A^3 I

ARTIFICIAL^3 INTELLIGENCE

AN INANIMATE REASONER

Ph.D.THESIS

DEPARTMENT OF MATHEMATICS AND INFORMATICS
UNIVERSITY OF CATANIA

Christian Napoli
December 2015

≪*Tempus plantandi, et tempus evellendi quod plantatum est.*≫
Ecclesiastæ, 3.2

Contents

CHAPTER 1

Introduction

An architect, a hooker and a programmer were talking: "Everyone knows mine is the oldest profession," said the hooker. "But before your profession existed, there was not the divine architect of the universe?" The architect asked. Then the programmer spoke up: "and before an architect, what was there?" "Darkness and chaos," the hooker said. "And who do you think created chaos?" the programmer said.

Science is a hobby of mine as well as strolling, and such two hobbies do not differ so much after all. Science is the act of walking around concepts, theories, approaches and experiments by continuously getting lost along the paths of thoughts and streams of consciousness. Both such two hobbies of mine can be extremely satisfactory when you finally reach new and unknown places, nevertheless it has been said that the way itself should matter more than the destination, and I agree. As a matter of fact, along the trip that I am called to recap in this thesis, I've increasingly found

myself unable to write anything regarding my destination without accounting the path itself. Therefore, in the following pages, I will try to explain the destination of my researches, which is intriguingly contained in the title of this thesis, however not before I have asked for the reader's kindness in order to be followed along the quite loopy upcoming path. I make a promise though: every bit of information contained in such a path will be shown to be a mandatory piece to build our destination. As for the title of this work I am well aware that every name contains a promise, and my promise is to keep up with a challenge: trying to tackle with the nature itself of human intelligence, trying to model it, manage it, predict it. In the very end, only to ennoble it. Of course, after a fast scroll through the table of contents, it will be obvious that the linchpin of this work is all about crowdsourcing: the nature itself, the involved architecture, the manageability, the possible improvements and the related ethical consequences. The latter are the motif forces that drive the entireness of this work, on the other hand I will briefly discuss about them only at the very end of this thesis. One of the step to undertake to understand this work involves a new conception of crowd and crowd sourcing models, that in a certain portion of this work will be modeled similarly as clouds of some-how-connected brains that, not differently from hardware processors and computational nodes, are concurrently involved on the solution of a set of tasks. The definition is not occasional since also the major crowdsourcing platform, the Amazon's Mechanical Turk, has been defined by its founder as: an Artificial Artificial Intelligence. Despite that, in this thesis, I would like to improve and exceed the limits of such a definition which, as it will be unveiled, is not only literally disrespectful of the human nature of the Turkers, the back end workers of the Mechanical Turk, but also disagreeable in its profound implications. As for any other scientific work also this thesis is born from a couple of questions: is it possible to do it better? Is it possible to join both the clear advantages of crowdsourcing with a more agreeable consequences for the workers? Is it possible to obtain an automatism of some kind to model and manage such human worker crowds?

Those questions have been asked in a precise moment and in a precise place: the year is 2015, the place is the Lipari isle. The occasion to ask such questions and

elaborate on them was given by a fortunate meeting with prof. Michael Bernstein from Stanford University and actual inspirer for the title of this thesis. The conclusions we reached were few but important:

1. Crowdsourcing is based on the certainty of workers availability
2. A skill-based pyramidal structure tampers with the previous point
3. It is paramount then to model availabilities from human behaviors
4. The execution of a crowd-based workflow should become predictable
5. New approaches to crowd modeling and behavior prediction are required

Surprisingly, I discovered that the best part of the researches performed during and before my PhD studies, were complying with many of the key points needed for what prof. Bernstein was calling the "crowdsourcing revolution". At that point the path of this thesis simply unfolded by itself: the milestone where there, the rest was only a matter of connecting them. As said before, the path is what matters. During my studies and my personal growth I had the undeserved fortune to be led on a path crossing disciplines and methods, mixing Physics, Astrophysics, Cosmology, Engineering, Energetics, and, finally, Computer Science. This path gave me a personal perspective on chaotic phenomena that I tried to apply at the best of my possibilities to the topic of this thesis. The results were as surprising as unexpected, but, eventually, quite interesting. Modeling human beings is not an easy task and I would not have been up to this challenge if I would have only based my approach on my own ab initio assumptions. Instead of that I have tried to let the models emerge by themselves while applying some mathematical methods such as wavelet analysis and several advanced tools such as neural networks.

The developed methodologies and algorithms have been published or submitted to important international journals such as IEEE Transactions on Neural Networks, IEEE Transactions on Industrial Informatics, Neural Systems, Applied Energy, the International Journal of Mathematics and Computer Sciences, etc. Note that a portion of the mainstream concepts have been based on countless insights received while presenting our solutions to conferences such as the IEEE International Joint Conference

on Neural Networks, the International Conference on Artificial Intelligence and Soft Computing and IEEE Symposium Series on Computational Intelligence, for several editions. The help of the scientific community, its constant support, the comments and criticisms received, has been fundamental for the continuation and improvement of the presented research works. A special tribute of gratitude must be anyway recognized to prof. Włodzisław Duch, now deputy Minister for Science and Education in Poland, former head of the Department of Informatics at the Nicolaus Copernicus University, not only an esteemed scientist but also excellent teacher with whom several times I had the privilege to chat, in front of a cup of tea or sandwiches, and learn the fundamentals of Neural Network and Neurocognitive Informatics directly by one of the main contributors of such fields, vastly known in the scientific community for his computational approach to human mind modeling using Information Theory and Neural Networks.

As I will explain in Chapter 2 the possibility to play with chaos and extrapolate chaotic models take advantages of centuries of meditations and has very long roots. Taking advantage of the gigantic studies already performed by the greatest names of Science I have tried to interpret chaos and find a peculiar order in it starting with Information Theory and Wavelet Analysis. I will give the mathematical basis in Chapter 3. Wavelet analysis permits us to pack the information related to a phenomenon into a few numerical coefficients, this is the perfect kind of set you can try to model using neural networks. Of course to let the reader appreciate the advancement proposed in the field of Neural Networks I firstly propose some mathematical basis in Chapter 4, then I will explain in Chapter 5 how I have decided to enhance those tools by means of a personally developed new architecture called Wavelet Recurrent Neural Network, then an extended set of examples and ground tests will be given to the reader. The developed approach, especially when bound to specific mathematical formulations of the problem, among all the other proposed applications, gives us a way to obtain resources availability predictions beforehand in order to preserve the Quality of Service for distributed systems. As it will be shown, this approach has been shown to be extremely useful when applied, in a particular fashion, to workers availability for crowdsourcing projects. In Chapter 6 this thesis will finally enter on

the playground of the Mechanical Turk itself while a new conception of Artificial^2 Intelligence will be formulated by using Radial Basis Probabilistic Neural Networks as classifiers. The latter will be constituting optimal crowd clustering tools in order to understand, model and predict the behavior of human groups as well as their natural predisposition to connect in social or collaborative networks. The developed approach will be used for the specific purpose of this thesis, in fact in Chapter 7 it will be shown that it is possible to model and create workgroups of people in order to better accomplish a certain task basing on their characteristics, professional skills, experiences, etc. As a matter of fact it will be shown that it is possible to concentrate all those choices, generally made by a company manager, into an Inanimate Reasoner. Finally, this last frontier of automated company management based on Radial Basis Probabilistic Neural Network will be joined with the said Wavelet Recurrent Neural Networks, also providing an appropriate mathematical model and the related software system in order to model, manage and optimize the execution of complex workflows. Therefore in Chapter 8 the Artificial^3 Intelligence will be presented as an overall ensemble of mathematical and technological tools that jointly aims at making the announced "crowdsourcing revolution" possible. Due to the the complexity of the path it seemed reasonable to split it in steps, namely one for each chapter. At the end of each chapter therefore a specific section will try to summarize the meaning and implications of the chapter itself and lead the reader to the following chapter by highlighting the connections and motifs. Finally, in the remaining part of this thesis, and oppositely with respect to this Introduction and, eventually, the final Acknowledgments, the use of singular pronouns will be abandoned. In fact, despite of the traditions and rules which compel me to write only my name as author of this thesis, none of the following content would have been possible without the key contribution of a lot of other co-authors which populate part of the bibliography nearing my name. Therefore, from this moment I will use the plural noun "we", since this is not a one man work but the non-linear superposition of contributions coming from an active collaboration of which I am only a very small part. Enjoy the reading.

CHAPTER 2

Bringing chaos into order

> Chaos: When the present determines the future, but the approximate present does not approximately determine the future.
>
> Edward Norton Lorenz

In 1795, by a decree of the National Convention, a new professor of mathematics was called to join the École Normale de Paris. He was named Pierre Simon, marquis de Laplace, but he is now universally known as Laplace. Laplace was also recognized for his so called newtonian fever and he left no occasion to remark his deep beliefs on the deterministic nature of the Universe. Therefore, the words he pronounced during his Course of Probability (reported on A Philosophical Essay on Probabilities, [118] one of his most famous writings) are not surprising: ≪We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes.≫ These are known facts, as well as the invitation that the

same École Normale de Paris extended in 1797 for the same position, yet held also by Laplace, to Giuseppe Lodovico Lagrangia, also known as Joseph-Louis Lagrange. Laplace and Lagrange soon established a deep bound of respect and friendship, on the other hand, as Cournot let us know [54], the two mathematicians were also well known for the strength of their verbal fightings. During one of such verbal fights Lagrange pronounced his famous sentence: «I regard as quite useless the reading of large treatises of pure analysis: too large a number of methods pass at once before the eyes. It is in the works of applications that one must study them; one judges their ability there and one apprises the manner of making use of them.». Also the words of Lagrange are not surprising, in fact Marcus Du Sautoy describes him pronouncing similarly strong words [65] during a successive episode. The mentioned episode recalls both Lagrange and Laplace while participating to an high level cultural gala in the Palais du Luxembourg. During this gala a colleague senator, Cauchy his name, while accompanied by his very young son, met the mathematicians. After an interesting talk with the senator's son, Lagrange gave further use of his strong speaking, addressing the other numerous dignitaries of France, while pointing his fingers to the young boy in a corner: «See that little young man? Well! He will supplant all of us in so far as we are mathematicians», and then addressing his father «Don't let him touch a mathematical book till he is seventeen [...] if you don't hasten to give Augustin a solid literary education his tastes will carry him away» [18]. Augustin Louis Cauchy was indeed the name of the boy: "the" Cauchy, father of modern Math. Was Lagrange saying that mathematical education was not important in XIX century? Of course not. Lagrange was well aware that great changes would suddenly reform the basis of mathematics, logics and sciences. The optimistic view of Laplace was in facts definitively doomed: the scientific community was beginning to recognize that, despite the perfection and elegance of pure analysis, it was not sufficient to describe the rounding nature and its phenomena. The scientist, under the newtonian assumption of a deterministic world, were convincing themselves that starting by an approximated knowledge of the initial status of a dynamic system, it would have been possible to determine its temporal evolution. This philosophical assumption was at the very core of XIX century science revolution. Despite of that

assumption, in fact, there were increasingly strong evidences of systems that were not describable by the existent analytical laws which, while able to describe the vastness of planetary motion, where completely falling apart when trying to the describe the motion of dust.

2.1 Building chaos

In 1870s finally the word chaos was legitimately admitted in the scientific dictionary thanks to Ludwig Boltzmann and James Clerk Maxwell and their theory of statistical thermodynamics. Their molecular chaos assumption postulated that during a two-body collision, between particles in an ideal gas system, there is no correlation of velocity. This assumption allowed us to develop a molecular chaos theory of bodies in gas phase. As a side effect the theory finally introduced the concept of chaotic dynamics. Since that moment the most important scientists contributed to the foundation of chaos theory: from Poincaré to Hadamard, from Lorenz to Kolmogorov. The latter, Andreï Nicolaïevitch Kolmogorov, can be surely considered the legitimate father of modern chaos theory due to his revision of the theories yet developed by Poincaré [112]. Kolmogorov demonstrated that a quasiperiodic regular motion can persist in an integrable system even when a slight perturbation is introduced. This statement is also know as the KAM theorem due to the initials of the authors, Kolmogorov, Arnold and Moser, who independently reached the same conclusions [112, 140, 197]. While the intentions of the authors were to indicate the limits to the integrability of map functions, this theorem also gives the basis to understand the transition of a stationary deterministic system into a chaotic state. As a matter of fact in an integrable system all the possible status transitions are regularly quasi periodic and introducing a neglectable perturbation with respect to the system, the quasi periodic properties are not affected. On the other hand, as shown by the KAM theorem, if we consider an increasingly strong perturbation, the probability to affect the quasi periodicity of the system increases until chaotic trajectories are developed and a totally chaotic status is reached. In this latter stage the constants of motion are not preserved excepted the total energy, for this reason the consequences of the KAM

theorem are also generally called ergodic principle. Despite its apparent strangeness with respect to the topic of this thesis, the ergodic principle gives us the very deep meaning of this work when applied to complex systems. In a system describable by means of linear equations, an effect is simply a sum of a certain number of causes in a totally deterministic fashion, accordingly to the principle of causality postulated by René Descartes. On the other hand, such equations are rarely found to be accurate when trying to describe a natural phenomenon in detail due to its intrinsic discontinuous or non stationary nature. The latter requires more complex models using non linear solutions in order to reach an accurate enough description of the physical laws. Unfortunately, the non linearity of such systems makes it impossible to preserve the causality principle as described by Descartes since the effect of a small interaction is not anymore accountable by such mathematical models. Non linear models, as developed in the XIX century, were often not robust enough with respect to small interactions, therefore small variations of the initial conditions can lead them to non deterministic states.

2.2 Chasing butterflies

While the KAM theorem and the ergodic principle gave the basis to understand chaos, also due to the works of their predecessors, the official discoverer of chaos is considered Edward Norton Lorenz with his work on deterministic nonperiodic flow [124]. Ironically, the bases of his theory were discovered by chance in 1961 while playing with different approximations with the aim to produce accurate weather forecast with a quite primitive computer. It is reported in literature [86] the famous episode of Lorenz obtaining different solutions to his algorithm basing on the different approximations (3 or 6 decimal digits). The problem, discovered Lorenz, was not the approximation capabilities of the machine, but the non linear nature of the system that, due to small multiplication errors at each iteration, was suffering of an induced exponentially divergent trajectory in the state space. In 1972, Philip Merilees, chairman of the International Conference of the Association for the Advancement of Science, invited Lorenz to present his work, but was Merilees itself to decide the

title. The paper was entitled "Predictability: Does the Flap of a Butterfly's wings in Brazil Set off a Tornado in Texas?" [123], it contained the formulation of what is commonly called the butterfly effect. Finally, in 1975, Edward Norton Lorenz produced the results of a computer simulation graphically describing the divergence effect caused by approximation, and suddenly he understood that the simulations should have been reversible. The consequent results contributed to another discovery: the Lorenz's attractors. These attractors were immediately defined by Lorenz itself "strange attractors" due to their shape and evolution in double spirals like the wings of a butterfly. The comprehensive theory including Lorenz's discoveries will then be called with the name of Chaos Theory by James A. Yorke [120]. Yorke will say that Lorenz's attractor was capable to devise the intrinsic structure and the stability of systems that apparently show no stability or structure. On the other hand no one was yet aware of the real capabilities offered by the newborn theory.

2.3 Chaotic influences

As shown by Yorke, Lorenz, Packard, Mandelbrot, and many others it was possible to devise structures, recurrences and mathematical properties from a multitude of apparently indeterministic phenomena. Chaos theory was shedding light on a new interpretation of nature's complexity: a natural order describing the interaction of forces and evolution of states that, despite of the intrinsic complexity of such phenomena, could be defined by fairly simple solutions. In such an interpretation a stationary phenomena is nothing more than a stable and totally deterministic system, therefore predictable, where even a small interaction can lead to non deterministic states, therefore a non predictable behavior. The key factor for such a kind of systems is the minimum amplitude that a small interaction should have in order to break the linearity of the system. The first to take into account such a question was Henry Bérnard in 1901 while studying the water movements when exposed to an heat source: he discovered that, while heating the water, it exists a critical temperature after which the water begins to move in small vertical independent cells of fluid. Unfortunately Bérnard, while responsible of describing the phenomenon, was

not able to explain it, in fact an explanation was firstly given after sixteen years by John William Strutt, baron of Rayleigh, that we universally know as Lord Rayleigh. It must be said that the given explanation was technically wrong since Rayleigh was basing its interpretation on slightly different conditions, on the other hand the baron had caught the general reasons at the basis of the phenomenon: during its heating process, the water undergos a reckless battle between viscosity and heat. The first is responsible of maintaining the original uniformity of status and no coherent motion, the second is responsible of breaking the original status introducing a collection of states in uniform motion. Rayleigh understood that such dynamic systems are generally lead by two kinds of forces concurrently trying to impose a configuration by eliminating the other one. Therefore it exists a minimum perturbation amplitude after which the new forces are capable to win the inertia of the system, driving it away from its original configuration and letting its steady and uniform status to undergo a chaotic transition. Fortunately chaos theory can provide us the mathematical tools to build a perturbative model. On the other hand, while it is possible to use chaos theory to model such physical phenomena, is it possible to use the same approach for far more complex system such as a social system? The field of social systems involves far more complex phenomena with respect to any other system. Certainly, as the history of Internet and communication networks shows, social systems do not share the same geometrical properties and spatial symmetries of a pot of water on the fire. On the other hand, such a very complex social system manifests a different kind of order of its own, a set of symmetries that must be extrapolated and that are not evident at a first sight. Bérnard experiment shows that a chaos can emerge from order, but, as Rayleigh has proven, also from chaos a peculiar kind of order can emerge, but only after a chaotic transition. In order to understand a social system then, it is paramount to understand its chaotic transitions. As Mark Buchanan suggests [32], a typical example of chaotic transition is the explosion of a riot: two man have a fight into a bar, someone else get involved, and so on, until the bar is on fire as well as neighboring shops and hundreds of policemen are fighting against hundreds of hooligans for several weeks (Bradford, April 2001). Despite the cultural and psychological interpretations of the single phenomenon, the answer should be

searched elsewhere. In 1978 Mark Granovetter, working with Christopher Winship, Douglas Danfort and Bob Philips, mathematically demonstrated that models of collective behavior are developed for situations where actors have alternatives and the cost or the benefits of each depend on how many other actors choose which alternative [88]. The key concept is that it exists a threshold for the number of actors choosing the same alternative, this threshold acts as a transition point from the steady equilibrium status to a chaotic evolution until a new status is reached. This threshold is of course different for each individual, on the other hand it is also susceptible of the influences of the collective in which the individual is integrated. Granovetter also demonstrates that a small variation on such an individual threshold for one of the actors can have a profound influence on the entire group. Such influences have been proven to be both hierarchical and iterative processes. The first property results more evident in the social context of companies or working environments, the second in groups of people with common interests. In the year 2000 Malcom Gladwell proved [85] that viral content gets amplified from an individual to another creating a series of iterations and reaching out to the masses. In other words it is possible to build maps of influence starting from a main influencer and building hierarchical or interest based relationships to his second grade influencers and so on. It means that chaotic transitions in social systems are started by influences mappable in peculiar kinds of networks: the small world or scale free networks.

2.4 A small world

In 1998 an article appears in the international journal Nature showing a simple network model that can be tuned to introduce increasing amounts of disorder showing that such systems can be highly clustered while preserving a small characteristic path lengths like random graphs [199]. The authors of the article are Duncan Watts and Steven Strogatz, and they call those systems small world networks, by analogy with the small-world [135] phenomenon (popularly known as six degrees of separation). The neural network of the worm Caenorhabditis elegans, the power grid of the western United States, and the collaboration graph of film actors are shown in the article

to be small-world networks. Such networks are represented by the authors as mathematical graph where very few nodes are neighboring each others, while in few hops it is possible to reach any destination node starting from any other node. In this kind of networks the distances in terms of numb of hops $\delta(n_0, n_1)$ among two nodes n_0 and n_1 is generally proportional to the logarithm of the size of the network itself, defined as the number of nodes N, therefore

$$\delta(n_0, n_1) \propto \log N \tag{2.1}$$

This so called small world effect is particularly noticeable in many social contests such as social networks, collaborative networks, companies and workgroups. The underlying structure of the Internet itself shows to be a fairly evident small world network. Successively, in 2002, Reuven Cohen and Shlomo Havlin have shown that some yet known small world networks, in particular, shares a peculiar property: their degree distribution follows a power law, at least asymptotically [52]. In other words, in such particular networks, considering the nodes retaining a number k of connections as a fraction $P(k)$ of the total number of nodes in the network, this fraction $P(k)$ typically [43] scales as a power law

$$P(k) \propto k^{-\gamma} \tag{2.2}$$

where generally $\gamma \in [2, 3] \subset \mathbb{R}$. Cohen and Havlin have also shown that the average distance between nodes in scale free networks is much smaller than that in regular random networks. Moreover, their studies on percolation in scale free networks demonstrates that while a chaotic transition generally occurs only in the limit of extreme dilution, several nodes are often critical to determine the evolution of the entire network. Finally, in [51] Cohen and Havlin demonstrate that scale free networks are ultra-small worlds. Speaking of a whole network in its entirety, in a small-world network if we define a diameter d as

$$d = \max_{n_0, n_1 \in V} \{\delta(n_0, n_1)\} \tag{2.3}$$

where V represents the set of nodes composing the network, then, due to (2.1), it immediately follows that

$$d \propto \log N \tag{2.4}$$

On the other hand, as the authors demonstrated, scale free networks within the constraint $\gamma \in [2, 3] \subset \mathbb{R}$ have a much smaller diameter with

$$d \propto \log \log N \tag{2.5}$$

While the derivation is valid for uncorrelated networks, the authors also detail that for assortative networks the diameter is expected to be even smaller. This latter kind of networks was described by Newman in [160] where he gives the definition of assortative network as a network where the nodes that have many connections tend to be connected to other nodes with many connections. Newman also finds that social networks are mostly assortatively, while technological and biological networks that tend to be disassortative, moreover he finds that assortative networks percolate more easily being also more robust to vertex removal. These considerations paint a unique picture: social networks are stable and resilient and therefore prone to spread influences among nodes, whether we are talking of diseases in a city, information in an online social network, interactions among workers of a company, or influences among users or maintainers of a service. Influence spreading on a social network have therefore peculiar patterns that guarantee the outcome. A question remains: is it possible to model such patterns and then predict the behavioral evolution of a social network over time? Again a bit of chaos theory, and in particular the ergodic principle, should be taken into account. If, in fact, as said before, chaotic transitions in social system are started by influences, and given that such influences resulted mappable as scale free networks, it seems legitimate to ask if it is possible to study social networks in terms of chaotic transitions. In this latter scenario two elements must be defined: the influences leading to such transitions, and a critical threshold driving the system. From this point the next step is obvious, since there exists only one considerable kind of influences flowing uninterruptedly in a social network: information. As a matter of fact, it has been proven many times that information dynamic does not

differ from epidemic dynamic and that there is no difference between the spreading of a pathogen or the diffusion of an idea, the models are similar, both of them non linear, both of them chaotic, both of them ergodic. But in order to define things such as transitions, thresholds and system energy, it is imperative to quantify by means of a measurement. It is then imperative to measure information.

2.5 All form is formless

While trying to analyze social networks in terms of information flows is an inevitable starting point, it is also a comforting arrival point that closes the circle of chaos theory. As a matter of fact, it would be impossible to understand any aspect of chaos theory without considering its interpretation under the lights of information theory. This latter, on the other hand, is based on a primitive concept: information. Due to the primitivity of this concept, it is quite difficult to give an appropriate definition of information. Generally speaking, in order to measure information, it would seem necessary to give an a priori definition of the form in which information is retained. Fortunately, whether or not information and its form can be properly defined, information can certainly be measured and algebraically represented. As well as Lorenz can be considered the legitimate author of Chaos Theory, the same role, relatively to Information Theory, should be assigned to Claude Elwood Shannon, but without forgetting that his work has been based on the previous studies of Gibbs, Boltzmann and a lot of other predecessors. Shannon's work [183] explains that there is no real need to know the form as far as it is possible to define a proper mathematical domain describing the information flow. The Shakespeare's character of Cardinal Pandulph would say that «all form is formless, order orderless», as he does in the third act of King John, unwillingly giving us a perfect definition of information: a structure-less structure. If it is in fact impossible to obtain an a priori definition of information, on the other hand it is quite easy to measure it once its mathematical definition has been given. Shannon's work not only gives that, but he also assigns a prediction probability, nowadays known as Shannon's entropy, in order to universally define the information content. In particular, given a content carrying information,

and known a certain portion of such a content, we could want to immagine the rest of the message. If we assign a probability p_i to each possible outcome, we can measure the information carried by that outcome as

$$H = -\sum_i p_i \log_2 p_i \tag{2.6}$$

The consequence is quite obvious as its interpretation: if we are able to completely determine with total certainty the outcome without having seen it, then its information content is zero [184]. As a matter of fact, we have no new information after the outcome becomes known, and, from this point of view, this kind of information measures tell us how much unexpected is an outcome. Therefore it is the quantity of information characterizing our a priori knowledge that affects the quantity of information we can get from the outcome. Certainly this aspect of Shannon's entropy should affect also our mathematically exact analytical models, as well as any other reproducible algorithm. As a matter of facts, given detailed enough initial conditions, a mathematical model is able to predict one and only one result with an outcoming probability of 1. Shannon's interpretation of such a model is simple and terrible: the model predicts nothing. In facts no information is predicted by an analytical model, due to its certainty, therefore all the information must be contained in the initial data. It is only well hidden. Then, if wherever it is possible to formulate a working mathematical set of laws and conditions capable to predict the future outcome of a phenomena starting by a data set of initial conditions and boundary conditions, it means that all the information is contained in such a data set, the next question can be only one. Where is such information hidden? The reason to bind wavelet analysis with neural networks is contained in the last question as well as in this chapter. While the historical importance of chaotic models has been shown, we hereby propose a new methodology to cope with non linear phenomena without imposing any constraint or mathematical model ab initio. Wavelet analysis will permit us to obtain a different representation of the data. Such a representation will be more suitable for neural networks which will be used to cope with chaotic systems due to their robustness to noise and misleading data. By binding the said two approaches it will be shown that

it is possible to model the chaotic behavior of non linear systems, then allowing us to tackle with the nature itself of human behavior in terms of chaotic transitions and information measurements. The purpose is to let the models be born from the experimental data themselves making no unjustified assumption. This will be possible only thanks to the high versatility of the used mathematical and computational tools as well as their yet proven suitability for chaotic models.

CHAPTER 3

Bringing order into chaos

> Chaos was the law of nature.
> Order was the dream of man.
>
> Henry Adams

The wavelet analysis give us a powerful tool to achieve major improvements on information-based representation of data especially when such data are affected by an apparently chaotic behavior. In general wavelet decomposition is used to extract a shortened number of non-zero coefficients from a signal representative of the phenomenon. In this manner the wavelet analysis can be used in order to reduce the data redundancies so obtaining representation which can express their intrinsic structure. The main advantage of the wavelet use is the ability to pack the energy of a signal, and in turn the relevant carried informations, in few significant uncoupled coefficients. In particular the performance of the wavelet decomposition can be noticed for non-linear dynamical systems and predictors [57]. The wavelet transform, in fact, packs the energy of a signal reducing the redundancies and showing the intrinsic structure in time and frequencies. The so obtained representation with wavelet decomposition offer advantages while used with neural networks, as will be introduced in the next chapters. For now we will give a general description of the mathematical support.

3.1 Wavelet theory

A general isomorphic linear application on a metric field, e.g. $\mathbb{R}$, which is able to describe a signal as a series expansion of waveforms, e.g. the Fourier expansion, is called linear time-frequency transform. The waveforms involved in such a transform are called time-frequency kernels and can be used as basis elements of an Hilbert space. Considering a signal represented by a function $f : \mathbb{R} \to \mathbb{R}$, and $f \in L^2(\mathbb{R})$, it is possible to define a transform as a bijection $T : L^2(\mathbb{R}) \to L^2(\mathbb{R})$ so that

$$\begin{cases} T[f(t)] &= \xi(t) \\ T^{-1}[\xi(t)] &= f(t) \end{cases} \tag{3.1}$$

where $T^{-1} : L^2(\mathbb{R}) \to L^2(\mathbb{R})$ represents the inverse transform.

In order to understand the (3.1) it is necessary to recall few elements of algebraic and functional analysis. Let S^n be a n-dimensional a linear space, therefore a set of n linearly independent elements $\{x_k \in S^n\}_{k=1}^n$ form a basis for that space. Moreover, since S^n is a finite-dimensional linear space, it can be equipped with an inner product. Such a product allow us to obtain an orthonormal basis of S^n from any other basis using the Gram-Schmidt process [187]. Finally any basis $\{v_k \in S^n\}_{k=1}^n$ for S^n is the image under an invertible linear transformation of an orthonormal basis of S^n. In a more extensive scenario, taking into account also infinite-dimensional spaces, let S_H be an Hilbert space, or, in other words, let S_H be an abstract vector space possessing the structure of an inner product that allows lengths and angles to be measured and that, consequently, is also a complete metric space with respect to the distance function induced by the inner product [14].Under the said conditions, a collection of vectors $\{x_k \in S_H\}_{k\in\mathbb{Z}}$ is a Riesz basis for S_H if it is the image of an orthonormal basis for S_H under an invertible linear transformation. In other words, if there is an orthonormal basis $\{e_k\}_{k\in\mathbb{Z}}$ for S_H and an invertible transformation M such that

$$Me_k = x_k \quad \forall\ k \in \mathbb{Z} \tag{3.2}$$

Finally, if $\{x_k \in S_H\}_{k\in\mathbb{Z}}$ is a Riesz basis for S_H then there is a unique collection

$\{\tilde{x}_j \in S_H\}_{j\in\mathbb{Z}}$ such that $\langle x_k|\tilde{x}_j\rangle = \delta_{kj}$, where δ_{kj} is the Kronecker delta. Such collection $\{\tilde{x}_j \in S_H\}_{j\in\mathbb{Z}}$ is called biorthogonal basis with respect to $\{x_k \in S_H\}_{k\in\mathbb{Z}}$ for S_H, and it is also a Riesz basis of S_H. If $\{x_k \in S_H\}_{k\in\mathbb{Z}}$ and $\{\tilde{x}_k \in S_H\}_{kj\in\mathbb{Z}}$ are Riesz basis forS_H then there are non-negative constants α and β with $\alpha \leqslant \beta$ such that

$$\begin{aligned} \forall\ x \in S_H \Rightarrow \alpha||x||^2 \leqslant \sum_{k\in\mathbb{Z}} |\langle x|x_k\rangle|^2 \leqslant \beta||x||^2 \\ \forall\ x \in S_H \Rightarrow \alpha||x||^2 \leqslant \sum_{j\in\mathbb{Z}} |\langle x|\tilde{x}_j\rangle|^2 \leqslant \beta||x||^2 \end{aligned} \tag{3.3}$$

The (3.3) si also called Cauchy-Schwarz frame inequality for Hilbert spaces [44]. It is important to highlight that if $\{x_k \in S_H\}_{k\in\mathbb{Z}}$ and $\{\tilde{x}_k \in S_H\}_{k\in\mathbb{Z}}$ are orthonormal basis then the biorthogonality condition is obvius (e.g. taking $\tilde{x}_k = x_k$) and the (3.3) is trivially verified by means of the Plancherels formula with $\alpha = \beta = 1$. In fact if $\{x_k \in S_H\}_{k\in\mathbb{Z}}$ and $\{\tilde{x}_k \in S_H\}_{k\in\mathbb{Z}}$ satisfies the frame inequality with $\alpha = \beta = 1$ and if $|x_k| = |\tilde{x}_k| = 1 \ \ \forall\ k \in \mathbb{Z}$, then $\{x_k \in S_H\}_{k\in\mathbb{Z}}$ and $\{\tilde{x}_k \in S_H\}_{k\in\mathbb{Z}}$ are biorthonormal basis for S_H. A different way to characterize some of these properties is to think of two operators T and T^{-1} associated to the Riesz basis $\{x_k \in S_H\}_{k\in\mathbb{Z}}$. The first can be called analysis operator $T : S_H \to L^2$ given by $T(x) = \{\langle x|x_k\rangle\}_{k\in\mathbb{Z}}$; the second can be called synthesis operator $T^{-1} : L^2(\mathbb{R}) \to S_H$ given by $T^{-1}(T(x)) = x$. In this formalism, $\{x_k \in S_H\}_{k\in\mathbb{Z}}$ is a Riesz basis if and only if T is a bounded linear bijection from S_H onto L^2, but then it follows that $S_H = L^2(\mathbb{R})$. It follows then it is possible to define a transform as a bijection $T : L^2(\mathbb{R}) \to L^2(\mathbb{R})$ so that, given a signal $f : L^2(\mathbb{R}) \to L^2(\mathbb{R})$,

$$\begin{cases} T[f(t)] &= \xi(t) \\ T^{-1}[\xi(t)] &= f(t) \end{cases}$$

where $T^{-1} : L^2(\mathbb{R}) \to L^2(\mathbb{R})$ represents the inverse transform, coming back to the (3.1).

Consider now a complete generator set for $L^2(\mathbb{R})$ and let the elements of such a set be time-frequency kernels $\{\phi_\gamma\}_{\gamma\in\Gamma}$ where ϕ represents a multi parametric function of index γ and Γ is the indexing frame. Moreover let it be $\phi_\gamma \in L^2(\mathbb{R})$ with $||\phi_\gamma|| = 1$ so that the set $\{\phi_\gamma\}_{\gamma\in\Gamma}$ is a Rietz basis for $L^2(\mathbb{R})$. It is then possible to define a

time-frequency transform $T_\gamma : L^2(\mathbb{R}) \to L^2(\mathbb{R})$ as

$$T_\gamma [f(t)] = \langle f | \phi_\gamma \rangle = \int_{-\infty}^{+\infty} f(t)\phi_\gamma^*(t)dt \tag{3.4}$$

For such a transform, thanks to the Parseval theorem it is possible to demonstrate that

$$T_\gamma [f(t)] = \int_{-\infty}^{+\infty} f(t)\phi_\gamma^*(t)dt = \frac{1}{2\pi}\int_{-\infty}^{+\infty} f(\omega)\phi_\gamma^*(\omega)d\omega \tag{3.5}$$

Then it is possible to decompose the signal function f into a continuous superposition of γ-intervals both in the time domain and in the frequency domain. Therefore if the value of $\phi(t)$ is near to zero for any t in a small interval around a certain value t_u, then $\langle f(t) | \phi_\gamma \rangle$ will depend only from the value of $f(t)$ on the selected interval. Analogously if $\phi_\gamma(\omega)$ is near to zero for any ω in a small interval around a certain frequency ω_u, then the second member of the (3.4) proofs that $\langle f(\omega) | \phi_\gamma \rangle$ reveals the properties of $f(\omega)$ around ω_u. The two properties yet explained have tremendous consequences for the field of signal analysis since, due to this theorem, it is possible to construct a short time Fouriers transform by means of a small subdomain g, called shifting window, and then translate it by a shift t_u and modulating it by a frequency ω_u so that

$$\phi_\gamma(t) = g_{\omega_u t_u}(t) = e^{i\omega_u t_u} g(t - t_u) \tag{3.6}$$

Similarly, it is possible to construct a wavelet kernel by means of a scaling factor s and a shifting factor t_u starting from a motherwavelet function $\psi :: L^2(\mathbb{R}) \to L^2(\mathbb{R})$ so that

$$\phi_\gamma(t) = g_{st_u}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t - t_u}{s}\right) \tag{3.7}$$

This latter is also called time-frequency scaling and shift relation. By using the wavelet kernels defined in (3.7) it is then possible to transform the space $L^2(\mathbb{R})$ by means of a complete Rietz basis $\{\phi_\gamma\}_{\gamma \in \Gamma}$ by varying on a specific set of scaling and shift parameters for each ϕ_γ. The main advantage of such a decomposition is the possibility to identify precise and specific locations in the time-frequency domain where

the signal amplitude is distributed. In this manner it is possible to associate to each signal in $L^2((R)$ an unique energy distribution known as time-frequency signature of the signal in the wavelet domain. Such a signature it is often more useful with respect to the classical spectrum resulting from a Fourier transform (continuous or discrete), since this latter only decompose a signal in limited frequency bands. By using a time-frequency representation such as the wavelet decomposition a greater number of information can be carried. It is finally possible to reconstruct the original information with the time-frequency product $\langle f|\phi_\gamma\rangle$ represented on the time-frequency plan $\{(t,\omega)\} \subset \mathbb{R}^2$ as a geometrical region. The position and surface of such a region depend by the time-frequency spread of the kernel ϕ_γ and therefore are mainly determined by the wavelet scaling factor s and shift factor t_u. One of the properties of a motherwavelet is the unitary norm, therefore

$$||\phi_\gamma(t)||^2 = \int_{-\infty}^{+\infty} |\phi_\gamma|^2 dt = 1 \tag{3.8}$$

It this the possible to cope with $|\phi_\gamma(t)|^2$ as with a probability distribution centered on a point $t_{u\gamma}$ defined as

$$t_{u\gamma} = \int_{-\infty}^{+\infty} t|\phi_\gamma|^2 dt \tag{3.9}$$

following this mental schema interpreting $|\phi_\gamma(t)|^2$ as a probability distribution, it is also possible to define a standard deviation and consequently to obtain the definition of a variance

$$\sigma^2_{t_u}(\gamma) = \int_{-\infty}^{+\infty} (t - t_{u\gamma})^2 |\phi_\gamma(t)|^2 dt \tag{3.10}$$

It is then possible to recognize $\sigma^2_{t_u}(\gamma)$ as the time-frequency spread of the kernel ϕ_γ associated to the scaling factor s and the shift factor t_u. By the Plancherel theorem

it is possible also to prove

$$\sigma_{t_u}^2(\gamma) = \int_{-\infty}^{+\infty} |\phi_\gamma|^2 d\omega = 2\pi ||\phi_\gamma||^2 \tag{3.11}$$

It is then possible to represent an equivalent of (3.9) on the frequency domain. This point is the central frequency of ϕ_γ and is defined as

$$\omega_{u\gamma} = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \omega |\phi_\gamma(\omega)|^2 d\omega \tag{3.12}$$

similarly it is possible to represent the time-frequency spread also on the frequency domain as

$$\sigma_{\omega_u}^2(\gamma) = \int_{-\infty}^{+\infty} (\omega - \omega_{u\gamma})^2 |\phi_\gamma(\omega)|^2 d\omega \tag{3.13}$$

It is now clear how important is the time-frequency spread associated with a certain wavelet decomposition starting from the chosen wavelet kernels $\{\phi_\gamma\}$: such a spread determines the resolution in time and frequency of each wavelet kernel. Basing of such resolution the wavelet transform is able to devise the time-frequency signature of a signal by characterizing the energy distribution for each one of the different time and frequency resolutions resulting from the wavelet kernels. For a given time-frequency plan $\{(t, \omega) \in \mathbb{R}^2\}$, each wavelet kernel ϕ_γ selects a region $\Omega_\gamma \subset \{(t, \omega) \in \mathbb{R}^2\}$ called Heisemberg box and defined as

$$\Omega_\gamma = \left[t_{u\gamma} - \frac{\sigma_t^2(\gamma)}{2}, t_{u\gamma} + \frac{\sigma_t^2(\gamma)}{2} \right] \times \left[\omega_{u\gamma} - \frac{\sigma_\omega^2(\gamma)}{2}, \omega_{u\gamma} + \frac{\sigma_\omega^2(\gamma)}{2} \right] \subset \mathbb{R}^2 \tag{3.14}$$

Finally it is possible to prove [200] the existence of a minimum for such a region and, due to the (3.8) it results [128] a typical Heisemberg's formula

$$\text{surf}(\Omega_\gamma) = \sigma_t^2(\gamma)\sigma_\omega^2(\gamma) \geqslant \frac{1}{2} \tag{3.15}$$

Since the time-frequency resolution of the related ϕ_γ is defined as the surface Ω_γ, it

follows that, differently with respect to the Fourier transform, it is not possible to obtain a point to point bijective correlation between a signal and its wavelet transform since the time-frequency signature of such a signal is not concentrated in a point but results from the superposition of different scales (therefore a different scaling factors) at different time and frequencies (different values of t_u and ω_u) at different time-frequency resolutions. In an over-simplified example, let suppose that for a pair $(t_u, \omega_u) \in \{(t, \omega)\} \subset \mathbb{R}^2$ we use an unique wavelet kernel $\phi_{\gamma_{(t_u,\omega_u)}}$ centered in (t_u, ω_u), then it is possible to obtain the related time-frequency resolution $\Omega_{\gamma_{(t_u,\omega_u)}}$. The obtained $\Omega_{\gamma_{(t_u,\omega_u)}}$ is a rectangular neighborhood of (t_u, ω_u) where it is possible to locate a portion of the energy of the signal f as

$$\varepsilon\left[f(\Omega_{\gamma_{(t_u,\omega_u)}})\right] = \left|\langle f|\phi_{\gamma_{(t_u,\omega_u)}}\rangle\right|^2 = \left|\int_{-\infty}^{+\infty} f(t)\phi^*_{\gamma_{(t_u,\omega_u)}}(t)dt\right|^2 \tag{3.16}$$

3.2 Wavelet transform

As devised in the previous section, transforming the $L^2(\mathbb{R})$ space by using a Rietz basis of time-frequency kernels allow us to let different properties of a signal emerge in the time-frequency plan. Since different signals can be characterized by very different properties both in time and frequency domains, it is therefore agreeable that a lot of different time-frequency kernels basis can be used and customized in order to analyze specific kinds or families of signals. A quite sizable group of time-frequency decompositions constitutes the wavelet transform family. A wavelet transform decomposes a signal transforming the space with a specific basis computed starting by a common motherwavelet function properly shifted and scaled. Specifically we call wavelet a linear function $\psi : L^2(\mathbb{R}) \rightarrow L^2(\mathbb{R}$ quadratically summable and with zero average and belonging to a class of γ-neighborhoods centered on a point $t_u = 0$. Then it follows that

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0 \tag{3.17}$$

$$\int_{-\infty}^{+\infty} |\psi(t)|^2 dt = 1 \tag{3.18}$$

While the latter is the normalization condition $||\psi|| = 1$, due to the said properties, as in (3.7), it is possible to obtain the time-frequency scaling and shift relations

$$\psi_{t_u s}(t) = \frac{1}{\sqrt{s}} \psi \left(\frac{t - t_u}{s} \right) \tag{3.19}$$

Moreover for each resulting kernel the normalization condition must be preserved so that

$$||\psi_{t_u s}|| = \int_{-\infty}^{+\infty} |\psi_{t_u s}(t)|^2 dt = 1 \qquad \forall\ t_u, s \tag{3.20}$$

If the enlisted conditions are verified, given a signal $f \in L^2\mathbb{R}$, a shift factor t_u and a scaling factor s, it is possible to define the wavelet transform centered on t_u at a scale s as the functional application $W_{st_u} : L^2\mathbb{R} \to L^2\mathbb{R}$ so that

$$W_{st_u}[f] = \int_{-\infty}^{+\infty} f(t) \psi^*_{t_u s}(t) dt = \int_{-\infty}^{+\infty} f(t) \frac{1}{\sqrt{s}} \psi^* \left(\frac{t - t_u}{s} \right) dt \tag{3.21}$$

From (3.21) follows the wavelet convolution rule

$$\begin{cases} W_{st_u}[f] &= \langle f | \psi_{t_u s} \rangle \ = \ f \star \tilde{\psi}_s(t_u) \\ \tilde{\psi}_s(t_u) &= \frac{1}{\sqrt{s}} \psi^* \left(\frac{-t}{s} \right) \end{cases} \tag{3.22}$$

Changing to the frequencies domain it is possible to transform $\tilde{\psi}_s(t)$ obtaining its Fourier transform

$$\hat{F}[\tilde{\psi}_s(t)] = \tilde{\psi}_s(\omega) = \sqrt{s} \psi^*(s\omega) \tag{3.23}$$

Since due to (3.17) it is

$$\psi(\omega)\big|_{\omega=0} = \int_{-\infty}^{+\infty} f(t) \psi(t) dt = 0 \tag{3.24}$$

starting from (3.23) it is possible to demonstrate that $\psi(\omega)$ act trivially as a band-pass filer on the signal f. Therefore computing the convolution $f \star \tilde{\psi}_s(t_u)$ of (3.22) we obtain a wavelet scale transform as a band-pass filter on f properly scaled and shifted by means of the factors s and t_u.

3.3 Wavelet filters and projectors

Since the wavelet transform, once selected a scale factor s and a shift factor t_u , acts as band-pass filters it is natural to associate the overall wavelet transform to a filter banks varying the parameters s and t_u. Let Ψ a set of dyadic wavelet kernels

$$\Psi = \left\{ \psi_{jn} \middle/ \psi_{jn}(t) = \frac{1}{\sqrt{2^j}} \psi \left(\frac{t - 2^j n}{2^j} \right) \right\}_{j,n \in \mathbb{Z}} \tag{3.25}$$

it is possible to show that Ψ is an orthogonal basis of $L^2(\mathbb{R})$, and then that

$$f(t) = \sum_{j,n} \langle f(t) | \psi_{jn} \rangle \psi_{jn}(t) = \sum_{j,n} d_{jn}(t) \psi_{jn}(t) \qquad \forall \ f \in L^2(\mathbb{R}) \tag{3.26}$$

where $d_{jn} = \langle f(t) | \psi_{jn} \rangle$ are called wavelet coefficients. The (3.26) is the practical link among the wavelet functions and the classical functional analysis based on the Fourier transform. The wavelet family Ψ is called biorthogonal wavelet family and is composed by kernels ψ_{jn} that are computed starting by the same motherwavelet $\psi = \psi_{00}$ by using a scaling factor 2^j and a shift factor $2^j n$. In this manner each one of the wavelet kernels ψ_{jn} identifies a portion of the information carried by the signal on the band with resolution 2^j. Using a truncated approximation of (3.26) it is then possible to obtain a multi-resolution approximation of the signal since the wavelet kernels act in this case as ideal conjugate mirror filters, and in this case the overall decomposition resembles a multi-rate filter bank. Taking advantage of such similarities, the wavelet transform can be easily implemented as filter banks and therefore lowering the computational complexity of the operation. For a discrete signal composed by N samples then the computational complexity is in the order of

$\mathcal{O}(N)$ operations. For discrete signals then is possible to construct orthogonal wavelet basis with compact support by means of unidimensional filters. In this manner the construction of a wavelet basis is reduced to the resolution of a multi resolution approximation problem for a signal. Let $f \in L^2(\mathbb{R})$ be a signal, then the expansion (3.26) can be interpreted as the difference of an approximations of the signal with resolution 2^{-j+1} with respect to another approximations of the signal with resolution 2^{-j}. The multi resolution decomposition of a signal represents the signal itself by means of its projection of a set of orthogonal spaces $\{\mathcal{V}_j\}_{j\in\mathbb{Z}}$. Such projection are performed by means of the devised filters which acts as a orthogonal projectors on for the orthogonal spaces. The multi resolution analysis of a signal allows us to highlight some characteristics of the signal itself by means of the time-energy signature even when such characteristics are hidden because of interferences, noise or aberrations of the signal, since those interferences generally presents different signatures. The approximation of a signal f at a resolution 2^{-j} is performed by means of a grid o samples and the related local means on a dimension proportional to 2^j. Formally it is possible to define the approximation of a function $f \in L^2(\mathbb{R})$ at a resolution of 2^{-j} as the orthogonal projection of f on the space $\mathcal{V}_j \subset L^2(\mathbb{R})$. Moreover, in the space $\mathcal{V}_j$ are contained all the possible approximation of f at resolution 2^{-j}. It follows that

$$\forall\ f \in L^2(\mathbb{R})\ \ \exists\ \left\{ f_{\nu j} = P^{\nu}_{\mathcal{V}_j}[f] \Big/ f_{\nu j} \in \mathcal{V}_j \subset L^2(\mathbb{R}) \right\}_{\nu} \qquad \forall\ j \in \mathbb{Z} \tag{3.27}$$

therefore by fixing a general approximation basis for $L^2(\mathbb{R})$, the orthogonal projection of f in $\mathcal{V}_j$ results to be the function $f_j \in L^2(\mathbb{R})$ so that $||f - f_j||$ is minimal. Then it follows the definition of Mallat [127] and Meyer [134] of multiresolution. A set $\{\mathcal{V}_j\}_{j\in\mathbb{Z}}$ of closed subspaces of $L^2(\mathbb{R})$ is called multi resolution if the following conditions are

verified

$$\begin{aligned}
&\mathcal{V}_{j+1} \subset \mathcal{V}_j && \forall\ j \in \mathbb{Z} \\
&f(t) \in \mathcal{V}_j \Leftrightarrow f(t/2) \in \mathcal{V}_{j+1} && \forall\ j \in \mathbb{Z} \\
&f(t) \in \mathcal{V}_j \Leftrightarrow f(t - 2^j k) \in \mathcal{V}_j && \forall\ j,k \in \mathbb{Z} \\
&\lim_{j \to +\infty} \mathcal{V}_j = \bigcap_{j \in \mathbb{Z}} \mathcal{V}_j = \{\underline{\mathbf{0}}\} \\
&\lim_{j \to -\infty} \mathcal{V}_j = \bigcup_{j \in \mathbb{Z}} \mathcal{V}_j = L^2(\mathbb{R})
\end{aligned} \tag{3.28}$$

then it is possible to state that it exists a set of motherwavelets $\{\Phi(t-n)\}_{n\in\mathbb{Z}}$ which is a Rietz basis for $\mathcal{V}_0$. Intuitively it follows that $\mathcal{V}_j$ is invariant for any translation proportional to the scale factor 2^j. As a result of such properties, a sequence $\{h_k\}$ exists such that the scaling function satisfies a refinement equation

$$\varphi(x) = 2\sum_k h_k \varphi(2x - l) \tag{3.29}$$

The set of functions $\{\varphi_{j,l}(x) | l \in \mathbb{Z}\}$ with

$$\varphi_{j,l}(x) = \sqrt{2^j}\varphi(2^j\, x - l) \tag{3.30}$$

is a Riesz basis of V_j. Define now W_j as a complementary space of V_j in V_{j+1} , such that $V_{j+1} = V_j \oplus W_j$, $v(2x) \in W_{j+1}$, and $v(x) \in W_0 \Leftrightarrow v(x+1) \in W_0$. Consequently

$$\bigoplus_{j=-\infty}^{+\infty} W_j = L^2(\mathbb{R}) \tag{3.31}$$

A function $\phi(x)$ is a wavelet if the set of functions $\{\varphi(x-l) | l \in \mathbb{Z}\}$ is a Riesz basis of W_0 and also meets the following two conditions:

$$\int_{-\infty}^{+\infty} \psi(x)dx = 0 \tag{3.32}$$

and

$$\|\psi(x)\|^2 = \int_{-\infty}^{+\infty} \psi(x)\psi^*(x)dx = 1 \tag{3.33}$$

If the wavelet is also an element of V_0, a sequence $\{g_k\}$ exists such that

$$\psi(x) = 2\sum_k g_k\, \varphi(2x - l) \tag{3.34}$$

The set of functions $\{\varphi_{j,l}(x)|l \in \mathbb{Z}\}$ is now a Riesz basis of $L^2(\mathbb{R})$. The coefficients in the expansion of a function in the wavelet basis are given by the inner product with dual wavelet $\tilde{\psi}_{j,l}(x) = \sqrt{2^j}\tilde{\psi}(2^j x - l)$ such that

$$f(x) = \sum_{j,l} \langle f, \widetilde{\psi}_{j,l}\rangle\, \psi_{j,l}(x) \tag{3.35}$$

Likewise, a projection on V_j is given by

$$P_j f(x) = \sum_l \langle f, \widetilde{\varphi}_{j,l}\rangle\, \varphi_{j,l}(x) \tag{3.36}$$

where $\tilde{\varphi}_{j,l}(x) = \sqrt{2^j}\tilde{\psi}(2^j x - l)$ are the dual scaling functions. The dual functions have to satisfy the biorthogonality conditions

$$\langle \varphi_{j,l}, \widetilde{\varphi}_{j,l'}\rangle = \delta_{l-l'} \tag{3.37}$$

and

$$\langle \psi_{j,l}, \widetilde{\psi}_{j',l'}\rangle = \delta_{j-j'}\delta_{l-l'} \tag{3.38}$$

They satisfy refinement relations similar to (3.29) and (3.34) involving sequences $\{\tilde{h}_k\}$ and $\{\tilde{g}_k\}$. In case the basis functions coincide with their duals, the basis is orthogonal. As shown by Cybenko [56], given a continuous function $f : \mathbb{R} \to R$ it exist a discriminating function σ so that is possible to obtain a representation

$$f(x) = \sum_{i=1}^{\infty} w_i \sigma(a_i^{\intercal} x + b_i) \tag{3.39}$$

where $w_i, b_i \in \mathbb{R}$ and $a_i \in \mathbb{R}^n$ is a n-dimensional real vector. If σ is dense in $[0,1]^n$ it follows that any continuous function f can be approximated by a finite sum. Analogously according to wavelet theory we can state that

$$f(x) = \sum_{i=1}^{\infty} w_i \det(D_i^{1/2}) \psi(d_i x - t_i) \tag{3.40}$$

is dense in the $L^2(\mathbb{R}^n)$. We will call $\{d_i\}$dilatation vector and t_i translation vector, while will be $D_i = \text{diag}(d_i)$ and ψ will be called mother wavelet function whose translates and dilates forms the basis for the $L^2(\mathbb{R}^n)$ space [176]. In this scenario, as a matter of facts, the space $\mathcal{V}_j$ can be interpreted as a topological grid of cells of dimension 2^j characterizing a resolution of 2^{-j}. From the (3.28) also follows that given an approximation with a resolution 2^{-j} it contains all the information in order to compute a further approximation with a resolution 2^{-j-1}. By iterating the process the resolution of course will be reduced of a factor of 2 for each iteration, therefore, approaching 0, all the information details will be lost since

$$\lim_{j \to -\infty} ||f_j|| = 0 \tag{3.41}$$

On the other hand, if we immagine on the opposite to make the resolution diverge to $+\infty$ it obviously follows that the approximation should converge to the original signal itself and that

$$\lim_{j \to +\infty} ||f_j - f|| = 0 \tag{3.42}$$

where the norm $||f_j - f||$ is indeed the approximation error.

