

POLITECNICO DI MILANO

School of Industrial and Information Engineering

Computer Science and Engineering - Master's Degree

Department of Electronics, Information and Bio-engineering



POLITECNICO
MILANO 1863

**Taming uncertainty in models used by self-adaptive
software: A Neuro-Fuzzy approach**

Supervisor: Prof. Raffaella MIRANDOLA

Cosupervisor: Eng. Diego PEREZ-PALACIN

Author:

Alessandro PAGLIALONGA Matr. 863643

Academic Year 2017–2018

*Lentamente muore chi diventa schiavo dell'abitudine,
ripetendo ogni giorno gli stessi percorsi,
chi non rischia, chi non parla a chi non conosce.*

*Muore lentamente chi evita una passione,
chi preferisce i puntini sulle 'i' ad un insieme di emozioni,
proprio quelle che fanno gli brillare gli occhi.*

*Lentamente muore chi è infelice sul lavoro,
chi non rischia la certezza per l'incertezza per inseguire un sogno,
chi non si permette almeno una volta
nella vita di fuggire ai consigli sensati.*

*Lentamente muore chi non viaggia,
chi non legge, chi non ascolta musica.
Muore lentamente chi non si lascia aiutare,
chi passa i giorni a lamentarsi
della propria sfortuna o della pioggia incessante.*

*Lentamente muore chi abbandona un progetto prima di iniziarlo,
chi non fa domande sugli argomenti che non conosce,
chi non risponde quando gli chiedono qualcosa che conosce.*

Martha Medeiros - Lentamente muore

Ringraziamenti

(Acknowledgments)

Un gesto.

Con un solo piccolissimo gesto non immaginiamo neanche l'entità con cui possiamo cambiare la giornata di una persona. Pensate allora cosa succederebbe con un sorriso spontaneo, un complimento spensierato o semplicemente un 'grazie' inaspettato, se questi fossero ripetuti giorno dopo giorno. Sfortunatamente però, così come è facilissimo rendere migliore la giornata di una persona così è altrettanto facile renderla peggiore. Non odiate però queste persone, è anche grazie a loro se si è più capaci di riconoscere chi compie quel tipo di gesti che ci fanno sentire così amati e sicuri di sé. Io le riconosco adesso più che mai, ed è a tutte loro che dedico questi ringraziamenti.

Un grazie a chi, durante questo lavoro, pur non essendo strettamente il loro compito, mi ha fatto sentire adeguato nonostante le mie mille incertezze, mostrandosi come insegnanti ancor più che professori.

Un grazie quindi a chi mi fa sentire sempre amato da 25 anni, sia che lo faccia come madre, come padre, come fratello o come nonno.

Un grazie a chi da anni cresce ogni giorno con me, con quella complicità caratteristica di chi sta insieme da sempre ma con la gentilezza di chi si conosce da pochi giorni, accogliendo quelle debolezze che io stesso non oso saper accettare di me.

Un grazie agli amici veri, quelli che ti accolgono sempre e che ti fanno sentire a tuo agio in ogni situazione.

Un ultimo grazie, il più importante fra tutti, non misurabile, bagnato da lacrime, a chi avrebbe preso per la prima volta nella vita "l'apparecchio" per venire fino a Milano per vedermi conseguire la laurea.

Grazie.

Abstract

Uncertainty affects every aspect of the real world; even if at different degrees no research field can declare to be immune from it. To better reach the meaning of this concept there simply needs to think about the scientific observations that characterize the research world: these are not the snapshots of 'things as such', i.e. as they are in that precise moment, but rather the snapshots of 'observations of things', which means that we see them as they appear to us (as we observe them) and not as they really are. Computer software and systems are not exempt from uncertainty and in this thesis work we are going to discuss how they are affected from it. We will focus on self-adaptive systems, specifically concerning on the effects of uncertainty in the decision-making part which affects the runtime behavior of these systems.

After a first summary showing one of the different possible taxonomies of uncertainty in the IT world we will present the possible technologies used to mitigate uncertainty, including approaches from the field of Artificial Intelligence. From this latter branch of Computer Science we picked our main weapon: Neuro-Fuzzy technology.

At last we will explain the advantages of the chosen architecture to face the problem, and then we'll dig into the details of its implementation, which uses as dataset the public accessible logs of the Skype Super-Peers network. This work is deeply characterized by an unusual approach with the parameters of the Neuro-Fuzzy system, which may represent an unexplored path to walk in the future.

Sommario

L'incertezza colpisce ogni aspetto del mondo reale che conosciamo, tant'è che nessun campo di ricerca può infatti dichiarare di essere immune da essa. Per carpire al meglio il significato di questo concetto basta semplicemente pensare alle osservazioni scientifiche: quest'ultime non sono l'istantanea delle 'cose così come sono' in un dato momento, ma piuttosto sono l'istantanea delle 'osservazioni di cose', che significa che noi le vediamo così come ci appaiono e non come sono realmente. I software e i sistemi informatici non sono esenti dal concetto dell'incertezza e in questo lavoro di tesi andremo a discutere come quest'ultimi ne sono affetti. Ci concentreremo sui sistemi self-adaptive preoccupandoci nello specifico degli effetti che l'incertezza ha nel modulo responsabile del processo decisionale (decision-making) che modifica il comportamento a runtime di questi sistemi.

Dopo un primo excursus delle possibili differenti tassonomie dell'incertezza del mondo IT andremo a presentare le possibili tecnologie che vengono usate per mitigare l'incertezza, inclusi gli approcci appartenenti al campo dell'intelligenza artificiale. Da quest'ultima branca dell'informatica abbiamo scelto la nostra arma principale: i Sistemi Neuro-Fuzzy.

In ultimo esporremo i vantaggi dell'architettura scelta per poter affrontare il problema; scavaremo poi nei dettagli dell'implementazione adoperante un dataset basato sui log, accessibili pubblicamente, della rete dei Super-Peers di Skype. Questo lavoro è profondamente caratterizzato da un approccio inusuale adottato per i parametri del sistema Neuro-Fuzzy che può rappresentare un cammino inesplorato da percorrere in futuro.

Contents

| | |
|--|-----------|
| Introduction | 1 |
| 1 Importance of the problem of Uncertainty | 5 |
| 1.1 What is Uncertainty? | 5 |
| 1.2 Uncertainty in Computer Science | 8 |
| 2 State of the art methodologies to face uncertainty | 15 |
| 2.1 Common techniques to handle uncertainty | 15 |
| 2.2 Game Theory | 20 |
| 2.3 Fuzzy Logic | 21 |
| 2.4 MAPE-K | 25 |
| 3 State of the art methodologies to face uncertainty | 29 |
| 3.1 Neuro-Fuzzy Systems | 29 |
| 3.1.1 Brief introduction to Neuro-Fuzzy technology | 29 |
| 3.1.2 Basics of Neural Networks | 30 |
| 3.1.3 Feedforward Neural Network | 31 |
| 3.1.4 Advantages and disadvantages of Neural Networks | 31 |
| 3.1.5 Advantages and disadvantages of Fuzzy Logic | 33 |
| 3.1.6 Neuro-Fuzzy Logic in detail | 35 |
| 3.2 Different types of Neuro-Fuzzy Systems | 37 |
| 3.2.1 FALCON: Fuzzy Adaptive Learning Control Network | 40 |
| 3.2.2 GARIC: The Generalized Approximate Reasoning based Intel- ligence Control | 41 |
| 3.2.3 NEFCON : The Neuronal Fuzzy Controller | 42 |
| 3.2.4 EFuNN/dmEFuNN Architecture | 42 |
| 3.2.5 ANFIS: The Adaptive Network based Fuzzy Inference System | 43 |
| 3.3 ANFIS seen in detail | 45 |
| 3.3.1 Fuzzy Models | 45 |
| 3.3.2 Mamdani Model | 45 |
| 3.3.3 Takagi-Sugeno Model | 46 |

| | | |
|----------|---|-----------|
| 3.3.4 | ANFIS Layers | 47 |
| 3.3.5 | Hybrid Learning algorithm | 48 |
| 4 | Proposal | 51 |
| 4.1 | Our application scenario | 51 |
| 4.2 | Characteristic of a fuzzy system | 52 |
| 4.3 | Description of the Dataset | 54 |
| 4.4 | Our approach with parameters | 56 |
| 5 | Practical implementation and results | 61 |
| 5.1 | Challenges of the implementation | 61 |
| 5.2 | Analyzing the results | 62 |
| | Conclusions | 73 |
| | Bibliography | 78 |

List of Figures

| | | |
|-----|--|----|
| 1.1 | Igloo of Uncertainty [1] | 6 |
| 1.2 | Taxonomy of the concept of Uncertainty [1] | 7 |
| 1.3 | The three dimensions of Uncertainty [2] | 9 |
| 1.4 | The three dimensions of uncertainty along with their subcategories | 10 |
| 1.5 | Subcategories of the location dimension of uncertainty | 11 |
| 1.6 | Armour’s taxonomy of uncertainty levels [3] | 12 |
| 1.7 | Diagram showing the inversely proportional relationship between the five orders (degrees) of uncertainty and knowledge | 12 |
| 1.8 | Nature dimension of Uncertainty splitting in Aleatory and Epistemic types | 13 |
| 1.9 | Table relating the uncertainty sources in models and self-adaptive systems to the dimensions of Location and Nature of the reference taxonomy. | 14 |
| 2.1 | Block diagram of a negative feedback loop control system | 16 |
| 2.2 | Macro-categories of the most common techniques to handle uncertainty | 19 |
| 2.3 | Payoff matrix. The rows are the possible actions of player one (“Action 1, ...”) while the columns are the possible actions of player two (“Action 1, ...”). | 20 |
| 2.4 | Payoff matrix. The rows are the possible actions of player one (“Acts 1, ...”) while the columns are the different events happening in the environment (“Event 1 , ...”). | 21 |
| 2.5 | Scheme of the processing of Fuzzy Systems from input to output | 22 |
| 2.6 | Fuzzy sets of the measure of temperature. The outer-most sets (the one on the left and the one on the right) have trapezoidal shapes because the interval between 0 °C and 5 °C and the interval over 45 °C require no further precision and consequently have membership degree equal to 1. | 23 |
| 2.7 | Example of a fuzzy controller used in a feedback loop | 24 |
| 2.8 | Autonomic control loop [4] | 26 |

| | | |
|------|---|----|
| 2.9 | Full view of a Managed system controlled by the MAPE-K and interactive with the environment | 27 |
| 3.1 | Detailed view of a neural network node. | 30 |
| 3.2 | Three-layers Neural Network | 32 |
| 3.3 | Five-layers Neural Network | 36 |
| 3.4 | Cooperative Neuro-Fuzzy System | 37 |
| 3.5 | Generic Concurrent Neuro-Fuzzy System | 38 |
| 3.6 | Concurrent Neuro-Fuzzy System with inputs pre-processed by the Fuzzy System | 38 |
| 3.7 | Concurrent Neuro-Fuzzy System with inputs pre-processed by the ANN | 39 |
| 3.8 | Hybrid Neuro-Fuzzy System | 39 |
| 3.9 | FALCON Architecture Layers | 41 |
| 3.10 | GARIC Architecture Layers | 42 |
| 3.11 | Nefcon Architecture Layers, from [5] | 43 |
| 3.12 | EFuNN Architecture Layers, from [5] | 44 |
| 3.13 | ANFIS Architecture Layers, from | 44 |
| 4.1 | Some rows of the dataset | 55 |
| 4.2 | The number of active superpeers as function of time (in seconds) | 55 |
| 4.3 | Some rows of the dataset after data transformation | 55 |
| 4.4 | The number of active superpeers as function of timesteps | 56 |
| 5.1 | Real number of active Superpeers vs Predicted number of active Superpeers with outliers | 65 |
| 5.2 | Real number of active Superpeers vs Predicted number of active Superpeers without outliers | 66 |
| 5.3 | Detailed table 1 | 67 |
| 5.4 | Detailed table 2 | 69 |
| 5.5 | Comparison graph of the error of the Dynamic System and fixed Systems | 70 |
| 5.6 | Comparison barcharts of the error of the Dynamic System with the median of the total error of fixed Fuzzy Systems | 70 |
| 5.7 | Comparison barcharts of the error of the Dynamic System with the mean of the total error of fixed Fuzzy Systems | 71 |

List of Tables

| | | |
|-----|---|----|
| 3.1 | Comparison table of the characteristics of NNs and Fuzzy System . . . | 36 |
| 3.2 | Table depicting the possible kind of weights, inputs, outputs of Neuro-Fuzzy Systems and their application scenarios. | 39 |
| 3.3 | Table depicting the possible kind of weights, inputs, outputs of Neuro-Fuzzy Systems and their application scenarios. | 49 |
| 4.1 | Table depicting the possible kind of weights, inputs, outputs of Neuro-Fuzzy Systems and their application scenarios. | 59 |

Introduction

During this master thesis work we tried to mitigate the uncertainty which intrinsically resides in the models used by Self-Adaptive Systems. Self-Adaptive systems were originally conceived to be able to adapt their behavior at run-time to the uncertainty related with the happening of unexpected events using imperfect information about their environment. The presence of uncertainty showing under different dimensions and degrees implies the need of Information Technology to handle the negative effects it brings in every field of research. Since uncertainty is heavily linked with our real-existing world (and cannot be untied from it) and involves an incredible number of research fields, complex Artificial Intelligence techniques are constantly developed and frequently engineered to give birth to always better performing autonomous decision-making systems. Researchers conducted several studies regarding the classification of the different dimensions and sources of uncertainty present in Computer Science, thus leading to developing ad-hoc methodologies to tame uncertainty in the crucial decision-making process of a system. Our main goal in this work will be to try to as much as possible remove the uncertainty afflicting a time-series prediction problem by exploiting an Artificial Intelligence technique with a different “twist” on the system parameters with respect to the classical approaches. During whole **Chapter 1** we discuss and present the problem of uncertainty, starting from **Chapter 1.1** where we report the common accepted definitions of the general concept of uncertainty without lingering to our specific field of interest. In **Chapter 1.2** we focus more on the concept of uncertainty in Computer Science reporting the taxonomy of [2, D.Perez-Palacin,R.Mirandola] which illustrates, one by one, the different dimensions and sources of uncertainty in system models. In **Chapter 2** we present the State of the art methodologies which commonly represent a powerful weapon to successfully handle uncertainty. We begin (**Chapter 2.1**) by giving a quick introduction to theories to represent uncertainty such as Model averaging, Model discrepancy, Possibility Theory, Probability Theory, Interval Theory, Evidence Theory, to then dedicate two paragraphs to explain the important approaches of Game Theory (**Chapter 2.2**), linked to Probability Theory, and Fuzzy Theory (**Chapter 2.3**), linked to Possibility Theory. In the final paragraph of this chapter (**Chapter**

2.4) we introduce the MAPE-K Adaption Control Loop, presented for the first time by IBM researchers in 2005 [6] which was truly considered the main blueprint to engineer Self-Adaptive Systems through the use of feedback loops. In **Chapter 3** we move to talk about Neuro-Fuzzy Systems, starting from **Chapter 3.1** we digress about the advantages and disadvantages characterizing their founding theories, Neural Networks and Fuzzy Theory, closing the paragraph with a general presentation of this hybrid technology. After a brief introduction on the world of Neuro-Fuzzy System we dived into details (**Chapter 3.2**) dedicating a paragraph for each of the different typologies of implementation of this technology: Fuzzy Adaptive Learning Control Network (FALCON) , Generalized Approximate Reasoning based Intelligence Control (GARIC) , Neuronal Fuzzy Controller (NEFCON), Evolving Neural Fuzzy Network (EFuNN) and Dynamic Evolving Neural Fuzzy Network (dmEFuNN), Adaptive Network based Fuzzy Inference System (ANFIS). Across the paragraphs we did not limit ourselves to simply report the systems' description but we also argued their strengths and weaknesses in order to find the best suited system to fit our purpose: ANFIS. Last paragraph of this chapter, **Chapter 3.3**, is dedicated to a detailed explanation of the ANFIS architecture. From **Chapter 4** we finally start to expose the core of our project. We begin (**Chapter 4.1**) with explaining the dataset which is going to be used in our time-series prediction problem; it contains the analysis of Skype¹ SuperPeers [7] This dataset contains a set of 4000 nodes participating in the Skype SuperPeer network to which were sent an application-level ping every 30 minutes for one month, considering the state 'up' for a node at a given time if the most recent ping succeeded. Moving on the structure of the thesis in **Chapter 4.2** we present the three available ANFIS clustering types, Grid Partition, Subtractive Clustering and Fuzzy C-Means Clustering. From this three-valued set we choose the Grid Partition type motivated by its relatively low number of different parameters and parameters' values compared to the other clustering methodologies. With **Chapter 4.3** we arrive to the core of this project: the diversified approach we undertook with the system tunable parameters characterizing the Fuzzy Logic side of the Neuro-Fuzzy system. Usually in each Fuzzy Logic implementation, especially in the case of Neuro Fuzzy Systems, there's the need of an Expert User which thanks to his experience and through the use of the time-expensive and inefficient trial and error method, understands how the system behaves to the changing values of its parameters consequently finely tuning them. Throughout this paragraph we exhaustively explain the considered parameters whose values cannot be labeled "good" or "bad" taken individually but they are rather tied with each other consequently lead-

¹Skype is a proprietary freeware for instantaneous messaging and Voice over IP. It merges the usual characteristics of chatting, file transfer, etc., to a phone call system based on a Peer-to-Peer network.

ing to the problem of finding a good combination of values more than just individual ones. The problem gets now divided in two parts:

- How do we evaluate a combination as a good or a bad one?
- How do we find this combination?

The first part of the problem is faced by basing our project on an idea: by performing online learning we thought that it is likely probable that the system with the combination of fuzzy parameters which better performed in the prediction of the last timestep output will also perform best in the prediction of the current timestep output. Having settled this, for the second part of the problem we focused on the idea that many times there's no fuzzy logic human expert to refer to so we created a system that trains simultaneously 185 different combination of fuzzy parameter values, dynamically switching to the best fuzzy combination for each current timestep prediction. The evaluation of the best fuzzy parameters combination values is based on the answer to the first part of the problem written above. **Chapter 5** is dedicated to the implementation of our dynamic system including an explanation of the challenges we faced (**Chapter 5.1**) and a complete analysis of the results we obtained through this work (**Chapter 5.2**). The last part of this thesis is as usual dedicated to the conclusions of this project and the future works.

Chapter 1

Importance of the problem of Uncertainty

1.1 What is Uncertainty?

Uncertainty has been studied in many fields during the last few centuries. Discussions on uncertainty are frequently encountered in fields like: decision sciences, artificial Intelligence, legal fact-finding, economics, medical science, organizational open system theory, psychology, physics, etc. Uncertainty touches almost every aspect of life, showing itself when we make decisions that have consequences which we cannot predict. When there's uncertainty there will consequently be risks associated with it. For example the common lack of certainty about the weather leads to the risk of getting wet if we leave the house without an umbrella; the uncertainty about the stock market instead implies the risk of a money loss as the consequence of a poor investment. Of course, there are some decisions that carry way more severe risks than getting wet or losing money; in our modern world the decision to approve a new drug or also a new chemical compound can have far-reaching consequences for health, the environment, society and economies. There are cases where uncertainty puts lives, either human or animal forms, are at stake and decision-making and the handling of uncertainties acquire important ethical dimensions. The most intuitive way to deal with these problems is to diminish uncertainty by acquiring knowledge of the issue. But what if, no matter how much we gather information, there are certain topics for which an intrinsic uncertainty can't be rooted out? This is exactly what researchers nowadays are struggling with. Any scientist knows that knowledge is never absolutely complete, and research can do no more than producing the results of scientific observations, which, by being 'observations of things' can differ from 'things as such'.

We show below the 'igloo of uncertainty', figure 1.1, presented in the very inter-

1. Importance of the problem of Uncertainty

esting article [1, Tannert et al], which was partly inspired by [8].

