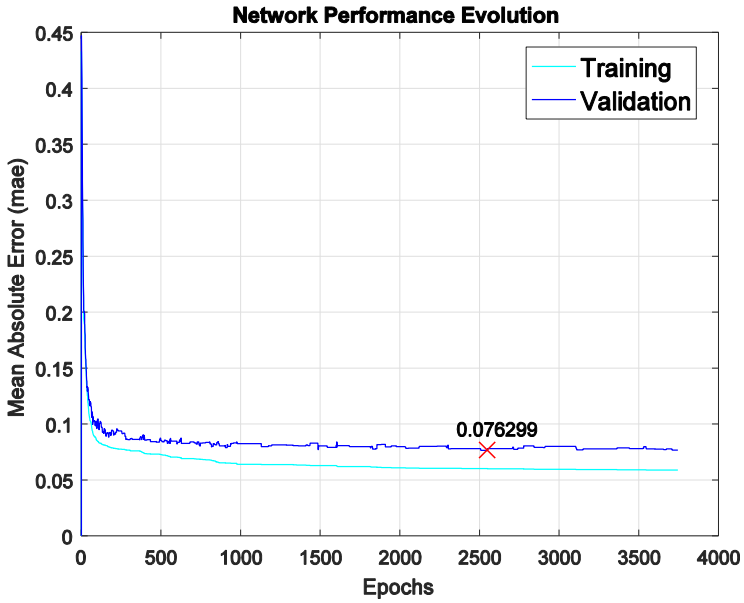


Figure 3.15 - Network performance illustration (Training/Validation error evolution).

The most common stopping criterion is a maximum number of iterations, however more metrics related with the network performance (supervised training) can be used.

A good practice is to stop the training when the ANN presents a good generalization ability together with a tolerable error accuracy. To achieve this a performance based criterion can be implemented, as illustrated in the figure above, where marked with a red cross, is the point where the best validation performance was recorded, then after a predefined number of epochs without improvement the network stops the training process.

Other difficulty, with some significance when working with ANN's is the number of layers (hidden layers) as well as the number of neurons assigned to each hidden layer definition [71]–[73]. The state of the art in this subject presents some procedures to guide the process, and they can be grouped into four categories, trial and error, heuristics, exhaustive search and pruning and constructive algorithms. In all of them there is a direct connection with the input layer size (number of inputs).

3.5.3 Local minima problem

Once backpropagation based learning methods are essentially local optimization methods, sometimes it's possible to encounter a local minima problem, similar to the one shown in the figure below, where the learning process was trapped in some local minima, therefore not reaching the global minima.

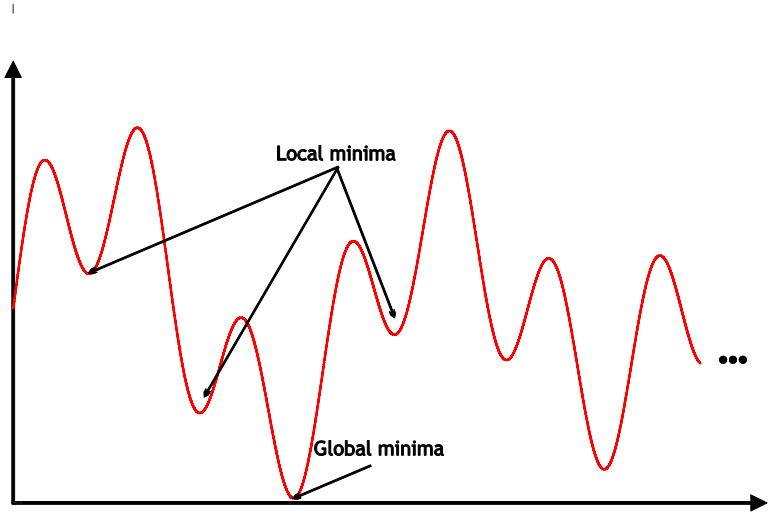


Figure 3.16 - Local minima problem.

A way to lessen this problem when using gradient based methods is to restart the training process, guaranteeing that different starting positions (ANN initial weights) are tested, or just by introducing random disturbances in the weighs during the training process.

Another approach which have been efficaciously employed is to combine classical techniques with global optimization methods like metaheuristic algorithms to properly train ANNs.

Chapter IV

4 Optimization Algorithms

The task of searching for an optimal state as always been a discipline with a lot of interest given its scope, this chapter will especially focus on metaheuristic methods, these threat problems as black-box-procedures. In the last 15 years a vast literature has been proposed, where evolutionary algorithms, swarm intelligence and nature-inspired algorithms are common denominators.

4.1 Metaheuristic Algorithms

This type of stochastic algorithms, are "high level" heuristics, with the purpose of finding acceptable solutions for hard optimization problems, without guarantying that a globally optimal solution can be found but rather feasible ones. The same is achieved evaluating an objective function, then these algorithms seek a balance between diversification (sweep the search space) and intensification (convergence to the optimal candidate solution). Metaheuristic algorithms can be classified as single-solution when the focus is in a single candidate solution and its trajectory, or population-based where the focus is the collective behavior of the swarm/population, these last are better suited to ensure the scan of the search and avoid local minima problems. Other classification can be made regarding the type of search strategy, local search vs. global search algorithms.

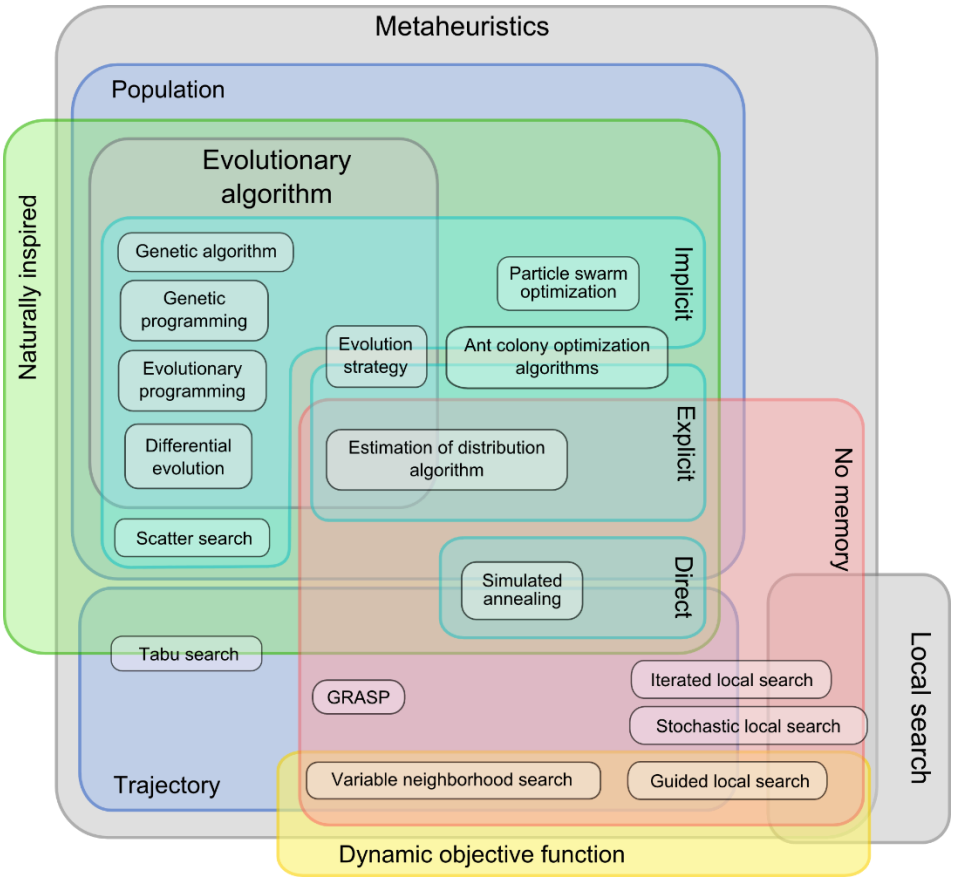


Figure 4.1 - Metaheuristic algorithms taxonomy [74].

A factor to take into consideration when updating solutions on each dimension is that we have to guarantee that they stay inside the domain search space, for that purpose we use hyperbolic confinement described in [75]. This method adjusts the velocities (v_i^t) to make sure that the obtained positions x_i^t, stay inside the domain search space.

$$If\ v_i^t > 0\ \ then\ v_i^t = \frac{v_i^t}{1 + \left|\frac{v_i^t}{x_{max} - x_i^t}\right|} \tag{4.1}$$

$$If\ v_i^t < 0\ \ then\ v_i^t = \frac{v_i^{t+1}}{1 + \left|\frac{v_i^t}{-x_{max} + x_i^t}\right|} \tag{4.2}$$

Another time and performance wise related question in these meta-heuristic algorithms, is the choice of a stopping criterion, as for example a determined number of iterations, a tolerance in the fitness, a number of particles in a certain radius, etc.

4.2 Particle Swarm Optimization

Presented in 1995 [76] the Particle Swarm Optimization (PSO) algorithm was developed by studying social behavior of birds. In the algorithm, the individuals called particles travel on a multidimensional search space.

4.2.1 Implementation

First, particle (solutions) are randomly initialized using the following equation:

$$x_i^0 = lb + (ub - lb) * \varepsilon \ , i \in [1, N] \tag{4.3}$$

where lb and ub are the lower and upper bounds (domain space limits) for the particle i, ε is a random number $\in [-1,1]$ and N is the population size.

So, each particle represents a potential solution, the particle travels in the search domain space and is attracted to the position of the current best g_{best} (achieved by any particle), and to its own recorded best position p_{best}. The evaluated particles, try to achieve the right balance between a global and local search, as illustrated in Figure 4.1.

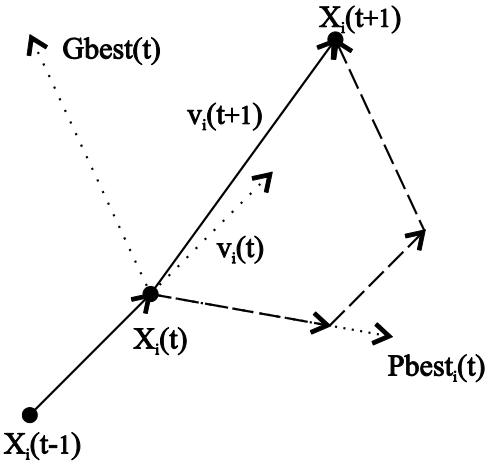


Figure 4.2 - Particle position evolution (graphical overview).

Particles update they're velocity (v_i^t) and positions (x_i^t) according to the following equations:

$$v_i^t = v_i^{t-1} + \alpha\varepsilon_1 \odot \left[p_{i(best)} - x_i^{t-1}\right] + \beta\varepsilon_2 \odot \left[g_{(best)} - x_i^{t-1}\right] \tag{4.4}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{4.5}$$

where $\varepsilon_1$and ε_2 are random numbers comprehended in the interval $[0,1]$, α and β are learning factors (acceleration factors) and usually set: $\alpha = \beta = 2$. The symbol $\odot$ denotes elementwise vector multiplication. To ensure convergence the condition $\alpha + \beta \leq 4$ must be respected.

An important aspect for the algorithm's performance is its topology, the way the particles communicate with each other. Among the many topologies in the literature, the most common are:

i. Ring topology each particle only communicates with a certain k number of adjacent neighbors, Figure 4.3 (a);
ii. Star topology every particle communicates with one another, Figure 4.3 (b);
iii. In this topology, a central particle performs the connection between her and the other isolated particles Figure 4.3 (c);

iv. Von Neumann topology particles are connected such that particles in one extremity communicate with particles in the opposite extremity, Figure 4.3 (d);

v. Cluster topology inside each cluster the particles communicate with one another and certain particles communicate between clusters, Figure 4.3 (e).

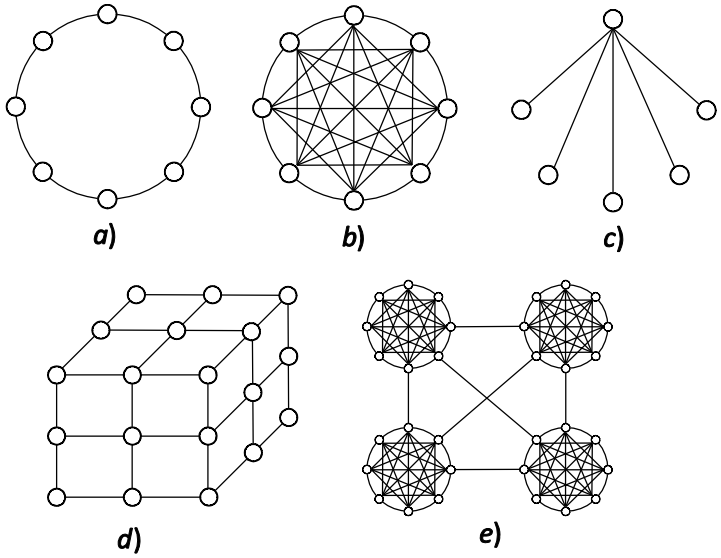


Figure 4.3 - Graphical representation of the most common PSO topologies.

Particles positions on each dimension are clamped to the domain search space, for that reason as we saw hyperbolic confinement is used (equations 4.1 and 4.2). This iterative process continues, in the search for the best fitness values (minimizing the objective function), until the stopping criterion is reached.

Many variants of this standard algorithm have been developed in order to improve performance, including for example the Modified Particle Swarm Optimization [77], which introduces an inertia weight $\theta(t)$. This inertia weight improves the control over the particle movement, forcing the particle to travel at different velocities in the search space. In other variants the iteration count has been also introduced in many ways, for example determining an inertia weight given the number of iterations in the PSO variant called Linearly Decreasing Weight PSO (LDW-PSO) [78], or determining the acceleration parameters $\boldsymbol{\alpha}$ and β given the iteration count in the PSO variant called Particle Swarm Optimization with Time Varying Acceleration Coefficients (PSO-TVAC) [79]. In the case of the PSO variant called Supervisor Student Model in Particle Swarm Optimization (SSM-PSO) [80], the particles are prevented from flying out of a search space, without checking the validity of the position at each iteration, by introducing a variable called momentum factor. In the case of the variant Parameter Free PSO (pf-PSO) [81], the PSO parameters are removed and the local best ($Pbest$) and global best ($Gbest$) positions are used to update the position of the particles directly in the position update equation. Concerning the convergence, the use of the constriction coefficient can be viewed as a recommendation to make the particles travel in "smaller steps" and eventually converge. The convergence mentioned here obviously implies that the particle's velocity reaches zero. But the correspondent position is not necessarily the

wanted convergence point [82], [83]. There are also PSO variants that avoid premature convergence before finding the true local minimum (guaranteed convergence PSO) [84]–[86]. The main idea is to introduce an additional particle, which searches the region around the current global best position. In that manner, the current global best particle is treated also as a member of the swarm (e.g. particle) [85].