3.4 Conjugate wavelet mirror filters

As explained in the previous section, it is possible to generate an orthogonal basis for each $\mathcal{V}_j$ starting from a scaling function Φ. Each one of the generated function can also be described by means of conjugate filters. From (3.28) follows that, given a

function Φ generating an orthonormal basis for $\mathcal{V}_0$ it is possible to obtain it using

$$\frac{1}{\sqrt{2}}\Phi\left(\frac{t}{2}\right) \in \mathcal{V}_1 \subset \mathcal{V}_0 \tag{3.43}$$

and since $\{\Phi(t-n)\}_{n\in\mathbb{Z}}$ is an orthonormal basis for $\mathcal{V}_0$ it is possible to decompose it as the expansion

$$\frac{1}{\sqrt{2}}\Phi\left(\frac{t}{2}\right) = \sum_{n\in\mathbb{Z}} h[n]\Phi(t-n) \tag{3.44}$$

where h[n] represents the discrete conjugate filter

$$h[n] = \left\langle \frac{1}{\sqrt{2}}\Phi\left(\frac{t}{2}\right) \middle| \Phi(t-n) \right\rangle \tag{3.45}$$

Applying the Fourier transform to the members of the (3.44) it follows that

$$\begin{cases} \Phi(2\omega) &= \frac{1}{\sqrt{2}}h[\omega]\Phi(\omega) \\ \\ h[\omega] &= \sum\limits_{n\in\mathbb{Z}} h[n]e^{in\omega} \end{cases} \tag{3.46}$$

At this point it is useful to represent $\Phi[\omega]$ as direct product of $h[\omega]$ so that

$$\Phi(2^{1-p}\omega) = \frac{1}{\sqrt{2}}h[2^{-p}\omega]\Phi(2^{-p}\omega) \quad \forall\ p \geqslant 0 \tag{3.47}$$

and then by substitution and truncation to the $P-th$ factor it is possible to obtain

$$\Phi(\omega) \approx \left(\prod_{p=1}^{P} \frac{h[2^{-p}\omega]}{\sqrt{2}}\right)\Phi(2^{-P}\omega) \tag{3.48}$$

If $\Phi(\omega)$ is continuous in 0 then $\lim\limits_{P\to+\infty}\Phi(2^{-P}\omega) = \Phi(0)$, and therefore

$$\Phi(\omega) = \left(\prod_{p=1}^{+\infty} \frac{h[2^{-p}\omega]}{\sqrt{2}}\right)\Phi(0) \tag{3.49}$$

Since then it is possible to implement the wavelet scaling function by means of conjugate mirror filters both in the time and frequency domain, then it is logical to use such filters in order to implement a fast computation method for wavelet decomposition. A solid theoretical support on the topic was developed by Cohen [46, 47].

3.5 Orthogonal wavelet basis

It is finally possible to introduce the orthogonal wavelet basis. Since the approximation of the signal $f \in \mathrm{L}^2(\mathbb{R})$ at a scale 2^j corresponds to its orthogonal projection on $\mathcal{V}_j \subset \mathcal{V}_{j-1} \subset L^2(\mathbb{R})$, if $\mathcal{W}_j$ is the orthogonal complement of $\mathcal{V}_j$ in $\mathcal{V}_{j-1}$, then

$$\mathcal{V}_{j-1} = \mathcal{V}_j \oplus \mathcal{W}_j \tag{3.50}$$

where $\mathcal{V}_j \oplus \mathcal{W}_j$ represents the direct sum among the two spaces $\mathcal{V}_j$ and $\mathcal{W}_j$. The orthogonal projection of f in $\mathcal{V}_j$ can be expanded as the sum

$$P_{\mathcal{V}_{j-1}}[f] = P_{\mathcal{V}_j}[f] + P_{\mathcal{W}_j}[f] \tag{3.51}$$

where $P_{\mathcal{V}_{j-1}}$,$P_{\mathcal{V}_j}$ and $P_{\mathcal{W}_j}$ are the orthogonal projector operators respectively associated to $\mathcal{V}_{j-1}$,$\mathcal{V}_j$ and $\mathcal{W}_j$. It is evident that $P_{\mathcal{W}_j}$ is able to give as output the details of the signal f at a scale 2^{i-1} that are evidently neglected at a scale 2^j. The Mallat-Meyer theorem demonstrates that it is possible to construct an orthonormal basis of $\mathcal{W}_j$ by scaling and translation of a wavelet basis Ψ. From this theorem follows a lemma to demonstrate that the wavelet family $\Phi = \{\psi_{jn}\}_{j,n\in\mathbb{Z}}$ is an orthonormal basis of $\mathcal{W}_j$ if and only if

$$\begin{cases} \frac{1}{2}|g[\omega]|^2 + \frac{1}{2}|g[\omega+\pi]|^2 = 1 \\ \\ g[\omega]h^*[\omega] + g[\omega+\pi]h^*[\omega+\pi] = 0 \end{cases} \tag{3.52}$$

where $g[\omega]$ and $h[\omega]$ are the Fourier transforms of the filters $g[n]$ and $h[n]$ defined as

$$\begin{cases} g[n] = & \left\langle \frac{1}{2}\psi\left(\frac{t}{2}\right) \middle| \psi(t-n) \right\rangle \\ h[n] = & \left\langle \frac{1}{2}\Phi\left(\frac{t}{2}\right) \middle| \Phi(t-n) \right\rangle \end{cases} \tag{3.53}$$

It finally follows the Mallat-Mayer formula

$$\psi(\omega) = \frac{1}{\sqrt{2}} g\left[\frac{\omega}{2}\right] \Phi\left(\frac{\omega}{2}\right) \tag{3.54}$$

From the inverse Fourier transform of (3.53) and from (3.54), it is finally possible to express the wavelet decomposition equations in filter form as

$$\begin{array}{rcl} g[n] & = & (-1)^{1-n} h[1-n] \\ g[\omega] & = & e^{-i\omega} h^*[\omega + \pi] \end{array} \tag{3.55}$$

The (3.55) represent the main equations to implement the commonly used algorithms of fast wavelet transform. All the wavelet kernels Φ and ψ families associate to the coniugate filter banks $h[n]$ and $g[n]$ can be composed to form many orthonormal basis for $\mathcal{V}_j$ and $\mathcal{W}_j$. Now on we will call these latter spaces, respectively, space of the residuals and space of the coefficients (or details) of the signal at a wavelet scale of 2^j. The Fast Wavelet Transform (FWT) algorithms are computed as cascade convolutions where $h[n]$ and $g[n]$ constitutes subsampling filters for the signal.

3.6 Fast Orthogonal Wavelet Transform

The FWT algorithm is a procedural and iterative decomposition of the signal which makes use of the discrete conjugate wavelet filters of (3.55). The algorithm decompose iteratively the signal obtaining, at the j-th step, an approximation $P_{\mathcal{V}_j}[f]$ and the related coefficients $P_{\mathcal{W}_j}[f]$, then it iterates by obtaining from $P_{\mathcal{V}_j}[f]$ a coarser approximation $P_{\mathcal{V}_{j+1}}[f]$ and the related coefficients $P_{\mathcal{W}_{j+1}}[f]$. Viceversa the inverse transform reconstructs an approximation $P_{\mathcal{V}_j}[f]$ from a coarse approximation $P_{\mathcal{V}_{j+1}}[f]$

and the related coefficients $P_{\mathcal{W}_{j+1}}[f]$. Given an orthonormal basis $\{\phi_{jn}\}_{j,n\in\mathbb{Z}}$ of $\mathcal{V}_j$ and an orthonormal basis $\{\psi_{jn}\}_{j,n\in\mathbb{Z}}$ of $\mathcal{W}_j$, it is so possible to define the related projections

$$\begin{aligned} \underline{a}_j &= a_j[n] &= \langle f|\phi_{jn}\rangle \in \mathcal{V}_j \\ \underline{d}_j &= d_j[n] &= \langle f|\psi_{jn}\rangle \in \mathcal{W}_j \end{aligned} \tag{3.56}$$

By means of the decomposition theorem of Mallat, from the properties of the wavelet filters, it follows that

$$\begin{aligned} \underline{a}_{j+1} &= \underline{a}_j \star h^*[2n] \\ \underline{d}_{j+1} &= \underline{a}_j \star g^*[2n] \end{aligned} \tag{3.57}$$

and viceversa, for the inverse transform, it follows that

$$\underline{a}_j = \underline{a}_{j+1} \star h[n] + \underline{d}_{j+1} \star g[n] \tag{3.58}$$

Finally, considering $\underline{a}_0 = f$, all the properties of the wavelet transform are verified both for the signal analysis and synthesis. Using (3.57) and (3.58) it is then possible to construct the analysis and synthesis schema by recursion. Starting from $\underline{a}_0 = f$, each successive iteration will compute the wavelet coefficients $\underline{d}_{j+1}$ and the residuals of $\underline{a}_{j+1}$ at an approximation scale of 2^{j+1} from the $\underline{a}_j$ residuals. At each iteration, then, the resulting $\underline{a}_{j+1}$ will constitute a subsampling of the signal with a factor 2 with respect to $\underline{a}_j$. Viceversa, for the synthesis procedure, starting from a signal $\underline{a}_{j+1}$ and the related details $\underline{d}_{j+1}$ at each iteration a finer approximation $\underline{a}_j$ can be obtained incrementing the signal resolution of a factor 2. Considering the FWT schema, it is possible to obtain an orthogonal representation of $\underline{a}_L = \langle f|\psi_{Ln}\rangle$, this latter will be composed by a collection of wavelet coefficients obtained by the projection $P_{\mathcal{W}_j} f$ at a scale 2^j with $2^L < 2^j \leqslant 2^{L+J}$, where J represents the maximum approximation level and therefore 2^{L+J} the maximum approximation scale. It is finally possible to formalize an operative definition of the complete wavelet transform to the scale 2^{L+J}, or simply a J-levels wavelet transform of $\underline{a}_L$ as

$$\hat{W}_J[\underline{a}_L] = (\underline{d}_{L+1}|\underline{d}_{L+2}|\cdots|\underline{d}_{J-1}|\underline{d}_J|\underline{a}_J) \tag{3.59}$$

and similarly the inverse J-levels wavelet transform of $\hat{W}_J[\underline{a}_L]$ as

$$\tilde{W}_J[(\underline{d}_{L+1}|\underline{d}_{L+2}|\cdots|\underline{d}_{J-1}|\underline{d}_J|\underline{a}_J)] = \underline{a}_L \tag{3.60}$$

3.7 Biorthogonal wavelet basis

The biorthogonal wavelets use a particular transform, which is invertible but not necessarily orthogonal, based on coupled filters. Biorthogonal wavelets allows more degrees of freedom respect to traditional orthogonal (and of course the not orthogonal) wavelets, it because these have two different scaling functions which can generate twin resolution analysis with two different wave functions ψ and ψ'. The freedom hypothesis lead us to conclude that the size N and M of the coefficients sets $\{a\}$ and $\{a'\}$ can differ. The scaling sequence must satisfy the biorthogonality condition

$$\sum_{n=1}^{N} a_n a'_{n-2m} = 2\delta_{m0} \quad \forall\ m \in [1, M] \cap \mathbb{N} \tag{3.61}$$

The wavelet coefficients set in his general form are generated by

$$\begin{cases} b_n &= (-1)^n a'_{N-1-n} \\ \\ b'_n &= (-1)^n a'_{M-1-n} \end{cases} \tag{3.62}$$

The main idea of the proposed algorithm is that the underlying filtering operations, rather than being one sided, are two sided: filters are not required to be symmetric, but they must be of length $2k + 1$ ($k \in N$) and the middle sample being taken as the filter coefficient attached to zero lag [37].

3.8 Biorthogonal Wavelets filters

The definitions and notions given until now are mandatory in order to understand the biorthognoal wavelets. As presented before, to obtain the wavelet decomposition

or synthesis of a signal it is necessary to use conjugate mirror filter banks, on the other hand such filters responds as finite functional pulses. As shown by the Vetterli theorem [196], two wavelet filters $\tilde{h}$ and $\tilde{g}$ perfectly decompose or reconstruct a signal if and only if

$$\begin{cases} \hat{F}\Big[h^*[\omega+\pi]\Big]\hat{F}\Big[\tilde{h}[\omega]\Big] + \hat{F}\Big[g^*[\omega+\pi]\Big]\hat{F}\Big[\tilde{g}[\omega]\Big] &= 0 \\ \\ \frac{1}{2}\hat{F}\Big[h^*[\omega]\Big]\hat{F}\Big[\tilde{h}[\omega]\Big] + \frac{1}{2}\hat{F}\Big[g^*[\omega]\Big]\hat{F}\Big[\tilde{g}[\omega]\Big] - 1 &= 0 \end{cases} \tag{3.63}$$

By this theorem it is also possible to demonstrate that for any class of filters h and g it exists a special pairs of filters $(\tilde{h}, \tilde{g})$ named exact or perfect filters. Such filters can perfectly decompose and reconstruct the signal and are completely computable starting by a general pair of filters $(\tilde{h}, \tilde{g})$ by solving the composite filters problem (3.63) here also proposed in matricidal form:

$$\begin{pmatrix} \hat{F}\Big[h[\omega]\Big] & \hat{F}\Big[g[\omega]\Big] \\ \hat{F}\Big[h[\omega+\pi]\Big] & \hat{F}\Big[g[\omega+\pi]\Big] \end{pmatrix} \times \begin{pmatrix} \hat{F}\Big[\tilde{h}^*[\omega]\Big] \\ \hat{F}\Big[\tilde{g}^*[\omega+\pi]\Big] \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \tag{3.64}$$

and by a 2×2 inversion as

$$\begin{pmatrix} \hat{F}\Big[\tilde{h}^*[\omega]\Big] \\ \hat{F}\Big[\tilde{g}^*[\omega]\Big] \end{pmatrix} = \frac{2}{\Delta[\omega]} \begin{pmatrix} \hat{F}\Big[g[\omega+\pi]\Big] \\ -\hat{F}\Big[h[\omega+\pi]\Big] \end{pmatrix} \tag{3.65}$$

where $\Delta[\omega]$ is the determinant

$$\Delta[\omega] = \hat{F}\Big[h[\omega]\Big]\hat{F}\Big[g[\omega+\pi]\Big] - \hat{F}\Big[h[\omega+\pi]\Big]\hat{F}\Big[g[\omega]\Big] \tag{3.66}$$

By the determinant in (3.66) depends the stability of the filters bank [193], in facts the stability condition is verified only if

$$\exists\ \omega_* \in [-\pi, \pi] : \Delta[\omega_*] \neq 0 \tag{3.67}$$

as proven by Vaidyanathan. Finally it is possible to extend the theorem to the finite response filters [194]. On the other hand, for the finite case, two parameters must be introduced: an amplification coefficient $a \in \mathbb{R}\backslash\{0\}$ and a shift $l \in \mathbb{Z}$ so that

$$\begin{cases} \hat{F}\Big[g[\omega]\Big] &= ae^{-i(2l+1)\omega}\hat{F}\Big[\tilde{h}^*[\omega+\pi]\Big] \\ \\ \hat{F}\Big[\tilde{g}[\omega]\Big] &= \frac{1}{a}e^{-i(2l+1)\omega}\hat{F}\Big[h^*[\omega+\pi]\Big] \end{cases} \tag{3.68}$$

and by the normalization condition $a = 1$ and with null shift for $l = 0$ it is possible to obtain the system on the time domain

$$\begin{cases} g[n] &= (-1)^{1-n}\tilde{h}[1-n] \\ \tilde{g}[n] &= (-1)^{1-n}h[1-n] \end{cases} \tag{3.69}$$

ad consequently the following properties for the perfect biorthogonal conjugate mirror filters

$$(g, \tilde{h}) = (\tilde{g}, h) \tag{3.70}$$

It is then obvious that the couples of stable filters (g, h) and stable filters $(\tilde{g}, \tilde{h})$ have a symmetrical role and can be substituted each other. This inversion in (3.70) is very useful in order to interprete the discrete wavelet decomposition operated by the conjugate filter banks as a functional expansion in a base of $\ell^2(\mathbb{Z})$. Therefore the residuals $\underline{a}_j$ and the details $\underline{d}_j$ can be expressed as inner products expansions in $\ell^2(\mathbb{Z})$, and therefore it follows that

$$\begin{aligned} \underline{a}_{j+1} &= \sum_{n\in\mathbb{Z}} a_j[n]h[n-2l] = \langle a_j[n]|h[n-2l]\rangle \\ \underline{d}_{j+1} &= \sum_{n\in\mathbb{Z}} d_j[n]g[n-2l] = \langle d_j[n]|g[n-2l]\rangle \end{aligned} \tag{3.71}$$

Similarly it is possible to expand a synthesized signal as

$$\underline{a}_j = \sum_{l\in\mathbb{Z}} a_{j+1}[l]\tilde{h}[n-2l] + \sum_{l\in\mathbb{Z}} d_{j+1}[l]\tilde{g}[n-2l] = \langle a_j[n]|h[n-2l]\rangle \tag{3.72}$$

From (3.71) and (3.72) it follows

$$\underline{a}_j = \sum_{l\in\mathbb{Z}} \langle f|h[n-2l]\rangle \tilde{h}[n-2l] + \langle f|g[n-2l]\rangle \tilde{g}[n-2l] \tag{3.73}$$

The (3.73) is immediately recognizable as the decomposition of a signal $\underline{a}_j \in L^2(\mathbb{R})$ by two dual sets of filters $\{\tilde{h}[n-2l], \tilde{g}[n-2l]\}_{l\in\mathbb{Z}}$ and $\{h[n-2l], g[n-2l]\}_{l\in\mathbb{Z}}$. These latter sets, by theorem, and basing on the definition given at the beginning of this chapter, constitutes two families of orthogonal Riets basis of the set $\ell^2(\mathbb{R})$, moreover, these sets are also biorthogonal since given

$$\langle \tilde{h}[n]|h[n-2l]\rangle = \langle \tilde{g}|g[n-2l]\rangle = \delta[l] \tag{3.74}$$

it immediately follows that

$$\langle \tilde{h}[n]|g[n-2l]\rangle = \langle \tilde{g}|h[n-2l]\rangle = 0 \tag{3.75}$$

An infinite cascade of filter pairs (h, g) and $(\tilde{h}, \tilde{g})$ constitutes the scaling function and the motherwavelet function so that their Fourier transforms verifies the conditions

$$\begin{cases} \hat{F}\Big[\phi(2\omega)\Big] &= \frac{1}{\sqrt{2}}\hat{F}\Big[h(\omega)\Big]\hat{F}\Big[\phi(\omega)\Big] \\ \hat{F}\Big[\psi(2\omega)\Big] &= \frac{1}{\sqrt{2}}\hat{F}\Big[g(\omega)\Big]\hat{F}\Big[\psi(\omega)\Big] \\ & \\ \hat{F}\Big[\tilde{\phi}(2\omega)\Big] &= \frac{1}{\sqrt{2}}\hat{F}\Big[\tilde{h}(\omega)\Big]\hat{F}\Big[\tilde{\phi}(\omega)\Big] \\ \hat{F}\Big[\tilde{\psi}(2\omega)\Big] &= \frac{1}{\sqrt{2}}\hat{F}\Big[\tilde{g}(\omega)\Big]\hat{F}\Big[\tilde{\psi}(\omega)\Big] \end{cases} \tag{3.76}$$

while in the time domain the following conditions are verified

$$\begin{cases} \phi(t) &= \sqrt{2}\sum\limits_{n\in\mathbb{Z}} h[n]\phi(2t-n) \\ \psi(t) &= \sqrt{2}\sum\limits_{n\in\mathbb{Z}} g[n]\psi(2t-n) \\ \tilde{\phi}(t) &= \sqrt{2}\sum\limits_{n\in\mathbb{Z}} \tilde{h}[n]\tilde{\phi}(2t-n) \\ \tilde{\psi}(t) &= \sqrt{2}\sum\limits_{n\in\mathbb{Z}} \tilde{g}[n]\tilde{\psi}(2t-n) \end{cases} \tag{3.77}$$

When using wavelet filters organized with the presented structure, due to the complete analogy among the filter themselves with respect to the biorthogonal basis of $\ell^2(\mathbb{R})$, it is finally possible to implement the biorthogonal FWT by using pairs of exact filters, therefore it is possible to apply such a transform to sampled signals. Let $x[n]$ be a digital signal sampled with discrete time intervals of $N^{-1} = 2^L$, then exists $f \in \mathcal{V}_L$ so that

$$a_L[n] = \langle f|\phi_{Ln}\rangle = \frac{x[n]}{\sqrt{N}} \tag{3.78}$$

If as before we call $a_j[n] = \langle f|\phi_{jn}\rangle$ and $d_j[n] = \langle f|\psi_{jn}\rangle$ it is possible to compute iteratively the approximations and details of a biorthogonal wavelet decomposition as

$$\begin{aligned} \underline{a}_{j+1} &= \underline{a}_j \star h[2n] \\ \underline{d}_{j+1} &= \underline{a}_j \star g[2n] \end{aligned} \tag{3.79}$$

As well as it will be possible to synthesize the signal by the inverse biorthogonal wavelet transform by iterating

$$\underline{a}_j = \underline{a}_{j+1} \star \tilde{h}[n] + \underline{d}_{j+1} \star \tilde{g}[n] \tag{3.80}$$

Basically if $x[N]$ is constituted by N samples, then, by means of (3.79) and (3.80) it is possible to decompose the signal to the successive level or to synthetise the signal to the previous level with $\mathscr{O}(N)$ operations. One example of wavelet performances can be noticed for nonlinear dynamical systems [177]. Another good example, often used as a benchmark for nonlinear predictors can be the Mackey Glass chaotic time series [58]. It has been repeatedly shown in literature that a very compact representation of the input data are ideal to be used with Neural Networks technology. In Chapter 5 we will show how to bound toughener Wavelet Analysis and Neural Networks, but before that we will introduce, in the next chapter, the mathematical basis behind the Neural Network we are going to use for this work.

CHAPTER 4

AI: Artificial Intelligence

Intelligence is the ability to adapt to change.

Stephen Hawking

Neural networks constitutes an exceptional set of powerful tools to solve classification problems as well as linear regression problems, non linear control, prediction, forecast etc. One of the main advantage of such an approach is the heuristically approach to problem solving which is generally implemented by emulation of known solutions. Another advantage is on the elaboration speed of such network once trained, as well as the possibility to extract phenomenological models from that a without any a priori assumption. This characteristics give us an alternative to analytical studies that sometimes are not possible due to the lack of experimental evidences appreciable at a first sight, or due to their unfeasibility when it comes to non deterministic behaviors or chaotic dynamics. The inspiration for the development of artificial neural network architectures came from the studies on the biological neural system. Starting from the features manifested in the human brain and nervous apparatus, until today many different kind of artificial neural networks has been realized in order to solve a wide range of problems. Here the basis of such architectures are devised.

4.1 Biological and Artificial Neural Networks

In the human physiology the neurons are biological cells electrically active which, only the human brain, reach a magnitude of 10^{11}. The neural dendrites can be considered the electrical input of the neural cell as well as the axion can be considerate its electrical output. The neuron can communicate by means of the junctions among axons and dendrites, such junctions are called synapsis and allows the neurons to tap into network of thousands of reciprocally connected units. The human neurons have several resemblances with respect to the logic gates of a calculator, in facts, just like logic gates, the neurons can be found in two states respectively of activity or rest. When the neuron is activated it causes a electrical potential difference, such a potential make it possible to transport electrical receptors along the axion to the synapsis which, consequently, release chemical substances called neurotransmitters. Through the synapsis the neurons can modify their electrical potential as well as modify the electrical state of charge of neighbor neurons, and moreover, exchange neurotransmitters. Such neurotransmitters are excitants or inhibitors of the neural activity, therefore the exchange of these molecules can increase or decrease the activity of the neighboring neurons, sometime with cascade effects. The effect produced by a certain concentration of neurotransmitters is somehow similar to an algebraic sum of positive and negative weights to determine the aptitude and effect of a certain activation function. While this description represents of course a quite mechanical ad inaccurate view of the mechanisms at the basis of the human brain, it is a quite self explanatory representation of the basis functionalities of a network of biological neurons. In such a view it is then possible to artificially emulate the devised processes by means of electrical units or software systems. If we imagine the activation of a neuron as a function, and if we imagine the output of this function to be null when the inputs values are less or equal than a certain threshold, it is then quite easy to design an electrical circuit or software equivalent in order to implement the described process. A collection of so manufactured units can be then called artificial neural network. From this point, if not differently specified, the term neural network will be referred to such artificial neural networks.

4.2 Activation functions

The artificial neuron is are the information processing units used for the construction of neural networks. It is possible to agree on the fact that the principal elements of a neuron (both artificial neuron or biological neuron) are constituting by bidirectional information buses for the information transfer, independently by the artificial or biological nature of this kind of connection. Therefore from an informational point of view the human brain and an artificial neural network shares the same functioning processes. Each synapses, or link, represents such kind of information bus and is characterised by specific properties such as the throughput, the amplification factor, the speed or the responsiveness to a stimulus, as well as the influence on a linked neuron. The neurons, in facts, are constantly exposed to electrical or biochemical signals (or their digital counterpart), and the resulting effect is given by the superposition of all those signals, opportunely mediated by the synapses. At the end, the synaptic mediation is the responsible mechanism for the functionalities of the human brain and the nervous system. In the human nervous system, while a synapse transmits an out coming signal of a certain strength, such a strength is amplified or damped proportionally to the synaptic section and length, moreover such signals can be amplified, damped or delayed by means of special neurotransmitters. In a quite oversimplified analysis of the mechanism it is then possible to imagine a neuron as a concentrated functional unit. The input of this functional units is given by an algebraic sum of different contributions corresponding to the different input synapses, on the other hand different synapses in different moments are capable of enhancing or reducing the signal strength, therefore such an algebraic sum should also take into consideration a multiplicative weight representing the synaptic effect on each contribution. Finally we can imagine a neuron as a function $f : \mathbb{R} \to \mathbb{R}$, and its response as a out coming signal

$$y = f\left(\sum_i w_i x_i\right) \tag{4.1}$$

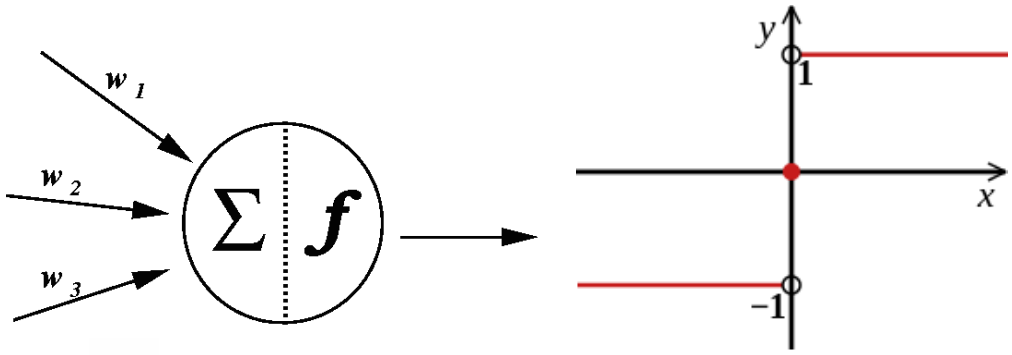

Figure 4.1: On the left: a typical neuron unit. On the right: the signum function of (4.2).

where w_i and x_i represent respectively the synaptic weight and the related original signal to be transmitted.The result of the neural function will determine the activation of the neuron and the consequent signal transmission, therefore the function f will be called activation (or transmission) function. Unfortunately it is not possible to concentrate in an unique and simple description all the possibilities concerning activation functions since it exist a very large set of possible functions granting different performance when used in different contests. Moreover sometime several activation functions cannot be used in certain situations due to problems raising from the intrinsic structure of the function itself. As a general condition every function which permits to finitely integrate its absolute value within a finite interval can be used as an activation function. In set theory then an activation function is a function $f \in L^1(\mathbb{R})$, on the other hand this condition, while mandatory, is not sufficient to grant the optimality of the selected function. A suitable activation function must be able to correctly discriminate whether activate the neuron or not, and the output values to transmit to the successive neuron. In order to correctly set few ideas on the activation functions a very simple example could be helpful. Suppose to select as activation function the signum function $\text{sgn} : \mathbb{R} \to \{0, 1\}$, then suppose to use this function to activate a neuron which receives a tuple of input data (x_i) to which correspond a synaptic weights tuple (w_i). In this scenario (as in Fig. 4.1) the output

y of the neuron can be defined as

$$y = \operatorname{sgn}\left(\sum_i w_i x_i\right) = \begin{cases} +1 & \text{if } \sum_i w_i x_i \geqslant 0 \\ -1 & \text{if } \sum_i w_i x_i < 0 \end{cases} \tag{4.2}$$

It is trivial to recognize that the signum function is able to recognise and act differently when it receives two possible configurations os total inputs since it differently responds basing on the sign of the algebraic sum of the inputs. Such kind of activation function perfectly emulate some neurobiological functions based on thresholds, on the other hand many phenomena cannot be modeled in terms of thresholds and binary activation threes, therefore, often, more complex activation functions are required. Nevertheless, while mathematically selectable, not every function in $L^1(\mathbb{R})$ resembles a good activation function for many possible reasons such as too large or frequent oscillations, local minima or maxima or other strange mathematical characteristics that can have bad influence on the behavior of the function when exposed to certain classes of inputs. The purpose of a good activation function is to both discriminate and generalize so that, jointly with an appropriate and feasible selection of synaptic weights, the resulting neurons can constitute a model of the relations that links the inputs to the outputs. Such generalization and discrimination capabilities strictly depends on the selection of an adeguate activation function for the artificial neurons.

4.3 Feedforward neural networks

Basing on the given description of artificial neurons and activation functions, it is now possible to introduce a simple model of neural network. Mathematically a neuron model is constituted by a weighted functional expansion of non linear isomorphisms capable to transform an independent input vector $\underline{u}$ into a dependent output vector $\underline{y}$. More in depth, lets suppose to organize a set $\mathscr{N}$ of neurons γ_j into several disjoint ordered subsets $\mathscr{N}_i$, so that

$$\begin{aligned} \bigcup_i \mathscr{N}_i &= \mathscr{N} \\ \bigcap_i \mathscr{N}_i &= \varnothing \end{aligned} \tag{4.3}$$

and let suppose that the neurons inputs and outputs are organized as a chain so that the composite output of the neurons in a set is used also as input for the neurons in the successive set, therefore

$$\gamma_{j_i}^{(i)} \in \mathscr{N}_i \Leftrightarrow \gamma_{j_i}^{(i)} : \bigoplus_{j_{i-1}=1}^{|\mathscr{N}_{i-1}|} \mathrm{Cod}\left(\gamma_{j_{i-1}}^{(i-1)}\right) \to \bigoplus_{j_{i+1}=1}^{|\mathscr{N}_{i+1}|} \mathrm{Dom}\left(\gamma_{j_{i+1}}^{(i+1)}\right) \tag{4.4}$$

where $\mathrm{Cod}(\gamma)$ indicates the codomain of the function γ, $\mathrm{Dom}(\gamma)$ the domain of the function and the operator $\oplus$ defines the direct sum. Within the given organization, each set $\mathscr{N}_i$ is called layer and the resulting network is called neural network. Therefore it will follow a natural formalism where, if we call $\underline{x}^{(i)}$ the tuple of outputs of the i-th layer, it will also identify the tuple of inputs of the $(i+1)$-th layer. Finally, if for each layer $\mathscr{N}_l$ it exist a layer activation function γ^l and a tuple of weights $\left(w_{jk}^{(l)}\right)_{j=1}^{|\mathscr{N}_{l-1}|}$ so that for each $\gamma_{k_l}^l \in \mathscr{N}_l$ it follows

$$\gamma_k^{(l)}\left(\underline{x}^{(l-1)}\right) = \gamma^l\left(\sum_{j=1}^{|\mathscr{N}_{l-1}|} w_{jk}^{(l)} x_j^{(l-1)}\right) \tag{4.5}$$

then the resulting structure is called multilayer neural network. From this point the term neural network will refer to such a kind of networks if not differently specified. It follows that

$$\underline{x}^{(l)} = \left(\gamma_k^{(l)}(\underline{x}^{(l-1)})\right)_{k=1}^{|\mathscr{N}_l|} \tag{4.6}$$

The functional form of the isomorphism representing a multilayer neural network then strictly depends from its topological structure (now on simply called topology), by the layer activation function (now on only activation function if not differently specified) e by the tuple of weights (now on layer weights). It follows that in general a neural network is representable as a vectorial application $\nu : \mathbb{R}^N \times \mathbb{R}^P \to \mathbb{R}^M$ so that for an input vector $\underline{u} \in \mathbb{R}^N$, and given a multidimensional matrix of weights W, it associates an output vector $\underline{y} = \nu[\underline{u}, W]$. As a trivial extension of the given formalism, then, we will call $\underline{x}^{(0)} = \underline{u}$ input layer. Moreover, if the inner topology of the network is composed by $L-1$ layers, called hidden layers, then we will call $\underline{y} = \underline{x}^{(L)}$ output

layer. It is possible to expand the (4.5) considering $x_i^{(0)} = u_i \quad \forall\ i \in [1, N] \cap \mathbb{N}$. It follows that

$$x_j^{(1)} = \gamma^{(1)} \left(\sum_{i=1}^{N} w_{ij}^{(1)} x_i^{(0)} \right) \quad \forall\ j \in [1, |\mathscr{N}_1|] \cap \mathbb{N} \tag{4.7}$$

where $\gamma^{(1)}$ is the activation function of the first hidden layer and $\underline{x}^{(1)} = \left(x_j^{(1)} \right)_{j=1}^{|\mathscr{N}_1|}$ the related output. Continuing from (4.7), if the topology consists of $L > 1$ hidden layer, it follows

$$x_k^{(l)} = \gamma^{(l)} \left(\sum_{j=1}^{|\mathscr{N}_l|} w_{jk}^{(1)} x_j^{(l-1)} \right) \quad \forall\ k \in [1, |\mathscr{N}_l|] \cap \mathbb{N}, \quad \forall\ l \in [2, L] \cap \mathbb{N} \tag{4.8}$$

Finally, given $\underline{y} \in \mathbb{R}^M$ it follows

$$\gamma_m = x_m^{(L)} = \gamma^{(L-1)} \left(\sum_{k=1}^{|\mathscr{N}_{L-1}|} w_{km}^{(L)} x_k^{(L-1)} \right) \quad \forall\ m \in [1, M] \cap \mathbb{N} \tag{4.9}$$

Reading back from (4.9) to (4.7) it is evident that each element γ_m of the output vector $\underline{y}$ depends from a multiplicity of nested functional sums:

$$\gamma_m = \gamma^{(L)} \left(\sum_{k=1}^{|\mathscr{N}_{L-1}|} w_{km}^{(L)} \gamma^{(L-1)} \left(\sum_{j=1}^{|\mathscr{N}_{L-2}|} w_{jk}^{(L-1)} ... \gamma^{(1)} \left(\sum_{i=1}^{N} w_{ij}^{(1)} x_i^{(0)} \right) ... \right) \right) \tag{4.10}$$

Adopting as a convention the $\otimes$ operator to define the nested sum of (4.10) it is common use to describe a feedforward neural network with the compact expression

$$\underline{y} = \bigotimes_{l=0}^{L} \underline{w}^{l+1} \underline{x}^{(l)} \tag{4.11}$$

It is also possible to have a more rigorous formulation by using bidimensional a index μ_l on a range Ω_l given as the cartesian product of the indexing ranges of the l-th layer and the $(l-1)$-th layer, so that

$$\underline{y} = \bigotimes_{l=0}^{L} \langle \hat{\gamma}_l | w_{\mu_l}^{l} x_{\mu_l}^{l-1} \rangle \tag{4.12}$$

In literature it is often used the $\oplus$ operator instead of the $\otimes$, on the other hand the adopted notation avoids any confusion with the direct sum operator, yet used in this chapter.

4.4 Approximation theorem

While aesthetically pleasant and mathematically comforting, the given formalism is far away from an efficient description of how a neural network operates. As a matter of facts, a neural network can be used to create models of phenomena with an unclear nature, or that require a too complex model to be deduced analytically. In order to exploit such a feature a neural network must undergo a process called training. The Cybenko theorem (also known as universal approximation theorem in its general form), support the latter statement. Such a theorem, in its general form [55], states that a feedforward network with a single hidden layer containing a finite number of neurons (i.e., a multilayer perceptron), can approximate continuous functions on compact subsets of $\mathbb{R}^n$, under mild assumptions on the activation function. The theorem thus states that simple neural networks can represent a wide variety of functions when provided with an appropriate weights set, and it does not touch upon the algorithmic learnability of those parameters. The first versions of the theorem was proved by George Cybenko for sigmoid activation functions [56], while two years later Kurt Hornik showed [97] that it is not the specific choice of the activation function, but rather the multilayer feedforward architecture itself which gives neural networks the potential of being universal approximators. In this proof the output units are always assumed to be linear, on the other hand the general case can easily be deduced from the following set up. Let γ be a non-constant, bounded, and monotonically-increasing continuous function, then given a continuous function f defined in the M-dimensional unitary hypercube of $\mathbb{R}^M$ so that $f \in C([0,1]^M)$, it is possible to demonstrate that exists an approximation $\tilde{f}$ of the function f in the form

$$\tilde{f}_N\left(\underline{x}|W, \{\alpha_i\}_{i=1}^N, \{\beta_i\}_{i=1}^N\right) = \sum_{i=1}^{N} \alpha_i \gamma \left(\sum_{j=1}^{M} W_{ij} x_j + \beta_i \right) , \qquad W \in \mathbb{R}^{N \times M} \tag{4.13}$$

also granting that$\forall\ \varepsilon > 0\ \exists\ N_\varepsilon \in \mathbb{N}$ so that

$$\forall\ i \in [1, N_\varepsilon] \cap \mathbb{N}\ \exists\ \{\alpha_i, \beta_i \in \mathbb{R}\}_{i=1}^{N_\varepsilon} :\ \forall\ \underline{x} \in [0,1]^M \Rightarrow ||\tilde{f}_{N_\varepsilon} - f|| < \varepsilon \tag{4.14}$$

Notice that (4.14) does not give any constraint on the W, that because, independently from W, the (4.14) has a form of a limit. Indeed the theorem proofs that for any positive value of ε, as small as wished, it exists an approximation of f truncated to the N_ε-th order. Then it follows that

$$\lim_{N_0 \to \infty} \tilde{f}_{N_0}\left(\underline{x} | W, \{\alpha_i\}_{i=1}^{N_0}, \{\beta_i\}_{i=1}^{N_0}\right) = f(\underline{x}) \tag{4.15}$$

The (4.15) reveals then that, given γ with the specified properties, any $f \in C([0,1]^M)$ can be expressed in the form of a Fourier expansion

$$f(\underline{x}) = \sum_{i=1}^{\infty} \alpha_i \gamma \left(\sum_{j=1}^{M} W_{ij} x_j + \beta_i \right) \tag{4.16}$$

This also means that $\{f_N\}$ is dense in $C([0,1]^M)$ and it obviously holds replacing $[0,1]^M$ with any compact subset of $\mathbb{R}^M\ \forall\ M \in \mathbb{N}$. Finally it has to be noticed that γ presents all the properties of a transfer function, therefore it is immediately verified that (4.13) describes the output of a single layer neural network with M input neurons, N hidden neurons implementing a transfer function γ, a synaptic weights matrix W, neural biases β_i associated to the neurons, and amplification factors α_i resulting from the linear combination of outputs of each hidden neurons. The (4.13) is the very backbone of the mathematical structure that permits to model signals or make predictions by means of neural network. If carefully analyzed (4.13) actually states that it should be possible to approximate any function by means of a single layer neural network with no constraint on the related weights, since, the theorem hold also if we assume $W_{ij} = 1$. On the other hand this theorem does not give any information relatively to the optimum setup in terms of number of neurons or the approximation error held by $\tilde{f}$ or, above all, generalization capabilities. For this reason at the moment of the computational implementation it is ofter preferable to

select a fixed number of neurons N organized in one more than one layer, sometime implementing different transfer functions, and then solve an optimum problem by searching for the best performing weight matrix W. It is in this process that such a matrix gains a very important role for the neural network training.

4.5 Network training

The essence of the modeling capabilities of a neural network is the possibility to train the networks in order to approximate the often unknown mathematical functions at the basis of a certain phenomena. A successful network training should solve the problem of mapping a set of inputs into a set of known outputs, called targets, by reconfiguring the synaptic weights in order to reach an optimal approximation point. As explained before a neural network transforms an input $\underline{u}$ into an output $\underline{y}$ only after weights W has been assigned. The neural network training takes care of the selection and modification of the best performing weights. An adeguate selection of weight is responsible for the correct functioning of the neural network and it is at the basis of its capability to model phenomena and correlate inputs and outputs revealing the underlying model. Given a network topology $\mathcal{T}$, the topology-dependant sets of activation functions $\{\gamma^{(l)}\}_{\mathcal{T}}$, weights $\{W^{(l)}\}_{\mathcal{T}}$, and an input $\underline{u}$, the tuple $(\mathcal{T}, \{\gamma^{(l)}\}_{\mathcal{T}}, \{W^{(l)}\}_{\mathcal{T}}, \underline{u})$ represents the neural network to which is associated a discrete functional $\mathcal{N}_{\mathcal{T}}$ so that

$$\underline{y}_{\mathcal{T}} = \mathcal{N}_{\mathcal{T}}[\underline{u}] \tag{4.17}$$

After a topology $\mathcal{T}$ has been selected in terms of neurons, layers and connections, as well as the layer-related activations function set $\{\gamma^{(l)}\}_{\mathcal{T}}$, it is necessary to train the network searching for the optimal weights $\{w^{(l)}\}_{\mathcal{T}}$ in order modify the outputs $\underline{y}_{\mathcal{T}}$, which depends from the topology $\mathcal{T}$, in order to have those outputs to manifest the correct correlation with the inputs $\underline{u}$. The training procedure is generally performed starting from a set of known data, these latter set permits us to adjust the weights embedded in the network in order enable it to correlate the inputs set to the yet known outputs set. The known outputs used to train the network constitute a target point

for the association capabilities of the neural network, therefore this set is also called targets set. To grant consistency with the algebraic formalism used in chapter, the targets will be now interpreted as a vector and the target vector will be represented as $\underline{t}$. In this form the neural network training corresponds to the resolution of the functional equation

$$\tilde{\mathcal{N}}_{\mathcal{T}}[\underline{u}] \approx \underline{t} \tag{4.18}$$

Please notice that in (4.18) we denote with $\tilde{\mathcal{N}}_{\mathcal{T}}$ the neural network, with a slightly different notation with respect to $\mathcal{N}_{\mathcal{T}}$ as used in (4.17). The difference is due to the interpretation of the neural network status in these equations: while in (4.17) the neural network is interpreted as a finalized structure that generates an output vector $\underline{y}$ starting from an input vector $\underline{u}$, in (4.18) both the input vector $\underline{u}$ and the target vector $\underline{t}$ are known, while the network $\tilde{\mathcal{N}}_{\mathcal{T}}$ is being adjusted in order to obtain the best possible approximation of $\underline{t}$. In general the procedure is performed starting by several configurations taking into account families of networks with similar topologies and activation functions, on the other hand neglecting any constraint on the degrees freedom related to topologies and functions, it would be impossible, both for humans and computers, to tackle the problem. A more pragmatic and practical approach then would start considering a certain topology by reducing the freedom degrees (e.g. fixing the number of layers and the adopted activation functions) and then, considering a selected number of neurons, trying to adjust the weights of the network. Let now suppose that a certain phenomena is obser and that As a consequence of the Cybenko theorem in its maximum generalization, we can assume that, fixed the number of layers and the related activation functions, it exists at least one perfect topology $\mathcal{T}_\star$ so that, given two vectors $\underline{u}$ and $\underline{y}$ it follows that

$$\mathcal{N}_{\mathcal{T}_\star}[\underline{u}] = \underline{y} \tag{4.19}$$

on the other hand such a perfect topology $\mathcal{T}_\star$ is unknown, and, since it is up to the human choices to shape the topology of a neural network, it is also highly improbable to easily select such a $\mathcal{T}_\star$ among all the possible combinations. For this reason, starting from an arbitrary topology $\mathcal{T}$ the only applicable solution on the problem is to modify

the weights in order to permit to the network $\mathcal{N}_{\mathcal{T}}$ to act as closely as possible with respect to $\mathcal{N}_{\mathcal{T}_\star}$.

4.6 Learning from errors

The training procedure starts from a random state of $\mathcal{N}_{\mathcal{T}}$, more in detail i starts with random weights, on the other hand during the training some choice functions are to be applied in order to modify such weights. Each step of the training procedure is called epoch and it ends after the weights of the network, and consequently its status, has been modified, and the network is then ready to undergo a new training epoch. During each epochs the performance of the network must be estimated in order to proceed further. Let consider a trainee network and let $\mathcal{N}_{\mathcal{T}}^{\tau}$ be the network state after its t-th training epoch, then it follows that, at each epoch τ, we can obtain an estimate of the network performances starting from the distance of the given output $\underline{\tilde{y}}(\tau))$ with respect to the expected output, therefore with respect the target $\underline{t}$. Then it is necessary to obtain an estimate of the network accuracy after each training epoch by using some kind of error function. It is of course possible to define different kind of error functions, for practical reasons only one example will be given. Let immagine the output to be M-dimensional so that $\underline{t}, \underline{\tilde{y}}(\tau) \in \mathbb{R}^M \ \forall \ \tau \in \mathbb{N}$. It is possible to compute the vectorial distance $\underline{\delta}(\tau)$ among those vectors as

$$\underline{\delta}(\tau) = \underline{y}(\tau) - \underline{t} \tag{4.20}$$

and consequently to define an error function, like the Mean Square Error $E_{\text{MSE}}(\tau)$, as

$$E_{\text{MSE}}(\tau) = \frac{1}{M} \sum_{m=1}^{M} \delta_m^2(\tau) = \frac{1}{M} \sum_{m=1}^{M} \left(y_m(\tau) - t_m\right)^2 \tag{4.21}$$

In this particular case the Mean Squared Error (MSE) is also called instantaneous error energy $\mathscr{E}(\tau)$. Each successive training epochs should reduce the value of $\mathscr{E}(\tau)$. so that

$$\mathscr{E}(\tau + k) \leqslant \mathscr{E}(\tau) \quad \forall \ \tau \in \mathbb{N} \tag{4.22}$$

So that for an ideal training algorithm it should be possible to obtain

$$\lim_{\tau \to \infty} \mathscr{E}(\tau) = 0 \tag{4.23}$$

On the other hand it is not possible to reach such a result due to a lot of different reasons from the unknown analytical structure of such ideal training procedure, which should also depend by the problem itself, as well as due to the numerical approximations occurring when using a finite precision calculator. Moreover it is not possible to conceive a properly defined algorithm requiring an infinite number of epochs in order to assure the null error energy condition of (4.23). For these reason during the training algorithms are mainly concerning the step-by-step minimization of $\mathscr{E}(\tau)$ with all related the consequences in terms of possible loops, deadlocks, local minima etc.. In general the most diffuse training algorithms differently implement the delta minimization rule, also called minimization process of Widrow-Hoff [201]. In order to understand the process let suppose, with vivid imagination, that in order to train the neural network we could focus only on the last layer, therefore modifying only the synaptic weights $w^{(L)_k m}$ related only to the neuron in the L-th layer, where L is the total number of layers in the network. In this scenario after the τ-th training epochs, following the Widrow-Hoff rule, the weights will be updated by adding a factor

$$\Delta w_{km}^{(L)} = \eta \delta_m(\tau) x_k^{(L-1)}(\tau) \tag{4.24}$$

where η identifies a constant called learning rate. In other words the Widrow-Hoff rule states that the status update must be proportional to the error signal as well as the input signal. Of course, from a software point of view it means that the error signal must be measurable and visible, as well as the inputs to the neurons. Such visibility is often an important concern at the moment of the parallelization of neural networks in high performances environments [143, 27, 22, 26]. From (4.24) it follows that at each epochs $(\tau + 1)$ the weights status will have been modified with respect to the previous status at τ, therefore

$$w_{km}^{(L)}(\tau + 1) = w_{km}^{(L)}(\tau) + \Delta w_{km}^{(L)}(\tau) \tag{4.25}$$

Since the (4.25) correspond to an evolution equation for the layer weights vector $\underline{w}^{(L)}$ it is possible to rewrite it as

$$\underline{w}^{(L)}(\tau+1) = \hat{U}_{\tau}^{(L)}\underline{w}^{(L)}(\tau) \tag{4.26}$$

where $\hat{U}_{\tau}^{(L)}$ represents the evolution operator for the L-th layer at a time step τ. From this basis it is possible to extend the application to consider multilayer neural network. This will be done by propagating from the last to the first layer the necessary adjustments to the related synaptic weight, for this reason the algorithm is called backpropagation.

