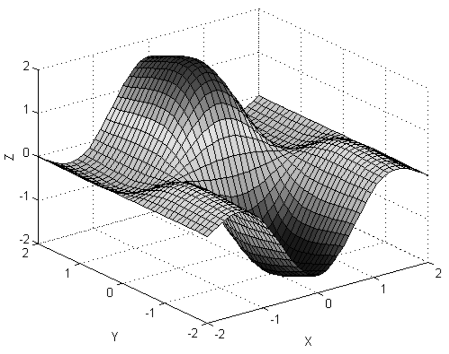
N°0019/2001

THESE PRESENTEE POUR OBTENIR LE GRADE DE DOCTEUR
ES SCIENCES DE L'UNIVERSITE NATIONALE DU BENIN

OPTION : Physique mathématique

Par Eugène C. EZIN

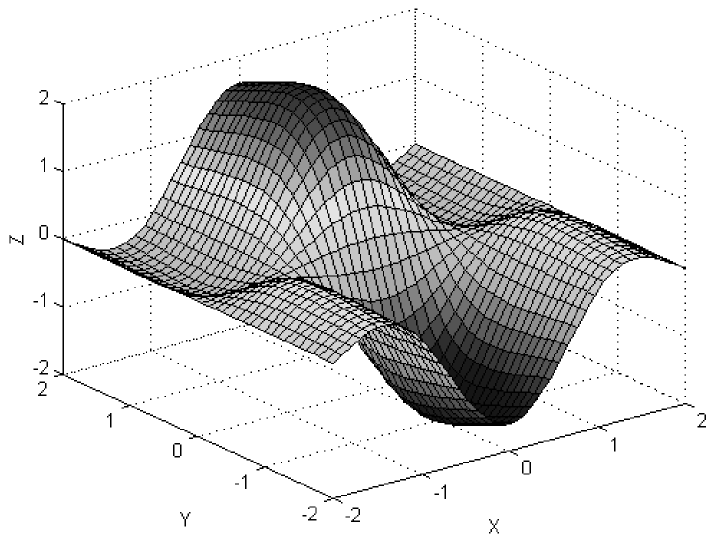
Sujet : NEURAL NETWORKS AND NEURAL FUZZY SYSTEMS FOR SPEECH APPLICATIONS



Soutenue le 28 mars 2001, devant le Jury

Président	: Prof. Jean-Pierre EZIN, IMSP, Université Nationale du Bénin, BENIN
Rapporteurs	: Dr. Anna ESPOSITO, IIASS and Salerno University, ITALY Prof. Maria MARINARO, IIASS and Salerno University, ITALY Prof. Emmanuel TONYE, ENP, LETS, Yaoundé, CAMEROUN Prof. M. Norbert HOUNKONNOU, IMSP, Université Nationale du Bénin, BENIN
Examinateur	: Prof. Cyprien GNANVO, FAST, Université Nationale du Bénin, BENIN
Co-Directeurs	: Dr. Anna ESPOSITO, IIASS and Salerno University, ITALY Prof. Maria MARINARO, IIASS and Salerno University, ITALY Prof. M. Norbert HOUNKONNOU, IMSP, Université Nationale du Bénin, BENIN

NEURAL NETWORKS AND NEURAL FUZZY SYSTEMS FOR SPEECH APPLICATIONS



Eugène C. EZIN

eugene.ezin@imsp-uac.org

Institut de Mathématiques et de Sciences Physiques

UNIVERSITE NATIONALE DU BENIN

Tel :+229/20 222 455 Fax :+229/20 222 455

UNIVERSITE NATIONALE DU BENIN

International Centre for Theoretical Physics ABDUS SALAM - ICTP

Institut de Mathématiques et de Sciences Physiques – IMSP
B. P. 613 Porto-Novo, Bénin.
Tel: +229 22 24 55 Fax: +229 22 24 55 email: hkimsp@syfed.bj.refer.org.

International Institute for Advanced Scientific Studies – IIASS
Via Pellegrino 19, 84019 Vietri sul Mare, SA, Italy.
Tel: +39 089 761167 Fax: +39 089 761189 email: iiass@tin.it.

Ph.D. Dissertation

Physics – Mathematics

NEURAL NETWORKS AND NEURAL FUZZY SYSTEMS FOR SPEECH APPLICATIONS

Supervised by:
Dr. Anna ESPOSITO
Prof. Maria MARINARO
Prof. Norbert M. HOUNKONNOU.

Presented by:
Eugène C. EZIN

February 2001

ABSTRACT

Speech is the most used means for human beings communication. Among the major constraints for general applications of automatic speech recognition, the presence of unavoidable background noise is of great importance since the speaker cannot be isolated to obtain a clean acquisition of the uttered speech to be processed. The separation of noise and speech with traditional filtering techniques is hard since respective spectra overlap each other in frequency-domain. One of the successful approaches to remove noise from speech is the adaptive noise canceling which aims to subtract noise from a received signal in an adaptive manner. Several algorithms developed under this approach have shown good performances. Optimization techniques like neural networks and fuzzy inference systems offer the potentiality to deal with encoded data by learning and reasoning as humans do. Neural Networks have a remarkable learning capability such that a desired input-output mapping can be discovered through learning by examples. Their use in an adaptive noise cancellation system, allows the compensation of channel effects. On the other hand, Fuzzy Inference Systems success is mostly due to the fact that fuzzy if-then rules are well-suited for capturing the imprecise nature of human knowledge and reasoning process. The resulting system that combines these two schemes produces robust systems for noise cancellation problem.

We report, on this dissertation, an analysis in integrating both neural networks and fuzzy inference systems for noise cancellation from speech signal.

We implemented a nonlinear model to cancel noise from the speech signal based on Adaptive Neuro-Fuzzy Inference System and exploiting Widrow's approach. The system is tested in different noisy environments. Performance studies are carried out for qualitative and quantitative evaluations.

Time is an important factor when designing a system for a given task. A fast neuro-fuzzy inference system is proposed for speech noise cancellation in almost real-time computation.

Since noise is distorted by a highly nonlinear process before corrupting the speech signal, we proposed some passage dynamic functions that model the environment through *which the noise waves undergo before corrupting the speech signal. Moreover, we implemented a neuro- fuzzy system able to identify the noise source before canceling its waves from the speech signal.*

Classification is an important problem in the pattern recognition field. Before being used by the speech recognition system, speech sentences must be preprocessed. Two preprocessing algorithms namely Linear Prediction Coding (LPC), and RASTA (RelAtive SpecTrAl) techniques are used for extracting features from the speech signals to classify both English stops and noise soures. Time Delay Neural Networks and Recurrent Neural Networks are used as classifiers.

ACKNOWLEDGMENTS

The contributions I present in this dissertation could not have been possible without the collaboration and constant support that I received from all the members of the International Institute for Advanced Scientific Studies. I am immensely grateful to my Supervisor, Professor Maria Marinaro, Dean of IIASS for offering me a job position as research Assistant in her Institute and for making available all the equipment necessary to go further in research. My very special thanks have to be expressed to my Supervisor Dr. Anna Esposito, who introduced me into speech processing and neural networks. For nearly five years, we have had an intense cooperation, and I think she is not only an excellent technical colleague, but most important a friend. Part of this dissertation was based on her input and original ideas. I am indebted to Dr. Carlos Reyes-Garcia who directed me into fuzzy systems. He is not only an advisor but has also become a Mexican brother. I also realized that neuro-fuzzy systems constitute the future of Soft Computing. My gratitudes to my Supervisor, Professor Norbert M. Hounkonnou for his advice. My gratitudes to Professor Jean-Pierre Ezin, Director of IMSP for his efforts to the development of research in Africa.

Special thanks to the kind and dynamic secretaries Ornella de Pasquale and Tina Nappi at IIASS.

I would like to thank Dr. Alfredo Petrosino, Dr. Antonio Natale, Dr. Joël T. Hounsou, Dr. Joël Tossa, for their fruitfull collaboration.

My deep gratitudes to my brothers and sisters for their unvaluable help over the years. Special thanks to my sisters in-laws.

My infinite gratitudes to my wife Sylphide who shared with me this incredible experience and whose love and support gave me the energy necessary to reach my goal.

Last but not least, my very special thanks to Mr. Innocent Abatti, Mr. & Mrs Adjovi, Mr. Gilbert Avocetien, Mr. & Mrs. Durand, Mrs. Aimée Freitas, Mr. & Mrs. Kpara, Mr. Benoît Lokossou, Mr. Soulé R. Sanny, Mr. Octave Amadou, for being in contact *with me for encouragments in all these years. It is sometimes, a good timing to receive letters, electronic mails or phone calls.*

This work is financially supported by IIASS - Eduardo R. Caianiello and the International Centre for Scientific Culture (ICSC – World Laboratory).

DEDICACE

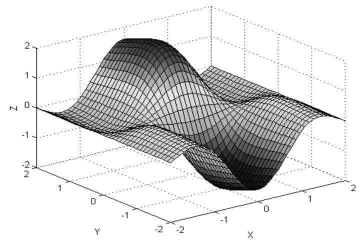
I dedicate this work to:

my Late Parents David & Irène

my Spouse Claire Sylphide,

my Daughter Vénérose and ...

NEURAL NETWORKS AND NEURAL FUZZY SYSTEMS FOR SPEECH APPLICATIONS



Eugène C. EZIN
email: e_ezin@hotmail.com.
IMSP / UNB, BP. 613 Porto-Novo
Tel: +229 222455, Fax: +229 222455

Contents

List of Figures

1. Introduction

So far as the laws of mathematics refer to reality they are not certain. And so far as they are certain, they do not refer to reality.
– Albert Einstein.

1.1. Introduction

One of the major problems in Automatic Speech Recognition (ASR) is the use of speech as input to a computer. For most real applications, the speaker cannot be isolated for acquiring a clean speech sentence. In general, the background noise and internal noise corrupt the speech signal. Many filtering techniques have been proposed to remove noise from a speech signal. Most of such methods are based on time-domain and frequency-domain. Systems based on time-domain do not always lead to efficient results since the estimated sentence is not qualitative. On the other hand, the drawback of systems based on frequency-domain is that noise and speech signals have overlapped spectra. One of the successful approaches is the adaptive filtering which aims to subtract noise from a received signal in an adaptive manner. Several algorithms developed under this concept have shown that it is possible to achieve good performance. Due to the changing nature of noise, systems with the capabilities to adapt their answer to the changing pattern features and able to implement a nonlinear mapping from the input to the output space should perform better than other traditional filtering techniques. Recognition systems based on artificial neural networks have the capability of learning from samples while fuzzy inference systems use the imprecision of human reasoning to deal with data. A better approach is to use hybrid systems which combine these two learning techniques.

Systems implemented in the present dissertation exploit both neural network potentialities and fuzzy system capabilities.

1.2. Motivation

Noise is everywhere and is introduced into the speech signal through different environments such as moving cars or crowded public gathering places. Modeling a system for nonlinear adaptive noise cancellation could be useful for a wide range of off-line applications namely voiced mail, voiced messaging, cleaning recordings for criminal investigations, hand-free mobile telephony, audio-conferencing. System modeling based on mathematical tools such as transfer equations, differential equations are appropriated and well justified when it is possible to describe exactly each parameter involved in the system. However, when the variables describing the system are unclear or are known with uncertainty, mathematical tools become less efficient. Neural Networks and Fuzzy Logic modeling have been proposed as alternative optimization techniques to overcome the above mentioned problem since they can provide satisfactory solutions due, to their capabilities to deal with learning by examples, fault tolerance, parallel distribution of data, imprecise nature of human reasoning. Neural Network modeling is motivated by biological neural systems while Fuzzy Logic modeling is closely related to psychology and cognitive science. Integrating these two schemes can generate hybrid models that provide good systems able to solve real-world problems like noise cancellation from a speech signal. Many hybrid systems are proposed depending on the objective. Among such systems, the Adaptive Neuro-Fuzzy Inference System (ANFIS) [40] is one of the most promising for parameters' estimation. ANFIS is a hybrid learning algorithm that combines the least-squares method and the backpropagation gradient descent method.