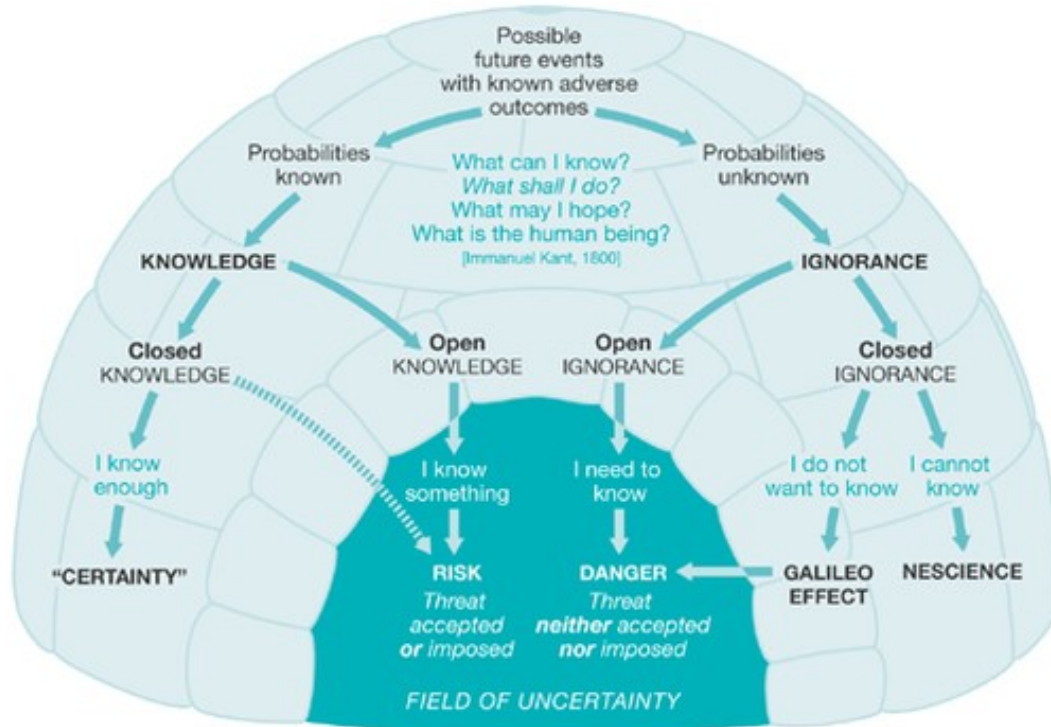


Figure 1.1: Igloo of Uncertainty [1]

In their article Tannert, Elvers, Jandrig deeply focus their attention on the potential dangers arising from the presence of different forms of uncertainty and the consequent ethical duty of research to prevent these hazards.

In the igloo the authors mainly distinguish between open and closed forms of both ignorance and knowledge basing this distinction on the concept of probability. Let's keep going through the schema by mentioning their own words:

“Within that framework, dangers are defined in terms of the possible outcomes of a given situation. To understand the potential adverse effects of a decision, we therefore require an approximation of the quality of dangers in any given event. Consequently, a rational approach is to give an estimate of the probability that the respective event will happen, and to assess the hazard and the possible impact of the event.”

Although research is able to fill some of the gaps caused by uncertainty, Tannert, Elvers and Jandrig claim that ignorance presents a greater challenge. The authors discern two kinds of ignorance, *open ignorance*, caused by lack of knowledge which can be acquired with research and study, and *closed ignorance* (or *nescience*), which cannot be reduced owing to stochastics and the randomness of the matter under study [9].

To get the full picture on the different facets Tannert, Elvers, and Jandrig created a taxonomy (1.2) that recognizes two fundamental forms of uncertainty: *Objective Uncertainty* and *Subjective Uncertainty*.

The former further splits in Moral Uncertainty and Rule Uncertainty and is originated from the presence of gaps in knowledge which can be filled by research. The latter splits in *Ontological Uncertainty* and *Epistemological Uncertainty* and is originated from the inability to apply appropriate moral rules.

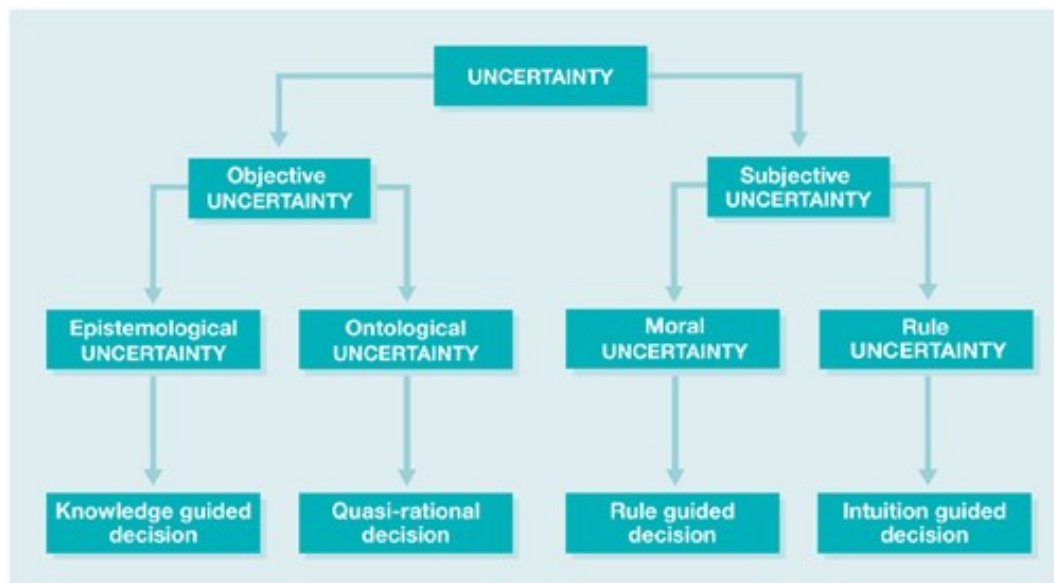


Figure 1.2: Taxonomy of the concept of Uncertainty [1]

“Uncertainties challenge the central claim of science: that all problems are presumed to be solvable by research.”

The ethics of uncertainty - In the light of possible dangers, research becomes a moral duty - Christof Tannert, Horst-Dietrich Elvers, Burkhard Jandrig

There coexist different views and thoughts about uncertainty which are all related with the nature of problems in different fields. According to Weber [10] [11] these views are epistemologically biased. He states:

“the concept of uncertainty is epistemologically biased, in that uncertainty is viewed as an attribute of how we know what we know. This epistemological bias has led to the development of four branches of uncertainty literature based on an actor’s (individual, group, or organization): (1) ability to gather and process information; (2) ability to predict consequences of actions; (3) use of intuition; or (4) perception of the environment.”

Klin and Yuan in their work [12] try to assess uncertainty with fuzzy logic and define three basic types of uncertainty: nonspecificity, strife and fuzziness. According to Walker’s theory [13] author who focused on the uncertainty residing in Model-Based Decision Support Systems, uncertainty has three dimensions in legal fact-finding. These are linguistic, logical and causal dimensions. Walker then classified the uncertainties into six types for scientific evidence about generic causation: concept uncertainty, measurement uncertainty, calculation uncertainty, sampling uncertainty, mathematical modeling uncertainty and causal uncertainty.

1.2 Uncertainty in Computer Science

After going through the general concept of uncertainty we now move to discuss uncertainty in engineering, computer engineering more specifically. First things first, let’s give a former definition of uncertainty by using the famous used one of [14, Walker et al.]:

Definition of Uncertainty:

“Any deviation from the unachievable ideal of completely deterministic knowledge of the relevant system”

We’ve already seen a first taxonomy on the generic concept of uncertainty with the one proposed by [1] therefore we now consider a taxonomy which uniquely considers the uncertainty residing in models in the computer science field. The reference taxonomy is the one proposed by [2, D.Perez and R.Mirandola] which combines the concept of different dimensions of uncertainty proposed in [14] with the concept of the existence of different orders of uncertainty proposed in [3]

In [14] the authors make a distinction between uncertainty due to lack of knowledge and uncertainty due to variability inherent to the system under consideration. In agreement with other uncertainty experts they proposed the concept of three di-

mensions of uncertainty related to model-based decision support systems (figure 1.3). These are:

- i the location of uncertainty – where the uncertainty manifests itself within the model complex;
- ii the level of uncertainty – where the uncertainty manifests itself along the spectrum between deterministic knowledge and total ignorance;
- iii the nature of uncertainty – whether the uncertainty is due to the imperfection of our knowledge or is due to the inherent variability of the phenomena being described.

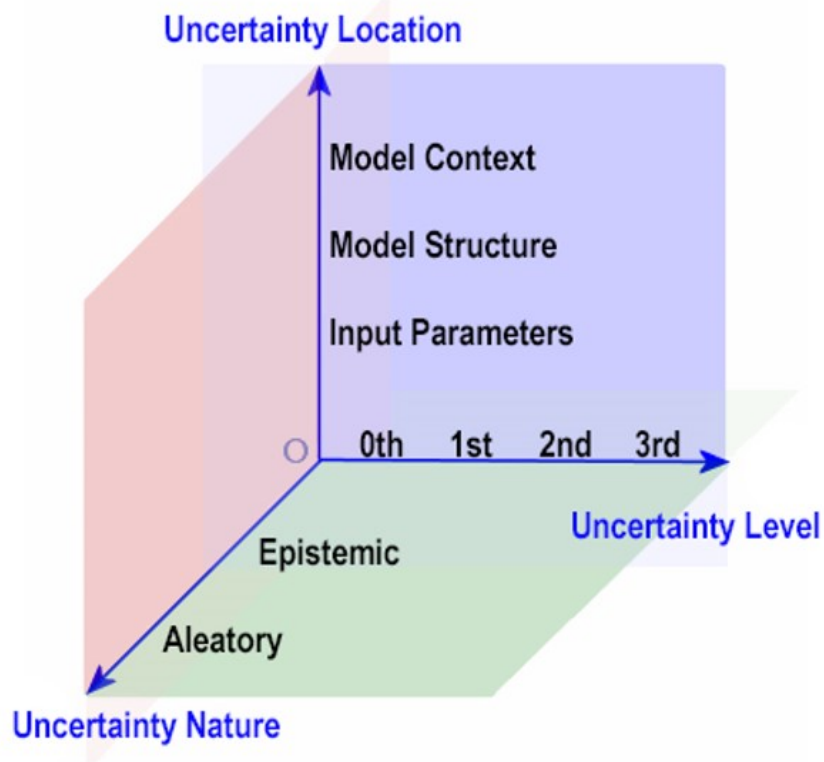


Figure 1.3: The three dimensions of Uncertainty [2]

We are now going to explain each of these dimensions and their subcategories

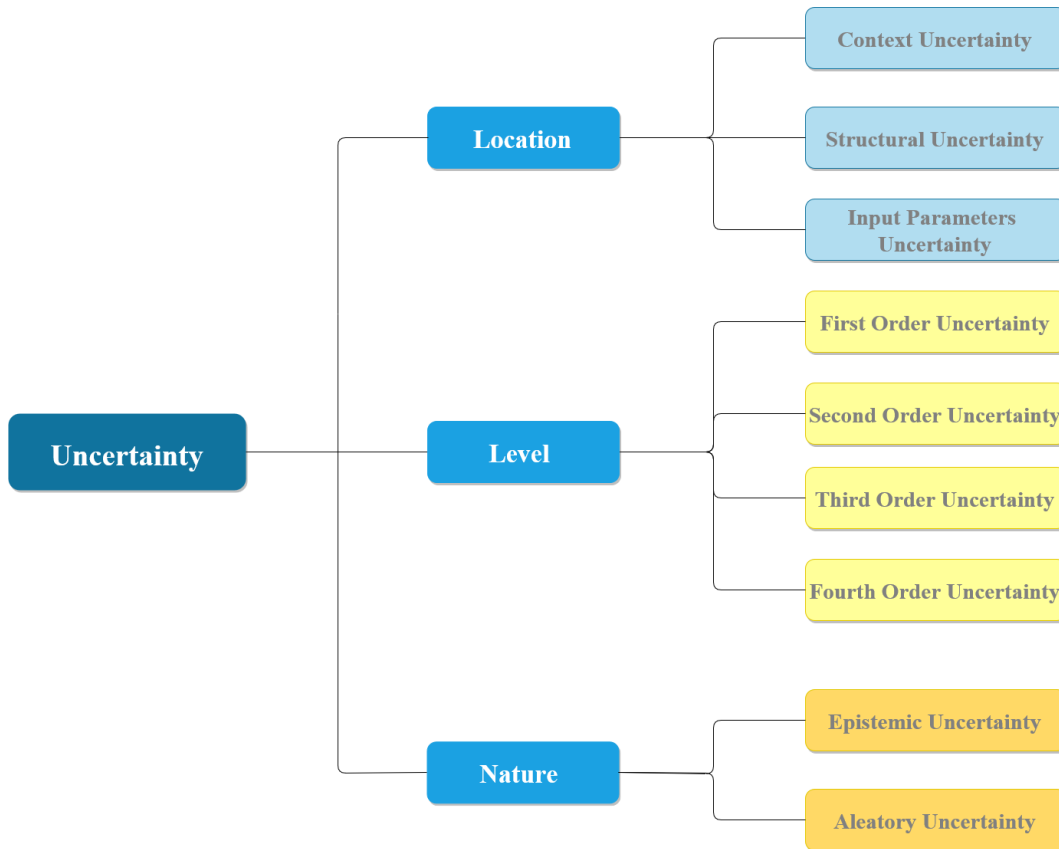


Figure 1.4: The three dimensions of uncertainty along with their subcategories

Location

The *Location* of uncertainty is an identification of where uncertainty manifests itself within the whole model complex. An uncertainty can be located in different sections of a model, according to these we have:

- *Context uncertainty* which is an identification of the boundaries of the system to be modelled, and thus the portions of the real world that are inside the system, the portions that are outside, and the completeness of its representation.
- *Model uncertainty* which is associated with both the conceptual model (i.e., the variables and their relationships that are chosen to describe the system located within the boundaries and thus constituting the model complex) and the computer model. Model uncertainty can, therefore, be further divided into two parts: model structure uncertainty, which is uncertainty about the form of the model itself, and model technical uncertainty, which is uncertainty arising from the computer implementation of the model.

- *Input parameters uncertainty* which is often identified as parameter uncertainty and it is associated with the actual value of variables given as input to the model and with the methods used to calibrate the model parameters.

Walker then reports an additional section who doesn't really bond to a specific location:

- Model Outcome Uncertainty which is the accumulated uncertainty caused by the uncertainties in all the above locations (context, model, and input parameters) that are propagated through the model and are reflected in the resulting estimates of the outcomes of interest. It is sometimes called prediction error, since it is the discrepancy between the true value of an outcome and the model's predicted value.

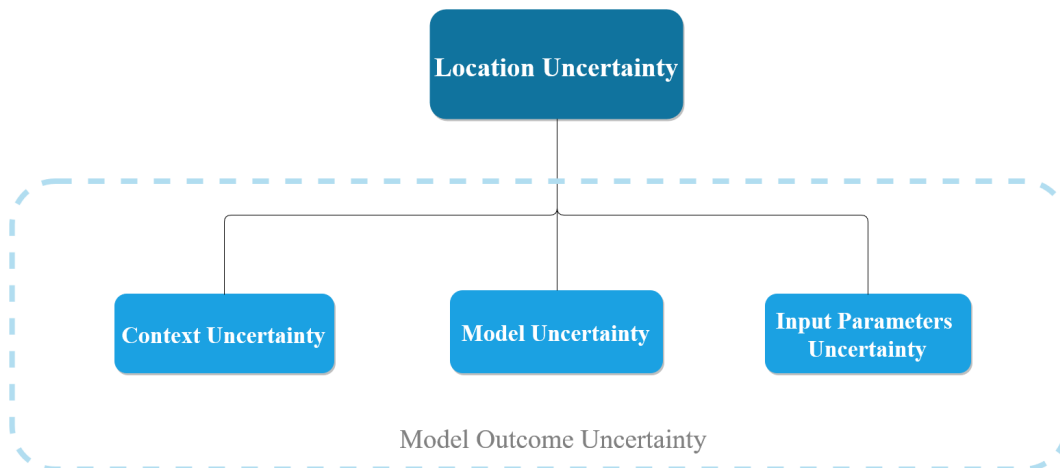


Figure 1.5: Subcategories of the location dimension of uncertainty

Levels of Uncertainty

“The level of uncertainty is where the uncertainty manifests itself along the spectrum between deterministic knowledge and total ignorance.”

Uncertainties in the Modeling of Self-adaptive Systems: a Taxonomy and an Example of Availability Evaluation - D. Perez-Palacin, R. Mirandola

While Walker's proposed a scale of uncertainty levels based of how much knowledge lacks to achieve the knowledge necessary for studying the system deterministically (figure 1.6), we'll refer to the taxonomy composed of five levels of ignored presented for the first time in [3].

The five proposed levels of ignorance, which become of uncertainty, are:

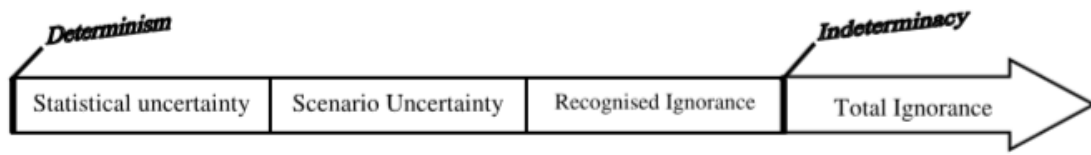


Figure 1.6: Armour's taxonomy of uncertainty levels [3]

- 0th order of uncertainty. Lack of uncertainty, i.e., knowledge.
- 1st order of uncertainty. Lack of knowledge. The subject lacks knowledge about something but she is aware of such lack (i.e., known uncertainty).
- 2nd order of uncertainty. Lack of knowledge and lack of awareness. The subject does not know that she does not know.
- 3rd order of uncertainty. Lack of process to find out the lack of awareness. The subject does not have any way to move from not knowing that she does not know to, at least, be aware of the existence of the uncertainty.
- 4th order of uncertainty. Meta-uncertainty. Uncertainty about orders of uncertainty.

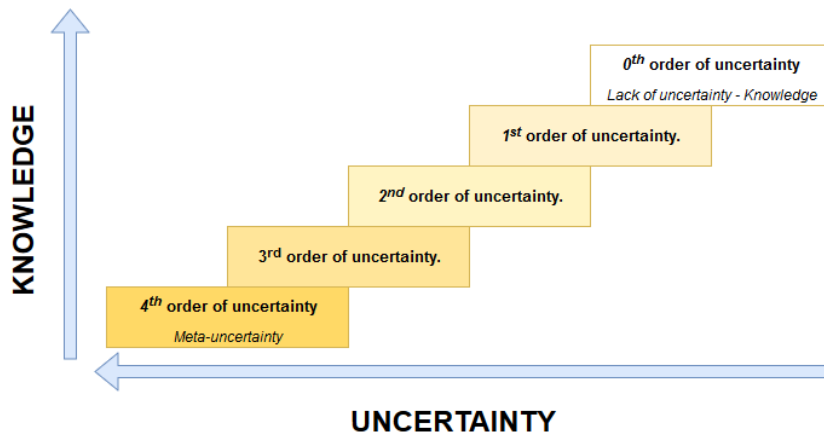


Figure 1.7: Diagram showing the inversely proportional relationship between the five orders (degrees) of uncertainty and knowledge

Nature of Uncertainty

We now introduce the third dimension of the concept of uncertainty: the *nature of uncertainty*. The nature of uncertainty refers to whether the uncertainty is due to the imperfection of the acquired knowledge or is due to the inherent variability of

the phenomena being described. An important feature of the nature of uncertainty is the distinction between two extremes:

- *Epistemic uncertainty*: The uncertainty due to the imperfection of our knowledge, which may be reduced by more research and empirical efforts.
- *Aleatory uncertainty*: The uncertainty due to inherent variability, which is especially applicable in human and natural systems and concerning social, economic, and technological developments.