4.7 Backpropagation algorithm

To correctly define the backpropagation algorithm it is necessary to introduce a general simplified notation for the indexes in order to narrow the upcoming formulas. In the following the latin indexes (i,j,k...) will be used to identify neurons in different layers, moreover we will label the layers with the same letter used as index for their neurons, but in uppercase form. In this manner a neuron indexed with i will belong to the layer I, a neuron indexed with j will belong to the layer J, and so on. Finally the layers will also be ordered basing on the letters, therefore layer I will precede layer J that will precede layer K and so on. Within the given notation, at the n-th step, we will call $d_j(n)$ the desired output of the j-th neuron within the J layer, while the obtain output from the same neuron will be written $x_j(n)$, this latter is also called signal function. Following the instructions it is possible to define the error signal of the j-th neuron a the n-th time step as

$$e_j(n) = x_j(n) - d_j(n) \tag{4.27}$$

and and instantaneous energy error value for the j neuron at the n-th time step as

$$E_j(n) = \frac{1}{2}e_j^2(n) \tag{4.28}$$

Similarly to (4.22) it is possible to define an instantaneous value for the total error energy as

$$\mathscr{E}(n) = \sum_j E_j(n) = \frac{1}{2}\sum_j e_j^2(n) \tag{4.29}$$

The backpropagation algorithm considers the transfer functions of the neurons as an induction field, therefore it exist an ideal local field $v_j(n)$ representable as

$$v_j(n) = \sum_j w_{ij}(n)x_j(n) \tag{4.30}$$

so that the function signal $x_j(n)$, exiting as output of the j-th neuron of layer J at the n-th step is

$$x_j(n) = \gamma^{(J)}[v_j(n)] \tag{4.31}$$

where $\gamma^{(J)}$ represents the transfer function for the layer J. The back propagation algorithm then updates the weights $w_{ij}(n)$ by means of an additional variation $\Delta w_{ij}(n)$ proportional to the partial derivative of $\mathscr{E}(n)$ with respect to $w_{ij}(n)$. On the other hand, applying the derivation rule for composite functions it also follows that

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ij}(n)} = \frac{\partial \mathscr{E}(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial x_j(n)}\frac{\partial x_j(n)}{\partial v_j(n)}\frac{\partial v_j(n)}{\partial w_{ij}} \tag{4.32}$$

Differentiating both the members of (4.29) with respect to $e_j(n)$ it also follows

$$\frac{\partial \mathscr{E}(n)}{\partial e_j(n)} = e_j(n) \tag{4.33}$$

Moreover differentiating (4.27) with respect to $x_j(n)$ it results

$$\frac{\partial e_j(n)}{\partial x_j(n)} = -1 \tag{4.34}$$

and trivially (4.31) with respect to $v_j(n)$ it is possible to write

$$\frac{\partial x_j(n)}{\partial v_j(n)} = \gamma^{(J)\prime}[v_j(n)] \tag{4.35}$$

where $\gamma^{(J)'}$ represents the first total derivative of $\gamma^{(J)}$ with respect its argument. Finally differentiating (4.30) with respect to $w_{ij}(n)$ then it results

$$\frac{\partial v_j(n)}{\partial w_{ij}(n)} = x_j(n) \tag{4.36}$$

Substituting (4.33), (4.34), (4.7) and (4.35) in , (4.32) it trivially follows that

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ij}(n)} = -e_j(n)\gamma^{(J)'}[v_j(n)]x_j(n) \tag{4.37}$$

It is now useful to recall the definition of local gradient $\Gamma_j^{(J)}(n)$ as

$$\Gamma_j^{(J)}(n) = \frac{\partial \mathscr{E}(n)}{\partial v_j(n)} = \frac{\partial \mathscr{E}(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial x_j(n)}\frac{\partial x_j(n)}{\partial v_j(n)} \tag{4.38}$$

And basing on the same substitution operated to obtain (4.37), from (4.38) it is possible to deduce

$$\Gamma_j^{(J)}(n) = e_j(n)\gamma^{(J)'}[v_j(n)] \tag{4.39}$$

Eventually, applying the delta rule, it is possible to compute the weights update factor as

$$\Delta w_{ij}(n) = -\eta\frac{\partial \mathscr{E}(n)}{\partial w_{ij}(n)} = \eta\Gamma_j^{(J)}(n)x_j(n) \tag{4.40}$$

so that

$$w_{ij}(n+1) = w_{ij}(n) + \eta\Gamma_j^{(J)}(n)x_j(n) \tag{4.41}$$

It appears now obvious that the weights update procedure in the backpropagatio algorithm is driven by the layer local gradient $\Gamma_j^{(J)}(n)$. This gradient, as visible in (4.39) is computed for each neuron from the product of the related error signal and the total derivative of the activation function with respect to its argument. The back propagation algorithm is then applied in two steps: a forward and a backward step. The forward pass does not update yet the weights while, instead, it computes the function signals x_j of the network on all the neurons starting from the local field $v_j(n)$. As the name suggest the forward step proceeds from left to right, from the

first to the last layer of the network. After the forward step, the backward step, viceversa, will proceed from right to left, therefore from the last to the first layer of the network. During the backward step the local gradients $\Gamma_j^{(J)}$ are computed for each neuron and consequently the related updates Δw_{ij}. Finally the weights w_{ij} are modified accordingly to the computed updates Δw_{ij}.

4.8 Stopping criteria

For many reasons, as explained previously, the training algorithm cannot reach a zero error point, on the other hand a well performing training algorithm should manifest a certain convergence until minimum is reached and it is not possible to improve it. Moreover there is no fixed rule to determine when to stop the training procedures as well as it is not possible to predetermine a well defined number of training epochs. On the other hand some criteria can be applied accordingly to the logical structure of the back propagation algorithms. One of the most important and applicable criterion on the stopping conditions has been formulated by Karmer and Sangiovanni-Vinticelli [113] and is well known in literature as the KSV principle. Basing on the KSV rule, the algorithm is considered concluded by convergence when the euclidean norm of the error vector lowers down to a certain threshold. The main problems of the KSV principle is that it is impossible to estimate a priori if the prefixed threshold can be reached and how much time it should require to be reached, moreover it requires to compute the euclidean norm of the error vector at each step, which is computationally unadvisable. On the other hand, with a different approach, Haykin [93] suggests to consider a more suitable convergence condition by fixing a threshold not just for the error itself but for the absolute gradient ratio for the total error energy (e.g. estimated by means of the MSE). This approach considers the algorithm converged when it is not possible anymore to lower the error of a considerable quantity. The selection of a stop criterion also concern the resulting trained network and its generalisation capabilities, the Haykin's criterion ha been now generally adopted in literature, as well as for the presented approach, because this criterion make it possible to obtain good enough generalisation capabilities of the trained network. As a matter of facts, the intrinsic

nature of the training procedure is prone to the so called polarisation effect. This effects is highly probable when unsuitable stop criteria or learning rates are selected for the training procedure, on the other hand can be easily triggered by numerical effects given by the coupling among certain topologies, the related activation functions and the datasets used to train the network. When a network polarises it outperform the expected approximation capabilities announced by the Cybenko theorem, on the other hand it become able to perfectly reproduce the model only for the given training set, while missing its goal when exposed to new inputs. In facts in order to keep under control the polarisation effect during the training procedure the network is also regularly exposed to a small set of data, called validation set, in order to verify if the neural network predicts within the same errors both the targets and the validation targets while not trained on the latter. This procedure is also part of the so called early stopping method [74] which as been prove to be one of the best training methods actually implemented. The early stopping method makes use of three separated data sets: a training set, a validation set and a test set then monitoring the performances of the network while trying to predict these different sets. Early stopping is in facts a typical methods to regularise parametric regression problems as the problem of neural network training. For a given input space X output space Y and samples drawn from an unknown probability measure $\rho : X \times Y \rightarrow [0, 1] \subset \mathbb{R}$, the goal of a regression problem is to approximate a regression function $f_\rho : X \rightarrow Y$ given as

$$f_\rho(x) = \int_Y y d\rho(x, y) \tag{4.42}$$

where $\rho(x, y)$ is the conditional distribution at x induced by ρ [186]. Regularisation is, therefore, especially important for these methods. The early stopping rule solve the regularisation problem using an iterative procedure such as gradient descent and analysing at each step the upper bounds on the generalization error as a function of the iteration number [211]. Moreover, while the global error of the network is evaluated basing on the training set, it is the error on the prediction of the validation set that permits to evaluate the generalisation capabilities. Those two errors should proceed tighter, in facts if the training errors drops down dramatically while the validation

error increases ti ca be interpreted as a clue of possible polarisation. Finally the test set is used independently independently from the predecessors in order to obtain a finalised estimate of the network performances after the training. Finally it must be highlighted that pushing further the training, far over the early stopping point, where the training error set decreases and the validation error increases, it is equivalent to train the network to overfit the training set, or, in other words, to train the network to reproduce only a specific portion of a signal and the related noise, being the incapable to reconstruct any different signal from a different input.

4.9 Recurrent neural networks

The neural networks model has been implicitly describe until now as feedforward neural networks. On the other hand such models lacks of a temporal dynamics since are not capable of retain any memory of the past data. As a matter of facts the temporal dynamics is a paramount characteristic in order to model time-evolving phenomena. this kind of dynamics characterises the Recurrent Neural Networks (RNN), which are so called due to the typical recursive feedback that are typically used to reintroduce as inputs the outputs coming from a previous temporal stage, therefore introducing memory by means of delayed input lines. Starting from the recurrent neural network topology several different kind of neural architectures can be built (e.g. fully recurrent or layer recurrent, with complete or incomplete feedbacks, with or without delayed lines etc...). While the explained characteristics makes the network suitable for time-evolving models, the presence of the described feedbacks and delays introduces instabilities that mainly outburst during the training procedure. Moreover the over-structure represented by feedbacks and delayed lines introduces more degrees of freedom on the topological characteristics of the network, therefore it makes even harder to find an optimal setup that, ultimately, has to be constructed by a try and check process. While is not possible to discuss here all the stability conditions concerning Recurrent Neural Networks, it is available a quite extensive literature on the matter [10, 50, 89, 96, 172]. From this moment the terminology Recurrent Neural Network will be used referring to fully connected multilayer networks with global

feedbacks, input delay lines but no internal layer feedback if not differently specified. In facts the mathematical model for such a kind of topology naturally comes from the yet described model of feedforward neural network. A Recurrent Neural Network, as told, is composed by a single input vector $\underline{u}$ with several delay lines (also called memories) and delayed feedbacks from the output vector $\underline{y}$. Let now suppose to have q delay lines for the inputs, as well as r delay lines from the output. In this scenario, at a time step n, the input values are represented as $\underline{u}(n)$, while the output values as $\underline{y}(n)$. On the other hand, with this topology, the signal vector $\underline{x}^{(0)}$ constituting the input layer, or if preferred, the input to the first hidden layer, is not equivalent to the input vector $\underline{u}$, since it is also constituted by the relative input delays and output feedbacks, therefore it will be

$$\underline{x}^{(0)}(n) = \left[\underline{u}(n)|\cdots|\underline{u}(n-q)|b|\underline{y}(n-1)|\cdots|\underline{y}(n-r)\right] \tag{4.43}$$

where b is a constant, or bias, used for stability and rescaling purposes. In this case the $\underline{u}(n)$ vector will be also called exogenous input vector, since coming from an external source with respect to the network itself, while the delayed elements of $\underline{y}$ will be called regressive set, since originated within the network in previous time steps. This kind of network topology is therefore also called NARX (Nonlinear AutoRegressive with eXogenous inputs). Starting from this topology it is possible to create infinite different similar and affine networks called generalised NARX networks where also the possibility of internal layer or multilayer feedbacks is introduced. This latter represents the most general form of artificial neural network and is used to create functional models on sates spaces in order to simulate finite states automation [137, 114] which make it possible to use this kind of topology in order to reconstruct, model or simulate phenomena with a temporal dynamics [93]. Of course the modeling capabilities of such network are entirely dependent by the success of the adopted training procedure.

4.10 Real time learning algorithm

On the field of training algorithms a lot of different choices are available, on the other hand, as generalized by Williams and Zipser [203], those algorithms are mainly classifiable in two principal categories: epochwise training and continuous training. In the fist kind are classified all those algorithms that starts from an initial random state of the network and than proceed with the training sequence until a predefined number of epochs has been reached making it possible for the network to reach a new state. In such conditions the training is thens topped and the network reset to a new initial state. The algorithms classifiable as continuous, on the other hand, does not require any reset or stop while proceeding trough different training epochs. On the other hand, while in this latter option the network states are continuously updated, such an approach does not take advantage of any predefined stop condition. Since no predefined stop condition is defined, then an external stopping criterion must be adopted by mens of a control agent. The stopping criteria generally applied in literature with this kind of algorithms belong to the yet devised category of the early stopping criteria. Different training algorithms presents several generalizations and modifications in order to make them suitable for both the approaches. On the other hand it is also common to attribute different names to very similar algorithms basing on the implemented approach: e.g. the yet defined back propagation algorithm, when strictly applied to an epochwise training paradigm is called BackPropagation Trough Time algorithm (BPTT), while, on the contrary, if applied within a strictly continuous training paradigm is called Real Time Recurrent Learning algorithm (RTRL). On the other hand, while applied to different paradigms, those two algorithms retains and commonly share several main characteristics that are typical of the back propagation. As a matter of fact, among all the other common features, the most important is related to the weight update procedure that, in both the implementations, is based on a gradient descent approach. Another important advantage on the adoption of the back propagation approach si the standardized implementation that, due to its robustness and reliability, often compensates the long convergence time which typically characterizes this approach. On the other hand, focusing on a continuous

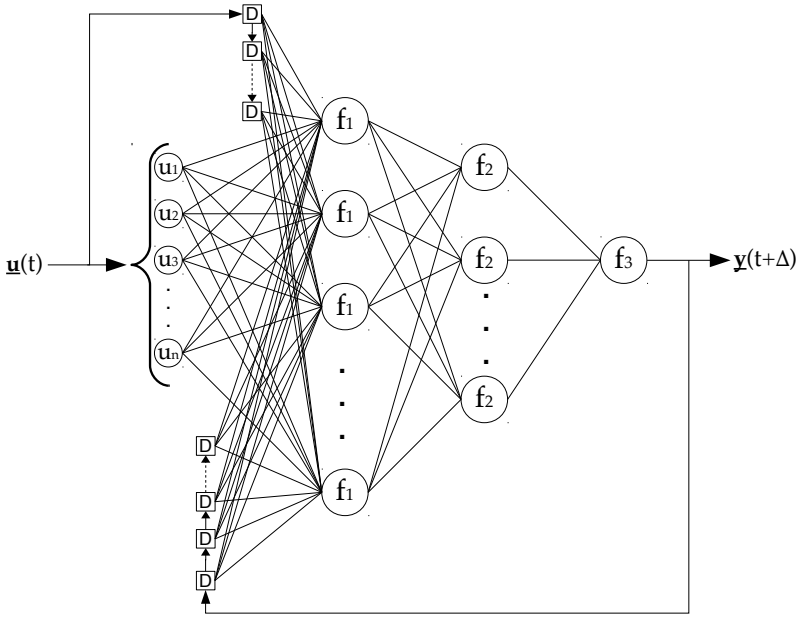

Figure 4.2: A typical model of nonlinear autoregressive with exogenous inputs recurrent neural network.

training approach, also more commonly used to train the nonlinear autoregressive with exogenous inputs neural networks, the real time recurrent learning algorithms is also advantageous because it uses the minimum possible information since the updates procedure computes the instantaneous gradient values in order to determine the update parameters for the synaptic wights embedded within the network. As a matter of facts the name of real time recurrent learning algorithms derives from the techniques adopted to update the neural network synaptic weights, in facts the updates are performed while the algorithms continues to process the signal functions without any interruption [202]. Due to the optimal performance of the approach this has been adopted for this work. Now the algorithmical model will be described. Let start from a general nonlinear autoregressive neural network with exogenous inputs as in Fig. 4.2, and let suppose that it consist of q neurons with m exogenous inputs and p outputs. As seen before the input layer of the network will results from the concatenation of the exogenous input vector with both the delayed inputs and the regressive set given as feedback from the previous output states. For simplicity let

differ this ideal network from Fig. 4.2 supposing that only one hidden layer is present (the counter generalization is trivial). It is possible to describe the non linear states space by the non linear system of equations

$$\begin{cases} \underline{x}(n+1) &= \tilde{\varphi}\left[\hat{W}\underline{x}(n) + \hat{V}\underline{u}(n)\right] \\ \underline{y}(n) &= \hat{C}\underline{x}(n) \end{cases} \tag{4.44}$$

with $\hat{W} \in \mathbb{R}^{q\times q}$, $\hat{V} \in \mathbb{R}^{q\times(m-1)}$ and $\hat{C} \in \mathbb{R}^{p\times q}$ and finally $\tilde{\varphi} \in \mathbb{R}^q \to \mathbb{R}^q$ is a topological map function so that

$$\forall\ \underline{x} = [x_i]_{i=1}^q \quad \longrightarrow \quad \tilde{\varphi}\,[\underline{x}] = [\varphi(x_i)]_{i=1}^q \tag{4.45}$$

for a generic transfer function $\varphi : \mathbb{R} \to \mathbb{R}$. Therefore the system (4.44) maps by means of φ the status of each neuron, in facts, expanding the first equation of (4.44) and transforming it in explicit variables it follows that

$$\underline{x}(n+1) = \left[\varphi\left(\hat{w}_j^{\intercal}\underline{\xi}(x)\right)\right]_{j=1}^q \tag{4.46}$$

While equation (4.46) is symbolic, in order to give it a meaning it must be assigned the values of the adopted a equivalent weights matrix $\hat{w}$ as

$$\hat{w}_j = \begin{bmatrix} \left[\hat{W}_{ij}\right]_{i=1}^q \\ \\ \left[\hat{V}_{ij}\right]_{i=1}^q \end{bmatrix} \quad \forall\ j \in [1,q] \cap \mathbb{N} \tag{4.47}$$

and defining the equivalent input signal function $\underline{\xi}$ as

$$\underline{\xi}(n) = \begin{bmatrix} \underline{x}(n) \\ \underline{u}(n) \end{bmatrix} \tag{4.48}$$

where $\underline{x}(n)$ is the input signal function represented by a vector of $q \times 1$ elements and $\underline{x}(n)$ the exogenous inputs vector composed of $(m+1) \times 1$ elements, where the $(m+1)$-th element is the bias constant. In order to simplify the indexes which

would otherwise require a too heavy notation several matrix have to be introduced: $\hat{\Lambda}_j(n)$, $\hat{U}_j(n)$, and $\hat{\Phi}(n)$. The first is $\hat{\Lambda}_j(n) \in \mathbb{R}^{q\times(q+m+1)}$ and is defined as the partial derivative of the input signal function vector with respect to the weights associated linearized vector

$$\hat{\Lambda}_j(n) \triangleq \frac{\partial \underline{x}(n)}{\partial \hat{w}_j} \quad \forall\ j \in [1,q] \cap \mathbb{N} \tag{4.49}$$

Also the second matrix belongs to the same space so $\hat{U}_j(n) \in \mathbb{R}^{q\times(q+m+1)}$ but is defined as null matrix excepted for the j-th rows which is identically equal to the transposition of $\underline{\xi}(n)$, therefore

$$\hat{U}_j(n) \triangleq \left[\delta_{ij}\underline{\xi}^{\intercal}(n)\right]_{i=1}^{q} \tag{4.50}$$

where δ_{ij} is the Kronecker delta. Finally the third is a diagonal matrix $\hat{\Phi}(n) \in \mathbb{R}^{q\times q}$ so that the only non null elements, on the first diagonal, are defined as the first derivate of the activation functions with respect to their argument so that

$$\hat{\Phi}(n) \triangleq \left[\delta_{ij}\varphi'\left(\hat{w}_j^{\intercal}\underline{\xi}(n)\right)\right]_{i,j=1}^{q} \tag{4.51}$$

again where δ_{ij} is the Kronecker delta. Within the given definitions it is then possible to differentiate (4.46) with respect to $\hat{w}_j$ by applying the composite functions derivation rule and obtaining the following recursive equation

$$\hat{\Lambda}_j(n+1) = \hat{\Phi}(n)\left[\hat{W}(n)\hat{\Lambda}_j(n) + \hat{U}_j(n)\right] \quad \forall\ j \in [1,q] \cap \mathbb{N} \tag{4.52}$$

The (4.52) finally perfectly describes the temporal evolution of the dynamical non linear states of an nonlinear auto recursive with exogenous inputs recurrent neural network during the realtime recurrent learning procedure. On the other hand one piece is still missing in order to complete the analytical description of the process: the correlation among the matrix $\hat{\Lambda}_j(n)$ to the error gradient computation. Starting from (4.44), at each time step n, it is possible to obtain a vector $\underline{e}(n) \in \mathbb{R}^{p\times 1}$ representing the error with respect to an expected value $\underline{e}(n) \in \mathbb{R}^{p\times 1}$ so that

$$\underline{e}(n) = \underline{d}(n) - \underline{y}(n) = \underline{d}(n) - \hat{C}\underline{x}(n) \tag{4.53}$$

and immediately follows the definition of the total error at a time step n as

$$\mathscr{E}(n) = \frac{1}{2}\underline{e}^{\intercal}(n)\underline{e}(n) \tag{4.54}$$

In this notation the final goal of the learning process is not just the reduction of the instantaneous error itself, but the minimization of the cumulative sum of the total error trough the time steps $\mathscr{E}_{\text{TOT}}$ defined as

$$\mathscr{E}_{\text{TOT}} = \sum_{n} \mathscr{E}(n) \tag{4.55}$$

Differentiating (4.54) with respect to $\hat{w}_j$ also follows

$$\frac{\partial \mathscr{E}(n)}{\partial \hat{w}_j} = \frac{\partial \underline{e}^{\intercal}(n)}{\partial \hat{w}_j}\underline{e}(n) = -\hat{C}\frac{\partial \underline{x}(n)}{\partial \hat{w}_j}\underline{e}(n) = -\hat{C}\hat{\Lambda}_j\underline{e}(n) \quad \forall\ j \in [1, q] \cap \mathbb{N} \tag{4.56}$$

Then it follows that the weight update applicable to $\hat{w}_j(n)$ relative to the j-esime neuron at a time step n is immediately computed as

$$\Delta\hat{w}_j(n) = -\eta\frac{\mathscr{E}(n)}{\partial \hat{w}_j} = \eta\hat{C}\hat{\Lambda}_j(n)\underline{e}(n) \quad \forall\ j \in [1, q] \cap \mathbb{N} \tag{4.57}$$

where again η represents the learning rate and the matrix $\Lambda_j(n)$ results self deterministic by means of the autoregression expressed in (4.52). In this set up only one freedom degree is allowed and depends by the initial conditions of the system which in the practice are generally random. Just for completeness reason here those initial conditions can be defined as

$$\hat{\Lambda}_j(0) = 0 \quad \forall\ j \in [1, q] \cap \mathbb{N} \tag{4.58}$$

It immediately follows that the computational complexity of such the modeled real-time recursive learning algorithm is of $\mathscr{O}(N_w N_s^2 L)$ operations with N_w weights, N_s states and L epochs. Finally, as it has been anticipated in Chapter 3, we will introduce in the following chapter our developed methodology to bound both Neural Networks and Wavelet Analysis.

CHAPTER 5

The next generation

> Prediction is very difficult,
> especially if it's about the future.
>
> Niels Bohr

In the previous chapters the basis concepts of wavelet analysis and neural networks has been introduced. The combined usage of both the wavelets and Recurrent Neural Networks (RNN), and more specifically, of the yet explained models autoregressive recurrent neural networks with exogenous inputs, permits us to create new tools for prediction, forecast and modeling purposes: the Wavelet Recurrent Neural Networks. The Wavelet Recurrent Neural Network architecture introduces a great deals of improvement in the accuracy of a forecasting model since this is also achieved by using the properties of a wavelet-based transform by means of a combination of artificial neural networks (ANNs). As it will be shown the Wavelet Recurrent Neural Networks are able to reconstruct a signal from wavelet coefficients, on the other hand those networks are also able to predict the wavelet coefficients in a future time step and then to reconstruct and predict the signal. This technique permits a better prediction and several advantages. In order to understand how this architecture works, the second-generation wavelets must be introduced too, since those play an important role in the

Wavelet Recurrent Neural Network model. As a matter of fact this is the key point to understand the novelty of the approach since, until now, in literature, the various attempt to create wavelet neural networks has been limited to simple feed-forward working on the wavelet coefficients domain [57, 99, 216, 188], on the other hand here a novel contribution is presented.

5.1 Second generation wavelets

While the wavelet analysis gives us a powerful mathematical tool, in order to become suitable to our purposes some limitations must be worked out. In particular the wavelet transform is usually designed for infinite or periodic signals but it is sometime tricky its adaption to a bounded domain. While in the wavelet theory it is quite common to describe a signal as a function in $L^2(\mathbb{R})$, in the real practice signals are not continuous function of integrable square, the signal domain is far from being infinite, and, finally, signals are not periodic. Moreover, due to the digitalization procedure, such signals are sampled, and also, often, not regularly sampled. Therefore it became inconsistent with the problem a mathematical model that tends to analyze functions defined of surfaces or manifolds of a regular, flat and continuos metric field. That said, in the previous decade, a non trivial problem was raised on the applicability of the wavelet analysis to the real-world signals, or, at least, if there were any possibility to find a different approach while preserving the nature and the properties of the wavelet transform. Mainly it become necessary to extend beyond the regularity of a flat geometry the frequency localization and scaling properties of the wavelet, as well as their suitability for digital analysis due to their representability as digital filters. The solution was originally proposed by two independent works of David Donoho and Harten at the beginning of 1990s and the answer was to abandon the Fourier-like analytical models as well as the classical approach based on shifting and scaling of mother functions. The basic idea, which inspired its name,is to start with a very simple or trivial multiresolution analysis and gradually work ones way up to a multiresolution analysis with particular properties. The lifting schema allows one to custom-design the filters needed in the transform algorithms to the situation at hand.

In this sense, it provides an answer to the algebraic stage of a wavelet construction [45]. Wavelet functions ψ_{jm} are traditionally defined as the dyadic translations and dilations on one particular $L^2(\mathbb{R})$ function, i.e., the mother wavelet ψ so that

$$\psi_{jm} = \psi(2^j x - m) \tag{5.1}$$

Of course (5.1) refers to a first generation wavelet construction schema. On the contrary the lifting schema makes use of a more general setting where the wavelets are not necessarily translations and dilations of each other but still retains all the powerful properties of first generation wavelets. As a matter of facts in the second generation wavelet transform (SGWT) there are not mother wavelets or filters explicitly designed ab initio, in factor the transform consist of a sequence of steps of lifting, therefore this approach is also called lifting schema. Moreover it has to be noticed that the lifting schema, once known, can be represented as a regular discrete wavelet transform. On the other hand this latter is an unnecessary procedure since the lifting schema itself provide both the design and the application of the second generation wavelet transforms by its own. This happens because the lifting schema avoids the necessity to design the wavelet transform in the frequency domain as for the classical wavelet transform or, as it will be called from now, the first generation wavelets such as the Discrete Wavelet Transform and the Continuous Wavelet Transform.

5.2 The lifting schema

The complete formulation of the lifting schema is due to Win Sweldens [189] and presents a both general and simple but quite powerful tool, to construct second generation wavelets. The so called lifting is made by a space-domain construction of biorthogonal wavelets. The basic idea is to first split a signal into its even and odd samples. Let the signal be representable as an array of $2N$ values $\underline{x}$, therefore it is

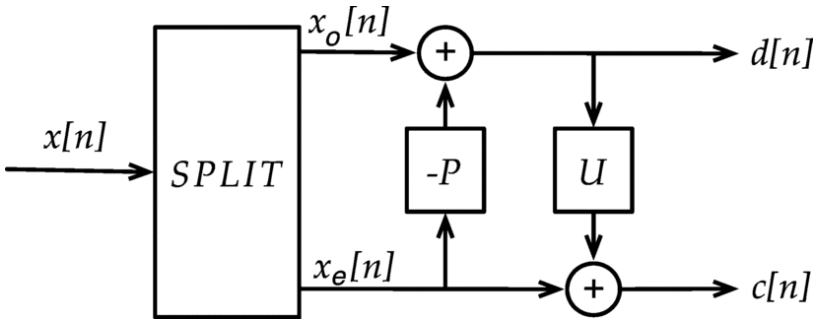

Figure 5.1: The lifting schema and its stages: split, predict, and update.

possible to split it into two half-sized arrays $\underline{x}_o$ and $\underline{x}_e$ so that

$$\begin{aligned} \underline{x}_o &= [x_{2k-1}]_{k\in[1,N]\cap\mathbb{N}} \\ \underline{x}_e &= [x_{2k}]_{k\in[1,N]\cap\mathbb{N}} \end{aligned} \tag{5.2}$$

where $\underline{x}_o$ is called odd set and $\underline{x}_e$ is called even set with respect the original signal $\underline{x}$. It is now necessary to recall that wavelet theory is mainly based on the locality of the signal structure which is highly correlated. The goal of the lifting schema is the to predict the odd set $\underline{x}_o$ starting from the values of the even set $\underline{x}_e$. The odd set $\underline{x}_o$ will take the same role of the residuals in the first generation wavelet since is as a matter of facts it is low-resolution representation of $\underline{x}$, while the unpredictable portion of $\underline{x}_e$, which will be determined as a prediction error, will constitute the details. The even samples $\underline{x}_e$ then needs to be adjusted in order to represent the coarse version of the original signal. These adjustments are performed so that the average of the fine and coarse version of the signal remains constant. The procedure can be summarized in the following steps

1. Split $\underline{x}$ in $\underline{x}_o$ and $\underline{x}_e$

2. Predict $\underline{x}_o$ from $\underline{x}_e$

3. Update $\underline{x}_e$ using $\underline{x}_o$

The prediction stage consists in generating the wavelet details $\underline{d}$ as the error in predicting $\underline{x}_o$ from $\underline{x}_e$ using a prediction operator $\hat{P}$ so that

$$\underline{d} = \underline{x}_o - \hat{P}\underline{x}_e \tag{5.3}$$

The update stage consists in combining $\underline{x}_e$ and $\underline{d}$ to obtain the scaling coefficients $\underline{c}$ which represent a coarse approximation to the original signal $\underline{x}$. This is accomplished by applying an updating operator $\hat{U}$ to the wavelet coefficients and adding the resulting vector to $\underline{x}_e$ so that

$$\underline{c} = \underline{x}_e + \hat{U}\underline{d} \tag{5.4}$$

Tese three steps form a lifting stage. This procedure is repeated until it ends up with a signal consisting of a single number equal to the average value of the original signal. Iteration of the lifting stage on the output $\underline{c}$ creates the complete set of discrete wavelet transform (DWT) scaling and the wavelet coefficients $\underline{c}^{(j)}$ and $\underline{d}^{(j)}$. The lifting steps are easily inverted even if $\hat{P}$ and $\hat{U}$ are nonlinear or space-varying. Rearranging (5.3) and (5.4) it follows the non linear system describing the lifting schema (Fig. 5.1):

$$\begin{cases} \underline{x}_e = \underline{c} - \hat{U}\underline{d} \\ \underline{x}_o = \underline{d} + \hat{P}\underline{x}_e \end{cases} \tag{5.5}$$

The described lifting schema also leads to a fast in-place calculation of the wavelet transform, i.e., an implementation that does not require auxiliary memory. In the lifting framework, the update structure depends on the predictor structure. Hence, if $\hat{P}$ is space-varying or nonlinear, then so is $\hat{U}$, and the design procedure becomes unwieldy. A crafty detour around this problem is to perform the update step first, followed by the prediction. The relevant equations then become

$$\begin{cases} \underline{c} = \underline{x}_e + \hat{U}\underline{x}_o \\ \underline{d} = \underline{c} - \hat{P}\underline{x}_e \end{cases} \tag{5.6}$$

After designing an update filter to preserve the first M low-order polynomials in the data, we can apply any space-varying or nonlinear predictor without affecting the coarse approximation $\underline{c}$. Since the update and predict lifting stage creates $\underline{c}$ prior to $\underline{d}$, the prediction operator can be designed to optimize the performance criteria in addition to polynomial suppression capability [45]. To see the method more clearly, let us say that we have a signal $\underline{s}^{(k)}$, where j represents the sampling level. The lifting procedure yields $\underline{s}^{(j-1)}$ and $\underline{d}^{(j-1)}$. The step is repeated on the coarse signal multiple times in order to complete the wavelet transform at the desired level, since, after each step, it is possible to obtain the details of the decomposition. Moreover, since at each step the signal length it reduced by one half, iterating the procedure the outputs will finally end in a a single value vector, then no more iterations will be possible. This single values, as told before, must be equal to the average value of the original signal. The lifting schema not only provides for the second generation wavelet decomposition, but also for the signal reconstruction. The original signal can be reconstructed from the derived second generation wavelet details by means of an inversion algorithm. While the diagram in Fig. 5.1 represents the forward second generation wavelet transform, the inverse transform is trivially constructed starting by the same diagram by changing the verse of the arrows in Fig. 5.1. It is immediately clear that the inverse transform acts very similarly to the first generation wavelet inverse transform, and this fact is far from being surprising. As a matter of facts, as told at the beginning of this chapter, after the design of the $\hat{P}$ and $\hat{U}$ operators, the lifting schema can be easily represented as a first generation discrete wavelet transform, therefore is quite trivial that the reconstruction procedure follows the same steps of a first generation discrete wavelet inverse transform. Moreover it is also easy to extrapolate similarly the Fourier coefficients by using the coarse signal $\underline{c}$ and the details $\underline{d}$ as it was done with the first generation wavelets. Another advantage of the presented schema is on the preservation of its functionalities also for signals that are defined on a finite interval as well as being unevenly sampled, since, due to their attributes, the predict and update operator are not localized and can vary on the application interval. Finally, depending on adopted the meshing while defining the functional relations held by the prediction and update operator, it is

possible to generalize to higher dimensions and manifolds. Another important aspect on the adoption of the second generation wavelet lifting schema is that both the forward and inverse transforms are invertible, moreover the possibility to invert those transform does not depends on the selection of $\hat{P}$ and $\hat{U}$. This means that second generations wavelets allows us both to construct wavelets transforms with any degree of smoothness, as well as to find a solution in terms of $\hat{P}$ and $\hat{U}$ to implement both the forward and inverse wavelet transform starting from any kind of first generation wavelet mother function. This last property is extremely powerful and important, for this reason it represents the key point of this work as well as a bridge among the wavelet analysis field and the neural network design.

5.3 A neural network based approach

Here a completely new paradigm is proposed for the second generation wavelet analysis that substitutes the lifting schema with a recurrent neural network based approach for the buildup of the prediction and update operators $\hat{P}$ and $\hat{U}$, as yet published in [39]. In [142] and [141] the same authors have proven that Recurrent Neural Networks are able to exploit the intrinsic features of time series in order to predict its temporal evolutionIn the aim to design a predictive neural network, wavelet coefficients as input set give a better and efficient expression of these intrinsic features, packing in a few significant coefficients all the energy and information carried by the input signals. In [37] and [39], the authors also shown that a properly designed hybrid neuro-wavelet recurrent network that is also able to execute wavelet reconstruction and prediction of a signal. It is now understandable why it is here proposed to derive a lifting and updating construction based on a polynomial signal suppression and preservation argument. Exploiting the generalization and prediction properties of the RNNs, we realize both the operators $\hat{P}$ and $\hat{U}$, thereby tailoring the relative structures to do the lifting and predicting stages. More generally, we let the signal itself dictate the structure of the predictor. In a scale-adapted transform, we adapt the predictor in each lifting stage in order to match the signal structure at the corresponding scale.

Table 5.1: Number of neurons ($2N$) used for the different kinds of wavelets, the relative RMS, the correlation coefficient (Γ) between the experimental and the predicted data, and the convergence epochs

Kind	$2N$	RMS	Γ	Epochs
Biort. 2.8	6	< 3 %	0.9987	5000
Biort. 3.7	10	< 1 %	0.9991	8000
Biort. 3.9	10	< 1 %	0.9990	10000
Coiflet 2	12	< 1 %	0.9988	11000
Daub. 4	8	< 3 %	0.9985	6000
Daub. 6	12	< 1 %	0.9985	6000
Daub. 8	12	< 1 %	0.9988	12000
Simlet 4	12	< 1 %	0.9986	10000
Simlet 7	12	< 1 %	0.9988	9000

The basic idea is to use an RNN to adapt the predictor to the signal. This optimization produces predictors that can match both polynomial and non-polynomial signal structures. The optimization itself is a straightforward N-dimensional constrained least squares problem. The constraint is that we require the predictor to suppress $N \leqslant M$-th order polynomials, they being related to the well known vanish moments characterizing the various wavelet systems and summarized in Table 5.1. Now let x_o denote the odd-indexed data we wish to predict, and let $X_e : [X_e]_{n,k} = x_e[n-k]$ be a matrix composed of the even-indexed data used in the prediction. The vector of prediction errors then is given by

$$e = x_o - X_e p \tag{5.7}$$

The goal is to find the prediction coefficients that minimize the sum of squared prediction errors $e^\intercal e$ while satisfying the $N \leqslant M$ polynomial constraints. This we solve as

$$\min \|x_o - X_e p\|^2 \tag{5.8}$$

The optimal prediction coefficients for this constrained least squares problem can be found efficiently. We call the proposed system Wavelet Recurrent Neural Network. In fact, it is able to reconstruct a signal from wavelet coefficients, and also to predict

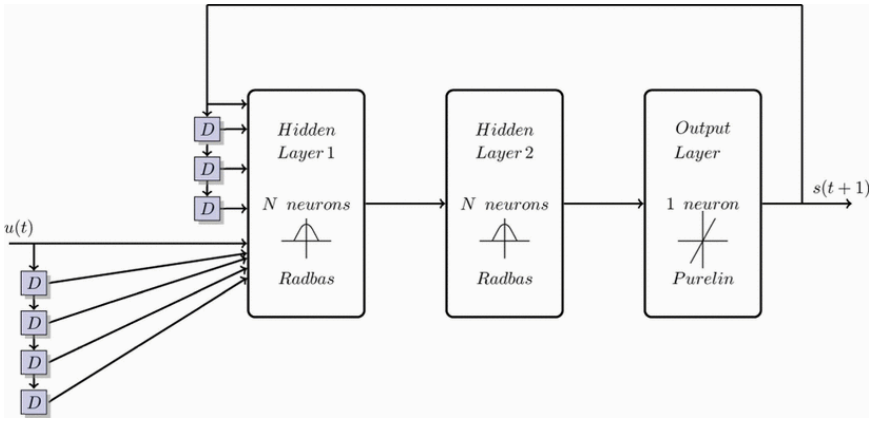

Figure 5.2: Selected RNN topology for the buildup of $\hat{P}$ and $\hat{U}$.

these wavelet coefficients aiming to reconstruct and forecast the signal. To obtain this behavior hidden layers' neurons transfer function has to simulate a wavelet function. It is not possible to implement a wavelet function itself as transfer function for a forecast-oriented time-predictive neural network, because wavelets do not verify some basic properties such as the absence of local minima and do not provide by themselves a sufficiently graded response [91]. In the existing range of possible transfer functions, only some particular classes approximate the functional form of a wavelet, such as Radial Basis Functions. Radial Basis Functions are chosen as transfer functions for the selected Wavelet Recurrent Neural networks because these particular kinds of functions well describe half of a wavelet in first approximation, even though these do not verify the properties shown by (3.32) and (3.33). Anyway, after scaling, shifting, and repetition of the chosen activation function, it is possible to obtain several mother wavelet filters. Let $f : [-1; 1] \to \mathbb{R}^+$ be the chosen transfer function; then it verifies all the properties of a wavelet function

$$\tilde{f}(x + 2k) = \begin{cases} +f(2x + 1) & x \in [-1, 0] \\ -f(2x - 1) & x \in [0, +1] \end{cases} \qquad \forall\ k \in \mathbb{Z} \tag{5.9}$$

So it is possible for the selected neural networks to simulate a wavelet by using the radial basis functions defined in the $[-1; 1]$ real domain. It is indeed possible to verify

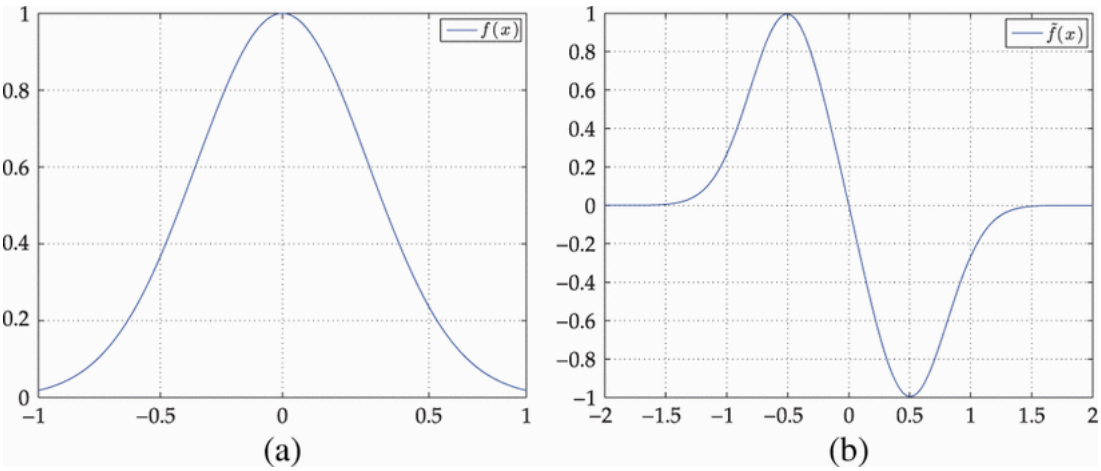

Figure 5.3: (a) RBF transfer function $f(x)$ and (b) relative wavelet function $\tilde{f}(x)$.

that

$$\int_{2h+1}^{2k+1} \tilde{f}(x)dx = 0 \quad \forall\ h < k \in \mathbb{Z} \tag{5.10}$$

It was shown that, in order to simulate a wavelet function, these chosen transfer functions have to be symmetrically periodic to emulate a wavelet. This is the reason why we choose a pair of neurons with the aim of having the same number of positive and negative layer weights in the reconstruction layer. Theoretically, if this happens, then the neuron pairs of the second layer emulate exactly a reconstruction filter. Although this is a theoretical schema, there are strong reasons for the weights in this experimental setup to have a nonzero sum because the neural network beyond to perform the inverse wavelet transform must perform also the signals prediction. Initially, several wavelet decompositions were used to obtain an input vector of wavelet coefficients, with the aim to study the capability of the selected neural network to reconstruct and predict the signal simulating different kinds of wavelet functions. Different kinds of topology and size variations were also implemented to select the better performing neural network design. The implemented wavelet decompositions permit the location of the coefficient bands related to the timescale relative to the prediction goals. By thresholding to zeros the bands unrelated to the selected timescales, the resulting coefficients and residuals carry relevant information for the predictions. These wavelet coefficients were then provided as input $(\vec{u}_i(t))$ to the system. The selected neural

network is composed of two hidden layers of 16 neurons and a single output neuron. The wavelet decomposition of the time series is given as $N \times 4$ input vectors at time t_0 with a three-step delay and a one-step output feedback to predict the output signals $s(t_0 + 1)$. To improve the generalization of the selected RNN, we have used the common method called early stopping. The considered RNN topology for the proposed novel buildup of $\hat{P}$ and $\hat{U}$ is shown in Fig. 5.2. The novelty of this approach is that the proposed WRNN does not provide the wavelet coefficients coming from the intrinsic information from other input coefficients; indeed, it is able to reconstruct them directly from the sampled signal from band- selected coefficients. From the computational point of view, the proposed approach is very efficient because of the fast convergence of the neural network, as well as robust to data errors. Finally, it represents an innovation from the methodological point of view. The simulation results obtained with the proposed forecasting method show a very low RMS error compared to those obtained by other solar radiation prediction methods based on hybrid neural networks already developed.

5.4 WRNN based approximators

Since the developed wavelet recurrent neural network presents many interesting characteristics, this novel approach to multi resolution analysis and modeling has been tested by applying it to different fields. During one of the first applications of the developed architecture the approximation capabilities of such a kind of neural network have been tested. Due to the yet encountered Cybenko-Hornik, a good neural network should also serve its scope as function approximator, therefore the application of wavelet recurrent neural networks for such a purpose permits emulate the functional trends embedded in a signal in order to let us devise some information even when a portion of the signal is missing. This properties were tested and then applied for the reconstruction of missing data in a astronomical photometric surveys [40]. Photometrical surveys are vastly used for the investigation of solar-like oscillations in order to probe the star interiors. For ground based observations the most important difficulties in properly identifying the true oscillation frequencies of the

stars are produced by the gaps in the observation time-series and the presence of atmospheric plus the intrinsic stellar granulation noise, unavoidable also in the case of space observations. Then my innovative neuro-wavelet method has been applied for the reconstruction of missing data from photometric signals i by using a composite neuro-wavelet reconstruction system composed by two wavelet recurrent neural networks separately trained. The combination of these two neural networks obtains a "forward and backward" reconstruction. For ground-based observations the most important difficulties in properly identifying the true oscillation frequencies of the stars are produced by the gaps in the observation time-series and the presence of atmospheric plus the intrinsic stellar granulation noise, the latter unavoidable also in the case of space observations. The gaps are caused by the alternation of day and night and casual interruptions of data flow due to bad weather conditions; the first introduces possible shifts of 11.57 μHz n the identified frequencies and the second spurious frequencies. The noise can produce peaks whose amplitude is even larger than the real stellar frequencies. All the mentioned disturbs make the identification of stellar oscillations uncertain in several cases. The developed approach permitted to reduce the data redundancies and selectively remove stellar granulation noise so obtaining a representation that can express their intrinsic structure, while exploiting the complexity of non-linear data correlation and to perform the data prediction. Moreover the idea to implement two different networks for forward and backward reconstruction permits to minimize the error propagation. The first one is trained to predict the signal samples one step ahead in the future, while the second one is trained to predict the signal samples one step backward in the past. The combination of these two neural networks obtains a "forward and backward" reconstructor (FWBWR) as shown in Fig. 5.4. This composite ForWard and BackWard Reconstructor uses as input several time steps of the signal, in the past and in the future with respect to the gap. The data used in the experimental testing were collected by Kepler satellite with a sampling rate of about 58.7 s, as light flux measurement and corrected flux estimation with the related absolute error. The presence of huge gaps equally spaced in the time-series, as in the case of the daily gap, causes the arising of fictitious peaks in the power spectrum, which are not real frequencies of oscillation

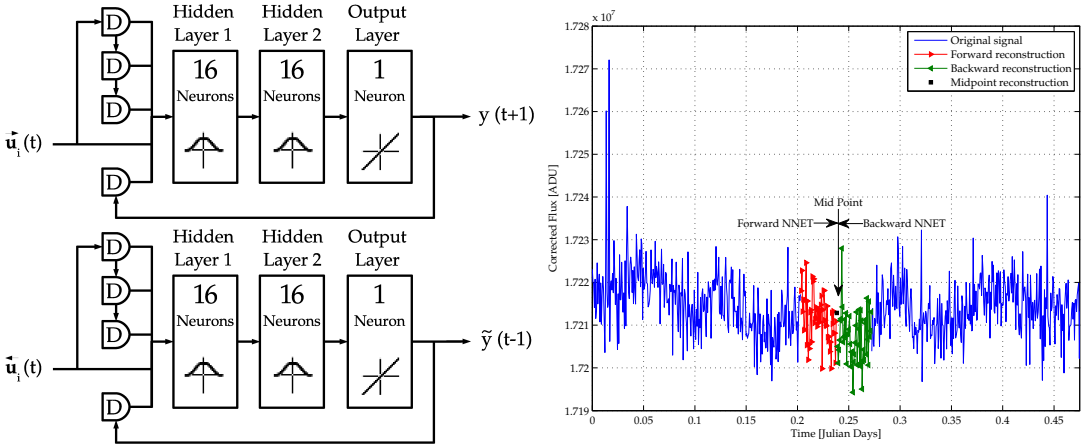

Figure 5.4: Neural networks structures (left), Forward and Backward reconstruction (right)

and that consequently affect the identification of the true p modes by hampering the true pattern of the solar-like excess of power in the power spectrum. The application of the wavelet recurrent neural network (WRNN) approach is highly suitable for deterministic dynamical behaviors, since the observation at a current time point can be modeled as a function of a certain number of preceding observations. Of course the neural network do not try to achieve credit assignment back through time but instead use the previous state as a part of the current input. Such a simple approach may be seen as a natural extension to feedforward the networks in much the same way that ARMA models generalize autoregressive models. As a matter of fact the implemented WRNNs has been able to be fed of a signal represented by its wavelet coefficients, to predict these wavelet coefficients for a future step, and, then, to reconstruct the predicted signal. To obtain this behavior some rules had to be applied during the design and implementation work. For reasons that will be cleared ahead, all the hidden layers have a pair neuron number, and, also, to permit in sequence the wavelet coefficient exploitation and the signal reconstruction, a double hidden layer is required in the proposed architecture. The trained forward and backward reconstruction system was able to reconstruct the missing data with an error greatly

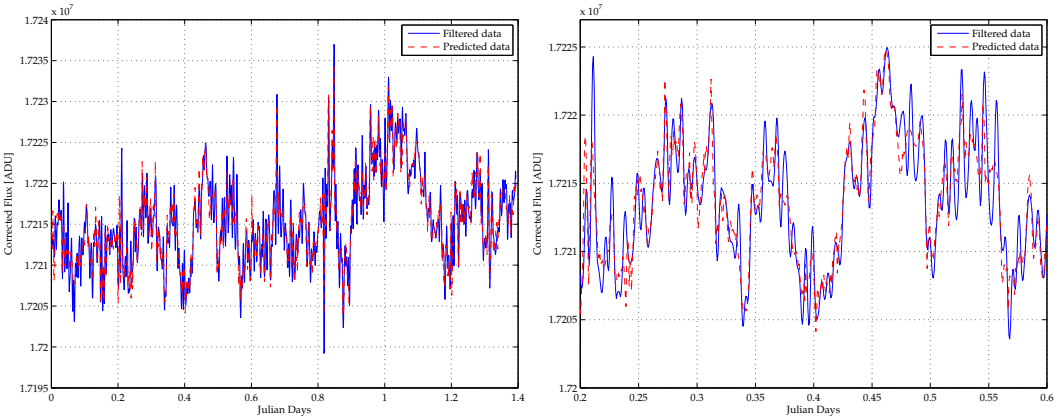

Figure 5.5: Simulation results of the forward and backward reconstruction

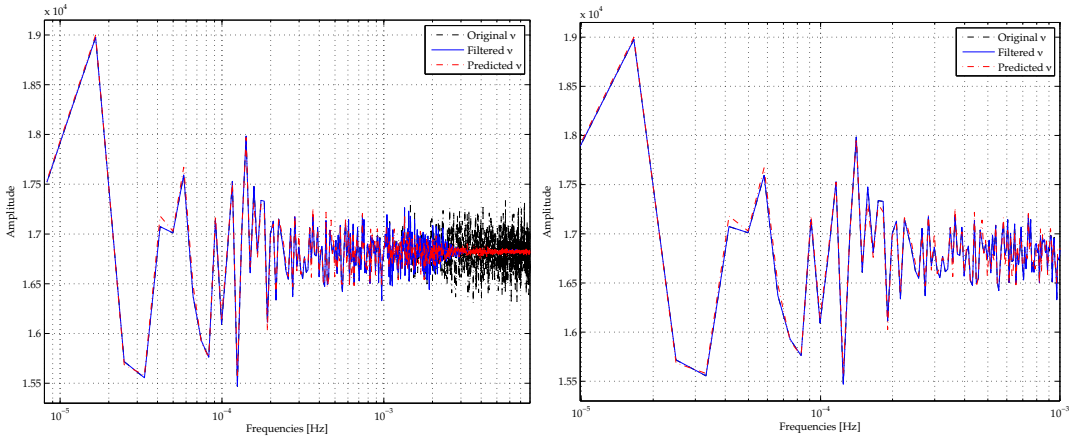

Figure 5.6: Simulation results in the frequency domain

lower than the absolute a priori measurement error. The reconstructed signal frequency spectrum matches the expected spectrum with high accuracy, as shown in Figs. 5.5 and 5.7. The novelty of the Wavelet Recurrent Neural Networks then leads to implement a new generation of tools based on recurrent neural networks with the future possibility of developments such as embedded system for data reconstruction of corrupted time-series for noise-affected survey contests.