1.3. Thesis Objective

In this dissertation, we report on nonlinear adaptive models based on ANFIS that integrates both Neural Network and Fuzzy Inference System potentialites, to cancel noise from a speech signal in different noisy environments, and to identify the nature of the noise sources. The proposed systems are based on Widrow approach [84] in defining the primary input for the implemented system. We showed that it is possible to achieve very good performance with systems based on neuro-fuzzy approaches. The results obtained are evaluated quantitatively and qualitatively.

Modeling the channels through which the noise goes to corrupt the speech signal is a challenge and difficult problem in nature. Indeed, there is a priori no precise statistical knowledge about how the channels behave and on the effects of the changing environment upon them. To model them, we proposed some nonlinear functions (called passage dynamic functions) that describe the channel through which the noise waves undergo to corrupt the speech signal [19, 20, 21].

To perform well Adaptive filtering systems require a sample of noise source. This requirement limits their application to situation where the recording of noise sources are available. One of the requirements when applying adaptive filtering techniques [84] is that the noise source must be known and not correlated to the speech. To overcome such a requirement, we proposed a neuro-fuzzy system able to identify a noise source before evaluating the estimated sentence. Performance of the implemented system is evaluated [23].

Another problem which arises in implementing neuro-fuzzy systems is the computational time which is very high due to the training process. Our approach, to avoid such a problem for the proposed system, is to train the system over the longest sentence in our bilingual database. The fuzzy inference system generated is saved to evaluate each sentence in the database. The results showed that in a few seconds, the implemented system is able to produce the estimated sentence close to the original one [22].

To be used by a recognition system, speech sentences must be preprocessed. Classification with neural networks is an important problem. We reported two experiments in classifying speech units and four different noise sources. We found that neural networks are good classifiers when appropriate architecture and setup are used [24, 25, 26].

1.4. Outline

This dissertation is divided into two major sections. The first one is devoted to the theory necessary to understand fuzzy logic modeling, neural networks modeling and noise filtering techniques whereas the second presents our contribution to this dissertation.

The first section is composed of three chapters. Chapter 2 contains a review on fuzzy sets theory. This first discussion will introduce the necessary notations, fundamental concepts, and fuzzy real-world applications are presented. Chapter 3 presents the major architectures used in Neural Network theory and the well-known and most used back-propagation algorithm. Adaptive Networks, Multilayer Perceptron, Time Delay Neural Networks, Radial Basis Function Networks, Recurrent Networks and Adaptive Networks are reviewed. We focused on a hybrid learning algorithm that combined the least-squares estimator and the gradient descent algorithm. In Chapter 4, we review some existing techniques relative to noise cancellation from speech signal.

The second section contains five chapters that are our contribution to this thesis. Chapter 5 presents the ANFIS architecture used in our system implementation with the bilingual database for testing the system performance. We gave a full description of a hybrid system proposed for noise cancellation from speech signal [19, 20]. The learning procedure is reported. Chapter 6 presents a fast neuro-fuzzy system for noise cancellation in almost real-time computation [21, 22]. Chapter 7 presents a full description of a system able to identify the noise waves in a noisy sentence [23]. Identification criterion is introduced. Chapter 8 presents one of the most attractive topics in speech recognition that is classification with neural networks. In this chapter, we report the results of the classification of speech units and four noise sources. Different preprocessing algorithms are reported [24, 25, 26]. Chapter 9 concludes this work giving ideas for further investigation.

2. Overview of Fuzzy Sets Theory

As complexitiy rises, precise statments lose meaning and meaningful

statments lose precision.

– **Lotfi A. Zadeh.**

Introduction

Classical sets are suitable for various applications and have proved to be an important tool in mathematics and computer science. However, when asking anyone how the weather is, the natural answer could be "it is hot", or "it is cold" and so on - but not the exact temperature value, i.e. 25° Celsius for example. It appears that mathematical tools do not reflect the nature of human concepts, thoughts and reasoning. As a further example, let's assume A *to be a set defined as* $A = \{x \in \mathbb{R} \mid x > 1\}$ *and let's consider the two numerical values* $x_1 = 1.001$ *and* $x_2 = 0.999$*. The dichotonomous nature of the classical sets theory would classify* x_1 *in* A *but not* x_2 *since the transition between inclusion and exclusion is sharp. In contrast to classical sets, fuzzy sets are without crisp boundaries. In other words, the transition from belonging or not belonging to a set is gradual. This smooth transition is characterized by membership functions that give fuzzy sets flexibility in modeling commonly used linguistic expressions.*

This chapter introduces the basic definitions, notations and operations for fuzzy sets, including their membership function representations, sets theoretic operators (AND, OR, NOT), and advanced operators such as T-norm and T-conorm. Finally, fuzzy reasoning, fuzzy relations and fuzzy logic real-world applications are presented.

2.1. Definition

Let X be the discrete or continuous universe of discourse. A fuzzy set A in X is a set of ordered pairs of the form $A = \{(x, \mu_A(x)) \,|\, x \in X\}$ where $\mu_A(x)$ is a membership function grade for the fuzzy set A. In other words, a fuzzy set is a class with unsharp boundaries in which the transition from membership to nonmembership is gradual rather than abrupt [53]. In practice, most fuzzy sets used have a universe of discourse consisting of the real line $\mathbb{R}$. Fuzzy sets are introduced by generalizing the characteristic function.

2.1.1. Convexity and Normality

A fuzzy set A in X is convex according to its membership function μ_A, if and only if the following condition is satisfied:

$$\forall\, x_1,\ x_2 \in X,\ \forall\, \lambda \in [0,1],\ \mu_A[\lambda x_1 + (1-\lambda)x_2] \geq \min[\mu_A(x_1), \mu_A(x_2)].$$

This definition of convexity of a fuzzy set is not as strict as the common definition of convexity of a crisp set in $\mathbb{R}^n$, n-dimensional space [1]. A fuzzy set A in X is normal if and only if:

$$\exists\, x_0 \in X \,:\, \mu_A(x_0) = 1.$$

A fuzzy set A in X that satisfies both the convexity and normality requirements is called a fuzzy number. Most fuzzy sets used in literature are fuzzy numbers.

2.1.2. Cardinality

A basic notion of an ordinary or crisp set A is its cardinality $|A|$, that is the number of elements belonging to A. Since the notion of belonging loses its meaning when A is a fuzzy set, it is not meaningful to speak of the number of elements in a fuzzy set. However, the notion of cardinality may be extended to fuzzy sets by defining it in a more general sense as

$$|A| = \sum_{x \in X} \mu_A(x)$$

[1]A crisp set C in $\mathbb{R}^n$ is convex if and only if, the following condition is satisfied

$$\forall\, x_1,\ x_2 \in C,\ \forall\, \lambda \in [0,1],\ \lambda x_1 + (1-\lambda)x_2 \in C.$$

where $\mu_A(x)$ is the membership function of the fuzzy set A in X. For instance, if X is a group of four men named *Bob, John, Peter, Tony,* and A the fuzzy set of *Tall men* in X, then we can write symbolically:

$$X = \{Bob,\ John,\ Peter,\ Tony\}$$

and

$$A = \{(Bob, 0.4),\ (John, 0.7),\ (Peter, 0.1),\ (Tony, 0.5)\}$$

where $\mu_A(Bob) = 0.4$, $\mu_A(John) = 0.9$, $\mu_A(Peter) = 0.1$, and $\mu_A(Tony) = 0.5$. One has $|X| = 4$ and $|A| = 1.7$.

2.2. Membership Functions

Let X be a discrete or continuous universe of discourse and A a fuzzy set in X. The membership function for the fuzzy set A denoted $\mu_A(x)$ maps each element of X to a membership grade onto the set M which could be assumed to be $[0,1]$ as usual without loss of generality. Otherwise, the resulting set M could be normalized [2].

$$\begin{array}{cccll} \mu_A & : & X & \rightarrow & M \\ & & x & \longmapsto & \mu_A(x) \end{array}$$

When the membership grade is restricted to either 0 or 1, then the fuzzy set A is reduced to a classical set where $\mu_A(x)$ is the characteristic function. An object x belongs to a classical set A if and only if $1_A(x) = 1$ where 1_A denotes the characteristic function defined as

$$1_A(x) = \begin{cases} 1 & \text{if} \quad x \in A \\ 0 & \text{if} \quad x \notin A. \end{cases}$$

Classical sets are closely related to characteristic function.

The construction of a fuzzy set requires both the identification of a suitable (discrete or continuous) universe of discourse and of an appropriate membership function. As pointed out by Zadeh [89], a fuzzy set expresses the degree to which an element belongs to a set. Hence, the characteristic function of a fuzzy set is allowed to have values between 0 and 1 and such values indicate the degree of membership of an element in the given set.

[2]Assume $M = [p, q] \subset \mathbb{R}$ with $p < q$. It is easy to prove that $t \in M$ is equivalent to $\frac{t-p}{q-p} \in [0,1]$

Assignement of a membership function to a fuzzy set is subjective in nature. However, it cannot be assigned arbitrarly. A more convenient and concise way to define a membership function is to express it as a mathematical formula. There exists various types of membership functions depending upon the concept and the problem faced [62]. Fuzzy sets considered reviewed in this dissertation are convex and normal except those with sigmoidal membership function. The following functions are mostly used in practice, due to their tractability in modeling the linguistic variables.

2.2.1. Gaussian Membership Functions

Gaussian membership functions are characterized by two parameters: the mean c and the standard deviation β. The numerical value β represents the center of the membership function whereas c determines the width[3] and the shape of the membership function. A Gaussian membership function is expressed as:

$$gaussmf(x; c, \beta) = e^{\frac{-1}{2}(\frac{x-c}{\beta})^2}.$$

Gaussian membership functions are becoming popular for specifying fuzzy sets since they are well smooth. Figures 2.1 and 2.2 show the effects of changing different values of the mean c and the standard deviation β.

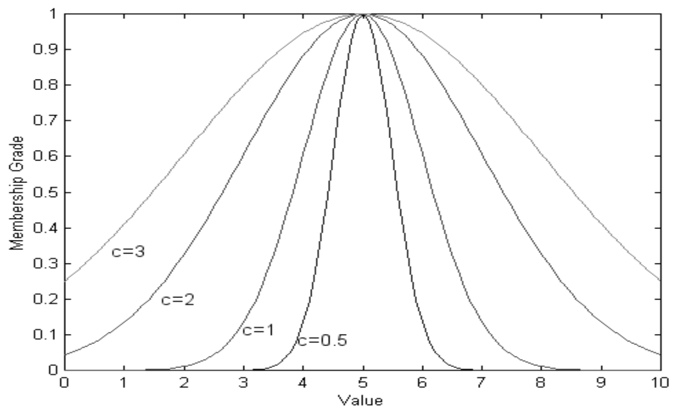


Figure 2.1.: Effect of changing the values of the mean c in a Gaussian membership function. Numerical values used are 0.5, 1, 2, 3 respectively whereas the standard deviation β is fixed to 5.

[3]The width of a membership function is defined as the Euclidean distance between the two unique crossover points.