In this work, the choice fell on the Accelerated PSO, which introduces the inertia function $\theta(t)$ in equation 4.4, so the velocity update is computed as:

$$v_i^t = \theta(t)\, v_i^{t-1} + \alpha\varepsilon_1 \odot \left[p_{i(best)} - x_i^{t-1}\right] + \beta\varepsilon_2 \odot \left[g_{(best)} - x_i^{t-1}\right] \tag{4.6}$$

This modification improves the control over the particle movement, forcing the particle to travel more or less in the search space. To better understand its behavior, the PSO pseudo-code is provided:

1. Initialize parameters:
    a. Population size N (swarm size);
    b. Number of dimensions (objective function variables);
    c. Lower and Upper bound vectors (to define admissible search space);
    d. Learning factors α and β;
    e. Inertia function as a constant;
2. Generate the initial locations for the particle swarm (x_i) based on equation 4.3;
3. Define initial fitness values, where $G(x_1, x_2, \dots, x_n)$ is the objective function:
    a. $G\left(p_{i(best)}\right) = \infty,\ i \in [1, N]$;
    b. $G\left(g_{(best)}\right) = \infty$;
4. While (! stopping criterion):
    a. For each particle, calculate fitness value;
    b. If $G\left(p_{i(best)}\right) > G(x_i^t)$ or by words when the current fitness value is better than the previous fitness value recorded;
        i. Then $p_{i(best)} = x_i^t$, so we set the current value as the new local best;
    c. Choose the particle with the best fitness value of all the particles evaluated and obtain the new global best $g_{(best)}$;
    d. For each particle, calculate particle velocity according to equation 4.6
    e. Update particle position according equation 4.5;
5. If the stop criterion is reached then jump to next step, otherwise return to step 4 a);
6. Display the best location for the objective function and its fitness.

4.3 Bat Algorithm

Bat Algorithm (BA) was introduced in 2010 [87] and since then it has been tested in an wide-ranging set of benchmark functions and problems. Once it outperforms the traditional PSO and others bio inspirited algorithms [88] this metaheuristic algorithm as received much attention and developments in the last years [89].

As its name indicates the algorithm is based on the bat behavior and in particular on some smaller species and their hunting practices. They make extensive use of their echolocation ability. As known echolocation allows them to detect preys, avoid obstacles and find their caves in the dark. These bats emit a "high/bulky" sound wave and then capture the echo which highlights/reflects the surroundings. The bandwidth varies, most of them uses short signal frequencies. These bats use the delay time between the emission and detection and volume variations in echo to build a three-dimensional "image" of the surrounding environment.

4.3.1 Implementation

First each bat (possible solution) is assigned with frequency between $[f_{min}, f_{max}]$ according to the search domain. And $\beta \in [0,1]$ is a random vector obtained from a uniform distribution.

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{4.7}$$

Second, we define the rules of how positions x_i and velocities v_i are updated inside the multidimensional search space domain at each time step t, accordingly:

$$v_i^t = v_i^{t-1} + (x_i^t - x^*)f_i \tag{4.8}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{4.9}$$

where x^* represents the location of the current global best solution (bat), so the new location is updated accordingly (global walk).

A local search procedure is imbedded in the algorithm, as such at certain times a new solution is generated locally through a “random walk” process, computing the equation:

$$x_{new} = x_{old} + \varepsilon A^t \tag{4.10}$$

where ε represents a random number in the interval [-1, 1] and $A^t = A_i^t$ represents the loudness parameter at each time step t. If the bat is closer to the optimal solution and we accept the new locations as better, then we must adjust loudness and pulse emission rate r_i^t, accordingly to the following equations:

$$r_i^t = r_0[1 - \exp(-\gamma t)], \gamma > 0 \tag{4.11}$$

$$A_i^{t+1} = \alpha A_i^t,\ 0 < \alpha < 1 \tag{4.12}$$

where γ and α are constant parameters, and consequently $lim_{t\to\infty} A_i^t = 0$ and $lim_{t\to\infty} r_i^t = r_0$.

So the pseudo-code behind BA functioning is presented below.

1. Initialize parameters:
    a. Population size N (swarm size);
    b. Initial loudness coefficient A_0;
    c. Maximum pulse rate $r_0$the (largest burst);
    d. Minimum and maximum pulse frequency [f_{min}, f_{max}];

e. Finally, the auxiliary constants α and γ;

2. Generate the initial locations for the swarm (x_i);
3. Evaluate the objective function and determine the fitness for every bat $G(x_i)$;
4. Rank the bats and find the best one so far (x^*);
5. While (! stopping criterion):
    a. For each bat;
    b. Generate new solutions according to equations **Erro! A origem da referência não foi encontrada.**, 4.8 and 4.9;
    c. If a random number > r_i:
        1. Place the bat around the best-known position, through a "random walk" process (equation 4.10);
    d. If (random number < $A_i^t \wedge G(x_i^t) < G(x_i^{t-1})$, we accept new solution (bat location), in a random flight process;
    e. If $G(x_i^t) < G(x^*)$ then we replace $G(x^*)$ and we accept x_i^t as the best bat. Afterwards we adjust bat loudness and pulse rate according to equations 4.11 and 4.12;
6. If the stop criterion is reached then jump to next step, otherwise return to step 5 a);
7. Display the best location for the objective function and its fitness.

One important feature when implementing the bat algorithm are the initializations made in step 1 (pseudo-code), the author of the proposed algorithm [87] suggests:

$$\begin{cases} A_i^0 \rightarrow [1,2] \\ r_i^0 \rightarrow [0,1] \\ \alpha = \gamma = 0.9 \\ f_{min} = 0 \end{cases} \quad (4.13)$$

However in an attempt to find the best initial parameter settings the authors in [90] have implemented orthogonal experimental design technique to achieve better initialization practices.

4.4 Cuckoo Search

Cuckoo Search (CS) is a nature-inspired metaheuristic algorithm proposed in 2009 [91]. The cuckoo search is based on the brood parasitic behavior of some cuckoo species. These birds have an aggressive reproduction strategy, laying their eggs in communal nests, and they even remove other eggs to increase the hatching and survival chances.

Some female cuckoo parasites developed the ability to mimic other cuckoo's species eggs, reducing the chance of the host cuckoo to abandon the eggs. The timing in which cuckoos lay their eggs, is also important, normally parasite cuckoos choose the nests where the host as just laid the eggs.

4.4.1 Implementation

Translating the ideas expressed before for the algorithm's heuristic implementation, each egg in a nest represents a solution and every cuckoo can only put an egg in the nest, with the purpose of using new and potentially better solutions, cuckoos will eventually replace previous solutions (eggs). The algorithm could be extended to accommodate multiple eggs in the same nest. Therefore, in a simple way we can describe the algorithm's operation in 3 rules:

i. Every cuckoo lays his egg in a randomly chosen nest;
ii. The best nests, i.e. with the best eggs are used (remain) for the next generations;
iii. The number of available host nests is constant, and the parasite egg detection rate, by the host cuckoo, is given by a probability $p_a \in [0,1]$, i.e. the host can get rid of the egg, or abandon the nest and build a new one.

Despite this all, the common implementation form does not make any distinction between the cuckoo, nest and egg. This algorithm it's based on a combination of a local "random walk" (equation 4.14) and a global one (equation 4.15), controlled by p_a. And the new position is given by:

$$x_i^{t+1} = x_i^t + s \odot H(p_a - \varepsilon) \odot (x_j^t - x_k^t) \tag{4.14}$$

where is the x_i^t current position (solution) at iteration t, x_j^t and x_k^t x_i are two randomly originated solutions (random permutation), $H(u)$ is the Heaviside function (equation 3.24), $\boldsymbol{\varepsilon}$ is a random number given by a random distribution, and s represents the step size. The symbol $\odot$ denotes elementwise vector multiplication.

On the other side, the "global walk" is carried out using "Lévy flights" through the expression:

$$x_i^{t+1} = x_i^t + \alpha L(s,\lambda) \tag{4.15}$$

where $L(s,\lambda) = \lambda\Gamma(\lambda)\sin\left(\frac{\pi\lambda}{2}\right)/\pi\, s^{1+\lambda}$ is the Lévy function, where λ and α are constant factors and $\Gamma(\lambda) = \int_0^\infty \mathrm{t}^{\lambda-1}e^{-t}\,dt$ is the gamma function.

A random direction is provided by steps following Lévy distribution, so the step size s is determined computing the following:

$$s = \frac{u}{|v|^{1/\lambda}} \tag{4.16}$$

with u e v obtained from the normal distributions $u \sim N(0,\sigma_u^2)$ e $v \sim N(0,\sigma_v^2)$.

Typical values for the cuckoo population size are $n \in [15,40]$, the detection rate probability $p_a \in [0.25, 0.5]$, $\lambda \in [1,1.5]$ and $\alpha = 0.01(u_b - l_b)$ making steps not "aggressive", where u_b and l_b are the upper and lower bounds for the solution (position) vector.

Therefore, some of the new solutions are generated by "Lévy flights" around the best solutions already obtained, and this will accelerate local search. However, a substantial part

of new solutions will be widespread, i.e. with distant positions from the best solution, which is good feature to avoid getting trapped in a local minimum. The suitability of new solutions is proportional to the objective function value.

4.5 Optimization algorithms as ANN learning methods

As reviewed in section 3.2 there are a considerable amount of (classical) learning approaches to the ANN's, however the use of optimization (meta-heuristic) algorithms to successfully train ANN's as proven to be a trend, due to the numerous computational advantages and good performance revealed, as the following table exemplifies:

Table 4.1 - ANN training methods.

Reference(s)	*Method/Algorithm*
[15]	Particle Swarm Optimization
[92]	Genetic Algorithms
[93]	Differential Evolution
[94]	Simulated Annealing
[95]	Bat Algorithm
[28]	Cuckoo Search
[96]	Artificial Bee Colony Algorithm

Chapter V

5 Load Forecast

As mentioned on the motivation and goals section, one of the objectives of this dissertation is the development of an effective methodology for load forecasting. Therefore, this chapter presents and details the proposed approach, as well as the main results achieved.

5.1 Proposed Methodology

The flowchart of the proposed methodology is presented in Figure 5.1 this scheme illustrates in a generic way the various steps, starting from the load time series and ending in the 24h load forecast.

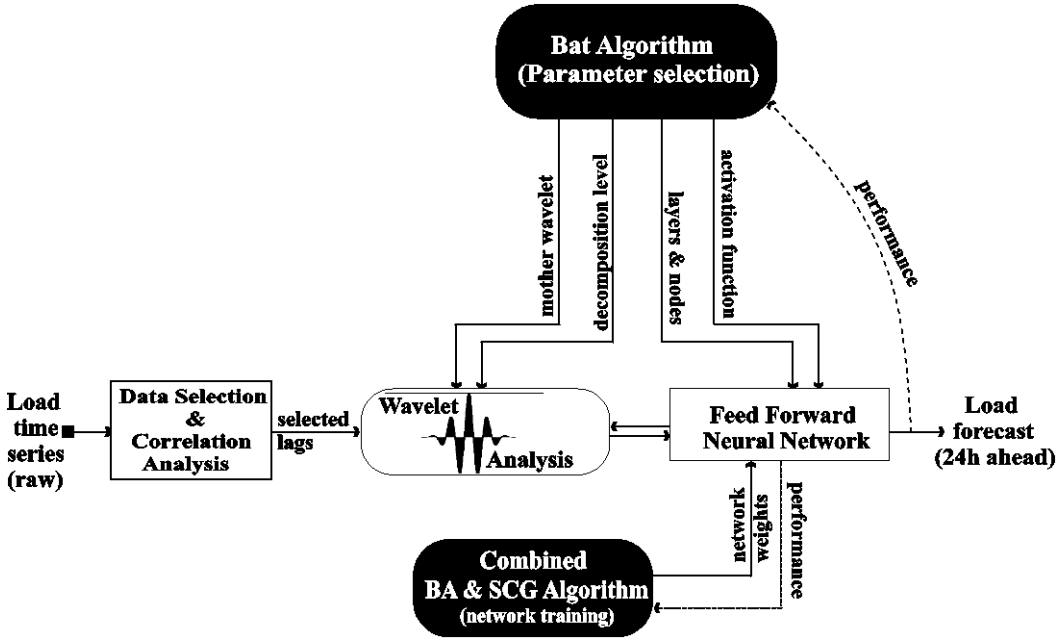


Figure 5.1 - Proposed methodology (Overview)

5.1.1 Input Selection and Features Extraction

The initial task is to handpick the training data for the ANN, serving two objectives: assembling data with information enabling a good forecast, and data of reasonably low size, to avoid excessive time consumption. To serve this purpose, the process is as follows: given a current (selected) day load, find historical days with a similar load profile.

Assuming given repetitive patterns in the load time series, the targets for these similar days should be reasonably close to the current day's target (24h ahead). In addition, load data from recent days, prior to the current day is also included, providing information about load recent behavior. As such, the training set is formed only by the three most similar days and the preceding two days.

In Figure 5.2 this process is illustrated, the first plot shows the current day in blue within the region in grey. This region is delimited by the three most similar days (maximum deviations from the current day). The second plot in Figure 5.2 the correspondent target day (24h ahead) in red, and the region formed by the targets for the similar days in grey.

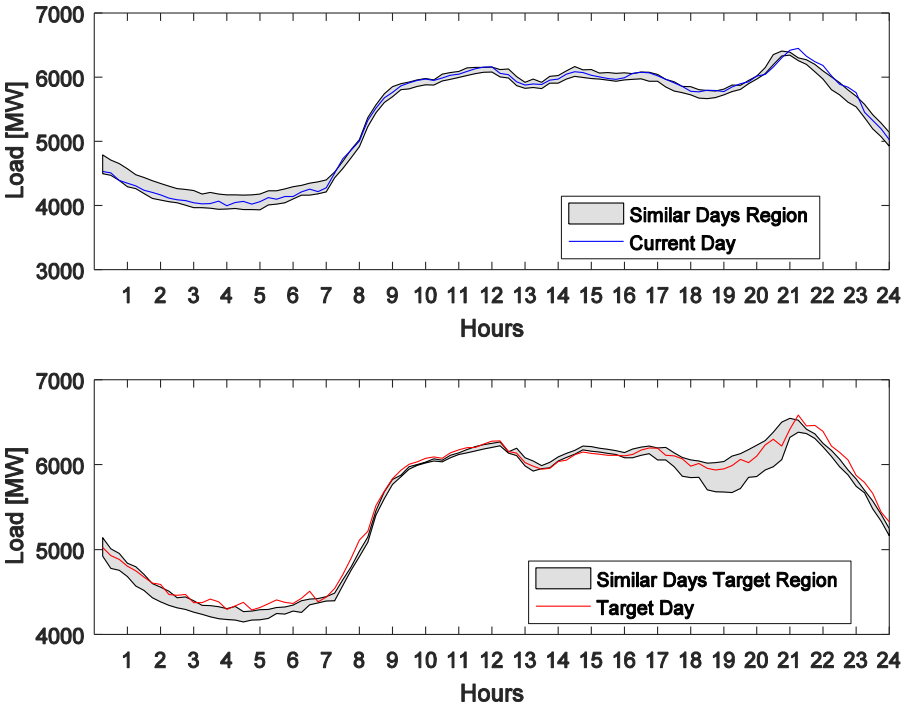


Figure 5.2 - Input selection process

The next step is to define the ANN input vector, to achieve that the most important load values were extracted. To select them properly, a few practices are described in the literature. For example in [23] mutual information (MI) was used, this measure evaluates the mutual dependence between two random variables. Another option derived from the previous one is conditional mutual information (CMI) [13] where MI between two random variables is evaluated when another variable is known.

In this work, as in other load forecast literature, to make the correct assessment of the relevant data series lags for each network training, we determined the partial autocorrelation (PAC), as illustrated in Figure 5.3.