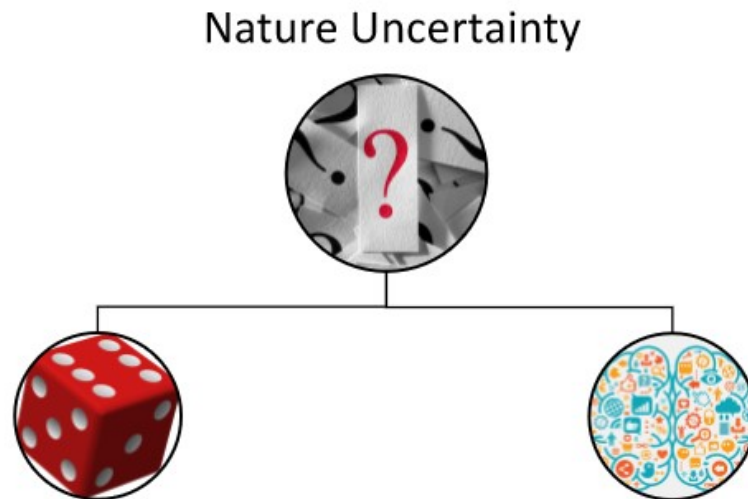


Figure 1.8: Nature dimension of Uncertainty splitting in Aleatory and Epistemic types

Source of Uncertainties in Models

As things are right now, model outcome uncertainty (which is the resulting output uncertainty) is unavoidable for each model so self-adaptive systems should be aware of the uncertainties contained in their models and consequently apply specific methods to fix them during the model analysis phase. We include here the classification, reported in [2], of the sources of uncertainties discerning the effect they entail in the model managed by the self-adaptive system. Each source of uncertainty is classified according to its location and nature dimensions of the taxonomy while the level dimension is not reported as depends on the implemented capabilities in the system. Perez's and Mirandola's intent aims to classify the different sources with a taxonomy, instead of using lists, in order to help the choice of general approaches

1. Importance of the problem of Uncertainty

that can deal with a group of uncertainties concurrently, rather than using a specific approach at a time for each different uncertainty.

| Source of Uncertainty | Classification | |
|---------------------------------|--|---------------------------|
| | Location | Nature |
| Simplifying assumptions | <i>Structural/Context</i> | <i>Epistemic</i> |
| Model Drift | <i>Structural</i> | <i>Epistemic</i> |
| Noise in sensing | <i>Input parameters</i> | <i>Epistemic/Aleatory</i> |
| Future parameters value | <i>Input parameters</i> | <i>Epistemic</i> |
| Human in the loop | <i>Context</i> | <i>Epistemic/Aleatory</i> |
| Objectives | <i>Input parameters/Context</i> | <i>Epistemic</i> |
| Decentralization | <i>Structural/Context</i> | <i>Epistemic</i> |
| Execution Context / Mobility | <i>Structural/Context/Input parameters</i> | <i>Epistemic</i> |
| -Cyber-physical system | <i>Context/Structural/Input parameters</i> | <i>Epistemic</i> |
| Automatic learning | <i>Structural/Input parameters</i> | <i>Epistemic/Aleatory</i> |
| Rapid evolution | <i>Structural/Input parameters</i> | <i>Epistemic</i> |
| Granularity of models | <i>Context/Structural</i> | <i>Epistemic</i> |
| Different source of information | <i>Input parameters</i> | <i>Epistemic/Aleatory</i> |

Figure 1.9: Table relating the uncertainty sources in models and self-adaptive systems to the dimensions of Location and Nature of the reference taxonomy.

Chapter 2

State of the art methodologies to face uncertainty

2.1 Common techniques to handle uncertainty

All these uncertainty factors seen in the previous chapters, location, degree, nature, challenge the confidence that should be associated with the decision-making process implemented in a system and therefore needs to be explicitly captured in the engineering process of self-adaptive systems. Nowadays the need for software systems that are able to self-adapt dynamically has considerably increased and has given rise to several research initiatives to tackle uncertainty in different ways. We present the following techniques, already introduced in the paper [2] and thoroughly explained in [15]:

- *Model Averaging*
- *Model Discrepancy*
- *Interval Analysis*
- *Evidence Theory*
- *Probability Theory*
- *Possibility Theory*

Let's go through them one by one.

Model Averaging: This approach proposes the use of multiple models of the same system which can also be defined by adopting different modeling languages and model domains. The core idea is that all the proposed models should be plausible as final

model while being in competition between themselves. The final model will then be obtained by averaging the original models which are all assigned a probability or a weight. In case there's a probability assigned to each model it represents the likelihood of the relative model of being the "true" model, i.e. a measure of model adequacy. We can show an example of the formula needed to compute the best value (according to model averaging) of a parameter α .

$$Average(\alpha) = \frac{\sum_{i=1}^{Number\ of\ Models} (\alpha_i * w_i)}{\sum_{i=1}^{Number\ of\ Models} (w_i)} \quad (2.1)$$

Model discrepancy: This approach assumes that the utilized model is not the "true" model of the system. In fact, this method tries to measure the discrepancy between the implemented model and the target ideal one. The discrepancy is a measure which explains how far our current model deviates from the ideal one and is modeled by adding a "discrepancy term". Ideally, the discrepancy term is equal to the difference between the output of the model running at its best (i.e., where the input values of the model are all equal to the real values) and the real results (the true target quantity). This is the concept idea which stands behind the known "feedback loop" broadly used in automation and control engineering. In the figure below we have an example of closed loop control; the feedback loop is used to monitor a system variable in order to measure the deviation from the desired output and consequently adapt the system to reduce this deviation to zero.

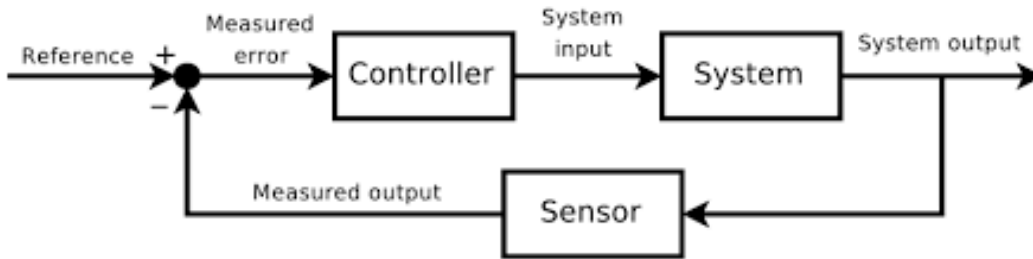


Figure 2.1: Block diagram of a negative feedback loop control system

We now move to present the remaining four techniques based on mathematical theories created to represent uncertainty. The last two theories, probability and possibility theory are the most common general techniques to estimate uncertainty.

Interval Analysis: it is based on the assumption that for a given variable x_i its set of possible values (interval) is known with no specified uncertainty structure on the distribution of values within the interval. Thus, all that is assumed to be known

about x_i is that its value is contained within the set of values X_i .

Evidence Theory: this theory, also known as Dempster-Shafer theory, provides a different representation of uncertainty based on the use of a triple (X_i, α_i, f_i) for a given variable x_i , with X_i being the set of possible values for x_i , α_i being a countable collections of subsets of X_i and f_i being a function defined for subsets β of X_i such that:

$$\begin{cases} f_i(\beta) > 0 \text{ if } \beta \in \alpha_i \\ f_i(\beta) = 0 \text{ if } \beta \notin \alpha_i \end{cases}$$

Two more measures of likelihood for subsets of X_i complete this formalism: plausibility and belief. Specifically, for a subset γ of X_i they are defined as

$$Pl_i(\beta) = \sum_{\beta \cap \gamma \neq \emptyset} f_i(\beta) \quad (2.2)$$

and

$$Bel_i(\beta) = \sum_{\beta \subset \gamma} f_i(\beta) \quad (2.3)$$

The plausibility provides a measure of the amount of information that could possibly be associated with γ , belief provides instead a measure of the amount of information that is known to be associated with γ . Given a variable Z , these two measures are linked by the following relation:

$$Plausibility(Z) \leq Belief(Z) \quad (2.4)$$

Once all the beliefs from the different sources have been collected they can be fused together by means of a specific rule of combination, also called fusion operator, such as the well known Dempster's rule of combination. Further details of this theory can be found in [16]

Probability Theory: this theory originated as the branch of mathematics concerned with the study of random phenomena and embodies the most widely used

approach to represent uncertainty. Similarly to evidence theory, it is based on the specification of a triple (X_i, α_i, p_i) for a variable x_i , with X_i being the set of possible values for x_i , α_i being a suitable restricted collection of subsets of X_i and p_i defining the probability for elements of X_i .

The terminology often refers to them this way: X_i is the sample space, the elements of α_i are events, and p_i is a probability measure. Unlike possibility theory and evidence theory, which provide two measures, probability theory provides only one measure of likelihood: probability. Probability, as the name suggests, is the measure of the likeliness that an event will occur, quantified as a number in the real interval $[0,1]$ (where 0 indicates impossibility and 1 certainty). The use of this representation of uncertainty gave birth to a new kind of programming called stochastic programming. Unfortunately to adopt this coding technique there is the need of probability to follow a normal distribution and most of the times this is not the case of Self-Adaptive systems. Within the macro-category of probability theory different approaches have been developed:

- *Bayesian Theory*, based on subjective interpretation of the probability. In this interpretation the probability is defined as an expression of a rational agent's degrees of belief about uncertain propositions.
- *Frequentist Theory*, whose interpretations of random phenomena employ information relative to the frequencies of past actual outcomes to derive probabilities that represent the likelihood of possible outcomes for future events. This interpretation of probability is widely employed to deal with different sources of uncertainty in self-adaptive systems.
- *Game Theory*, which is the study of mathematical models of conflict and cooperation between intelligent rational decision-makers. This approach is better explained in the next chapters.

Possibility Theory: it provides a representation for uncertainty that permits a more structured representation than interval analysis, based on the specification of a pair (X_i, δ_i) for a variable x_i , with X_i being the set of possible values for x_i and δ_i a function defined on X_i such that:

$$0 \leq \delta_i(x_i) \leq 1 \text{ for } x_i \in X_i \text{ and } \sup \{ \delta(x_i) : x_i \in X_i \} = 1 \quad (2.5)$$

The function δ_i provides a measure of the amount of "confidence" that is assigned to each element of X_i and is referred to as the possibility distribution function for

2.2 Game Theory

Game theory is defined as “the study of mathematical models of conflict and cooperation between intelligent rational decision-makers”. This theory was conceived to be applied in situations where there are multiple decision makers (*players*) who have a variety of sequences of actions (*strategies* or *policies*) to employ in order to achieve a particular outcome for themselves (*payoff*). Game theory has been applied in a wide variety of fields, such as, Economics, Biology, and Computer Science to study systems that exhibit competitive behavior as well as a range of scenarios that might include cooperation. In the former case we have ‘normal’ games and zero-sum games which can be both represented in *strategic* or *extensive form*. The peculiarity of zero-sum games is that the positive payoff of a player is in sense by the loss of the other players. Let’s explain it with a practical example using two players and representing a game in strategic form:

| | | PLAYER TWO | | | |
|---|------------|-----------------|-----------------|-------|-----------------|
| | | Action '1' | Action '2' | | Action 'h' |
| P L A Y E R O N E | Action '1' | Payoff('1','1') | Payoff('1','2') | | Payoff('1','h') |
| | Action '2' | Payoff('2','1') | Payoff('2','2') | | Payoff('2','h') |
| | | | | | |
| | Action 'k' | Payoff('k','1') | Payoff('k','2') | | Payoff('k','h') |

Figure 2.3: Payoff matrix. The rows are the possible actions of player one (“Action 1, ...”) while the columns are the possible actions of player two (“Action 1, ...”).

A game in strategic form is traditionally described by a payoff matrix, where the rows correspond to the strategies of the first player and the columns correspond to the strategies of the second player. In the case of Zero-Sum games the real number $Payoff('Action\ 2', 'Action\ 1')$ is simply the payoff that the first player receives from the second player (or gives to the second player if this value is negative) if the first player chooses strategy “i” and the second player chooses strategy “j”. This means that with player 2 performing 'Action 1' and player 1 performing ('Action 2', player 1 will get a payoff of η and player 2 will get a payoff of $-\eta$, with η being a real number. This was an example of a competitive game; when players in a coalition coordinate to choose joint strategies by agreement (consensus) to have a better payoff for each of them, we have the so-called cooperative games. Further details for this kind of games can be found in [19]

The approach to handle uncertainty using Game Theory performs controller synthesis based on a (non-stochastic) zero-sum game played between the system and the environment. In the payoff matrix we now have the actions performable by the system and the events happening in the environment.

| | Event '1' | Event '2' | ... | Event 'h' |
|----------|---------------|---------------|-----|---------------|
| Acts '1' | Payoff (1, 1) | Payoff (1, 2) | ... | Payoff (1, h) |
| Acts '2' | Payoff (2, 1) | Payoff (2, 2) | ... | Payoff (2, h) |
| ... | ... | ... | ... | ... |
| Acts 'k' | Payoff (k, 1) | Payoff (k, 2) | ... | Payoff (k, h) |

Figure 2.4: Payoff matrix. The rows are the possible actions of player one (“Acts 1, ...”) while the columns are the different events happening in the environment (“Event 1 , ...”).

In general every event has associated probabilities assigned. These probabilities are assumed to represent the degree of knowledge (or lack of knowledge) about states of environment. Every action then has a value that represents expected utilities (or expected payoffs) for decision makers. The rational decisions are supposed to be the ones that maximize expected utilities. Now, the idea of maximizing the expected utility exposes the following question: what is the best action or sequence of actions (also called policy) to perform in order to maximize the utility?

The ‘best policy’ depends on the different solution concepts, the famous ones are the following : *Nash Equilibrium*, *Dominant Strategy*, *Pareto Optimality*, *Social Welfare*, *Backward Induction*, *Forward Induction*, *Perfect Bayesian Equilibrium*

Beware that some of these concepts of solution only exists in specific types of games. The theory behind the auctions and negotiations among autonomous agents heavily depends on Game Theory concepts; an application example is the auctions held for selling advertising spaces sold by search engines and websites.

2.3 Fuzzy Logic

The first appearance of Fuzzy Logic and Fuzzy Sets happens in Zadeh’s work of 1965, “Fuzzy Sets” [20]. The latter work and “Fuzzy Sets and systems” [21] represent vast volumes of literature explaining fuzzy logic and fuzzy system modeling (FSM) in particular. Zadeh’s intention was to create a methodology that resembled the reasoning of humans; after all what better kind of thinking than the one of the creators of the concept of uncertainty does exist to handle uncertainty? It all makes

2. State of the art methodologies to face uncertainty

sense that, our abilities, capabilities, intuitions and perceptions are the real ingredient in this subject. As Zadeh wrote many times in his works, Fuzzy logic is creating an approximate reasoning mechanism to handle the uncertainty associated with human perception. Moreover Zadeh's words declare on fuzzy sets logic:

"The notion of a fuzzy set provides a convenient point of departure for the construction of a conceptual frame-work which parallels in many respects the framework used in the case of ordinary sets, but is more general than the latter and, potentially, may prove to have a much wider scope of applicability, particularly in the fields of pattern classification and information processing. Essentially, such a framework provides a natural way of dealing with problems in which the source of imprecision is the absence of sharply defined criteria of class membership rather than the presence of random variables."

Let us now schematize in a figure the process leading from input to output in a fuzzy system :

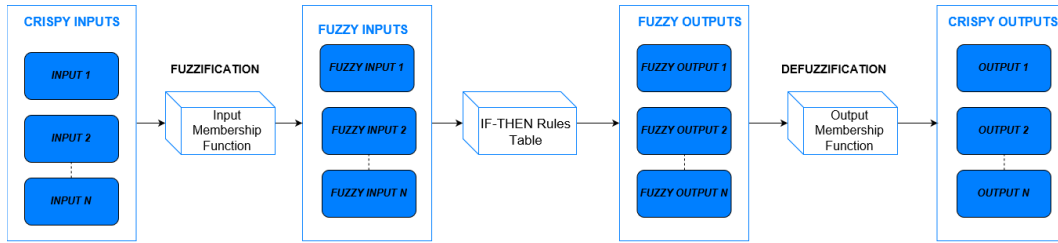


Figure 2.5: Scheme of the processing of Fuzzy Systems from input to output

The original inputs, which are still of type crisp (value 0 or 1), undergo a fuzzification process in which aggregation operators and membership functions are used to finally make these inputs belong to Fuzzy Sets, i.e. sets whose elements have degrees of membership spacing in the interval $[0,1]$. Fuzzy Sets can be applied to domains where the information is incomplete or imprecise. For instance, fuzzy sets have been used in linguistics to deal with vagueness and ambiguity of the statements. The classically used example is the one of temperature. Temperatures are not considered to be cold and warm are not uniquely defined and they may be different from person to person. Sometimes, there are temperatures that can be considered both cold and warm to some extent.

In fuzzy theory, every element belongs to a concept class, let's say A, to a partial degree:

$$\mu_A : X \rightarrow [0, 1], \mu_A(x) = a \in [0, 1], x \in X \quad (2.6)$$

where μ_A is the membership assignment of an element 'x' to a concept class A in a proposition. The membership assignment μ_A is one of the many *membership functions* functions used in a fuzzy-logic based system, this type of functions maps a crisp input to a fuzzy input and fuzzy output. The value $\mu_A(x)$ is called *membership degree* and quantifies the grade of membership of element x to the fuzzy set A. A value of 0 means that x is not a member of the fuzzy set in question, a value of 1 indicates the fully membership of x to the fuzzy set in question while values between 0 and 1 characterize members which belong to the fuzzy set in question only partially.

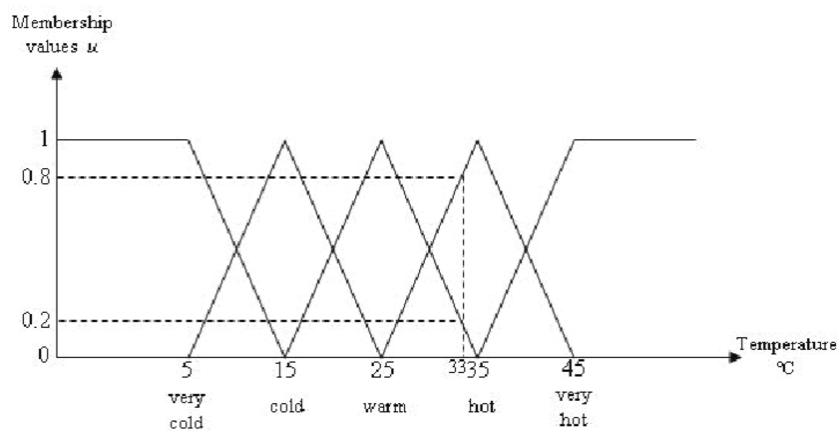


Figure 2.6: Fuzzy sets of the measure of temperature. The outer-most sets (the one on the left and the one on the right) have trapezoidal shapes because the interval between 0°C and 5°C and the interval over 45°C require no further precision and consequently have membership degree equal to 1.

The above representation is generally accepted as Type-1 fuzzy sets which assumes the membership values are certain. Unfortunately most of all concepts in fuzzy theory are assumed to be definable to be true to a degree as in most of the situations the uncertainty in a system may not be captured by Type-1 fuzzy sets. Zadeh few years later from his first work on Fuzzy Theory introduced Type-2 fuzzy sets as an extension of Type-1 fuzzy set. Zadeh's Type-2 fuzzy sets are fuzzy sets whose membership functions are classified as Type-1 fuzzy sets. This means that way the value of membership function becomes fuzzy so that these values are true to a degree. Type-2 fuzzy sets have grades of membership that are themselves defined by Type-1 membership functions, which are called secondary membership functions. Fuzzy sets of Membership functions of type-1 are two-dimensional, whereas membership functions of type-2 fuzzy sets are three-dimensional. It is this new third-dimension of type-2 fuzzy sets that provides additional degrees of freedom that make it possible to directly model uncertainties. After explaining the process of fuzzification we now

expose the relationship between input and output variables. Fuzzy system models (FSM) based of Fuzzy Logic define relationships between input and output variables of a system by using linguistic labels in a table of IF-THEN rules. Usually Mamdani and Takagi-Sugeno are the most commonly adopted rule-based approaches. These two approaches are discussed in the next chapters.

Let us define a k-th information vector where in the number of attributes on information.

$$X_k = x_{1k}, x_{2k}, \dots, x_{nk} \quad (2.7)$$

Then the reasoning for this information vector is a rule which generally is written as:

$$\begin{aligned} & \text{IF } x_1 \in X_k \text{ isr } A_{1j} \text{ AND } x_2 \in X_k \text{ isr } A_{2j} \text{ AND...AND} \\ & \quad x_n \in X_k \text{ isr } A_{nj} \text{ THEN } y \in Y_k \text{ isr } B_j \end{aligned}$$

where A_{ij} and B_j are linguistic assignments for input and output information objects, respectively, for the j-th rule of the whole number of rules in the fuzzy rule base.

A trivial practical example for a heating system could be:

$$\begin{aligned} & \text{IF temperature is TOO HIGH AND heating is ON} \\ & \quad \text{THEN turnoff heating is TRUE} \end{aligned}$$

which turns off the already-on heating system when the temperature is classified as too high.