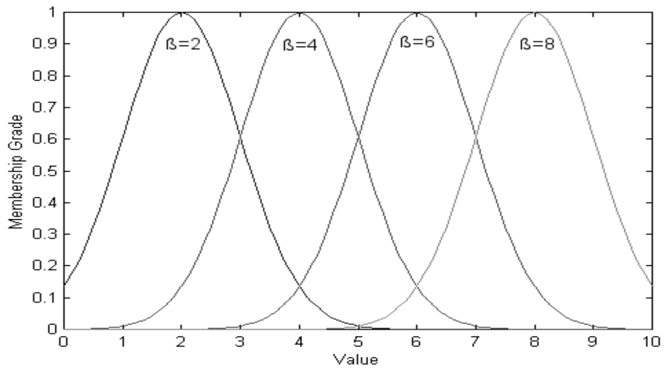


Figure 2.2.: Effect of changing the values of the standard deviation β in a Gaussian membership function. The values used are 2, 4, 6, 8 respectively whereas the mean value c is fixed to 1.

Gaussian functions are well known in probability and statistics, and they possess useful properties such as invariance under multiplication, and Fourier transform [70] and can model the fuzzy concept like *around*.

2.2.2. Generalized Bell Membership Functions

The generalized bell membership functions have one more fitting parameter than the Gaussian membership functions. Therefore, they have one more degree of freedom to adjust the steepness at the crossover points[4]. A generalized bell membership function is defined as

$$gbellmf(x;a,b,c) = \frac{1}{1+(\frac{x-c}{a})^{2b}}$$

where a is half the width of the membership function, b controls the slopes at the crossover points and c determines the center of the membership function. Figures 2.3, 2.4, 2.5 illustrate the effects of changing each parameter in a bell membership function.

2.2.3. Sigmoidal Membership Functions

The asymmetric property of a sigmoidal function is important for representing concepts such as *very large* or *very negative*. Sigmoid functions are widely used in the artificial neural network theory as activation function and can be considered as a continuous

[4]A crossover point of a fuzzy set A is a point $x \in A$ at which $\mu_A(x) = 0.5$

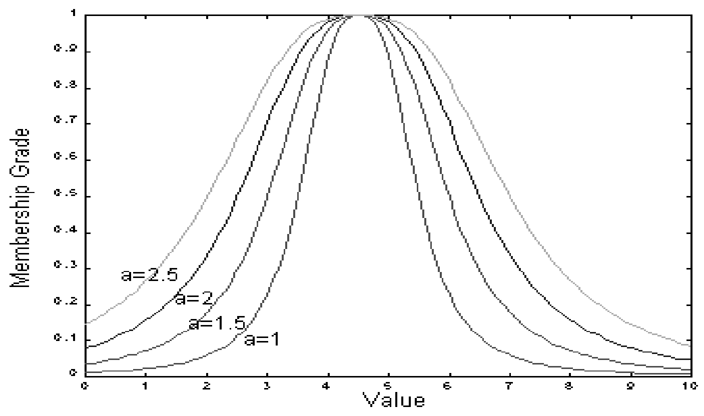


Figure 2.3.: Controling the width of a generalized bell function. The values of the parameter a are 1, 1.5, 2, 2.5 respectively whereas parameters b and c are fixed ($b = 1.5$ and $c = 4.5$).

approximation of the step function[5]. A sigmoid function depends upon two parameters and is defined as:

$$sigmf(x; a, c) = \frac{1}{1 + e^{-a(x-c)}}.$$

One can easily prove that the crossover point is reached at $x = c$. In practice, a is a positive value. The logistic function is a special case of sigmoid function when $a = 1$ and $c = 0$. Figure 2.6 shows the effects of using negative values for the parameter a whereas Figure 2.7 shows the plot of logistic function.

2.2.4. Triangular Membership Functions

Triangular membership functions are defined as:

$$triangmf(x; a, b, c) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ \frac{c-x}{c-b} & b \leq x \leq c \\ 0 & x \geq c. \end{cases}$$

The parameters a, b, c (with $a \leq b \leq c$) determine the x coordinates of the three corners of the underlying triangular membership functions. One can easily prove that a

[5]Also known as signum function, the step function is defined as

$$h(x) = \begin{cases} 1 & \text{if} \quad x > x_0 \\ 0 & \text{if} \quad x \leq x_0. \end{cases}$$

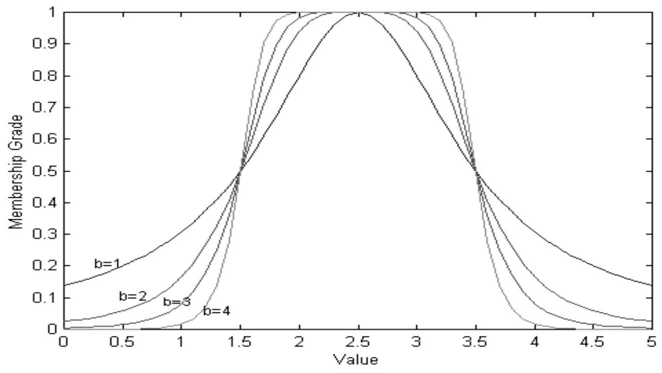


Figure 2.4.: Controling the slopes of a generalized bell function. The values used for the parameter b are 1, 2, 3, 5 respectively whereas the parameters a and c are fixed ($b = 1$ and $c = 2.5$).

triangular function can also be expressed as:

$$triangmf(x;a,b,c) = \max\left(\min\left(\frac{x-a}{b-a},\frac{c-x}{c-b}\right),0\right).$$

Figure 2.8 shows the effects of changing values of different parameters that define the triangular function.

2.2.5. Trapezoidal Membership Functions

Trapezoidal membership functions are defined as:

$$trapezoidmf(x;a,b,c,d) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & b \leq x \leq c \\ \frac{d-x}{d-c} & c \leq x \leq d \\ 0 & x \geq d. \end{cases}$$

The parameters a, b, c, d determine the x coordinates of the four corners of the underlying trapezoidal membership functions. Notice that when b is equal to c, trapezoidal membership functions are reduced to triangular membership functions. Due to their simple formulas and computational efficiency, both triangular and trapezoidal membership functions have been used extensively, especially in real-time implementations [23, 39]. Figure 2.9 shows the shapes of different trapezoidal functions.

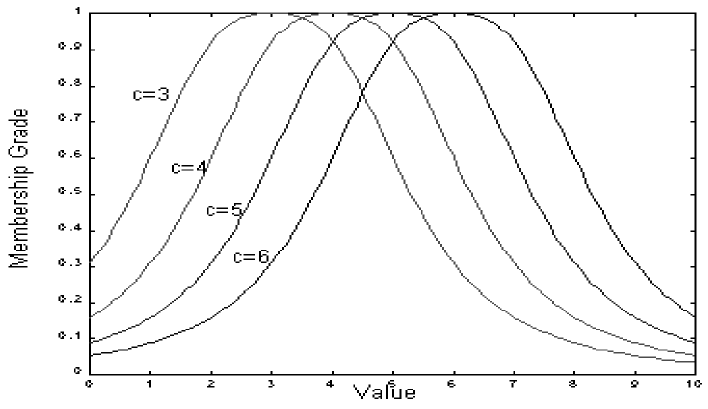


Figure 2.5.: Determination of a center of a generalized bell function. The values used for the parameter c are 2, 2.3, 2.6, 2.9 respectively whereas the parameters a and b are fixed ($b = 1.5$ and $b = 4.5$).

2.3. Classical Fuzzy Set Operators

Classical fuzzy operators as *equality, containment, intersection, union, and complement* are the generalization of the crisp set operators. In fuzzy logic, such operators are defined in terms of their membership functions. Moreover, the last three operators can also be viewed as the generalization of logical OR, logical AND and logical NOT respectively [62]. In this section and the followings, we will assume X to be a discrete or continuous universe of discourse.

2.3.1. Fuzzy Equality

A fuzzy set A in X is equal to a fuzzy set B in X if and only if their respective membership functions are equal. In symbols,

$$A = B \Longleftrightarrow \forall x \in X,\ \mu_A(x) = \mu_B(x).$$

This definition of equality is crisp. Lin and Lee in [54], proposed the *similarity measure* to check the degree of equality of two fuzzy sets. Given two fuzzy sets A and B in X, the *similarity measure* is defined as $E(A,B) = \frac{|A \cap B|}{|A \cup B|}$.

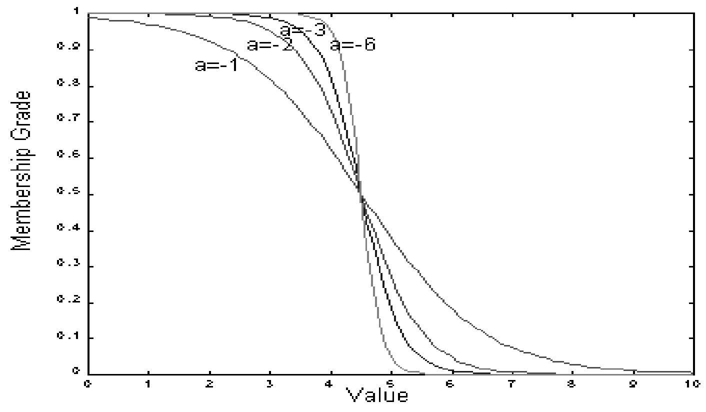


Figure 2.6.: Effects of using negative values of the paramter a for sigmoid function. The values used for the parameter a are -1, -2, -3, -6 respectively whereas the parameters c is fixed to 4.5.

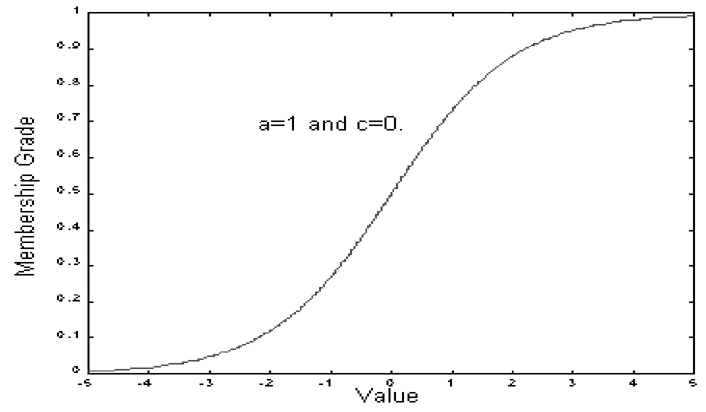


Figure 2.7.: Plot of logistic function ($a = 1$ and $c = 0$).

2.3.2. Fuzzy Subset

A fuzzy set A in X is contained in a fuzzy set B in X if and only if their membership functions μ_A and μ_B satisfy the inequality $\mu_A \leq \mu_B$. In other words,

$$A \subset B \Longleftrightarrow \forall x \in X,\ \mu_A(x) \leq \mu_B(x).$$

As for fuzzy equality, the definition of fuzzy subset (also known as fuzzy containment) is crisp. To check the degree that A is a subset of B, Kosko in [49] proposed the *subsethood measure* defined as $S(A, B) = \frac{|A \cap B|}{|A|}$.

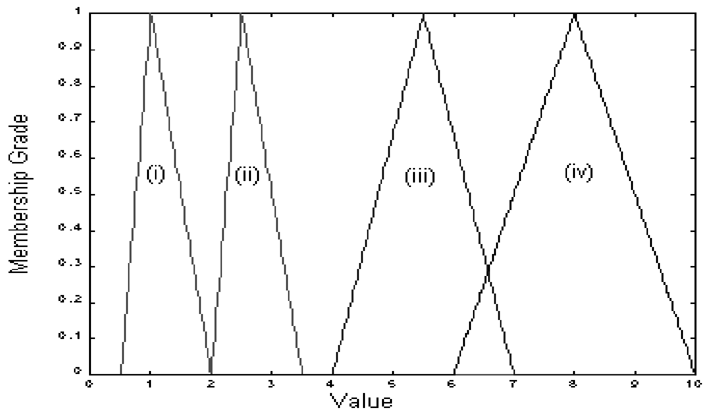


Figure 2.8.: Effects of changing values of the different parameters that define the triangular function. One has (i): $a = 0.5$, $b = 1$, $c = 2$; (ii): $a = 2$, $b = 2.5$, $c = 3.5$; (iii): $a = 4$, $b = 5.5$, $c = 7$; (iv): $a = 6$, $b = 8$, $c = 10$.