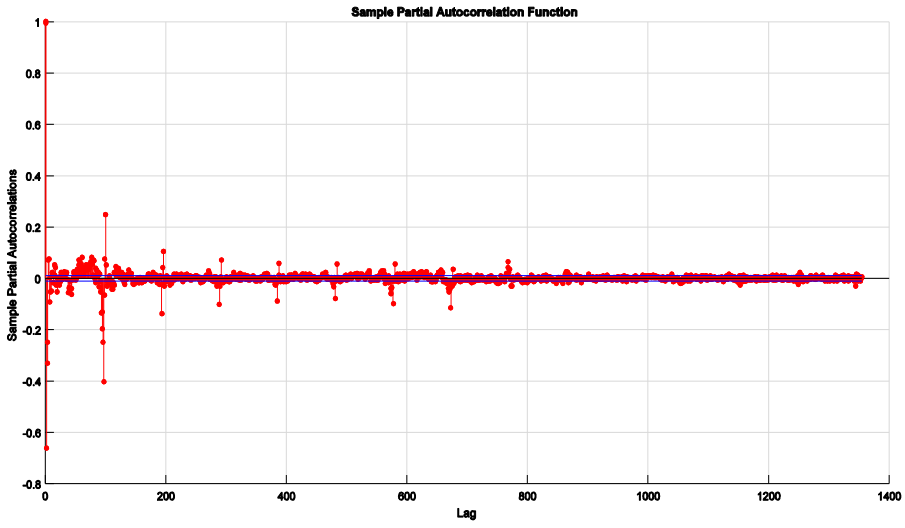


Figure 5.3 - Load time series partial autocorrelation (PACF)

This selection process is performed for each training set, i.e. selected lags may be different.

5.1.2 Mother Wavelet Pre-Selection

When using WT an important decision has to be made: the selection of a suitable mother wavelet, in the particular STLF case, the choice of *Daubechies* family wavelets is almost consensual, with special emphasis on *Daubechies* of order 2 and 4 (db2 and db4) [12], [15], [21], [97]. However, larger temporal data sets, specifically a load time series, can present considerable variability over time. Thus, the best mother (base) wavelet for a certain time interval may not be the most appropriate for others, for example in [97] the authors found that different orders *Daubechies* were best suited for each season of the year. The same reason led the authors in [13] to implement a wavelet-based ensemble scheme composed by eight mother wavelets, sectioning the load data series and finding the mother wavelet that best fits each section. Yet when selecting the eight best candidates, the authors also relied on the consensus about *Daubechies* in selecting half the base wavelets, while the remaining four where picked from the *Coiflets* family, based on experimental results.

So the usual approach to test different ones in the context of the implemented method, thus this experimenting and evaluating task is time consuming [13], and will culminate in only one chosen and all the remaining abandoned. Alternatively, we could rely on qualitative methods, which are basically supported by past experiences. However, the time gains achieved may not compensate the risk of choosing a non-suited mother wavelet, therefore not allowing a good extraction of features from our time series.

To speed up the process, we used minimum Shannon entropy criterion, which revealed us the mother wavelets with higher energy concentrations, therefore with low entropy magnitudes, enabling the effective extraction of features by the wavelet decomposition. Table 5.1 shows the best orthogonal mother wavelets selected for both case studies, and as expected there is a prevalence of *Daubechies* mother wavelets.

Table 5.1 - Wavelet entropy results (STLF).

Mother Wavelet Pre-Selection			
Case Study I	db2, db3, db5, db7, db8, sym5, sym7, bior1.5 and coif3.	**Case Study II**	db4, db6, db7, sym4, coif3, rbio1.3, rbio1.5, bior3.1 and bior3.3

5.1.3 ANN Training

An important task when using ANNs is its training process, i.e. extract existing dependencies and relations from the input training data, without losing its generalization capacity (over-fitting). However, potential problems, such as sensitivity to the initial positioning, getting trapped in a local minimum, and slowness may affect ANN performance [15].

Avoiding these issues allows ANNs to be a reliable tool for STLF. In this paper, the evolutionary BA was used in combination with the SCG algorithm as a training algorithm for the ANN. The two methods were combined (hybridized) because the BA alone can suffer from stagnation [98] and premature convergence in high dimensional problems [99].

To avoid these problems, BA and SCG algorithms were employed together in an offline and supervised learning process. The solutions ("bats") correspond to the network weights, and the performance function is given by equation 3.6. At the end of each BA iteration, the best "bat" is evaluated by the SCG algorithm, which by default (initially) only performs one training epoch. If after this process the error does not decrease, then in next iteration of the SCG algorithm, the number of learning epochs is double. The training process is completed when the chosen stopping criterion is reached. This involves a minimum number of iterations, and a threshold on the training and validation performance.

In addition, to tackle the initial guess problem, the initial weights ("bats") were positioned according to the well referenced "*Nguyen-Widrow*" algorithm [70]. We also used a cross-validation technique [100], to prevent overfitting and ameliorate the initial weights guess problem. Therefore, the network performance is evaluated k_{fold} times, alternating the validation subset and training subsets $(k_{fold} - 1$ subsets$)$ in a circular form.

After determining the validation and training subsets, these were normalized between -1 and 1, using the normalization method *Min-Max* expressed by equation 5.1, guaranteeing that the dataset is stationary, i.e. mean and variance are constant.

$$x_{map} = \left(\max x_{map} - \min x_{map}\right) * \frac{x - \min x_{old}}{\max x_{old} - \min x_{old}} + \min x_{map} \tag{5.1}$$

where x is the actual value, x_{map} is the mapped (normalized) value, $\max x_{map} = 1$ and $\min x_{map} = -1$ are the new bounds and $\max x_{old}$ and $\min x_{old}$ are the old ones.

5.1.4 ANN Architecture and Wavelet Analysis tuning

As mentioned earlier, there is a set of parameters involving neural networks architecture and wavelet analysis for which there is no consensus concerning optimal arrangement. Thereby an optimization algorithm can be implemented to find the best settings, e.g. [95]. In wavelet analysis, those parameters concern the definition of the appropriate decomposition level and the choice of the best mother wavelet, and, in ANNs, the choice of the optimal architecture (number of layers and number of hidden neurons).

To carry out this optimization a BA was used, operating over four dimensions, i.e. optimizing four parameters within an acceptable framework. First, it chooses the mother wavelet from the best candidates, presented in Table 5.1. Second, it determines the best decomposition level for the wavelet analysis, from two to five levels, since more levels of decomposition would lead to unnecessary complexity for the ANN. Third, it decides between one or two hidden layers, although one hidden layer is almost always sufficient, it can lead to a huge number of hidden neurons, and therefore a network with two-hidden layers may

enable a more reasonable configuration [101]. Each layer contains up to 99 neurons. Fourth and last, it decides upon the activation function on the output layer, to maintain the network's ability to represent linear sections. A linear activation function (*purelin*) is preferable over the common *tansig* [15]. The codification of the domain search space for the combined SCG-BA is illustrated in Figure 5.4.

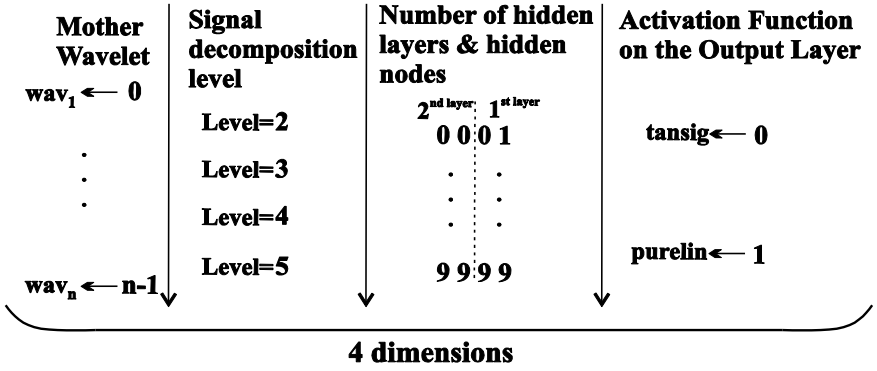


Figure 5.4 - Codification of the domain search space

One important feature when implementing the bat algorithm are its initial settings: loudness A_0, pulse emission rate r_0 maximum frequency f_{max} (frequency upper bound) and the local search parameter ε. Since there isn't an optimal paradigm and given that the framework where the BA is employed is relevant, the PSO algorithm was employed to choose the most fitted initial settings.

Optimization was completed after 15 runs. The results are presented in the form of histograms for each dimension, Figure 5.5. The four histograms reveal that, with regard to local search parameters, the best value approaches 0.002; with respect to loudness, and as suggested by the author in [87], the conventional values are between 1.5 and 2; the pulse emission rate parameter should be less than 0.4, which means it favors in all iterations the "random walk" mechanism. Lastly, regarding the maximum frequency, there is a greater distribution in all classes compared with the dimensions referred previously. However, values close to 10 proved to be a good choice.

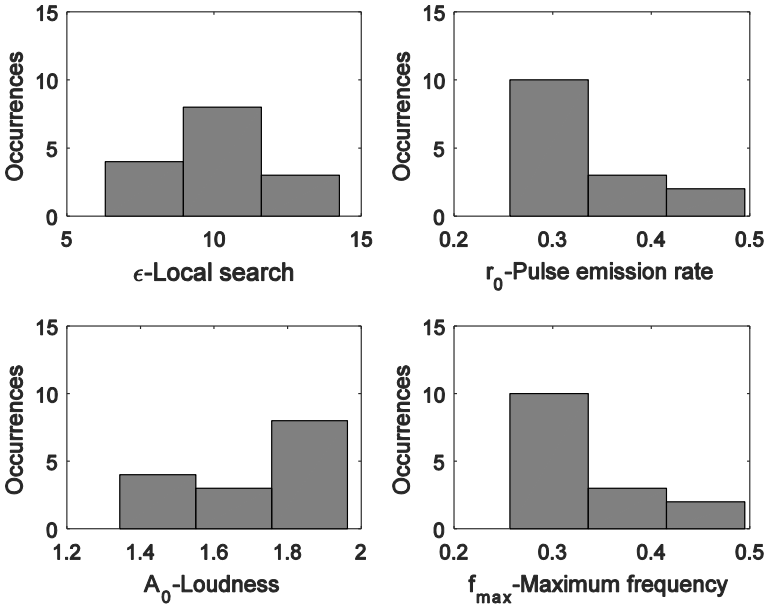


Figure 5.5 - PSO results for BA initial settings tuning.

5.1.5 Forecasting accuracy measure

This methodology was implemented and tested in two different scenarios (case studies), and to measure method's prediction capacity we used the mean MAPE, which is computed as follows:

$$\text{MAPE}\ (\%)\ = \frac{100}{N}\sum_{t=1}^{N}\left|\frac{L_t - \hat{L}_t}{L_t}\right| \tag{5.2}$$

where L_t and $\hat{L}_t$ are the actual and the forecasted load values at t hour respectively and $N = 24$ is the of hourly samples in a day.

5.2 Case Study I

In the first case study, the proposed method was tested using the Portugal load data series (as quarter-hourly data set) from 1 September 2015 to 31 August 2016. Figure 5.6 shows the data series for every month and hour of the day.

According to Regulation (EC) no. 714/2009 of the European Parliament and of the Council of 13 July 2009, the transparency information on the market of electrical energy must be made available free of charge and accessible. So the data was provided by the Portuguese National Electricity Transmission Grid (REN) and is available in [102].

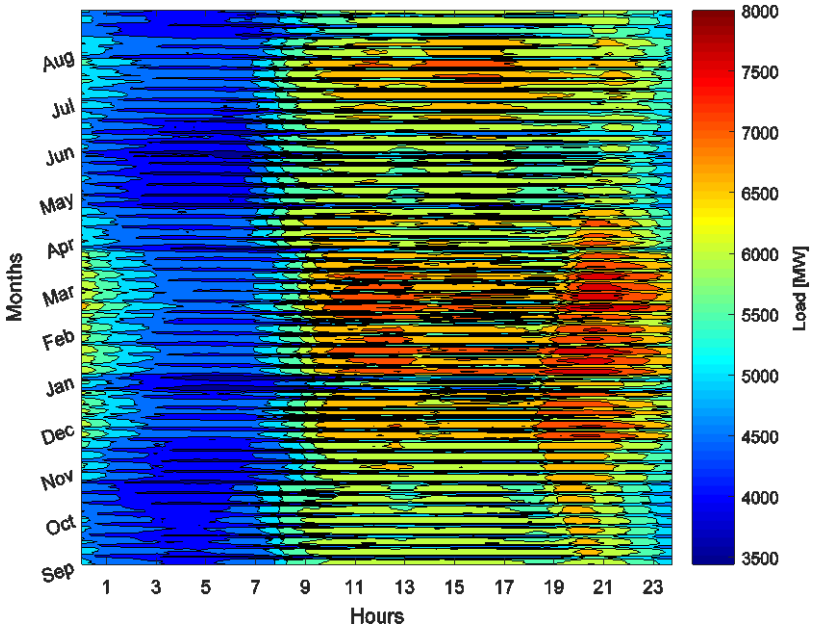


Figure 5.6 - Load [MW] from 1 September 2015 to 31 August 2016 (PT)

To properly analyze and depict the data series, as well as to confirm some of the characteristics and seasonalities in the considered 1 period year, several box-and-whisker diagrams were constructed, these diagrams can translate several statistical parameters as extremes, median, quartiles and in addition the mean location. The vertical lines (whiskers) give us indication about the variability outside the upper (above 3rd Quartile) and Lower (below 2st Quartile) quartiles. Finally, the outliers, reveal abnormal (distant) observations in the data series. With these diagrams, we hope to better understand data distribution, variability, seasonality, uncertainties and other trends.

In Figure 5.7 we can see the load monthly distribution, where we can infer the following remarks:

i. There is greater variability (larger amplitude) and greater demand in the winter months (December, January and February), with December claiming greater variability with a standard deviation of 1092.45 MW;
ii. Highest average demand occurred in the month of February with 6067.09 MW (average values represented in Figures as "+" symbol);
iii. The maximum demand was 8169.24 MW and occurred in the month of February, on 17/02/2016 at 19:45 (Wednesday);
iv. There is a less variability in the Summer months where August has the lowest variability with a standard deviation of 793.59 MW;
v. Minimum average demand occurred in the month of May with 5191.23 MW;
vi. The minimum value of the demand was 3441.77 MW and took place on 01/01/2016 at 08.15 (Friday).