5.5 WRNN based forecast

If the developed wavelet recurrent neural networks are able to reproduce a function or a signal in order to reconstruct some missing portion, it should be also possible to reconstruct such a portion when this is located to the end of a signal. In other words, such networks, if properly trained, are able to predict the temporal evolution in the future of signal if such a signal is expression of a law, therefore if the signal outcome from measurements of a physical phenomena. This is the case of a second testing ground for the wavelet neural network: the prediction of the temporal evolution of incoming solar radiation. Solar radiation is considered as the most important parameter in meteorology, solar conversion, and renewable energy applications, particularly for the sizing of stand-alone photovoltaic (PV) systems [69], [133], [95], [159]. The behavior of solar radiation is complex, either periodic or random, and the wavelet-transformed frequency components corresponding to various time-frequency domains of solar radiation show a similar behavior. This type of data is usually presented as a time series, whose prediction is an important scientific task. Situations where an underlying model generating the observed data is not known are especially challenging. Modeling a time series includes the stochastic prediction and the optimal prediction of a signal sample (in a minimum mean-square sense), given a finite number of past samples. In the literature, several methods to predict solar radiation have been reported, in particular statistical methods. The conventional statistical models can be considered as times-series-based models. These include auto-regressive (AR) and AR integrated moving average (ARIMA) models, Markov chains, and the Markov transitions matrix (MTM) approach. It is well known that these models are based on simplifying statistical assumptions about the measured data, which are not always true. Many researchers have attempted modeling solar radiation. The existing models established by classical approaches include, e.g., the so-called clear-day solar radiation, half-sine, ColaresPereira and Rabl, and ARIMA hour-by-hour solar irradiation models [4], [111], [126], [139], [8]. Most existing models give relatively large errors and are sometimes difficult to use widely. For this reason I've implemented the wavelet neural network architecture to find a forecasting model for the prediction of

solar radiation. The main objective of this application was to investigate the Wavelet Recurrent Neural Network architecture for modeling and prediction of the daily total solar radiation. The IDRILAB laboratories of the University of Catania provided the experimental study of the electrical behavior of PV modules and strings. The facility incorporates meteorological instrumentation with a data acquisition system, and electronic loads to obtain the plot of the current versus voltage (IV curves) of the PV modules, while recision spectral pyranometers (thermopile and photodiode) are used to measure the total solar radiation. A three-cup anemometer and wind vane assembly is used to measure the wind speed and wind direction. Ambient temperature is measured using a perforated-tip type-T thermocouple sensor enclosed in a naturally ventilated multiplate radiation shield. The meteorological data are provided in wireless mode by means of a ZigBee interface in conjunction with a software platform based on the functional schema of the meteorological data station. The meteorological variables are stored in a MySQL database. The power supply of the sensor module has two voltage levels: 5 VDC for the anemometer, ambient temperature, relative humidity, and photodiode pyranometer; and ± 12 VDC for the termopile pyranometer. The processed data was gathered by a WSENS metero unit from the following sensors. The utilize data comes tom a long-term survey made from June 2007 to November 2008. During the acquisition period, the data were registered as described in the following: wind velocities were recorded using a polycarbonate cup with magnetic switch, capable of measurements in the range 3241 km/h with 5% accuracy. RH measurements were made with a capacitive polymer with digital output, providing 2% accuracy in the range 10%-90%. For the temperature measurements, we used a digital bandgap in the range -20°C to 50°C with 1° uncertainty. The solar radiation on the horizontal plane was measured by the pyrometer. The experimental data were collected at sampling intervals of 10 min and automatically stored as synchronized time series by the information infrastructure managed by the laboratory staff. The collected input dataset was at first decomposed using the wavelet decompositions reported in Table 5.1. The wavelet scale was chosen to have a dyadic expansion so that the first band could be representative of 2-day time steps. This was made for the temperature, RH, and wind speed time series. The measured horizontal plane

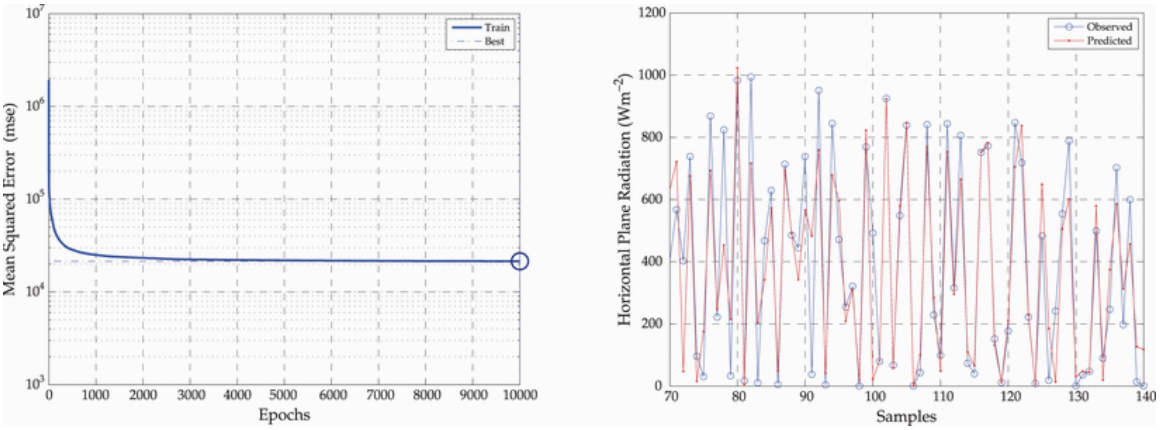

Figure 5.7: Wavelet biortogonal 2.8 decomposition set.

solar radiation ranged from 0 to 1300 Wm^2. The input pattern was composed of 12 element vectors (4 elements of 3 types of measurements). The input elements were the $a_1(t_0)$ residues, and the $d_1(t_0)$, $d_2(t_{0^-})$, and $d_2(t_{0^+})$ details. The neural networks were trained to predict 1-D output signals $s(t_1)$ at about two days in the future. In Table 5.1, we report the main features of the different networks implemented for the different kinds of wavelets relative to the simulation results depicted in Figs. 5.7–5.15. The data series are very long, therefore, in order to plot in a clear manner the obtained solar radiation prediction, only a small part is presented Figs. 5.7–5.15, and the data sample was chosen in an arbitrary manner. The quality of prediction was the same for the time data series as a whole. Here, the time refers to 10000 epochs. As shown in Table 5.1, the number of neurons doubles the number of vanishing moments plus a variable number. In fact, the synthesis filter affects this variable because of the use of this filter by the network for the prediction in the wavelet domain.

The good forecasting performance the implemented neural networks should lead to the following conclusions: the novelty of this approach is that the proposed WRNN does not provide the wavelet coefficients coming from the intrinsic information from other input coefficients; indeed, it is able to reconstruct them directly from the sampled signal from band-selected coefficients, thereby, a relatively accurate forecast of solar irradiation could be achieved due to its robustness to data errors. In facts the simulation results show a very low RMS error compared to those obtained by other

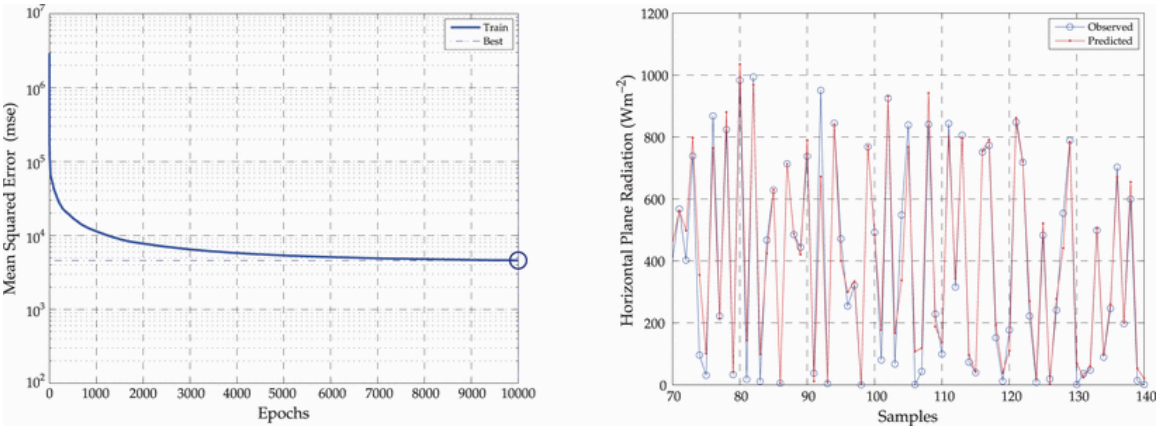

Figure 5.8: Wavelet biortogonal 3.7 decomposition set.

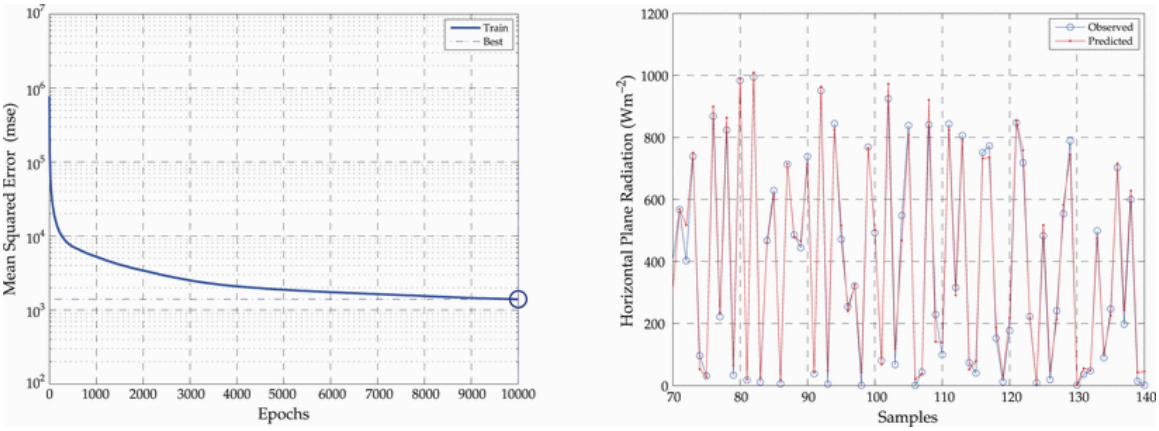

Figure 5.9: Wavelet biortogonal 3.9 decomposition set.

solar radiation prediction methods based on hybrid neural networks as in [37].

5.6 WRNN based predictors

Due to the low error and the high performance shown by the WRNN technology when predicting non periodical signal, such as the shown example of solar radiation time series, it comes naturally possible also to implement this architecture for the prediction of the connection requests to online services. In the following an example will be given regarding the application of WRNN based predictors to the field of online services

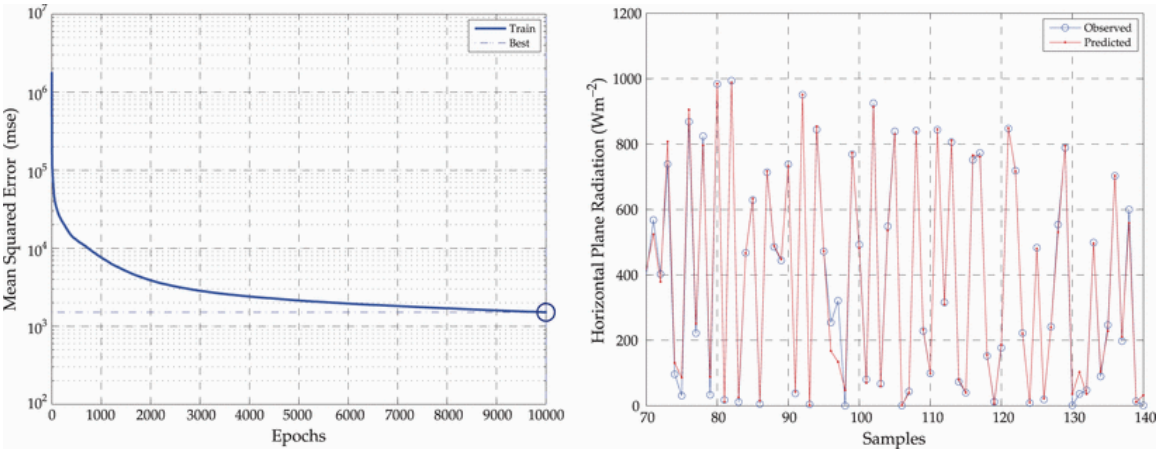

Figure 5.10: Wavelet coiflet 2 decomposition set.

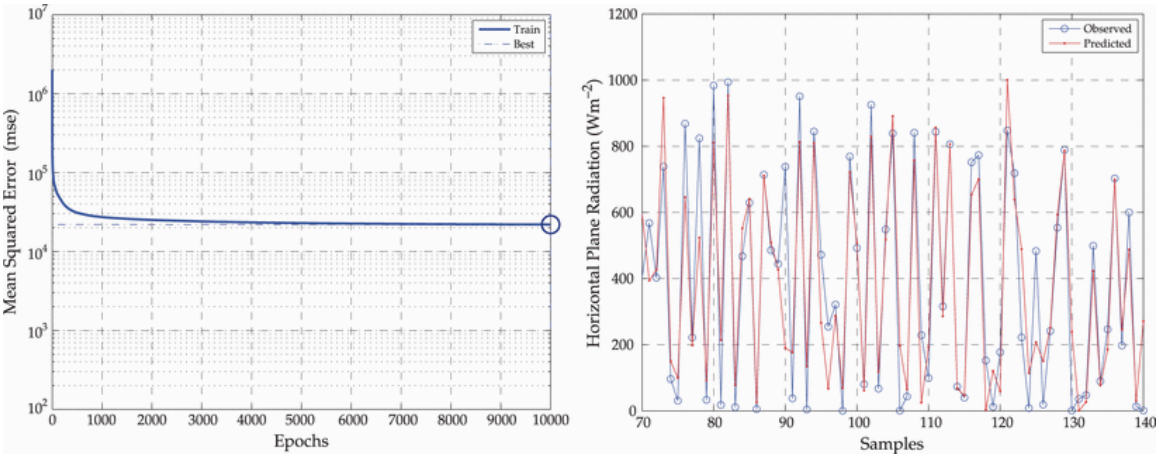

Figure 5.11: Wavelet Daubechies 4 decomposition set.

ad mean to predict resource availability and preserve the related quality of service. For distributed systems to properly react to peaks of requests, their adaptation activities would benefit from the estimation of the amount of requests. The developed WRNNs permitted us to propose a solution to produce a short-term forecast based on data characterising user behaviour of online services. Thanks to this kind of predictions, advance resource provision can be performed for the duration of a request peak and for just the right amount of resources, hence avoiding over-provisioning and associated costs. Moreover, reliable provision lets users enjoy a level of availability of services unaffected by load variations. Artificial neural networks have been used

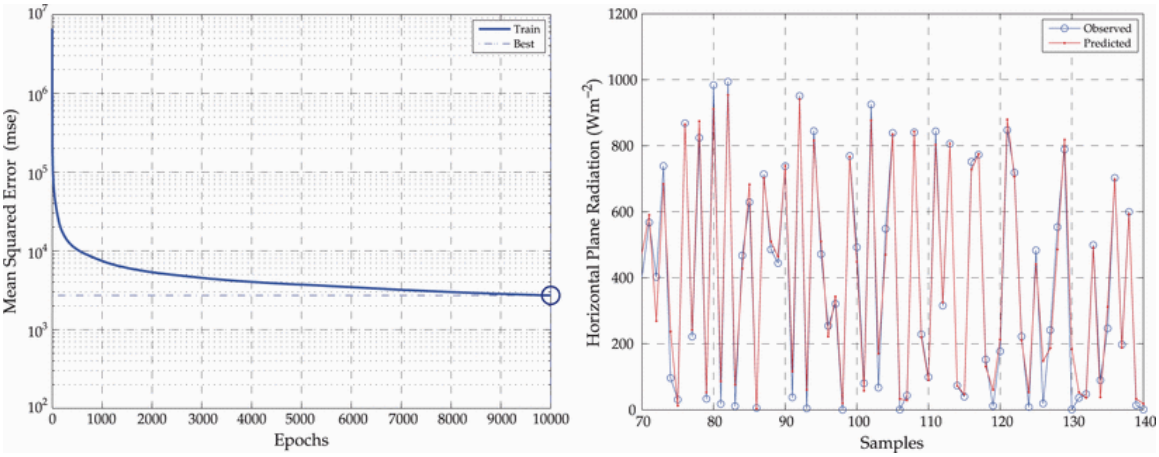

Figure 5.12: Wavelet Daubechies 6 decomposition set.

in several ways to provide an accurate model of the QoS evolution over time. Linear regression models have been applied with the support of neural networks in [102], however without some proper mechanisms, such as time delays or feedback, it is still not possible to dynamically follow the evolution of the extended time series. Machine learning approaches have also been used [173], however such approaches were not designed for on-the-fly adaptation, and are unable to give advantages with respect to user perceived responsiveness [180]. For the above approaches in which the amount of connection requests is unknown, to avoid overloading the server-side, only load balancing and admission control policies have been used. Still when the amount of requests overcomes the available resources, service usability worsening or denial of service cannot be avoided. On the other hand, when more resources can be dynamically allocated, since the amount of the required resources is unknown in advance, it often results in over-provisioning, with negative effects on management and related cost. In our experiments, the proposed WRNN has been used to analyse data for the Page view statistics Wikimedia(TM) project, produced by Domas Mituzas and released under Creative Common License. The estimated result is fundamental for a management service that performs resource preallocation on demand. The precision of our estimates allows just the right amount of resources to be used. The proposed WRNN has two hidden layers with RBF transfer function. The the initial dataset

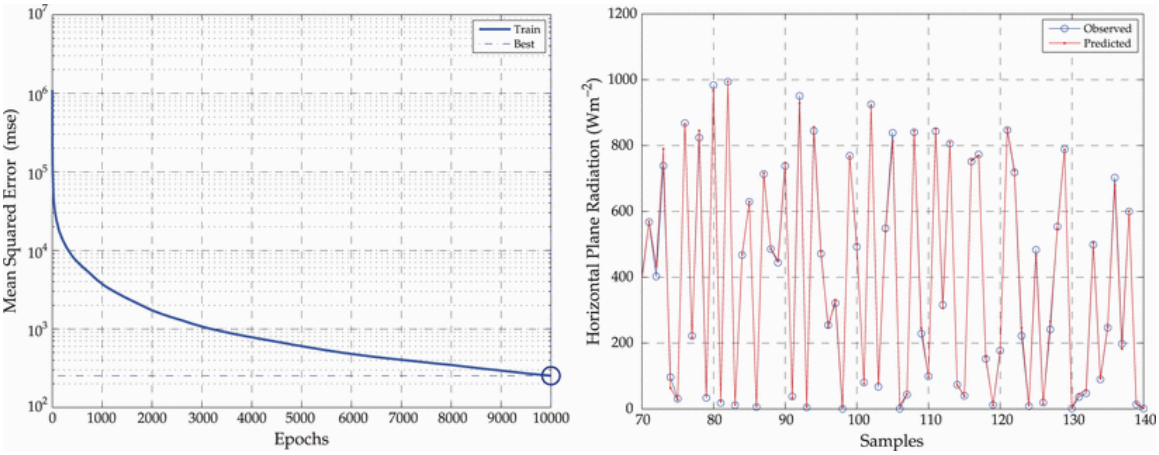

Figure 5.13: Wavelet Daubechies 8 decomposition set.

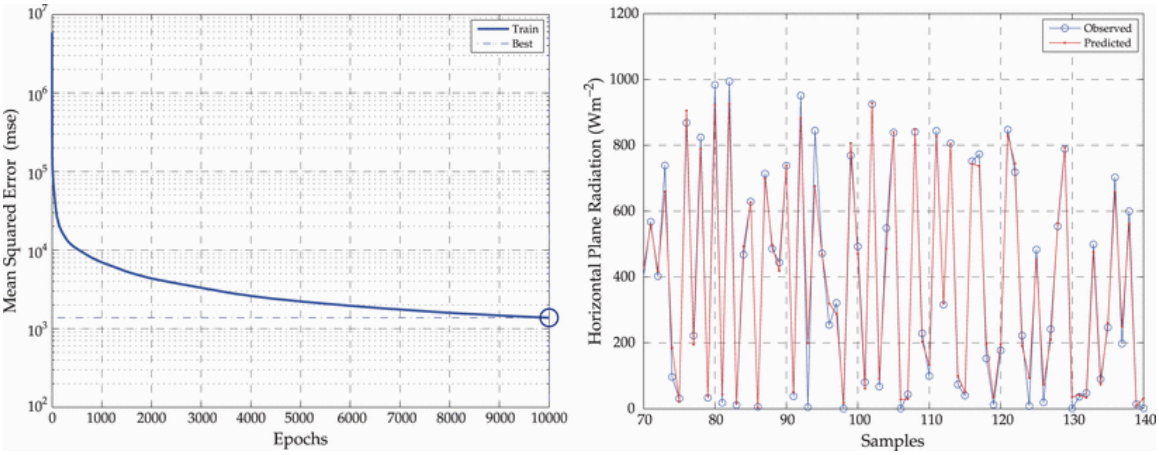

Figure 5.14: Wavelet Simlet 4 decomposition set.

was a time series representing access requests coming from users. Each row of this dataset is given as input value to the M input neurons of the proposed WRNN. The properties of this network make it possible, starting from an input at a time step τ_n, to predict the effective number of access requests at a time step For this usage, a 4-level wavelet decomposition has been selected that properly characterises data under analysis. Therefore, the devised WRNN uses a 5 neuron input layer (one for each level detail coefficient d_i and one for the residual a_5). Inputs are given to the WRNN in the following form:

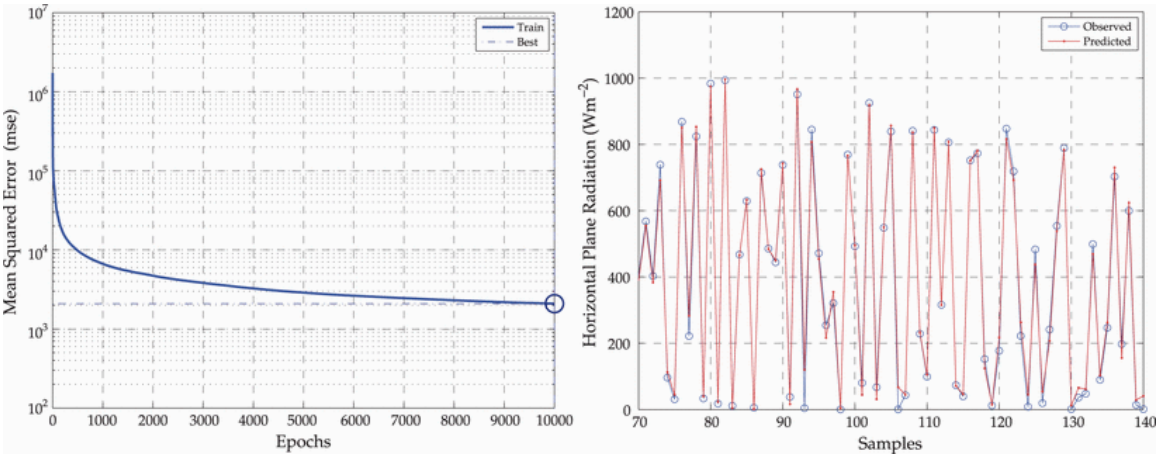

Figure 5.15: Wavelet Simlet 7 decomposition set.

- The wavelet decomposition of the time series $\mathbf{u}(\tau_n)$ for time step τ_n
- The previous delayed decompositions $\mathbf{u}(\tau_{n-1})$ and $\mathbf{u}(\tau_{n-2})$
- The last four delayed outputs $x(\tau_{n+r})$ predicted by the WRNN

As said before for the case study proposed in this paper we have used the raw data of connection requests over time to predict the behaviour of the users of a widely used an internet service, i.e. the one provided by Wikimedia(TM). Raw data were taken from the page-view statistics Wikimedia(TM) project and released by Domas Mituzas under Creative Common License. Original data report the amount of accesses and bytes for the replies that were sampled in time-steps of one hour for each web page accessible in the project. Data were collected for the whole services offered by Wikimedia projects including Wikipedia(r), Wikidictionary(r), Wikibooks(r) and others. Data were gathered and composed by an automatic procedure, obtaining the total requests made to the wikimedia servers for each hour. Therefore, a 2-years long dataset of hourly sampled access requests has been reconstructed. Then, this dataset was decomposed by using a wavelet biorthogonal decomposition identified by the couple of numbers 3.7, which means that are implemented by using FIR filters with 7th order polynomials degree for the decomposition and 3rd order for the reconstruction. The network was trained by using a gradient descent back-propagation algorithm

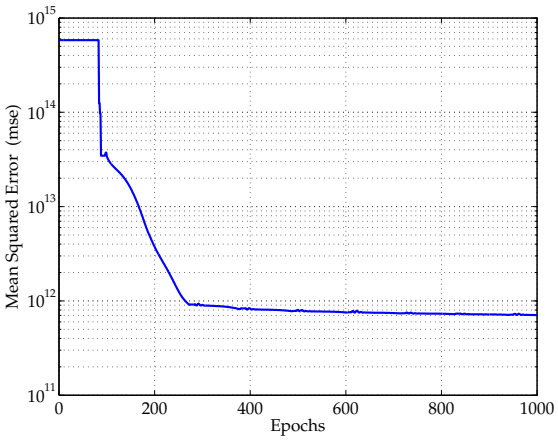

Figure 5.16: Error while training the network

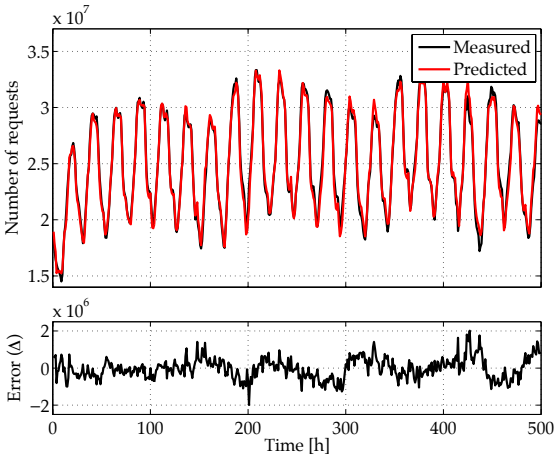

Figure 5.17: Actual time series and predicted values

with momentum led adaptive learning rate as presented in [94]. For a prediction 6 hours in advance of the time series of the amount of requests, the root mean squared error of prediction for the access requests over time was of $1.3156 \cdot 10^{04}$ requests, which means a relative error of less than 0.6 per thousands (less than six requests over ten thousands). Figure 6.4 shows the shape of the mean square error while training is being performed. Figure 6.5 shows the actual time series of the incoming requests in black, and the predicted values for the incoming requests in red for a time period of 500 hours. The actual values for the shown time period had not been given as input to the neural network for training, however have then been used to compare with the predicted values and to compute the error (see the bottom part of Figure 6.5). A smaller period of time has been shown in Figure 5.18 to highlight the differences of actual and predicted values. As can be seen, the implemented wavelet recurrent neural networks manage to closely predict even relatively small variations of the trends. The output of the neural network was then given to a resource management service to perform allocation requests in terms of needed bandwidth and virtual machines [80].

We can then affirm that the developed ad-hoc WRNN based architecture has been able to predict the amount of incoming requests performed by users when accessing a website. Moreover the performed experiments have proven that the provided ensemble is very effective for the desired prediction, since the computed error can

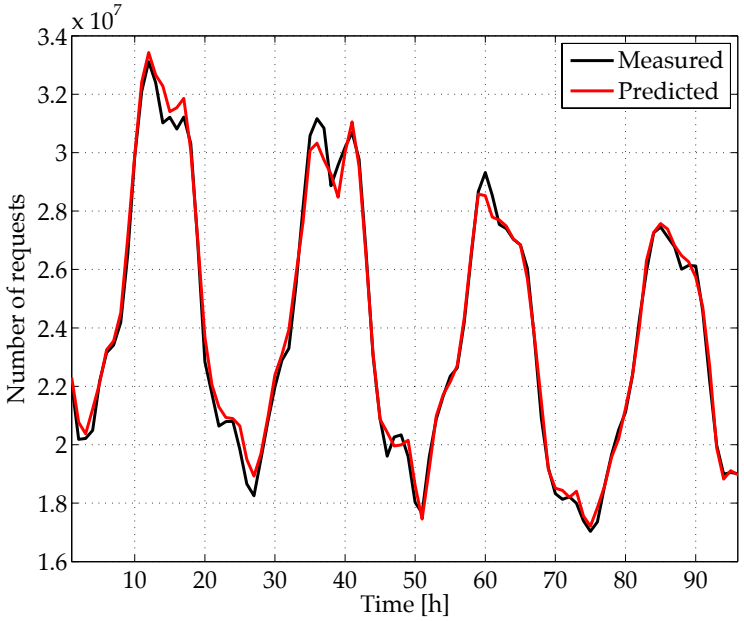

Figure 5.18: Top of the curves for actual time series and predicted values, for an arbitrary chosen time-interval

be considered negligible. Estimates can be fundamental for a resource management component, on a server side of an internet based system, since they make it possible to acquire just the right amount of resource (e.g. from a cloud). Then, in turn it is possible to avoid an unnecessary cost and waste of resources, whilst keeping the level of QoS as desired and unaffected by variations of requests.

5.7 A P2P model for WRNNs

The features shown by the WRNN architecture also permits us to implement it as a modeling tool for various purposes. In the following example the WRNNs has been used to model the availability of resources on BitTorrent also by using a file fragments diffusion model. As it is well known BitTorrent splits into fragments the files that are shared on a P2P network and then spreads file fragments by giving the highest priority to the rarest fragment. In [149] we propose a mathematical model for regulating fragment diffusion that factors in peer distances, as communication

delays, and the fragment availability at future time points, which in our approach is estimated by means of a neural network modelling the behaviour of peers. The ensemble comprising the proposed mathematical model and neural network provides a solution for choosing the file fragments that have to be spread first, in order to ensure their continuous availability, taking into account that some peers will disconnect. In peer to peer (P2P) systems using BitTorrent, a shared file will be split into fragments and the least available file fragments are automatically chosen to be sent first to users requesting the file [49]. Fragments availability is measured by the number of peers storing a file fragment at a given moment, and periodically computed by a *tracker* server storing peer ids, fragments held, and files requested [48]. For computing the scattering priority for fragments, data synchronisation towards the tracker is paramount, at best fragment availability has been freshly updated, however peers can leave the system anytime hence changing file fragment availability, possibly before the rarest fragments have been spread [106]. This occurs so frequently that such a fundamental BitTorrent mechanism may become ineffective, and as a result some fragments can quickly become unavailable. Moreover, the mechanism choosing fragments to spread is unaware of communication latencies among peers, as a consequence a fragment spreading occurs sooner on peers nearby the ones holding the fragment to be spread and the furthest peers could disconnect before receiving the whole fragment. The proposed model for spreading file fragments that considers: (i) latencies among peers, (ii) a time-dependent priority for a fragment to spread, and (iii) the behaviour of peers for estimating their future availability. We take into account the fact that more time is needed to have a replica on the farthest peer ready to be served to other peers, when compared to a nearer peer. Moreover, priority of fragments to spread will be computed again over time, as their availability changes. The variation of priority is regulated in our model in such a way to maximise the availability of fragments over time. To determine the dynamic of fragment spreading we use a *diffusion model* developed by analogy to a diffusion model on a porous medium. Moreover, we enhanced our mathematical model by using the results of an appropriate *WRNN predictor* This neural predictor aims at estimating the status evolution of the BitTorrent system, hence overcoming the sparse updates between peers

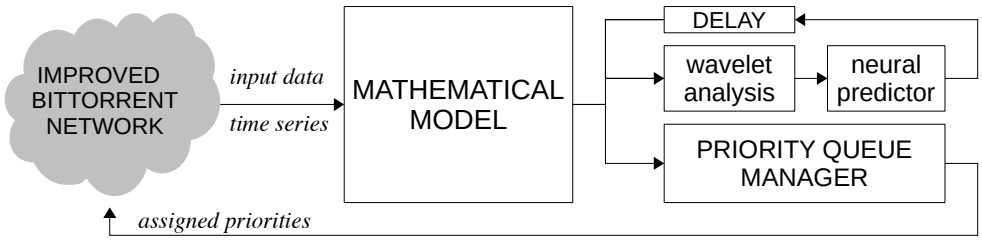

Figure 5.19: An overview of the ensemble of components for the proposed solution

and the tracker. Results provided by the neural network are fed to the above said mathematical model computing the fragments to be spread. As a result BitTorrent clients could take early actions in order to facilitate the diffusion of file fragments, in order to cope with the evolving fragments availability. Figure 5.19 shows an overall view of the proposed main components and their interactions. In order to develop our diffusion model for BitTorrent based on a physical porous medium, some conventions must be chosen and some extrapolations are needed. We first describe a continuum system using a continuum metric, however later on we will single out a few interesting discrete points of the continuum. Due to the analogy we make between a physical system and BitTorrent, we use a distance metric (named δ), which will be defined as the network *latency* among nodes, i.e. the hosts on a network holding *peers*, playing as *seeds* (peers providing fragments) or *leeches* (peers downloading fragments).For the nodes we use notations n^i or n^i_α: the first indicates a generic i-esime node on the BitTorrent network, the second indicates the α-esime node as seen from the i-esime node. Of course, n^i_α and n^j_α could be different nodes when $i \neq j$. Double indexing is needed since when we use something like δ^{ij}, it will represent the distance of the j-esime node as measured by the i-esime node. Moreover, let us express P^{ij}_k as the *probability of diffusion* for the k-esime file fragment from the i-esime node to the j-esime node. Finally, we distinguish between time and time steps: the first will be used for a continuum measure of temporal intervals and will be expressed by the latin letter t, the second will indicate time steps (e.g. the steps of an iterative cycle) and we will use for it the greek letter τ. Therefore, while $\delta^{ij}(t)$ will represent the continuous evolution during time t of the network latency δ, which measures the distance from the i-esime node to the j-esime node, the notation $\delta^{ij}(\tau)$ represents the same

measure at the τ-esime step, i.e. the time taken by a ping from the i-esime node to the j-esime node, only for the specific time step τ. Finally, we will suppose that each node has the fragment z_k of a file z and is interested in sharing or obtaining other portions of the same file, hence we will compute the probability-like function that expresses how easily the k-esime shared fragment is being copied from the i-esime node to the j-esime node at a certain step τ and we will call it $P_k^{ij}(\tau)$. Eventually, we are interested in an analytical computation for the urgency to share a fragment z_k from n^i to n^j for a time step τ, and we will call it $\chi_k^{i,j}(\tau)$. In the following sections we will distinguish between a measured value and a value predicted by a neural network using a tilde for predicted values as in $\tilde{x}$. In our work we compare the spreading of file fragments for a shared file to the diffusion of mass through a porous means. To embrace this view, it is mandatory to develop some mathematical tools, which are explained in the following. Users in a P2P BitTorrent network can be represented as points spread on a unidimensional space where a distance metric is given by the corresponding network communication latency. Therefore, for each node $n^i \in N$, set of the nodes, it is possible to define a function

$$\delta : N \times N \to \mathbb{R} \;\; / \;\; \delta(n^i, n^j) = \delta^{ij} \quad \forall \; n^i, n^j \in N \tag{5.11}$$

where δ^{ij} is the amount of time taken to bring a small amount of data (e.g. as for a ping) from n^i to n^j. By using the given definition of distance, for each node n^i, it is possible to obtain an ordered list Ω^i so that

$$\Omega^i = \left\{ n_\alpha^i \in N \right\}_{\alpha=0}^{|N|} : \delta(n^i, n_\alpha^i) \leqslant \delta(n^i, n_{\alpha+1}^i) \tag{5.12}$$

In such a way, the first item of the list will be $n_0^i = n^i$ and the following items will be ordered according to their network latency as measured by n^i. Using this complete ordering of peers, it is possible to introduce the concept of content permeability and diffusion. The adopted mathematical model will be defined in a continuous set by means of a variable δ indicating the distance between two points. In order to represent the BitTorrent network we need to associate one point to one *peer* (or *node*) of the network and to obtain such a map we implement a discrete interpretation

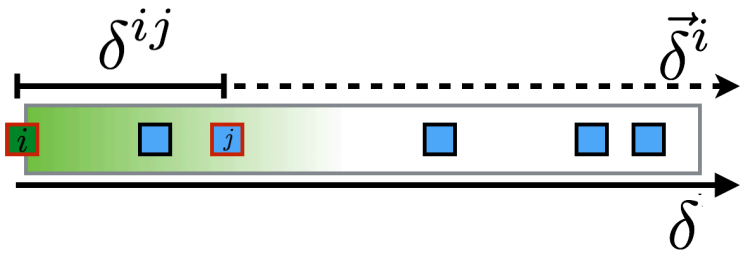

Figure 5.20: The adopted discretisation for the δ variable.

of this mathematical model. Therefore, while the following model will be developed as a continuous model, then we will make use only of several discrete points, each mapping the nodes of the network, and the model allows us to obtain their distance as a δ^{ij}. Therefore, for each node $n_j \in N$ it exists a point j in our discrete set so that it will be possible to define a *discrete distance* $\delta^{ij} \quad \forall\ n_i, n_j \in N$, while the points of the continuous model lacking a correspondent real node of the network will be ignored. The motivation for having a continuous model to start with is evident when considering how users share files on a P2P system: each file consists of several fragments, then sharing fragments can be seen as a diffusion phenomenon. For this reason we model fragment spreading in terms of the Fick's diffusion law, which is described in the following. Fick's second law is commonly used in physics and chemistry to describe the change of concentration per unit time of some element diffusing into another. Using both the First and Second Fick's laws the diffusion of a content into a mean is given as the solution of the vectorial differential equation

$$\frac{\partial \Phi}{\partial t} = \nabla \cdot (D \nabla \Phi) \tag{5.13}$$

where Φ is the concentration, t the time and D the permeability to the content. Since this is a separable equation and we make use of a 1–dimensional metric based on the distance δ, and assuming D as constant among the nodes, equation (5.13) can be written as a scalar differential equation

$$\frac{\partial \Phi}{\partial t} = D \frac{\partial^2 \Phi}{\partial \delta^2} \tag{5.14}$$

The partial differential equation (5.14), once imposed the initial and boundaries conditions, admits at least a solution known as the Green's function, which describes how a single point of probability density (in this case initially at $\delta = 0$) evolves in time and space. Thus the evolution of the system from any initial condition can be found simply by adding up the right amount of probability density at the right points in space, given by

$$G(\delta, t) = \frac{1}{2\pi} \int e^{-D\xi^2 t} e^{-i\xi^2 \delta} d\xi \tag{5.15}$$

It suffices to find a particular normalised solution, so that

$$\int G(\delta, t) d\delta = 1 \tag{5.16}$$

In order to find an appropriate solution for the problem of fragment spreading through the BitTorrent network, it is possible to apply the infinite-source diffusion boundary conditions and initial conditions. The resulting particular solution can then be written as

$$G(\delta, t) = \frac{1}{\sqrt{4\pi D t}} \, e^{-\frac{\delta^2}{4Dt}} \tag{5.17}$$

The found Green's Function permits us to study the diffusion dynamics of a single content and, as a matter of facts, it can be rewritten as a solution of the equation (5.14) in the form:

$$\Phi(\delta, t) = \Phi_0 \Gamma\Big(\frac{\delta}{\sqrt{4Dt}}\Big) \quad , \qquad \Phi_0 = \frac{1}{\sqrt{4\pi D t}} \tag{5.18}$$

where Γ is the complementary gaussian error function

$$\Gamma(x) = 1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-\xi^2} d\xi \quad \forall \; x \in \mathbb{R}^+ \tag{5.19}$$

The equation (5.19) can be computed as successive iterations from a Taylor series:

$$\Gamma(x) = 1 - \frac{2}{\sqrt{\pi}} \sum_{j=0}^{\infty} \frac{x}{2j+1} \prod_{k=1}^{j} \frac{-x^2}{k} \quad \forall \; x \in \mathbb{R}^+ \tag{5.20}$$

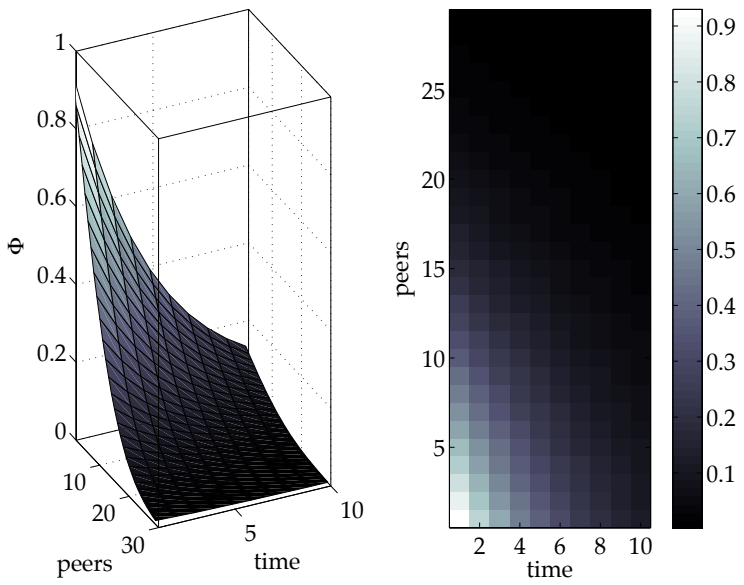

Figure 5.21: An approximation of the concentration Φ as in (5.22), computed on a set of points having distances δ^{ij}.

In [42] a pure exponential approximation for the equation (5.20) has been proposed in which, within an error of the order of 10^{-9}, $\Gamma(x)$ is calculated as

$$\Gamma(x) \approx \frac{1}{6}e^{-x^2} + \frac{1}{2}e^{-\frac{4}{3}x^2} \qquad \forall\ x \in \mathbb{R}^+ \tag{5.21}$$

Using the (5.21) in (5.18), it eventually follows

$$\begin{cases} \Phi(\delta, t) \approx \quad \Phi_0 \left[\frac{1}{6}e^{(\Phi_0\delta)^2} + \frac{1}{2}e^{-\frac{4}{3}(\Phi_0\delta)^2} \right] \\ \\ \qquad \Phi_0 = \quad (4\pi D t)^{-\frac{1}{2}} \end{cases} \tag{5.22}$$

for every node at a certain distance $\delta \in \mathbb{R}^+$ at a time $t \in \mathbb{R}^+$. Figure 5.21 shows a representation of concentration Φ, which can be computed for file fragments, as guven by equations (5.22), while varying the amount of peers and over time. In equation (5.22) the scaling factor Φ_0 is a function of the time t. On the other hand, the used formalism was developed mainly to focus on the distance δ and handling t merely as a parameter. The above mathematical formalism is valid as long as

the distances $\delta(n^i, n^j)$ remain time-invariant. The common practice considers the distance between nodes δ as time-invariant, however the actual network latencies vary (almost) continuously, with time, and a stationary Ω^i ordered set is a very unlikely approximation for the network. In our solution, we make the latency time-dependent. In turn, this makes it possible to choose a different fragment to be shared over time. For the P2P system, the equation (5.22) states that a certain file fragment z_k^i in a node n^i at a time t_0 has a probability $P_k^{ij}(t_0, t)$ to be given (or diffused) to node n^j, at a distance $\delta^{ij}(t_0)$ from n^i, within a time t, which is proportional to the $\Phi(\delta, t)$ so that

$$P_k^{ij}(t_0, t) = p_k^{ij} \left[\frac{1}{6} e^{\left(p_k^{ij}\ \delta^{ij}\right)^2} + \frac{1}{2} e^{-\frac{4}{3}\left(p_k^{ij}\ \delta^{ij}\right)^2} \right] \tag{5.23}$$

where $p_k^{ij} = p_k^{ij}(t_0, t)$, i.e. it depends on time t_0 and t and carries both the diffusion factors and the temporal dynamics. And since we are interested in a simple proportion, not a direct equation, we can also neglect the factor 4π and then write p_k^{ij} in the normalised form

$$p_k^{ij}(t_0, t) = \frac{1}{\sqrt{4\pi}} \cdot \frac{1}{\sqrt{D_k(t_0)}} \cdot \frac{1}{\sqrt{t}} \tag{5.24}$$

It is now important to have a proper redefinition of the coefficient D. Let us say that T_k is the number of users interested in file fragment z_k (whether asking or offering it), S_k is the number of seeds for the file fragment and ρ_k is the mean share ratio of the file fragment among peers (including leeches), then it is possible to consider the urge to share the resource as an osmotic pressure which, during time, varies the permeability coefficient of the network D. In order to take into account the mutable state in a P2P system, D should vary according to the amount of available nodes and file fragments. We have chosen to define

$$D_k(t_0) \triangleq \frac{T_k(t_0)}{S_k(t_0) + \left[T_k(t_0) - S_k(t_0)\right] \rho_k(t_0)} \tag{5.25}$$

then by formally substituting D with D_k in Φ_0 in equation (5.22), we obtain the analytical form of the term p_k^{ij}. Indeed, the physical nature of the adopted law works in the entire variable space, however for the problem at hand discrete-time simplifications are needed. Let us suppose that for a given discrete time step $\tau = 0$ node

n^i effectively measures the network latencies of a set of nodes $\{n^j\}$, then an ordered set Ω^i as in equation (5.12) is computed. Now, for every node n^i probability P_k^{ij} is computed for each of its own file fragment z_k and for every node n^j. This probability corresponds to a statistical *prevision* of the *possible* file fragments spreading onto other nodes. Suppose that for a while no more measures for δ have been taken, at a later discrete time step τ file fragment z_k^i will be copied to the first node to be served, which is chosen according to the minimum probability of diffusion, latencies and time since last measures were taken (see following subsection and equation (5.28)). Then, such a file fragment is reaching other nodes if the latency for such nodes is less than time t_k^i, computed as

$$t_k^i(\tau) = \sum_{\alpha_k=0}^{\tau} \delta(n^i, n_\alpha^i) \tag{5.26}$$

Index k is used in equation (5.26) to refer to file fragment z_k^i. Indeed, it should be highlighted that since nodes need and offer their own file fragments, the ordered set of nodes referred by a given node should depend on resource z_k, i.e. $\Omega_k^i = \{n_{\alpha_k}^i\}$. It is now possible to have a complete mapping of the probability of diffusion by reducing the time dependence from (t_0, t) to a single variable dependence from the discrete time-step τ. For each resource z_k as $P_k^{ij}(\tau)$ stated that it is possible to reduce $D_k(t_0, t)$ to a one-variable function $D_k(\tau)$ by assuming that at t_0 we have $\tau = 0$ and considering only the values of $D_k(t_0, t)$ when t is the execution moment of a computational step τ. Once all the $P_k^{ij}(\tau)$ have been computed, and values stored into a proper data structure, it is actually simple to determine the most urgent file fragment to share, which is the resource that has the least probability to be spread, i.e. the k for which $P_k^{ij}(\tau)$ is minimum. Furthermore, we should consider that, over time, an old measured δ differs from the actual value, hence the measure becomes less reliable. To take into account the staleness of δ values, we gradually consider less bound to δ the choice of a fragment, and this behaviour is provided by the negative exponential in equation (5.27). Given enough time, the choice will be based only on the number of available fragments. However, we consider that by that time a new measure for δ would have been taken and incorporated again into the model choosing the fragment. Generally, for nodes having the highest latencies with respect to a

given node n^i, more time will be needed to receive a fragment from the node n^i. We aim at compensating such a delay by incorporating into our model the inescapable latencies of a P2P network. Therefore, the node that will receive a fragment first will be among the farthest. For the model we have then chosen a decay law. Now it is possible to obtain a complete time-variant analytical form of the spreading of file fragments (see Figure 5.22) defined as in the following:

$$\chi_k^{ij}(\tau) = \frac{e^{-c\tau\delta^{ij}}}{P_k^{ij}(\tau)} \tag{5.27}$$

where the decay constant c can be chosen heuristically, without harming the said law, and tuned according to other parameters. If k indicates a file fragment and k^* the index of the most urgent file fragment to share, this latter is trivially found as the solution of a maximum problem so that

$$k^* : \chi_{k^*}^{ij}(\tau) = \max_k \left\{\chi_k^{ij}(\tau)\right\} \tag{5.28}$$

Figure 5.22 shows the decay of several computed χ values for different peers requiring a file fragment z_3 (3 is the fragment index). Of course, all the priorities depend on the value of the bidimensional matrix of values of P_k^{ij} (we mark that the index i does not variate within the same node n^i). Among these values, there is no need to compute elements where $j = i$ and for those elements where the node n^j is not in the queue for resource z_k. In both cases it is assumed $P_k^{ij} = 1$. Moreover, after n^i having completed to transfer z_k to the node n^j, the element of indexes (j, k) is set to 1. In a similar fashion, each peer is able to identify a possible resource to ask for in order to maximise the diffusion of rare ones instead of common ones.

5.8 WRNN-based model enhancements

Although the model proposed is based on initial conditions, essentially fragment availability, measured at a certain time. On the other hand, only when new data are received (e.g. when the tracker of the BitTorrent network sends new information on the

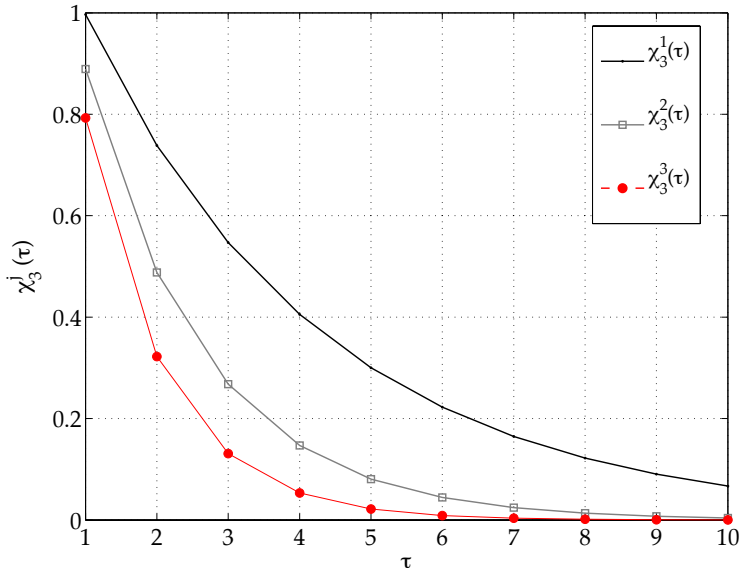

Figure 5.22: The time decay of some normalised $\chi_k^j(\tau)$ for increasing time steps τ

state of the network, the number of peers and seeds for the file fragments), then an updated result can be obtained by changing the initial conditions in our mathematical model as well. Therefore, while integrating a certain dynamics the mathematical model alone cannot predict or anticipate future network conditions by itself. In order to predict the future state of the BitTorrent network, and then suggest the appropriate priority actions as a consequence, we developed an appropriate predictor which takes advantage of several analysis methods as well as machine learning techniques in order to tamper with the timeline. This is the case of the WRNN predictors. For this purpose the initial dataset was a time series representing the past values of χ_k^{ij} in equation (5.27). For a more practical notation, we indicate such a time series as $x(\tau)$, where τ is the discrete time step of the data, sampled with a fixed ratio. We considered the time series complete (with no missing informations or data gaps) since the delivery of the series is the responsibility of the tracker and since the BitTorrent protocol requires to periodically negotiate with the tracker. On the other hand, in the BitTorrent network, such values are given on a file-related basis, in fact we have that a file is a set of fragments. Therefore, for the l-esime shared file, represented as K_l, at a time t_0 the raw values given by the BitTorrent tracker correspond to vector

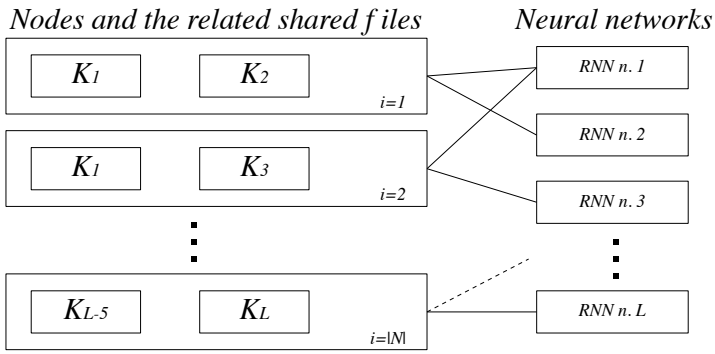

Figure 5.23: The used associative topology among neural networks and files.