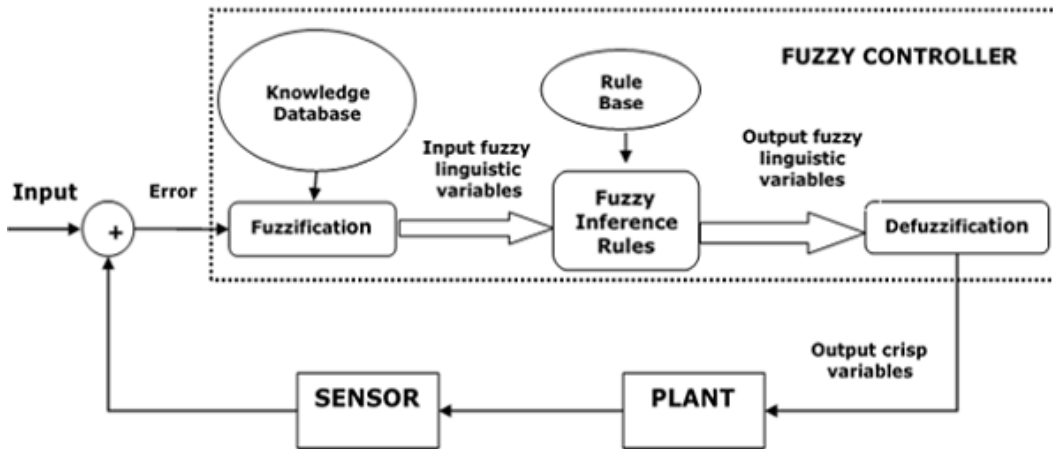


Figure 2.7: Example of a fuzzy controller used in a feedback loop

There are at least two advantages of Fuzzy systems models that attract the attention in the researchers' world:

- its power of linguistic explanation with resulting easy of understanding
- its tolerance to imprecise data which provides flexibility and stability for prediction and decision making

These are the two most important factors which led to a massive use of Fuzzy Logic in fields like Controls and Automation Engineering, Artificial Intelligence, Robotics, Economics and many others more. At last, let me quote a Fuzzy Logic application case scenario presented in a thesis named as “Dealing with uncertainties in availability models using fuzzy logic and neural networks”, written at Politecnico di Milano by my former colleague and friend Francesco Marchesani, by Professor R.Mirandola and Ph.d D. Perez-Palacin [22]. This work showed how Fuzzy Logic is a key point for modeling and consequently taming uncertainty in self-adaptive systems. Here is the case scenario as described by Mirandola and Perez: “Consider a software application whose functionality is the viewing video in streaming (real-time events, films, etc) meet its mission, it requires services that are offered by third-party service providers over the Internet; e.g., streaming video servers. The mission is accomplished when the user can watch the complete video, whose expected duration is published and hence it is known by the application in advance. As there may be multiple providers for each required service, to increase the application’s quality, it will be engineered with self-adaptive capabilities in terms of dynamic service provider selection; e.g., if it’s using a service provider and it becomes unavailable, the application will be able to autonomously zap to bind another different provider.” The project of Marchesani exposes the importance of the concept of Fuzzy Sets: here they are used to model the uncertainty present in the measurements/estimates of Availability in each of the all components of the environment considered (mobile antenna, WiFi, internet provider, server, etc.). The uncertainty analysis will be replicated for each of the three macro-categories of the Location uncertainty dimension (Input Parameters, Structure and Context) and for both the Aleatory and Epistemic kinds of nature of uncertainty. Several uncertainty models are computed starting by different assumptions and are there combined in a final model using the Model Average technique.

2.4 MAPE-K

We report here one of the most significative reference model for autonomic and self-adaptive systems, the MAPE-K Adaption Control Loop . The original idea appeared for the first time in the paper published by the IBM researchers in [6]. Later

on, the concept idea of an Adaption Control Loop was borrowed to the field of self-adaptive systems by Brun in [23]. Let's start by introducing how a feedback loop is structured as reported in the just mentioned work of Brun.

A feedback loop typically involves four key activities: collection phase, analysis phase, decision phase, and action phase. During the collection phase sensors or probes collect data from the system in execution including its context about its current state. The gathered data are then cleaned, filtered, pruned and finally, stored for future reference to depict an accurate model of past and current states. In the diagnosis phase the controller then analyzes the data to infer trends which are going to be used to predict the future and identifies symptoms which will underline possible errors/noises to be prevented. Subsequently, after the planning module attempted to predict the future and decided how to act these actions take effect into the executing system and its context through actuators or effectors. This generic model of a feedback loop is often referred to as the autonomic control loop, the details are shown in 2.8. This model is a refinement of the AI community's sense-plan-act approach of the early 1980s to control autonomous mobile robots.

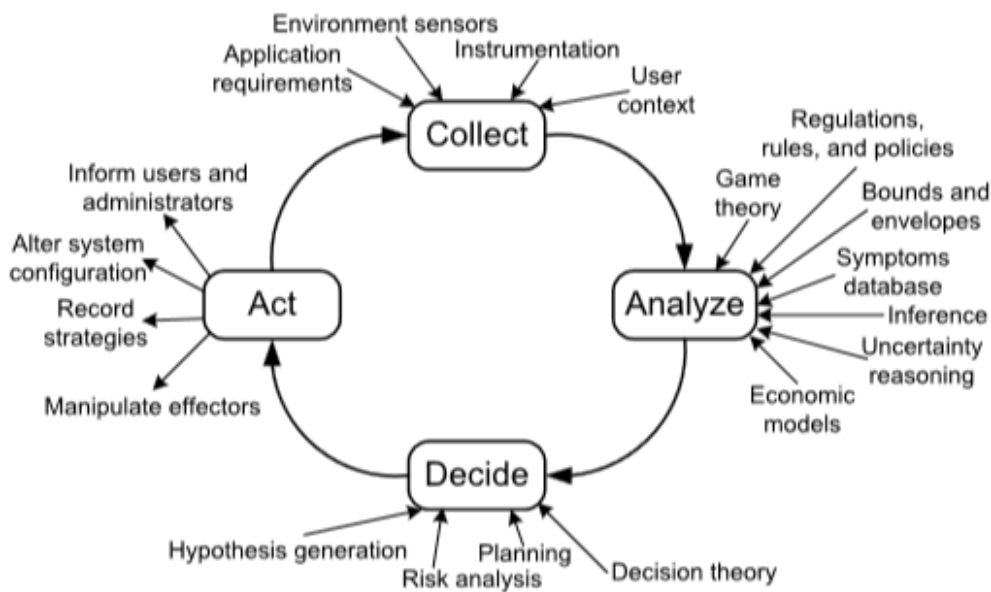


Figure 2.8: Autonomic control loop [4]

To summarize, closed loop systems are instrumented with monitors of sensors and with reconfiguration actions or actuators which are related by a control and decision component, which is what implements the dynamic adaptation policy or strategy. All this can be implemented with the MAPE-K approach, with sub-components for Analysis of Monitored data, Planning of response actions, Executions of these actions, all of them based on a Knowledge representation of the system under ad-

ministration.

It is time to see in details the MAPE-K (Monitor-Analyze-Plan-Execute over a shared Knowledge) feedback loop, a model that has become the most influential reference control model for autonomic and self-adaptive systems.

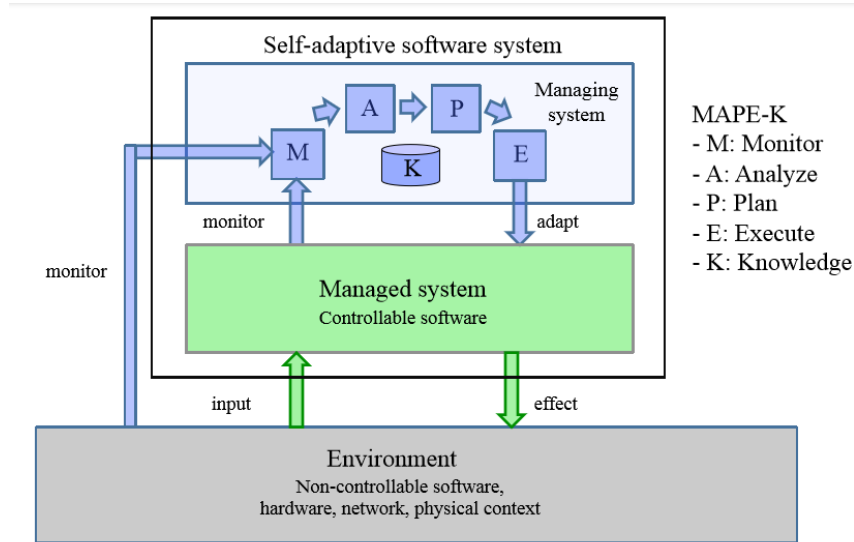


Figure 2.9: Full view of a Managed system controlled by the MAPE-K and interactive with the environment

The manager or controller is composed of two manageability interfaces, the sensor and the effector, and the monitor-analyze-plan-execute (MAPE-K) engine consisting of a monitor, an analyzer, a planner, and an executor which share a common knowledge base.

- The **Monitor** senses the managed process and its context, filters the accumulated sensor data, and stores relevant events in the knowledge base for future reference.
- The **Analyzer** compares event data against patterns in the knowledge base to diagnose symptoms and stores the symptoms for future reference in the knowledge base.
- The **Planner** interprets the symptoms and devises a plan to execute the change in the managed process through its effectors. It structures and plans the actions needed to achieve goals and objectives to be achieved by the controlled system.
- The **Executor**, under the directives of the Planner, changes the state of the controlled system using the effectors, also called actuators in fields like

Robotics or Automation

- The **Knowledge Base** continuously gathers information from the managed element as well as from the current and past states. The knowledge can be collected from various knowledge sources and the information acquired will be used to decide the behavior of the system according to its control objective.

Chapter 3

State of the art methodologies to face uncertainty

In this chapter we will disclose the Neuro-Fuzzy technology and all its different implementations. Moreover, we'll expose the reasons which led us to see this technique as best-suited for our project.

3.1 Neuro-Fuzzy Systems

3.1.1 Brief introduction to Neuro-Fuzzy technology

Neuro-Fuzzy technology, as the name easily suggests, has been derived from the two most advanced recently developed technologies, Artificial Neural Network and Fuzzy Logic Systems. Both technologies are extremely good and are still used in different scenarios so what did cause the need of merging these two technologies? While neural networks are good at recognizing patterns, they are not good at explaining how they reach their decisions. On the other hand Fuzzy Logic systems, which can reason with imprecise information, are good at explaining their decisions but cannot automatically acquire the rules they use to make those decisions. It is clear that both these two technologies, despite being widely effective in a large variety of problems, have some lacks in specific application contexts. The reason behind the birth of this new technology was to incorporate advantages of both its two parent technologies while at the same time overcoming their disadvantages. To better understand what we're talking about let us briefly present Neural Networks, while Fuzzy Systems have already been presented in 2.3

3.1.2 Basics of Neural Networks

Artificial Neural Networks represent a computational model whose creation was thought to resemble the way in which the human brain processes information through its biologic network of neurons. It achieved many unimaginable breakthrough results in important fields such as Speech Recognition, Text Processing, Computer Vision, Image classification, etc. The basic computational unit of a neural network is of course the **neuron**, frequently referred to as **node**. Its schema is quite simple and can be better understood by first having a look at 3.1

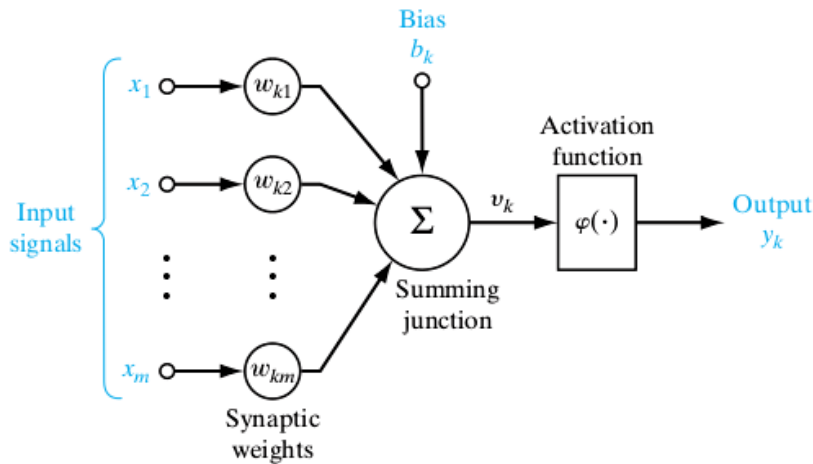


Figure 3.1: Detailed view of a neural network node.

The input signals of a node consist of output signals exiting from other nodes of the network or input signals coming from external sources. Each input has a weight associated that states its relative importance with respect to other inputs. Besides the weighted inputs an additional input is added before summing up all the input signals: the **bias**.

The bias is a trainable constant value which is critical for the success of the learning process. Once the input signals are summed up the neuron then applies a non-linear function called **activation function** which will decide the output value fired out from the node. The purpose of the activation function is to add non-linearity to the output of a neuron in order to learn the non-linear representations that are frequently present in our real world. Different activation function exists, the most common are:

- *Sigmoid* – the input having a real value becomes a value belonging to the interval $[0-1]$ through the function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3.1)$$

- *Hyperbolic tangent* – the real-valued input is squeezed in the interval $[-1,+1]$ through the function:

$$\tanh(x) = 2\sigma(2x) - 1 \quad (3.2)$$

- *Rectified Linear Unit* – the real-valued input loses its negatives values and starts from 0 through the function:

$$f(x) = \max(0, x) \quad (3.3)$$

An Artificial Neural Network is composed of many neurons and depending on how these neurons communicate among each other we have different kinds of neural networks. The current most used ones are: Feedforward Neural Network, Recurrent Neural Network, Convolutional Neural Network. The latter two types obtained great success respectively in speech recognition and text analysis, and computer vision. The first one is the first kind of neural network created and moreover the simplest type; it is often used as reference model when describing neural networks for the first time and we won't break this tradition.

3.1.3 Feedforward Neural Network

In this kind of neural network, whose name meaning will be explained soon, the multitude of nodes are grouped into **layers**. Nodes belonging to different layers communicate among each other through connections, also known as **edges**. The characteristic of this type of neural network, that is what it inherits its name from, is that information moves only in one direction, forward, starting from **input nodes**, moving through **hidden nodes** (if any), entering and exiting **output nodes**. Input nodes are nodes belonging to the homonym layer that do not perform any computation on the information which is passed as is to the next layers. Hidden nodes belong to the homonym layer and are so defined because they have no direct contact with the outside world. They perform computations and transfer information from input (or previous hidden layers) to the output layer (or next hidden layers). At the end of the network we find the output nodes, that are responsible for the computation and the transfer of information to the outside of the network.

3.1.4 Advantages and disadvantages of Neural Networks

After understanding both two founding technologies of Neuro-Fuzzy Theory we are ready to meticulously have a look at the advantages and disadvantages of the two respective technologies.

Advantages characterizing Neural Networks:

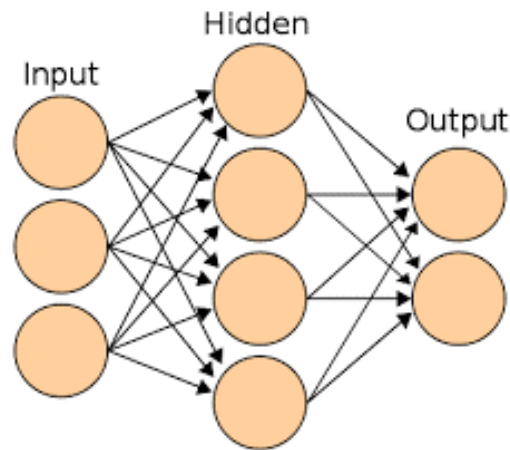


Figure 3.2: Three-layers Neural Network

- Adaptive learning: NNs learn how to perform in a problem basing on the training data or the initial knowledge (experience);
- Self-Organization: ANNs are free to create their representation and organization of the information received during learning process;
- Real Time Operation: ANNs can have multiple computations carried out in parallel, in fact hardware devices are designed and manufactured specifically to exploit this capability;
- Various types of Learning or Training Mechanism just require modification of synaptic weight link between Input neuron and Output neuron;
- Their ability to adapt to changes in surrounding (external and internal) by changing synaptic weight make them incredibly suitable for time variant applications;
- High Robustness provides high degree of fault tolerance;
- Provide output with high degree of confidence with accurate decision;
- NNs can be trained using several learning algorithms;

- A mathematical model of the problem is not needed and furthermore there's no need to provide any prior knowledge;
- High generalization capacity;

Disadvantages of Neural Networks:

- NNs can only be used with training data. This means that training data are mandatory for this kind of technology;
- The results coming out of the learning process are usually not human interpretable. This technology is a “Black Box” in which we have no knowledge of its internal workings;
- The neural network must learn everything from scratch and so the learning process can take very long and furthermore there's usually no guarantee of success;
- The training takes not only a long time but also a lot of data. The data required for training is limited to the specific problem and this implies that the same data can't be utilized on a wide scope;
- Don't deal with zero error and impression; deal only with minimization of error to converge system towards stability;
- For some problems it's hard to determine the number of neurons and number of layers;

3.1.5 Advantages and disadvantages of Fuzzy Logic

In this paragraph we simply report the advantages and disadvantages of Fuzzy Logic having already given an adequately good presentation of Fuzzy Logic and its application in self-adaptive systems in chapter 2.3 Let us start from its advantages:

- Easy to understand and implement;

3. *State of the art methodologies to face uncertainty*

- Provide more “user-friendly” and efficient performance. It’s easy to understand the results because of the natural rules representation;
- Deals with imprecision which make it capable to represent uncertainties of knowledge with linguistic variables;
- Describes Input and Output with the help of linguistic variables which make interpretation about the system and interaction with system easy;
- Robustness is high since it addresses the uncertainties, imprecision and disturbances;
- Easy extension of the base of knowledge through the addition of new rules. Rule Base or Fuzzy Set can be easily modified as per requirements;
- Non-linear functions can be easily modeled;
- No mathematical model needed;

Disadvantages of Fuzzy Logic:

- Not capable to learn;
- Hard to develop a model from a fuzzy system;
- Before being completely implemented the model requires fine tuning and simulation;
- It’s usually not easy to find the suitable membership values for fuzzy systems;
- Fuzzy Systems are more suited to solve problems in which knowledge about the solution is available in the form of linguistic IF-THEN rules;

- Not really capable of generalizing, or either, it only answers to what is written in its rule base;
- Incapable to generalize, or either, it only answers to what is written in its rule base;
- not robust in relation the topological changes of the system, such changes would demand alterations in the rule base;
- It's heavily dependent on a expert capable of determining the inference logical rules and building the Rule Base;
- A priori knowledge is essential;

3.1.6 Neuro-Fuzzy Logic in detail

Now that we've individually schematized the advantages and disadvantages of both the founding theories of Neuro-Fuzzy Logic it is time to take a closer look to this technology. As we have already stated, the concept idea of combining fuzzy systems and neural networks is to design an architecture that uses a fuzzy system to represent knowledge in a human interpretable manner and the learning ability of a neural network to optimize its parameters. The two major drawbacks of both the individual approaches, the black box behavior of neural networks, and the problems of finding suitable membership values for fuzzy systems, could thus be avoided by merging these two theories together. The result of the combination of the two technologies is an interpretable model with learning capacity and ability to use problem-specific prior knowledge. Fuzzy Logic and ANN seem to complement each other, as shown in figure, and therefore the Neuro-Fuzzy integrated approach is a combination of the pros of NN and Fuzzy Logic, enabling the creation of a new kind of intelligent decision-making systems. This incorporates the generic advantages of artificial neural networks like massive parallelism, robustness, and learning in data-rich environments into the system, and the advantages of fuzzy logic like the modeling of imprecise and qualitative knowledge and the concept of uncertainty. All these features explain why neuro-fuzzy methods are especially suited for applications where user interaction in model design or interpretation is desired. Here's a very useful table comparing the two technologies from different features:

| | Fuzzy Systems | Neural Networks |
|----------------------------|------------------|----------------------|
| <i>Knowledge Source</i> | Human Experts | Sample Sets |
| <i>Learning Mechanism</i> | Induction | Adjusting Weights |
| <i>Reasoning Mechanism</i> | Heuristic Search | Parallel Computation |
| <i>Learning Speed</i> | High | Low |
| <i>Reasoning Speed</i> | Low | High |
| <i>Fault tolerance</i> | Low | Very High |
| <i>Implementation</i> | Explicit | Implicit |
| <i>Flexibility</i> | Low | High |

Table 3.1: Comparison table of the characteristics of NNs and Fuzzy System

A neuro-fuzzy system is usually represented as a special three-layer feedforward neural network as is shown in 3.2, while several approaches also use five layers where the fuzzy sets are encoded in the neurons of second and fourth layer, as shown in 3.3.