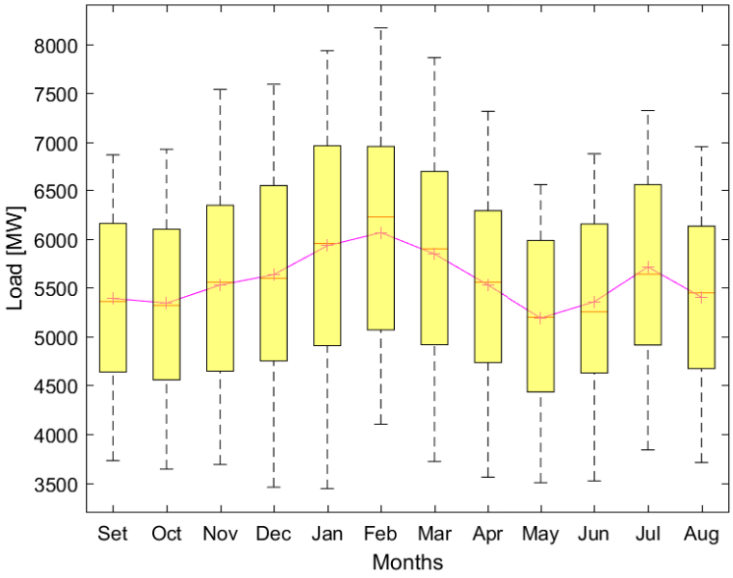


Figure 5.7 - Box-Plot for the monthly distribution of Portuguese load

To obtain a more detailed daily behavior, Figures 5.8 to 5.14 display the hourly distribution of demand (data grouped by days of the week and by season), where we can infer the following:

i. There is a greater average demand in weekdays;
ii. Demand profile is very similar in all weekdays;
iii. There is a smaller average demand on weekends;
iv. The evolution of demand on weekends is similar;
v. Although on weekends there is a lower demand there is a great variability;
vi. There is greater variability and greater demand among the 19 and 20 hours regardless of the day of the week;
vii. As expected lower demand values and lower standard deviations occur in the period between the 1 and 6 hours, being more pronounced between the 4 and the 5 hours regardless of the day of the week;
viii. Notice the existence of consecutive outliers forming a shape/profile in a different scale. This indicates that we are in the presence of a holiday. For example, in Figure 5.8 we clearly can identify the occurrence of a holiday on a Monday (Summer and Spring);
ix. With exception of outliers related to holydays (previous point) the Winter is the season most susceptible to suffer from abnormal observations, what is closely related to the fact that is the season with greatest standard deviation and as a result more difficult to predict accurately;
x. Another fact is that in the summer and in some extent in the spring the curves are smoother which means smaller slopes, a straighter morning ramp and with an off and peak demand closer to the average.

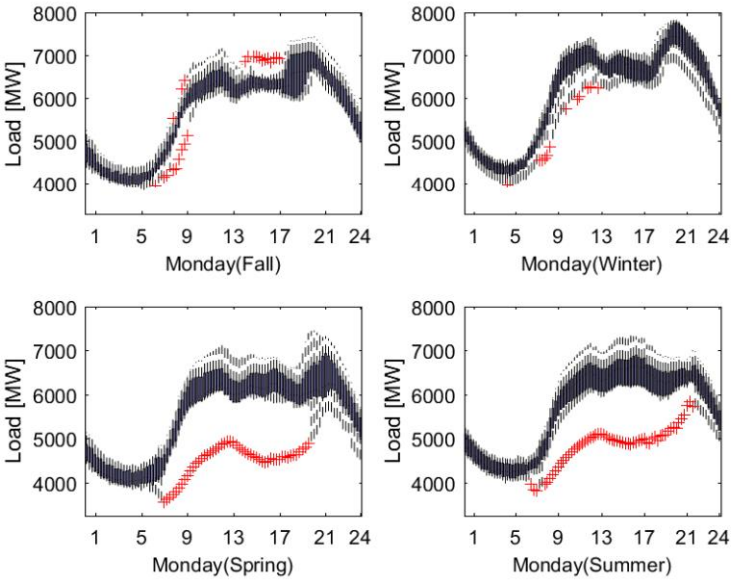


Figure 5.8 - Box Plot of load grouped by seasons on a Monday (PT)

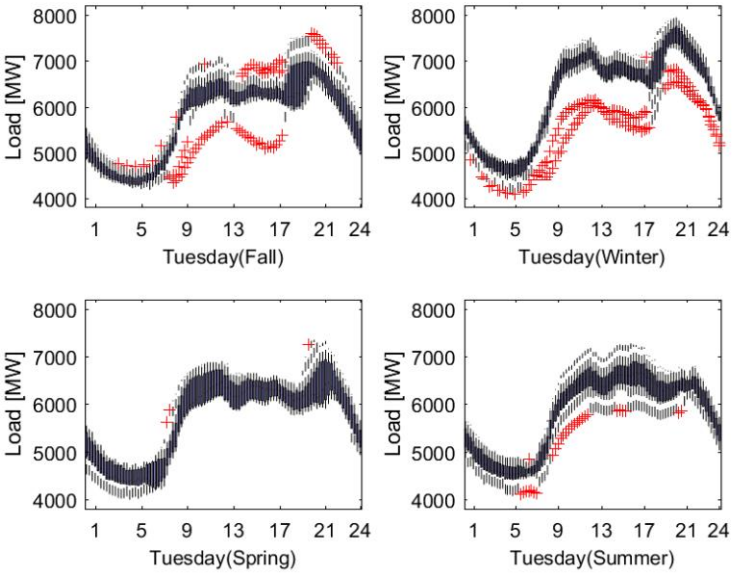


Figure 5.9 – Load box-plot grouped by seasons on a Tuesday (PT)

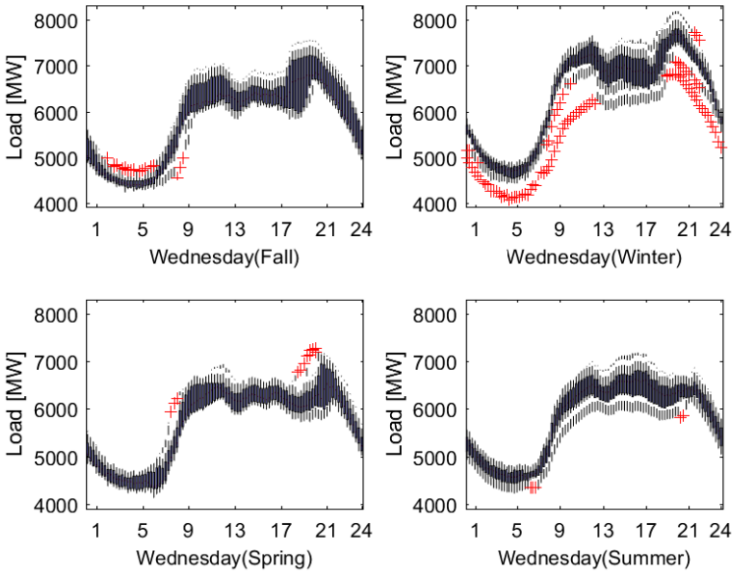


Figure 5.10 - Load box-plot grouped by seasons on a Wednesday (PT)

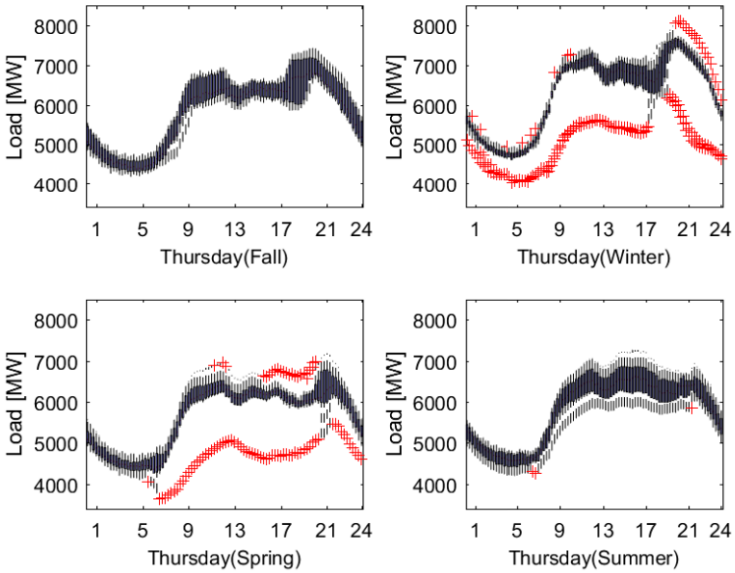


Figure 5.11 - Load box-plot grouped by seasons on a Thursday (PT)

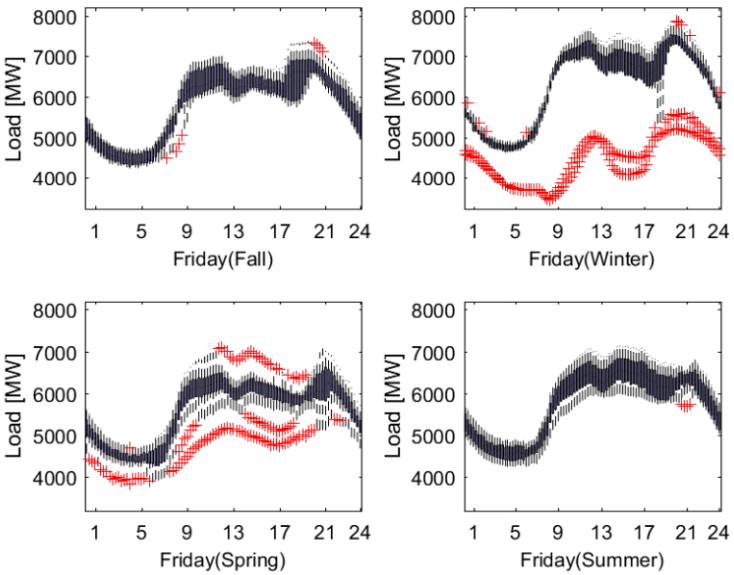


Figure 5.12 - Load box-plot grouped by seasons on a Friday (PT)

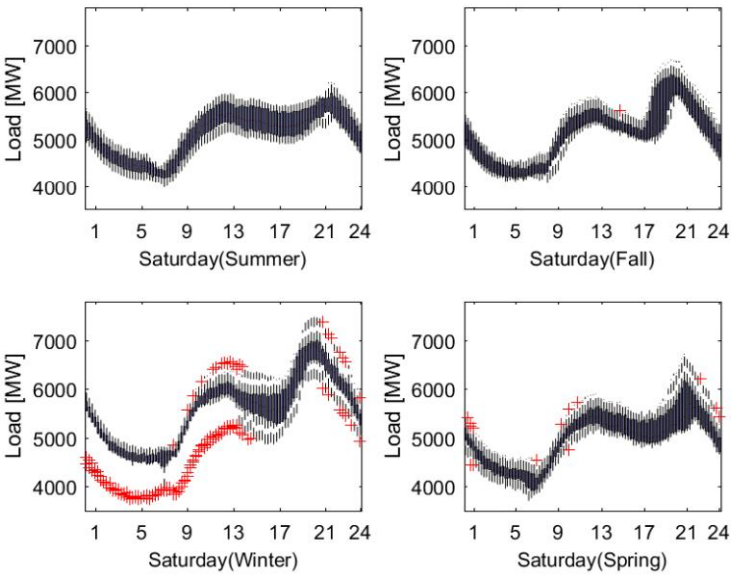


Figure 5.13 - Load box-plot grouped by seasons on a Saturday (PT)

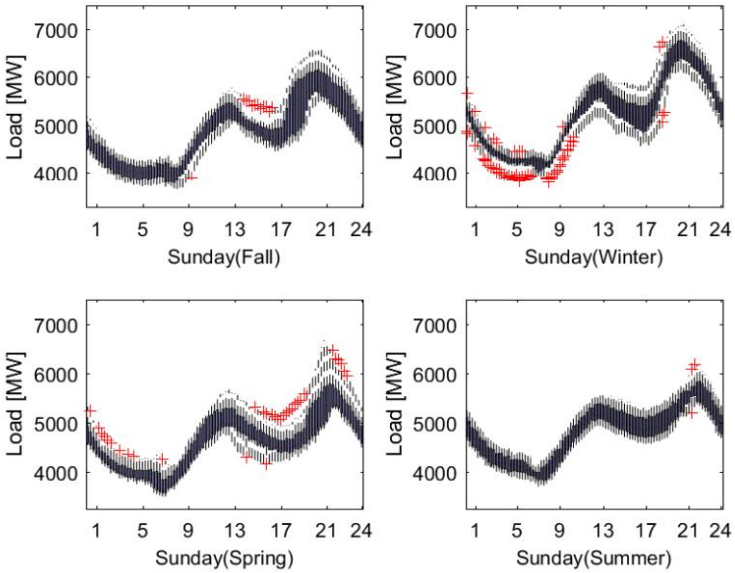


Figure 5.14 - Load box-plot grouped by seasons on a Sunday (PT)

5.2.1 Numerical Results

In total for the first case study, ANN input vector is formed by the 21 most correlated lags, an input with the respective hour, another with the correspondent weekday and a binary input revealing if we are in the presence of a holiday or not, making a total of 25 input entries.

As mentioned earlier, cross-validation technique was used. In this way, the data set is partitioned into subsets mutually exclusive ($k_{fold} = 4$), where 25% of the data set forms the validation set and 75% the training set.

So, in this case study, the proposed method was tested in 5 different set-ups, one week for each season and one on a holiday: (i) fall (15-21 November 2015); (ii) winter (15-21 January 2016); (iii) spring (15-21 May 2016); (iv) summer (15-21 July 2016); (v) 25 April 2016 (Monday and holiday).

Table 5.2 - 24H ahead forecasting-MAPE(%) (PT).

Weekday/Season	Fall	Winter	Spring	Summer
Monday	0.884	1.093	0.709	0.754
Tuesday	0.764	0.593	0.399	0.686
Wednesday	0.502	0.882	0.560	1.247
Thursday	0.507	0.985	0.575	0.627
Friday	0.606	0.651	0.432	0.815
Saturday	1.542	0.822	1.035	0.786
Sunday	0.852	0.804	0.846	0.859
Weekly Average	0.808	0.833	0.651	0.825
Holiday (Monday)	0.989			

The results are shown in Table 5.2 and by analyzing them we can infer the following: (i) overall, the results are quite satisfactory, the method performed well, with MAPE values (equation 5.2) consistently below 1%; (ii) the worst MAPE was obtained on 21 November of 2015 (Saturday), which may be justified by the greater variability during weekends in the Portuguese load profile; (iii) the best MAPE took place on 17 May 2016 (Tuesday); (iv) the week with the best performance occurred in the Spring and the worst in the Winter, as expected given the greatest variance recorded in the winter months, and the smallest variance in the spring months; (v) finally the MAPE obtained on holiday was 0.989%, revealing the method accurateness even in situations of greater variability, such as on weekends.

In all scenarios, the performed optimization revealed that ANN architectures with 2 hidden layers and a linear activation function in the output layer are favored. Figure 5.15 illustrates the predicted (forecasted) load versus the actual verified load for the chosen winter week.

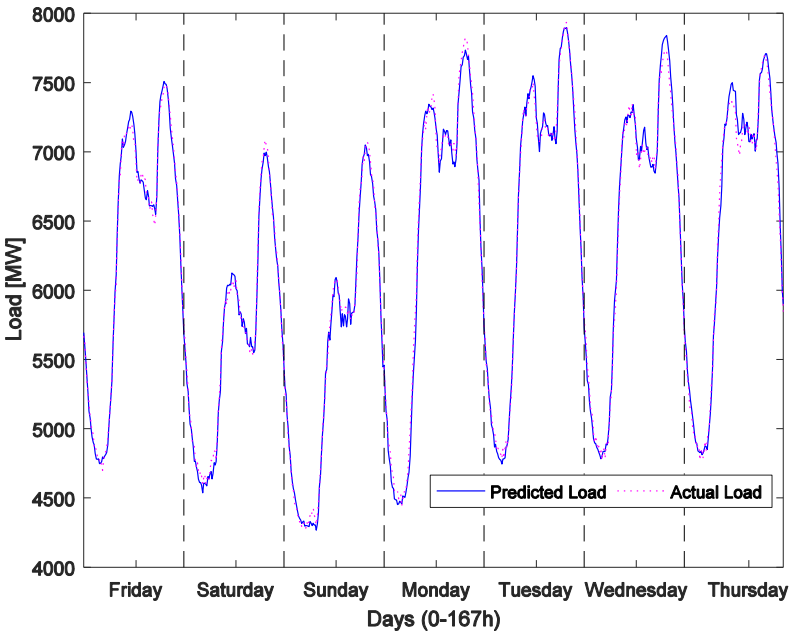


Figure 5.15 - Forecasted load for the 7-day winter season (PT).