ξ_1

$$\xi_1 = \begin{pmatrix} T_k(t_0) \\ S_k(t_0) \\ \rho_k(t_0) \end{pmatrix} \quad \forall\ k : z_k \in K_l \tag{5.29}$$

where $T_k(t_0)$, $S_k(t_0)$ and $\rho_k(t_0)$ are the ones used in equation (5.25). This means that at time t_0 we can compute $D_k(t_0)$ and consequently Φ from equation (5.22), as well as $\chi_k^{ij}(t_0)$ from equation (5.27). I.e. for each time step τ we indirectly obtain $\chi_k^{ij}(\tau)$ from the data given by the tracker and then using our mathematical model. As in equation (5.27), we note that i and j are the indexes of the nodes in the BitTorrent network, and k represents a file fragment. Once the time series of values χ_k^{ij} has been obtained, we want to predict the future availability of each k-esime file fragment. Therefore, the above developed RNN predictor has been trained for each shared file (not just for a file fragment, since the time series to be fed to the RNN are the same for each fragment belonging to the same file). Moreover, given the definition of file as a set of fragments it follows that

$$K_{l_1} \cap K_{l_2} = \varnothing\ \forall\ l_1 \neq l_2 \tag{5.30}$$

Therefore, for L shared files we would have L neural networks (each one associated to a file) to obtain L predictions of the parameter vectors in equation (5.29). Then, since files are shared among nodes, the results of the predictions referring a file are

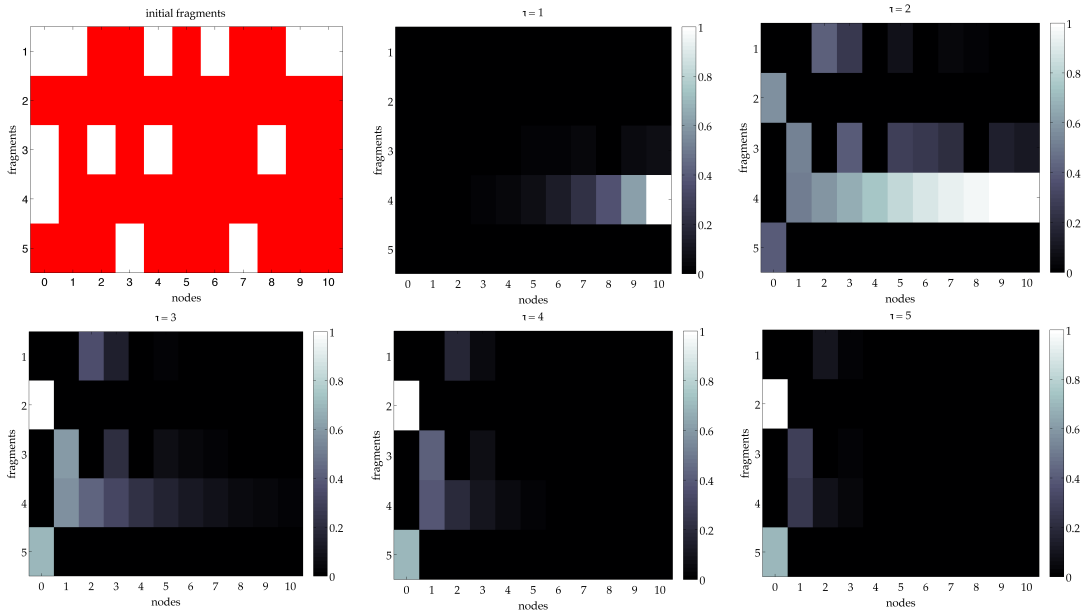

Figure 5.24: In order: the initial condition (white cells represent the fragments), the normalized $\chi_k^{ij}(\tau)$ for a certain node n^i at different time steps τ

spread to the corresponding nodes.The L trained networks have all the same topology, hence we need to store the trained weight matrices only,in case of restart. Then, each node n^i uses a subset of all predictions, i.e. the ones related to the files the node has got (see Figure 5.23). For a prediction 2 hours in advance of the time series the relative error was less than 6%.The output of the WRNN are the selected file fragment ids that have to be sent first. By considering both the predicted $\tilde{x}_k(\tau_{n+r})$ and the modeled $\chi_k(\tau_{n+r})$, it is possible, at a time step τ_n, to take counteracting actions and improve the availability estimated for a future time τ_{n+r}, hence increasing the diffusion of rare file fragments. This is achieved, in practice, by using altered values for $D_k(\tau_{n+r})$, which account for the forecast of future time steps. Such modified values are computed by our RNNs, then predicted future values for $T_k(\tau_{n+r})$, $S_k(\tau_{n+r})$ and $\rho_k(\tau_{n+r})$ are send to each node active as a peer.Each time a new file becomes shared on the P2P BitTorrent network, then a new RNN is created and trained on a server, (e.g. requested from a cloud system [31, 154]), in order to provide predictions related to peer availability of the novel set of shared fragments. Values indicating the prediction

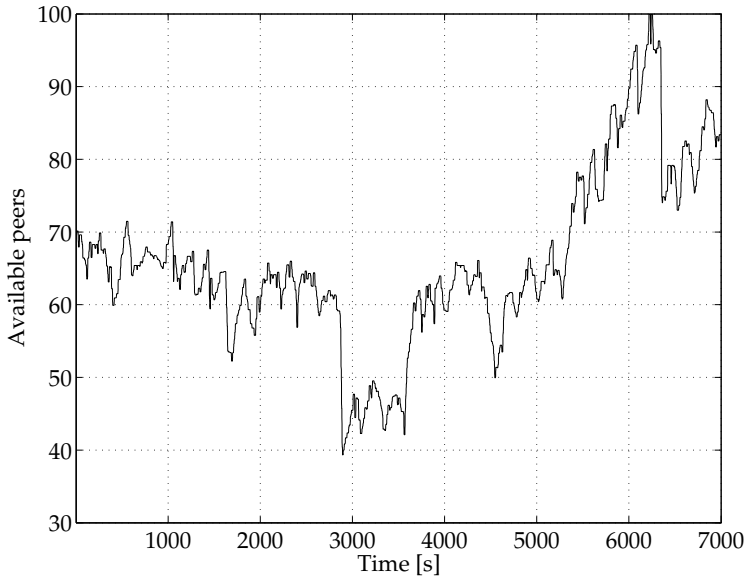

Figure 5.25: Measured node availability

are sent to the peers periodically, and allow peers to update their values of $D_k(\tau)$. The update frequency can be tuned in order to correctly match the dynamic of peers.

Figure 5.26 shows the measured distances of the available nodes measured by the first node ($i = 1$). For our experiments we used a mixture of hosts connected by the Italian research and education fast network (GARR). The simulated BitTorrent network comprised 42 nodes sharing 5 files. For the sake of clarity we also simulated a subnetwork of 10 nodes sharing 5 file fragments (see Figure 5.27). In the latter example, at the initial condition of the system four of the file fragments happen to be heterogeneously spread among peers of the P2P network, while a fifth fragment (namely z_2) is not present within the connected nodes. In the order, step after step each node selected a file fragment to require and a file fragment to send: e.g. at the time step $\tau = 1$ the node n^1 has tried to send file fragment z_4 to as many nodes as possible because of its urgency (since it is the rarest fragment) starting from n^2 (since it is the farthest node from n^0). Simultaneously, the nodes n^2, n^3, n^6, n^7, n^8 and n^9 were sending the only fragment they had at $\tau = 0$. Since both z_1 and z_3 are equally rare, then the node n^4 at $\tau = 0$ was sending these two fragments on a node distance-basis (the farthest the first). At a successive time step ($\tau = 1$) the situation seems

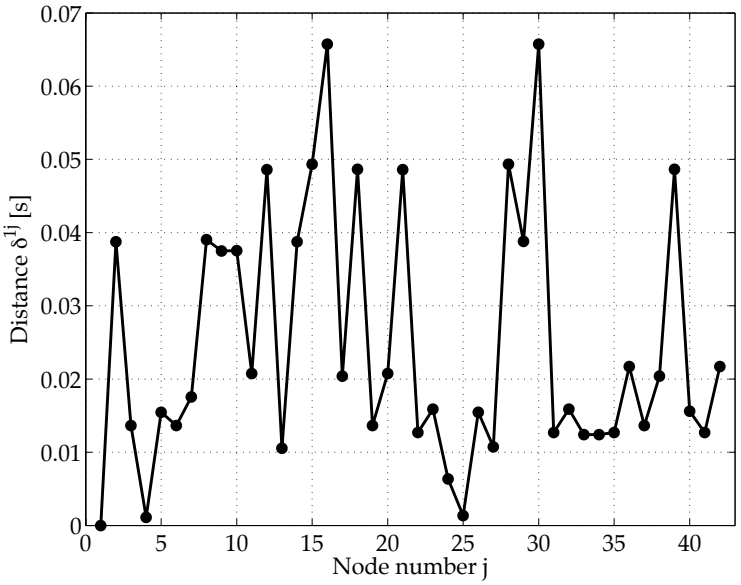

Figure 5.26: Distances measured by node n^1 with respect to all the 42 nodes available in the experiment

to change radically because of the fragments that have been just transferred among nodes. In this simulation all fragments, except z_2 because it is actually unavailable on any node, have been shared among nodes, in a very low number of time steps. It should be pointed out that from $\tau = 1$ to $\tau = 3$ some previously-rare fragments have been rapidly spread and that only later on the most common fragments will be transferred. At $\tau = 3$ the system of peers seems to reach a steady situation: all fragments have been shared, except fragment z_2, since unavailable, hence all the nodes are waiting for it. Let us now suppose that, during the time step $\tau = 4$, an eleventh node (additional to the previous network of peers) transfers z_2 to n^1, the result is then depicted in the scenario at $\tau = 5$. In this second part of the experiment, while the rarity of z_2 is not important, then only the distance of the nodes leads to the order of distribution. E.g. when $5 \leqslant \tau \leqslant 6$ node n^1 sends the file to n^9 which is the most distant node with respect to n^1. The same strategy is then adopted by other nodes receiving it until the fragment has been shared with all nodes ($\tau = 9$). The described behaviour has been determined by the model in equation (5.28). Moreover, the shown evolution does not consider the file fragments that could have been passed among the

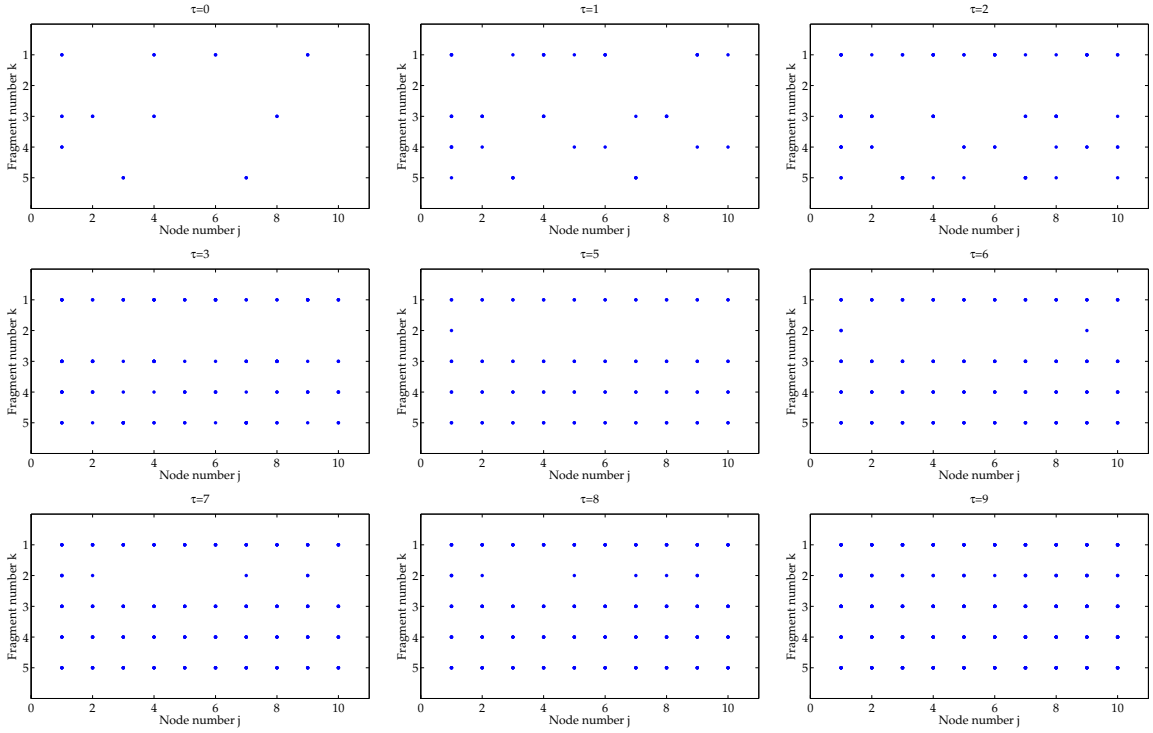

Figure 5.27: The evolution of a subnet composed of 10 nodes sharing 4 different file fragments (since z_2 is missing). At a time step $\tau = 4$ a the fifth file fragment (z_2) is injected on node n^i and then spread all over.

nodes in between two different updates, and so that for each step the value of χ for n^{10} would drop to zero (the highest values of χ are an indication of the urgency of receiving a fragment). The described model and formula allow subsequent sharing activities, after the initial time steps, to be determined, in terms of which fragments should be sent. In the long run, this law will privilege the near nodes, while on the short term, distant nodes are often the ones having higher priority. A more extensive comparison was performed by simulating both our approach and the standard BitTorrent protocol. We wanted to share a file having size 1 GB among 100 peers, therefore sharing 65536 file fragments having each size 16 KB. In our initial conditions there was only one seed (i.e. a node with all the fragments) while each of the other peers was provided with 1 file fragment (a different fragment for each peer, therefore multiple replica of the fragments where on the network). We decided to start with this setup in order

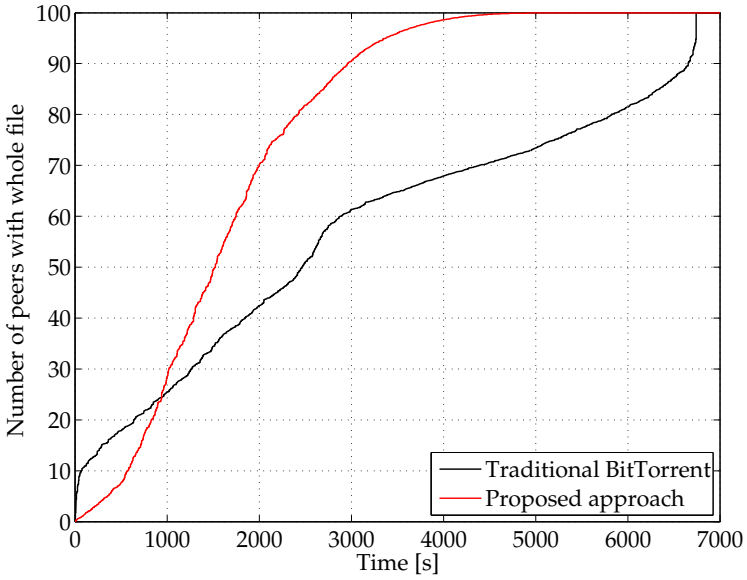

Figure 5.28: Performances of the proposed approach compared with a traditional BitTorrent network for a 1GB file shared among 100 nodes.

to simplify the comparisons of the results excluding the transient phase (i.e. when only one seed begins to share a file with peers that are not yet able to share the file). Finally, we supposed that each peer could send one fragment and receive 5 fragments at the same time. For the simulations we used network latencies and nodes availability from real data: we measured the latencies in our network (a partial amound of data is given in Figure 5.26), while we used a scaled profile of real peers availability on the traditional BitTorrent network (see Figure 5.25). The resulting comparison is shown in Figure 5.28: while our approach has a slow start (since it prefers to diffuse replicas to remote peers instead of giving them to the nearest peers), however it definitively prevails over the standard BitTorrent protocol due to the said ability to quickly adapting to the number of replicas and peers available. Therefore the proposed WRNN based models offers solutions that effectively improve the availability of fragments on a P2P BitTorrent system by adopting a mathematical model and a neural network, each properly devised for the problem at hand. The model is able to precisely describe the fragments diffusion and the urgency to share fragments, thanks to the mapping that we have proposed of a mass diffusion through a porous

means and the derived equations. The wavelet recurrent neural network approximates the availability of peers, and hence fragments, at later time points, by retaining the characteristics of the behaviour of users. This has been achieved firstly by wavelet-transforming the time series of peer availability, and secondly by feeding such results to a nonlinear autoregressive neural network, which is able to both perform predictions and apply an wavelet inverse transform. By using the estimates of future fragments availability provided by our neural network into the fragment diffusion model, we can then select the fragments that have to be quickly spread to counteract their disappearance due to some user disconnection.This choice would tap into a resource, the tracker, which is an existing component which peers have to connect to. For the computational cost, an instance of our ensemble (predicting neural network and fragment diffusion model) suffices to give accurate suggestions for a file and all its fragments, and updates to peers are given at widely spaced time intervals.

5.9 The next generation of resources

As this chapter demonstrates, it is possible to use the Wavelet Recurrent Neural Networks, eventually embedded in a properly developed mathematical model, in order to predict the availability of resource and the quality of services also when those two depend by the human choice and behaviors. The next chapter will be dedicated to crowd sourcing projects as new form of productive cycles. From the point of view of process engineering as well as from a social and scientific point of view, crowdsourcing is introducing many modification to our concept of development. Somehow the effects of this so called crowd revolution revolution does not differs from the impact that new production system had seven or eight decades ago on the automobile industries (e.g. the introduction of the Toyota Production Model, also studied by Ford to improve the productivity of his company). Speaking of production models, in a crowd sourcing based project, each single worker can be defined as a resource, therefore it would be agreeable to model the availability of such resources, as well as to understand how to predict the execution of workflows and the overall temporal evolution of the project. Following the given paragon, the application of the yet presented Wavelet Recurrent

Neural Network can help us to tamper with the timeline, on the other hand, we are not only willing to do that, but also to reach major improvements in the concept and social effects of crowdsourcing. In order to do that further actions should be taken in order to properly model human-based companies. In the following chapter we will challenge the definition itself of crowdsourcing by creating an enhanced artificial intelligence of our own in order to reach a better understanding of the Amazon's Mechanical Turk and, possibly, improve it by means of a complete revolution of the approach.

CHAPTER 6

A^2I: Artificial Artificial Intelligence

> Whatever question you can come up with, there's a person that can provide the answer.
>
> Jesse Heitler

Crowdsourcing and crowd-based services are increasingly becoming an effective solution for executing tasks and deploying end-products to users. Such a solution is an effective means for large organisations which rely on crowdsourcing to cut down the production time and costs. However, the reliability and quality of service for such a production process is at stake, since the crowd is a volatile resource. In this work we use wavelet recurrent neural networks predictors to model the worker behaviours of a crowd-based service provider. As a result we obtain a human resource availability prediction that lets us handle preemptive queue-management and resource scheduling beforehand. Then, it is possible to satisfy the target quality of service and possibly improve it over time. Generally, the capability of a multitude of people is greater than that of an individual. This is not only common sense but the simplest motivation behind crowdsourcing projects and collaborative development platforms, as well as the main force triggering the composition of workgroups and companies. Traditionally, enterprises and firms rely on a well-structured organisation and some

defined workflows that regulate the execution of several tasks, which are entrusted to their employees to produce a final product [31]. For modern production processes, relying on information technologies and especially for massless products, tasks could be outsourced, i.e. performed by workers hired on-demand, and possibly each worker could perform just one task. Such an approach mainly relies on the availability of a fluctuating workforce rather than on the specialisation and competence of known employees. In order to let "interchangeable" workers intervene in to the production process, this has to be described in terms of small and simple tasks, which can be concluded in a small amount of time by people from the crowd (back-end workers of a collaborative project). However, hiring such a cheap and highly available workforce requires some organisational effort since a trade off is needed between task size, complexity, cost and the entire workflow duration. I.e. the smaller and easier the task the most affordable and unskilled workforce can be possibly found for performing it, however the more tasks have to be completed and they have to be governed by a workflow. Although micro-task markets have great potential for rapidly collecting users' manpower at low costs, special care is needed in formulating tasks in order to harness the capabilities of the approach [109]. A notable example of tools enacting crowd-based work is Amazon's Mechanical Turk, which is emerging both as a pioneer and leader among online services, mainly due to its high usability, as well as for the potential of the paradigm itself, which enables anyone to quickly and cheaply hire a large workforce.

6.1 Amazon's Mechanical Turk

Crowdsourcing services, such as e.g. Amazon's Mechanical Turk, have just begun to provide end-users with all the major assets generally used by an actual company, such as worker pools, design streamlines, compensation systems, recruitment procedures, etc. It is then natural to anticipate a further increase on the popularity and diffusion of the crowd-based approach not only for developing some kind of product but also for providing crowd-based intelligent services. Actually, there are tasks that, though very simple, can not be executed by a machine due to its lack of a unique feature:

the human intelligence. On the other hand, the crowd-based approach fully relies on such a characteristic, since it requires humans as the source of task completion. The same humans can implement a human-based artificial intelligence (quoting the Mechanical Turk trademark it can be called an "artificial artificial intelligence") that provides the required capabilities, such as e.g. discover information that is missing from a database, perform non algorithmic functions, collect and match data, rank or aggregate results based on fuzzy criteria, etc. [77]. Such an advanced outcome from the crowd is highly agreeable since it could lead us to software products that integrate a piece of human intelligence inside: the crowd intelligence. In this sense, the crowd would then provide distributed knowledge, computation, intelligence and finalised end-user services, by taking advantage of the back-end workers and their platforms, in a similar manner to more classical distributed approaches, such as cloud computing [154]. Therefore, as any other distributed system, in order to properly react to peaks of requests the workers of the crowd should be provided with a resource reservation system. This would benefit from the estimation of the amount of incoming requests as well as the amount of available back-end crowd workers. Crowdsourcing services, such as Amazon Mechanical Turk, allow for easy distribution of small tasks to a large number of workers. Unfortunately, since manually verifying the quality of the submitted results is hard, malicious workers often take advantage of the verification difficulty and submit answers of low quality. Currently, most requesters rely on redundancy to identify the correct answers. However, redundancy is not a panacea. Massive redundancy is expensive, increasing significantly the cost of crowdsourced solutions. Therefore, we need techniques that will accurately estimate the quality of the workers, allowing for the rejection and blocking of the low-performing workers and spammers. However, existing techniques cannot separate the true (unrecoverable) error rate from the (recoverable) biases that some workers exhibit. This lack of separation leads to incorrect assessments of a worker's quality. We present algorithms that improve the existing state-of-the-art techniques, enabling the separation of bias and error. Our algorithm generates a scalar score representing the inherent quality of each worker. We illustrate how to incorporate cost-sensitive classification errors in the overall framework and how to seamlessly integrate unsupervised and supervised

techniques for inferring the quality of the workers. We present experimental results demonstrating the performance of the proposed algorithm under a variety of settings [101]. In this work we propose a solution to produce a short-term forecast of the future worker availability, based on data characterising the back-end worker behaviour. Again in order to obtain such a forecast the well explained WRNN architecture has been used.

6.2 WRNN based crowdsourcing

In the following we will demonstrate that it is possible to manage a crowdsourcing backend infrastructure by means of accurate predictions of the number of users that will be available on the back-end of a crowdsourcing project. We call this series $x(\tau)$, where τ is the discrete time step of the data, sampled with intervals of one hour. A biorthogonal wavelet decomposition of the time series has been computed to obtain the correct input set for the WRNN as required by the devised architecture. This decomposition has been achieved by applying the wavelet transform as a recursive couple of conjugate filters in such a way that the i-esime recursion $\hat{W}_i$ produces, for any time step of the series, a set of coefficients d_i and residuals a_i, and so that

$$\hat{W}_i[a_{i-1}(\tau)] = [d_i(\tau), a_i(\tau)] \qquad \forall\ i \in [1, M] \cap \mathbb{N} \tag{6.1}$$

where we intend $a_0(\tau) = x(\tau)$. The input set can then be represented as an $N \times (M+1)$ matrix of N time steps of a M level wavelet decomposition, where the τ-esime row represents the τ-esime time step as the decomposition

$$\mathbf{u}(\tau) = [d_1(\tau), d_2(\tau), \ldots, d_M(\tau), a_M(\tau)] \tag{6.2}$$

Each row of this dataset is given as input value to the M input neurons of the proposed WRNN. The properties of this network make it possible, starting from an input at a time step τ_n, to predict the effective number of workers available at a time step τ_{n+r}.

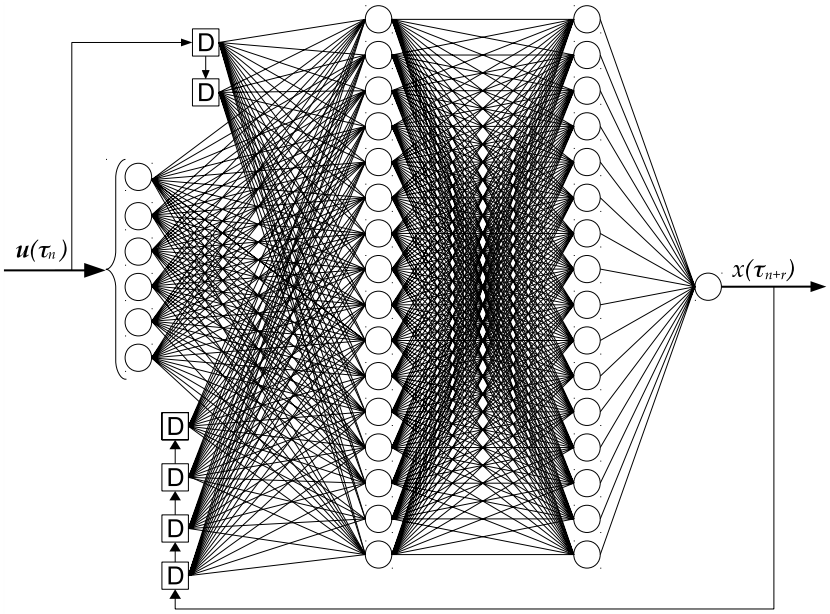

Figure 6.1: Devised neural netwok

In this way the WRNN acts like a functional

$$\hat{N}[\mathbf{u}(\tau_n)] = x(\tau_{n+r}) \tag{6.3}$$

where r is the number of time steps of forecast in the future. In the presented experiments we wanted to recreate a typical crowdsourcing model without being linked to the generality of a provider such as Amazon, on the other hand we cannot neglect the importance of the Mechanical Turk as an excellent example of crowd-based services. Moreover, we wanted to focus on a specific while still large project such as Wikipedia. Therefore, we tried to collect all the possible assumptions regarding both Wikipedia and Amazon's Mechanical Turk in order to work with a scenario eventually compatible with both. When collecting and analysing the statistics coming from the Mechanical Turk, it should be noted that the relatively large worker pool is slightly more demographically diverse than common Internet users, since participation is mainly affected by compensation rate and task length, while participants can be recruited rapidly and inexpensively [33]. On the other hand, this distributed population of thousands of anonymous workers is changing over time, shifting from a

primarily moderate-income worker population towards an increasingly international group with a significant population of young and well-educated workers mainly distributed in India [179] and other highly populated countries. This information gave us the idea to use the raw data of the Wikimedia(TM) foundation focusing on the page edit from the indian region (selected by profile nationality or ip class), since roughly who is willing to spend time improving text on Wiki pages for free is willing to be paid for some small task to perform a somewhat similar task on his/her computer. Original data report the amount of accesses and bytes for the replies that were sampled in time-steps of one hour for each web page accessible in the project. Data were collected for the whole services offered by Wikimedia projects including Wikipedia(r), Wikidictionary(r), Wikibooks(r) and others and then restricted by region. Data were gathered and composed by an automatic procedure, obtaining the total requests made to the wikimedia servers for each hour. Therefore, a 2-years long dataset of hourly sampled data has been reconstructed. Then, the said dataset was decomposed by using a wavelet biorthogonal decomposition identified by the couple of numbers 3.7, which means that such a decomposition is implemented by using FIR filters with 7th order polynomials degree for the decomposition (see also Figure 6.2 and 6.3) and 3rd order for the reconstruction. An accurate study has shown that the biorthogonal wavelet decomposition optimally approximates and denoises the time series under analysis. Such a wavelet family is in good agreement with previous optimal results, obtained by the authors, for the decomposition of other physical phenomena. In fact, such a decomposition splits a phenomenon in a superposition of mutual and concurrent predominant processes with a characteristic time-energy signature. For stochastically-driven processes and for a large category of complex systems wavelet decomposition gives a unique and compact representation of the leading features for a time-variant phenomenon. Decomposed data were then given as inputs for the wavelet recurrent neural network implemented by means of an object oriented toolbox presented in [155]. The network was trained by using a gradient descent back-propagation algorithm with momentum led adaptive learning rate as presented in [94]. For this work, a 4-level wavelet decomposition has been selected that properly characterises data under analysis. Therefore, the devised WRNN uses

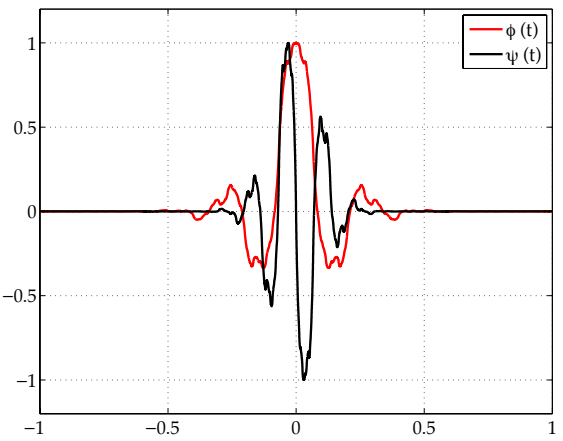

Figure 6.2: Biorthogonal wavelet 3.7 decomposition (ψ) and scaling (ϕ) functions

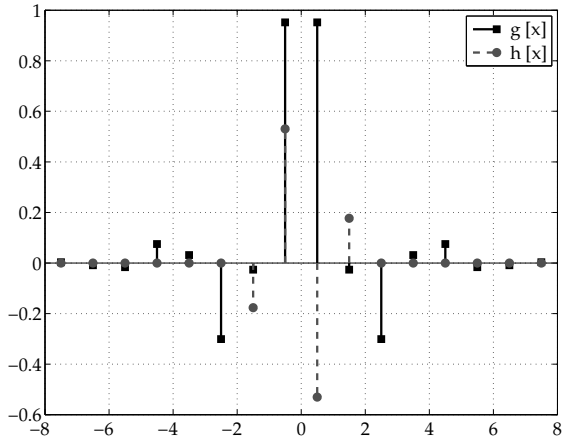

Figure 6.3: Biorthogonal wavelet 3.7 high-pass (h) and low-pass (g) filers

a 5 neuron input layer (one for each level detail coefficient d_i and one for the residual a_5). This WRNN architecture presents two hidden layers with sixteen neurons each and realises a radial basis function.nputs are given to the WRNN in the following form:

- The wavelet decomposition of the time series $\mathbf{u}(\tau_n)$ for time step τ_n
- The previous delayed decompositions $\mathbf{u}(\tau_{n-1})$ and $\mathbf{u}(\tau_{n-2})$
- The last four delayed outputs $x(\tau_{n+r})$ predicted by the WRNN

Delays and feedback are obtained by using the relative delay lines and operators. These feedback lines provide the WRNN with internal memory, hence the modelling abilities for dynamic phenomena. For a prediction 2 hours in advance of the time series of the amount of crowd workers, the root mean squared error of prediction for the access requests over time was of 0.0186 crowd workers, which means a relative error of less than 1%. Figure 6.4 shows the actual time series of the available workers in black, and the predicted values in red for a time period of 600 hours. Figure 6.5 shows a 10 hour interval of the same data to give the detail of the prediction and make it possible to see the close approximation. The actual values for the shown time period had not been given as input to the neural network for training, however have then been used to compare with the predicted values and to compute the error.

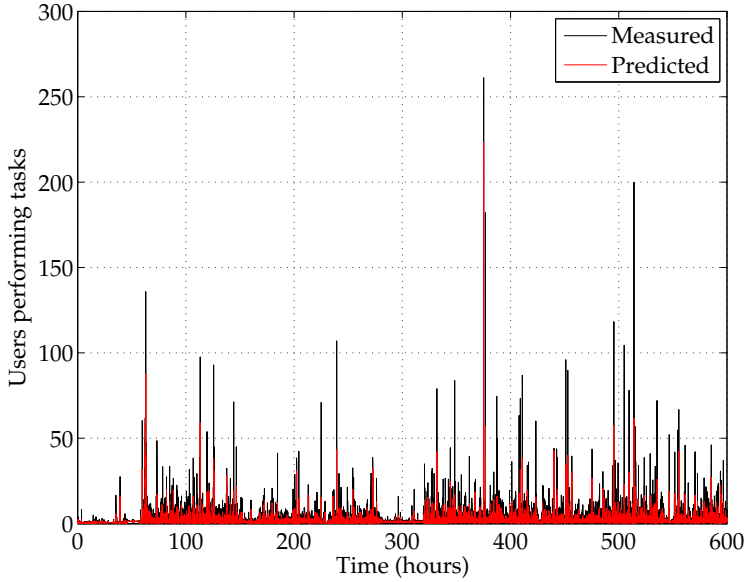

Figure 6.4: The curves for the actual time series and predicted values, for an overall time-interval of 600 hours

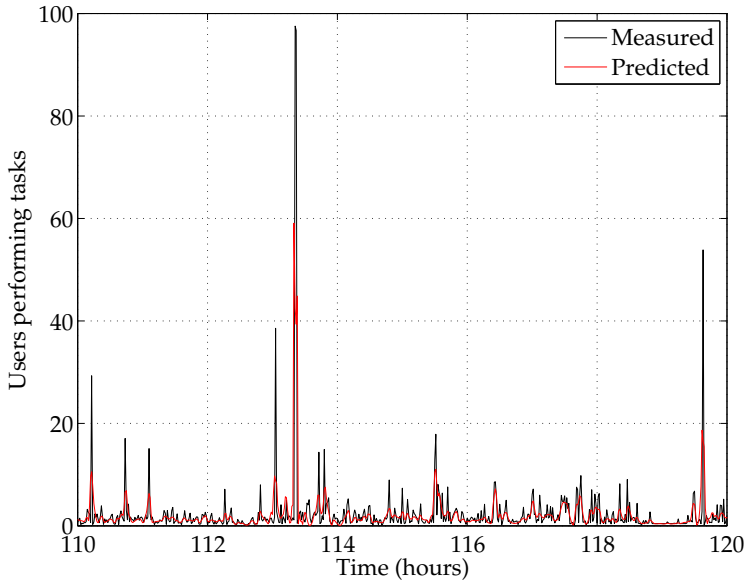

Figure 6.5: A detail of the curves for the actual time series and predicted values, for an arbitrary chosen time-interval of 10 hours

6.3 WRNN based workflow manager

This approach has provided a general architecture, based on ad-hoc WRNNs, that can predict the amount of workers that will be valuable over time for a specific crowdsourcing project. This can be used as a support when having to deploy, execute, and monitor crowd-based services, which therefore can be managed like virtual machines on a cloud platform, and related to each other as a series of mutually dependent tasks, hence as a workflow. Firstly, the past time series has been analysed by means of wavelets, which appropriately retain only the fundamental properties of the series. Secondly, the neural network has been used to embed both the ability to perform wavelet analysis and prediction of the future amount of available workers. The performed experiments have proven that the provided ensemble is very effective for the desired prediction, since the computed error can be considered negligible. Estimates can be fundamental for a crowd management component since they make it possible to acquire just the right amount of workforce. Then, in turn it is possible to avoid unnecessary costs and waste of resources, whilst keeping the level of quality of service as desired and unaffected by variations of workers availability avoiding negative effects on the production time and costs. Moreover, the obtained human resource availability prediction permits us to: (i) improve the quality of service, since other factors being equals we can determine the priority of tasks to carry out; and (ii) preserve such quality over time, by performing queue-management and resource scheduling beforehand. The proposed solution is independent from the specific service to be provided, as well as being general and therefore applicable to a wide range of different crowd-sourcing environments. Such an approach can also be used in order to schedule and manage the task executed by a crowd in relation to the availability windows of the crowd-based service workers. The proposed solution can be also embedded in a component devised specifically in order to minimise the possible delays affecting a crowd-based service, which are mainly caused by a temporary unavailability of workers or by a work distribution that could benefit of further optimisation. The latter can be easily provided thanks to the proposed approach by modeling the incoming worker availability and the completion time of their tasks in order to schedule future

work. Then, the solution here devised can be very beneficial for the management of the operations on crowd-based resources in advance, hence shunning delays, and also avoiding over-provisioning as an alternative to reduce delays, hence emerging as a cost-effective solution. As seen also before, distributed systems to properly react to peaks of requests, their adaptation activities benefits from the estimation of the amount of requests. Several means are needed, including client devices, such as smart phones, to request transportation vehicles, and a cloud-based intelligence that serves requests by governing the available resources. An enterprise providing services handled by means of workflows needs to monitor and control their execution, gather usage data, determine priorities, and properly use computing cloud-related resources. This paper proposes a software architecture that connects unaware services to components handling workflow monitoring and management concerns. Moreover, the provided components enhance dependability of services while letting developers focus only on the business logic. business enterprises organise their provided operations by means of a pre-defined workflow, i.e. a flow of execution relating activities supported by software services and regulated by an engine [71, 145]. Generally, several services connect with others, or provide data to others, according to a predefined workflow, while sharing common computing resources, available e.g. as a cloud-computing facility [12, 191]. By resorting to cloud-computing, transparent access e.g. to shared services, hardware and data is made possible, thus enabling a higher level of *availability* as well as higher performances [154]. One of the primary needs for enterprises is to handle service executions in such a way to: (i) monitor human activities, (ii) have a smooth execution on servers, and (iii) achieve a defined degree of *dependability* [13]. For the latter, when a server handles numerous requests, to counteract slowing responsiveness and enhance *availability*, services can be activated on other cloud resources [27] or by means of software agents [148]. For an enterprise, *monitoring* their workflows means gaining metrics related to business service performances, employee productivity, etc. Such metrics enable decision making for optimising how human resources provide their assistance, handling priorities, such as the allocations of tasks to human resources or processes to hosts. Services activated by means of a workflow could include the additional code that supports both dependability and

monitoring issues, however such a solution would bring a high complexity level for services, lower maintainability and increase development costs [171, 192, 146]. Moreover, when using other traditional solutions for supporting additional concerns, such as non-functional ones, services have to conform to an ad-hoc supporting framework or development model [125, 169]. This limits *modularity* and forces some components to be manually adapted. In order to support modularity, we propose a software infrastructure that seamlessly activates workflow execution, provides services with *monitoring* and enhances *dependability*. The technology empowering the said seamless integration is *aspect-orientation* (AO), defined in [107, 116]. A software *aspect* is a module that includes portions of code (comparable to methods) that are injected into an existing component according to defined rules. AO systems have been built to separate QoS-enhancing code from functional code, design pattern-related code from classes [15, 82, 84] etc. Unlike previous approaches our proposal can be applied to services, and workflows, without relying on any assumptions, is not intrusive for the underlying support (such as JVM or OS libraries), enhances modularity, and is not disrupting in terms of execution flow, development process and freedom for service developers. A typical enterprise workflow is generally formalised by means of a description, which is given to a workflow engine, e.g. JBOSS jBPM, that timely starts the several services described [71]. The workflow engine is able to send notifications about the state (e.g. just started, executing, finished) of the services on a workflow, and therefore alert humans or gather statistical data about the execution. Figure 6.6 shows an example of a simplified workflow, dubbed city planning, whereby several services are executed each corresponding to a step that can be performed after receiving a user request for a permit. A reference model for the software system assisting such steps will have one or more client applications enabling the user to submit a request or to gather replies. Hence, e.g. the step send permit request could be performed using a web browser or an ad-hoc application connecting to a service, receive possible date could be a message received as a status update on a web page or an email, etc. Services on the server side are processes running, or started, according to the indication given by the workflow steps, hence e.g. receive requests is the first step of an ad-hoc workflow, and is a process listening for incoming requests, analyse

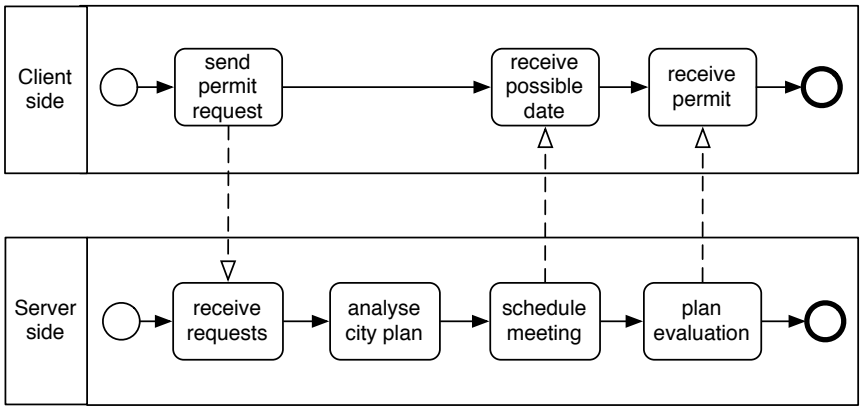

Figure 6.6: An example of workflow city planning on a distributed system

city plan is a process started as a second step of the workflow once the previous step has been performed, etc. Services within a workflow are generally of different nature, e.g. have their own data or processing requirements, hence they should be handled differently when dealing with dependability. Let us suppose that analyse city plan is a resource consuming process whose execution time has to be guaranteed, instead, another service could simply provide immutable stored documents. Then, handling requests that trigger service execution requires the provisioning of ad-hoc computing resources to ensure dependability.

6.4 Companies as humans' clouds

Cloud-based solutions provide a means to limit costs for an enterprise needing high-performance hardware resources [12]. Therefore, the server-side software components can be partially or wholly supported by cloud-based resources. In such a scenario, additional concerns can be identified, as *workflow monitoring* and *dependability enhancements*. Our contribution provides such features by inserting support into existing services. Business related services are automatically provided with monitoring, i.e. how and when such services are used. Monitoring allows typical usage behaviour of enterprise employees to be detected, thanks to their interactions with services. This makes it possible to compute metrics on productivity, workflow execution time,

and potential bottlenecks of workflows. Data gathered while monitoring services and workflows are sent to components located on cloud-based resources, such that even though large amount of data are accumulated, they can be easily processed. Therefore, our monitoring concern is different from cloud metering services that observe resource consumption and exhibit the status of several resource-based metrics. As a means to improve availability, we resort to an automatic and transparent support that detects whether a service to be executed needs to be assisted by secondary, cloud-based hosts, which execute services and provide contents. By delegating a portion of the service processing requested to the secondary host, we enhance overall response times and limit the workload on hosts. The further benefit given by the use of secondary hosts is a marked reliability increase for the whole system, for additional hosts can easily take over the workload assigned to one that incidentally incurs into faults. This makes the overall enhanced system capable of providing uninterrupted service under load peaks and faults. In order to support the features outlined above, we propose a few software components that enhance services. The connection between provided components and workflow services is accomplished by means of *aspect-oriented* programming. A software *aspect* is a component defining *pointcuts* and *advices* [107, 116]. The pointcuts trigger execution of advices when given join points, i.e. some points on the code of a service, are executed. Aspects allow *crosscutting concerns* to be dealt with in a modular way, thus making components maintainable and prone to be reused [82, 84, 83]. Figure 6.7 shows how our components operate, as seen by an external observer. The chief actors are: the deployed services, such as Web-Service1 (the middle layer), which clients can connect to; a set of cloud-servers, such as Cloud-Server1 (the rightmost layer); a set of regular clients. Services, comprising their business logic, can be exposed on the web, hence users can interact with their browser, or reside on a cloud resource. Our provided aspect Reporter connects services exposed on the web with other cloud-based services. For this, an ensemble of several supporting classes are used, i.e. mainly ReqHandler, Scheduler, NNPredictor, LocalTimingStore. In the example above (see Figure 6.6), the steps of a defined workflow are associated with indications that allow resource allocations. Table 6.1 provides the workflow services with such additional data. In our

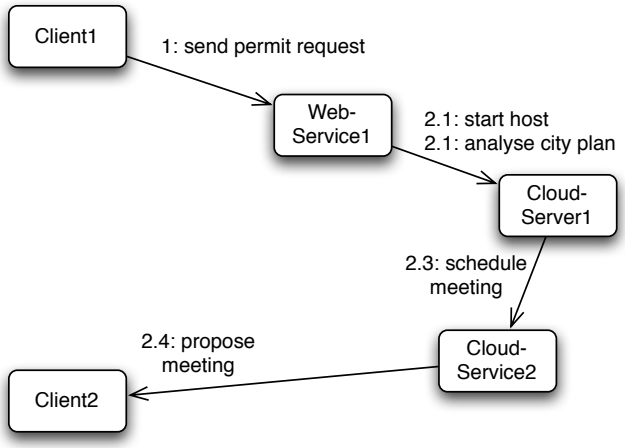

Figure 6.7: Client requests served by a web server and a cloud-server

Table 6.1: The characterisation of services for workflow city planning in order to manage of cloud resources

service name	start, end status	resource type
receive requests	on, on	no cloud
analyse city plan	off, off	dedicated, large VM
schedule meeting	standby, standby	shared, small VM
plan evaluation	off, off	shared, medium VM

solution, a component, dubbed Controller, takes as input the above data and accordingly during runtime handles the current request. Typically in our scenario, a new instance of a workflow is started by means of a request coming from a web service (see the following section), then Controller is alerted, finds the workflow that has to be executed, and starts operations on cloud resources for the following services. In the above example, after executing service receive requests on a web server, Controller receives an alert, finds the related workflow and prepares resources for the execution of service analyse city plan by starting a dedicated VM. From the second row in the above table, we can see that the service has to be started on a dedicated large VM. For each workflow, additional data have to be provided, such as e.g. the priority, the allowed deadline, the number of instances that can be concurrently executed. Moreover, for each service on the workflow, the number of its instances that can be concurrently active is also given. Aspect Reporter enhances a service responding on

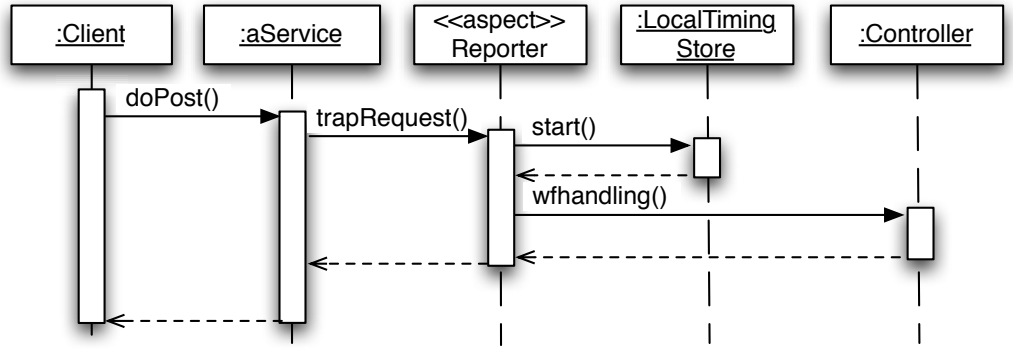

Figure 6.8: Sequence diagram showing aspect Reporter that monitors invocations to a service performed by a client class Client and connect the proper workflow related service

a known IP address, and transparently provides it with abilities to effectively handle request bursts. Such an aspect relies on other components that are located on cloud-based servers. Aspect Reporter connects with services listening HTTP requests, monitors the timing of their execution, and triggers execution of other offloading services when needed. Once Reporter has gathered workflow-related metrics, data are sent to the local class LocalTimingStore. In turn, such data will be periodically sent to service TimingStore located on a cloud server. The latter appropriately merges data coming from different users and services, or different instances of the same service. Figure 6.8 shows an UML sequence diagram for the said measuring operations, since when Reporter pointcut trapRequest() intervenes into the execution of a service. We have defined pointcuts for capturing a variety of possible implementations, because an HTTP request can be handled by a servlet, or as a EJB. Servlet implementations, which handle HTTP requests, are subclasses of HttpServlet class (available into Java package javax.servlet.http) and have to have methods doPost() or doGet(). Whereas when having services implemented as EJBs, annotation @WebService is used, or to expose only a method as a part of a web service, annotation @WebMethod is used (such annotations are defined into package javax.jws). The corresponding *pointcuts* trapRequest() and trapRequestWebServ() are shown in the listing of Figure 6.9. The first pointcut captures all the points of the program that invoke methods doPost() or doGet() implemented in a subclass of HttpServlet; whereas the second pointcut

captures all the invocations to any method of a class that has been marked with annotation @WebService. The timing of operations exposed as a web service gives a great amount of details on the workflow activities that are performed. This aspect encapsulates a concern that is crosscutting both for the components of a service, and for several services available within workflows defined by the enterprise. Nevertheless, the code of such an aspect is independent of any observed services. Compared with alerts that can be set up on the workflow engine to be notified of service execution, aspect Reporter additionally gives a means to control operations within a service, hence supports fine-grained monitoring and adaptivity. Thanks to this aspect, workflows and services can be given resources according to different policies. For the proposed solution to be effectively used, aspect Reporter needs to be deployed enterprise-wide. I.e. aspect Reporter has to be deployed on every service to be enhanced with the monitoring and dependability support. Aspect Reporter temporarily stops an incoming HTTP request and Controller determines whether the request should be handled by the local host, or by another service on a cloud resource. How to handle a request depends on the current load, request type, requesting user, and the applicable workflow. For each service, Controller holds the configuration needed, including: priority, number of allowed concurrent instances, flavour and status of the VM, etc. From such data and the actual status Controller determines whether to let the request forward, start another VM, etc. Controller is the main component assisting Reporter. Figure 6.10 shows the relevant interactions, i.e. when a request arrives on a handling web service, Reporter aspect intervenes to let the request be served according to our model. Hence, pointcut trapRequest() is activated by rules defined as in Figure 6.8 and the listing in Figure 6.9. Such an aspect then extracts data to identify the request. The activated advice checks with Controller the priority and whether the current request is allowed and can continue execution on the local host (see call to Controller.wfhandling()). The check is handled by first calling getResources() (Figure 6.10), and this finds which service and host are needed to serve the response. When the host on the local web server can not handle the request, then the list of needed resources is passed to CloudFacade class, calling its startVM() method or a proper method (such as e.g. enqueRequest(), etc.), which allow the requested service

```
public aspect Reporter {
   private final LocalTimingStore lts =
                LocalTimingStore.getInstance();

   // capture method invocations on a servlet
   pointcut trapRequest() :
                call(void HttpServlet+.doPost(..)) ||
                call(void HttpServlet+.doGet(..));

   void around() : trapRequest() {
      HttpServlet t = (HttpServlet) thisJoinPoint.getTarget();
      Object[] arg = thisJoinPoint.getArgs();
      HttpServletRequest r = (HttpServletRequest) arg[0];
      lts.start(t, r, System.nanoTime());
      if (!Controller.getInstance().wfhandling(t))
                        proceed();
      lts.end(t, r, System.nanoTime());
   }

   // capture method invocations on annotated classes
   pointcut trapRequestWebServ() :
                call(@WebService * *.*(..));

   void around() : trapRequestWebServ() {
      Object t = thisJoinPoint.getTarget();
      lts.start(t, System.nanoTime());
      if (!Controller.getInstance().wfhandling(t))
                        proceed();
      lts.end(t, System.nanoTime());
   }
}
```

Figure 6.9: Aspect Reporter that intercepts invocations on operations of classes implementing a service exposed as HTTP and records the starting and finishing time

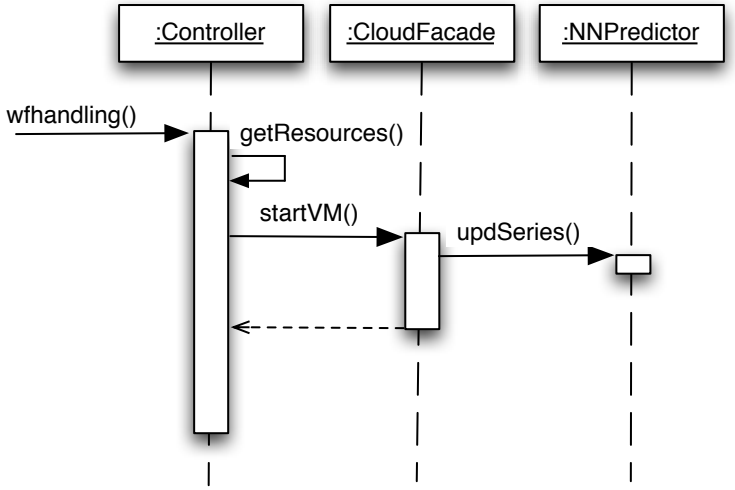

Figure 6.10: Sequence diagram showing interactions with a cloud and the resource usage predictor

to be later executed on the cloud resource. Class CloudFacade handles the details for the operation to be started on a cloud facility. Whenever a new request arrives, an hybrid neural network engine [39] is updated for the benefit of future forecasts using the updSeries() call. Moreover, NNPredictor subsystem periodically computes the load estimate as an amount of received requests to services of the enterprise and will boot up or shut down cloud-hosted servers. NNPredictor can estimate up to six hours ahead with a relative error of 0.6 per thousands [145]. When a subsequent service on the same workflow instance has to be executed, the needed cloud resorces have been started and made ready to serve by NNPredictor. Thus, Controller collects the addresses of which host will be assisting the current session in order to let the following service execute. Each client is identified by a set of characteristics, such as IP address, user id, priority, etc. and may pertain to different categories, such as response time, connections per second, etc. Accordingly, Controller determines whether to accept the current request, and in case it is allowed compute a priority for all the services of the related workflow. The priority is based on the identity of the requester and the workflow whereby the current request belongs, and in turn according to such a priority cloud resources (e.g. VM flavours) are selected. As far as the load balancing part is concerned two important benefits are achieved: (i) the load of a request in a

workflow instance is distributed according to the several hosts involved in the execution, and their state, (ii) initial requests can be accepted, put on hold or rejected, by our Controller.