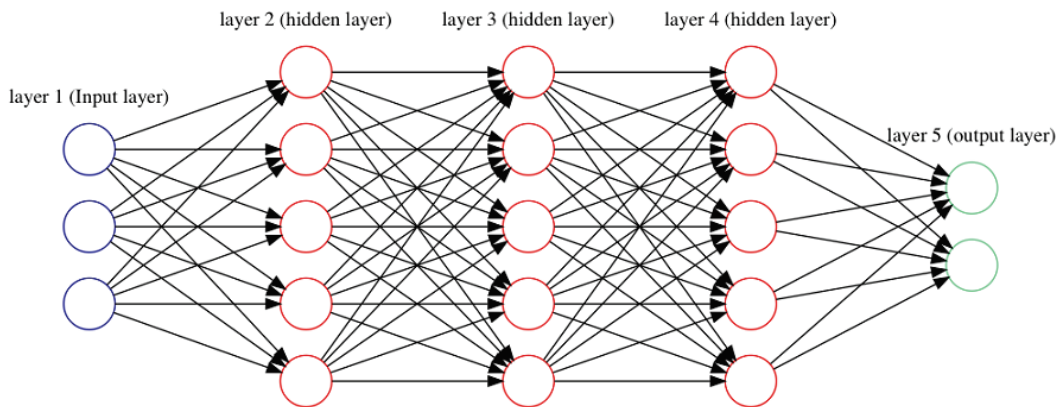


Figure 3.3: Five-layers Neural Network

Let's analyze all the layers of the conventional three-layers architecture:

- The first layer corresponds to the input variables.
- The second layer, or hidden layer, symbolized membership functions and fuzzy rules.
- The third layer represents the output variables.
- The fuzzy sets are converted as (fuzzy) connection weights.

3.2 Different types of Neuro-Fuzzy Systems

The two individual techniques, Fuzzy Logic and ANN, can be combined in a multitude of ways; the different combinations of these techniques can be divided, in accordance with [5], [24], [25], [26], in the following classes: *Cooperative Neuro-Fuzzy System*, *Concurrent Neuro-Fuzzy System*, *Hybrid Neuro-Fuzzy System*

Cooperative Neuro-Fuzzy System: In cooperative systems there is a pre-processing phase where the neural networks use its mechanisms of learning from training data to determine some sub-blocks of the fuzzy system. For instance, it could be determined the fuzzy sets and/or fuzzy rules, or there could be used clustering algorithms to determine the rules and fuzzy sets position. After the fuzzy sub-blocks are calculated the ANN is taken away, executing only the fuzzy system. Because the ANN is only used at the beginning to determine the sub-blocks of the fuzzy system the results are not completely interpretable, and this surely is a disadvantage for a technology whose one of the most important pro of the merging of ANN and Fuzzy Logic was exactly the human interpretability of the results.

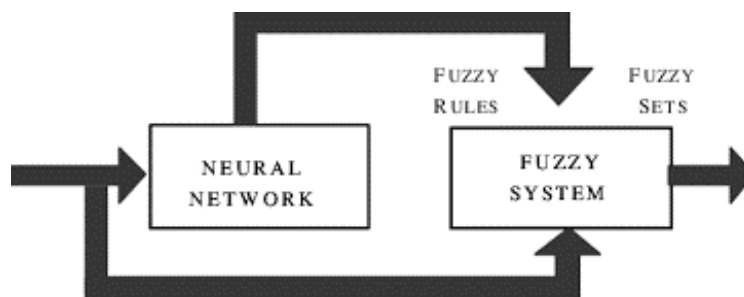


Figure 3.4: Cooperative Neuro-Fuzzy System

Concurrent Neuro-Fuzzy System: In concurrent systems the neural network and the fuzzy system work continuously together, and this is why they're not considered a neuro-fuzzy system in a strict sense. The input enters the fuzzy system, they're pre-processed in Fuzzy Logic and at the end the ANN processes the outputs of the system, figure 23, or the reverse way, figure 24. Again, also for this case the results are not completely interpretable.

Hybrid Neuro-Fuzzy System: In this kind of architecture a Neural Network is used to learn the parameters of the fuzzy system (parameters of the fuzzy sets, fuzzy rules and weights of the rules) in an iterative way. This kind of architecture is the most commonly used and most researchers use the neuro-fuzzy term to refer to hybrid neuro-fuzzy systems.

According to Nauck [24] definition: "A hybrid neuro-fuzzy system is a fuzzy system that uses a learning algorithm based on gradients or inspired by the neural

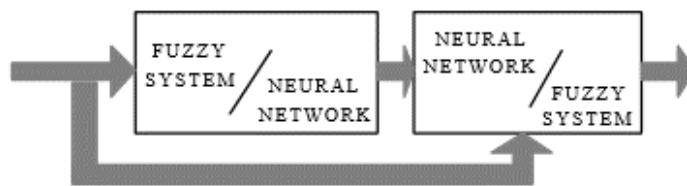


Figure 3.5: Generic Concurrent Neuro-Fuzzy System

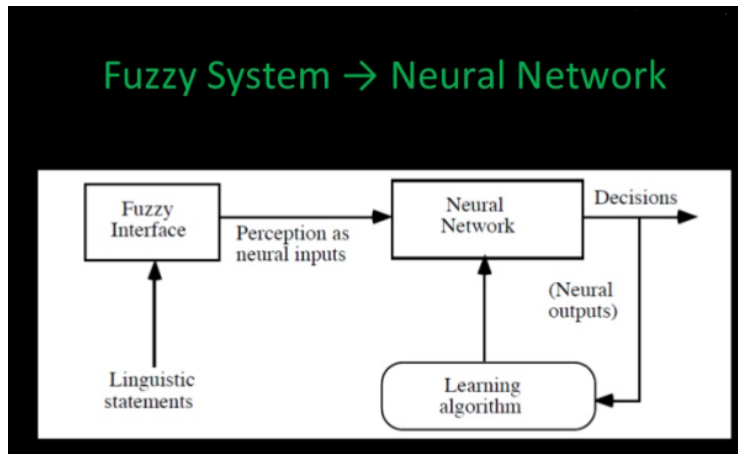


Figure 3.6: Concurrent Neuro-Fuzzy System with inputs pre-processed by the Fuzzy System

networks theory (heuristic learning strategies) to determine its parameters (fuzzy sets and fuzzy rules) through the patterns processing (input and output)". The advantage of such hybrid NFS is their architecture since both fuzzy system and neural network do not have to communicate with each other anymore. They are one fully fused entity which can both learn offline or online. The major difference with respect to the previous two neuro-fuzzy architectures is the complete interpretability of the results. This surely motivated us to choose it for our research purposes. There are several ways to use this kind of system in different problems according to the type of inputs and outputs chosen, as figure shows, moreover there are several different ways to develop hybrid neuro-fuzzy systems and so each researcher has defined its own particular architectural models.

The models reported in the table are similar in its essence, but they still present basic differences. Among the implementations of the several hybrid neuro-fuzzy architectures we can find:

- **Fuzzy Adaptive Learning Control Network (FALCON)** [27, C.T. Lin, C.S. Lee];

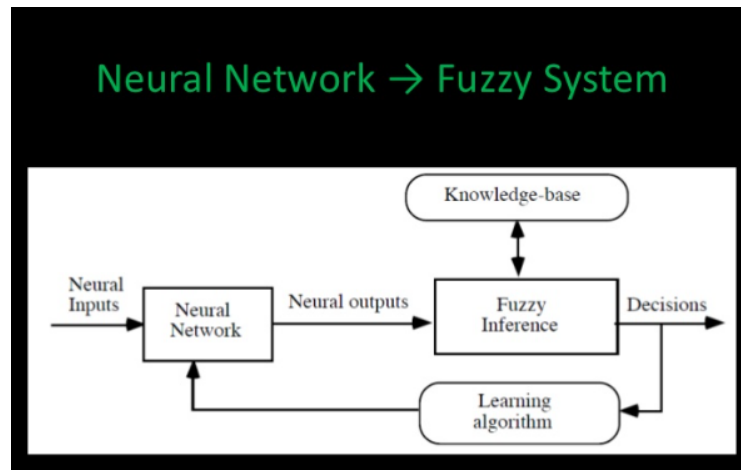


Figure 3.7: Concurrent Neuro-Fuzzy System with inputs pre-processed by the ANN

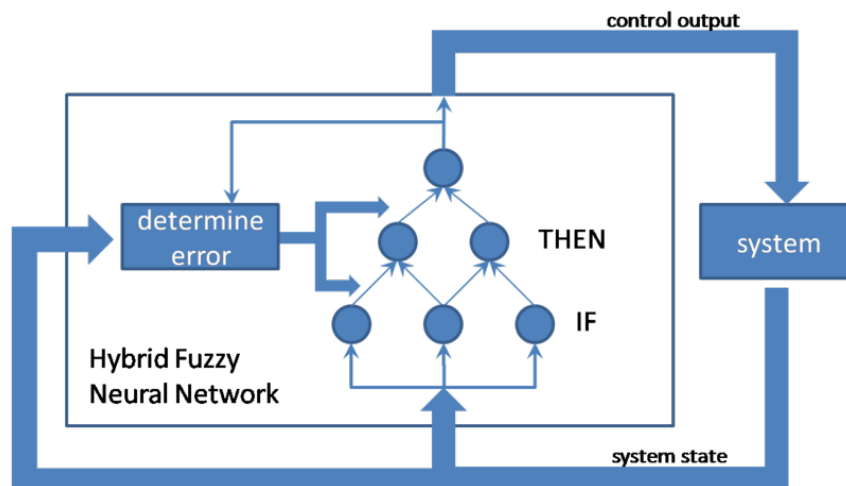


Figure 3.8: Hybrid Neuro-Fuzzy System

| Type | Weights | Inputs | Outputs | Applications |
|--------|---------|--------|---------|----------------|
| Type 0 | Crisp | Crisp | Crisp | N/A |
| Type 1 | Crisp | Fuzzy | Crisp | Classification |
| Type 2 | Crisp | Fuzzy | Fuzzy | Fuzzy IF-THEN |
| Type 3 | Fuzzy | Fuzzy | Fuzzy | Fuzzy IF-THEN |
| Type 4 | Fuzzy | Crisp | Fuzzy | Fuzzy IF-THEN |
| Type 5 | Crisp | Crisp | Fuzzy | Unrealistic |
| Type 6 | Fuzzy | Crisp | Crisp | Unrealistic |
| Type 7 | Fuzzy | Fuzzy | Crisp | Unrealistic |

Table 3.2: Table depicting the possible kind of weights, inputs, outputs of Neuro-Fuzzy Systems and their application scenarios.

- **Adaptive Network based Fuzzy Inference System (ANFIS)**, [?, R.R. Jang];
- **Generalized Approximate Reasoning based Intelligence Control (GARIC)**, [28, H. Berenji];
- **Neuronal Fuzzy Controller (NEFCON)**, [29, Nauck and Kruse];
- **Fuzzy Inference and Neural Network in Fuzzy Inference Software (FINEST)**, [30, Tano, Oyama and Arnould];
- **Fuzzy Net (FUN)**; [31, Sulzberger]
- **Self Constructing Neural Fuzzy Inference Network (SONFIN)**, [32];
- **Fuzzy Neural Network (NFN)**, [33];
- **Dynamic/Evolving Fuzzy Neural Network (EFuNN and dmEFuNN)** [34];

We'll stick on describing the five most popular architectures, which were also our choice pool, following the guidelines of the already very detailed description contained in [5].

3.2.1 **FALCON: Fuzzy Adaptive Learning Control Network**

FALCON [27] has a five-layered architecture, as shown in 3.9 . The first layer represents the input layer while the first hidden layer is responsible for the mapping of the input variables relatively to each membership function. The Second hidden layer defines the preconditions of the rule followed by rule consequents in the third hidden layer. There are two linguistic nodes for each output variable; one is for training data (desired output) and the other is for the actual output of FALCON. FALCON uses a hybrid-learning algorithm comprising of unsupervised learning to locate initial membership functions/rule base and a gradient descent learning to optimally adjust the parameters of the MF to produce the desired outputs.

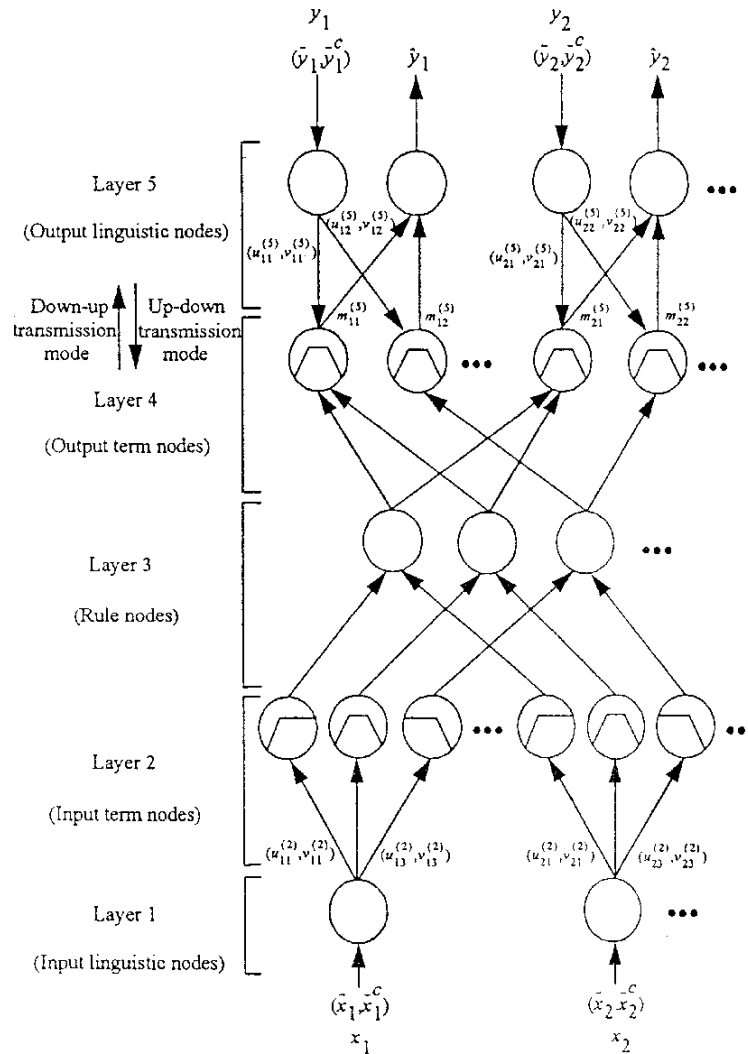


Figure 3.9: FALCON Architecture Layers

3.2.2 GARIC: The Generalized Approximate Reasoning based Intelligence Control

GARIC [27] implements a neuro-fuzzy controller by using two neural network modules, the ASN (Action Selection Network) and the AEN (Action State Evaluation Network). The AEN is an adaptive evaluator of the actions of the ASN. The ASN instead, is a feedforward network with five layers. 3.10 illustrates the structure of GARIC – ASN. A peculiarity of this architecture is represented by the connections between layers which are not weighted. The first hidden layer stores the linguistic values of all the input variables. Each input unit is only connected to the units of the first hidden layer, which represent its associated linguistic values. The second hidden layer represents the fuzzy rules nodes, which determine the degree of fulfillment of a rule using a softmin operation. The third hidden layer represents the linguistic values

of the control output variable. Conclusions of the rule are computed depending on the strength of the rule antecedents computed by the rule node layer. GARIC makes use of local mean-of-maximum method for computing the rule outputs. This method needs a crisp output value from each rule. Therefore the conclusions must be defuzzified before they are accumulated to the final output value of the controller. GARIC uses a mixture of gradient descent and reinforcement learning to fine-tune the node parameters.

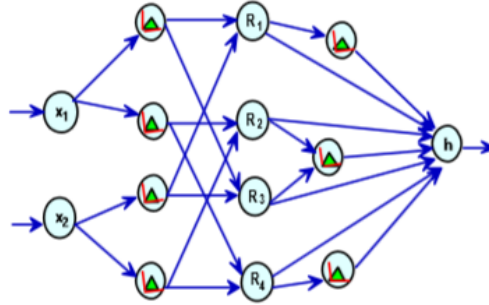


Figure 3.10: GARIC Architecture Layers

3.2.3 NEFCON : The Neuronal Fuzzy Controller

The Neuronal Fuzzy Controller NEFCON [29] was originally drawn to implement a Mamdani type inference fuzzy system as illustrated in figure ???. The connections between nodes are weighted with fuzzy sets and rules using the same antecedents (called shared weights), which are represented by the drawn ellipses. This is done to assure the integrity of the base of rules. The input units assume the function of fuzzyfication interface while the logical interface is represented by the propagation function and the output unit is responsible for the defuzzyfication interface. The process of learning in this architecture is based on a mixture of reinforcement learning and backpropagation algorithm. NEFCON architecture can be used to learn the rule base from the beginning, if there is no à priori knowledge of the system, or to optimise an initial manually defined rule base. NEFCON has two variants NEFPROX (for function approximation) and NEFCLASS (for classification tasks) [31]

3.2.4 EFuNN/dmEFuNN Architecture

In Evolving Neural Fuzzy Network EFuNN [34] all nodes are created during the learning phase. The first layer passes data to the second layer which is responsible of calculating the degrees of compatibility in relation to the predefined membership functions. The third layer contains fuzzy rule nodes representing prototypes of input-

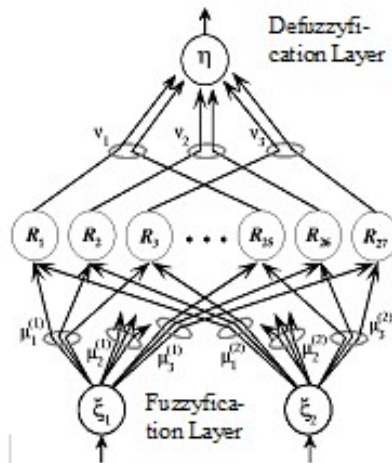


Figure 3.11: Nefcon Architecture Layers, from [5]

output data as an association of hyper-spheres from the fuzzy input and fuzzy output spaces. Each rule node is defined by two vectors of connection weights, which are adjusted through a hybrid learning technique. The fourth layer calculates the degree to which output membership functions are matched to the input data and the fifth layer carries out the defuzzification and calculates the numerical value for the output variable. Dynamic Evolving Neural Fuzzy Network (dmEFuNN) [34] is a modified version of the EFuNN with the idea that not only the winning rule node's activation is propagated but that a group of rule nodes that is dynamically selected for every new input vector and their activation values are used to calculate the dynamical parameters of the output function. While EFuNN implements Mamdani type fuzzy rules, dmEFuNN implements Takagi-Sugeno fuzzy rules.