5.3 Case Study II

To prove the efficiency of the method, the same procedure was applied in a second case study: the load data series from the New England USA region (as an hourly data series), for the period between 1 January to 31 December 2016. This data was provided by the U.S. Energy Information Administration (EIA) and available in [103]. The correspondent data series is represented in Figure 5.16 for every month and hour of the day.

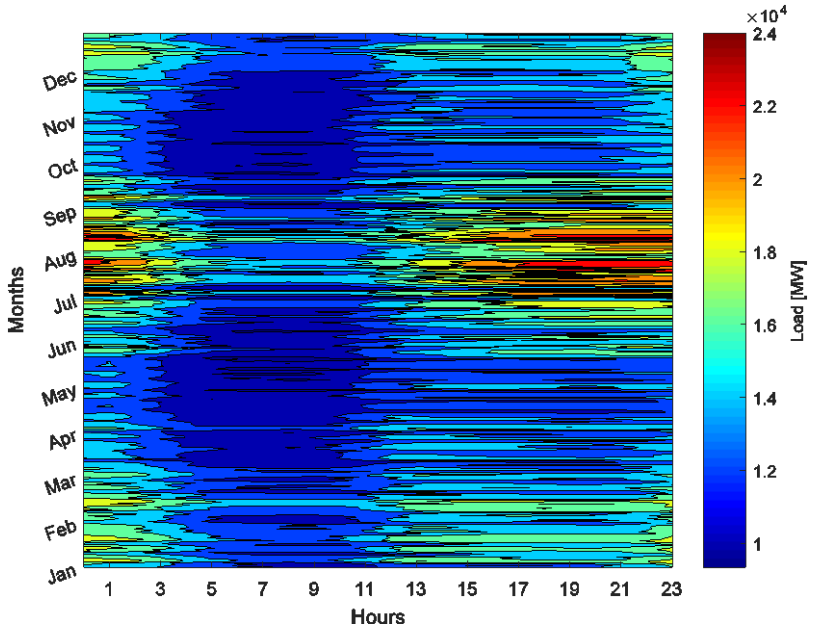


Figure 5.16 - Load [MW] from 1 January 2016 to 31 December 2016 (NE)

5.3.1 Numerical Results

In the second case study, the available database contained fewer days, and as such it was necessary to select more lags from the PAC. Therefore, the input vector was formed by the 29 most correlated lags and the time indicators (hour, weekday and a binary input indicating presence/absence of a holiday), for a total of 33 inputs.

The proposed method was also tested in 5 different scenarios, the first four corresponding to one month per season: (i) January 2016; (ii) April 2016; (iii) July 2016 (iv) November 2016; (v) and a last scenario on a holiday (4 July 2016). As for case I, the daily results from one week from each season are also presented.

The results are shown in Table 5.3 and Table 5.4 and by analyzing them we can infer the following: (i) overall results are quite satisfactory, the method performed well, with all MAPE values below 1%, except for the fall month in Table 5.4; (ii) the worst MAPE was obtained on the 4th of July of 2016 (Monday), which may be justified by the greater variability during (federal) holydays; (iii) the best MAPE took place on the 30 November 2016 (Wednesday); (iv) the week with the best performance occurred in the Summer and the worst in the Fall. As in the first case study, the performed optimization revealed that ANN architectures with 2 hidden layers and a linear activation function (*purelin*) in the output layer are favored.

Table 5.3 – 24h ahead forecasting MAPE (%) (WEEK-ISONE).

Weekday/Season	Winter	Spring	Summer	Fall
Monday	0.455	0.455	0.764	0.518
Tuesday	0.385	0.569	0.303	0.772
Wednesday	0.368	0.757	0.356	0.295
Thursday	0.496	0.488	0.295	0.429
Friday	0.391	0.451	0.347	0.829
Saturday	0.330	0.489	0.311	0.403
Sunday	0.754	0.419	0.489	0.409
Weekly Average	0.454	0.518	0.409	0.522
Holiday (Monday)	0.899			

Table 5.4 -24h ahead forecasting MAPE (%) (Month-ISONE).

Season	Winter	Spring	Summer	Fall
Min	0.099	0.292	0.295	0.295
Max	0.754	0.780	0.995	1.361
Average	0.368	0.522	0.515	0.634

Figure 5.17 illustrates the predicted (forecasted) load versus the actual verified load for the chosen fall week.

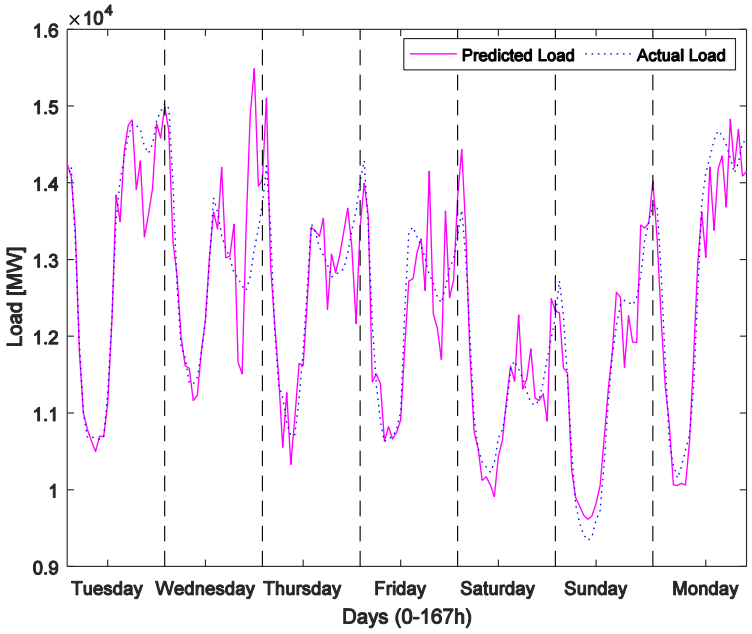


Figure 5.17 - Forecasted load for the 7-day spring season (ISNE).

For comparison purposes, Table 5.5 is presented, revealing the minimum and maximum average MAPE values in different case studies but all considering the load from New England (ISONE). It can be observed that the proposed method can produce superior results, showing MAPE values consistently lower than 1%.

Table 5.5 - Comparison to the state-of-the-art—MAPE(%) for ISONE.

	[13] Case I	[34] Example 2	[25] Table II	[32] Table IV	[32] Table VI	**Proposed**
Min	1.11	1.27	0.93	1.26	1.24	**0.368**
Max	2.37	2.59	1.86	2.7	2.53	**0.634**

Chapter VI

6 Price Forecast

In the same way as pointed out in the previous chapter, one of the objectives of this dissertation is the development of an effective methodology for price forecasting. Therefore, this chapter presents and details the proposed approach, as well as the main results achieved.

6.1 Proposed Methodology

The proposed methodology is very similar to the one presented in the chapter V for the STLF task. Subsequently, a similar version will be used in the STPF context. Figure 6.1 shows the proposed methodology flowchart. Illustrating in a generic way the various steps, starting from the price time series and ending in the 24h price forecast.

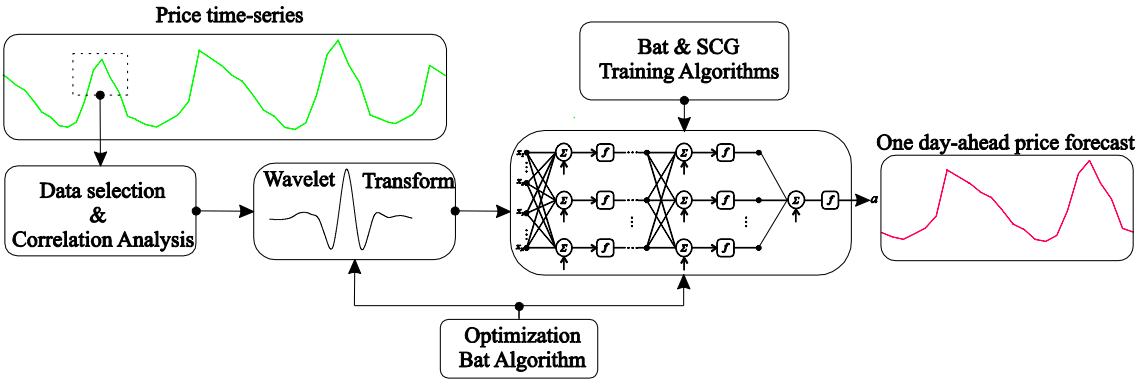


Figure 6.1 - Proposed methodology (flowchart).

Since it is basically the same methodology proposed in chapter V, only the different points will be highlighted:

i. To build the forecasting model, the input selection is formed with only the three most similar days and the preceding hourly historical price data of the seven days previous to the current day, without information from exogenous variables (simplicity and comparison purposes);

ii. Regarding STPF, the choice of small order *Daubechies* is typically a good choice [33], because of their smoothness and support interval offer a proper trade-off, with special emphasis on order four *Daubechies* (db4) [49], [51], [104], [105]. So, in the same way as for the STLF, minimum Shannon entropy criterion was used, the best orthogonal mother wavelets are presented in Table 6.1, and as expected there is a prevalence of *Daubechies* and *Symlets* mother wavelets.

Table 6.1 - Wavelet entropy results (STPF).

Mother Wavelet Pre-Selection
db2, db3, db4, coif1, fk4, sym2, sym3, sym4 and bior1.1

6.1.1 Forecasting accuracy measures

Typically, daily price profiles are classified as weekdays, from Monday to Friday, and weekend days, Saturday and Sunday, which are different. Another consideration besides weekend are public holidays, known as the calendar effect, since price profiles on non-holidays are particularly different from those on public holidays [37].

To measure method's prediction capacity we used three common error measures (criterions) [45], [106]. The first is the mean absolute percentage error (MAPE), already presented in equation 5.2. The second metric is the useful weekly MAPE, which is better to assess problems related with close to zero prices or spiking prices, and is calculated as:

$$MAPE_{Week}(\%) = \frac{100}{168}\sum_{t=1}^{N}\left|\frac{P_t - \hat{P}_t}{P_{Week}}\right| \quad (6.1)$$

where $P_{Week} = \sum_{t=1}^{168} P_t$, that is the average weekly price. Finally, the last error measure is the weekly error variance, which is as an important measure of the model uncertainty (variability), and is expressed as:

$$\mathrm{EV} = \frac{1}{168}\sum_{t=1}^{N}\left|\frac{P_t - \widehat{P}_t}{P_{\mathrm{Week}}} - \mathrm{MAPE}_{\mathrm{Week}}\right|^2 \tag{6.2}$$

The ANN input vector for both case studies was formed by the 36 most correlated lags, an input with the respective hour, another with the correspondent weekday and a binary input the presence of a holiday, making a total of 39 input entries.

As mentioned before, cross-validation technique was used. Therefore, the data set is partitioned into subsets mutually exclusive (k_{fold}=4), where 25% of the data set forms the validation set and 75% the training set.

6.2 Case Study I

The first case study was the duopoly Spanish market with a dominant player, resulting in price changes related to the strategic behaviour of the dominant player, which are hard to predict [33]. For comparison purposes the following reference weeks were used [37]: the winter week from February 18 to February 24, 2002; the spring week from May 20 to May 26, 2002; the summer week from August 19 to August 25, 2002 and lastly the fall week from November 18 to November 24, 2002. This data was provided by the Red Electrica de España (REE) and is available in [107].

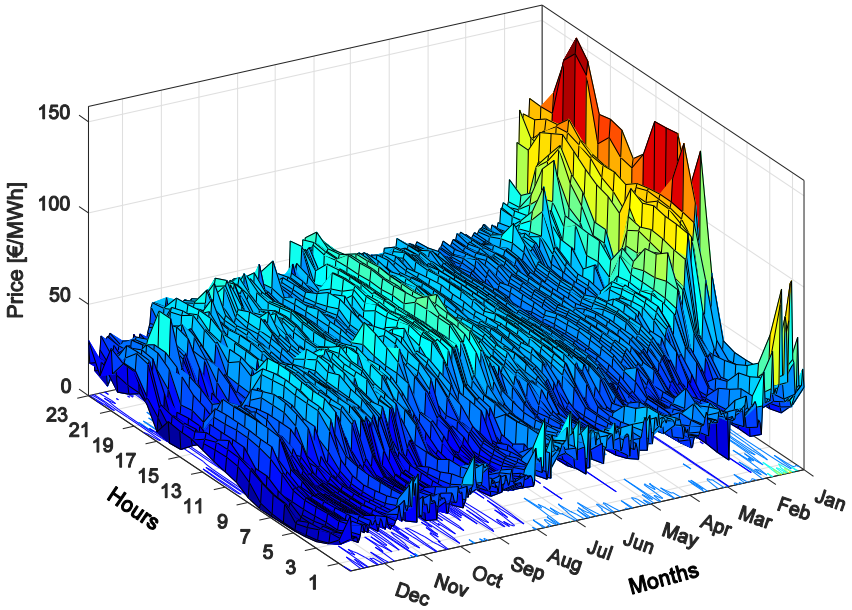


Figure 6.2 – Electricity prices from 1 January 2002 to 31 December 2002 (SP)

For each hour and month, prices are represented by a surface plot in Figure 6.2. First, a brief statistical analysis was made of the prices in the electricity market of mainland Spain in 2002. The average price and variance for the whole year was 37.4 €/MWh and 263.1 €^{2}/MWh^2 respectively, the maximum recorded price was 158.4 €/MWh on the 10th of January.

In contrast, on 2nd of April, prices reached the minimum of 0 €/MWh. A distinctive trait in the winter months, especially in January, is the high average (62.0 €/MWh) and variance (787.4 $€^2/MWh^2$), illustrating a typical seasonal effect in the Spain price time series.

6.2.1 Numerical Results

The results obtained for the four weeks referred above are presented in Table 6.2 for all the selected error measures (MAPE, $MAPE_{Week}$ and EV). These three criterions revealed quite satisfactory results, and as we can see the winter week presents the best results, despite its higher variance, meaning that the data selection and features extraction performed well.

Table 6.2 - Weekly forecasting errors for the four test weeks (Spain 2002).

	Winter	**Spring**	**Summer**	**Fall**	**Average**
MAPE (%)	0.61	1.22	1.30	1.81	1.24
$MAPE_{Week}$ (%)	0.57	1.11	1.08	1.46	1.06
EV	1.7E-04	2.7E-04	8.5E-04	6.8E-04	4.9E-04

For comparison and validation purposes, results are compared in Tables 6.3 and 6.4 with a variety of other price forecasting methods, ranging from the classical Parsimonious methods to the recent hybrid methods (with the same test weeks and criterions).

Table 6.3 - Comparative $MAPE_{Week}$ (%) for the STPF of the Spanish Electricity Market.