6.5 A lesson from Quantum Electrodynamics

Quantum electrodynamics inspired us an ad-hoc description for a computational cloud in order to follow the dynamical state of the system during time. If we define a computational cost for a task (such a cost could refer to memory, cpu time, complexity or other software measurements), then it is possible to use the defined cost for an appropriate fitness function. Since we are interested in finding the optimum solution for the allocation of resources on a cloud, it would be equivalent to compute the local minima of such a fitness function, once taken into account all the constraints of the case. Such a problem is similar to a typical partitioning problem for a fermion particle gas, therefore solvable with a few basic tools from quantum electrodynamics. In our description, then, the said computational cost will have the same meaning of the energy of a fermion. Therefore, the resources provided by a VM can be seen as the local minima of a potential cost field. Similarly to fermions that naturally tend to the minimum energy state, then our cloud must evolve in order to reach the minimum cost state. In particle physics the fermions exhibit antisymmetric wave functions, beside the mathematical formulation and the physical meaning, front this fact depends what is generally known as the Pauli exclusion principle. Such a principle states that in a quantum system two fermions can not share the same pure state, in facts the Pauli exclusion principle is a selection rule for forbidden states. While in quantum mechanics such a rule is taken into account as a brute fact, this natural feature of our universe can as well give us the solution to the considered problem. Within the parallelism among cloud computing and quantum electrodynamics, we will describe the available resources of a cloud as points of local minima for a fitness function. We decided then to use as a fitness function the total energy of a quantum system defined by a partition function for fermion particles, therefore describing the system by a Fermi-Dirac statistical distribution. We ordered the resources according to their

related potential cost, so that the resource R_i covers a less or equal potential cost with respect to R_{i+1}. Then we have imagined the resource requests as free particles, while we described an assigned resource as a bound unoccupied state. Therefore, inspired by the statistical thermodynamics of a fermions gas, we defined an occupation function. Let us suppose to have N_r resources $R_i \in R$ waiting to be occupied in order to fulfil N_q requests $q_j \in Q$. The sets R and Q are continuously populated each time new resources are freed (or added) to the cloud and each time new resource requests are made. Suppose also that while time goes pairs (r_i, q_j) are formed. In this case, if we define $\epsilon(r_i)$ the potential cost of a resource r_i, and $\epsilon(q_j)$ the effective cost[1] of a request, then the occupation function will be algorithmically populated by using the following rule:

$$Z_n = (r_i, q_j) : \epsilon(r_i) \geqslant \epsilon(q_j) \quad \forall \; Z_n \in R \times Q \tag{6.4}$$

where Z_n is the n-esime occupation. For each Z_n then we can obtain an energy difference defined as $\delta : R \times Q \to \mathbb{R}$ so that

$$\delta(Z_n) = \delta(r_i, q_j) = \epsilon(q_j) - \epsilon(r_i) \quad \forall \; Z_n \in Q \times R \tag{6.5}$$

We then define a fitness function $f(\tau) : R \times Q \to \mathbb{R}$ where τ identifies a discrete time step, so that

$$\big[f(Z_n)\big]_\tau = \big[f(r_i, q_j)\big]_\tau = \big[\delta(r_i, q_j) - w_i\big]_\tau \tag{6.6}$$

with $\big[w_i\big]_0 = 0 \;\; \forall \, r_i \in R$ if $\tau = 0$ is the first time step. By means of this fitness function, it is possible to identify a subset of perfectly matching pairs $\Omega = \{Z_n : \big[f(Z_n)\big]_0 = 0\}$. Moreover, we can define the set of non matching pairs as the complementary set $\Theta = \{Z_n : Z_n \notin \Omega\}$. While Ω identified the pairs of resources and requests that naturally permit us to waste virtually no resources, it is still needed to find the minimum configuration for the pairs in Θ. Therefore, at each time step a gradient descent algorithm [29, 38, 22] is applied in order to modify the coefficients $\big[w_i\big]_\tau$ so that:

$$\big[w_i\big]_{\tau+1} = \left[w_i - \eta f(Z_n) \frac{\partial f}{\partial w}\right]_\tau \quad \forall \; Z_n \in \Theta \tag{6.7}$$

[1] We intend as affective cost the real cost that will be paid when chosing resource r_i.

Algorithm 1 Minimization Gradient Descent Algorithm

```
Start,
Define fitness condition f(·),
Define step size η and time horizon T,
Crossmatch the resources and the requests,
Compute the perfectly matching pairs set Ω,
Compute the complementary set Θ,
t = 1,
while t ⩽ T do
    Compute [w_i]_t using (6.7),
    t + +
end while
Return Ω ∪ Θ,
Stop.
```

where η is a fixed step size. Coming back to the imagined fermion gas, this gradient descent algorithm is equivalent to the natural motion caused by the difference of energy with respect to the bound state which is therefore unstable. By imposing as constraint (as it is in nature) that all the free particle states have higher energy with respect to the bound states, then the system naturally evolves to the lowest energy overall state. The same concept is applicable then to the potential cost corrected with the terms $[w_i]_\tau$. It comes trivially that the system also evolves naturally to the optimum, moreover adding or removing resources and requests to the system does not modify the dynamic evolution of the system, but, in this latter case, before to apply the evolutionary algorithm expressed by (6.7), it could be necessary to recompute the population of Ω and Θ. The adopted gradient descent algorithm is a local optimization technique generally based on algorithmic attempts. Such techniques permits to obtain the optimum configuration with a very simple procedure, therefore computationally advantageous with respect to more computationally expensive analytical calculations. The presented numerical algorithm searches for a solution in the defined space by means of recurrent adjustments dependent by the gradient of the fitness function. In Algorithm 1 the developed gradient descent algorithm is presented. The algorithm starts with the population of the perfectly matching pairs set Ω and by computing its complementary set Θ. In the successive steps it computes the gradient ∇f and

then the results of equation (6.7). It has to be noticed that it is the gradient sign that determines the direction of the function descent. At the end of each time step the new configuration of Ω is computed, and the algorithm is iterated until a time horizon T is reached, or it comply with a predefined stop condition.

6.6 Assisting workflow executions

In this chapter it was shown that the presented architectures tackle the need of starting workflow services, while monitoring their execution and controlling the used resources. Services that are part of a workflow can be controlled by our proposed aspect-oriented solution and make these able to automatically resort to cloud resources, without any reengineering effort. Aspects can collect timing related data about the usage of any service involved in the workflow, thus allowing the computation of several metrics about the usage of the services. The automatic scaling on cloud resources, whose number is estimated by a neural network predictor, is performed by distributing requests according to a fitness function that minimises costs. Thus enhancing the availability of services (appropriately replicated on cloud hosts) without having to bear the cost due to in-house hosting. In the previous chapter it was shown that Wavelet Recurrent Neural Networks are able to tamper with the timeline in order to predict the availability of resource. Is it the possible to model human resources with wavelet recurrent neural network, therefore, to manage human workers as if they were nodes on a cloud? Before to answer to such a question a further step must be taken toward models of the human behavior, or, better, to model the human interaction within groups of human. Therefore in the next chapter the radial basis neural networks will be presented as an useful tool to model the behaviors of the living component of the mechanical turk: the human workers.

CHAPTER 7

Inanimate reasons

> One old lady, in particular, who had not forgotten the tales she had been told in her youth [...] went and hid herself in a window seat, as distant as she could from the evil spirit, which she firmly believed possessed the machine.
>
> Karl Gottlieb Windisch

All people, when participating in online activities, social networks, or just while surfing the web, constantly provide personal information, discuss topics of interest, disseminate data regarding their activities. In this work we want to obtain accurate models of some people's behaviour, namely the workers of a crowdsourcing project, in order to create an accurate predictor for their availability in crowdsourcing related projects. In order to do so we will generally refer to users of an online service, trying then to detail the approach for the workers we want to model.

7.1 Keeping a profile

Such a modeling procedure starts by gathering data related to the users' activities in order to build for them an identity out of sparse data. Such an extrapolated identity

provides us with further knowledge and can also indicate some anomalous activities, such as e.g. the tentative of some users to deceive others. In our approach, user profiles are processed by a Radial Basis Probabilistic Neural Network that retains relevant profiles data. Moreover, similarity indexes are computed for user data gathered from profiles and activities. Our analysis unveils the most likely user category and shows commonalities on user behaviours. Given the great importance that online social networks are assuming, it seemed the perfect field of application of the developed techniques also due to the importance that trust and reliability issues assume in this context for both providers and subscribers. However, analysing social networks is greatly expensive in terms of computational time, i.e. data could easily grow into hardly manageable amounts, even when considering high end computers, and such data have to be periodically explored to scan user triggered updates. The large size of an online social network, in terms of subscribers, number of mutual interactions and links (such as *friendship*, *following*, message exchange, endorsements, group memberships, etc., according to the social network at hand), generates a big amount of data that can only be handled by an *automatic support* to efficiently ensure security and validate the provided content. In this field, user *feature* classification and *behavioural* analysis can be considered an interesting and important means to help providers check that contents are suitable and safe for subscribers. This paper provides a solution built upon the said means. There are many ways for identifying categories of users. Interesting performances have been given by user field-of-interest detection, however, in general, such systems lack of scalability and are only intended for a small context, or for an analysis on a single-user basis. Moreover, mathematical statistical methods make it possible to characterise specific features and interests for a single user [108], however it is still difficult to obtain a valid analytical model describing user interactions. The main problem of analytical approaches lies in the vastness of the datasets describing the activities typical of an online social network, such as the number of connections, and the undetermined number of attributes that have to be gathered in order to recognise a user behaviour, etc. A huge amount of categories could be used to characterise subscribers, however a significant portion of values for the attributes in each category could be missing for many subscribers,

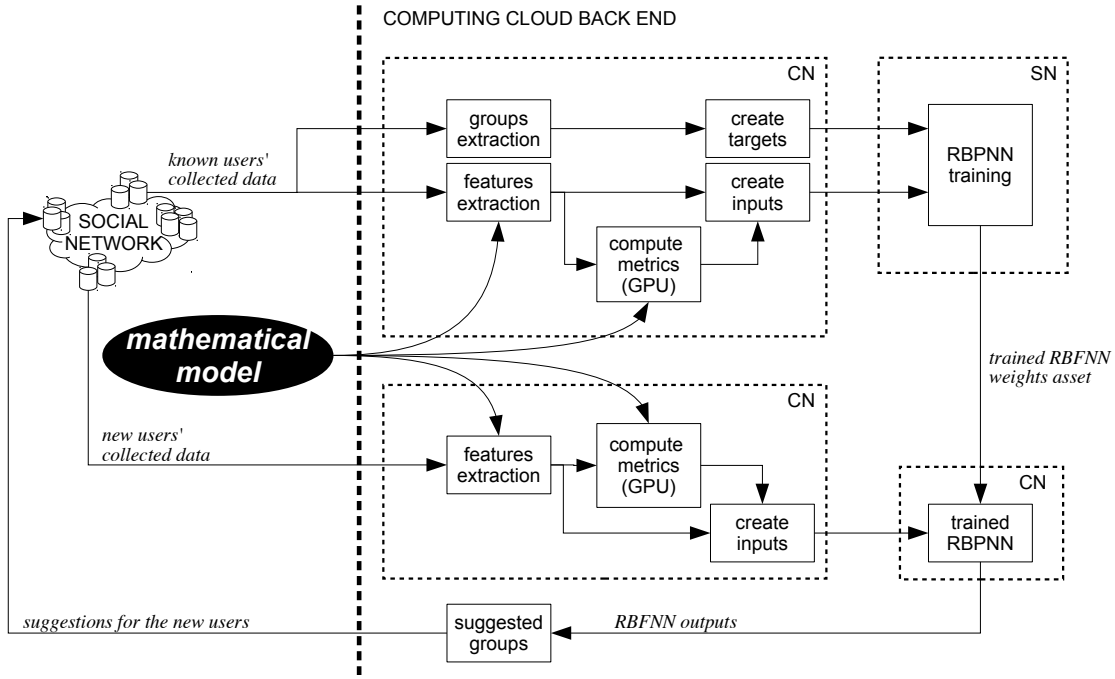

Figure 7.1: An overall schema of the proposed architecture

making it difficult to obtain a correct and complete formulation of a comprehensive analytical system. Other limitations of an analytical model for representing a social network, and restraining the possibility to analyse the behaviour of subscribers, are given by the dynamic changes of the state, i.e. the whole amount of data available on the online social network itself. Such changes would require frequent reiterations on the formulation of the analytical model. Moreover, since data are continuously updated and modified, defining the appropriate variable and parameter size needed to solve the problem analytically can be rather difficult. Despite such difficulties, it is highly desirable to have an automatic support that can dynamically incorporate new data from the online social network over time, and that could be largely used for advanced services. We propose a solution based upon a specific architecture called Radial Basis Probabilistic Neural Network (RBPNN) described in the following. Such an architecture is well known for its capability to classify and generalise given datasets by creating a model of input sets. Since Artificial Neural Networks can be continuously trained to recognise novel features on data, an Artificial Neural Networks can

easily cope with the rapid changes of the dataset, without the need to start all over the analysis. The proposed approach puts forward a classification model computed out of the large amount of data coming from user profiles of different portions of the social networks [103]. As for any social networks, so as for online social networks (like Facebook), the main difficulties are due to, firstly, the unknown total size in terms of the number of: subscribers, friendship relations, groups, followers, etc.; and secondly, the unknown size of data and features for each subscriber. We overcome such difficulties thanks to the use of RBPNN that can cope with partial data and act as a modeler for dynamically changing user profiles. On top of our proposed solution several more specific user behaviour analyses can be additionally built. E.g. it would be possible for an administrator to mark a profile for a subscriber that has been recognised as *bad behaving*, and let the RBPNN find other profiles having strong similarities. The proposed RBPNN analyses a large amount of users and finds categories for them as well as profiles of possibly bad behaving users, which can be further investigated. Furthermore, the early identification of autogenous threats can be automated, the proposed RBPNN identifies incoherent profiles, when the predicted user behaviour does not match the real behaviour, or when the characteristics of a subscriber definitely match other known profiles of undesirable users. The proposed approach also makes use of a massively parallel solution to compute metrics for large amounts of data produced by online social network users. Figure 7.1 shows an overview of our proposed solution.

7.2 Social networks dynamics

It is possible to represent a social network or any other kind of collaborative network as a graph, where nodes are the users and arcs are the relations among users (i.e. friendship, sharing, following, etc.). Social networks follow a *scale-free* behaviour [16], i.e. a few nodes act as important hubs centralising a large number of links. However, in 'ego' social networks, the small-world properties [6] represent an important characteristic related to the real social dynamic of the network, as it will be described in the following. This work analyses social networks, and Facebook is a significant

representing example. We consider that a bidirectional interaction between a pair of subscribers exists when a friendship exchange exists between the pair. Moreover, Facebook provides *groups*, i.e. a user is given means to broadcast contents to all the members of the same group where s/he belongs to. Users belonging to the same group have a relationship with each other. As social networks can be represented as graphs where the users are nodes, and the arcs are their interactions, then we state that a *user* is an element in a set $\mathcal{U}$ that we call *user pool*, since it consists of all the users of the social network. On such a set it is possible to abstract the users in several ways (i.e. an adjacency graph, based on mutual relationships), to uncover different properties of the social networks. As much as users, interactions, information exchange and mutual properties can be formalised. We define three kinds of singular or mutual properties among users, as follows:

1. *feature*: a function
$$f : \mathcal{U} \to \mathscr{P}(\mathcal{F}),$$
where $\mathscr{P}(\mathcal{F})$ is the part set of the feature set $\mathcal{F}$;

2. *relation*: a binary function
$$r : \mathcal{U} \times \mathcal{U} \to \{1, 0\};$$

3. *category*: an equivalence class $\mathscr{C}$ of $\mathcal{U}$ with respect to users, features or relations.

The given definitions make it possible to obtain a functional $F : \mathcal{U} \times \mathcal{U} \to \mathscr{P}(\mathcal{F})$ which associates a pair of users $u_0, u_n \in \mathcal{U}$ with the set of features that the said users share. Hence, we have that

$$F(u_0, u_n) = f(u_0) \cap f(u_n) \tag{7.1}$$

Examples of features can be gender, age, city, work, etc., or some interest or profile content, as well as any other comparable characteristic (also the color and theme of the page could be interpreted as features, though the latter were not used for this work).

While the described features characterise the commonalities among two users (hence, among two nodes of the equivalent graph), we will now extend the given formalism in order to represent the interactions among users (hence, the arcs of the graph). In the most popular social networks, when user u_1 performs an interaction with user u_2, then u_1 will be a *friend* or a *follower* (or *endorser*, or any other similar definition) of u_2. Since for such an interaction it is involved a mutual exchange of information (profile content, posts, interests, shared content, etc..), we will call this interaction a "mutual interaction". The given formalism describes such a mutual interaction as a *relation* r_f among two users u_1 and u_2 so that $r_f(u_1, u_2) = 1$ when u_1 has established a mutual interaction with u_2. In the same way and for the same reasons regarding the mutual exchange of information and content, we want to additionally describe the interaction occurring when two users have a membership to the same group (e.g. a Facebook group, which is also the selection of users having a common interest). In this case we will make use of another *relation* that we will indicate as r_g, in order to distinguish it from a friendship (or similar) *relation* r_f. Now, within the given formalism, it is possible to define the *categories* as classes of equivalence $\mathscr{C}$ among users according to a given mix of *features* and *relations*, i.e. *categories* depend on the features, interactions and common interests among users (as well as possible derived data). Therefore, categories model the partitions of users according to their interests and behavioural patterns.

7.3 Paths and distances

The relations and features give us a set of mathematical tools that we use to represent an online social network as a graph with convenient properties. Once complete, this formal structure will allow us to define two kinds of distances for users in the social network. Such distances will be of key interest in order to characterise the social network. Now on we will represent the social network by means of a graph, therefore we will use a pair of sets $(\mathcal{U}, A)$. While $\mathcal{U}$ is the set of users in the social network and will act as a *vertex set* for the graph, the set A is the *arc set* A, which, in the

proposed formalism, is defined as

$$A = \{a(u_i, u_j) : r_f(u_i, u_j) + r_g(u_i, u_j) \geqslant 1\} \tag{7.2}$$

$\forall\ (u_i, u_j) \in \mathcal{U} \times \mathcal{U}$ and whereby r_f and r_g are the mathematical relations defined yet. If there is a finite number of *arcs* $a \in A$ that go from a vertex u_0 to a vertex u_n of $\mathcal{U}$, then it exists at least one *path*

$$P_G(u_0, u_n) = \left\{a(u_i, u_{i+1})\right\}_{u_0}^{u_n} \tag{7.3}$$

where $a(u_i, u_i)$ are arcs from u_i to u_i. If a path exists from u_0 to u_n, then a *length* $l[P_G(u_0, u_n)]$ can be attributed to this path. In this work we will define the length of a path as the cardinality $|P_G(u_0, u_n)|$, that is the number of arcs forming the path. With the given definitions of *arcs* and *length* it is then possible to devise a *graph distance* $d_G : \mathcal{U} \times \mathcal{U} \to \mathbb{R}_0^+$ as

$$d_G(u_0, u_n) = \begin{cases} 0 & u_0 = u_n \\ \min\left\{l[P_G(u_0, u_n)]\right\} & \exists P_G(u_0, u_n) \\ +\infty & \nexists P_G(u_0, u_n) \end{cases} \tag{7.4}$$

The graph distance allows us to consider the existence of two different kinds of links among users: a *direct link* when a pair of users has exchanged a friendship, then the distance is simply 1, otherwise an *indirect link*, when a pair of users are "friends of a friend". In this case the distance is greater than 1 as, from the definition (7.4), graph distance is the minimum count of friendship-hops separating the pair of users sharing such an indirect link. Following the definition given in equation (7.2), users that are members of the same group are linked by an arc, which reflects their existing relation, hence the value of r_g is 1. In this case, for such a pair of users, since in our representation they are connected by one arc, the distance is cut down to 1. This kind of property is obtained because of the structure of the arc set and is helpful in order to characterise the commonality of interests of the users, which in general can decide to become members of the same group. As a matter of fact, groups could

be also described as a means for the information to rapidly flow across users who generally share no friendship (or equivalent relation in other online social networks). Finally, as anticipated earlier, our model makes use of two kinds of distances. These two kinds of distances are needed in order to compute the numerical indexes that are of high significance. We will now introduce a second kind of distance, with respect to d_G. This second kind of distance takes in consideration the features characterising the users. As said before, each user on a social network has a profile with a certain number of features which has been formalised as a function to the set $\mathcal{F}$. On the other hand, by means of its algebraical properties, $\mathcal{F}$ is also a basis for an Hilbert space. In such a space a user u can be classified according to a *feature vector* $\phi(u)$ that we defined as

$$\phi(u) = \Big[\varphi_u(f_\alpha)\Big]_{\alpha=1}^{N} \;:\; u \in \mathcal{U}, f_\alpha \in \mathcal{F} \tag{7.5}$$

where $N = |\mathcal{F}|$, and $\varphi_u(f_\alpha) = 1$ if the profile of user u shows the feature f_α, otherwise with $\varphi_u(f_\alpha) = 0$. The given definition of *feature vector* takes into consideration pairs of users u_0 and u_n, then it is possible to define a *feature distance* $d_{\mathcal{F}}(u_0, u_n)$ as

$$d_{\mathcal{F}}(u_0, u_n) = 1 - \frac{1}{N}\sum_{\alpha=1}^{N} \delta\Big(\varphi_{u_0}(f_\alpha), \varphi_{u_n}(f_\alpha)\Big) \tag{7.6}$$

where δ represents the delta of Kronecker. It follows that two hypothetical users (u_1, u_2) with the same profile (let us assume that each feaure has the same value for the users) will have a distance $d_{\mathcal{F}}(u_0, u_n) = 0$, while two other users u_3, u_4 with nothing in common will have a distance $d_{\mathcal{F}}(u_3, u_4) = 1$. This definition of distance can be implemented in a simple manner by using the *Hamming distance*, since it is trivial to reduce it to the cardinality of the set resulting from the functional $F(u_0, u_n)$ computed as in (7.1). In the same manner, other kinds of mutual information indexes, such as the *Jaccard similarity* are appropriate to find the degree of commonalities for user profiles.

7.4 A lesson from Mutual Information Theory

The aim of the defined distances is to obtain significative and coherent information about the behavioural proximity of different users of a certain social network, on the other hand this task can be formalised as the computation of the mutual similarities starting from a set of features. In the introduced formalism for each user we have defined a feature vector $\phi(u)$ of boolean values that identifies the presence or absence of certain features on the user profile. Since the features $f_\alpha(u)$ are independent of each other (let there be N different features), it is possible to define from $\phi(u)$ the relative partially ordered set

$$\Phi(u) = \{(f_\alpha(u), \alpha) : \alpha \in [1, N] \subset \mathbb{N}\} \ \forall \ u \in \mathcal{U} \tag{7.7}$$

Starting from such a set of ordered features, it is then possible to give a similarity meaning to the defined distances using a pair of sets $(\Phi(u_0), \Phi(u_n))$ and by computing the Jaccard similarity coefficient and Hamming distance.

$$\begin{aligned} J(u_0, u_n) &= \frac{|\Phi(u_0) \cap \Phi(u_n)|}{|\Phi(u_0) \cup \Phi(u_n)|} \\ H(u_0, u_n) &= 1 - \frac{|\Phi(u_0) \cap \Phi(u_n)|}{N} = d_{\mathcal{F}} \end{aligned} \tag{7.8}$$

$J(u_0, u_n)$ is the Jaccard similarity coefficient and $H(u_0, u_n)$ the Hamming distance of user u_0 with respect to user u_n; such parameters have been computed for this work in a highly parallelised way [147], since such computations can be managed independently. The two indexes J and H are relevant for suggesting an effective affinity of interests among users who are not necessarily linked directly. Figure 7.2 shows the Jaccard coefficients for each pair of users examined, according to their features. In our analysis, we consider that when a user posts a content into a group, then this results as a one-to-all interaction with other members of the group, who generally share a limited number of interests with each other. From the friends list of each subscriber, we identify *clusters* of users $\Omega_k \subset \mathcal{U}$. A cluster consists of users having a higher number of friendships toward users within the same cluster rather

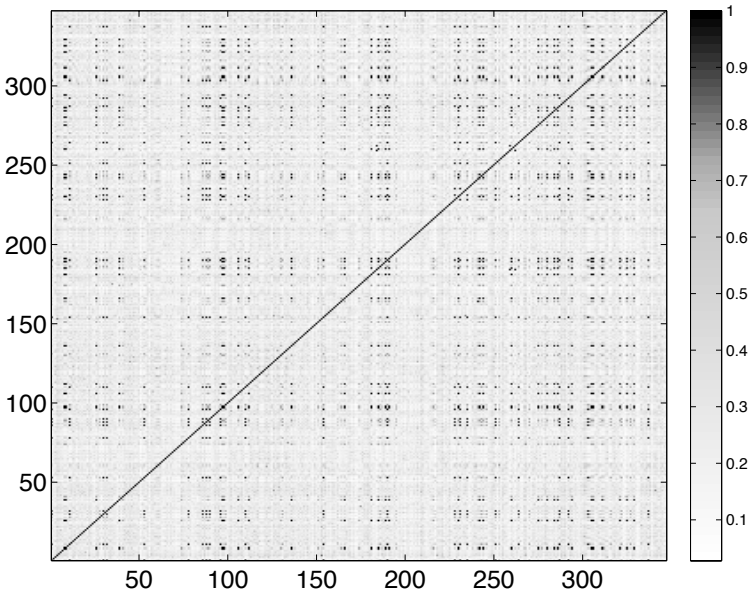

Figure 7.2: Jaccard similarity coefficient $J(u_0, u_n)$ computed for the features of user pairs.

than toward users not belonging to the cluster. Therefore, we have that

$$\begin{aligned} &\max\{d_G(u_{k0}, u_{kn})\} < \min\{d_G(u_{k0}, u_{k*})\} \\ &\forall\ u_{k0}, u_{kn} \in \Omega_k \subsetneq \mathcal{U} \\ &\forall\ u_{k*} \in \mathcal{U} \backslash \Omega_k \neq \varnothing \end{aligned} \tag{7.9}$$

Analogous to the distance between users, we define d_C as the distance between a cluster pair as the minimum count of hops separating one user on the first cluster from one in the second cluster:

$$d_C(\Omega_h, \Omega_k) = \min\{d_G(u_h, u_k) : u_h \in \Omega_h, u_k \in \Omega_k\} \tag{7.10}$$

It follows that if $\Omega_h \cap \Omega_k \neq \oslash \Rightarrow d_C(\Omega_h, \Omega_k) = 0$, otherwise if $\Omega_h \cap \Omega_k = \oslash$, the two clusters can be considered independent parts of the social network that still satisfy the scale-free properties. Let us suppose that two users belong to different clusters, while being on the same group. When considering the relationship between users and a group, we can see that a group is acting as a bridge for the contents to flow

from a cluster to another (the clusters of the correspondent users). Hence, different parts of the network become mutually capable of exchanging contents, fostering the small-world behaviour of the social network [181]. In this way, user clusters, representing small parts of the social network, communicate by using *weak* links, i.e. their mutual membership to the same group, rather than using *strong*, i.e. their mutual relations. Nevertheless, clusters generally consist of users sharing a set of interests and activities, and users on a cluster form a sort of social neighbourhood [87]. Such an assumption makes it possibile to represent a network as a certain number of independent adjacency structures, each focusing on a partition of a social network graph. I.e. an adjacency structure is representing a cluster of socially close people and their relative bridges to other clusters. The main difference between a formal scale-free graph and an online social network is given by the percolation of links [182, 9]. In real life, how worth a certain friend is tends to decrease if there is no good reason to maintain the relationship. This decrease of interest is still true even in a social network, however it has no corresponding support in practice. The impossibility to accordingly classify links results into inaccurate data when performing an automatic analysis. Generally, for social networks that let users participate in a group, an average subscriber tends to sign into a large number of groups, while only a small amount of such groups are really interesting for the user. The said widely-spread user behaviour provides additional inaccuracy on the data to be analysed. In turn, automatic selections and suggestions of the posts provided by friends or their groups becomes less useful, because of such inaccuracies. Moreover, it is difficult to distinguish between trustworthy users and dishonest or unreliable ones. Even though the profile of a user can be potentially genuine, differently from social networks, human networks evolve following a homophily law [132] leading a person to connect with others having similar 'real' interests. Hence, the homophily law lets us detect and reason with small, though relevant, differences between social networks and theoretical scale-free networks. Because of such differences, an existing online social network cannot adhere to a simple mathematical model, instead, since the stochastic behaviour typical of human beings is exhibited, an advanced nonlinear model is needed, which cannot be built using an analytical solution. Due to the said untrustworthy, erratic, inconstant

and unreliable behaviour of users, we maintain that it is paramount to uncover hidden or un-explicit interests, which give a representation of the effective relationships among users. Such (hidden) relationships are significant to find categories of users that unveil their real behaviour. In this way, clustered users uncover the existence of a non-explicit group. The clustering operations are performed by means of a specific neural network architecture: the yet encountered which is explained in the following section.

7.5 The RBPNN classifier

Classical models suffer of the incompleteness of the initial input dataset. In the past, to overcome such problems neural networks have been largely used to uncover data classification and to find probabilistic categories for data clustering. For the work proposed here, we use Probabilistic Neural Networks (PNN), relying on the hypothesis that it is possible to define *groups* of users as statistical categories. PNNs have a topology similar to common FeedForward Neural Networks (FFNN) with BackPropagation Training Algorithms (BPTA): the primary difference only lies in the activation function that, instead of being a sigmoid function or a similar activation function, is a statistical distribution or a statistically significant mathematical function. The kinds of activation functions used for PNNs have to meet some important properties to preserve the generalisation abilities of the ANNs. In addition, these functions have to preserve the decision boundaries of the PNNs. This kind of neural architecture if correctly trained is capable to generate a model for the latent features [136] for which there is a non explicit link among users [121]. Figure 7.3 shows the selected RBPNN architecture that takes advantage from both the PNN topology and the Radial Basis Neural Networks (RBNN) used in [24]. In a RBPNN both the input and the first hidden layer exactly match the PNN architecture: the input neurones are used as distribution units that supply the same input values to all the neurones in the first hidden layer that, for historical reasons, are called *pattern units*. In a PNN, each pattern unit performs the dot product of the input pattern vector $\mathbf{u}$ by a weight vector $\mathbf{W}^{(0)}$, and then performs a nonlinear operation on the result. This nonlinear

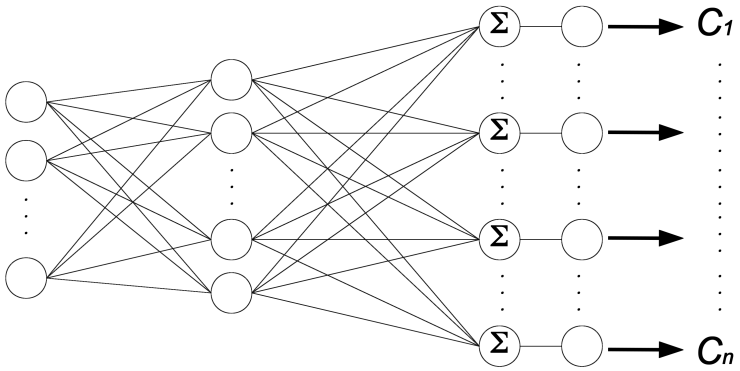

Figure 7.3: A representation of a Radial Basis Probabilistic Neural Network

operation gives output $\mathbf{x}^{(1)}$ that is then provided to the following summation layer. While a common sigmoid function is used for a standard FFNN with BPTA, in a PNN the activation function is an exponential, such that, for the j-esime neurone the output is

$$\mathbf{x}_j^{(1)} \propto \exp\left(\frac{||\mathbf{W}^{(0)} \cdot \mathbf{u}||}{2\sigma^2}\right) \tag{7.11}$$

where σ represents the statistical distribution spread. The given activation function can be modified or substituted while the condition of Parzen (window function) is still verified. In this case, while preserving the PNN topology, to obtain the RBPNN capabilities, the activation function is substituted with a radial basis function (RBF); an RBF still verifies all the conditions stated before. It then follows the equivalence between the $\mathbf{W}^{(0)}$ vector of weights and the centroids vector of a radial basis neural network, which, in this case, are computed as the statistical centroids of all the input sets given to the network. We name f the chosen radial basis function, then the new output of the first hidden layer for the j-esime neurone is

$$\mathbf{x}_j^{(1)} \triangleq f\left(\frac{||\mathbf{u} - \mathbf{W}^{(0)}||}{\beta}\right) \tag{7.12}$$

where β is a parameter that is intended to control the distribution shape, quite similar to the σ used in (7.11). The second hidden layer in a RBPNN is identical to a PNN, it just computes weighted sums of the received values from the preceding neurones.

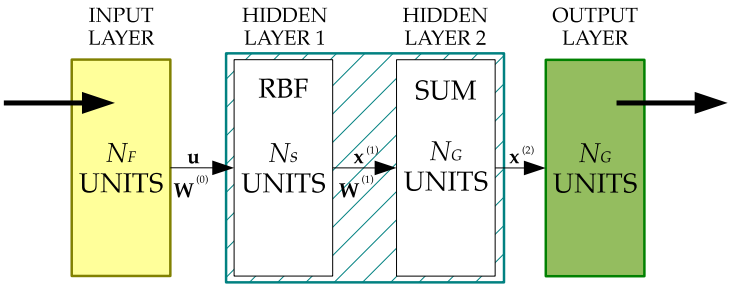

Figure 7.4: Setup values for the proposed RBPNN: N_F is the number of considered features, N_S the number of analysed subscribers, and N_G is the desired number of groups.

This second hidden layer is called indeed summation layer: the output of the k-esime summation unit is

$$\mathbf{x}_k^{(2)} = \sum_j \mathbf{W}_{jk}\mathbf{x}_j^{(1)} \tag{7.13}$$

where $\mathbf{W}_{jk}$ represents the weight matrix. Such weight matrix consists of a weight value for each connection from the j-esime pattern units to the k-esime summation unit. These summation units work as in the neurones of a linear perceptron network. The training for the output layer is performed as in a RBNN, however since the number of summation units is very small and in general remarkably less than in a RBNN, the training is simplified and the speed greatly increased [59]. The devised topology enables us to distribute to different layers of the network different parts of the classification task. While the pattern layer is just a nonlinear processing layer, the summation layer selectively sums the output of the first hidden layer. The output layer fullfils the nonlinear mapping such as classification, approximation and prediction. In fact, the first hidden layer of the RBPNN has the responsibility to perform the fundamental task expected from a neural network [217]. In order to have a proper classification of the input dataset, i.e. of users into groups, the size of the input layer should match the exact number N_F of features given to the RBPNN, whereas the size of the pattern units should match the number of subscribers N_S. The number of the summation units in the second hidden layer is equal to the number of output units, these should match the number of groups N_G we are interested in for classifying users

(Figure 7.4).

7.6 User clustering from RBPNNs

As described in the previous section, the implemented RBPNNs, once trained, starting from the features vector $\phi(u_n)$ given as the input vector $\mathbf{u}(u_n)$ of the network, assign the user u_n to one category. Moreover, the RBPNNs are designed to choose among the available categories by means of a probabilistic deduction performing a max-argument selection of the input values given to the last neuron. By dynamically on the fly the weight values, it is possible to intercept the last chosen category for a certain user, and then force to zero the corresponding input (modifying to zero the relative connection weight) given to the last neuron, and subsequently ask for a second choice, as a second–best category. By iteration this process produces the top-ranking category list. Then, when some conditions are satisfied, the weight can be simply restored to the original value before another user is processed. It follows that for each user it is possible to use the RBPNNs to obtain a categories set (unordered)

$$\Gamma(u_n) = \{\mathscr{C}_k(u_n)\} \quad \forall\ u_n \in \mathcal{U} \tag{7.14}$$

where $\mathscr{C}_k(u_n)$ represents the selected categories and k a numerical index. With such categories the RBPNN is able to perform user clustering, e.g. in order to suggest new groups to the users, to identify anomalous behaviour, control the autogenous threats or simply obtain an accurate statistic on the user preferences. The implemented RBPNN is able to classify users starting from a set of inputs given to the network. Therefore it is important to feed the network with appropriate inputs capable to characterise the users and their similarities. While the user profile features are characterised by the feature vector $\phi(u)$ as defined by equation (7.5), it is also useful to characterise the distances among the same user and a group (or category) of users of interest: e.g. what is the average distance between the user u and the users belonging to a certain group, or, in alternative, measure the average distance between u and several users yet identified as dangerous or misbehaving. To obtain the distances

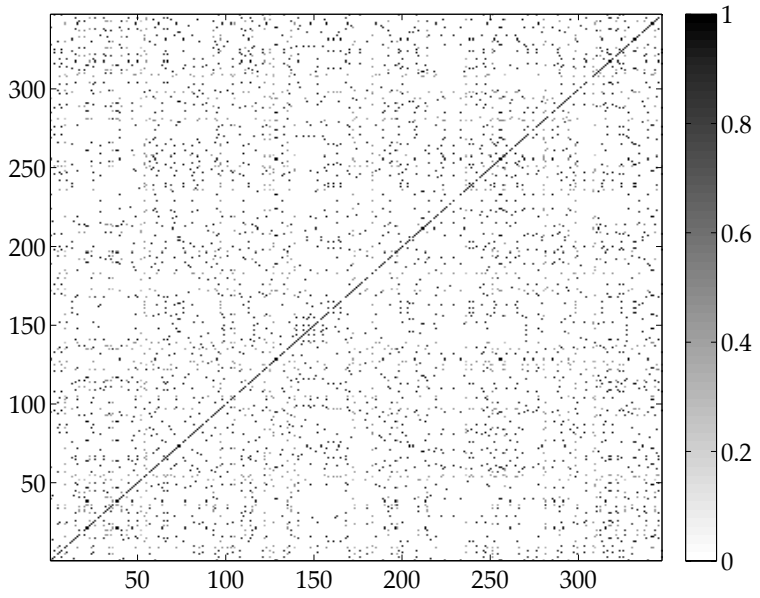

Figure 7.5: The $J_{\mathscr{C}}(u_0, u_n)$ similarity coefficients

we will make use of the analytical form of equations (7.4). On the other hand, in order to obtain d_F, instead of using (8.4), basing on equations (7.7) and (7.8), it is appropriate to make use of mutual information indexes. Moreover, while d_F can be computed as the Hamming distance yet introduced for from the features vector $\phi(u)$, it is useful to obtain at a minimum cost (within the same computation) other mutual information indexes such as the Jaccard similarity coefficient. Moreover, also d_C can be computed in a similar manner with

$$\begin{aligned} J_{\mathscr{C}}(u_0, u_n) &= \frac{|\Gamma(u_0) \cap \Gamma(u_n)|}{|\Gamma(u_0) \cup \Gamma(u_n)|} \\ H_{\mathscr{C}}(u_0, u_n) &= 1 - \frac{|\Gamma(u_0) \cap \Gamma(u_n)|}{M_{\mathscr{C}}} \end{aligned} \tag{7.15}$$

where $M_{\mathscr{C}}$ is the total number of considered categories. Basing on the previously given information, let suppose to classify users into three different categories $(\mathscr{C}_1, \mathscr{C}_2, ..., \mathscr{C}_M)$. for each user classified, the output set will be a vector composed of three binary components, in order to encode the membership of the user to the three given categories, e.g. $(1, 0, ..., 1)$ will indicate that the user belongs to the categories $\mathscr{C}_1$ and $\mathscr{C}_M$ but

don't belongs $\mathscr{C}_2$. On the other hand the input set will depend by the categories themselves. In facts in this specific example we will feed the RBPNN with the following input set:

$$\mathbf{u} = \begin{pmatrix} f_1 \\ ... \\ f_N \\ 1 \\ < d_G >_1 \\ < H >_1 \\ < J >_1 \\ < J_{\mathscr{C}} >_1 \\ < H_{\mathscr{C}} >_1 \\ ... \\ < d_G >_M \\ < H >_M \\ < J >_M \\ < J_{\mathscr{C}} >_M \\ < H_{\mathscr{C}} >_M \end{pmatrix} \tag{7.16}$$

where f identifies the features as in (7.7), followed by a bias equal to 1, followed by some average measurements. Such measurements are a group of five indexes for each output categories and are computed as the mean mutual index among the users to which is referred the input vector, and all the other users belonging to the category where to classify the input vector. This groups of five index is composed by:

1. the average graph distance $< d_G >$ from (7.4);
2. the average Hamming index $< H >$ from (7.8);
3. the average Jaccard index $< J >$ from (7.8);
4. the average Hamming index $< H_{\mathscr{C}} >$ from (7.15);
5. the average Jaccard index $< J_{\mathscr{C}} >$ from (7.15).

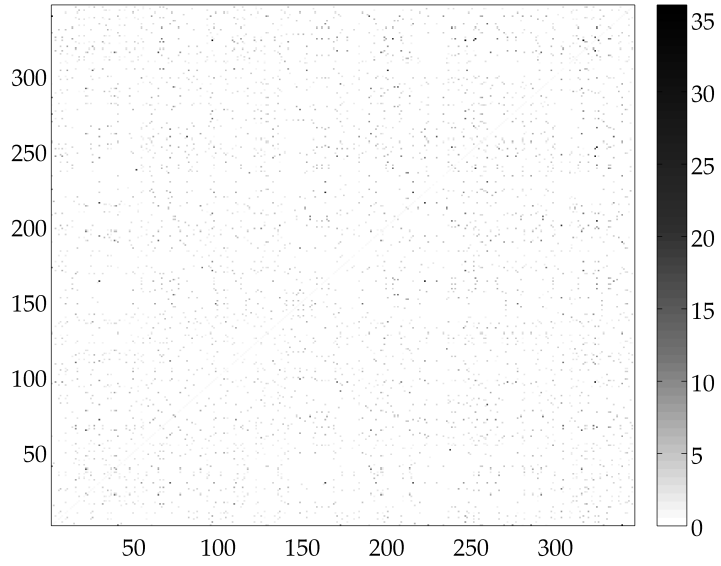

Figure 7.6: The relative ratio $J(u_0, u_n)/J_{\mathscr{C}}(u_0, u_n)$

Although the high-level point of view masks the size of the data involved in order to create such an input vector, an example the dimension of the problem could be devised by Figure 7.5 where are shown the values of the Jaccard similarity coefficients for user pairs when considering categories (a black dot indicates a high level of similarity). As well, Figure 7.6 shows whether two users are similar for their features vs their categories (or groups or behaviour). It is trivial to understand that in order to obtain average values for each users, with respect to each user in each category, the problem size increases dangerously. Therefore, while the architecture is linear enough, the support of parallel method is required in order to cope with the required computations. In the following section the proposed parallel implementations are explained.

7.7 A parallel implementation

The above example has examined a very small cluster consisting of 350 users with their 250 features, considering only 32 categories for the classification. For this cluster,

the needed data to train the RBPNN and calculate Jaccard similarity coefficient and Hamming distance amounted to a few MB input files, so as the resulting output. On the other hand, data sizes can dramatically increase when considering more users. For a cluster of N_u users characterised by N_f features to classify in N_c categories, the input data sizes can be described as:

1. size of the graph for the cluster:

$$S_g = N_u^2 \times \texttt{sizeof(double)}$$

2. size of the users feature list:

$$S_f = N_u \times N_f \times \texttt{sizeof(int)}$$

3. size of the users categories list:

$$S_c = N_u \times N_c \times \texttt{sizeof(int)}$$

It follows that the size S_D of the datasets scales as

$$S_D = S_g + S_f + S_c \propto N_u^2 + \mathscr{O}(N_u) \tag{7.17}$$

S_D further increases when a user appears into several clusters or when a larger number of categories is considered. A social networks like Facebook handles 1.01 billion users and 630 million groups (data released by Facebook Inc. on september 2012): it means that, counting only the 250 features used for this work, and neglecting the repetition of users in different clusters, the total amount of data is of the order of 2.0721×10^{25} entries, for a total amount of 1.2061×10^{15} TB of data. Therefore, evaluating only small but relevant portions of a social network can not be performed by a single machine, moreover it would be impossible to obtain the result of $N_u^2 + \mathscr{O}(N_u)$ operations in an acceptable timeframe. The size of data due to the activities of a social networks requires that selection rules are put into place in order to select portions of the social

network to be analysed. While administrators can be empowered to request a specific analysis on selected users or categories, indeed an automatic means should be provided to continuously explore and analyse the social network. Several kind of selection rules can be set to cluster the network and choose the raw data to analyse. The same care should be applied on the selection of the categories used to classify the users. While for some kinds of categories (e.g. the category of badly behaving users) there is an evident and explicit a priori interest, it would be also important to automatically find anomalous or endangering groups to use as a category for the identification of treats and suspect users. Such automatic selection could refer to a linguistic interpretation of the contents shared on the groups (e.g. post and comments). Once the selection of users and categories ends, then the analysis can be performed. Still given the large amount of data selected, it is desired to use a massively parallel system for the proposed analysis. The following describes a hybrid architecture taking advantage of both Message Passing Interface (MPI) and GPGPU Computing. A GPGPU consists of several MIMD (multiple instruction multiple data) multiprocessors each containing a set of SIMD (single instruction single data) processors. Each MIMD multiprocessor is equipped with a shared memory that can be accessed from each of its SIMD processors, and a global memory common to all multiprocessors. In CUDA programming model, an application consists of a *host* program that executes on the CPU and other parallel *kernel* programs executing on the GPU [167]. A *kernel* program is executed by a set of parallel threads. The *host* program can dynamically allocate device *global* memory to the GPU and copy data to (and from) such a memory from (and to) the memory on the CPU. Moreover, the *host* program can dynamically set the number of threads that run on a *kernel* program. Threads are organised in blocks, and each block has its own *shared* memory, which can be accessed only by each thread on the same block. For maximum performances, threads on a GPU should ideally be given a task that can run unconstrained, i.e. without having to synchronise with others [161]. Moreover, it is paramount that interactions between CPU and GPU are minimised, this avoids communication bottlenecks and delays due to data transfers. Following such general guidelines, we have developed a parallel software system, that takes as

input the list of features associated to a user in the cluster under analysis, and completes the set with the classification resulting from the RBPNN-performed analysis running on the node. We have used arrays for holding relevant data, since they can be easily and efficiently passed to the GPU, i.e. the *host* program allocates memory and transfers data to such a memory, which CUDA *kernel* program can use, by calling the standard `cudaMemcpy()` function. The computation tasks are performed on the GPU devices and programmed in a *kernel* function calling two different *device* functions for each of the two implemented metrics to compute. The values on the arrays are read by each thread, however threads need not write any value on the arrays, hence no synchronisation has been used for accesses. For the device functions computing the similarity metrics, each available thread is given a range of users pairs, representing a subset of all the available pairs to be analysed. For the given range of users, a thread executing inside our function computes all the Jaccard similarity coefficient and Hamming distance values between one fixed user and all the other users. The selection of the range of user pairs to be given to a thread is easily determined by the maximum number of pairs available, the maximum number of cores on the device and the `ThreadId`, available in CUDA programs, indicating the current working thread. Each thread stores results into its own local array, i.e. separately from other threads, hence minimising the need of synchronisation. Given the large amount of pairs, only meaningful values are stored, i.e. only values that are greater than zero (otherwise we risk filling up all available memory). Once a thread has finished executing, it will have computed and stored a given amount of results, which likely differs in number from that of other threads. This is because each thread will find a different number of zeros as a result. The meaningful values will have to be stored on a globally accessed array, so that other functions on the device can use them. For this, each thread reserves an amount of locations to store its computed values. Reservation has been performed by updating a global variable, shared by threads, hence by using the `atomicAdd()` function. This is the only moment for threads to synchronise with each other.The gain that we have obtained is in good agreement with previous assessments of other programs, when we compare an appropriate solution using GPU resources with an optimised solution on a CPU [119].

7.8 Cloud-based strategies

The aim of the proposed solution is to perform fast and distributed computation while preserving the independency of the data sources and authorised clients that access results. Moreover, since the workload for the analysis is unpredictable, and we want to handle the analysis for different kinds of clusters, even when required by an administrator, the proposed solution includes the ability to allocate and release computation resources over time.

To satisfy such requirements, the proposed solution takes advantage of cloud-based resources. We assume that a *resource manager* is put into place to find and allocate resources. Basically, when new data have been produced on the social network, then gathered and properly enqueued for analysis, the resource manager indicates an available idle node to handle the processing, thus dequeueing data. On the other hand, when the queue reaches a given threshold, meaning that the analysis being performed is slow compared with the rate of data produced, new nodes will be allocated and used as parallel resources that can perform the desired computation. Moreover, as an optimisation for the classification task that has to be performed, our solution recognises the kind of RBPNN topology and training required, among available ones that have been previously employed, for the new set of data at hand. This allows starting with the proper trained RBPNN and avoid time consuming training of new neural networks. When a data set matches the training and topology of an existing neural network, the input will be sent to the node having used it, or, if needed, a new node is required from the cloud and the neural network is simply created as a copy of the existing one. The tasks performed to initialise and run analyses can be divided into several main phases and relative steps, here we propose a simplified schema. Analyses are performed both periodically and when requests arise by an administrator of the social networks. Firstly, users, their connections and categories (i.e. groups or some identified types of users) are selected for being analysed as a result of a query considering multiple criteria, such as e.g. a set of keywords, changing user locations, etc. For each included user the features on his/her profile are gathered, here some rules can be set up by an administrator to extract only some features

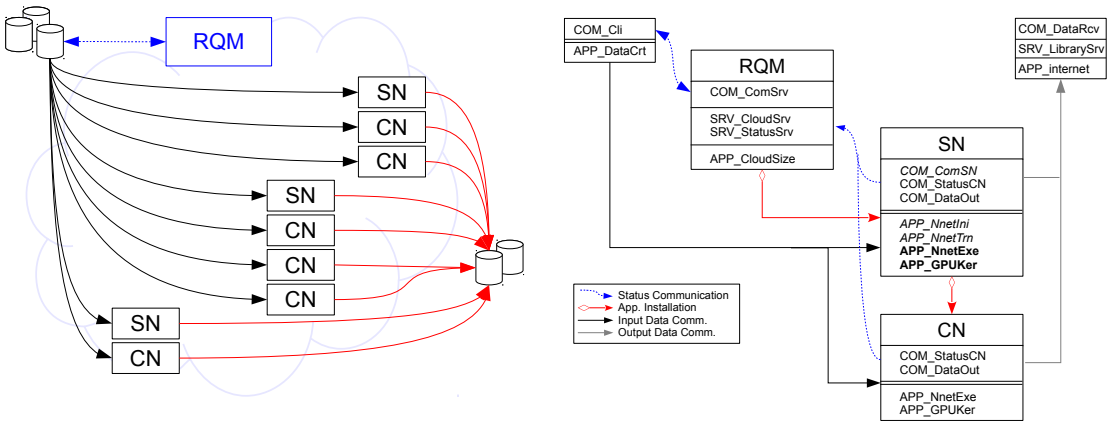

Figure 7.7: The cloud architecture selected.

for the further analysis to be performed. Once users, feature kinds and categories have been selected, the handling system can connect to cloud-resources. When an analysis is required on demand, it is also possible for the administrator to choose a set of users (or even pick up only one user) and then provide a set of rules to limit the analysis to a certain cluster: in this case it is mandatory to select a threshold T_{d_G} for the graph distance d_G. In this way, a user u will be considered belonging to the cluster only if $\exists\ u_\Omega \in \Omega : d_G(u, u_\Omega) < T_{d_G}$, where u_Ω is a subscriber selected by the administrator to begin the analysis, and Ω indicates the selected cluster. Another possibility for selection is given when the analysis on different clusters has to be started simultaneously, then clusters can be selected by means of a set of composition rules. In the latter case a a threshold T_{d_C} for the cluster distance d_C is mandatory: a cluster Ω will be analysed only if it was manually selected a cluster $\Omega_* : d_C(\Omega, \Omega_*) < T_{d_C}$. Before starting the analysis, if it is the first time that some user categories have been selected, the administrator should provide a training set of human-driven associations of users and categories, so that the mapping user-cluster is later on possible on a neural network. The proposed distributed software architecture mainly consists of the following components and resources (see Figure 7.7).

- The data collector on a storage node holding row data
- A resource and queue manager (RQM) holding the state of each virtual machine

available from the cloud

- Several skimming nodes (SN), each building a RBPNN, by creating and training it using a category
- Several computing nodes (CN) analysing data according to an existing RBPNN
- A repository for the resulting data elaborated by CNs

Cloud communication is managed by using specialised components (named `COM_` in Figure 7.7). Moreover, management services (`SRV_`) and application tasks (`APP_`) are separate and independent from each other. The initialisation phase is as follows. Firstly, RQM allocates virtual machines to select the social network data according to the rules set up by the administrator. When a graph distance was provided for clustering purposes then users are separated into clusters with $d_G < T_{d_G} >$, otherwise if a cluster distance was provided, only clusters with $d_C < T_{d_C} >$ are formed. When such an analysis ends, then the data are gathered and sent to RQM. Component `SRV_CloudSrv` within RQM handles the requests of task allocation into cloud resources, the number of resources required is according to the number of users and clusters to analyse. Then the distribution of data is performed by means of component `APP_DataCrt`. Each different collection of features or categories corresponds to a RBPNN topology. This means that each RBPNN has to be trained separately and that its analysis differs from other analyses on different RBPNNs. Accordingly, for each RBPNN topology RQM selects a SN among available ones. Component `APP_CloudSize` within RQM is responsible to check the amount of cloud-based resources used and trigger the growth (or decrease) as necessary, hence installing on each SN the required software packages. Each SN creates and initialises its own RBPNN using component `APP_NnetIni`, then the RBPNN is trained using component `APP_NetTrn` and selected data. Once the RBPNN has been trained and validated, then it is possible for SN to request other cloud resources and perform parallel analyses, given that the same classification task is required. The newly allocated resources, named *computing nodes* (CN), have a simpler analysis task and perform faster than

TN, however given the big amount of data these are added on demand. Cloud resources newly available are given a replica of the working RBPNN by an SN. The amount of CNs requested is so that the enqued data on RQM can be quickly processed. Each SN, after the expansion phase works, as a CN, for analysing a chunk of data.Since RQM holds the state of each available node, it determines the allocation between raw data and CNs. Then, data are transferred from the storage nodes to an available CN. The target CN is chosen taking into account that raw data are appropriately tagged, hence the matching RBPNN will be sought. Once data are received, each CN performs two tasks: RBPNN data classification; and the mutual information analysis obtained by computing the Jaccard similarity coefficient and the Hamming distance. The first task is performed by component `APP_NnetExe`; the second task is executed by component `APP_GPUKer`. The latter invokes a massively parallel process on the GPU accelerator that computes $N-1$ indexes and compare each pair of users. Once CN has finished such tasks the output will be sent to a repository, and since CN has become ready to receive data its state update will be sent to RQM. The above sequence of steps will be iterated till data remain or an administrator terminates the tasks. The said analysis produces results that are sent to a database that acts as a repository, which can be accessed by authorised clients. Data are continuously updated, however they can be explored in real time so as to have fresh results for analysed users. As a final remark, the developed distributed analysis is able to continuously receive and analyse data without any intervention from an administrator, once appropriate rules have been defined, hence it perfectly suits for network analysis automata.