2.4. Negation Complement

The negation complement of a fuzzy set A in X is a fuzzy set denoted $\overline{A}$ with the membership function defined as

$$\forall x \in X,\ \mu_{\overline{A}}(x) = 1 - \mu_A(x)$$

where $\mu_A(x)$ represents the membership function of the fuzzy set A. Assume A to be a fuzzy set described by $\mu_A(x) = gbellmf(x; a, b, c)$. One can prove that the classical fuzzy complement of A is described by the membership function defined as $\mu_{\overline{A}}(x) = gbellmf(x; a, -b, c)$.

2.4.1. Fuzzy Intersection

The intersection of two fuzzy sets A and B in X is the greatest fuzzy set contained both in the fuzzy sets A and B. The intersection of two fuzzy sets is specified by a membership function related to those of A and B as

$$\forall x \in X,\ \mu_{A\cap B}(x) = \min\left\{\mu_A(x), \mu_B(x)\right\}.$$

The law of contradiction well known in classical set theory is no longer true and valid in fuzzy set theory due to the lack of precise boundaries. That is, for fuzzy set A in X, $A \cap \overline{A} \neq \phi$. For instance, assuming X to be a universe of discourse made up of discrete

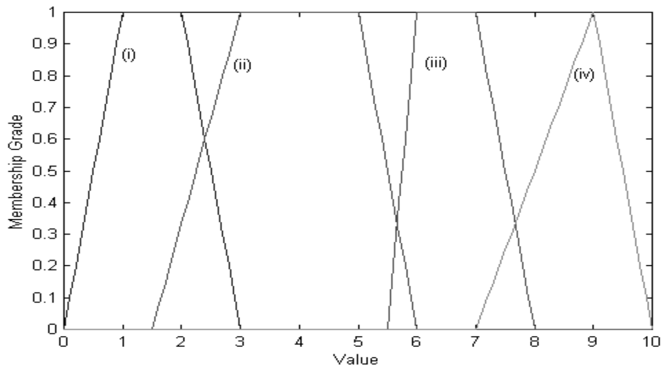


Figure 2.9.: Plot showing the shapes of different trapezoidal membership functions. One has (i): $a = 0$, $b = 1$, $c = 2$, $d = 3$; (ii): $a = 1.5$, $b = 3$, $c = 5$, $d = 6$; (iii): $a = 5.5$, $b = 6$, $c = 7$, $d = 8$; (iv): $a = 7$, $b = 9$, $c = 9$, $d = 10$.

numbers and A a fuzzy set in X defined as

$$A = \{(-2, 0.1), (1, 0.3), (2, 0.4), (5, 0.5), (8, 0.9)\}.$$

In other words, $\mu_A(-2) = 0.1$, $\mu_A(1) = 0.3$, $\mu_A(2) = 0.4$, $\mu_A(5) = 0.5$, $\mu_A(8) = 0.9$. The classical fuzzy complement set $\overline{A}$ is defined as

$$\overline{A} = \{(-2, 0.9), (1, 0.7), (2, 0.6), (5, 0.5), (8, 0.1)\}$$

since $\mu_{\overline{A}}(x) = 1 - \mu_A(x)$. The resulting classical fuzzy intersection set is defined as

$$A \cap \overline{A} = \{(-2, 0.1), (1, 0.3), (2, 0.4), (5, 0.5), (8, 0.1)\}.$$

2.4.2. Fuzzy Union

The union of two fuzzy sets A and B in X is the smallest fuzzy set containing both the fuzzy sets A and B with a membership function defined as

$$\forall x \in X,\ \ \mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}.$$

Considering the same hypotheses as in the previous example, on can show that the resulting classical fuzzy union set is given as

$$A \cup \overline{A} = \{(-2, 0.9), (1, 0.7), (2, 0.6), (5, 0.5), (8, 0.1)\}.$$

The law of the excluded middle well known in classical set theory[6] is also no longer true and valid due to the lack of precise boundaries. That is, for a fuzzy set A in X, $A \cup \overline{A} \neq X$.

2.5. Generalization of Fuzzy Set Operators

Although classical fuzzy set operators such as complement, intersection and union, possess more rigourous axiomatic properties, they are not the only ways to define reasonable and consistent operations on fuzzy sets. In this section, we will examine other definitions of the fuzzy complement, intersection and union that generalize the classical fuzzy operators reviewed in the previous section.

2.5.1. Fuzzy Complement

A fuzzy complement operator is a continuous function $N : [0,1] \rightarrow [0,1]$ which meets the boundary, monotonicity and involution axiomatic requirements.

Boundary: $N(0) = 1$ and $N(1) = 0$. This requirement guarantees the generalization of fuzzy complement to crisp sets.

Monotonicity: $\forall a,\ b \in [0,1],\ a \leq b$ implies $N(a) \geq N(b)$. This requirement specifies that N is a decreasing function. Therefore an increase in the membership grade of a fuzzy set must result in a decrease in the membership grade of its complement.

Involution: $N(N(a)) = a$. This requirement is optional but guarantees that the double complement of a fuzzy set is still the set itself.

All functions satisfying these requirements form the general class of fuzzy complements. The classical fuzzy complement defined in the previous section is a special case of this class of fuzzy complements. Other very well known classes of fuzzy complements are Sugeno's fuzzy complement and Yager's fuzzy complement defined respectively as

$$N_s(a) = \frac{1-a}{1+sa}$$

and

$$N_w(a) = (1-a^w)^{\frac{1}{w}}$$

where the parameter s is greater than -1 and w a positive parameter.

[6]In classical set theory, given X a universe of discourse, $\forall A \subset X,\ A \cup \overline{A} = X$.

2.5.2. Fuzzy Intersection

Before defining the fuzzy intersection operator, let us first review the so called triangular T-norm operator.

A T-norm (or triangular norm) operator is a two-place function satisfying the following axiomatic requirements:

Boundary: $T(0,0) = 0$ and $T(a,1) = T(1,a) = a$. This requirement imposes the correct generalization to crisp sets.

Monotonicity: $T(a,b) \leq T(c,d)$ if $a \leq c$ and $b \leq c$. This requirement implies that a decrease in the membership values in A or B must produce a decrease in the membership value in $A \cap B$.

Commutativity: $T(a,b) = T(b,a)$. This requirement indicates that the operator is indifferent to the order of the fuzzy sets to be combined.

Associativity: $T(a, T(b,c)) = T(T(a,b), c)$. This last requirement allows us to take the intersection of any number of sets in any order of pairwise groupings.

The intersection of the two fuzzy sets A and B is specified in general by a T-norm function from $[0,1] \times [0,1] \longrightarrow [0,1]$ that aggregates two membership grades as follows:

$$\mu_{A \cap B}(x) = T\left(\mu_A(x), \mu_B(x)\right).$$

Based on the T-norm definition, many T-norm operators are proposed in literature. Among them:

- **Minimum:** $T_{\min}(a,b) = \min(a,b)$. The classical definition of a fuzzy intersection reviewed above is based on this type of T-norm. For two fuzzy sets A and B, one has $\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$.

- **Algebraic product:** $T_{ap}(a,b) = ab$. The algebraic product of two fuzzy sets A and B in X is the fuzzy set $A \cdot B$ with the membership function $\mu_{A \cdot B}(x) = \mu_A(x) \times \mu_B(x)$.

- **Bounded product:** $T_{bp}(a,b) = max(0, a+b-1)$. The bounded product of two fuzzy sets A and B in X is the fuzzy set $A \odot B$ with the membership function defined as $\mu_{A \odot B}(x) = max\,\{0, \mu_A(x) + \mu_B(x) - 1\}$.

- **Drastic product:** $T_{dp}(a,b) = \begin{cases} a & \text{if} \quad b = 1 \\ b & \text{if} \quad a = 1 \\ 1 & \text{if} \quad a, b < 1. \end{cases}$

  The drastic product of two fuzzy sets A and B is the fuzzy set $A\hat{\bullet}B$ with the membership function defined as

  $\mu_{A\hat{\bullet}B} = \begin{cases} \mu_A(x) & \text{if} \quad \mu_B(x) = 1 \\ \mu_B(x) & \text{if} \quad \mu_A(x) = 1 \\ 1 & \text{if} \quad \mu_A()x < 1 \text{ and } \mu_B(x) < 1. \end{cases}$

2.5.3. Fuzzy Union

As in the previous subsection, before defining the fuzzy union, let us first review the triangular T-conorm operator.

A triangular T-conorm or S-norm is a two place function satisfying the following axiomatic requirements:

Boundary: $S(1,1) = 1$ and $\forall\, a \in [0,1]$, $S(a,0) = S(0,a) = a$. This requirement guaranties the correct generalization to crisp sets.

Monotonicity: $\forall\, a,\ b,\ c,\ d\ \in [0,1]$, $S(a,b) \leq S(c,d)$ if $a \leq c$ and $b \leq d$. This requirement ensures that a decreasing in the membership values in A or B cannot produce an increasing in the membership value in $A \cup B$.

Commutativity: $\forall a,\ b,\ \in [0,1]$, $S(a,b) = S(b,a)$.

Associativity: $\forall a,\ b,\ c, \in [0,1]$, $S(a, S(b,c)) = S(S(a,b),c)$.

Many T-conorm operators exist. Among them, the following:

- **Maximum :** $S(a,b) = \max(a,b)$. This is the classical definition of a fuzzy union operator. For two fuzzy sets A and B, one has $\mu_{A\cup B}(x) = max\,\{\mu_A(x), \mu_B(x)\}$.

- **Algebraic sum:** $\forall\, a,\ b\ \in\ [0,1]$, $S(a,b) = a + b - ab$. The algebraic sum of two fuzzy sets A and B in X is the fuzzy set $A + B$ with the membership function defined as $\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$.

- **Bounded sum:** $\forall\, a,\ b\ \in\ [0,1]$, $S(a,b) = 1 \wedge (a + b)$. The bounded sum of two fuzzy sets A and B is the fuzzy set $A \oplus B$ with the membership function defined as $\mu_{A\oplus B}(x) = \min\{1, \mu_A(x) + \mu_B(x)\}$.

- **Drastic sum:**

$$\forall\, a,\, b \in [0,1],\ S(a,b) = \begin{cases} a & \text{if} \quad b = 0 \\ b & \text{if} \quad a = 0 \\ 1 & \text{if} \quad a, b > 0. \end{cases}$$

The drastic sum of two fuzzy sets A and B is the fuzzy set $A\dot{\vee}B$ with the membership function defined as

$$\mu_{A\dot{\vee}B} = \begin{cases} \mu_A(x) & \text{if} \quad \mu_B(x) = 0 \\ \mu_B(x) & \text{if} \quad \mu_A(x) = 0 \\ 1 & \text{if} \quad \mu_A(x) > 0 \text{ and } \mu_B(x) > 0. \end{cases}$$

From these definitions of both T-norm and S-norm, we have the generalized DeMorgan Law specifying that T-norm and S-norm are dual. In other words, one has i.e. $T(a,b) = N\left(S(N(a), N(b))\right)$ and $S(a,b) = N\left(T(N(a), N(b))\right)$ where N denotes the fuzzy complement operator.