	Winter	Spring	Summer	Fall	Average
Naive [106]	7.68	7.27	27.30	19.98	15.56[a)]
ARIMA [106]	6.32	6.36	13.39	13.78	9.96[a)]
Wavelet-ARIMA [106]	4.78	5.69	10.70	11.27	8.11[a)]
Mixed-model [108]	6.15	4.46	14.90	11.68	9.30
NN [37]	5.23	5.36	11.40	13.65	8.91
WNN [109]	5.15	4.34	10.89	11.83	8.05
FNN [110]	4.62	5.30	9.84	10.32	7.52
HIS [111]	6.06	7.07	7.47	7.30	6.97
AWNN [112]	3.43	4.67	9.64	9.29	6.75
NNWT [113]	3.61	4.22	9.50	9.28	6.65
CNEA [114]	4.88	4.65	5.79	5.96	5.32
WPA [104]	3.37	3.91	6.50	6.51	5.07
RBFN-PSO [47]	4.27	4.58	6.76	7.35	5.74[a)]
Neuro-fuzzy [49]	3.38	4.01	9.47	9.27	6.53[a)]
CNN [115]	4.21	4.76	6.01	5.88	5.22
Mixed-Model 2[116]	4.22	4.39	5.55	5.66	4.95
MRMRMS + RBFNN [117]	3.56	3.90	6.24	6.06	4.94
MRMRMS + MLP [117]	3.60	3.94	6.12	5.83	4.87
MRMRMS + WNN [117]	3.36	3.84	5.96	5.74	4.72
CNN+SSM [50]	3.28	3.62	5.32	5.03	4.31
ARIMA/GARCH+WT [45]	0.63	0.65	1.19	2.18	1.16
Proposed	**0.57**	**1.11**	**1.08**	**1.46**	**1.06**

a)- Value determined using the MAPE values from the respective reference.

NA- means not available.

As can be verified, the results of the proposed method are among the best for both the average $MAPE_{Week}$ and EV of all test weeks. Only in the spring week did the ARIMA/GARCH+WT [45] slightly surpass the proposed method. Another peculiarity that can also be observed is that results typically get worse for the summer and fall test weeks [105]. The comparison also revealed that of all the soft computing and hybrid approaches, the proposed method presents a superior accuracy.

Table 6.4 - Comparative EV (10^{-4}) for the STPF of the Spanish Electricity Market

	Winter	Spring	Summer	Fall	Average
Naive [106]	3.40	4.30	73.8	33.8	28.8 [a)]
ARIMA [106]	3.40	2.00	15.8	15.7	9.23 [a)]
Wavelet-ARIMA [106]	1.90	2.50	10.8	10.3	6.38 [a)]
Mixed-model [108]	NA	NA	NA	NA	NA
NN [37]	1.70	1.80	10.9	13.6	7.00
WNN [109]	NA	NA	NA	NA	NA
FNN [110]	1.80	1.90	9.20	8.80	5.40
HIS [111]	3.40	4.90	2.90	3.10	3.60
AWNN [112]	1.20	3.10	7.40	7.50	4.80
NNWT [113]	0.90	1.70	7.40	4.90	3.70
CNEA [114]	3.60	2.70	4.30	3.90	3.60
WPA [104]	0.80	1.30	5.60	3.30	2.70
RBFN-PSO [47]	1.50	1.90	4.70	4.90	3.25 [a)]
Neuro-fuzzy [49]	NA	NA	NA	NA	NA
CNN [115]	1.40	3.30	4.50	4.80	3.40
Mixed-Model 2[116]	1.50	2.90	3.90	5.20	3.38
MRMRMS + RBFNN [117]	NA	NA	NA	NA	NA
MRMRMS + MLP [117]	NA	NA	NA	NA	NA
MRMRMS + WNN [117]	NA	NA	NA	NA	NA
CNN+SSM [50]	1.20	1.20	2.80	1.80	1.70
ARIMA/GARCH+WT [45]	0.20	0.20	0.90	0.80	0.53 [a)]
Proposed	**0.17**	**0.27**	**0.85**	**0.68**	**0.49**

a)- Value determined using the MAPE values from the respective reference.

NA- means not available.

The obtained numerical results are also shown graphically in Figures 6.3 and 6.4, respectively, for the winter, and fall weeks. In each figure, the dashed line represents forecast prices, while the actual prices are represented by a solid line. While for week days the proposed method presents a small error and the two curves are virtually overlapping, for weekends and holidays this error increases, and the difference between curves becomes more visible.

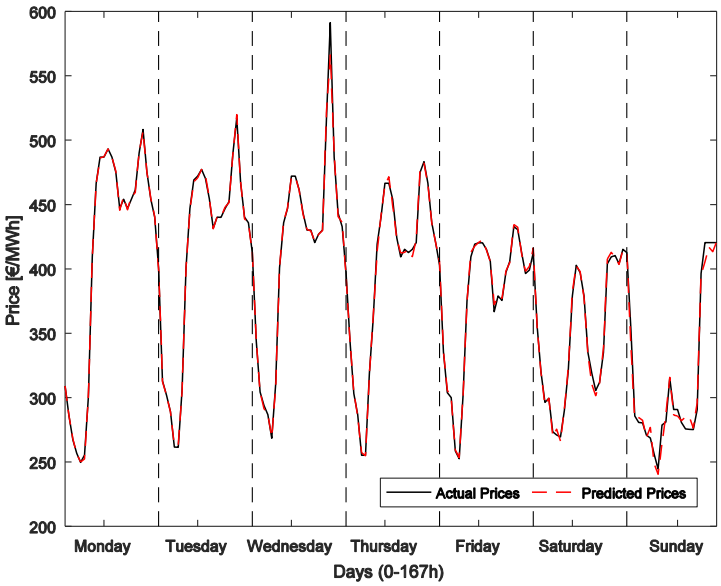


Figure 6.3 - Winter week for the Spanish market: actual prices, solid line, together with the forecasted prices, dashed line.

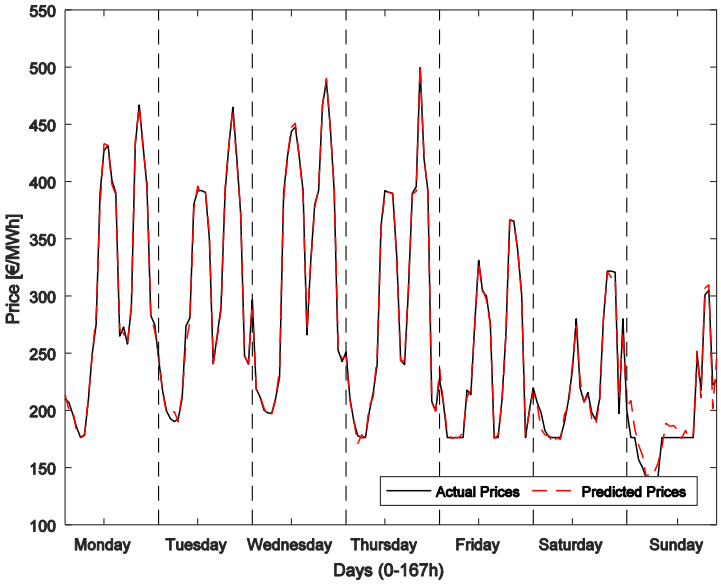


Figure 6.4 - Fall week for the Spanish market: actual prices, solid line, together with the forecasted prices, dashed line.

6.3 Case Study II

The second case study was the Pennsylvania-New Jersey-Maryland (PJM) electricity spot market, one of the Regional Transmission Organizations, which is well-known in the U.S. and beyond, and plays a vital role in the U.S. electric system [50]. For the PJM market, four test weeks corresponding to four seasons of year 2006 were considered for the sake of comparison. The four considered weeks were February 15 to February 21, May 15 to May 21, August 15 to August 21, and November 15 to November 21. Data of PJM electricity market is obtained from [118].

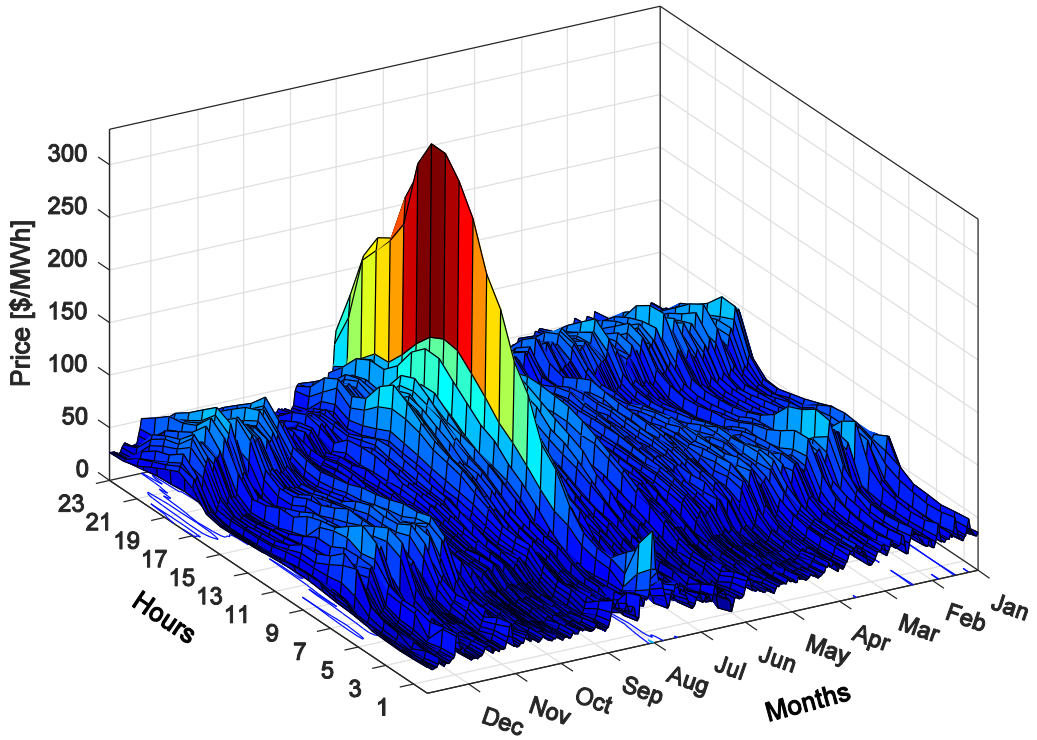


Figure 6.5 - Electricity prices in the PJM market for the 2006 year.

The 2006 prices in the PJM spot market for each hour and month are represented by a surface plot in Figure 6.5. As for the first case study, a brief statistical analysis was made. The average price and variance for the whole year were 48.1 $/MWh and 548.6 ^{2}/MWh^2 respectively. The maximum recorded price was 333.9 $/MWh, on the 3 of August. In contrast, the minimum of 0 $/MWh was reached on several days. A distinctive trait of the summer months of July and August was its high average and variance. For example, in August the variance reached 2242.6 ^{2}/MWh^2, with an average price value of 67.5 $/MWh.

6.3.1 Numerical Results

Price prediction results of the proposed forecasting method for referred test weeks are presented in Table 6.5, for all the selected error measures (MAPE, $MAPE_{Week}$ and EV). These three criteria revealed quite satisfactory results, and as we can see the fall week presents the best result. In contrast, the spring week reveals the worst result. As in case I, there was no direct relationship between the variance and the results for the test weeks.

Table 6.5 - Weekly forecasting errors for the four test weeks (PJM 2006).

	Winter	Spring	Summer	Fall	Average
MAPE(%)	0.79	0.80	0.81	0.54	0.75
$MAPE_{Week}$(%)	0.74	0.72	0.68	0.45	0.65
EV	7.20E-05	4.52E-05	1.54E-05	1.53E-05	3.70E-05

Once more, for comparison and validation purposes, criteria are compared in Tables 6.6 and 6.7 with a diversity of other price forecasting methods (with the same test weeks and

criterions). As shown, the results of the proposed method are among the best for both the average MAPE and $MAPE_{Week}$ of all test weeks. The average MAPE is equal to that presented by ARIMA/GARCH+WT [45] and surpasses the remaining approaches. The comparison also revealed that among all the soft computing and hybrid approaches, the proposed method presents a superior accuracy.

Table 6.6 - Comparative MAPE (%) for the STPF of the PJM Electricity Market.

	Winter	**Spring**	**Summer**	**Fall**	**Average**
NSA [117]	8.45	8.23	10.09	9.68	9.11
PCA [117]	7.31	8.12	10.28	9.59	8.83
CA [117]	11.10	8.07	9.94	6.82	8.98
MI [117]	6.46	6.34	6.87	6.93	6.38
MR [117]	6.20	6.41	6.38	6.35	6.33
CA + CA [117]	6.11	6.31	6.45	6.29	6.29
MI + CA [117]	5.65	6.10	6.23	5.95	5.98
MR + CA [117]	6.49	5.63	6.16	5.39	5.92
CA + MI [117]	5.59	5.14	5.76	5.62	5.53
MI + MI [117]	4.94	4.99	4.81	4.75	4.87
MR + MI [117]	4.83	4.79	4.81	4.88	4.83
MI + IG [117]	4.71	4.47	4.70	4.43	4.58
MRMRMS [117]	4.62	4.33	4.62	4.38	4.49
NN [48]	6.35	4.74	4.76	7.00	5.71
MLP + LM [45]	9.82	8.87	10.43	9.54	9.67
PCA + CNN [45]	8.61	7.34	8.17	8.36	8.12
WT + NN [45]	4.44	4.31	4.78	4.75	4.57
ARIMA/GARCH+WT [45]	0.73	0.60	0.75	0.86	0.74
Proposed	**0.79**	**0.80**	**0.81**	**0.54**	**0.74**

Table 6.7 - Comparative $MAPE_{Week}$ (%) for the STPF of the PJM Electricity Market

	Winter	**Spring**	**Summer**	**Fall**	**Average**
AC + NN [119]	11.29	7.39	8.27	6.89	8.46
PCA + NN [119]	8.13	7.43	8.68	9.57	8.45
RFE + NN [119]	5.97	5.36	5.95	5.73	5.75
GC + NN [119]	5.76	5.24	5.61	5.46	5.52
MR + NN [119]	5.21	5.13	5.57	4.79	5.18
CNN+SSM [50]	3.97	4.02	4.04	4.12	4.04
Proposed	**0.74**	**0.72**	**0.68**	**0.45**	**0.65**

The obtained numerical results are also shown in Figures 6.6 and 6.7, respectively, for the spring and summer weeks. In each figure, the dashed line represents forecasted prices, and the solid line represents actual prices. As in case I, the referred figures also show that, for week days, the proposed method presents a short error and therefore the two curves are virtually overlapping. However, for weekends and holidays, this error increases and the difference between curves is more evident.

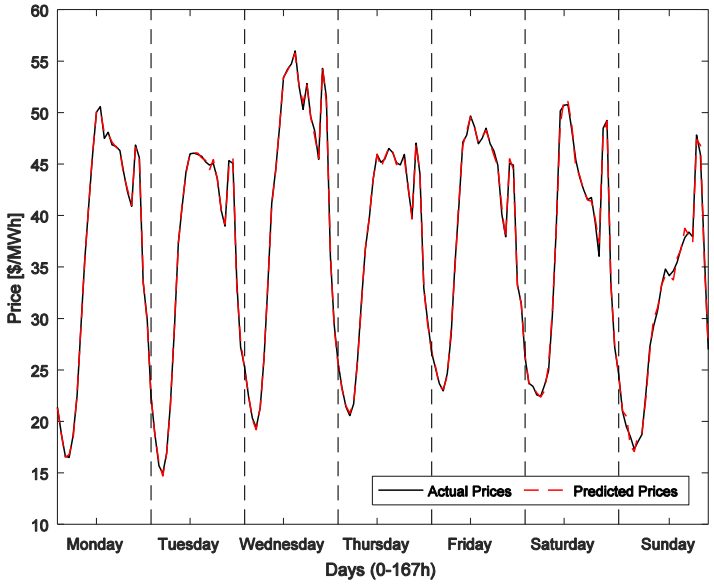


Figure 6.6 - Spring week for the PJM market: actual prices, solid line, together with the forecasted prices, dashed line.