3.2.5 ANFIS: The Adaptive Network based Fuzzy Inference System

ANFIS [?] implements a Takagi-Sugeno fuzzy inference system and it has five layers as shown in figure 31. The first hidden layer is responsible for the fuzzification of the input variables and the operator T-norm is applied in the second hidden layer to calculate the antecedents of the rules. The third hidden layer normalizes the rules strengths followed by the fourth hidden layer where the consequents of the rules are determined. The output layer calculates the global output as the summation of all the signals that arrive to this layer. ANFIS uses backpropagation learning to determine the input membership functions parameters and the least mean square method to determine the consequents parameters. Each step of the iterative learning

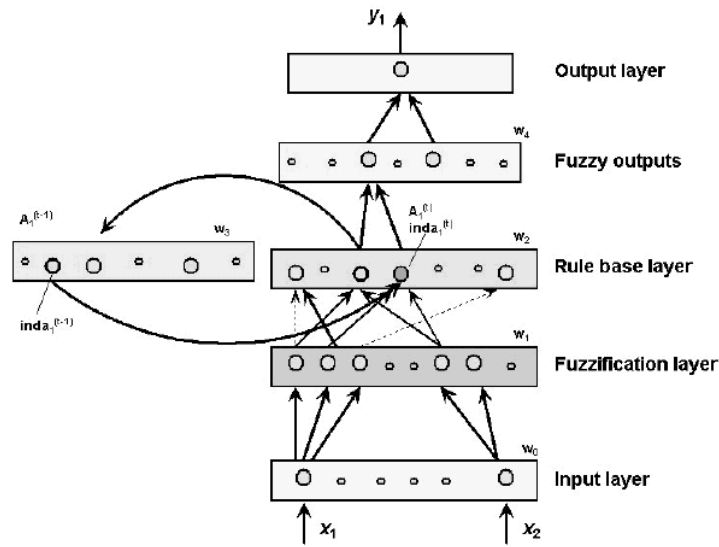


Figure 3.12: EFuNN Architecture Layers, from [5]

algorithm has two parts. In the first part, the input patterns are propagated and the parameters of the consequents are calculated using the iterative minimum squared method algorithm, while the parameters of the premises are considered fixed. In the second part, the input patterns are propagated again and at each iteration, the learning algorithm backpropagation is used to modify the parameters of the premises, while the consequents remain fixed. This procedure is then iterated.

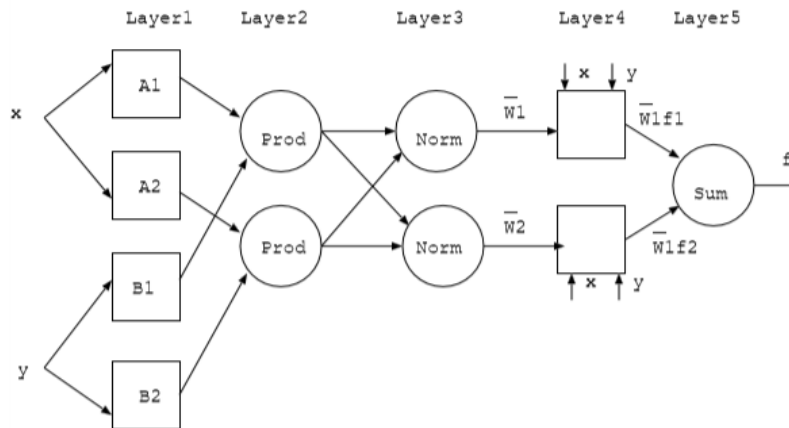


Figure 3.13: ANFIS Architecture Layers, from

ANFIS seems to represent the best-suited architecture for our study, providing, in addition to the already discussed advantages of the Neuro Fuzzy Systems, the following ones:

- *Smoothness*, due to the Fuzzy Logic interpolation.

- *Adaptability*, due to the NN backpropagation algorithm.

3.3 ANFIS seen in detail

The ANFIS architecture is an adaptive network which uses a supervised learning on a learning algorithm. Before going deep in details of the distinctive ANFIS learning process we briefly have to explain the possible mathematical models upon which the Fuzzy logic is based. These mathematical models are called Fuzzy Models.

3.3.1 Fuzzy Models

A mathematical model which in some way uses fuzzy sets is called a fuzzy model. In system identification, rule-based fuzzy models are usually applied. In these models, the relationships between variables are represented by means of if-then rules with imprecise (ambiguous) predicates, such as:

$$IF \textit{ heating is HIGH THEN temperature speed increase is FAST} \quad (3.4)$$

This rule defines in a rather qualitative way the relationship between the heating and the temperature in a room, for instance. To make such a model operational, the meaning of the terms ‘high’ and ‘fast’ must be defined more precisely. This is done by using fuzzy sets, i.e. sets where the membership is changing gradually rather than in an abrupt way. Fuzzy sets are defined through their membership functions which map the elements of the considered universe to the unit interval $[0,1]$. The extreme values 0 and 1 denote complete membership and non-membership, respectively, while a degree between 0 and 1 means partial membership in the fuzzy set. Depending on the structure of the if-then rules, two main types of fuzzy models can be distinguished: the Mamdani (or linguistic) model and the Takagi-Sugeno model.

3.3.2 Mamdani Model

The most commonly used fuzzy inference technique is the so-called Mamdani method [35] which was proposed, by Mamdani and Assilian, as the very first attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators. In Mamdani’s model the fuzzy implication is modeled by Mamdani’s minimum operator, the conjunction operator is min, the t-norm from compositional rule is min and for the rules aggregation the max operator is used. In this model, the antecedent (if-part of the

rule) and the consequent (then-part of the rule) are fuzzy propositions:

$$R_i : \text{if } x \text{ is } A_i \text{ then } y|; \text{ is } B_i, \quad i = 1, 2, \dots, K \quad (3.5)$$

Here A_i and B_i are the antecedent and consequent linguistic terms (such as ‘small’, ‘large’, etc.), represented by fuzzy sets, and K is the number of rules in the model. The linguistic fuzzy model is useful for representing qualitative knowledge, illustrated in the following example.

3.3.3 Takagi-Sugeno Model

The Mamdani model is typically used in knowledge-based (expert) systems while in data-driven identification, the model due to Takagi and Sugeno has become popular. ANFIS is one of the many systems adopting the Takagi-Sugeno model. In this model, the antecedent is defined in the same way as defined above for Mamdani, while the consequent is an affine linear function of the input variables:

$$R_i : \text{if } x \text{ is } A_i \text{ then } y_i = \mathbf{a}_i^T \mathbf{x} + b_i, \quad i = 1, 2, \dots, K \quad (3.6)$$

where \mathbf{a}_i is the consequent parameter vector and b_i is a scalar offset. This model combines a linguistic description with standard functional regression: the antecedents describe fuzzy regions in the input space in which the consequent functions are valid. The output y is computed by taking the weighted average of the individual rules’ contributions:

$$y = \frac{\sum_{i=1}^K (\beta_i(\mathbf{x}) y_i)}{\sum_{i=1}^K (\beta_i(\mathbf{x}))} = \frac{\sum_{i=1}^K (\beta_i(\mathbf{x}) (\mathbf{a}_i^T \mathbf{x} + b_i))}{\sum_{i=1}^K (\beta_i(\mathbf{x}))} \quad (3.7)$$

where $\beta_i(x)$ is the degree of fulfillment of the i – *th* rule. For the previous rule $\beta_i(x) = \mu_{A_i}(x)$, but it can also be a more complicated expression. The antecedent fuzzy sets are usually defined to describe distinct, partly overlapping regions in the input space. The parameters \mathbf{a}_i are then (approximate) local linear models of the considered nonlinear system. The TS model can thus be regarded as a smooth piecewise linear approximation of a nonlinear function or a parameter-scheduling model. Note that the antecedent and consequent variables may be different.

3.3.4 ANFIS Layers

Layer 1 : Considering $O_{1,i}$ as the output of the i – th node of layer 1 , we have that each node i in layer 1 is an adaptive node with function :

$$\begin{aligned} O_{1,i} &= \mu_{A_i}(x) \text{ for } i = 1, 2 \\ &\text{or} \\ O_{1,i} &= \mu_{B_{i-2}}(x) \text{ for } i = 3, 4 \end{aligned} \quad (3.8)$$

X (or y) is the input of node i and A_i (or B_{i-2}) is a linguistic label associated with this node. Consequently $O_{1,i}$ is the membership function grade of a fuzzy set (in this case A_1, A_2, B_1, B_2) and μ_A, μ_B are the membership functions.

$$\mu_A = \frac{1}{1 + \frac{x - c_i^{2b_i}}{a_i}} \quad (3.9)$$

where a_i, b_i, c_i is the parameter set and they're referred to as premise parameters.

Layer 2 : Every node in this layer is a fixed node labeled as Prod. The output of this type of node is the product of all the incoming signals coming from layer 1:

$${}_{2,i} = w_i = \mu_{A_i}(x) * \mu_{B_i}(y), \quad i = 1, 2 \quad (3.10)$$

Each node represents the “fire strength” of the rule and any other T-norm operator performing the AND operation can be used.

Layer 3 : Every node in this layer is a fixed node labeled as Norm. The i -th node computes the ratio of the i – th rule's firing strength over the sum of all rule's firing strengths.

$${}_{3,i} = \bar{w}_i = \frac{w_i}{w_1 + w_2}, \quad i = 1, 2 \quad (3.11)$$

Outputs are called normalized firing strengths.

Layer 4 : Every node in this layer is an adaptive node with node function :

$${}_{4,i} = \bar{w}_i f_i = \bar{w}_i (p_x + q_i y + r_i) \quad (3.12)$$

\bar{w}_i is the normalized firing strength coming from layer 3, while p_i, q_i, r_i is the parameter set for node i and they're referred to as consequent parameters.

Layer 5 : The unique node present in this layer is a fixed node labeled as sum, which computes the overall output as the summation of the incoming signals:

$$\text{Overalloutput} = {}_{5,i} = \sum_i (\bar{w}_i f_i) = \frac{\sum_i w_i f_i}{\sum_i w_i} \quad (3.13)$$

3.3.5 Hybrid Learning algorithm

In the ANFIS architecture the first and the fourth layers contain the parameters that can be modified over time (adaptive nodes). To update both the premise and the consequent parameters we need a learning method which is able to train both these parameters and to adapt to its environment. The algorithm used is the one proposed by Jang in [36]. This algorithm was introduced because the backpropagation algorithm commonly used in ANN was found problematic especially with a slow convergence rate and tend to be trapped in local minima. The hybrid learning algorithm is divided in two parts, the forward path and the backward path. During the forward path the parameters of the premises in the first layer are put in a steady state, then a Recursive Least Square Estimator (RLSE) method is applied to repair the consequent parameters in the fourth layer. Due to the consequent parameters being linear the RLSE method can be applied to accelerate the convergence rate in the whole learning process. Next after the consequent parameters are obtained, the input data is passed back to the adaptive network input and then the output generated will be compared with the actual output. While the backward path is run, the consequent parameters are now put in a steady state. Now the error occurred during the comparison between the output generated and the actual output is propagated back to the first layer, at the same time the parameter premises in the first layer are updated using learning methods of gradient descent or backpropagation. By combining RSLE and gradient descent methods, the hybrid learning ensures that the convergence rate is faster by reducing the dimensional search space in the original method of backpropagation. Further explanations of this kind of learning are present in [37] We conclude this paragraph schematizing the learning process in the next table:

| | <i>Forward Pass</i> | Backward Pass |
|------------------------------|------------------------|----------------------|
| Premise Parameters | Fixed | Gradient Descent |
| Consequent Parameters | Least-square estimator | Fixed |
| Signals | Node outputs | Error signals |

Table 3.3: Table depicting the possible kind of weights, inputs, outputs of Neuro-Fuzzy Systems and their application scenarios.

Chapter 4

Proposal

As previously exposed in chapter 3 we are going to face and handle uncertainty which intrinsically resides in self-adaptive systems. The intrinsic nature of Fuzzy Logic, combined with the learning capability of Neural Networks, makes it the best approach to study uncertainty.

4.1 Our application scenario

Like we declared since the earliest phases of this work we are going to use Neuro-Fuzzy systems to try to remove as much as possible the uncertainty in the decision-making process of a system. Let's see in detail how we decided to implement the Neuro-Fuzzy technology.

The first choice to made was whether to adopt an online learning method or a batch learning method.

The majority of Machine Learning algorithms implements batch learning techniques which perform learning over the entire training dataset at once. This implies that what is used to compute the performance of the loss function for the current evaluated model is the whole training set.

In online learning, in opposition with batch learning, data becomes available in a sequential order and is used to update the best predictor (model) for future data at each timestep. This kind of learning is commonly used when it is computationally infeasible to train over the entire dataset, or when it's necessary for the algorithm to adapt to new patterns found in the data, or also when the data is a function of time.

Our core idea is to develop a tool that can be easily applied to heterogenous datasets and environments. This is in perfect agreement with the concept of having an algorithm capable of adapting new patterns in data therefore leading our choice to an online learning method.

Exploiting the software engineering key concept of reusability, we'll use a MATLAB tool implementing the ANFIS architecture. In the next paragraph we dive deep in the details of the ANFIS implementation.

4.2 Characteristic of a fuzzy system

Before we can actually train a real fuzzy-logic based system we have to meticulously choose how the system will parametrize input and output membership functions. The choice is among three different clustering types:

- **Grid Partition**, generates input membership functions by uniformly partitioning the input variable ranges, and create a single-output Sugeno fuzzy system. The fuzzy rule base contains one rule for each input membership function combination.
- **Subtractive Clustering**, generates a Sugeno fuzzy system using membership functions and rules derived from data clusters found using subtractive clustering of input and output data.
- **FCM Clustering** (Fuzzy C-Means Clustering), generates a fuzzy system using membership function and rules derived from data clusters found using FCM clustering of input and output data.

Each of these clustering techniques has several technique-specific input and output options whose values are chosen by the expert user, therefore the choice between these three options will deeply affect the final results of the study. FCM Clustering and Subtractive Clustering have several more options than Grid Partitioning which has only three, and most of the former have a value range that goes from 0 to 1 leading to an almost infinite possible number of different values combinations. The choice of implementing a Grid Partitioning system seems understandable and will result even more comprehensible when our specific implementation of the algorithm will be shown and explained.

Grid Partitioning Options:

- **Number of input membership functions per input variable**, which allows the choice of the number of input membership functions for each input

variable (must be greater than 1).

- **Input Membership Function Type**, through which you can define a different membership function type for each input. The most common types which have also been considered by us are the generalized bell-shaped, the gaussian, the triangular, the trapezoidal and the sigmoidal.
- **Output Membership Function Type**, specifies the output membership function type for a single-output Sugeno system. Can be linear, which means that the output of each rule is a linear function of the input variables scaled by the antecedent result value, or constant, which means that the output of each rule is a constant scaled by the antecedent result value.

These were the parameters which are common to every kind of problem faced with this kind of technology. To explain the remaining ones we have to introduce the specific application of our Neuro-Fuzzy system, i.e. the kind of problem it will have to face. The system will be implemented to face the uncertainty present in the time-series prediction problem and consequently, considering the choice of an online learning method, will have the following additional parameters:

- *Number of past time instants*, which is deciding how many past values of the time series $y(t)$ should we use as input variables for the next timestep prediction.
- *Delay of the prediction*, which means how many steps ahead our system should predict a value. To make it more clear, let us suppose to have in a given timestep t , the value of the number of active superpeers $y(t)$. Using $y(t)$ as input variable we can decide a real-valued number d which indicates at which timestep we want the predicted output value, giving us $y(t + d)$. As one may easily guess, the farther we want to predict the more our prediction will be imprecise.
- *Choice of past time instants*, which means choosing which are the past instants of the function values we want to use to predict the next timestep. To clarify the concept by supposing to have a number of past time instants of two (which means that we'll use two past values of $y(t)$ as input variables for the systems),

we should choose whether the input values $y(t-5)y(t-2)$ or $y(t-2)y(t-1)$ are better for predicting the output value $y(t)$.

Before we introduce the innovative approach considered for the settings of the parameters values we can help the reader to better depict the problem by introducing him the dataset we used for the analyses.

4.3 Description of the Dataset

During this work we took as reference the Skype¹ network self-adaptive system, an architecture composed of many nodes, called superpeers, whose status can simply be active or inactive. The uncertainty here is represented by the unpredictable and continuous changing number of superpeers active at a given moment, consequently our problem will consist in predicting how this number will evolve in the future, in order to reduce the uncertainty of the availability of the Skype network. The dataset chosen for our study is the Skype superpeers one [7] This dataset contains a set of 4000 nodes participating in the Skype superpeer network to which were sent an application-level ping every 30 minutes for one month; a node has been considered up at a given time if the most recent ping succeeded.

The dataset is saved in a .evt file format (Event Trace) which is a simple chronological list with one event per line. Each line has the format $\langle time \rangle \langle node_name \rangle \langle event \rangle$:

- $\langle time \rangle$ is a number indicating the time in which the specific provides has become available or unavailable.
- $\langle node_name \rangle$ is an alphanumeric string representing the identifier of the provider.
- $\langle event \rangle$ is a flag which specifies the action performed by the provider, Up or Down.

The time unit measure is seconds.

Here's an example of the dataset with 7 different providers and their respective actions.

¹Skype is a proprietary freeware for instantaneous messaging and Voice over IP. It merges the usual characteristics of chatting, file transfer, etc., to a phone call system based on a Peer-to-Peer network.

```

0.000000 a72cc49accf5dfb80f89e0937c2a6c19 up
0.000000 2253fd6de1377412ce0e507c8e812b0b up
0.000000 b57b9ffc75f00a55c5750c51f40b56bd up
0.000000 ddc6fdb04549b110508bec392dbdce37 up
1814.000000 db2fdbcf129030b6d1e0ef03c829b7c4 down
1814.000000 bdfc82453cca72c01bfd3af5f47a262c up
1814.000000 ce134d6f221670351f9e1e11ba30ee46 up

```

Figure 4.1: Some rows of the dataset

We've a total number of 58435 samples, ranging from second '0' to '2478200.00' as depicted in the curve in figure 4.2.

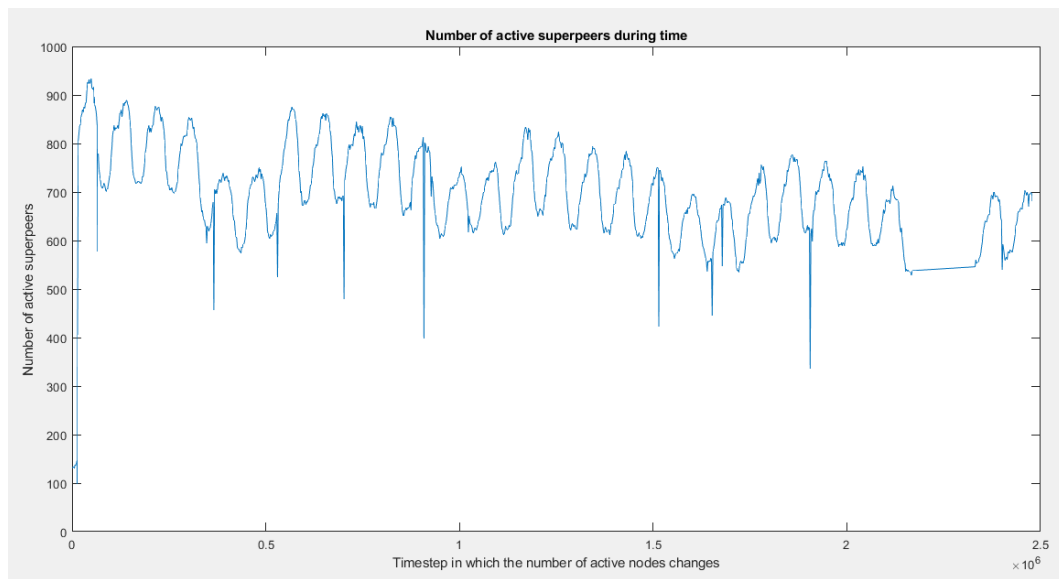


Figure 4.2: The number of active superpeers as function of time (in seconds)

However for our purpose we're only interested in the total number of active superpeers at each timestep, not worrying about which specific node was on and which was off. After performing some data transformation we structure the records as the example table below:

| Timestamp | Number of active superpeers |
|-----------|-----------------------------|
| 0 | 133 |
| 1814 | 132 |
| 3629 | 133 |
| 5444 | 131 |
| 7259 | 137 |

Figure 4.3: Some rows of the dataset after data transformation

The dataset now contains 1264 rows implying that many superpeers become

available/unavailable at the same time. This can easily be explained with the fact that Microsoft tries to adjust the number of providers based on the users' demand curve, including the day-night alternation. Then we delete the timestamp so that we can now replot our graph considering an increase of one timestep unit each time the number of active superpeers changes.