2.6. Linguistic Variables

The use of words or sentences rather than numbers is due to the fact that linguistic characterizations are in general less specific than numerical ones. Before introducing the formal definition of a linguistic variable, let us first define a fuzzy variable. A fuzzy variable is characterized by a triple $(V, X, R(V))$ in which V is the name of the variable, X is a universe of discourse, and $R(V)$ is a fuzzy subset of X which represents a fuzzy restriction imposed by V. As an example, $V =$ *"Cold"* with $X = \{-10°C,\ -5°C,\ 0°C,\ 20°C,\ 30°C\}$. One may have

$$R(V) = \{(-10°C, 0.8),\ (-5°C, 0.7),\ (0°C, 0.5),\ (20°C, 0.03),\ (30°C, 0)\}$$

that is a fuzzy restriction on *"Cold"*. Linguistic variable is a variable of higher order than fuzzy variable. Linguistic variables are characterized by a quintuple $(x, T(x), X, G, M)$ in which x is the name of the variable, $T(x)$ denotes the term set of x (that is the set of its linguistic values or linguistic terms), X is the universe of discourse, G is the syntactic rule. The syntactic rule refers to the way the linguistic values are generated in the term set $T(x)$. M is the semantic rule. The semantic rule defines the membership function of each linguistic value of the term set $T(x)$.

Sometimes the term set consists of several primary terms modified by the negation NOT, AND, OR, or the edges[7] and then linked by connectives such as, *and, or, either,*

[7]The linguistic hedge h is an operator for modifying the meaning of a fuzzy set A to create a new fuzzy set $h(A)$. For instance, in the fuzzy set "Too Cold", the term "Too" is a linguistic hedge.

neither. In short, a lingustic variable translates real values into linguistic values.

Example Let temperature θ be interpreted as a linguistic variable. It can be decomposed into the following terms:

$$T(\theta) = \{Too\ Cold,\ Cold,\ Okay,\ Hot,\ Too\ Hot\}$$

where each term in $T(\theta)$ is characterized by a fuzzy set in the universe of discourse $X = [-10°C, 40°C]$. We might interpret *"Too Cold"* as a temperature below $-5°C$, "Cold" as a temperature more close to $0°C$, *"Okay"* as a temperature close to $20°C$, *"Hot"* as a temperature close to $30°C$ and *"Too Hot"* as a temperature above $35°C$. One can represent the fuzzy set A_1 *"temperature more close to* $0°C$*"* with $\mu_{A_1}(x) = \frac{1}{1+10x^2}$ and the fuzzy set A_2 *"temperature close to* $30°C$*"* with $\mu_{A_2}(x) = \frac{1}{1+(x-30)^2}$. Gaussian membership functions can be used to represent the fuzzy concept *"Too Cold"* and *"Too Hot"*.

2.6.1. Fuzzy If-Then Rules

Through the use of linguistic variables and membership functions, a fuzzy if-then rule can easily capture the spirit of a *rule of thumb* used by humans. A fuzzy if-then rule assumes the form: **If** x **is** A **then** y **is** B where A and B are linguistic values defined by fuzzy sets on universes of discourse X and Y respectively. Usually "x *is* A" is called the antecedent or premise while "y is B" is called the consequence. Due to their concise form, fuzzy if-then rules are often employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision [43].

2.6.2. Fuzzy Reasoning

Human reasoning is fuzzy and vague in nature. Before involving on the most important mode of reasoning in fuzzy logic theory, let us review the meaning of fuzzy predicate modifiers, fuzzy quantifiers, and fuzzy qualifiers.

To provide the capability of approximate reasoning, fuzzy allows the use of fuzzy predicates, fuzzy predicate modifiers, fuzzy quantifiers, and fuzzy qualifiers in the propositions. For instance, *Tall, Ill, Young, Soon, Friend of, Hot, etc.* are fuzzy predicates.

Classical logic only offers as predicate modifier the negation NOT. In fuzzy logic, in addition to the negation modifier, there are a variety of predicate modifiers that act as hedges. For instance, *Very, Rather, More or Less, Slightly, a Little, Extremely, etc.*

Fuzzy logic allows the use of fuzzy quantifiers exemplified by *Most, Many, Several, Few, Much of, Frequently, Occasionally, About ten, etc.* From the example, *"Most Africans are kind"*, the quantifier *Most* provides an imprecise characterization of the cardinality of the fuzzy set *kind Africans.*

Fuzzy logic allows the use of the fuzzy qualifiers exemplified by *Not Very, Quiet, Almost.* For example, *(Today is fine) is* **not very** *true.* The qualified proposition is *Today is fine* whereas the qualifying fuzzy truth value is *Not very true.*

Fuzzy reasoning is an inference procedure that derives conclusions from a set of fuzzy if-then rules and known facts. There are four principal modes of fuzzy reasoning namely: *categorical reasoning, qualitative reasoning, syllogistic reasoning and dispositional reasoning.* The most important is the qualitative mode of reasoning [44] since it is the popular and easy formalism for representing knowledge. In this dissertation, we present only the qualitative reasoning. The reader should refer to [39, 53] for more details about other modes of reasoning in fuzzy logic.

Qualitative Reasoning

Qualitative reasoning is the most important mode of reasoning since it is based on *if-then rules* that are the popular formalism for representing knowledge. Qualitative reasoning plays a key role in many applications of fuzzy logic control. Qualitative reasoning refers to a mode in which the input-output relation of a system is expressed as a collection of fuzzy if-then rules in which the preconditions and conclusions involve fuzzy or linguistic variables. In the following section, we will describe the most basic rules of inference used in qualitative reasoning.

2.6.3. Basic Rules of Inference

The computation of fuzzy rules is called fuzzy rules inference. The inference is a calculus consisting of two main steps: *aggregation* and *conclusion.* Aggregation determines the degre to which the IF part of the rule is fulfilled, whereas the conclusion is the THEN part. Let us review some types of rules used in fuzzy logic.

Generalized Modus Ponens

The modus ponens is a basic rule of inference according to which we can infer the truth of a proposition q from the truth of a proposition p and the fuzzy implication $p \Longrightarrow q$. The modus ponens can be interpreted in the following form: *If p is true and if the proposition "If p is true then q is true" is also true, then the proposition q is true.* In other words, if we denote *"x is A"* a fact, *"if x is A then y is B"* a rule, the consequence or conclusion is *"y is B"*. For instance assume p is *"the tomato is red"* (fact), and q *"if the tomato is red then the tomato is ripe"* (rule). The conclusion should be *"the tomato is ripe"* [40]. The generalization of modus ponens is straightforward in using the predicate modifers. The idea is based on the fact that most human reasoning employs modus ponens in an approximate manner. For instance, assume we have the implication *if the tomato is red then it is ripe* and we know that *the tomato is more or less red.* Then we may infer that *the tomato is more or less ripe.* More generally, denoting "*x is* A'" the fact, if "*x is A then y is B*" the rule, the conclusion should be "*y is* B'" where A' is close to A and B$'$ is close to B.

Single Rule with Single or Multiple Antecedents

A fuzzy if-then rule with one antecedent is written as "*if x is A then z is C*". The corresponding problem for modus ponens could be expressed in the following way: assuming "*x is* A'" is the fact, "*if x is A then z is C*" the rule, the consequence is "*x is* C'".

A fuzzy if-then rule with two antecedents is usually written as "*if x is A and y is B then z is C*". The corresponding problem for generalized modus ponens could be expressed in the following way. Assume "*x is* A'" and "*y is* B'" is the fact, "*if x is A and y is B then z is C*" the rule, the consequence is "*z is* C'".

Multiples Rules with Multiples Antecedents

The interpretation of multiple rules is usually taken as the union of the fuzzy relations corresponding to the fuzzy rules. Therefore, if "*x is* A' and *y is* B'" is the fact, "*if x is* A_1 and *y is* B_1 then *z is* C_1" the first rule, "*if x is* A_2 and *y is* B_2 then *z is* C_2" is the second rule, the conclusion is "*z is* $C' = C_1' \cup C_2'$" where C_1' and C_2' are the inference fuzzy sets for the first and the second rules respectively.

2.7. Measures on Fuzzy Sets

In Fuzzy set theory, two categoricals of incertainties can be recognized: vagueness and ambiguity. Vagueness is associated with the difficulty of making a sharp or precise boundary in grouping objects of interest, while ambiguity is associated to the difficulty in making a choice between two or more alternatives. Measures of uncertainty related to vagueness are referred to as measure of fuzziness. We will review here only the measure of uncertainty related to the ambiguity namely, fuzzy measure and fuzzy possibility.

2.7.1. Fuzzy Measure

The concept of fuzzy measure was introduced by Sugeno to exclude the additivity requirement of standard measures[8].

Before defining the fuzzy measure, let us first review a Borel field. Consider X a discrete or continuous universe of discourse. A Borel field $\mathcal{B} \subset \mathcal{P}(X)$ is a family of subsets of X that satisfies the following conditions:

1. $\phi \in \mathcal{B}$ and $X \in \mathcal{B}$.
2. $\forall A \in \mathcal{B}$, $\overline{A} \in \mathcal{B}$ where $\overline{A}$ denotes the complement of A in X.
3. $\forall A, B \in \mathcal{B}$, $A \cup B \in \mathcal{B}$.

A fuzzy measure on $\mathcal{B}$ is a real-valued function $g : \mathcal{B} \longrightarrow [0,1]$ that satisfies the following axiomatic requirements:

Boundary conditions: $g(\phi) = 0$ and $g(X) = 1$. This axiom states that we always know that the element in question definitively does not belong to the empty set and definitively does belong to the universal set.

Monotonicity: For all crisp sets A, B such that $A \subseteq B$, $g(A) \leq g(B)$. In other words, g is a nondecreasing function.

Continuity: For every monotonic sequence $(A_i \in \mathcal{B})$ with $i \in \mathbb{N}$ of subsets of X, then

$$\lim_{i\to\infty} g(A_i) = g(\lim_{i\to\infty} A_i).$$

This requirement can be omitted when X is finite.

[8]A standard measure h is additive if for two disjoint sets A and B, one has $h(A \cup B) = h(A) + h(B)$

One can easily prove that, for every fuzzy measure g, the following property holds:

$$\forall A,\, B \in \mathcal{B},\; g(A \cap B) \leq \min\{g(A), g(B)\} \leq \max\{g(A), g(B)\} \leq g(A \cup B).$$

2.7.2. Possibility Measure

A fuzzy measure π on $\mathcal{B}$ is called a possibility measure when the following conditions are satisfied:

1. $\forall A,\, B \in \mathcal{B}$, $\pi(A \cup B) = \max\{\pi(A), \pi(B)\}$.

2. $\pi(\phi) = 0$ and $\pi(X) = 1$. These are called the boundary conditions.

One can easily show that:

$$\forall A \in \mathcal{B},\ \max\{\pi(A), \pi(\overline{A})\} = 1. \text{ Therefore } \pi(A) + \pi(\overline{A}) \geq 1.$$

The latter inequality can be formulated saying that the probability of an event completely determines the probability of the contrary event while the possibility of an event is weakly linked with the contrary event.

2.8. Sugeno Fuzzy Models

The Sugeno fuzzy model tries to develop a systematic approach to generating fuzzy rules from a given *(input,output)* data set. A typical fuzzy rule in a Sugeno fuzzy model has the form

$$\textit{if } x \textit{ is } A \textit{ and } y \textit{ is } B \textit{ then } z = f(x, y)$$

where A and B are fuzzy sets in the antecedent, while $z = f(x, y)$ is a crisp function in the consequent. Usually $f(x, y)$ is a polynomial in the input variables x and y but it can be any function as long as it can appropriately describe the output of the model within the fuzzy region specified by the antecedent of the rule. When the function $f(x, y)$ is a first-order Sugeno fuzzy polynomial, the resulting model is a first-order Sugeno fuzzy model [40, 54].

Jang [41] showed that under certain minor constraints, a zero-order Sugeno fuzzy model is functionally equivalent to a radial basis function network (see next chapter).

2.9. Fuzzy Logic

Fuzzy logic is an extension of set-theoretic bivalence logic in which the truth values are terms of the linguistic variable "truth". Its goal is to provide foundations for approximate reasoning with an imprecise proposition using fuzzy set theory. Unlike the situation in two-valued logic, the truth values of propositions in fuzzy logic are allowed to range over the fuzzy subsets of unit interval $[0, 1]$ or over a point in the interval.