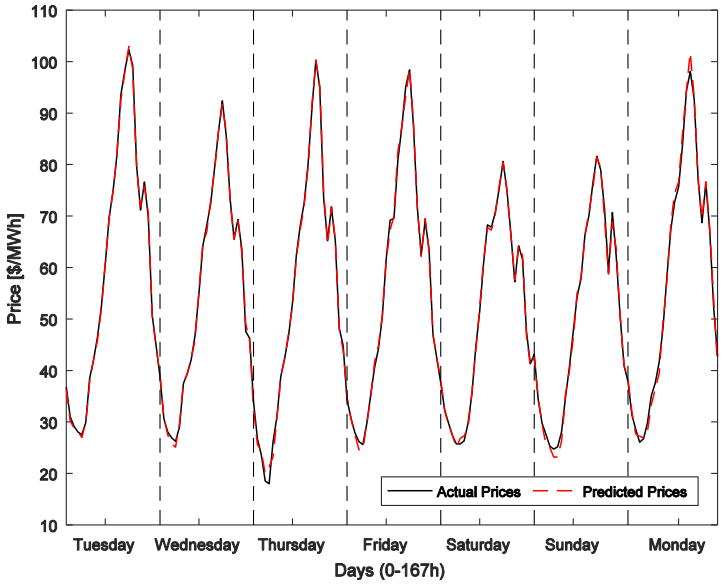


Figure 6.7 - Summer week for the PJM market: actual prices, solid line, together with the forecasted prices, dashed line.

Chapter VII

7 Final Remarks

In this chapter, the most important conclusions are presented, as well as directions for future works.

7.1 Conclusion

In an overall analysis of this dissertation, the research, study and method's conception work was a very fulfilling and enriching experience. The first four chapters worked as background parts, and were fundamental to the subsequent problem analysis, its comprehension and method implementation.

The main objective for this dissertation was the conception of a new forecasting method for STLF and STPF, following the recent literature propensity to combine different techniques and algorithms in hybrid approaches. Consequently an enhanced method combining synergies from different techniques, including recent/similar day selection, wavelet transform, neural networks, bat and scaled conjugate gradient algorithms and particle swarm optimization in an optimized arrangement. The domain space optimization (Figure 5.4) was performed using BA in order to find the best configuration from the available data, thus taking advantage of the best ANN and WT capabilities. The performed optimization revealed that ANN architectures with 2 hidden layers linear activation function (*purelin*) in the output layer, 2 or 3 decomposition levels for WT, *Symlets* or *Daubechies* as mother

wavelets, particularly db2, db4 and db8 for the STLF (Chapter V) and sym2, sym3 and db4 for STPF (Chapter VI) were favored.

For this reason, the proposed approach can be applied both to STLF (Chapter V) and STPF (Chapter VI) problems, the same is valid for different electricity markets (Portuguese, New England, Spanish and Pennsylvania-New Jersey-Maryland) without a loss of performance.

All the numerical studies were run on a PC with 8 GB RAM memory and an i7 processor (2.4 GHz). The average computational time, including training and forecasting, was 30 minutes to evaluate one configuration (Figure 5.4). Numerical results reveal that the proposed method produces excellent forecasting results, beyond other well-established methods and dealing well with the volatility evidenced in the price and load time-series. The average MAPE values are under or around 1%, and as expected the method has a better accuracy on weekdays (excluding holidays).

7.2 Future Works

The present dissertation opens prospects for updates and improvements (in several fields) to the proposed methodology, as well as different research and development paths to approach in upcoming works. The following topics are presented interesting features for further work:

i. Inclusion of exogenous inputs, such as wind production;
ii. Explore new optimization and artificial intelligence techniques;
iii. Address new data selection techniques, such as Mutual Information;
iv. Reduce the computational running time.

8 References

[1] O. Elgerd, *Electric Energy Systems Theory: An Introduction*, 3rd ed. New York, 1976.

[2] EIA and L. Doman, "International Energy Outlook (IEO)," *International Energy Outlook 2016*, 2016. [Online]. Available: https://www.eia.gov/todayinenergy/detail.php?id=26212. [Accessed: 20-Apr-2017].

[3] I. J. P. Arriaga, H. Rudnick, and M. Rivier, "Electric Energy Systems. An Overview," 2009.

[4] energymag, "Daily energy demand curve." [Online]. Available: https://energymag.net/daily-energy-demand-curve/. [Accessed: 20-Nov-2016].

[5] EIA, "Demand for electricity changes through the day," 2011. [Online]. Available: https://www.eia.gov/todayinenergy/detail.php?id=830. [Accessed: 19-Mar-2017].

[6] R. C. Novaes, L. C. de Morais, F. P. dos A. Lima, and S. S. F. Souza, "ARTIFICIAL NEURAL NETWORKS APPLIED IN FORECASTING THE PRICE IN THE IBERIAN ELECTRICITY MARKET," *Rev. Eng. em Ação UniToledo*, vol. 1, no. 1, pp. 153–168, 2016.

[7] E. a Feinberg and D. Genethliou, "Applied Mathematics for Restructured Electric Power Systems," in *IEEE Transactions on Automatic Control*, vol. 50, no. 9, 2005, pp. 269–285.

[8] B. U. Islam, "Comparison of Conventional and Modern Load Forecasting Techniques Based on Artificial Intelligence and Expert Systems," *Int. J. Comput. Sci.*, vol. 8, no. 5, pp. 504–513, 2011.

[9] S. S. Soman, H. Zareipour, O. Malik, and P. Mandal, "A review of wind power and wind speed forecasting methods with different time horizons," *North Am. Power Symp.*, pp. 1–8, 2010.

[10] N. K. Singh, M. Tripathy, and A. K. Singh, "A radial basis function neural network approach for multi-hour short term load-price forecasting with type of day parameter," *2011 6th Int. Conf. Ind. Inf. Syst. ICIIS 2011 - Conf. Proc.*, pp. 316–321, 2011.

[11] N. Amjady and F. Keynia, "Short-term load forecasting of power systems by combination of wavelet transform and neuro-evolutionary algorithm," *Energy*, vol. 34, no. 1, pp. 46–57, 2009.

[12] S. Li, P. Wang, and L. Goel, "A novel wavelet-based ensemble method for short-term load forecasting with hybrid neural networks and feature selection," *IEEE Trans. Power Syst.*, vol. 31, no. 3, pp. 1788–1798, 2016.

[13] V. H. Ferreira, A. P. Alves, and S. Member, "Network-Based Electric Load Forecasters," *IEE Trans. Power Syst.*, vol. 22, no. 4, pp. 1554–1562, 2007.

[14] Z. A. Bashir and M. E. El-Hawary, "Applying wavelets to short-term load forecasting using PSO-based neural networks," *IEEE Trans. Power Syst.*, vol. 24, no. 1, pp. 20–27,

2009.

[15] C. L. Hor, S. J. Watson, and S. Majithia, “Daily load forecasting and maximum demand estimation using ARIMA and GARCH,” *2006 9th Int. Conf. Probabilistic Methods Appl. to Power Syst. PMAPS*, pp. 11–16, 2006.

[16] I. Moghram and S. Rahman, “Analysis and Evaluation of Five Short Term Load Forecasting Technique,” *IEEE Trans. Power Syst.*, vol. 4, no. 4, pp. 1484–1490, 1989.

[17] K. Liu *et al.*, “Comparison of very short-term load forecasting techniques,” *IEEE Trans. Power Syst.*, vol. 11, no. 2, pp. 877–882, 1996.

[18] A. Lahouar and J. Ben Hadj Slama, “Day-ahead load forecast using random forest and expert input selection,” *Energy Convers. Manag.*, vol. 103, pp. 1040–1051, 2015.

[19] P. Mandal, T. Senjyu, K. Uezato, and T. Funabashi, “Several-hours-ahead electricity price and load forecasting using neural networks,” *2005 IEEE Power Eng. Soc. Gen. Meet.*, pp. 2205–2212, 2005.

[20] S. . Yao, Y. . Song, L. . Zhang, and X. . Cheng, “Wavelet transform and neural networks for short-term electrical load forecasting,” *Energy Convers. Manag.*, vol. 41, no. 18, pp. 1975–1988, 2000.

[21] K. B. Sahay and M. M. Tripathi, “Day ahead hourly load and price forecast in ISO New England market using ANN,” *2013 Annu. IEEE India Conf. INDICON 2013*, 2013.

[22] Ying-Ying Cheng, P. P. . Chan, and Zhi-Wei Qiu, “Random forest based ensemble system for short term load forecasting,” *2012 Int. Conf. Mach. Learn. Cybern.*, pp. 52–56, 2012.

[23] D. Niu, Y. Wang, and D. D. Wu, “Power load forecasting using support vector machine and ant colony optimization,” *Expert Syst. Appl.*, vol. 37, no. 3, pp. 2531–2539, 2010.

[24] E. Ceperic, V. Ceperic, and A. Baric, “A strategy for short-term load forecasting by support vector regression machines,” *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4356–4364, 2013.

[25] D. K. Ranaweera, N. F. Hubele, and G. G. Karady, “Fuzzy logic for short term load forecasting,” *Int. J. Electr. Power Energy Syst.*, vol. 18, no. 4, pp. 215–222, 1996.

[26] A. M. Al-Kandari, S. A. Soliman, and M. E. El-Hawary, “Fuzzy short-term electric load forecasting,” *Int. J. Electr. Power Energy Syst.*, vol. 26, no. 2, pp. 111–122, 2004.

[27] T. Herawan, R. Ghazali, and M. M. Deris, “Recent Advances on Soft Computing and Data Mining: Proceedings of the First International Conference on Soft Computing and Data Mining (SCDM-2014) Universiti Tun Hussein Onn Malaysia, Johor, Malaysia June, 16th-18th, 2014,” *Adv. Intell. Syst. Comput.*, vol. 287, pp. 163–172, 2014.

[28] C.-H. Kim, B.-G. Koo, and J. H. Park, “Short-term Electric Load Forecasting Using Data Mining Technique,” *J. Electr. Eng. Technol.*, vol. 7, no. 6, pp. 807–813, 2012.

[29] H. Nie, G. Liu, X. Liu, and Y. Wang, “Hybrid of ARIMA and SVMs for short-term load forecasting,” in *Energy Procedia*, 2011, vol. 16, no. PART C, pp. 1455–1460.

[30] S. Annamareddi, S. Gopinathan, and B. Dora, “A Simple Hybrid Model for Short-Term Load Forecasting,” vol. 2013, 2013.

[31] Ying Chen *et al.*, "Short-Term Load Forecasting: Similar Day-Based Wavelet Neural Networks," *IEEE Trans. Power Syst.*, vol. 25, no. 1, pp. 322–330, 2010.

[32] A. J. Conejo, M. A. Plazas, R. Espinola, and A. B. Molina, "Day-ahead electricity price forecasting using the wavelet transform and ARIMA models," *Power Syst. IEEE Trans.*, vol. 20, no. 2, pp. 1035–1042, 2005.

[33] S. Li, P. Wang, and L. Goel, "Short-term load forecasting by wavelet transform and evolutionary extreme learning machine," *Electr. Power Syst. Res.*, vol. 122, pp. 96–103, 2015.

[34] M. Ulagammai, P. Venkatesh, P. S. Kannan, and N. Prasad Padhy, "Application of bacterial foraging technique trained artificial and wavelet neural networks in load forecasting," *Neurocomputing*, vol. 70, no. 16–18, pp. 2659–2667, 2007.

[35] L. C. M. de Andrade and I. N. da Silva, "Very Short-Term Load Forecasting Based on ARIMA Model and Intelligent Systems," in *2009 15th International Conference on Intelligent System Applications to Power Systems*, 2009, pp. 1–6.

[36] J. P. S. Catalão, S. J. P. S. Mariano, V. M. F. Mendes, and L. A. F. M. Ferreira, "Short-term electricity prices forecasting in a competitive market: A neural network approach," *Electr. Power Syst. Res.*, vol. 77, no. 10, pp. 1297–1304, 2007.

[37] 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.

[38] A. Angelus, "Electricity Price Forecasting in Deregulated Markets," *Electr. J.*, vol. 14, no. 3, pp. 32–41, 2001.

[39] J. Contreras, R. Espínola, F. J. Nogales, and A. J. Conejo, "ARIMA models to predict next-day electricity prices," *IEEE Trans. Power Syst.*, vol. 18, no. 3, pp. 1014–1020, 2003.

[40] H. Liu and J. Shi, "Applying ARMA-GARCH approaches to forecasting short-term electricity prices," *Energy Econ.*, vol. 37, pp. 152–166, 2013.

[41] Z. H. Z. Hua, X. L. X. Li, and Z. L. Z. Li-zi, "Electricity price forecasting based on GARCH model in deregulated market," *2005 Int. Power Eng. Conf.*, 2005.

[42] F. J. Nogales, J. Contreras, A. J. Conejo, and R. Espínola, "Forecasting next-day electricity prices by time series models," *IEEE Trans. Power Syst.*, vol. 17, no. 2, pp. 342–348, 2002.

[43] J. Crespo Cuaresma, J. Hlouskova, S. Kossmeier, and M. Obersteiner, "Forecasting electricity spot-prices using linear univariate time-series models," *Appl. Energy*, vol. 77, no. 1, pp. 87–106, 2004.

[44] Z. Tan, J. Zhang, J. Wang, and J. Xu, "Day-ahead electricity price forecasting using wavelet transform combined with ARIMA and GARCH models," *Appl. Energy*, vol. 87, no. 11, pp. 3606–3610, 2010.

[45] V. Chintham, "Electricity price forecasting of deregulated market using Elman neural network," in *IEEE INDICON*, 2015, pp. 1–5.

[46] M. Shafie-Khah, M. P. Moghaddam, and M. K. Sheikh-El-Eslami, “Price forecasting of day-ahead electricity markets using a hybrid forecast method,” *Energy Convers. Manag.*, vol. 52, no. 5, pp. 2165–2169, 2011.

[47] M. Gholipour Khajeh, A. Maleki, M. A. Rosen, and M. H. Ahmadi, “Electricity price forecasting using neural networks with an improved iterative training algorithm,” *Int. J. Ambient Energy*, vol. 0, no. 0, pp. 1–12, 2017.

[48] J. P. S. Catalão, H. M. I. Pousinho, and V. M. F. Mendes, “Short-term electricity prices forecasting in a competitive market by a hybrid intelligent approach,” *Energy Convers. Manag.*, vol. 52, no. 2, pp. 1061–1065, Feb. 2011.

[49] O. Abedinia, N. Amjady, M. Shafie-khah, and J. P. S. Catalão, “Electricity price forecast using Combinatorial Neural Network trained by a new stochastic search method,” *Energy Convers. Manag.*, vol. 105, no. 2015, pp. 642–654, 2015.

[50] P. Mandal, Haque, J. Meng, Srivastava, and R. Martinez, “A novel hybrid approach using wavelet, firefly algorithm, and fuzzy ARTMAP for day-ahead electricity price forecasting,” *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1041–1051, 2013.

[51] Matlab, “Continuous Wavelet Transform and Scale-Based Analysis,” *Matlab online*. [Online]. Available: https://www.mathworks.com/help/wavelet/gs/continuous-wavelet-transform-and-scale-based-analysis.html. [Accessed: 20-Feb-2017].