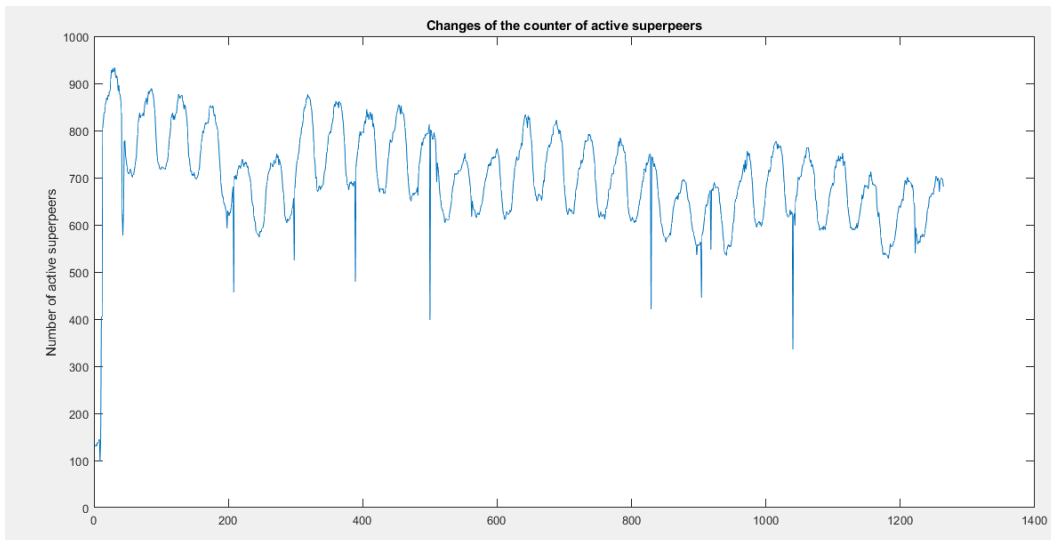


Figure 4.4: The number of active superpeers as function of timesteps

At this point everyone can see a transient phase at the beginning of the curve which rapidly grows from 133 to around 900. Our interest is focused on the asymptotic behavior of the curve so in order to have a significative result we'll cut the initial transient off the analyses. After explaining the dataset we'll use for the analysis of the performance of our system we can finally move to expose our taken approach with parameters.

4.4 Our approach with parameters

After choosing Grid Partition as clustering type we're left with the choice of the parameters. Let us start with the ones which are easier to set:

- Choice of past time instants
- Delay of the prediction
- Output membership function type

We can discuss their values by starting from the first one reported in the list above: choice of past time instants.

Usually this parameter is finely tuned based on the specific behavior of the time-series, i.e. if we know to have a function which seems to have a period of 5 time measures it is suggested to use the value of $y(t - 5)$ to predict the next timestep value $y(t)$. However we don't have any warranties on the possible periodic behavior during time of the number of Skype Superpeers, therefore it is reasonable to decide using the most recent data available, $y(t - 1), y(t - 2), \dots$.

Moving to the second parameter value to set, delay, we have to remind that we removed the precise time in seconds during which the number of active peers changes. This means that we're focused on the very next value of the number of the active peers and not on the farthest future ones, therefore leading to a choice of $d = 1$.

At last for the output membership function type we leave the default value setting of '*linear*' as it is widely reported to be the most effective. Now we're left with finding the best values for the following parameters:

- number of input membership functions
- input membership functions type
- number of input variables (past values of $y(t)$ to use)

Differently from the previous ones, for these parameters the literature reports no exact method to choose their best value. They are a classic example of parameters which have to be tuned with the trial and error method, moreover quick experiments showed that there's no best value for each of them taken individually. They're correlated between each other therefore we can't anymore find three separate good values for each of the parameters but we'll now look for a combination of these three values which gives a good result.

Consequently the problem is now divided in two parts:

- How do we evaluate a combination as a good or bad one?
- How do we find this combination?

This is where our approach differentiates from the others since we try to implement a systematic method, compared to the use of trial and error performed by expert users, to get the correct values for the parameters. For the first part of the problem we decided to create a tool which meticulously tries, one by one, all the different combinations of values for the three parameters. Now we're left with how to practically evaluate a combination of values, which means that we need a way to understand if a combination is better than another.

But how is it possible to know what's the better combination of parameters values to predict an output which we are not able to know yet?

In short terms it's not possible, but if we don't know both the future (the output) and the present (the combination of values) all we have left is of course the past. Then the concept idea becomes to predict the next timestep output $y(t + 1)$ with what resulted as the best parameters values combination for the output of the very previous timestep $y(t)$. Now let us put ourselves at the start of timestep t , which means that we know the real output values of timestep $t - 1$ and $t - 2$, then we can classify a combination as a good or bad one by calculating the root squared error²:

$$Error = \sqrt{(y_p(t - 1) - y_o(t - 1))^2}; \quad (4.1)$$

where $y_p(t - 1)$ represents the number of active superpeers predicted by the system with what resulted as the best combination at timestep $t - 2$, while $y_o(t - 1)$ is the number of active superpeers calculated with the combination of parameter values whose given predicted output was the closest to the actual real output $y(t - 1)$. The focused reader might have noticed something: there's no best combination of values for the previous output at the first timestep. To overcome this we set an initializing phase of 2 timesteps in which we let the system learn the real output while also making it do the prediction. At the beginning of the third timestep we pick the combination of values which gave the best predicted output for the second timestep; then we let the execution proceed normally as explained before.

The tool we implemented creates a fuzzy system for each of the all possible combinations of the three parameters values; each of these combinations represents a different Fuzzy System which will be trained with the Neural Network. However not all possible combinations of parameters values are meaningful or acceptable for different causes. The two parameters causing some limitations are the *number of input membership functions per input variable* and the *number of input variables*. The product of these two gives the system total number of input membership func-

²The reported formula of the Root Squared Error doesn't have arithmetic operations between different terms under the squared parenthesis, so in this case corresponds to: $|y_p(t - 1) - y_o(t - 1)|$

tions with the latter one having the most restricted range of values. In addition, the number of input variables highly increases the number of ANFIS parameters that will be tuned by the Neural Network, causing the entire system to overfit. To give an idea, with 7 input variables and 2 membership functions per each input variable we have an overall of number of 1080 tunable parameters, 1024 being linear and 56 nonlinear, and 128 fuzzy rules.

This shows that we cannot increase the number of input variables more than 7 and that even for smaller values this limits the number of membership functions per input variable. A special case exists when the number of input variables is equal to 1 as increasing the number of membership functions per input variable more than 14 brings no noticeable difference in the outcome.

Now after showing in words the special cases of the two parameters we present a table resuming all the different Fuzzy Systems formed by the combination of the three “free” parameters:

| <i>Number of Input Variables</i> | <i>Number of Membership Functions per Input Variable</i> | <i>Membership Function Type</i> |
|----------------------------------|--|-------------------------------------|
| 1 | 2,3,...,14 | Trapmf,trimf,psigmf,gbellmf,gaussmf |
| 2 | 2,3,...,14 | Trapmf,trimf,psigmf,gbellmf,gaussmf |
| 3 | 2,3,...,6 | Trapmf,trimf,psigmf,gbellmf,gaussmf |
| 4 | 2,3 | Trapmf,trimf,psigmf,gbellmf,gaussmf |
| 5 | 2,3 | Trapmf,trimf,psigmf,gbellmf,gaussmf |
| 6 | 2 | Trapmf,trimf,psigmf,gbellmf,gaussmf |
| 7 | 2 | Trapmf,trimf,psigmf,gbellmf,gaussmf |

Table 4.1: Table depicting the possible kind of weights, inputs, outputs of Neuro-Fuzzy Systems and their application scenarios.

Where trapmf, trimf, psigmf, gbellmf, gaussmf respectively stand for trapezoidal triangular, p-sigmoidal, gaussian bell and gaussian membership function. We also included the last case which is going to overfit, as written before, in order to analyze in the results if the tool is going to choose this fuzzy system over the others.

Considering one system for each possible combination reported in the table above, the tool is going to tune the parameters of 185 different Fuzzy Systems. This shows the massive time complexity required by the tool which has to find the best values for the tunable ANFIS parameters, for each of the 185 different Fuzzy Systems, for each row of the dataset.

Chapter 5

Practical implementation and results

5.1 Challenges of the implementation

The ANFIS tool, despite being very useful and complete, doesn't provide the possibility to perform an online learning, leaving the choice of using batch learning only; this represents a big problem for our application idea. Eventually we came with an idea that mimicked the behavior of a system performing online learning: by making data available in a sequential order and updating the best predictor at each timestep we performed a fully-fledged online training from the output point of view. To implement this, the tool, at each timestep, has to re-learn the entire dataset rows available until now instead of just adding to the learning set the new row with the next timestep, forgetting what has learned in the past and will again learn now. This implies that if the number of rows increases the time-consume of the tool raises by multiple factors. The time-consume problem doesn't concern the until now proposed approach but is unfortunately caused by the application of the above explained 'patch' to the lack of online learning of the MATLAB ANFIS tool. This results in the algorithm having very expensive time needs that we show now with a quantitative practical example (again we make it clear that it only depends on the specific MATLAB implementation and won't concern a proper implementation with a native online learning method). We applied the tool on 80 rows of the dataset, with a number of epochs set to 40, and it took several hours to complete on a common-performance laptop (in a range of 7-9 hours); at this point it seems hard even to think to perform the training on the entire 1264 rows dataset. Differently from what happens when using Neural Networks taken individually, reducing the number of epochs doesn't seem to greatly affect the results as the system shows huge dependency from the Fuzzy System initial parameters. Unfortunately

reducing the number of training epochs to 10 or 20 doesn't drastically reduce the time duration of the execution which is caused by the high number of different Fuzzy Systems that have to be trained and mainly by the implementation we used to overcome the limitation of the MATLAB ANFIS tool. Despite these limitations we still wanted to apply our tool to all the data we currently own in the 1264 rows of the dataset. We therefore decided to divide the learning phase in different sessions, each with a part of the dataset, each part being composed of 140 rows, with a number of epochs set to 10. In this way the analyses on the output are still meaningful without losing precious data.

5.2 Analyzing the results

We can now proceed to analyze and discuss the performance of the implemented tool.

To analyze the obtained results we report here the details of one of the many 140 rows datasets executions of the tool; picking a specific different execution, which means a different 140 rows block of the dataset, doesn't bring clearly seeable differences in the values with respect to the block we are going to analyze. We used different attributes to catch every aspect of the execution and we regrouped them using two different tables for better human readability.

The first table is structured in this way:

- **Timestep**, the timestep we're considering for the predicted and real output.
- **Number of input variables used**, number of input variables used by the best-valued combination of the three parameters for the prediction at the current timestep.
- **Number of Membership Functions per input variable**, number of Membership Functions per input variable used by the best-valued combination of the three parameters for the prediction at the current timestep.
- **Membership Function type**, Membership Function type used by the best-valued combination of the three parameters for the prediction at the current timestep.

- **Minimum error on previous timestep**, minimum absolute difference value between the output and the predicted outputs of the previous timestep of all the Fuzzy systems. This value is used to choose the best values combination to use for the prediction at the next timestep.
- **Predicted output**, the predicted outcome of the system chosen at the current timestep on the basis of the previous timestep performance.
- **Real Output**, the target outcome at the current timestep.
- **Error we would have had with the actual best system**, minimum absolute difference value between the real output and the predicted output, of the current timestep, of the system which will give us *minimum error on previous timestep* at the next timestep. This attribute is meant to show us what happened if we used the system we'll found out, at the next timestep, to be best for the current timestep. In short terms this attribute shows what we would have achieved with the real best combination of parameter values.
- **Current Error**, minimum absolute difference value between the output and the predicted output of the Fuzzy system chosen to perform the prediction.
- **Parameters values** characterizing the Fuzzy System, which are the *number of input variables used, number of membership functions per input variable, and Membership function type*.
- **Total error**, the sum of all the absolute difference values between the predicted and real output for that specific Fuzzy System. This represents the total error if we always used from the beginning this specific kind of Fuzzy System.
- **Number of times picked as best system** as best system, which is a numeric value indicating how many times the system with the specific characteristics reported in the first attribute has been chosen as the best predictor at some timestep.

To better compare the performance of our tool we used as frame of reference these attributes:

- **Total error of the “Dynamic System”**, which is the sum of all the errors resulting from using our tool which dynamically adopts a potentially different Fuzzy System to find the best predictor at each timestep.
- **Minimum recorded error**, the minimum among the total errors of each Fuzzy System.
- **Average total error**, the average of each Fuzzy System total error.
- **Average total error without ‘outliers’**, the average of each Fuzzy System total error after the removal of the so-called outliers. Outliers are highly greater or highly smaller values with respect to the majority of considered values. In this case we only had highly greater values than most of the values, which are the ones we removed for computing this attribute value.

Before showing the detailed results of the tables which are focused on a 140-rows block dataset we can observe the global performance the system with a graph showing two curves, the real output and the predicted one.

In the figure shown below we did not show the negative predicted values which come as result of a raw usage of the tool, that we can interpret as a ‘laboratory experiment’. This kind of results would be very easily to handle (e.g. by repeating the predictions if outcome values are negative) in a system which has to reach best possible outcome in its environment.

The predicted output seems to resemble well the shape of the predicted output curve; though it is noticeable that there are some outliers. We can detect these outliers using the powerful measure of the *Interquartile Range*; in this way we can build ‘a fence’ which will classify data outside it as *major outliers*. Our fence posts are calculated like this:

$$\begin{aligned} \text{Upper bound} &= Q3 + IQR * 3 = 1212.3 \\ \text{Lower bound} &= Q1 - IQR * 3 = 185.8 \end{aligned} \tag{5.1}$$

We replaced each value outside this range with the median value. Let’s finally see the new graph:

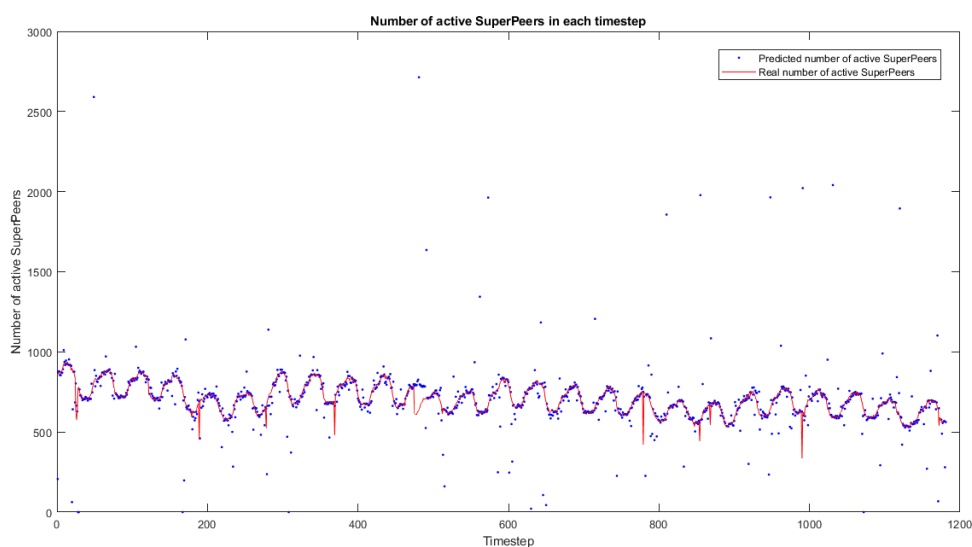


Figure 5.1: Real number of active Superpeers vs Predicted number of active Superpeers with outliers

From this graph we can see that the self-adaptive system overestimates the down spikes of the real number of active SuperPeers, which are due to the day-night alternation.

To avoid showing huge unreadable tables we report the first 30 rows of each table type to give an idea on how they are structured:

As the reader may notice we cut out the first two timestep of each 140-rows dataset as they are the two ‘pure learning’ rows. The error is quite unstable, going from being incredibly correct to a bit less correct, we suppose this is the consequence of our conjecture: we hypothesized that best combination of parameter values for the previous timestep prediction would still be good for the prediction at the current timestep. This of course isn’t necessarily true, moreover by making the system learn on numerous dataset rows while only focusing on the output prediction at the previous timestep it’s like we’re forcing the system to overfit on the result of a single timestep.

Other than the current error measure there’s another positive thing to notice: by summing all the 8th column values of the complete table, which represent the prediction error we could have had if we actually used the best possible parameters combination values (only known at the successive timestep), we have a total error of 73,2713. This is an astonishing result noticing that there are 140 timesteps considered, giving us an average error of 1,9107 per timestep. This statement indeed shows how importance has the prediction of the right parameters combination values.

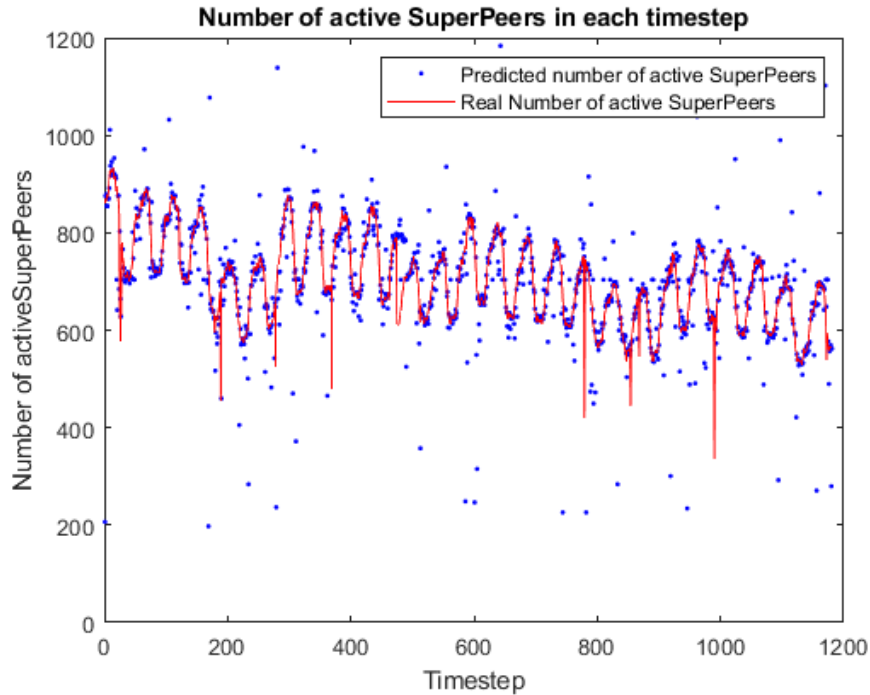


Figure 5.2: Real number of active Superpeers vs Predicted number of active Superpeers without outliers

Another important value we evinced from the tables is that 100 timesteps out of 140 timesteps record an error lower than 14.99, and 68 timesteps record an error lower than 9.99. This important fact shows that the system performs incredibly well on average while there are some minor outliers (which are NOT removed from the analysis) that importantly increase the total error.

Let us move to the second table; we report here 30 systems out of the 185. To assure an equal representation of the different systems we picked the best performing systems (error between 0 and 7000), good performing systems (error between 7001 and 14000), average performing systems (error between 14001 and 21000), bad performing systems (error between 21001 and 50000) and very bad performing systems (error over 50001), considering also the variety of different parameters values.