The truth value of a proposition "x *is* A" or simply the truth value of A which is denoted by $v(A)$ is defined to be a point in $[0, 1]$ (called the linguistic truth value) or a fuzzy set in $[0, 1]$.

Let us assume $v(A)$ and $v(B)$ to be two numerical truth values of propositions A and B respectively. Below are the four usual fuzzy logic operations:

- $v(\overline{A}) = 1.0 - v(A)$
- $v(A \text{ and } B) = v(A) \wedge v(B) = \min\{v(A), v(B)\}$
- $v(A \text{ or } B) = v(A) \vee v(B) = \max\{v(A), v(B)\}$
- $v(A \Longrightarrow B) = v(A) \Longrightarrow v(B) = \max\{1 - v(A), \min\{v(A), v(B)\}\}$.

From these definitions, it appears that fuzzy logic operations are an extension of conventional Boolean logic. This is known as the extension principle which states that the classical results of Boolean logic are recovered from fuzzy logic operations when all fuzzy membership grades are restricted to the set $\{0, 1\}$ [89]. In this way, fuzzy subsets and fuzzy logic are a generalization of classical set theory and logic.

2.10. Fuzzy Relations

Fuzzy relations as other operations have their background in crisp set theory. In this section, we reviewed crisp relations and how they can be extended for defining fuzzy relations. Through an example, we showed that crip relation cannot always model human judgment.

2.10.1. Crisp Relations

Assume A and B to be two sets. A crisp or binary relation $\mathcal{R}$ on $A \times B$ means that, for a pair of elements $(x\,,y) \in A \times B$, either $x\,\mathcal{R}\,y$ or $x\,\overline{\mathcal{R}}\,y$ holds. Denoting $x\,\mathcal{R}\,y$ by $\mathcal{R}(x,y)$, crisp relations are of the form:

$$x\,\mathcal{R}\,y \iff \mathcal{R}(x,y) = 1$$

and

$$x\,\overline{\mathcal{R}}\,y \iff \mathcal{R}(x,y) = 0.$$

In this sense, a crisp relation on $A \times B$ can be viewed as a mapping from $A \times B$ to $\{0,1\}$. Even though crisp relations have success in many applications, they cannot model a concept with imprecise boundaries.

2.10.2. Weakness of Crisp Relation

Le us consider the relation *much greater than* on the real line $\mathbb{R}$ defined as

$$x\;\Re\;y \iff x >> y.$$

One can write $\mathcal{R}(10,5) = 0$. The use of a crisp relation leads to $\mathcal{R}(100,10) = 1$. This is not true in practice. Therefore, the use of a membership grade is a possible way for such a relation to model human reasoning and thoughts.

2.10.3. Fuzzy Relations

The fuzzy concept *much greater than* can be viewed as a fuzzy relation on $\mathbb{R} \times \mathbb{R}$ defined as:

$$\mathcal{R}(x,y) = \begin{cases} f(x-y) & \text{if } x \geq y \\ 0 & \text{if } x < y \end{cases}$$

where f represents a monotone non decreasing function defined on nonnegative real numbers such that $f(0) = 0$ and $f(+\infty) = 1$. In [89], an example of f is given by

$$f(x) = \left[1 + \left(\frac{\gamma}{x}\right)^2\right]^{-1}$$

where γ is a positive parameter. The fuzzy relation *much less than* is represented by $\mathcal{R}^{-1}$ where $\mathcal{R}^{-1}(x,y) = \mathcal{R}(y,x)$.

2.11. Fuzzy Inference System

As pointed out by Jang [40], a Fuzzy Inference System (FIS) is based on the past known behaviour of a target system. It employs fuzzy if-then rules to model the qualitative aspects of human knowledge and reasoning without employing precise quantitative analyses. The major benefits of fuzzy techniques are the possibility to model systems and to give a system description by using linguistic rules [50]. The fuzzy system is then expected to be able to reproduce the behaviour of the target system.

A Fuzzy Inference System is composed of five functional blocks: a rule base containing a number of fuzzy if-then rules, a database which defines the membership function of the fuzzy sets used in the fuzzy rules, a decision-making unit which performs the inference operations on the rules, a fuzzification interface which transforms the crisp inputs into degrees of match with linguistic values, and a defuzzification interface which transforms the fuzzy results of the inference into a crisp output.

2.12. Fuzzy Applications

The applications of fuzzy theory are multi-disciplinary in nature and include automatic control, consumer electronics, signal processing, time-series prediction, information retrieval, database management, computer vision, data classification, decision-making, and so on [39].

2.12.1. Expert Systems

An expert system is a computer program that requires significant human expertise by using explicitly represented domain and computational decision procedures [44]. There are three reasons for the use of fuzzy set theory in expert systems:

1. The interfaces of the expert system on the expert's side as well as on the user's side are with human beings. Therefore, communication in a natural way seems to be the most appropriate and natural means, generally in the language of the expert or user.

2. The knowledge base of an expert system is a repository of human knowledge, and since most human knowledge is imprecise in nature, it is usually the case that the

knowledge base of an expert system is a collection of rules and fact that, for the most part are neither totally certain nor totally consistent. The storage of this vague and uncertain portion of knowledge by using fuzzy sets seems much more appropriate than the use of crisp concepts and symbolism.

3. Uncertainty of information in knowledge induces uncertainty in the conclusions, and therefore the inference engine has to be equipped with the computational capabilities to analyze the transmission of uncertainty from the premises to the conclusions and to associate the conclusion with some measures of uncertainty that are understandable and properly interpretable by the user. Human thinking, when modeled in expert systems, might also increase efficiency that is decrease answering time and so on.

2.12.2. Fuzzy Control

Fuzzy control is the branch of fuzzy set theory with the most applications and their number is steadily growing. The application boom was started when Japanese manufacturers applied fuzzy logic to processes ranging from home appliances to industrial control. Notable applications of the fuzzy logic controller, include a steam engine, a warm water process, aircraft flight control, automobile speed control, adaptive control etc. [54].

Summary

In this chapter, we have reviewed some basic notions, and notations about fuzzy set and fuzzy logic theory. Such a theory has shown important applications in practice. Fuzzy inference systems are the most important modeling tools based on fuzzy set theory. Optimization and adaptive control expand the applications of fuzzy inference systems to fields such as adaptive control, adaptive signal processing, nonlinear regression, and pattern recognition.

3. Artificial Neural Networks

We do not know how the brain represents high-level information, so cannot mimic that, but we do know

that it uses many slows units that are highly interconnected.

– R. Beale and T. Jackson.

Introduction

Artificial Neural Networks (ANNs) are a promising new generation of information processing systems, with the capability of modeling complex nonlinear processes to arbitrary degrees of accuracy. They are inspired by modeling networks of biological neurons in the brain. The individual processing elements in ANNs are known as artificial neurons, nodes or simply neurons. An ANN is composed of a large set of neurons, highly interconnected by a set of weighted links, with the potential of providing massive parallelism operations. As an artificial model of the human brain, the main properties of ANNs are the ability to learn, recall, and generalize from training patterns or training data [54]. The models of ANNs are classified in different categories according to functional or operation criteria such as supervised or unsupervised learning rules [43]. We reviewed in this chapter, some neural network architectures namely MultiLayer Perceptron, Time Delay Neural Networks, Radial Basis Function Neural Networks, Recurrent Neural Networks, and focused on Adaptive Networks as a superset of all kind of supervised learning networks.

3.1. Fundamental Concepts

The basic function of a biological neuron is to add up its inputs, and to produce an output if this sum is greater than some value, known as the *threshold* value.

The aim of a model is to produce a simplified version of a system which retains the same general behaviour, so that the system can be more easily understood. The model of a neuron must capture important features that are:

- The output from a neuron is either *"on"* or *"off"*.
- The output depends only upon the inputs and upon the activation function of the neuron. A certain number of inputs must be *"on"* at any time in order to make a neuron fire.

Since the inputs are passed through the model neuron to produce the output, the system is known as a *feedforward* one.

Assume there are n inputs x_i, and therefore n associated weights. The total input to the neuron can be written as

$$\begin{aligned}\text{net input} &= w_1x_1 + w_2x_2 + \ldots + w_nx_n \\ &= \sum_{i=1}^{n} w_ix_i.\end{aligned}$$

This sum has to be compared to the threshold value. If the net input is greater than the threshold value θ, the output is 1, otherwise the output is 0. This can be seen graphically in Figure 3.1 where the $x-$axis represents the input, and the $y-$axis the output. The threshold function is also known as the Heaviside function[1]. This model is known as Perceptron. Figure 3.2 shows the plot of the Heaviside function where the thresholding value is reached at θ. Let us denote y as the output of the neuron and f the Heaviside function. We can write

$$y = f\left(\sum_{i=1}^{n} w_ix_i - \theta\right).$$

[1]The Heaviside function is defined as

$$f(x) = \begin{cases} 1 & \text{if} \quad x > 0 \\ 0 & \text{if} \quad x \leq 0. \end{cases}$$

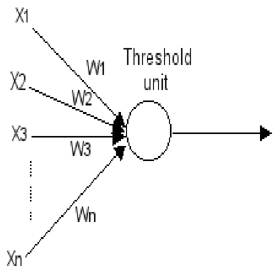


Figure 3.1.: Model of an artificial neuron.

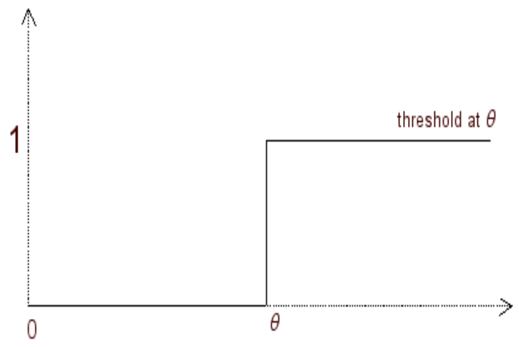


Figure 3.2.: The thresholding or Heaviside function.

It has been shown that a Perceptron can solve only linear separable problems [4, 7]. Another model known as MultiLayer Perceptron is introduced in literature for solving more complex discrimination or approximation tasks [28].

3.2. MultiLayer Perceptron

A MultiLayer Perceptron (MLP) consists of several layers of neurons with full interconnections between neurons in adjacent layers. Input data are presented to the network at the input layer, which contains no processing nodes. It serves only as a data source for the following hidden layers. Finally the network output is computed by neurons in the output layer. The nodes in the hidden layer compute internal representations of the data.

MLP's are useful for supervised pattern recogniton where the task is to learn a mapping $\mathbf{x}$ and outputs $\mathbf{t}$ given a set of training samples

$$\mathcal{T} = \{(\mathbf{x}_1, \mathbf{t}_1), \cdots, (\mathbf{x}_n, \mathbf{t}_n)\}.$$

The learning rule provides the method for adjusting the weights in the network. During the training phase, the weights are usually updated by an iterative learning algorithm called error propagation. After this procedure, the MLP can be used to map unseen patterns to the classes learned during the training. In order to successfully learn the net should minimize or maximize a cost function called the error function.

Let us denote E_p as the error function for the pattern p, t_{pj} the target output for pattern p on node j, o_{pj} the actual output at the node j and w_{ij} the weights from node i to node j. The error function is defined as

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2.$$

The activation of each unit j, for pattern p, can be written as

$$net_{pj} = \sum_i w_{ij} o_{pi}.$$

The activation function of each unit j is the sigmoid function[2], although any continuously differentiable monotonic function can be used [12, 15]. The output is given by

$$o_{pj} = f(net_{pj}).$$

Using the chain rule, we can write

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ij}}.$$

Let us define the change in error as a function of the change in the net inputs to a unit as

$$-\frac{\partial E_p}{\partial net_{pj}} = \delta_{pj}.$$

[2]A sigmoid function on $\mathbb{R}$ is defined as

$$f(x) = \frac{1}{1 + e^{-kx}}$$

where k is a positive constant that controls the spread of the function.