7.9 Facebook as a test ground

The proposed RBPNN architecture has been tested using data collected on Facebook. The dataset consists of features and friends lists from Facebook profiles. Data were collected by surveying participants using a Facebook application. The dataset is provided in anonymous form as part of the DARPA project by the Stanford University and the SNAP graph library for the Stanford Large Network Dataset Collection, which

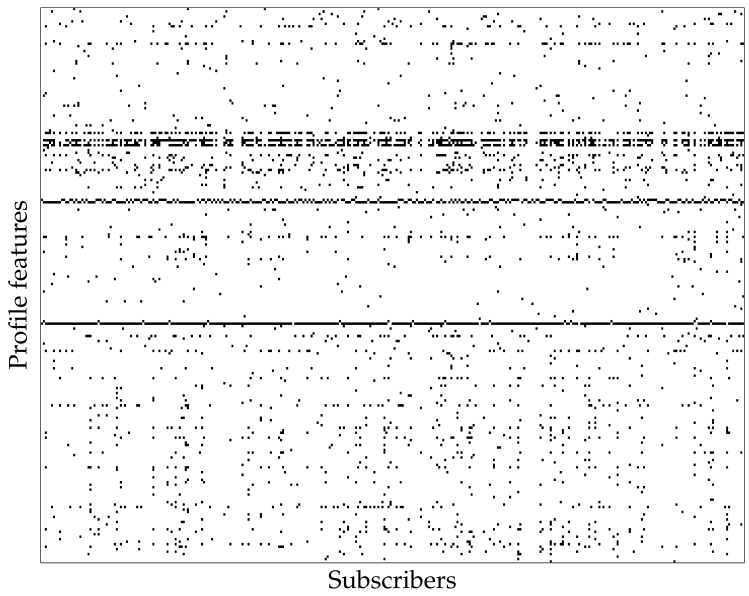

Figure 7.8: The feature list representation for each subscriber. Each row represents a different feature, while columns are the different profiles.

is publicly released [131]. For the provided feature lists, the interpretation of these features has been obscured, however understanding what the features are would have no use for the scope of this work neither for the analysis performed. The dataset used contains all boolean values for the features of each examined profile, group memberships for the user, and his/her friend list. As far as the collected feature lists is concerned (Figure 7.8), the data provide boolean values for the content of the profile for each user. The presence or absence of a specific value is expressed as a boolean flag, e.g. 1 if the user has declared his job or 0 if no job information is given in the profile. Among such boolean values there are also mutually exclusive values such as the gender, e.g. 1 if male or 0 if female. Note that the intrinsic structure of the dataset prevents us from considering only a reduced portion of the feature list for a user. A piece of information is usually largely spread over a certain number of variables, e.g. a boolean variable could express if the gender is stated or not, and only if stated a second variable could report if the user is male or female; then in case a selection of the profile excludes the gender, the second variable has no meaning and can not be taken into account. However, since the dataset is unlabelled, we can not exclude any

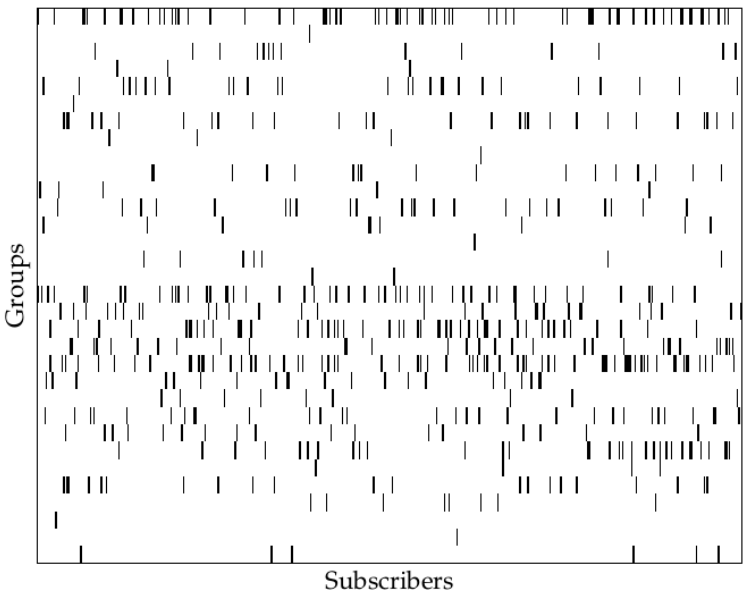

Figure 7.9: The distribution of the examined users for the considered groups

feature. Although data are anonymised, users are identified with a unique ID, and such IDs are used to characterise the friendship links between users, and reported as IDs pairs. On the contrary, the memberships of users to groups is indirectly identified from the list of subscribers to each group. Data that have to be given as input to our neural network for classification purposes, have been passed to a preprocessing stage. In this preprocessing, for each user the relative feature list has been associated to a list of group memberships, this contributes to realise a statistically driven classifier that identifies the main concerns regarding the group chosen by the users. Starting from the features of the profile, the described RBPNN was then used to determine such groups. We report an example for a dataset describing 350 users and 32 groups (Figure 7.9). For this dataset, the profile was characterised using 250 different boolean features (Figure 7.8) that was passed to the RBPNN as input set. Initially, we have asked the network to correctly reconstruct the groups of a set of 250 users. For this, both the user profiles, consisting of the features, and the membership to groups were provided to the network during the training step. Therefore, the RBPNN has learnt how to reproduce the correct paths to associate lists of profile features with groups (Figure 8.3). The RBPNN was able to correctly attribute the subscribers to the

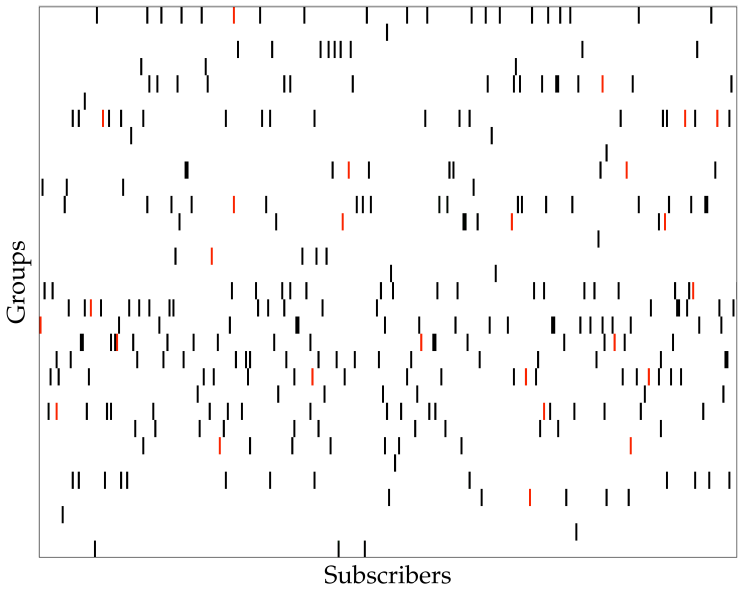

Figure 7.10: Group membership reconstruction made by the RBPNN: in black are the correct associations, while in red the associations that the RBPNN was unable to reproduce.

proper groups with only a 5.67% of missing associations: as a remarkable side effect of such an architecture, while a few groups were not associated, no false positive was given in the associations. Then, after the RBPNN had been trained with the first dataset, we have asked the RBPNN to identify groups for new users. In this case, the RBPNN was unaware of the actual preferences of users for groups. Figure 7.11 shows new users (in black and green) that are proposed for a group they have not expressed preferences in. For an appreciable percentage of total users (shown in green), i.e. about 20%, the proposed RBPNN has indicated for them a group that corresponds to one of the existing groups (unknown to the RBPNN) for which users have expressed actual membership. Again, no false positive was given as a result of the RBPNN analysis.

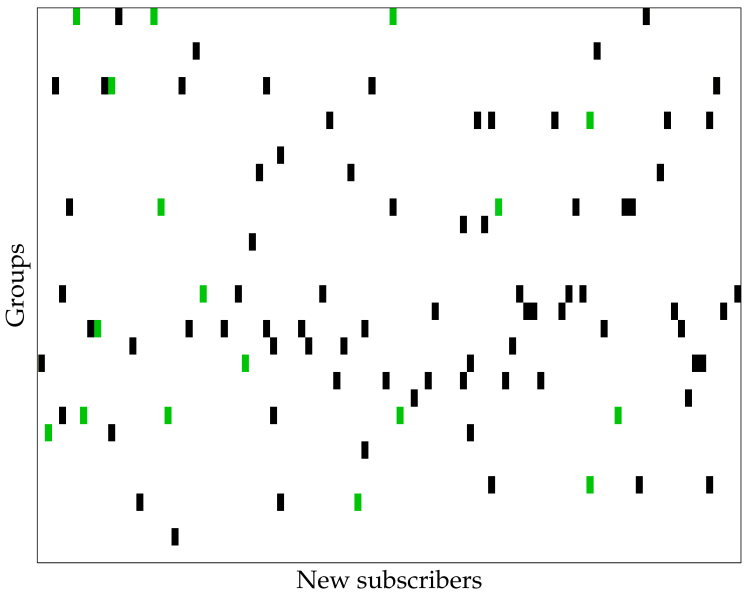

Figure 7.11: An heuristic assignment to groups for subscribers unknown to the RBPNN: in green are the subscribers identified as belonging to legitimate categories (according to the groups they belong to) proposed by the RBPNN.

7.10 Comprehensive identities

Categories have been chosen by the RBPNN classifier alone, which is statistically driven and such categories have a probabilistic meaning that contributes to identify the most appropriate conceivable model for users. The so called 'model' should be intended as a kind of representation of the behaviour of a user on the social network. The identified category, provided by the RBPNN, can complement and integrate the online identity of each subscriber. Such a comprehensive identity, assigned automatically to users, can help further understand the behaviour of users and given as an alert to the social network administrator for checking whether users are honest and reliable, or whether their profiles match more with misbehaving ones. When a user posts a content or subscribes to a group, then the social network administration will be alerted and aware of the implicit or explicit choices made by that user. A part of such data, once given to the RBPNN, can then be used to identify specific categories of users, and eventually such categories would depend only on past choices

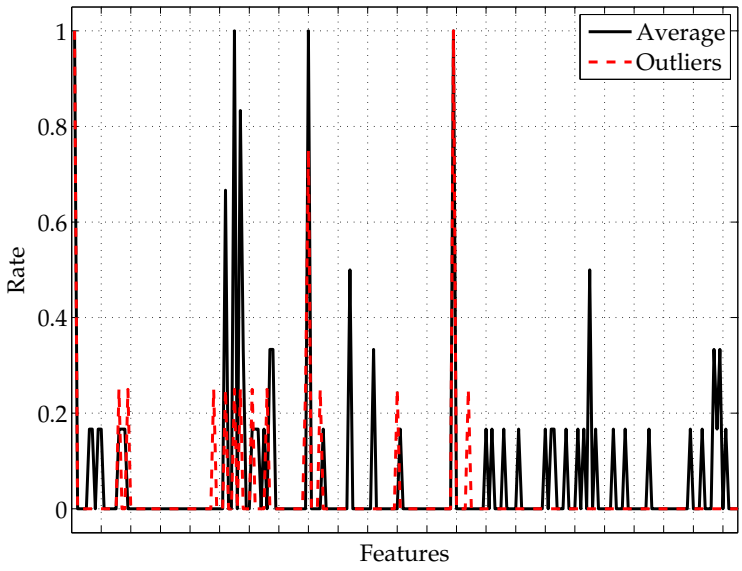

Figure 7.12: Feature rates for users of the same group (black). The RBPNN could not classify in such a group some users with a very different feature set (red).

and activities of the user, rather than on tentative categories identified a priori. We want to point out that the results of the RBPNN can be intended as an extension of the user identity. That extension is useful especially when no categorisation can be performed. Considering Figure 8.3, for some users the RBPNN was unable to provide a categorisation (red spots), however, on the other hand, if we compare the set of features for such uncategorised (unclassified) users and the average feature set of the group where they belong, relevant differences in their feature set can be uncovered with respect to the average (and correctly categorised) user of the group (see Figure 7.12). Eventually, just for validation purposes, we have performed the same comparison for users with an almost empty profile that the RBPNN could not insert into any cluster. Such users had not provided a significative feature set and then the RBPNN could not be able to find any statistical affinity with any of the categories used for the training. Figure 7.13 shows the different values of the features for an average user on a group (in black) and an unclassified one (in red). Theoretically, the RBPNN unveils a behavioural pattern that the user is expected to follow. If a user begins to act according to a different behaviour with respect to those for which s/he

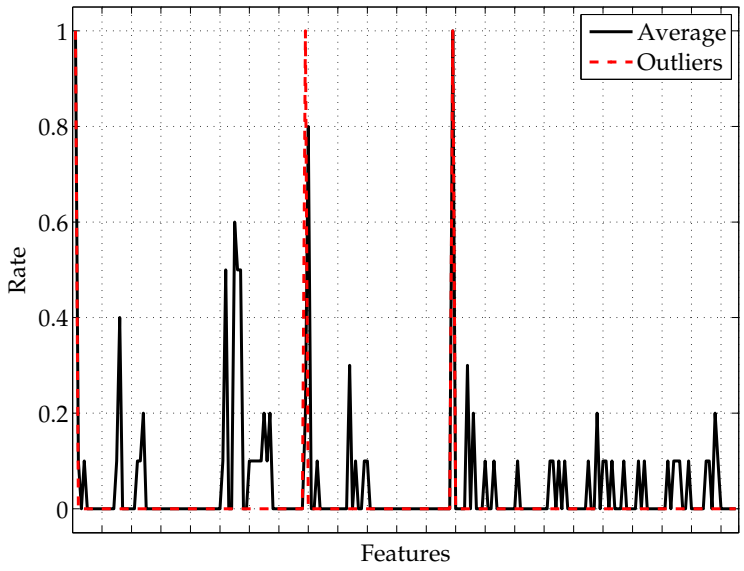

Figure 7.13: A test made with an almost empty profile which the RBPNN could not classify. The feature rates for a average user of a selected group are in black and the features for the almost empty profile are in red.

has been classified by the RBPNN, then this variation can be used as an alert that enables an administrator to monitor him/her and possibly apply some restrictions after a deeper check has occurred. E.g. if a bad behaving user takes over the account of a different legitimate subscriber, then an alert would arise as soon as the identity thief subscribes to groups or posts contents that contrast with the habits of the original user. The alerts arising from the RBPNN can be a notice that is sent to administrators of the social network, then further checks of the profile and the current usage is expected. Once a user account has been detected as compromised, or belonging to a bad behaving user, the automatic detection system can even be set to rise a warning toward all the users that are the target of the activities of the perpetrator, in order to possibly avoid tentative deceptions. The RBPNN can be used to avoid autogenous threats, such as a misbehaving user or a thief, as much as a wide range of other online frauds and several violations. The more online behaviour are modeled, by training the RBPNN with existing user data, the more positive and negative activities can be identified for other users. Although in this work the RBPNN has been used to

assign categories, which correspond to groups, the term 'categories' has been used for its more general meaning. The RBPNN is able to find and propose non explicit groups, i.e. groups that have not yet been chosen by a subscriber, but which are very likely to be eventually chosen since they match the preferences of the subscriber. In a similar way, this RBPNN can be arranged to select users having an high affinity toward a group. Hence, the RBPNN can be asked to unveil the affinity of a user with a certain category of users, who exhibit a common behaviour. Let us suppose that a user is inclined toward a bad behaviour, then the RBPNN would associate such a user with a category previously built by administrators, consisting of other misbehaving users. For building such a category, administrators would simply need to manually flag some selected users. Such a category identification would be expected to be silent, i.e. without any interaction or explicit advice toward users. Then, when e.g. a user is found to post inappropriate contents, s/he will be selected to belong to a pertinent category of users, which has been built to label users performing similar activities. Such a category can then become one among the categories identified by the RBPNN, and can be used for the early identification of users that could eventually act in the same way. Such a control system, although in some moments would depend on human activities (i.e. administrators), can then be used automatically to restrict, when urgent actions are needed, a small number of possibly dangerous users. This automatic selection of users would avert the risk of having to restrain the entirety of subscribers. The proposed a solution for automatic analysis of data collected on a social network and have shown that interesting results can be found when gathering sparse data related to the behaviour of users. The proposed solution is based on both similarity coefficients and RBPNNs in order to find for a user the most similar group/category s/he could belong to when considering user declared data (features) and their behaviour collected from their activities. Moreover, we have shown a parallel solution that thanks to cloud-based and GPU-equipped resources can handle massive amount of data, which are continously produced by users. The key strategies for having a high degree of parallelism are the ability to run replicas of trained RBPNN, and the largely independent threads running on GPUs without need to synchronise. Both strategies allows us to drastically reduce computing time

and hence afford an unprecedented analyses. The proposed solution can be integrated with the servers handling data of user on a social network. This would provide higher security levels as misbehaving and deceiving users could be timely detected. The safety of subscribers would then be preserved, e.g. by timely warning administrator to intervene to check and stop autogenous threats.

7.11 The inanimate reasoner

Of course the results obtained with the proposed RBPNN classifier are very interesting when applied to the field of online social networks, also for security and protection purposes. On the other hand, as it is said at the beginning of this chapter, we want to obtain accurate behavioral model for the workers of a crowdsourcing project, moreover, as said in the previous chapter, we wanted to use such a technology in order to model human resources as on a cloud. Everything is now ready, after a very long series of concepts, from chaos theory, to wavelet analysis, from neural networks, to second generation wavelets, from wavelet recurrent neural networks, to information theory and, finally, from workflow models to RBPNN classifiers. Now we are able to mix all those concepts to create a properly said inanimate reasoner. An Artificial Intelligence used to manage people, groups, companies, in order to improve what we know as crowdsourcing model, the artificial artificial intelligence, as it is called by Amazon. Since the author of this thesis does not believe that humans can be subjected to a mere emulation of an artificial intelligence, therefore completely rejecting the definition of Artificial Artificial Intelligence, it seems reasonable, then, to name the upcoming model an Artificial Artificial Artificial Intelligence or A^3I.

CHAPTER 8

A^3I: Artificial A^2I

> The only source of knowledge is experience.
>
> Albert Einstein

The introduced artificial intelligence methods for obtaining user classifiers can be additionally used in order analyze employees to form work groups. This can be done by means of a solution that analyses workers by using data gathered from their professional attitudes and skills, then suggests how to form groups of human resources within a company that can effectively work together, as well as for an online cooperative service such as the yet encountered crowd-sourcing oriented projects (e.g. the Amazon's Mechanical Turk). The same proposed tool provides employers, workers or crowd-sourcing projects' back-end users, with a fair and effective means for employee evaluation. In our approach, employee profiles are processed by a the Radial Basis Probabilistic Neural Network based classifier, which is able to find non-explicit custom-created groups. The accuracy of the classifier is very high, revealing the potential efficacy of the proposed classification system. The dedicated classifier helps us to create groups of workers automatically extrapolating a model that can easily be implemented in public administrations or companies, consisting of collaborative networks of people who share a common vision and goal. Moreover, since often novel

groups of employees coming from several branches have to be created in order to achieve some new goals, or for special tasks, we propose an automatic system that unveils professional affinities among employees, in order to create efficient teams. The found 'non-explicit' groups of employees, who possibly work on different areas or branches, could form a successful team because of the similarities underlying some of their attributes, such as interest, skill, competence, etc. What we tried to do is to emulate a decision making process by means of mutual information theory indexes and Radial Basis Probabilistic Neural Networks (RBNN) in order to unveil the underlying affinities in human groups of employees. Our solution takes advantage of the well known classification, clustering and generalization capabilities of the Radial Basis Probabilistic Neural Networks which from sparse datasets creates a model of the input sets even for partial input data [67]. Moreover, RBNNs can be continuously trained to recognize novel features, hence can easily cope with a changing dataset, as previously shown in [23]. Let us now present the collaborative group theory, our relation model and the classifier.

8.1 A network of collaborations

In general, it is possible to represent a collaborative network as a graph describing the collaborative relations among the agents represented as nodes in the network. In a work-related collaborative network, nodes are employees and arcs are professional relations among different employees in terms of collaborations, office dependencies or hierarchies involved to perform a task (e.g. employees of an area are connected with their area manager). In a very similar way with respect to the social network scenario, analyzed in the previous chapter, it is possible to represent a collaborative network as a graph describing the collaborative relations among the agents represented as nodes in the network. Collaborative networks, like social networks, follow a *scale-free* behavior (see [16]). A few nodes act as important hubs (i.e. employees who hold key positions or play important roles with major responsibilities). These hubs have a large number of relations with other employees (sometimes in different departments), hence the work of such hubs widely reflects that of collaborative networks.

I.e. generally, the distribution of a complex job handled by the employees in different departments follows a scale-free pattern. Moreover, the emerging *small-world* properties are important characteristics to consider in order to understand the social dynamics involved in the work flow and the related management [6]. Both the scale-free behavior and small-world properties make it difficult to analyze the network with conventional means, e.g. a stationary or analytical model describing job-related area. The analysis is complex because of an uncontrolled growing number of parameters. In order to obtain a fair evaluation of human resources, one can choose affine employees as a sampling cluster, i.e. the aim is to measure the performance of an employee with respect to the mean behavior of other employees. However, a sampling cluster should not trivially map a department or an office, because results would be affected by the mutual interactions of people under evaluation. Therefore, an appropriate employee clustering for a collaborative network becomes paramount for gaining accurate human performance measures. Moreover, successful positioning of new human resources, as well as their relocation or assignment to a different position within a company are critical decisions. Therefore, homogeneous and harmonious work groups that share a common background as well as professional attitudes and complementary skills need to be properly created, possibly, with little effort. Let us now present definitions and mathematical models of the collaborative network, which can be represented as a graph whereby the nodes are the employees and the arcs professional connections like dependencies, collaborations and interactions. We state that *employee* is an element in an employee pool $\mathcal{U}$, which consists of all the employees in the analyzed company. In order to formalize the structure and functions in collaborative networks we need to define the following:

- *employee*: $u_i \in \mathcal{U} = \{\text{employees set}\}$,
- *feature*: $f \in \mathcal{F} = f(\mathcal{U}) = \{f : \mathcal{U} \to \{0, 1\}\}$,
- *relation*: $r \in \mathcal{R} = \{r : \mathcal{U} \times \mathcal{U} \to \{0, 1\}\}$,
- *category*: an equivalence class $\mathscr{C}$ of $\mathcal{U}$ with respect to several features or relations.

It is then possible to define $\mathcal{F}$ as a finite set of *features* related to an employee. If the feature list of an employee u shows a feature f then $f(u) = 1$, otherwise $f(u) = 0$. Examples of features can be the gender (1 if male or 0 if female), the academic degree (1 if achieved or 0 if not), each of several responsibilities (1 if accountable for, otherwise 0), a professional achievement (1 if completed or 0 if still in completion or not compatible with the professional figure), etc. Moreover, it is possible to define a mutual interaction r_f among two employees u_1 and u_2 so that $r_f(u_1, u_2) = 1$ when a professional relation exists (an edge in the collaborative network links u_1 and u_2). In the same way, relation r_g indicates whether two employees are members of the same group (i.e. they are part of an existing team, office, division, etc.). Then, it is possible to define *categories* as classes of equivalence $\mathscr{C}$ among employees according to groups, relations, employee features, or other kinds of provided data. Therefore, categories can be used to model partitions of employees according to their features, skills, groups, area of interests, etc. The collaborative network graph is mathematically defined as $G = (\mathcal{U}, A)$, where the vertexes set is $V = \{v_i \sim u_i \in \mathcal{U}\}$, and the arcs set is

$$A = \{a(v_i, v_j) : r_c(u_i, u_j) + r_g(u_i, u_j) \geqslant 1\}\,, \tag{8.1}$$

whereby $(u_i, u_j) \in \mathcal{U} \times \mathcal{U} \backslash \{u_i\}$ identifies a pair of employees, and where r_c and r_g are two relations. Basing on the given definition of relations, $r_c(u_i, u_j) = 1$ if employees u_i and u_j have professional relations, otherwise $r_c(u_i, u_j) = 0$ if they do not share any professional collaboration. Similarly, $r_g(u_i, u_j) = 1$ if the two employees belong to one team or an explicitly defined area within the company, otherwise $r_g(u_i, u_j) = 0$. If there is a finite number of arcs $a \in A$ that connect two vertexes $v_0, v_n \in V$, then it exists at least one path $P_G(v_0, v_n) = \{a(v_i, v_j)\}_{v_0}^{v_n}$ where $a(v_i, v_j)$ are arcs from v_i to v_j, and the *length* $l_{0,n}$ is given as the cardinality $|P_G(v_0, v_n)|$, when $P_G(v_0, v_n) \neq \varnothing$, that is the number of arcs, from v_0 to v_n, used to form a path. The *graph distance*

$d_G : V \times V \to \mathbb{R}_0^+$, is

$$d_G(v_0, v_n) = \begin{cases} 0 & v_0 = v_n \\ \min\{l_{0,n}\} & P_G(v_0, v_n) \neq \varnothing \\ +\infty & P_G(v_0, v_n) = \varnothing \end{cases} . \tag{8.2}$$

The given definition of arcs and graph distance allows us to consider the existence of two different kinds of relations: when a pair of employees are linked by an arc $a \in A$, indeed the distance is 1, otherwise the distance is the minimum number of hops separating the pair. On the other hand, if the employees belong to the same group, ($r_g = 1$) the distance is cut down to 1. With this definition of distance it is possible to unveil the professional affinity of one employee with other employees. For our model, employees with similar interests and skills should have a very small distance, whereas employees having very high distances perform very different jobs and need not collaborate. The defined distance will be used as an adjunct parameter (other than employee features) for the developed classifier. The feature set $\mathcal{F}$ has to be profiled for each employee. Moreover, $\mathcal{F}$ can be a basis for a Hilbert space where the employees can be classified according to their features. Thanks to the formalism that we have just introduced, we can attribute a natural number to each employee, so that it will be possible to compare employee pairs $u_0, u_n \in \mathcal{U} \subset \mathbb{N}$. For each employee u_i we define a feature vector ϕ_i consisting of boolean values, each identifying the presence or absence of a certain feature for the employee profile

$$\phi_i = [f_1(u_i), f_2(u_i), ..., f_N(u_i)] \ \forall \ f_\alpha \in \mathcal{F}, u_i \in \mathcal{U}. \tag{8.3}$$

Then, it is possible to define a *feature distance*

$$d_{\mathcal{F}}(u_0, u_n) = \frac{1}{\sum_\alpha \delta(f_\alpha(u_0), f_\alpha(u_n))}, \tag{8.4}$$

whereby δ represents the delta of Kronecker and conventionally $\langle \phi_0 | \phi_n \rangle = 0 \Rightarrow d_{\mathcal{F}}(u_0, u_n) = +\infty$. The aim of such a defined distance is to obtain significant and coherent information about the behavioral proximity or affinity among employees in

a given collaborative network. Distances d_G, $d_{\mathcal{F}}$ are then used to pilot the RBPNN classifier in order to create collaborative work groups on demand. In this case, distances and similarity indexes among selected employees are used to qualify affinity when forming a new team.

8.2 RBPNN continuous learning

While this theoretical asset is quite solid, it still has to cope with the versatility of a real company or, even worse, with all the continuous changes that can affect a service such as the Amazon's Mechanical Turk or other crowdsourcing oriented initiatives. In order to give a more suitable dynamic to the implemented machine learning mechanism, we created an agent driven learning system that continuously train the RPBNNs to recognize novel features, hence can easily cope with changing data. The proposed neural network has been embedded into a *Classification Agent* that builds a model out of data coming from workers or back-end user profiles, and handled by other agents, such as a *Profiling Agent* and a *Crawler Agent*, which retain useful data. Specifically, our *Classification Agent*, according to the proposed RBPNN solution, can handle partial data, acting as a modeler for dynamically changing the profiled identities of the workers. For this reason our solution comprises different collaborating agents that make the company administrators able to classify and monitor the workers behaviour, other than enhancing their productivity rearranging them in new workgroups according to their skills. As said above, we used an actor-critic reinforcement learning architecture with Back Propagation Tranining Algorithms (BPTA). The correctness evaluation of the decision in the applied RBPNN is performed with respect to the human-made choices, where ξ is the error function. We consider that this evaluation is performed by a stationary agent (*critic*). It is possible both to use the *critic* in order to filter the effectiveness of RBPNN outputs (a human supervisor who can acknowledge or reject RBPNN suggestions) and to train an *adaptive critic*, which in the long run simulates the decisions of the human *critic* and then diminishes the need for human driven control. The *adaptive critic* needs to learn and this learning process is done by BPTA, which uses ξ as error function. For this reason the *adaptive*

critic can be trained by means of the traditional gradient descent algorithm so that the weight modification Δw_{ij} is

$$\Delta w_{ij} = -\mu \frac{\partial \xi}{\partial w_{ij}} = -\mu \frac{\partial \xi}{\partial \tilde{f}_i} \frac{\partial \tilde{f}_i}{\partial \tilde{u}_i} \frac{\partial \tilde{u}_i}{\partial w_{ij}}, \tag{8.5}$$

where $\tilde{f}_i$ is activation of i-th neuron, μ the learning rate, and $\tilde{u}_i$ is the i-th input to the neurone weighted as

$$\tilde{u}_i = \sum_j w_{ij} \tilde{f}_j(\xi_i). \tag{8.6}$$

The results of the adaptive critic determines whether or not to continue the training of the RBPNN with new data, as well as whether the last training results should be saved or discarded. Figure 8.1 shows the agents for our designed system: a *Crawler Agent* periodically and autonomously gathers user information from their company profiles, other than the list of their activities. After some preprocessing tasks, data are given to the *Classification Agent* that using the inner RBPNN assigns worker profiles to known categories, according to the statistical model built on user information during training phases. Due to the intrinsic dynamics that the social network imposes, this model is constantly and incrementally updated. The classification results, i.e. the associations between profiles and categories, are given to the *Verification Agent*, that asks the *Category Agent* to provide the categories already assigned to a specific user (if any), comparing them with the ones just given from the Classification Agent results. If a specific user had no category assigned, the Verification Agent will notify the Category Agent with the newly one found; if instead the user already had a category assigned, but differing from the one just discovered, we could think of this as a clue for a possible mismatch and should be reported to the administrator for further surveys on the worker and possible reassignment to a different group. This is achieved by giving the profile of the worker to the *Alert Agent*, that constantly handles all the received notifications, timely warning the administrator with the potential problems intercepted. The administrator has also the ability to manually define categories built over the activity information of several selected workers; if one of these categories is assigned to a user profile during classification, the Verification Agent will ask the Alert

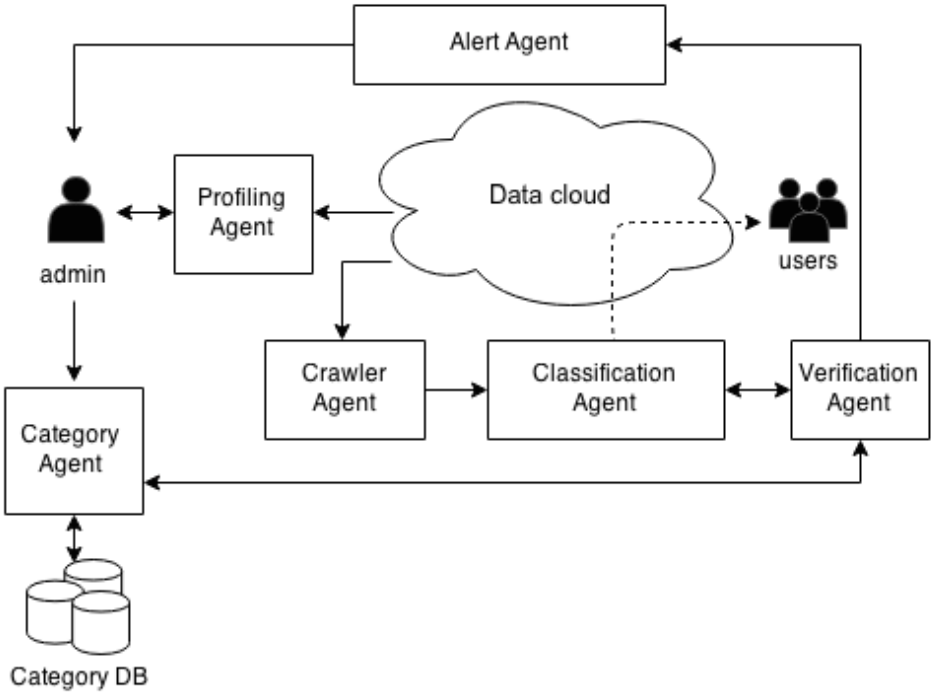

Figure 8.1: Schema of the data flow through the agents of the proposed system

Agent to notify the administrator with the detected issue. The administrator is then able to gather deeper information on the worker activities using the functionalities provided by another agent, the *Profiling Agent*. Using this information, s/he can decide what to do according to company policies. If the worker is considered not compliant with such policies, the administrator has the ability to use the classification information to automatically identify other workers in the company with the same behaviour, asking the Classification Agent to update the inner RBPNN model. On the other hand, if the behaviour of the worker can be considered aligned with his workgroup, then the new classification label can be simply passed to the Category Agent. Since we can see a *group* of a workers as a category of profiles, that gets together people with common goals, we can use the same approach just seen to classify profiles with the workgroups that better suit their professional skills. This type of classification results could be directly provided by the Classification Agent as recommendations for workgroups that workers can join.

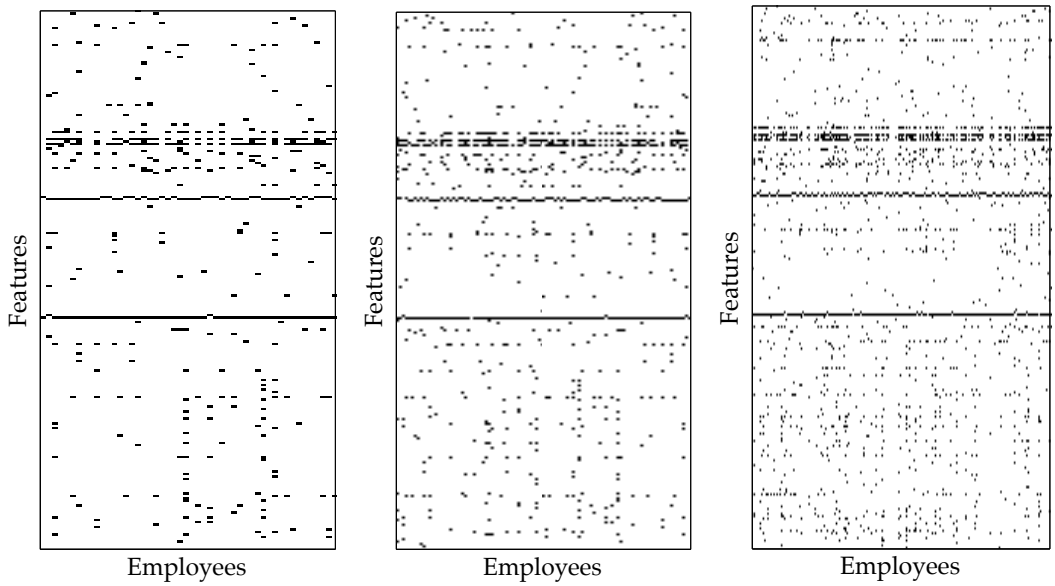

Figure 8.2: The feature list representation for each employee in a company, where each diagram represents a different company. In the charts, each row represents a different profile feature, while columns are the different employees.

8.3 RBPNN driven workgroups

RBPNN classifier has been tested using data collected in anonymous form from public and private companies. Datasets describe up to 200 employees and 32 groups for each one of the analysed companies, for a total of 7 companies. The profiles were characterized using 250 different boolean features that compose coded profile values passed to the RBPNN classifier as input sets. The dataset contains all boolean values for the features of each examined employee, work group memberships for the employee and his/her professional relations within the company. Figure 8.2 shows boolean diagrams representing each company, where employee profiles were coded to test our classifier. The intrinsic structure of the dataset prevents us from considering only a reduced portion of the feature list for an employee: a piece of information is usually largely spread over a certain number of variables (a boolean value expresses the gender, the academic degree, professional achievements, previous projects, etc.). Although data are anonymous, employees have been identified with a unique ID.

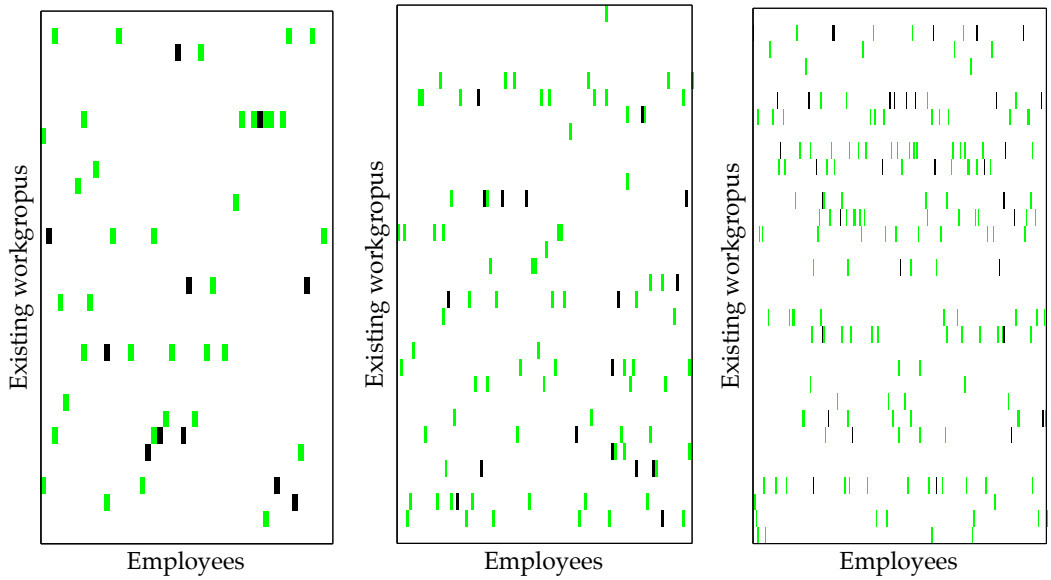

Figure 8.3: Group membership reconstruction, for each of the analyzed companies, performed by our RBPNN: green dots show the correct associations, while black dots give the associations that the RBPNN was unable to reproduce.

These IDs are used to characterize the links among employees themselves and reported by the classifier as IDs pairs. On the other hand, the memberships of employees to work groups are indirectly identified from the list of employees in each group. In RBPNN input data (coded employee profiles) preprocessing, for each employee the relative feature list has been associated to a list of memberships, this contributes to realize a statistically driven classifier that identifies the main concerns regarding the group chosen by the employees. Starting from the profile features, the described RBPNN was then used to determine potentially most efficient collaborative groups for future projects. Initially, we have asked our RBPNN model to correctly reconstruct the groups from the input sets. For this, both the employee profile features and the membership to work groups were provided to the network during the training step. Therefore, the RBPNN has learnt how to reproduce the correct paths of the collaborative network model to associate the lists of profile features with existent work groups. Figure 8.3 shows the results of the associations. Our RBPNN model was able to correctly attribute employees to the proper groups with an error less than

Table 8.1: The RBPNN classifier examinations results for each analysed company.

	analysed employees profiles		
Case study	**50**	**100**	**197**
Total combinations	1600	3200	6304
Correct results	1593	3182	6261
Wrong results	7	13	43
Correct assignments	11	17	34
Unmatched assignments	6	13	27
Missed assignments	1	5	16
Assignments correctness	61.11%	48.57%	44.16%
Overall correctness	**99.56%**	**99.44%**	**99.32%**

1%. As a remarkable side effect of such an architecture, while a few work groups were not assigned, no false positive occurred. Then, after the RBPNN has been trained with the first dataset, we have asked it to assign employees into new work groups. In this case, RBPNN results were successively compared to assignments made by a human employer, unaware of the RBPNN results. Finally, Table 8.1 summarises the main statistical values, which show that the proposed RBPNN classifier can efficiently build collaborative work groups for various companies. The table orderly reports three case studies, respectively with 50, 100 and 197 analysed employees, and the related number of total employee-group combinations, number of correctly given results on those combinations and the overall ratio of correctness. Moreover, along with such information also the number of correct assignments is reported, as well as the number of missed or unmatched assignments. The classification results are very promising and show the power of our presented solution. In conclusion, the classifier is statistically driven by means of categories with a probabilistic meaning and contributes to identify

the most appropriate conceivable model to have affinity-oriented work groups. The RBPNN classifier can be continuously trained to reflect the changes that affect the company while time goes on, using new employee profiles. New features and data can be continuously fed into the RBPNN, hence possible work group suggestions can be either confirmed, changed or withdrawn by the RBPNN according to recent activities. Thus, the presented solution can perform a refined analysis and advices can be given at any time. The proposed classifier could be integrated with the company servers handling employees data, by providing the employers with a useful tool for preemptive work group testing and hypothetical simulations of new human resource assets.

8.4 Smart workflows

Since the proposed RBPNN should assist companies to form work groups, it seems natural to apply the same concept to crowdsourcing. In particular, knowledge on the back-end users availability gives us a picture, changing over time, of an organizable asset. Each step on a workflow can then be organized using the crowd as we would do with a company, then, iterating each time step we can think to organize and manage an entire crowdsourcing project. On the other hand to reach such a level of optimization, it is necessary to organize each worker by representing his/her work as one of the services of a complex workflow for the execution of the crowdsourced project. As said before, workflows have become a recurrent solution to organise the execution of services assisting the operations of a large organisation, and cloud computing is the norm to rent computing resources. If it works on cloud environments then, since we are able to similarly model the crowd, we can apply the same concepts for the deploying and execution of workflows on a crowd-based infrastructure. The developed system, in fact, provides the needed support for deploying, executing, and monitoring the jobs handled by the crowd. The proposed solution, while being independent of the specific workflows, has shown that an additional description is paramount to determine e.g. the life-time of services executed by the crowd, or whether their execution has to be performed by one particular worker or not. Moreover, our suggested components realise all the interconnection work needed to let a workflow be executed

by the crowd. In our solution, several components have been specifically devised in order to minimise the possible overhead affecting some operations, such as e.g. hiring workers, split them into workgroups, etc. Such components also model the incoming workflows and their execution time in order to predict future workload. Then, we can start management operations on crowd resources in advance, hence shunning delays, and also avoid overheads, hence emerging as a cost-effective solution. Since, actually, it would not be possible to verify the approach with the help of a real crowdsourcing provider (e.g. such as Amazon), we then conducted our experimental campaigns by replicating the effect of the crowd by simulating it on a computational cloud. Therefore, a set of cloud-related services have been developed keeping in mind that they could be either executed on cloud nodes, or be real tasks performed by human workers, since the model does not depend on their implementation. Our model essentially depends on the execution time and availability of nodes/workers. In an example of a elementtary workflow, can involve citizens that are able to actively cooperate with the institutions to make the urban environment properly function by notifying the need for an improvement, due e.g. to failures on a road. In this workflow several services (activities) are executed, where each corresponds to a step that can be performed, starting from a user notification of an urban issue. In our reference model for the software system assisting such steps we will have one or more client applications enabling the user to submit a notification, and wait for a reply. Hence, e.g. the report urban issue step could be performed using a dedicated smartphone app that lets the user send detailed data about the urban issue, together with photos and GPS position automatically gathered. Services on the server side are processes running, or started, according to the indication given by the workflow description, hence e.g. receive requests is the first step of an ad-hoc workflow, and is a process listening for incoming requests, residing inside a persistent web service; and filter redundant requests is a process started as a second step of the workflow once the previous step has been performed, etc. Since each service needed for a workflow completion in general can have its own preconditions, data, and processing requirements, then each service should be handled ad-hoc to provide a proper quality of service. Let us suppose that filter redundant requests is a CPU-bound process whose execution time has to be guaranteed,

because, e.g. it could involve an image processing task to recognise, starting from the photo sent by the user, the type of issue it refers to, avoiding to notify it twice. Instead, another service could simply provide immutable stored documents. Then, handling requests that trigger service execution requires the provisioning of ad-hoc computing resources to ensure the quality of service. To let services be handled properly, each workflow activity has to be tagged with some details that are specifically crafted in order to let the managment component provide the service with the right corresponding resources during execution. We will see in the following how the flexibility of cloud environment can satisfy such requirements. In the above example (see Figure 6.6), the steps of a given workflow are associated with some indications that allow resource allocations. In our solution, a component, dubbed Workflow Scheduler, takes as input the above data and handles at runtime the current request. Typically in our scenario, a new instance of a workflow is started by means of a request coming from a Web Service (see the following section), then Workflow Scheduler is alerted, finds the corresponding workflow, and starts operations on cloud resources for the services within the workflow, as well as the services. In the above example, after the execution of service receive requests on a web server, Workflow Scheduler receives an alert, finds the related workflow and prepares resources for the execution of service filter redundant requests. We can see how each service is characterised with a start and an end status, together with a description of the cloud resource type: the start status is the condition in which we expect to find the service execution environment, such as already active, waiting for new process submissions (on); that has to be created and then turned on (off); that was already created but now it is in a standby state, in order to reduce resource consumption while reducing response time avoiding the creation of a dedicated service. If we specify that dedicated resources are needed, we will assign a separate and dedicate service for the execution of the corresponding task, while indicating that only shared resources are needed (useful e.g. to handle simple tasks like querying or updating a database or retrieving a document from the object storage) then we will run a separate process inside a shared service, that is the process execution environment. Not all the services need cloud resources; this is the case of the web application waiting for user requests (receive requests), and then triggering an

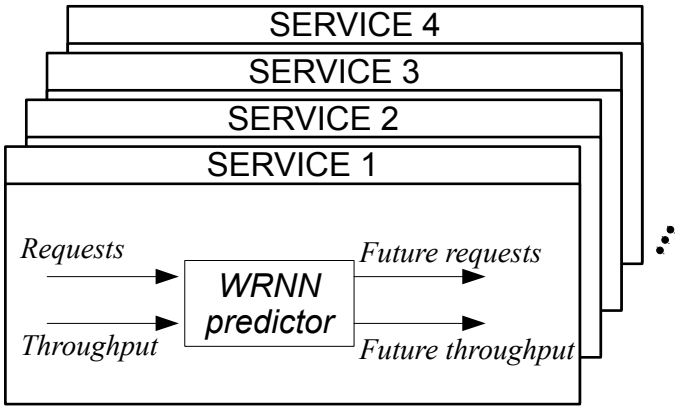

Figure 8.4: The adopted WRNN predictor models and predicts the future trends regarding the number of requests to a service and the related throughput. A predictor is specialised for one service, hence a set of predictors ae used.

actual workflow instantiation; in such cases, we will mark the related activities as a no cloud service. Together with the activity data described above, we have to provide additional data related to the whole workflow, such as e.g. the priority, the allowed deadline and the number of instances that can be concurrently executed. Moreover, for each service (regardless of the involving workflow), the number of its instances that can be concurrently active is also given. This is an important information to properly manage the constraints that must be applied for the concurrent execution of different instances of the same service inside the cloud. Even if we can think of a cloud as a limitless resource provider, we are forced to trade with the *pay-per-use* model to access resources. Therefore, the considered resource constraints are needed to limit the costs related to resources. The proposed workflow management system takes care of the current requests and the related workload. On the other hand, resource allocation is traditionally a contingency topic. In order to perfect allocation startegies, the knowledge of the actual requests is not enough since to make the right allocation choices it would be helpful to predict the future workload. The module for this purpose was called Workload Predictor, and its aim is to predict the future workload of the system in terms of *number of incoming requests for a service* and the related *service throughput*. Therefore, the number of active requests serviced in the

future and the estimated completion time can be easily computed. Prediction is based on the time series of service requests and throughput, observed by the said Workflow Manager. By considering the future expected workload a fine-tuned management of resources can be achieved, hence avoiding both server-side overloading and over-provisioning. Without a prediction system, resources are usually managed by load balancing and admission control strategies. Generally, the benefits of such strategies decrease with the workload increase. Moreover, when the amount of requests overcome available resources, service availability worsening or denial of service cannot be avoided. On the contrary, to obtain an effective and stable level of availability server replicas are needed. However, since the amount of required resources is unknown in advance, it often results in over-provisioning and possibly over-design, with negative effects on management. As anticipated, a Wavelet Recurrent Neural Network is the basis of our proposed Workload Predictor. This component predicts over time the amount of incoming requests estimating also the throughput of the resources related to each service and therefore the service workload. The prediction are then used by Workflow Manager to allocate resources according to the predicted workload. The predictor is used to model two different characteristics of the workflow execution: on one hand it estimates the number of incoming requests, and on the other hand it estimates when the service will be available again to process new requests. Moreover, the module here presented is able to specialise a neural network to make predictions related to one selected service (see Fig. 8.4). By specialising a neural network for each service provided on the cloud, it is possible to obtain a set of predictions that precisely map the status of the cloud-oriented services. In order to operate with the WRNN the workload predictor transforms the said time series in the wavelet domain. The wavelet transform permits us to reduce data redundancies and obtain a representation that can express the intrinsic structure far more precisely than traditional analysis methods (e.g. Fourier transform). While the wavelet analysis exposes the time-frequency signature of the time series on different scales, the WRNN topology is the perfect complement to model the complexity of non-linear data correlational and perform data prediction on different scales. Thereby, a relatively accurate forecast of the request number and throughput time series can be achieved even when load

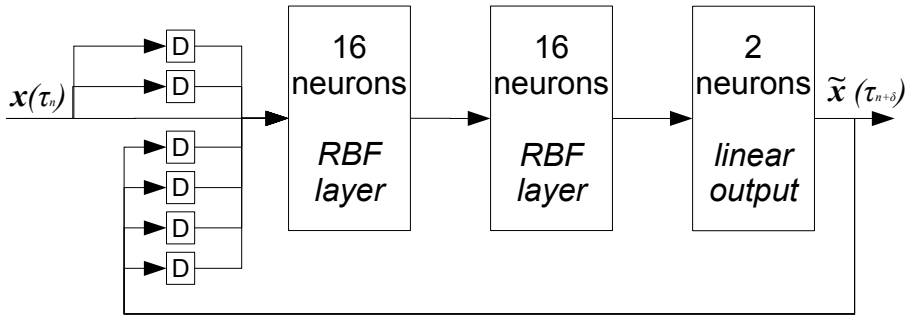

Figure 8.5: Devised neural network. Delays and feedback are obtained by using the relative delay lines and operators (D).

peaks arise. The proposed WRNN consists of an input layer of 8 neurons, two hidden layers of 16 neurons with a radial basis activation function, a linear output layer with two neurons, and two delay lines in input as well as 4 feedback delay lines from the output (see Fig. 8.5). The neural network is fed with the data constituted by time steps of a time series representing service requests and throughput. Using a discrete time index τ we can call $q^{\mu}(\tau)$ the number of requests received at a time τ for a certain service μ, and $r^{\mu}(\tau)$ the throughput reached at the same time by that service. By applying the wavelet transform to the time series $q^{\mu}(\tau)$ and $r^{\mu}(\tau)$ we obtain the related representation in the wavelet space. Since we used a 5 level transform then it follows that

$$\begin{aligned} q^{\mu}(\tau) &\xrightarrow{\hat{W}} \left[q^{\mu}_{a_0}(\tau), q^{\mu}_{d_0}(\tau), q^{\mu}_{d_2}(\tau), q^{\mu}_{d_3}(\tau), q^{\mu}_{d_4}(\tau)\right] \\ r^{\mu}(\tau) &\xrightarrow{\hat{W}} \left[r^{\mu}_{a_0}(\tau), r^{\mu}_{d_0}(\tau), r^{\mu}_{d_2}(\tau), r^{\mu}_{d_3}(\tau), r^{\mu}_{d_4}(\tau)\right] \end{aligned} \tag{8.7}$$

where the arrows represent the transform operation, $\hat{W}$ represents the biorthogonal wavelet decomposition and the resulting vectors are made of the components on the different decomposition scales (from scale 4 to scale 0, where the letter d indicates the wavelet coefficients and a_0 the residuals on the most gross scale). For reasons related to the noise signature we are not interested in the last component of the transformed series, therefore we are interested in an input vector $\mathbf{x}^{\mu}(\tau)$ in the form of

$$\left[q^{\mu}_{a_0}(\tau), q^{\mu}_{d_0}(\tau), q^{\mu}_{d_2}(\tau), q^{\mu}_{d_3}(\tau), r^{\mu}_{a_0}(\tau), r^{\mu}_{d_0}(\tau), r^{\mu}_{d_2}(\tau), r^{\mu}_{d_3}(\tau)\right] \tag{8.8}$$

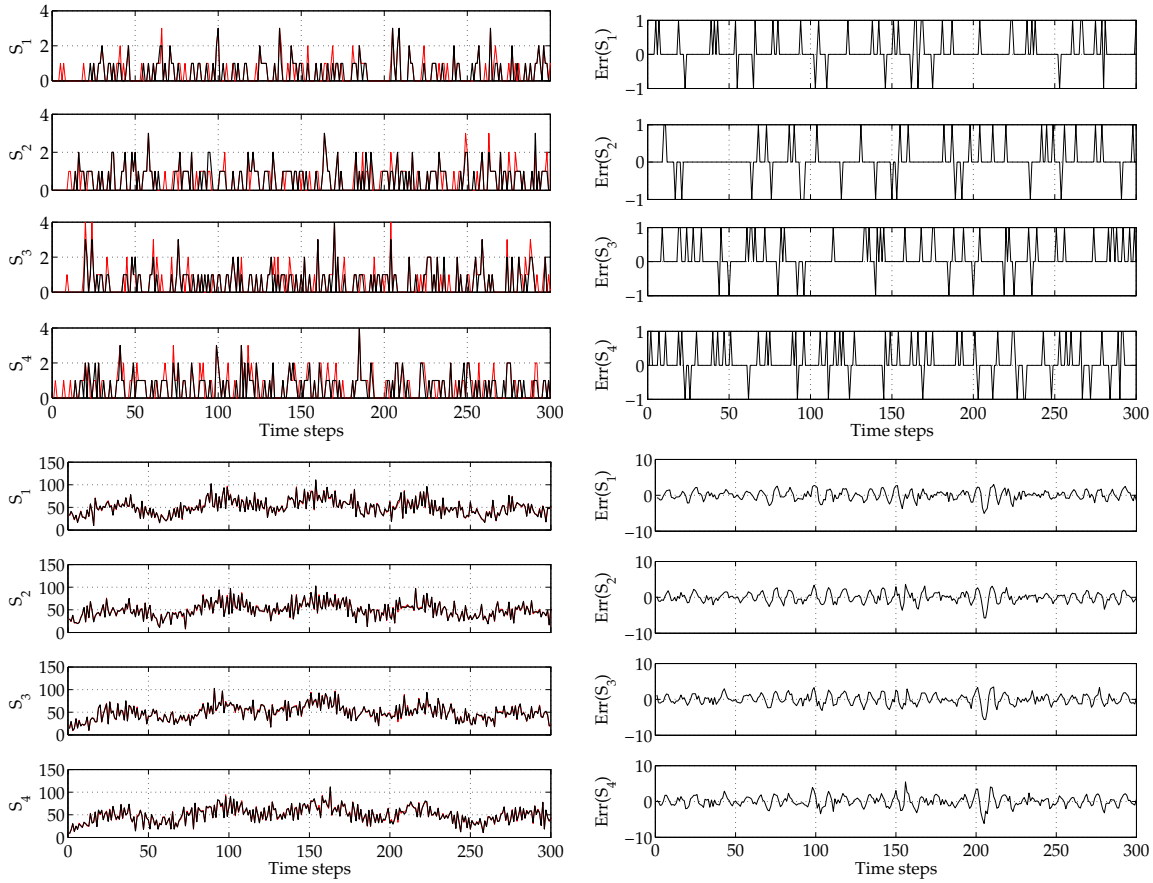

Figure 8.6: From left to right and top to bottom: the predicted (red) and measured (black) throughput, the errors on the throughput predictions, the predicted and measured number of requests, the errors on the requests predictions.