The Dynamic System achieved a total error of 5658 (we removed a single outlier with value over 1000). Notice that it is almost impossible to guess the behavior of the other fixed fuzzy systems before applying our tool. The graph shown below allows us to compare the Dynamic System performance with respect to the other systems:

The median of the total Error of all fixed Fuzzy Systems is 35095,80 which means that the majority of the 185 fixed fuzzy systems has a very high error implying that our system is performing fairly well:

| Timestep | Num of input vars used | Num of Memb. Fun. per input var. | Memb. Func. Type | Min. error on prev. timestep | Predicted Output | Real Output | Error with potential best system | Current error |
|----------|------------------------|----------------------------------|------------------|------------------------------|------------------|-------------|----------------------------------|---------------|
| 3 | 1 | 8 | psigmf | 12,92328051 | 787,3498702 | 792 | 0,102137151 | 4,650129827 |
| 4 | 2 | 3 | psigmf | 0,102137151 | 802,7111948 | 790 | 1,86847764 | 12,71119485 |
| 5 | 1 | 4 | psigmf | 1,86847764 | 789,8691117 | 809 | 0,075482901 | 19,1308883 |
| 6 | 4 | 2 | gaussmf | 0,075482901 | 885,8080332 | 811 | 2,650360319 | 74,80803323 |
| 7 | 2 | 3 | gbellmf | 2,650360319 | 807,4152837 | 812 | 0,012307417 | 4,584716273 |
| 8 | 1 | 2 | trapmf | 0,012307417 | 812,7834807 | 815 | 0,344161673 | 2,21651928 |
| 9 | 5 | 2 | gbellmf | 0,344161673 | 797,0223141 | 823 | 0,251637083 | 25,97768589 |
| 10 | 1 | 13 | gbellmf | 0,251637083 | 740,4139523 | 813 | 1,450126588 | 72,58604773 |
| 11 | 2 | 10 | gbellmf | 1,450126588 | 714,5853197 | 804 | 0,019006723 | 89,41468033 |
| 12 | 2 | 3 | gbellmf | 0,019006723 | 723,4333427 | 802 | 0,082564812 | 78,56665734 |
| 13 | 1 | 3 | trapmf | 0,082564812 | 799,7544804 | 792 | 0,095857707 | 7,754480368 |
| 14 | 2 | 3 | gaussmf | 0,095857707 | 1183,569481 | 801 | 2,176455374 | 382,5694814 |
| 15 | 1 | 2 | psigmf | 2,176455374 | 798,8596428 | 800 | 0,294590412 | 1,140357236 |
| 16 | 5 | 2 | gbellmf | 0,294590412 | 802,7752131 | 779 | 0,693247439 | 23,77521313 |
| 17 | 3 | 5 | gaussmf | 0,693247439 | 106,6398101 | 779 | 1,644967656 | 672,3601899 |
| 18 | 1 | 8 | gaussmf | 1,644967656 | 777,3550323 | 770 | 2,551566003 | 7,355032344 |
| 19 | 2 | 5 | psigmf | 2,551566003 | 808,4834555 | 738 | 1,636476892 | 70,48345546 |
| 20 | 6 | 2 | gbellmf | 1,636476892 | 782,4676405 | 701 | 2,187675501 | 81,46764054 |
| 21 | 2 | 13 | gaussmf | 2,187675501 | 43,75388169 | 684 | 3,124125694 | 640,2461183 |
| 22 | 3 | 3 | gbellmf | 3,124125694 | 675,9180438 | 664 | 0,047046577 | 11,9180438 |
| 23 | 2 | 13 | psigmf | 0,047046577 | 646,2583139 | 652 | 0,041077019 | 5,741686108 |
| 24 | 2 | 13 | psigmf | 0,041077019 | 636,4378699 | 634 | 0,003610821 | 2,437869904 |
| 25 | 1 | 3 | trimf | 0,003610821 | 607,9225611 | 626 | 0,151585027 | 18,0774389 |
| 26 | 2 | 2 | psigmf | 0,151585027 | 608,6688442 | 622 | 0,379060252 | 13,33115577 |
| 27 | 2 | 9 | gbellmf | 0,379060252 | 584,7002165 | 628 | 0,617124061 | 43,2997835 |
| 28 | 2 | 8 | gbellmf | 0,617124061 | 606,758406 | 635 | 0,182065655 | 28,24159398 |
| 29 | 2 | 2 | gbellmf | 0,182065655 | 635,4432636 | 634 | 0,801197657 | 1,443263615 |
| 30 | 3 | 5 | psigmf | 0,801197657 | 602,4676959 | 628 | 0,570978741 | 25,53230415 |
| 31 | 2 | 3 | gbellmf | 0,570978741 | 623,4265873 | 632 | 0,123260861 | 8,573412725 |
| 32 | 3 | 6 | trimf | 0,123260861 | 627,7427464 | 630 | 0,014218518 | 2,257253555 |

Figure 5.3: Detailed table 1

Let us remind that the best possible dynamic system (the ideal one) has an error of 73,2713, which means that there's more room to improve the performance and most important, we are going in the right path with the idea of a Dynamic System. The outliers are very important for our specific study because they illustrate the huge impact that choosing the right parameters combination has on the outcome. We can go from a minimum total error of 1539 to one of many millions. One peculiarity is that, on average, systems with a lot of input variables and membership functions seem to perform slightly worse than other systems. The reason we thought is that this kind of systems that have a high number of parameters which have to be tuned by the Neural Network, might lead the system in overfitting. We end this chapter remarking the goodness of a Self-Adaptive System like this, performing very well even if its performance evaluation is based on a mere (even if well-reasoned) hypothesis: at the next timestep the output is going to change and

what has been good for the previous timestep prediction might even be very bad for the current timestep prediction.

| Fuzzy System Parameter values | Total Error | Number of times picked as best system |
|---|-------------|---------------------------------------|
| Num.Segnali Passati :7, num. MF :2,MF type :trimf | 4321,340027 | 2 |
| Num.Segnali Passati :1, num. MF :4,MF type :psigmf | 1689,637542 | 1 |
| Num.Segnali Passati :1, num. MF :4,MF type :gbellmf | 1821,946656 | 0 |
| Num.Segnali Passati :7, num. MF :2,MF type :trimf | 4321,340027 | 2 |
| Num.Segnali Passati :1, num. MF :5,MF type :trapmf | 1683,126134 | 0 |
| Num.Segnali Passati :1, num. MF :5,MF type :trimf | 1758,582705 | 0 |
| Num.Segnali Passati :2, num. MF :7,MF type :trapmf | 10181,0608 | 3 |
| Num.Segnali Passati :6, num. MF :2,MF type :gaussmf | 10282,81328 | 0 |
| Num.Segnali Passati :3, num. MF :4,MF type :trapmf | 8978,783268 | 1 |
| Num.Segnali Passati :5, num. MF :2,MF type :gaussmf | 13741,70887 | 1 |
| Num.Segnali Passati :4, num. MF :2,MF type :gaussmf | 9333,573894 | 1 |
| Num.Segnali Passati :7, num. MF :2,MF type :gbellmf | 9513,939101 | 0 |
| Num.Segnali Passati :1, num.MF:12,MF type :gaussmf | 19449,3477 | 0 |
| Num.Segnali Passati :2, num. MF :6,MF type :trapmf | 17735,09858 | 1 |
| Num.Segnali Passati :2, num. MF :11,MF type :trapmf | 19939,24691 | 1 |
| Num.Segnali Passati :4, num. MF :2,MF type :gbellmf | 19108,4266 | 0 |
| Num.Segnali Passati :7, num. MF :2,MF type :trapmf | 18988,02084 | 0 |
| Num.Segnali Passati :5, num. MF :3,MF type :trimf | 15000,91636 | 0 |
| Num.Segnali Passati :2, num. MF :13,MF type :trapmf | 25738,43843 | 0 |
| Num.Segnali Passati :3, num. MF :4,MF type :trimf | 38026,16513 | 0 |
| Num.Segnali Passati :4, num. MF :3,MF type :gaussmf | 47856,43754 | 0 |
| Num.Segnali Passati :5, num. MF :2,MF type :gbellmf | 30198,97988 | 2 |
| Num.Segnali Passati :1, num. MF :9,MF type :gbellmf | 27309,27871 | 0 |
| Num.Segnali Passati :4, num. MF :2,MF type :trapmf | 49928,39621 | 1 |
| Num.Segnali Passati :1, num. MF :11,MF type :psigmf | 321145,9832 | 4 |
| Num.Segnali Passati :1, num. MF :12,MF type :psigmf | 159025,5279 | 0 |
| Num.Segnali Passati :1, num. MF :13,MF type :psigmf | 93904,68946 | 2 |
| Num.Segnali Passati :2, num. MF :6,MF type :psigmf | 100842,5201 | 0 |
| Num.Segnali Passati :2, num. MF :6,MF type :gbellmf | 156472,6781 | 2 |
| Num.Segnali Passati :2, num. MF :6,MF type :gaussmf | 181821,8757 | 0 |
| Num.Segnali Passati :5, num. MF :3,MF type :psigmf | 144806,7597 | 1 |
| Num.Segnali Passati :6, num. MF :2,MF type :psigmf | 601434,6847 | 1 |

Figure 5.4: Detailed table 2

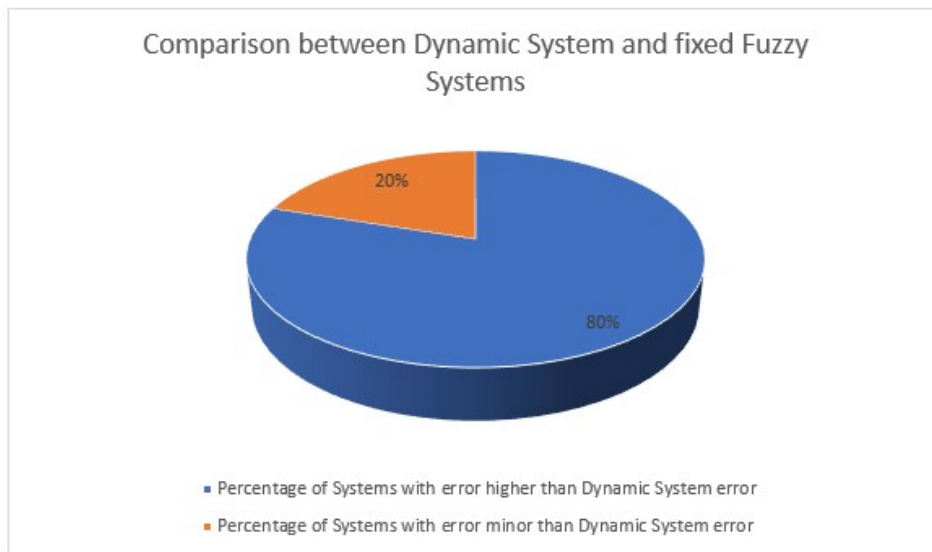


Figure 5.5: Comparison graph of the error of the Dynamic System and fixed Systems

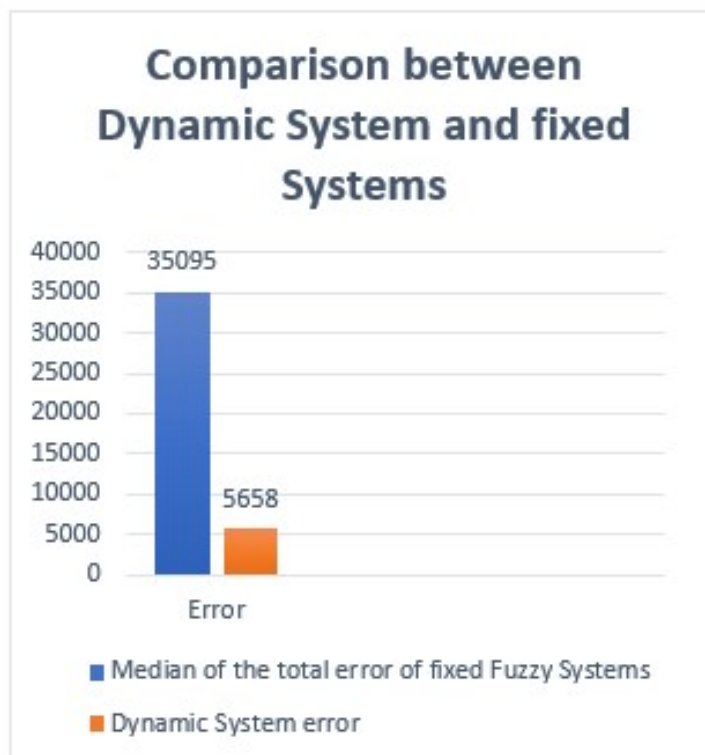


Figure 5.6: Comparison barcharts of the error of the Dynamic System with the median of the total error of fixed Fuzzy Systems

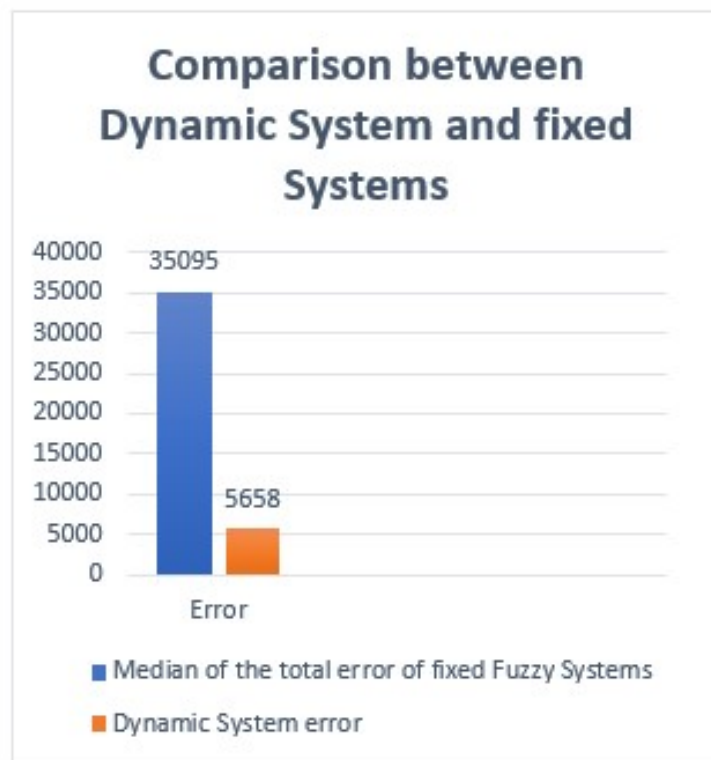


Figure 5.7: Comparison barcharts of the error of the Dynamic System with the mean of the total error of fixed Fuzzy Systems

Conclusions and future works

With our work we tried to create an innovative solution to the insidious problem of choosing the best Fuzzy Systems parameters. We've seen the huge impact in the prediction performance of picking a good or bad parameters values combination, giving birth to a very good or to a very bad estimator. In our specific case we noticed that on average there are certain membership function types, for us it was p-sigmoidal, which perform worse than others therefore resulting in a very high prediction error. This information was certainly not knowable before we applied our dynamic system. A possible future improvement would be to put a limit threshold, that has to be exceeded, on the attribute minimum error on previous timestep prediction. In this way if there are no fuzzy systems which are clearly better (for example considering only two numbers after the comma) than the system chosen for the previous timestep, the latter gets picked again. By doing this we'll be more able to understand which are the systems that get chosen more often and focus on them to improve the performance. This was just one of the many possible ideas that can improve what we did with our simple tool, in order to go in a direction leading towards always better results through the use of Artificial Intelligence techniques. There are a plenty of them already like Neural Networks, Genetic Algorithms, Fuzzy Systems, Swarm Intelligence, etc., and many more are coming into the world in this exact moment, making us confident that in the future we'll commit to achieve better results trying to get an asymptotic behavior when inexorably chasing (without reaching it) the perfection.

Bibliography

- [1] Christof Tannert, Horst-Dietrich Elvers, and Burkhard Jandrig. The ethics of uncertainty. In the light of possible dangers, research becomes a moral duty. *EMBO reports*, 8(10):892–6, oct 2007.
- [2] Diego Perez-Palacin and Raffaella Mirandola. Uncertainties in the Modeling of Self-adaptive Systems: a Taxonomy and an Example of Availability Evaluation. 2014.
- [3] Phillip G. Armour. The five orders of ignorance. *Communications of the ACM*, 43(10):17–20, oct 2000.
- [4] Remus-alexandru Dobrican and Denis Zampunieris. Moving Towards Distributed Networks of Proactive , Self-Adaptive and Context-Aware Systems : a New Research Direction ? 7(3):262–272.
- [5] José Vieira, Fernando Morgado Dias, and Alexandre Mota. Neuro-Fuzzy Systems: A Survey. Technical report.
- [6] An architectural blueprint for autonomic computing. Technical report, 2005.
- [7] Saikat Guha, Neil Daswani, and Ravi Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *Proceedings of The 5th International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 1 – 6, Santa Barbara, CA, February 2006.
- [8] Malte Faber, Reiner Manstetten, and John L.R. Proops. Humankind and the Environment: An Anatomy of Surprise and Ignorance. *Environmental Values*, 1(3):217–241, aug 1992.
- [9] Matthias Gross. The Unknown in Process. *Current Sociology*, 55(5):742–759, sep 2007.
- [10] Jeffrey A Weber. SPAEF A response to Public Administration’s lack of a general theory of uncertainty . A theoretical vision of uncertainty. Technical Report 1.

- [11] Jeffrey A. Weber. Certainty and uncertainty in decision making: A conceptualization. In Robert T. Golembiewski and Jack Rabin, editors, *Public budgeting and finance*, chapter 23, pages 449–473. Marcel Dekker Inc, 4th ed. edition, 1997.
- [12] Jiří Pospíchal. Fuzzy Sets and Fuzzy Logic: Theory and Applications. By George J. Klir and Bo Yuan. Prentice Hall: Upper Saddle River, NJ, 1995. 574 pp. \$60.00. ISBN 0-13-101171-5. Sales e-mail: corpsales@prenhall.com. *Journal of Chemical Information and Computer Sciences*, 36(3):619–619, jan 1996.
- [13] Vern Walker. Theories of Uncertainty: Explaining the Possible Sources of Error in Inferences. *Cardozo Law Review*, jul 2001.
- [14] W.E. Walker, P. Harremoës, J. Rotmans, J.P. van der Sluijs, M.B.A. van Asselt, P. Janssen, and M.P. Kraye von Krauss. Defining Uncertainty: A Conceptual Basis for Uncertainty Management in Model-Based Decision Support. *Integrated Assessment*, 4(1):5–17, mar 2003.
- [15] Jon C. Helton, Jay D. Johnson, William L. Oberkampf, and Cédric J. Sallaberry. Representation of analysis results involving aleatory and epistemic uncertainty. *International Journal of General Systems*, 39(6):605–646, aug 2010.
- [16] G Schafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
- [17] Naeem Esfahani, Ehsan Kouroshfar, and Sam Malek. *Taming Uncertainty in Self-Adaptive Software*. 2011.
- [18] Naeem Esfahani and Sam Malek. Uncertainty in self-adaptive software systems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7475 LNCS, pages 214–238. Springer, Berlin, Heidelberg, 2013.
- [19] Roberto Lucchetti. *A Primer in Game Theory*. Esculapio, 2011.
- [20] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, jun 1965.
- [21] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(1):3–28, jan 1978.
- [22] Francesco Marchesani. *Dealing with uncertainties in availability models using fuzzy logic and neural networks*. PhD thesis, Politecnico di Milano, 2017.
- [23] Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw. Engineering self-adaptive systems through feedback loops. In *Lecture Notes in Computer*

-
- Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5525 LNCS, pages 48–70. Springer, Berlin, Heidelberg, 2009.
- [24] Detlef Nauck, Rudolf Kruse, and F. Klawonn. *Foundations of neuro-fuzzy systems*. John Wiley, 1997.
- [25] Ernest Czogała and Jacek Łeski. *Fuzzy and Neuro-Fuzzy Intelligent Systems*, volume 47 of *Studies in Fuzziness and Soft Computing*. Physica-Verlag HD, Heidelberg, 2000.
- [26] Bart Kosko. *Neural networks and fuzzy systems : a dynamical systems approach to machine intelligence*. Prentice Hall, 1992.
- [27] C.-T. Lin and C.S.G. Lee. Neural-network-based fuzzy logic control and decision system. *IEEE Transactions on Computers*, 40(12):1320–1336, 1991.
- [28] H.R. Berenji and P. Khedkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Transactions on Neural Networks*, 3(5):724–740, 1992.
- [29] Detlef Nauck and Rudolf Kruse. Neuro-fuzzy systems for function approximation. *Fuzzy Sets and Systems*, 101(2):261–271, jan 1999.
- [30] Shun'ichi Tano, Takuya Oyama, and Thierry Arnould. Deep combination of fuzzy inference and neural network in fuzzy inference software — FINEST. *Fuzzy Sets and Systems*, 82(2):151–160, sep 1996.
- [31] S.M. Sulzberger, N. Tschichold-Gurman, and S.J. Vestli. FUN: optimization of fuzzy rule based systems using neural networks. In *IEEE International Conference on Neural Networks*, pages 312–316. IEEE.
- [32] Chia-Feng Juang and Chin-Teng Lin. An On-Line Self-Constructing Neural Fuzzy Inference Network and Its Applications. Technical Report 1, 1998.
- [33] M. Figueiredo and F. Gomide. Design of fuzzy systems using neurofuzzy networks. *IEEE Transactions on Neural Networks*, 10(4):815–827, jul 1999.
- [34] Nikola Kasabov and Qun Song. Dynamic Evolving Fuzzy Neural Networks with 'm-out-of-n' Activation Nodes for On-line Adaptive Systems The Information Science Discussion Paper Series Dynamic Evolving Fuzzy Neural Networks with Ôm-out-of-nÕ Activation Nodes for On-line Adaptive Systems 1. Technical report.

- [35] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, jan 1975.
- [36] Jyh-Shing Roger Jang. ANFIS : Adaptive-Ne twork-Based Fuzzy Inference System. Technical Report 3, 1993.
- [37] Jyh-Shing Roger Jang, C. T. Sun, and E. Mizutani. *Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, 1997.