The error function becomes,

$$\begin{aligned}
\frac{\partial E_p}{\partial w_{ij}} &= \frac{\partial E_p}{\partial net_{pj}} \frac{\partial}{\partial w_{ij}} \sum_k w_{kj} o_{pk} \\
&= \frac{\partial E_p}{\partial net_{pj}} \sum_k \frac{\partial w_{jk}}{\partial w_{ij}} o_{pk} \\
&= \frac{\partial E_p}{\partial net_{pj}} o_{pi} \\
&= -\delta_{pj} o_{pi}.
\end{aligned}$$

It can be shown that [4] the change in the error function is of the following form:

$$\begin{cases} \delta_{pj} = f'(net_{pj}) \sum_k \delta_{pk} w_{jk} & \text{for hidden units} \\ \delta_{pj} = f'(net_{pj})(t_{pj} - o_{pj}) & \text{for output units.} \end{cases}$$

The error value calculated in the output units is then passed back through the network to the earlier units to allow them to alter their connection weights. This algorithm is called backpropagation algorithm.

It has been reported that MLP's with at least one hidden layer can approximate any continuous function to any desired degree of accuracy, if there are enough hidden neurons available [5, 73]. This property is called *universal function approximation*. Thus, MLP networks with one hidden layer are sufficient, although additional hidden layers may improve performance over single hidden layer networks with an equal number of neurons through increased model complexity.

3.3. Time Delay Neural Networks

MultiLayer Perceptrons do not model time. Problems as speech recognition cannot be faced by MLP since it is necessary to take into account the temporal nature of speech. For this reason, Waibel in [80], introduced Time Delay Neural Networks (TDNNs) in order to perform a time sequence prediction in turning the temporal sequence into a spatial input pattern [54]. Time-Delay Neural Networks are separated into time frames called *windows*. A typical application of TDNNs is phoneme classification [24, 25, 80]. This net is constituted of one input layer, two hidden layers and one output layer. The size of a hidden layer is a fundamental question often raised in the application of multilayer feed-forward networks to real world problems [26]. The exact analysis of this issue is rather difficult because of the complexity of the network mapping and the non deterministic nature of the training procedures [54]. TDNNs are very suitable for phoneme recognition because it is believed that features learnt by such nets are invariant under translation in time.

3.4. Radial Basis Function Networks

The Radial Basis Function Network is a single hidden layer feedforward network with linear output transfer functions and nonlinear transfer functions ϕ_j on the hidden layer nodes [64]. The activation of a hidden unit is determined by the distance between the input vector and a prototype vector. The primary adjustable parameters are the final layer weights $\{w_{jk}\}$ connecting the $j-$th hidden node to the $k-$th output node. There are also weights $\{\mu_{ij}\}$ connecting the $i-$th input node with the $j-$th hidden node. Many basis functions exist in literature [7]. The most used is the Gaussian basis function expressed as

$$\phi_j(x) = e^{-\frac{||x-\mu_j||^2}{2\sigma_j{}^2}}$$

where μ_j is the vector determining the centre of ϕ_j and σ_j controls its width. The $k-$th component of the output vector y corresponding to the input pattern x is expressed as

$$y_k(x) = \sum_{j=1}^{M} w_{kj}\phi_j(x)$$

where M denotes the number of hidden nodes.

The main difference from multilayer perceptron is that the output of the hidden node j (h_j) is given as a radial function of the distance between each pattern vector and each hidden node weight vector, $h_j = \phi_j(||x_p - \mu_j||)$, rather than a scalar product $h_j = \phi_j(x \cdot \mu_j)$ for multilayer perceptron. Radial basis function networks are intensively used for function approximation [55].

3.5. Recurrent Neural Networks

In order to model dynamical functions of the brain or to design a machine that performs as well as the brain does, it is essential to utilize a system that is able to store internal states and implement complex dynamics. This is the motivation under which recurrent neural networks have been widely studied [46]. Recurrent neural networks are one in those self-loops and backward connections between nodes are allowed. Recurrent networks with symmetric weight connections always converge to state [54] whereas asymmetric recurrent networks have much complex dynamical. The diversity of dynamical behavior suggests that such networks may be well suited to the problem of time series prediction, and signal recognition [54, 60]. There are many types of recurrent neural networks according to the learning procedure and architecture. When a backpropagation algorithm is used for training a recurrent neural network, the resulting network is called recurrent back propagation network [46, 54].

3.6. Adaptive Networks

An adaptive network is a unifying model of all kinds of supervised, feedforward networks. Its network structure is composed of a set of nodes connected through directional links. Each node performs a static node function on its incoming signals and on a set of parameters pertaining to this node. Each node generates a single node output and each link specifies the direction of signal flow from one node to another. All or part of the nodes are adaptive and the node function may vary from one node to another. The learning rule specifies how parameters are updated to minimize an error measure between the actual and the desired output. Each change in the parameters modifies the node function and the overall network's behaviour. The links in an adaptive network have no weights associated to them [39]. To reflect the different adaptive capabilities a node with parameters is called *adaptive* and is represented by a square, while a node without parameter is called *fixed* and is represented by a circle.

The basic learning rule of the adaptive network is the steepest descent method, in which the gradient descent vector is derived by successive invocations of the chain rule [43]. The learning procedure more commonly used is the back-propagation algorithm which is applied to find the gradient vector in the network structure. After the gradient is found, the parameters are updated by means of an available number of derivative-based optimization and regression techniques [44]. Since in practice, this simple method is too slow to converge and likely to become trapped into local minima, the steepest descent method can be combined with the least-squares estimator for fast identification of parameters. This learning technique is used in ANFIS [39].

3.7. Network Learning Algorithms

The procedure used to adjust the parameters in order to improve the network performance using the training data set is referred as *learning algorithm.* The learning algorithm refers to the way the weights are changed in order to achieve a desired mapping between input and output [43]. There are many learning algorithms depending upon the context, among them, back-propagation, linear least-squares estimator, and hybrid learning.

3.7.1. Back-Propagation Algorithm

The back-propagation (BP) learning algorithm is one of the most important historical developments in neural networks. As a consequence neural networks trained through BP are the most commonly used for applications in a wide range of areas, such as pattern recognition, signal processing, data compression, and automatic control [44]. Figure 3.3 reports an architecture of a multilayer perceptron neural network, which is one of the models where a BP algorithm is mostly used. The general principle is as follows:

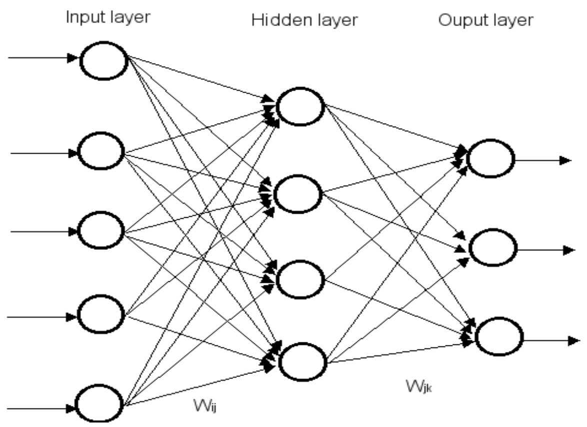


Figure 3.3.: Architecture of a multi-layer perceptron neural network.

given a training data set of input-output pairs $\{(x^{(l)}, d^{(l)}), l = 1, 2, \ldots, p\}$, the algorithm provides a procedure for changing the weights in the network in order to classify the given input patterns correctly [54]. Every time an input is presented to the network, an output is produced. The output produced is compared to the desired output. The error (that is the difference between the produced output and the desired output) of each of the output neurons is computed. Therefore the BP learning algorithm is applied to the network consisting of processing elements with continuous differentiable activation functions. Given the input-output pair $(x^{(l)}, d^{(l)})$, the backpropagation algorithm performs two phases of data flow: the *forward* and the *backward* phase.

Forward Phase

During the forward phase, the input pattern $x^{(l)}$ is propagated from the input layer to the output layer to produce output $y^{(l)}$. Then the errors resulting from the difference between $d^{(l)}$ and $y^{(l)}$ are back-propagated from the output to the previous layers to update their weights. Let us consider a three layers back-propagation network to illustrate the details for this learning algorithm. The results could be easily extended to networks with any number of layers.

Let us consider an I–J–K net architecture where I, J, and K represent the number of nodes in the first, second and third layer respectively. Let us consider the input-output pair (x, d) where the superscript is omitted for notation simplification. Let us assume that a neuron j in the hidden layer receives an input pattern x. Its total output will be:

$$net_j = \sum_{i=1}^{I} w_{ij} \cdot x_i$$

and its output will be:

$$z_j = f(net_j) = f\left(\sum_{i=1}^{I} w_{ij} \cdot x_i\right).$$

The net input for a neuron k in the output layer is then

$$net_k = \sum_{j=1}^{J} w_{jk} z_j = \sum_{j=1}^{J} w_{jk} f\left(\sum_{i=1}^{I} w_{jk} x_i\right)$$

and it produces an output of

$$\begin{aligned} y_k &= f(net_k) \\ &= f\left(\sum_{j=1}^{J} w_{jk} \cdot z_j\right) \\ &= f\left[\sum_{j=1}^{J} w_{jk} f\left(\sum_{i=1}^{I} w_{ij} x_i\right)\right] \end{aligned}$$

Backward Phase

During the backward phase, the error between the net output and the desired output is computed as:

$$\begin{aligned} E(\omega) &= \tfrac{1}{2}\textstyle\sum_{k=1}^{K}(d_k - y_k)^2 \\ &= \tfrac{1}{2}\textstyle\sum_{k=1}^{K}\left[d_k - f(net_k)\right]^2 \\ &= \tfrac{1}{2}\textstyle\sum_{k=1}^{K}\left[d_k - f\left(\sum_{j=1}^{J} w_{jk}\cdot z_j\right)\right]^2. \end{aligned}$$

- The weights connecting the node j in the hidden layer to the node k in the output layer are updated by the formula:

$$\Delta\omega_{jk} = -\eta\,\frac{\partial E}{\partial\omega_{jk}}.$$

That yields to

$$\begin{aligned} \Delta\omega_{jk} &= \eta\cdot(d_k - y_k)\cdot f'(net_k)\cdot z_j \\ &= \eta\cdot\delta_k^{(o)}\cdot z_j \end{aligned}$$

where $\delta_k^{(o)} = (d_k - y_k)f'(net_k)$ represents the error signal of the $k-$th node in the output layer.