The overall input set, considering N time steps, can be then represented as a $N \times 8$ matrix where the i-th row represents the i-th time step. Each row of this dataset is given as input value to the 8 input neurons of the proposed WRNN. The properties of this network make it possible, starting from an input at a time step τ_n, to predict the number of requests and throughput at a time step $\tau_{n+\delta}$. In this way the WRNN acts like a function $\hat{N}_\mu$ so that

$$\tilde{\mathbf{x}}^\mu(\tau_{n+\delta}) = \hat{N}_\mu[\mathbf{x}^\mu(\tau_n)] \tag{8.9}$$

where δ is the number of time steps of forecast in the future, and the tilde on the symbol $\tilde{\mathbf{x}}$ indicates that it is a prediction instead of a measurement. The number δ is not specified in the equation since it depends on the sampling frequency of the input and output time series. In our work, the data had a sampling period of 10 minutes, while we predicted the incoming service requests and throughput 2 hours ahead. To model the time series and then to predict their future evolution the neural network is firstly trained with the historical time series, several training epochs are interleaved with the related supervised pruning procedure. When the process in concluded, a network starts to provide forecasts related to a specific service for which it was trained. By collecting each service forecast we obtain a complete map $\tilde{S}(\tau_{n+\delta})$, predicted beforehand, constituted as

$$\tilde{S}(\tau_{n+\delta}) = \{\tilde{\mathbf{x}}^{\mu}(\tau_{n+\delta}) \ \ \forall \ \mu\} \, . \tag{8.10}$$

The results of the WRNN predictor are shown in Fig. 8.6. The predictor was able to propose an early estimate of the future service throughput with a maximum error of one service with respect to the successively measured throughput. Also the number of requests for each service was accurately predicted in advance with a minimum error (less than 8 requests). In our experiments the WRNN predictors have been used in order to schedule resources and manage the execution of services, and such operations are made possible thanks to predicted values, due to their availability in advance. Moreover, since each service has been associated to a WRNN predictor, the proposed solution is general for any number of production services constituting any number of workflows. Finally due to the paralelisation technology, the system could be expanded and scaled on demand.

8.5 Dear Jeff

Until now, all the presented approaches ended up in several main branches of solutions to:

- Model and predict resources availability

- Model and manage human work groups
- Model and optimize workflow execution
- Model and enhance crowdsourcing processes

In the latter point we refer to the promise to enhance crowd sourcing processes like the Amazon's Mechanical Turk. Actually, the last paragraph shows how an accurate workflow execution optimization can effectively end in a real economical benefit for the company. In order to well organize those workflows of course workgroups of people must be created, and in this scenario the RBPNN classifier, when jointly used with the WRNN technology, can do the trick. But is it only about production efficiency and resource cost? In December 2014, Kristy Milland, a 35-year old Canadian, wrote an email to Jeff Bezos, founder and CEO of Amazon. Kristy Milland is one of the back end users of the Amazon's Mechanical Turk, one of the resource, one of the real workers behind the so called Artificial Artificial Intelligence. After having completed more than 830.000 tasks on the Mechanical Turk in nine years earning an average of 20 cents of canadian dollars for each, she decided to write to Jeff in order to make him ≪realise that there are living, breathing human beings who rely on this service he provides to feed and shelter themselves and their families≫. With the strong words ≪I am a human being, not an algorithm≫ Kristy Milland pointed out that there is not a minimum income for the tasks performed on the Mechanical Turk and very few warranties for the workers. Since that moment a media campaign started, and several thousands of people decided to write similar "Christmas letters". The campaign was named Dear Jeff, for obvious reasons. The intent of the letters, as the "Turkers" state on their website, is to affirm that Turkers are not only actual human beings, but people who deserve respect, fair treatment and open communication; that Turkers are human beings, not algorithms, and should be marketed accordingly; that Turkers should not be sold as cheap labour, but instead skilled, flexible labour which needs to be respected. The sentiment has not been ignored by the scientific community. In [78] Gaikwad writes that while crowdsourcing marketplaces provide opportunities for autonomous and collaborative professional work as well as social engagement, on the other hand, in these marketplaces, workers feel disrespected due to unreasonable

rejections and low payments, whereas requesters do not trust the results they receive. The lack of trust and uneven distribution of power among workers and requesters have raised serious concerns about sustainability of these marketplaces.

8.6 Artificial^3 Intelligence

The major crowd-sourcing services providers, such as Amazon's Mechanical Turk, are maybe conceiving the human resources in an unsatisfactory fashion, maybe with no particular attention to their skills, capabilities, their possibility to be grouped in work groups basing on such characteristics, the professional relevance of their figures. While the Amazon's services are called The Mechanical Turk, the Turkers are called by Amazon itself an Artificial Artificial Intelligence: humans that artificially contributes to the operation of an artificial intelligence-like system. Is that so? For the author of this thesis the definition, in this interpretation, is not satisfactory, therefore a better kind of crowdsourcing model must be reached. Is it possible to challenge this definition of crowdsourcing and, possibly, completely change the approach? Despite the criticism, projects like the Mechanical Turk show an extreme flexibility and are prone to changes. So it seems reasonable to introduce in the Mechanical Turk, or in any other crowd sourcing project, a hiring component that recognizes the abilities of the workers, their skills their experiences. It can be done by using both the WRNN and RBPNN approaches jointly to our workflow models. As a matter of fact, an expert worker can easily be recognized as a supervisor, and therefore, be entitled of the execution a specific control job, while other users, not expert enough, can be entitled of low-level jobs. Also the payment can reflect the positioning of the workers in a pyramidal fashion, hence resembling the structure of conventional companies. The only problem in doing so, also by taking advantage of our workflow models, is that, differently from a standard company, a crowdsourcing context does not have the certainty of the number of workers in a certain moment of time, as well as their availability or possibility to work for the same project. Actually, a job is performed on such a platform by randomly creating, on-demand, groups of people sharing no common ground or knowledge and assigning them an elementary portion of a complex

task. This can now change. Workers availability can be predicted by means of the WRNNs as shown, and workgroups can be created by means of the RBPNNs, moreover, by joining the WRNN approach with the needed workflow model, jobs can be differentiated basing on competence, required skill, throughput of the worker, experience. Finally, the entire workflow can be predicted and optimized in this fashion by means of the technology developed and presented in this thesis.

CHAPTER 9

Conclusions

> Science is the best achievement of the human mind. Human mind will be the best achievement of science.
>
> Christian Napoli

In this thesis we hope to have fairly kept up with the promise contained in the title itslef. We invented new neural architectures taking advantage of quite complex mathematical tools and complicated portions of software in order to scratch the surface of an enormous field of study. By means of wavelet analysis, neural networks, and the results of our own creations, namely the wavelet recurrent neural networks and the radial basis probabilistic neural networks, used in the described fashion, we tried to better understand, model and cope with the human behavior itself. And as far as it is true that the first idea was to model the workers of a crowdsourcing project as nodes on a cloud-computing system, we also hope to have exceeded the limits of such a definition. We hope to have opened a door on new possibilities to model the behavior of socially interconnected groups of people cooperating for the execution of a common task. We showed how it is possible to use the Wavelet Recurrent Neural Networks to model a quite complex thing such as the availability of resources on an

online service or a computational cloud, then we showed that, similarly, the availability of crowd workers can be modeled, as well as the execution time of tasks performed by crowd workers. Doing that we created a tool to tamper with the timeline, hence allowing us to obtain predictions regarding the status of the crowd in terms of available workers and executed workflows. Moreover, with our inanimate reasoner based on the developed Radial Basis Probabilistic Neural Networks, firstly applied to social networks, then applied to living companies, we also understood how to model and manage cooperative networks in terms of workgroups creation and optimization. We have done that by automatically interpreting worker profiles, then automatically extrapolating and interpreting the relevant information among hundreds of features for each worker in order to create workgroups based on their skills, professional attitudes, experience, etc. Finally, also thanks to the suggestions of prof. Michael Bernstein of the Stanford University, we simply proposed to connect the developed automata. The result was called an Artificial^3 intelligence. We decided for this name due to the misconceptions we believe was introduced with crowdsourcing projects like the Amazon's Mechanical Turk. Despite the Amazon's definition of artificial artificial intelligence for their workers, we believe that humans cannot be resembled as pieces of software or technology. Therefore while representing a third level artificial intelligence, the title of this thesis also wants to challenge the original definition given by Amazon which implies it as being artificial. As for the first and deeper meaning of the title, we hope that the concept has been well unfolded in the previous chapters. We made use of artificial intelligence to model the availability of human resources, but then we had to use a second level of artificial intelligence in order to model human workgroups and skills, finally we used a third level of artificial intelligence to model workflows executed by the said human resources once organized in groups and levels according to their experiences. In our best intentions, such a three level artificial intelligence could address the limits that, until now, have refrained the crowds from growing up as companies, with a well recognizable pyramidal structure, in order to reward experience, skill and professionalism of their workers. We cannot frankly say whether our work will really contribute or not to the so called "crowdsourcing revolution", but we hope at least to have shedded some light on the agreeable possibilities that are

yet to come. Certainly, a lot of steps have yet to be undertaken, and a lot of work and energy will have to be spent on perfecting this technology. As I said on the first introduction, modelling the human behavior is not an easy task, mainly due to the very source of the related a phenomena: the human mind. And I am both unsettled and reassured by the littleness of the knowledge we have actually acquired regarding the human mind.

This long journey, represented by the topics summarized in this thesis, started several years ago, long before the beginning of the PhD studies of which this thesis should represent the conclusion. On the other hand, while this work did not start with this PhD course, we also believe that this work will not end with it. As a matter of fact, the only conclusion we can reach so far is that there is no conclusion. This is a continuing work, an unfading challenge, and of course, a pleasant journey. Therefore, with the same perseverance, and stubbornness, and satisfaction we dare to stand to the challenge. If science is the best achievement of the human mind, human mind will be the best achievement of science. With this certainty, we commit ourselves to the future. Thanks for the reading.

Acknowledgements

> There can always be new beginnings.
> Even for people like us.
>
> Joseph Michael Straczynski

As the reader will have now noticed, this thesis is not only the recap of a series of scientific results, but also the narration of a story. Namely my personal one: the story of a long voyage along a very loopy path. It has been said that a path does not work the same way twice for anyone, that the path follows you and rolls up behind you as you walk, forcing the next person to find his own way. I can finally conclude that I have found my path, and while this path rolls up behind, I realize that I've never walked it alone. Especially during my PhD studies, despite the shortness of this portion of my life, while walking this path on my own, I have met several extraordinary souls, more than collaborators or acquaintances: people to care for and that I can undeservedly call Friends. Certainly, if any accomplishment has been achieved, it was only due to the support and friendship that surrounded me. With this certainty only then I conclude this PhD course, turning a page, but only to begin a new chapter. I begin this new chapter of my life with more questions than answers, with more doubts than assurances, with an unshakeable sense of wonder for the universe that surrounds me, with a deep sense of gratitude for its greatness, with a restless will to investigate it, realizing how ridiculously lucky I am to do what I love for living.

Bibliography

[1] T. Abdelzaher and N. Bhatti. Web server QoS management by adaptive content delivery. In *Proceedings of IWQoS*, pages 216–225. IEEE, 1999.

[2] T. F. Abdelzaher, K. G. Shin, and N. Bhatti. Performance guarantees for web server end-systems: A control-theoretical approach. *IEEE Transactions on Parallel and Distributed Systems*, 13:80–96, 2002.

[3] L. Adamic and E. Adar. How to search a social network. *Social Networks*, 27(3):187–203, 2005.

[4] R. Aguiar and M. Collares-Pereira. Tag: A time-dependent, autoregressive, gaussian model for generating synthetic hourly radiation. *Solar Energy*, 49(3):167–174, 1992.

[5] C. W. Ahn and R. Ramakrishna. Qos provisioning dynamic connection-admission control for multimedia wireless networks using a hopfield neural network. *Transactions on Vehicular Technology*, 53(1):106–117, 2004.

[6] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of topological characteristics of huge online social networking services. In *Proceedings of World Wide Web*, pages 835–844. ACM, 2007.

[7] E. M. Airoldi, D. M. Blei, E. P. Xing, and S. E. Fienberg. Mixed membership stochastic block models. In *Advances in Neural Information Processing Systems (NIPS)*. Red Hook, 2009.

[8] S. Al-Alawi and H. Al-Hinai. An ann-based approach for predicting global radiation in locations with no direct measurement instrumentation. *Renewable Energy*, 14(1):199–204, 1998.

[9] R. A. Albert-László Barabási and H. Jeong. Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and its Applications*, 272(12):173–187, 1999.

[10] S.-i. Amari. Mathematical foundations of neurocomputing. *Proceedings of the IEEE*, 78(9):1443–1463, 1990.

[11] J. E. Arlot et al. The phemu09 catalogue and astrometric results of the observations of the mutual occultations and eclipses of the galilean satellites of jupiter made in 2009. *Astronomy & Astrophysics*, 572:A120, 2014.

[12] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, Apr. 2010.

[13] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[14] S. Banach. *Theory of linear operations*, volume 38. Elsevier, 1987.

[15] F. Bannò, D. Marletta, G. Pappalardo, and E. Tramontana. Tackling consistency issues for runtime updating distributed systems. In *Proceedings of International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8. IEEE, 2010. DOI: 10.1109/IPDPSW.2010.5470863.

[16] A.-L. Barabási. Scale-free networks: a decade and beyond. *Science*, 325(5939):412–413, 2009.

[17] S. K. Barker and P. Shenoy. Empirical evaluation of latency-sensitive application performance in the cloud. In *Proceedings of Multimedia systems*, MMSys, pages 35–46. ACM, February 2010.

[18] E. T. Bell. *Men of mathematics*. Simon and Schuster, 2014.

[19] F. Beritelli, G. Capizzi, G. Lo Sciuto, C. Napoli, E. Tramontana, and M. Woniak. Reducing interferences in wireless communication systems by mobile agents with recurrent neural networks-based adaptive channel equalization. In *SPIE proceedings*, volume 9662, 2015.

[20] J. M. Blanquer, A. Batchelli, K. Schauser, and R. Wolski. Quorum: flexible quality of service for internet services. In *Proceedings of NSDI*, pages 159–174. USENIX, May 2005.

[21] P. Bodık, R. Griffith, C. Sutton, A. Fox, M. Jordan, and D. Patterson. Statistical machine learning makes automatic control practical for internet datacenters. In *Proceedings of HotCloud*. USENIX, 2009.

[22] F. Bonanno, G. Capizzi, S. Coco, C. Napoli, A. Laudani, and G. Lo Sciuto. Optimal thicknesses determination in a multilayer structure to improve the spp efficiency for photovoltaic devices by an hybrid fem–cascade neural network based approach. In *Proceedings of IEEE International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, pages 355–362, Ischia, Italy, June 2014.

[23] F. Bonanno, G. Capizzi, A. Gagliano, and C. Napoli. Optimal management of various renewable energy sources by a new forecasting method. In *Proceedings of IEEE International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, pages 934–940, Sorrento, Italy, June 2012.

[24] F. Bonanno, G. Capizzi, G. Graditi, C. Napoli, and G. Tina. A radial basis function neural network based approach for the electrical characteristics estimation of a photovoltaic module. *Applied Energy*, 97:956–961, 2012.

[25] F. Bonanno, G. Capizzi, G. Graditi, C. Napoli, and G. Tina. A radial basis function neural network based approach for the electrical characteristics estimation of a photovoltaic module. *Applied Energy*, 97:956–961, 2012.

[26] F. Bonanno, G. Capizzi, G. Lo Sciuto, C. Napoli, G. Pappalardo, and E. Tramontana. A cascade neural network architecture investigating surface plasmon polaritons propagation for thin metals in openmp. In *Proceedings of International Conference on Artificial Intelligence and Soft Computing (ICAISC)*, volume 8467 of *Springer LNCS*, pages 22–33, Zakopane, Poland, June 2014.

[27] F. Bonanno, G. Capizzi, G. Lo Sciuto, C. Napoli, G. Pappalardo, and E. Tramontana. A novel cloud-distributed toolbox for optimal energy dispatch management from renewables in igss by using wrnn predictors and gpu parallel solutions. In *International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, pages 1077–1084. IEEE, 2014.

[28] F. Bonanno, G. Capizzi, and C. Napoli. Hybrid neural networks architectures for soc and voltage prediction of new generation batteries storage. In *Proceedings of IEEE international conference on clean electrical power (ICCEP)*, pages 341–344, Ischia, Italy, June 2011.

[29] F. Bonanno, G. Capizzi, and C. Napoli. Some remarks on the application of rnn and prnn for the charge-discharge simulation of advanced lithium-ions battery energy storage. In *Proceedings of IEEE International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, pages 941–945, Sorrento, Italy, June 2012.

[30] F. Bonanno, G. Capizzi, G. L. Sciuto, and C. Napoli. Wavelet recurrent neural network with semi-parametric input data preprocessing for micro-wind power forecasting in integrated generation systems. In *Clean Electrical Power (ICCEP), 2015 International Conference on*, pages 602–609. IEEE, 2015.

[31] G. Borowik, M. Woźniak, A. Fornaia, R. Giunta, C. Napoli, G. Pappalardo, and E. Tramontana. A software architecture assisting workflow executions on

cloud resources. *International Journal of Electronics and Telecommunications*, 61(1):17–23, 2015.

[32] M. Buchanan. *Nexus: small worlds and the groundbreaking theory of networks*. WW Norton & Company, 2003.

[33] M. Buhrmester, T. Kwang, and S. D. Gosling. Amazon's mechanical turk a new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5, 2011.

[34] R. Buyya, R. Ranjan, and R. N. Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *Algorithms and architectures for parallel processing*, pages 13–31. Springer, 2010.

[35] R. N. Calheiros, C. Vecchiola, D. Karunamoorthy, and R. Buyya. The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds. *Future Generation Computer Systems*, 28(6):861–870, 2012.

[36] G. Capizzi, F. Bonanno, and C. Napoli. A new approach for lead-acid batteries modeling by local cosine. In *Proceedings of IEEE International Symposium on Power Electronics Electrical Drives Automation and Motion (SPEEDAM)*, pages 1074–1079, Pisa, Italy, June 2010.

[37] G. Capizzi, F. Bonanno, and C. Napoli. A wavelet based prediction of wind and solar energy for long-term simulation of integrated generation systems. In *Power Electronics Electrical Drives Automation and Motion (SPEEDAM), 2010 International Symposium on*, pages 586–592. IEEE, 2010.

[38] G. Capizzi, F. Bonanno, and C. Napoli. Recurrent neural network-based control strategy for battery energy storage in generation systems with intermittent renewable energy sources. In *IEEE international conference on clean electrical power (ICCEP)*, pages 336–340. IEEE, 2011.

[39] G. Capizzi, C. Napoli, and F. Bonanno. Innovative second-generation wavelets construction with recurrent neural networks for solar radiation forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 23(11):1805–1815, 2012.

[40] G. Capizzi, C. Napoli, and L. Paternò. An innovative hybrid neuro-wavelet method for reconstruction of missing data in astronomical photometric surveys. In *Artificial Intelligence and Soft Computing*, pages 21–29. Springer Berlin Heidelberg, 2012.

[41] G. Chen, C.-L. Wang, and F. C. M. Lau. p-jigsaw: a cluster-based web server with cooperative caching support. *Concurrency and Computation: Practice and Experience*, 15(7-8):681–705, 2003.

[42] M. Chiani, D. Dardari, and M. K. Simon. New exponential bounds and approximations for the computation of error probability in fading channels. *IEEE Transactions on Wireless Communications*, 2(4):840–845, July 2003.

[43] K. Choromański, M. Matuszak, and J. Mikisz. Scale-free graph with preferential attachment and evolving internal vertex structure. *Journal of Statistical Physics*, 151(6):1175–1183, 2013.

[44] O. Christensen. An introduction to frames and riesz bases, appl. *Num. Harmon. Anal. Birkhauser Boston, Inc., Boston, MA*, 2003.

[45] R. L. Claypoole Jr, R. G. Baraniuk, and R. D. Nowak. Adaptive wavelet transforms via lifting. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 3, pages 1513–1516. IEEE, 1998.

[46] A. Cohen. Ondelettes, analyses multiresolutions et filtres miroirs en quadrature. *Annales de l'IHP Analyse non linéaire*, 7(5):439–459, 1990.

[47] A. Cohen and R. D. Ryan. *Wavelets and multiscale signal processing*. Springer, 1995.

[48] B. Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72, 2003.

[49] B. Cohen. The bittorrent protocol specification, 2008.

[50] M. Cohen, S. Grossberg, et al. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *Systems, Man and Cybernetics, IEEE Transactions on*, (5):815–826, 1983.

[51] R. Cohen and S. Havlin. Scale-free networks are ultrasmall. *Physical review letters*, 90(5):058701, 2003.

[52] R. Cohen, S. Havlin, and D. ben Avraham. Structural properties of scale-free networks. *Handbook of Graphs and Networks: From the Genome to the Internet*, pages 85–110, 2002.

[53] J. T. Connor, R. D. Martin, and L. Atlas. Recurrent neural networks and robust time series prediction. *Transactions on Neural Networks*, 5(2):240–254, 1994.

[54] A. A. Cournot and E. P. Bottinelli. *Souvenirs d'A. Cournot (1760-1860)*. Hachette, 1913.

[55] B. C. Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24, 2001.

[56] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[57] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *Information Theory, IEEE Transactions on*, 36(5):961–1005, 1990.

[58] I. Daubechies, A. Grossmann, and Y. Meyer. Painless nonorthogonal expansions. *Journal of Mathematical Physics*, 27(5):1271–1283, 1986.

[59] H. Deshuang and M. Songde. A new radial basis probabilistic neural network model. In *Proceedings of Conference on Signal Processing*, volume 2. IEEE, 1996.

[60] A. Di Stefano, M. Fargetta, G. Pappalardo, and E. Tramontana. Supporting resource reservation and allocation for unaware applications in grid systems. *Concurrency and Computation: Practice and Experience*, 18(8):851–863, 2006. DOI: 10.1002/cpe.980.

[61] J. Dilley, R. Friedrich, T. Jin, and J. A. Rolia. Web server performance measurement and modeling techniques. *Performance Evaluation*, 33(1):27–44, 1998.

[62] P. A. Dinda and D. R. O'Hallaron. Host load prediction using linear models. *Cluster Computing*, 3(4):265–280, 2000.

[63] T. Dornemann, E. Juhnke, and B. Freisleben. On-demand resource provisioning for bpel workflows using amazon's elastic compute cloud. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 140–147. IEEE, 2009.

[64] T. Dornemann, E. Juhnke, T. Noll, D. Seiler, and B. Freisleben. Data flow driven scheduling of bpel workflows using cloud resources. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 196–203. IEEE, 2010.

[65] M. Du Sautoy. *Finding moonshine: A mathematician's journey through symmetry*. Fourth estate London, 2008.

[66] W. Duch. Towards comprehensive foundations of computational intelligence. In *Challenges for Computational Intelligence*, pages 261–316. Springer, 2007.

[67] W. Duch and N. Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, 2(1):163–212, 1999.

[68] C. Dwyer, S. Hiltz, and K. Passerini. Trust and privacy concern within social networking sites: A comparison of facebook and myspace. In *Proceedings of Americas Conference on Information Systems*, pages 339–351, 2007.

[69] M. Egido and E. Lorenzo. The sizing of stand alone pv-system: a review and a proposed new method. *Solar Energy Materials and Solar Cells*, 26(1):51–69, 1992.

[70] B. Epstein and M. Schwartz. Reservation strategies for multi-media traffic in a wireless environment. In *Proceedings of Vehicular Technology Conference*, volume 1, pages 165–169. IEEE, 1995.

[71] T. Erl. *SOA design patterns*. Pearson Education, 2008.

[72] R. Fielding. Representational state transfer (rest). *Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine*, page 120, 2000.

[73] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[74] W. Finnoff, F. Hergert, and H. G. Zimmermann. Improving model selection by nonconvergent methods. *Neural Networks*, 6(6):771–783, 1993.

[75] A. Fornaia, C. Napoli, G. Pappalardo, and E. Tramontana. An ao system for oo-gpu programming. In *16th Workshop From Object to Agents (WOA15)*, volume 1382, pages 24–31. CEUR-WS, 2015.

[76] A. Fornaia, C. Napoli, G. Pappalardo, and E. Tramontana. Using aop neural networks to infer user behaviours and interests. In *16th Workshop From Object to Agents (WOA15)*, volume 1382, pages 46–52. CEUR-WS, 2015.

[77] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: Answering queries with crowdsourcing. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 61–72, Athens, Greece, 2011.

[78] S. Gaikwad et al. Daemo: A self-governed crowdsourcing marketplace. In *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface*

Software & Technology, UIST '15 Adjunct, pages 101–102, New York, NY, USA, 2015. ACM.

[79] E. Gamma, R. Helm, R. Johnson, and R. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, 1994.

[80] R. Giunta, F. Messina, G. Pappalardo, and E. Tramontana. Providing qos strategies and cloud-integration to web servers by means of aspects. *Concurrency and Computation: Practice and Experience*, 2013. DOI:10.1002/cpe.3031.

[81] R. Giunta, F. Messina, G. Pappalardo, and E. Tramontana. Kaqudai: a dependable web infrastructure made out of existing components. In *Proceedings of Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, 2013. DOI:10.1109/WETICE.2013.47.

[82] R. Giunta, G. Pappalardo, and E. Tramontana. Handling replica management concerns by means of aspects. In *Proceedings of WETICE*, pages 284–289. IEEE, 2007.

[83] R. Giunta, G. Pappalardo, and E. Tramontana. AODP: refactoring code to provide advanced aspect-oriented modularization of design patterns. In *Proceedings of SAC*, pages 1243–1250. ACM, March 2012.

[84] R. Giunta, G. Pappalardo, and E. Tramontana. Superimposing roles for design patterns into application classes by means of aspects. In *Proceedings of the ACM Symposium on Applied Computing, SAC*, pages 1866–1868. ACM, March 2012. DOI: 10.1145/2245276.2232082.

[85] M. Gladwell. Tipping points. *How Little Things can make a Big Difference*, 2000.

[86] J. Gleick. *Chaos: Making a new science*. Random House, 1997.

[87] M. Granovetter. The strength of weak ties. *The American Journal of Sociology*, 78(6):1360–1380, 1973.

[88] M. Granovetter. Threshold models of collective behavior. *American journal of sociology*, pages 1420–1443, 1978.

[89] S. Grossberg. Nonlinear difference-differential equations in prediction and learning theory. *Proceedings of the National Academy of Sciences of the United States of America*, 58(4):1329, 1967.

[90] J. Guitart, J. Torres, and E. Ayguadé. A survey on performance management for internet applications. *Concurrency and Comp.: Practice and Experience*, 22(1):68–106, 2009.

[91] M. Gupta, L. Jin, and N. Homma. *Static and dynamic neural networks: from fundamentals to advanced theory*. John Wiley & Sons, 2004.

[92] M. Hamdaqa, T. Livogiannis, and L. Tahvildari. A reference model for developing cloud applications. In *Proceedings of CLOSER*, volume 11, pages 98–103. SciTePress, May 2011.

[93] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2004.

[94] S. Haykin. *Neural networks and learning machines*, volume 3. Prentice Hall New York, 2009.

[95] S. Hokoi, M. Matsumoto, and M. Kagawa. Stochastic models of solar radiation and outdoor temperature. *ASHRAE Transactions (American Society of Heating, Refrigerating and Air-Conditioning Engineers);(United States)*, 96(CONF-9006117–), 1990.

[96] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.

[97] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[98] G. Iachello et al. Control, deception, and communication: Evaluating the deployment of a location-enhanced messaging service. In *Ubiquitous Computing*. Springer, 2005.

[99] S. A. Imhoff, D. Y. Roeum, and M. R. Rosiek. New classes of frame wavelets for applications. In *SPIE's 1995 Symposium on OE/Aerospace Sensing and Dual Use Photonics*, pages 923–934. International Society for Optics and Photonics, 1995.

[100] A. Iosup, N. Yigitbasi, and D. Epema. On the performance variability of production cloud services. In *Proceedings of Cluster, Cloud and Grid Computing (CCGrid)*, pages 104–113. IEEE, 2011.

[101] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP)*, pages 64–67, Washington DC, 2010.

[102] S. Islam, J. Keung, K. Lee, and A. Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28(1):155–162, 2012.

[103] C. Jones and E. H. Volpe. Organizational identification: Extending our understanding of social identities through social networks. *Journal of Organizational Behavior*, 32(3):413–434, 2011.

[104] C. G. Kang, Y. J. Kim, and M. J. Hwang. Implicit scheduling algorithm for dynamic slot assignment in wireless atm networks. *Electronics Letters*, 34(24):2309–2311, 1998.

[105] M. Karlsson, X. Zhu, and C. Karamanolis. An adaptive optimal controller for non-intrusive performance differentiation in computing services. In *Proceedings of ICCA*, volume 2, pages 709–714. IEEE, June 2005.

[106] S. Kaune, R. C. Rumin, G. Tyson, A. Mauthe, C. Guerrero, and R. Steinmetz. Unraveling bittorrent's file unavailability: Measurements and analysis. In *Proceedings of IEEE P2P*, pages 1–9, 2010.

[107] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, and J. Irwin. Aspect-oriented programming. In *Lecture Notes in Computer Science - ECOOP*, volume 1241 of *LNCS*, pages 220–242. Springer, 1997.

[108] C. Kiss, A. Scholz, and M. Bichler. Evaluating centrality measures in large call graphs. In *Proceedings of IEEE Enterprise Computing, E-Commerce, and E-Services*, 2006.

[109] A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 453–456, Florence, Italy, 2008.

[110] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. kvm: the linux virtual machine monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, June 2007.

[111] K. Knight, S. Klein, and J. Duffie. A methodology for the synthesis of hourly weather data. *Solar Energy*, 46(2):109–120, 1991.

[112] A. N. Kolmogorov. On the conservation of quasi-periodic motions for a small perturbation of the hamiltonian function. In *Dokl. Akad. Nauk SSSR*, volume 98, pages 527–530, 1954.

[113] A. H. Kramer and A. Sangiovanni-Vincentelli. Efficient parallel learning algorithms for neural networks. In *Advances in neural information processing systems*, pages 40–48, 1989.

[114] S. C. Kremer. On the computational power of elman-style recurrent networks. *Neural Networks, IEEE Transactions on*, 6(4):1000–1004, 1995.

[115] T. Kwon, I. Park, Y. Choi, and S. Das. Bandwidth adaption algorithms with multi-objectives for adaptive multimedia services in wireless/mobile networks. In *Proceedings of International Workshop on Wireless Mobile Multimedia*, pages 51–59. ACM, 1999.

[116] R. Laddad. *AspectJ in Action*. Manning Publications Co., Grennwich, Conn., 2003.

[117] C. A. Lampe, N. Ellison, and C. Steinfield. A familiar face (book): profile elements as signals in an online social network. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 435–444. ACM, 2007.

[118] S. Laplace and P. Simon. A philosophical essay on probabilities, translated from the 6th french edition by frederick wilson truscott and frederick lincoln emory, 1951.

[119] V. W. Lee et al. Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu. In *Proceedings of International Symposium on Computer Architecture (ISCA)*. ACM, 2010.

[120] T.-Y. Li and J. A. Yorke. Period three implies chaos. *American mathematical monthly*, pages 985–992, 1975.

[121] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[122] H. C. Lim, S. Babu, and J. S. Chase. Automated control for elastic storage. In *Proceedings of ICAC*, pages 1–10. ACM, June 2010.

[123] E. Lorenz. Predictability: Does the flap of a butterfly's wings in brazil set off a tornado in texas? In *139th meeting of the Association for the Advancement of Science*, 1972.

[124] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[125] J. Loyall, D. Bakken, R. Schantz, J. Zinky, D. Karr, R. Vanegas, and K. R. Anderson. Qos aspect languages and their runtime integration. In *Lecture Notes in Computer Science - LCR Workshop*, volume 1511 of *LNCS*, pages 303–318. Springer, May 1998.

[126] A. Maafi and A. Adane. A two-state markovian model of global irradiation suitable for photovoltaic conversion. *Solar & wind technology*, 6(3):247–252, 1989.

[127] S. Mallat. Multiresolution approximation and orthonormal bases of $l^2\mathbb{R}$. *Transactions of the American Mathematics Society*, 315:69–87, 1989.

[128] S. Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[129] M. Mao and M. Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *Proceedings of SC*, pages 1–12. ACM, 2011.

[130] Z. Marszałek, G. Woźniak, M. Borowik, R. Wazirali, C. Napoli, G. Pappalardo, and E. Tramontana. Benchmark tests on improved merge for big data processing. In *Asia-Pacific Conference on Computer Aided System Engineering (APCASE)*, pages 96–101, 14-16 July, Quito, Ecuador, 2015. IEEE. doi: 10.1109/APCASE.2015.24.

[131] J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems 25*, pages 548–556, 2012.

[132] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[133] A. Mellit, M. Benghanem, A. Hadj Arab, and A. Guessoum. Modelling of sizing the photovoltaic system parameters using artificial neural network. In *Control Applications, 2003. CCA 2003. Proceedings of 2003 IEEE Conference on*, volume 1, pages 353–357. IEEE, 2003.

[134] Y. Meyer and D. H. Salinger. *Wavelets and operators*, volume 1. Cambridge university press, 1995.

[135] S. Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

[136] K. Miller, T. Griffiths, and M. Jordan. Nonparametric latent feature models for link prediction. *Advances in neural information processing systems*, 22:1276–1284, 2009.

[137] M. L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967.

[138] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 251–260. ACM, 2010.

[139] L. Mora-Lopez and M. Sidrach-de Cardona. Multiplicative arma models to generate hourly series of global irradiation. *Solar Energy*, 63(5):283–291, 1998.

[140] J. Möser. On invariant curves of area-preserving mappings of an annulus. *Nachr. Akad. Wiss. Göttingen, II*, pages 1–20, 1962.

[141] C. Napoli, F. Bonanno, and G. Capizzi. Exploiting solar wind time series correlation with magnetospheric response by using an hybrid neuro-wavelet approach. In *IAU Symposium 274*, volume 6, pages 156–158. Cambridge University Press, 2010.

[142] C. Napoli, F. Bonanno, and G. Capizzi. An hybrid neuro-wavelet approach for long-term prediction of solar wind. In *IAU Symposium 274*, pages 247–249, 2010.

[143] C. Napoli, G. Pappalardo, G. Tina, and E. Tramontana. Cooperative strategy for optimal management of smart grids by wavelet rnns and cloud computing. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–14, 2015.

[144] C. Napoli, G. Pappalardo, M. Tina, and E. Tramontana. Cooperative strategy for optimal management of smart grids by wavelet rnns and cloud computing. *IEEE Transactions on Neural Networks and Learning Systems*, (in press) 2015.

[145] C. Napoli, G. Pappalardo, and E. Tramontana. A hybrid neuro–wavelet predictor for qos control and stability. In *AI*IA 2013: Advances in Artificial Intelligence*, volume 9119 of *Lecture Notes in Computer Science*, pages 527–538. Springer International Publishing, 2013.

[146] C. Napoli, G. Pappalardo, and E. Tramontana. Using modularity metrics to assist move method refactoring of large systems. In *International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pages 529–534. IEEE, 2013.

[147] C. Napoli, G. Pappalardo, and E. Tramontana. Using modularity metrics to assist move method refactoring of large systems. In *2013 7th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pages 529–534. IEEE, 2013.

[148] C. Napoli, G. Pappalardo, and E. Tramontana. An agent-driven semantical identifier using radial basis neural networks and reinforcement learning. In *XV Workshop "Dagli Oggetti agli Agenti"*, volume 1260. CEUR-WS, 2014.

[149] C. Napoli, G. Pappalardo, and E. Tramontana. Improving files availability for bittorrent using a diffusion model. In *23rd IEEE International WETICE Conference*, pages 191–196, 2014.

[150] C. Napoli, G. Pappalardo, and E. Tramontana. A mathematical model for file fragment diffusion and a nerual predictor to manage priority queues over bit torrent. *International Journal of Applied Mathematics and Computer Sciences*, (in press) 2015.

[151] C. Napoli, G. Pappalardo, E. Tramontana, G. Borowik, D. Połap, and M. Woźniak. Toward 2d image classifier based on firefly algorithm with simplified sobel filter. In *Asia-Pacific Conference on Computer Aided System Engineering (APCASE)*, pages 185–189, 14-16 July, Quito, Ecuador, 2015. IEEE. doi: 10.1109/APCASE.2015.40.

[152] C. Napoli, G. Pappalardo, E. Tramontana, Z. Marszałek, D. Połap, and M. Woźniak. Simplified firefly algorithm for 2d image key-points search. In *Symposium on Computational Intelligence for Human-like Intelligence (CHILI)*, Symposium Series on Computational Intelligence (SSCI), pages 118–125. IEEE, 2014.

[153] C. Napoli, G. Pappalardo, E. Tramontana, R. Nowicki, J. Starczewski, and M. Woźniak. Toward work groups classification based on probabilistic neural network approach. In *Artificial Intelligence and Soft Computing*, volume 9119 of *Lecture Notes in Computer Science*, pages 79–89. Springer International Publishing, 2015.

[154] C. Napoli, G. Pappalardo, E. Tramontana, and G. Zappalà. A cloud-distributed gpu architecture for pattern identification in segmented detectors big-data surveys. *The Computer Journal*, page bxu147, 2014. doi: 10.1093/comjnl/bxu147.

[155] C. Napoli and E. Tramontana. An object-oriented neural network toolbox based on design patterns. In G. Dregvaite and R. Damasevicius, editors, *Information and Software Technologies*, volume 538 of *Communications in Computer and Information Science*, pages 388–399. Springer International Publishing, 2015.

[156] C. Napoli, E. Tramontana, G. Lo Sciuto, M. Woźniak, R. Damaševičius, and G. Borowik. Authorship semantical identification using holomorphic chebyshev projectors. In *Asia-Pacific Conference on Computer Aided System Engineering (APCASE)*, pages 232–237, 14-16 July, Quito, Ecuador, 2015. IEEE. doi: 10.1109/APCASE.2015.48.

[157] C. Napoli, E. Tramontana, and M. Woźniak. Enhancing environmental surveillance against organised crime with radial basis neural networks. In *IEEE Symposium on Computational Intelligence for Human–like Intelligence*, (in press) 2015.

[158] R. Nathuji, A. Kansal, and A. Ghaffarkhah. Q-clouds: managing performance interference effects for qos-aware clouds. In *Proceedings of Eurosys*, pages 237–250. ACM, April 2010.

[159] M. Nehrir, C. Wang, K. Strunz, H. Aki, R. Ramakumar, J. Bing, Z. Miao, and Z. Salameh. A review of hybrid renewable/alternative energy systems for electric power generation: Configurations, control, and applications. *IEEE Transactions on Sustainable Energy*, 2(4):392–403, 2011.

[160] M. E. Newman. Assortative mixing in networks. *Physical review letters*, 89(20):208701, 2002.

[161] H. Nguyen. *GPU Gems 3*. Addison-Wesley, 2008.

[162] G. Novelli, G. Pappalardo, C. Santoro, and E. Tramontana. A grid-based infrastructure to support multimedia content distribution. In *Proceedings of the UPGRADE-CN workshop jointly held with HPDC*, pages 57–64. ACM, June 2007.

[163] J. Novovičová, P. Somol, M. Haindl, and P. Pudil. Conditional mutual information based feature selection for classification task. In *Progress in Pattern Recognition, Image Analysis and Applications*, pages 417–426. Springer, 2007.

[164] B. Nowak, R. Nowicki, M. Woźniak, and C. Napoli. Multi-class nearest neighbour classifier for incomplete data handling. In *Artificial Intelligence and Soft Computing*, volume 9119 of *Lecture Notes in Computer Science*, pages 469–480. Springer International Publishing, 2015.

[165] K. Nowicki and T. A. B. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087, 2001.

[166] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In *Proceedings of CCGRID*, pages 124–131. IEEE, May 2009.

[167] NVIDIA. CUDA compute unified device architecture programming guide. http://www.nvidia.com.

[168] M. Opper and D. Haussler. Generalization performance of bayes optimal classification algorithm for learning a perceptron. *Phys. Rev. Lett.*, 66:2677–2680, May 1991.

[169] G. Ortiz and B. Bordbar. Aspect-oriented quality of service for web services: A model-driven approach. In *Proceedings of ICWS*, pages 559–566. IEEE, July 2009.

[170] S. Pandey, D. Karunamoorthy, and R. Buyya. Workflow engine for clouds. *Cloud Computing: Principles and Paradigms*, pages 321–344, 2011.

[171] G. Pappalardo and E. Tramontana. Suggesting extract class refactoring opportunities by measuring strength of method interactions. In *Proceedings of Asia Pacific Software Engineering Conference (APSEC)*, pages 105–110. IEEE, December 2013.

[172] P. Peretto. Collective properties of neural networks: a statistical physics approach. *Biological cybernetics*, 50(1):51–62, 1984.

[173] R. Powers, M. Goldszmidt, and I. Cohen. Short term performance forecasting in enterprise systems. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 801–807. ACM, 2005.

[174] D. Poap, M. Woniak, C. Napoli, E. Tramontana, and R. Damaeviius. Is the colony of ants able to recognize graphic objects? In G. Dregvaite and R. Damasevicius, editors, *Information and Software Technologies*, volume 538 of *Communications in Computer and Information Science*, pages 376–387. Springer International Publishing, 2015.

[175] L. R. Rabiner and B. Gold. Theory and application of digital signal processing. *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975. 777 p.*, 1, 1975.

[176] S. S. Rao and B. Kumthekar. Recurrent wavelet networks. In *IEEE International Conference on Neural Networks*, volume 5, pages 3143–3147. IEEE, 1994.

[177] S. S. Rao and B. Kumthekar. A composite neural architecture and algorithm for nonlinear system identification. In *Proceedings of the 1995 artificial neural networks in engineering conference (ANNIE95)*, pages 77–84, 1995.

[178] Y. Ren, D. Bakken, T. Courtney, M. Cukier, D. Karr, P. Rubel, C. Sabnis, W. Sanders, R. Schantz, and M. Seri. AQuA: an adaptive architecture that provides dependable distributed objects. *IEEE Transactions on Computers*, 52(1):31–50, 2003.

[179] J. Ross, L. Irani, M. S. Silberman, A. Zaldivar, and B. Tomlinson. Who are the crowdworkers?: Shifting demographics in mechanical turk. In *Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 2863–2872, New York, NY, USA, 2010. ACM.

[180] S. Schechter, M. Krishnan, and M. D. Smith. Using path profiles to predict http requests. *Computer Networks and ISDN Systems*, 30(1):457–467, 1998.

[181] S. Schnettler. A structured overview of 50 years of small-world research. *Social Networks*, 31(3):165–178, July 2009.

[182] N. Schwartz, R. Cohen, D. ben Avraham, A.-L. Barabási, and S. Havlin. Percolation in directed scale-free networks. *Phys. Rev. E*, 66:015104, Jul 2002.

[183] C. E. Shannon and W. Weaver. *The mathematical theory of communication.* University of Illinois Press, 1959.

[184] R. Shaw. Strange attractors, chaotic behavior, and information flow. *Zeitschrift für Naturforschung A*, 36(1):80–112, 1981.

[185] M. R. Sherif, I. W. Habib, M. Nagshineh, and P. Kermani. Adaptive allocation of resources and call admission control for wireless atm using genetic algorithms. *Journal on Selected Areas in Communications*, 18(2):268–282, 2000.

[186] S. Smale and D.-X. Zhou. Learning theory estimates via integral operators and their approximations. *Constructive approximation*, 26(2):153–172, 2007.

[187] G. W. Stewart. Block gramschmidt orthogonalization. *SIAM Journal on Scientific Computing*, 31(1):761–775, 2008.

[188] G. Strang and T. Nguyen. *Wavelets and filter banks*. SIAM, 1996.

[189] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal on Mathematical Analysis*, 29(2):511–546, 1998.

[190] S. Tambe, A. Dabholkar, and A. S. Gokhale. CQML: Aspect-oriented modeling for modularizing and weaving QoS concerns in component-based systems. In *Proceedings of ECBS*, pages 11–20. IEEE, April 2009.

[191] R. Tolosana-Calasanz, J. Á. BañAres, C. Pham, and O. F. Rana. Enforcing qos in scientific workflow systems enacted over cloud infrastructures. *Journal of Computer and System Sciences*, 78(5):1300–1315, 2012.

[192] E. Tramontana. Automatically characterising components with concerns and reducing tangling. In *Proceedings of QUORS workshop at Compsac*, pages 499–504. IEEE, 2013.

[193] P. Vaidyanathan. Quadrature mirror filter banks, m-band extensions and perfect reconstruction techniques. *IEEE ASSP Magazine*, 4, 1983.

[194] P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice Hall, Englewood Cliffs, 1993.

[195] L. M. Vaquero et al. Dynamically scaling applications in the cloud. *SIGCOMM Comput. Commun. Rev.*, 41(1):45–52, Jan. 2011.

[196] M. Vetterli and J. Kovacevic. *Wavelets and subband coding*. Prentice Hall, Englewood Cliffs, 2007.

[197] A. VI. Proof of a theorem of an kolmogorov on the preservation of conditionally periodic motions under a small perturbation of the hamiltonian. *Uspekhi Mat. Nauk*, 5(113):18, 1963.

[198] L. Wang, M. Kunze, J. Tao, and G. von Laszewski. Towards building a cloud for scientific applications. *Advances in Engineering Software*, 42(9):714–722, 2011.

[199] D. J. Watts and S. H. Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442, 1998.

[200] H. Weyl. *The theory of groups and quantum mechanics*. Courier Corporation, 1950.

[201] B. Widrow, M. E. Hoff, et al. Adaptive switching circuits. Technical report, Defense Technical Information Center, 1960.

[202] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[203] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. *Back-propagation: Theory, architectures and applications*, pages 433–486, 1995.

[204] M. Wozniak, C. Napoli, E. Tramontana, and G. Capizzi. A multiscale image compressor with rbfnn and discrete wavelet decomposition. In *International Joint Conference on Neural Networks (IJCNN)*, page 12191225. IEEE, 2015. doi: 10.1109/IJCNN.2015.7280461.

[205] M. Woźniak, D. Połap, G. Borowik, and C. Napoli. A first attempt to cloud-based user verification in distributed system. In *Asia-Pacific Conference on Computer Aided System Engineering (APCASE)*, pages 226–231, 14-16 July, Quito, Ecuador, 2015. IEEE. doi: 10.1109/APCASE.2015.47.

[206] M. Woźniak, D. Połap, M. Gabryel, R. Nowicki, C. Napoli, and E. Tramontana. Can we process 2d images using artificial bee colony? In *Artificial Intelligence*

and Soft Computing, volume 9119 of *Lecture Notes in Computer Science*, pages 660–671. Springer International Publishing, 2015.

[207] M. Woźniak, D. Połap, L. Kosmider, C. Napoli, and E. Tramontana. A novel approach toward x–ray images classifier. In *IEEE Symposium on Computational Intelligence for Human–like Intelligence*, (in press) 2015.

[208] M. Wozniak, D. Polap, R. K. Nowicki, C. Napoli, G. Pappalardo, and E. Tramontana. Novel approach toward medical signals classifier. In *International Joint Conference on Neural Networks (IJCNN)*, page 19241930. IEEE, 2015. doi: 10.1109/IJCNN.2015.7280556.

[209] K. Xiong and H. Perros. Service performance and analysis in cloud computing. In *Proceedings of Services-I*, pages 693–700. IEEE, 2009.

[210] Z. Xu, V. Tresp, K. Yu, and H. peter Kriegel. Infinite hidden relational models. In *Proceedings of Uncertainity in Artificial Intelligence (UAI)*, 2006.

[211] Y. Yao, L. Rosasco, and A. Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

[212] D. Yuan, Y. Yang, X. Liu, and J. Chen. A data placement strategy in scientific cloud workflows. *Future Generation Computer Systems*, 26(8):1200–1214, 2010.

[213] D. Yuan, Y. Yang, X. Liu, and J. Chen. On-demand minimum cost benchmarking for intermediate dataset storage in scientific cloud workflow systems. *Journal of Parallel and Distributed Computing*, 71(2):316–332, 2011.

[214] D. Yuan, Y. Yang, X. Liu, G. Zhang, and J. Chen. A data dependency based strategy for intermediate data storage in scientific cloud workflow systems. *Concurrency and Computation: Practice and Experience*, 24(9):956–976, 2012.

[215] M. H. Zadeh and M. A. Seyyedi. Qos monitoring for web services by time series forecasting. In *Proceedings of International Conference on Computer Science and Information Technology (ICCSIT)*, volume 5, pages 659–663. IEEE, 2010.

[216] Q. Zhang and A. Benveniste. Wavelet networks. *IEEE Transactions on Neural Networks*, 3(6):889–898, 1992.

[217] W. Zhao, D.-S. Huang, and L. Guo. Optimizing radial basis probabilistic neural networks using recursive orthogonal least squares algorithms combined with micro-genetic algorithms. In *Proceedings of Neural Networks*, volume 3. IEEE, 2003.

[218] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *Proceedings of Data Engineering*. IEEE, 2008.