[52] R. Polikar, “http://users.rowan.edu/~polikar/WAVELETS/WTpart2.html,” *The Wavelet Tutorial*. .

[53] M. Misiti, Y. Misiti, G. Oppenheim, and J. M. Poggi, *Wavelets and their Applications*. 2010.

[54] J. L. Semmlow, *Biosignal and medical image processing*. 2009.

[55] E. G. de Souza e Silva, L. F. L. Legey, and E. A. de Souza e Silva, “Forecasting oil price trends using wavelets and hidden Markov models,” *Energy Econ.*, vol. 32, no. 6, pp. 1507–1519, 2010.

[56] A. J. R. Reis and A. P. Alves da Silva, “Feature Extraction via Multiresolution Analysis for Short-Term Load Forecasting,” *IEEE Trans. Power Syst.*, vol. 20, no. 1, pp. 189–198, 2005.

[57] F. Li, G. Meng, K. Kageyama, Zhongqing Su, and Lin Ye, “Optimal Mother Wavelet Selection for Lamb Wave Analyses,” *J. Intell. Mater. Syst. Struct.*, vol. 20, no. 10, pp. 1147–1161, 2009.

[58] S. G. Mallat, “A Theory for Multiresolution Signal Decomposition: The Wavelet Representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, pp. 674–693, 1989.

[59] R. X. Gao and R. Yan, “Wavelets: Theory and applications for manufacturing,” *Wavelets Theory Appl. Manuf.*, pp. 1–224, 2011.

[60] Z. Xu, Z. Y. Dong, W. Q. Liu, and S. Lucia, “Neural Network Models for Electricity,” no. December 2014, 1987.

[61] D. Benaouda, F. Murtagh, J. L. Starck, and O. Renaud, “Wavelet-based nonlinear

multiscale decomposition model for electricity load forecasting," *Neurocomputing*, vol. 70, no. 1–3, pp. 139–154, 2006.

[62] F. D. Lane and D. J. Foster, "Introduction to wavelet transforms," *66th Ann. Internat. Mtg*, p. 2044, 1996.

[63] W. K. Ngui, M. S. Leong, L. M. Hee, and A. M. Abdelrhman, "Wavelet Analysis: Mother Wavelet Selection Methods," *Appl. Mech. Mater.*, vol. 393, pp. 953–958, 2013.

[64] T. Mcculloch, "2.0 Literature Review," no. 1958, pp. 5–38, 1986.

[65] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, no. 4, pp. 525–533, 1993.

[66] T. Falas and A. Stafylopatis, "Implementing temporal-difference learning with the scaled conjugate gradient algorithm," *Neural Process. Lett.*, vol. 22, no. 3, pp. 361–375, 2005.

[67] A. Singh, "A Study of Various Training Algorithms on Neural Network for Angle based Triangular Problem," vol. 71, no. 13, pp. 30–36, 2013.

[68] M. Dorofki, A. H. Elshafie, O. Jaafar, and O. a Karim, "Comparison of Artificial Neural Network Transfer Functions Abilities to Simulate Extreme Runoff Data," *Int. Conf. Environ. Energy Biotechnol.*, vol. 33, pp. 39–44, 2012.

[69] A. Pavelka and A. Procházka, "Algorithms for initialization of neural network weights," *Sb. Prisp. 12. rocniku Konf. MATLAB 2004*, vol. 2, pp. 453–459, 2004.

[70] F. S. Panchal and M. Panchal, "Review on Methods of Selecting Number of Hidden Nodes in Artificial Neural Network," *Int. J. Comput. Sci. Mob. Comput.*, vol. 3, no. 11, pp. 455–464, 2014.

[71] K. G. Sheela and S. N. Deepa, "Review on methods to fix number of hidden neurons in neural networks," *Math. Probl. Eng.*, vol. 2013, 2013.

[72] D. Hunter, H. Yu, M. S. Pukish, J. Kolbusz, and B. M. Wilamowski, "Selection of Proper Neural Network Sizes and Architectures — A Comparative Study," *IEEE Trans. Ind. Informatics*, vol. 8, no. 2, pp. 228–240, 2012.

[73] Wikipedia, "Different classifications of metaheuristics," *Wikipedia*, 2011. .

[74] M. Clerc, "Confinements and Biases in Particle Swarm Optimization," vol. 0, p. 9, 2006.

[75] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Neural Networks, 1995. Proceedings., IEEE Int. Conf.*, vol. 4, pp. 1942–1948 vol.4, 1995.

[76] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, 1998, pp. 69–73.

[77] S. S. Pattnaik, K. M. Bakwad, S. Devi, B. K. Panigrahi, and S. Das, "Parallel Bacterial Foraging Optimization," in *Handbook of Swarm Intelligence: Concepts, Principles and Applications*, B. K. Panigrahi, Y. Shi, and M.-H. Lim, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 487–502.

[78] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical

particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, 2004.

[79] Y. Liu, Z. Qin, and X. He, "Supervisor-student model in particle swarm optimization," in *Congress on Evolutionary Computation, 2004. CEC2004*, 2004, p. 542–547 Vol.1.

[80] G. R. Murthy, M. S. Arumugam, and C. K. Loo, "Hybrid particle swarm optimization algorithm with fine tuning operators," *Int. J. Bio-Inspired Comput.*, vol. 1, no. 1/2, p. 14, 2009.

[81] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, 2002.

[82] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," *Evol. Comput. 2000. Proc. 2000 Congr.*, vol. 1, no. 7, pp. 84–88, 2000.

[83] F. Van Den Bergh and A. P. Engelbrecht, "A convergence proof for the particle swarm optimiser," *Fundam. Informaticae*, vol. 105, no. 4, pp. 341–374, 2010.

[84] P. Kumar Patel, V. Sharma, and K. Gupta, "Guaranteed Convergence Particle Swarm Optimization using Personal Best," *Int. J. Comput. Appl.*, vol. 73, no. 7, pp. 6–10, 2013.

[85] C. Leboucher, H. S. Shin, P. Siarry, S. Le M??nec, R. Chelouah, and A. Tsourdos, "Convergence proof of an enhanced Particle Swarm Optimisation method integrated with Evolutionary Game Theory," *Inf. Sci. (Ny).*, vol. 346–347, pp. 389–411, 2016.

[86] X. S. Yang, "A new metaheuristic Bat-inspired Algorithm," in *Studies in Computational Intelligence*, 2010, vol. 284, pp. 65–74.

[87] A. M. TAHA and A. Y. C. TANG, "BAT ALGORITHM FOR ROUGH SET ATTRIBUTE REDUCTION.," *J. Theor. Appl. Inf. Technol.*, vol. 51, no. 1, p. 1, 2013.

[88] I. Fister, X.-S. Yang, S. Fong, and Y. Zhuang, "Bat algorithm: Recent advances," in *CINTI 2014 - 15th IEEE International Symposium on Computational Intelligence and Informatics, Proceedings*, 2014, pp. 163–167.

[89] "Optimal parameter settings for bat algorithm Fei Xue and Yongquan Cai Yang Cao , Zhihua Cui * and Feixiang Li," vol. 7, no. 2, pp. 2–5, 2015.

[90] X. S. Yang and S. Deb, "Cuckoo search: Recent advances and applications," *Neural Computing and Applications*, vol. 24, no. 1. pp. 169–174, 2014.

[91] D. Liu, D. Niu, H. Wang, and L. Fan, "Short-term wind speed forecasting using wavelet transform and support vector machines optimized by genetic algorithm," *Renew. Energy*, vol. 62, pp. 592–597, 2014.

[92] L. Wang, Y. Zeng, and T. Chen, "Back propagation neural network with adaptive differential evolution algorithm for time series forecasting," *Expert Syst. Appl.*, vol. 42, no. 2, pp. 855–863, 2015.

[93] R. S. Sexton, R. E. Dorsey, and J. D. Johnson, "Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing," *Eur. J. Oper.*

Res., vol. 114, no. 3, pp. 589–601, 1999.

[94] N. S. Jaddi, S. Abdullah, and A. R. Hamdan, "Optimization of neural network model using modified bat-inspired algorithm," *Appl. Soft Comput. J.*, vol. 37, pp. 71–86, 2015.

[95] S. Nandy, P. P. Sarkar, and A. Das, "Training Feed-forward Neural Network With Artificial Bee Colony Based Back-propagation Method," *Int. J. Comput. Sci. Inf. Technol.*, vol. 4, no. 4, pp. 33–46, 2012.

[96] C. Kim, I. Yu, and Y. Song, "Kohonen neural network and wavelet transform based approach to short-term load forecasting," *Electr. Power Syst. Res.*, vol. 63, no. 3, pp. 169–176, 2002.

[97] I. Fister, J. Brest, and X. S. Yang, "Modified bat algorithm with quaternion representation," *2015 IEEE Congr. Evol. Comput. CEC 2015 - Proc.*, pp. 491–498, 2015.

[98] B. Zhu, W. Zhu, Z. Liu, Q. Duan, and L. Cao, "A Novel Quantum-Behaved Bat Algorithm with Mean Best Position Directed for Numerical Optimization," *Comput. Intell. Neurosci.*, vol. 2016, 2016.

[99] R. Setiono, "Feedforward Neural Network Construction Using Cross," vol. 2877, pp. 2865–2877, 2001.

[100] B. Guoqiang Zhang and M. Y. H. Eddy Patuwo, "Forecasting with artificial neural networks: The state of the artF," *Int. J. Forecast.*, vol. 14, pp. 35–62, 1998.

[101] REN, "Load Profiles." [Online]. Available: http://www.mercado.ren.pt/EN/Electr/MarketInfo/Load/Pages/LoadProfiles.aspx. [Accessed: 17-Nov-2016].

[102] eia, "Demand for New England ISO (ISNE)." [Online]. Available: https://www.eia.gov/opendata/qb.php?sdid=EBA.ISNE-ALL.D.H. [Accessed: 02-Feb-2017].

[103] J. P. S. Catalão, H. M. I. Pousinho, and V. M. F. Mendes, "Hybrid wavelet-PSO-ANFIS approach for short-term electricity prices forecasting," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 137–144, 2011.

[104] Z. Tan, J. Zhang, J. Wang, and J. Xu, "Day-ahead electricity price forecasting using wavelet transform combined with ARIMA and GARCH models," *Appl. Energy*, vol. 87, no. 11, pp. 3606–3610, 2010.

[105] A. J. Conejo, M. A. Plazas, R. Espinola, and A. B. Molina, "Day-Ahead Electricity Price Forecasting Using the Wavelet Transform and ARIMA Models," *IEEE Trans. Power Syst.*, vol. 20, no. 2, pp. 1035–1042, 2005.

[106] REE, "Daily Market Hourly Price." [Online]. Available: http://www.omie.es/files/flash/ResultadosMercado.swf. [Accessed: 04-Apr-2017].

[107] C. García-Martos, J. Rodríguez, and M. J. Sánchez, "Mixed models for short-run forecasting of electricity prices: Application for the Spanish market," *IEEE Trans. Power Syst.*, vol. 22, no. 2, pp. 544–552, 2007.

[108] A. T. Lora, J. M. R. Santos, A. G. Exposito, J. L. M. Ramos, and J. C. R. Santos,

"Electricity Market Price Forecasting Based on Weighted Nearest Neighbors Techniques," *IEEE Trans. Power Syst.*, vol. 22, no. 3, pp. 1294–1301, 2007.

[109] N. Amjady, "Day-Ahead Price Forecasting of Electricity Markets by a New Fuzzy Neural Network," *IEEE Trans. Power Syst.*, vol. 21, no. 2, pp. 887–896, 2006.

[110] N. Amjady and M. Hemmati, "Day-ahead price forecasting of electricity markets by a hybrid intelligent system," *Eur. Trans. Electr. Power*, vol. 19, no. 1, pp. 89–102, 2009.

[111] N. M. Pindoriya, S. N. Singh, and S. K. Singh, "An Adaptive Wavelet Neural Network-Based Energy Price Forecasting in Electricity Markets," *IEEE Trans. Power Syst.*, vol. 23, no. 3, pp. 1423–1432, 2008.

[112] J. P. S. Catalão, H. M. I. Pousinho, and V. M. F. Mendes, "Neural networks and wavelet transform for short-term electricity prices forecasting," in *2009 15th International Conference on Intelligent System Applications to Power Systems, ISAP '09*, 2009.

[113] N. Amjady and F. Keynia, "Day-Ahead Price Forecasting of Electricity Markets by Mutual Information Technique and Cascaded Neuro-Evolutionary Algorithm," *IEEE Trans. Power Syst.*, vol. 24, no. 1, pp. 306–318, 2009.

[114] N. Amjady and A. Daraeepour, "Design of input vector for day-ahead price forecasting of electricity markets," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 12281–12294, 2009.

[115] N. Amjady and A. Daraeepour, "Mixed price and load forecasting of electricity markets by a new iterative prediction method," *Electr. Power Syst. Res.*, vol. 79, no. 9, pp. 1329–1336, 2009.

[116] O. Abedinia, N. Amjady, and H. Zareipour, "A New Feature Selection Technique for Load and Price Forecast of Electrical Power Systems," *IEEE Trans. Power Syst.*, vol. 8950, no. c, pp. 1–1, 2016.

[117] PJM, "Day-Ahead LMP." [Online]. Available: http://www.pjm.com/markets-and-operations/energy/real-time/monthlylmp.aspx. [Accessed: 01-Jan-2017].

[118] N. Amjady and A. Daraeepour, "Day-ahead electricity price forecasting using the relief algorithm and neural networks," *2008 5th Int. Conf. Eur. Electr. Mark.*, pp. 1–7, 2008.

Annex A

Articles

A Bat-inspired Neural Network Optimization Method with Wavelet Transform and Data Selection for Short-Term Load Forecasting, **Journal:** IEEE Transactions on Power Systems, **Impact Factor:** 3.342 (Submitted).

Abstract: Short-term load forecasting is very important for reliable power system operation. This paper presents a new enhanced method for one day ahead load forecast, combing data selection and features extraction techniques (similar/recent day-based selection, correlation and wavelet analysis), which brings more "regularity" to the load data series, an important precondition for the successful application of neural networks. A combination of Bat and Scaled Conjugate Gradient Algorithms is proposed to improve neural network learning capability. Another feature is the method's capacity to fine-tune neural network architecture and wavelet decomposition, for which there is no optimal paradigm. Numerical testing reveals the superiority of the proposed method, showing promising forecasting results in comparison with other state-of-the-art methods.

A Bat Optimized Neural Network and Wavelet Transform approach for Short-Term Price Forecasting, **Journal:** Elsevier international journal.

Abstract: In the competitive power industry environment, electricity price forecasting is a fundamental task when market participants decide upon bidding strategies. This has led researchers in the last years to intensely search for accurate forecasting methods, contributing to better risk assessment, with significant financial repercussions. This paper presents a hybrid method that combines similar and recent day-based selection, correlation and wavelet analysis in a pre-processing stage. Afterwards a feedforward neural network is used alongside Bat and Scaled Conjugate Gradient Algorithms to improve the traditional neural network learning capability. Another feature is the method's capacity to fine-tune neural network architecture and wavelet decomposition, for which there is no optimal paradigm. Numerical testing was applied in a day-ahead framework to historical data pertaining to Spanish and Pennsylvania-New Jersey-Maryland (PJM) electricity markets, revealing auspicious forecasting results in comparison with other state-of-the-art methods.