- The weights connecting the node i in the input layer to the node j in the hidden layer are updated by

$$\begin{aligned} \Delta w_{ij} &= -\eta\cdot\tfrac{\partial E}{\partial w_{ij}} \\ &= -\eta\cdot\tfrac{\partial E}{\partial net_j}\cdot\tfrac{\partial net_j}{\partial w_{ij}} \\ &= -\eta\cdot\tfrac{\partial E}{\partial z_j}\cdot\tfrac{\partial z_j}{\partial net_j}\cdot\tfrac{\partial net_j}{\partial v_{ij}} \\ &= \eta\cdot\textstyle\sum_{k=1}^{K}(d_k - y_k)\,f'(net_k)\cdot\omega_{jk}\cdot f'(net_j)\cdot x_i \\ &= \eta\cdot\textstyle\sum_{k=1}^{K}\delta_k^{(o)}\cdot\omega_{jk}\cdot f'(net_j)\cdot x_i \\ &= \eta\cdot\delta_j^{(h)}\cdot x_i \end{aligned}$$

where $\delta_j^{(h)} = \sum_{k=1}^{K}\delta_k^{(o)}\cdot\omega_{jk}\cdot f'(net_j)$ is the error on node $j-$th in the hidden layer. When the bipolar sigmoid function[3] is used as activation function, one can easily show that

[3]A bipolar sigmoid function is defined as $g(x) = \frac{2}{1+e^{-x}} - 1 = \tan h(\frac{x}{2})$

$$\begin{cases} \delta_k^{(o)} = \frac{1}{2}(1-y_i^2)(d_k - y_k) & \text{for nodes in the output layer} \\ \delta_j^{(h)} = \frac{1}{2}(1-z_j^2)\cdot\sum_{k=1}^{K}\delta_k^{(o)}\,\omega_{jk} & \text{for nodes in the hidden layer.} \end{cases}$$

Other differentiable functions can be used as activation functions. In practice, the sigmoid function, and the hyperbolic tangent function are the most used since they have simple derivatives and make the implementation of the back propagation algorithm much easier [4].

3.7.2. Drawbacks of Back-Propagation Networks

Despite their success in solving real-world problems, MLP networks present some weakness. For instance, back propagation learning algorithms are very slow and may be trapped into local minima when applied to identify parameters in adaptive neural networks [43]. There are some parameters that depends upon the task under examination, the size of the training data, the architecture to use, etc.

For instance, the choice of learning rate η is critical. Another solution used is the introduction of the momentum term in the updating formula. In this case, the updating formula becomes

$$\Delta\omega_{iq}(t) = -\eta\,\frac{\partial E}{\partial \omega_{iq}} + \alpha\Delta\omega_{iq}(t-1)$$

where α $(0 < \alpha < 1)$ represents the momentum.

Alternative algorithms are proposed up today to avoid such problems. Also the network architecture is important for the net performance. The number of hidden layer nodes is a question often raised in applications of MLP to real-world problems [54].

3.7.3. Least-Squares Estimator

The least-squares estimator is known in statistics as linear regression. The output of a linear model y is given by the following expression

$$y = \theta_1 f_1(u) + \theta_2 f_2(u) + \cdots + \theta_n f_n(u)$$

where $u = [u_1, \cdots, u_m]^T$ is the model's input vector, $f_{1,}$ $f_2, \cdots, f_n$ are known functions of u, and $\theta_1, \cdots, \theta_n$ are unknown parameters. Substituting each of the m data pairs

(x_i, y_i) into the above expression yields to a set of m linear equations

$$\begin{cases} \theta_1 f_1(u_1) + \theta_2 f_2(u_1) + \cdots + \theta_n f_n(u_1) & = & y_1 \\ \theta_1 f_1(u_2) + \theta_2 f_2(u_2) + \cdots + \theta_n f_n(u_2) & = & y_2 \\ \vdots & \vdots & \vdots \\ \theta_1 f_1(u_m) + \theta_2 f_2(u_m) + \cdots + \theta_n f_n(u_m) & = & y_m. \end{cases}$$

Using matrix notation, the previous system takes the form $A\,\theta = y$ where matrix A is called the *design matrix*, $\theta = [\theta_1, \cdots, \theta_n]^T$, and $y = [y_1, \cdots, y_m]^T$.

The goal is to identify the unknown parameters of the system to be modeled from the data representing desired input-output pairs $\{(u_i, y_i),\ i = 1, \cdots, m\ (m \geq n)\}$. In practice, there are more data pairs than fitting parameters. Therefore, an exact solution satisfying all the m equations might be contaminated by noise, or the model might not be appropriate for describing the target system. For this reason, the matrix equation is modified by incorporating an error vector ϵ to account for random noise or modeling error as follows

$$A\theta + \epsilon = y.$$

Once the error has been included in the model, the problem is to search $\widehat{\theta}$ that minimized the sum of squared error defined by

$$\begin{aligned} E(\theta) &= \epsilon^T \epsilon \\ &= (y - A\theta)^T (y - A\theta) \end{aligned}$$

Theorem: *The least-squares estimator $\widehat{\theta}$ satisfies the normal equation*

$$A^T A \widehat{\theta} = A^T y.$$

Refer to [40] for a proof to this theorem.

3.7.4. Hybrid Learning Algorithm

A hybrid learning algorithm is a combination of at least two different learning algorithms. In the Adaptive Neuro-Fuzzy Inference System developed by Jang in [43], two learning paradigms are combined for identifying premise parameters and consequence parameters. The two learning algorithms are the least-squares estimator and the gradient descent. Two updating paradigm techniques can be used: the batch and the pattern learning which are described below.

Batch Learning Algorithm

For batch learning, the updating takes place only after the whole training data set has been presented. The generic premise[4] parameter α of the given adaptive network, is $\Delta\alpha = -\frac{\partial E}{\partial \alpha}$ where $\eta = \frac{k}{\sqrt{\sum_\alpha \left(\frac{\partial E}{\partial \alpha}\right)^2}}$. The step size k should be selected for varying the speed of convergence.

On-Line Learning Algorithm

For on-line learning, the parameters are updated immediately after each input-output pair has been presented. The updating is performed using the formula

$$\frac{\partial E_p}{\partial \alpha} = \sum_{O*\in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial \alpha}$$

where S is the set of nodes whose outputs depend upon the generic premise parameter α.

Hybrid Learning Algorithm

A good approach for solving the learning problems mentioned above is to combine the gradient descent and the least-square-estimator for parameter identification [39]. This hybrid learning works under the following consideration: Assuming that the adaptive network has only one output represented by $O = F(I, \mathcal{S})$ where I is the set of input variables and $\mathcal{S}$ is the set of parameters ($\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$), $\mathcal{S}_1$ is the set of premise parameters, $\mathcal{S}_2$ the set of consequent parameters, and F is the overall function implemented by the network. If there is a function H such that the composite function $H \circ F$ is linear in some of the elements of $\mathcal{S}$, then these elements can be identified by the least-squares estimator method [39, 44]. Now given values of elements in $\mathcal{S}_1$, plugging p training data in this equation, the following matrix equation is obtained.

$$AX = B$$

where X is an unknown vector whose components are consequent parameters[5] in $\mathcal{S}_2$. When combining the gradient descent and the least square estimator to update the

[4]Premise parameter will be reviewed in chapter 5 in details.
[5]Also called linear parameters. We will review linear parameters in Chapter 5.

parameters in an adaptive network, each epoch of this learning procedure is composed of a forward and a backward phase. In the forward phase, input data and functional signals go forward to calculate each node output, until the matrices A and B are obtained and the parameters in $\mathcal{S}_2$ are identified, by the sequential least square formulas,

$$\begin{cases} X_{i+1} = X_i + S_{i+1}a_{i+1}(b_{i+1}^T - b_{i+1}^T) \\ S_{i+1} = S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}} \end{cases}$$

where S_i is a covariance matrix and the least squares estimate X^* is equal to X_p. See [43] for details. After identifying the parameters in $\mathcal{S}_2$, the functional signals keep going forward till the error measure is calculated. In the backward phase, the errors propagate from the output to the input and the parameters in $\mathcal{S}_1$ are updated by $\Delta\alpha = -\eta\frac{\partial E}{\partial \alpha}$.

Summary

In this chapter, we have reviewed some supervised network models and some of the learning algorithms. Neural Networks possess optimization abilities, and connectionist structures. They can learn and generalize from training data. Neural Networks have been used successfully in various areas including control systems, vision, speech processing, optimization, signal classification, robotics, medical applications, and many others. The field of neural networks is expanding very rapidly. The combination of different learning techniques is a powerful approach for solving the drawbacks of some learning algorithms.

4. Noise Filtering Techniques

We cannot design speech recognition systems that work only in the quiet controlled environment of a laboratory. In real-world situations, there will be multiple voices and background noise - street sounds, music or the hum of an air conditioner to contend with.

– Pat Russo.

Introduction

Noise is everywhere and is easily added to a given signal. As pointed in [3], the enhancement of noisy speech is of great importance in voice communication systems that are to be used in real acoustic environments. The unavoidable noise often corrupts the speech signal in different ways. Noise could be introduced by the communication channel at the transmission point before the signal is transmitted, at the receiver side, or at several stages. The objective of a speech enhancement algorithm for human-human communication is the improvement of the perceptual aspects of the speech signal such as quality and intelligibility [18]. Quality is a subjective measure which indicates how pleasant or disturbing the signal is to the listener, while intelligibility is an objective measure of the amount of information in the signal that can be retrieved by the listener. Depending upon the different types of noise, the quality of the received waveform can range from being slightly degraded to being annoying to listen to, and finally to being totally unintelligible. This chapter presents different noise filtering techniques, namely Spectral Subtraction, RASTA filtering, Singular Value Decomposition, Adaptive Filtering.

4.1. Different Types of Noise

Noise is considered as any kind of sound which can be added to a given signal but which is not meaningful for the signal. A noise is said to be Gaussian if repeated sampling of the noise signal leads to a Gaussian normal distribution. There is a tendency for noise signals to be Gaussian because noise is usually the result of many independent random events [31]. According to Markowitz in [61] there are four various types of noise that affect speech recognition systems namely background noise, channel noise, speech noise and non-communication vocalizations.

4.1.1. Background Noise

Background noise is part of the surrounding speaking environment and it enters the input device along with the speech. It can include other voices (such as babble noise), machinery noise (such as car noise), and sounds resulting from human activity. Background noise is called additive since it is superimposed upon the speech input [61]. It can occur at any frequency or range of frequencies, including those critical for speech. Our work was aimed to cancel four different types of background noise: babble - car - traffic - and white noise [19, 20, 21, 22, 23].

4.1.2. Channel Noise

Channel noise refers to the effect of the speech input device on the spoken input. The input device is called speech channel. The functions of the speech channel are to transform sound waves into analog electrical signals and to transmit those signals [61]. The primary channel used for speech consists of a microphone which could be the microphone of the laboratory or the microphone in the telephone handset. Microphones vary in quality and type. Since it is extremely difficult to remove microphone noise characteristics from a signal they are generally encoded in the reference models of an system.

4.1.3. Non-Communication Speech Noise

The act of speaking produces sounds that cannot be considered as messages. Such speech noise includes lip smacks, air puffs, clearing of the throat, and tongue clicks. A lip smack noise at the start of an utterance, is predictable and can be ignored by the system or modeled as special words. Spontaneous speech contains mid-utterance corrections [61].

4.1.4. Non-Communication Vocalizations

Speakers hesitate, stutter, correct errors in the middle of words ("it is yell— Uh! No! blue") and fill pauses with non-communication sound like "Uh", self-corrections, and interjections. These patterns are beginning to be modeled but are still little understood [61].

4.2. Use of Neural Networks

Neural networks for speech enhancement are used as a direct nonlinear time-domain filter. Neural models are better at capturing the dynamics of speech than simple linear models. It is believed [82] that neural networks allow for compensation of nonlinear channel effects, and some relaxation of the requirement that the additive noise be independent of the signal. Tamura [88] suggested that hidden layers provide a transformed representation of the speech signal and noise which facilitates their separation.

4.3. Techniques for Noise Cancellation

There are many techniques for filtering noise from a speech signal. The usual method of estimating a signal corrupted by additive noise is to pass the composite signal through a filter that tends to suppress the noise while leaving the signal relatively unchanged. We have no intention of reviewing in this dissertation all of such existing techniques in different noisy environments. Doclo et al. [14] classified different noise reduction algorithms mainly into three categories. The first approach is based on *spectral subtraction.* The second class of algorithm is based on *classical adaptive filtering.* In this approach microphone recordings of all the noise sources are necessary. The third approach is based on *singular value decomposition.* The idea, in this case, is to consider the corrupted signal as a vector in an $n-$dimensional space and to separate it into a clean signal and a noise signal, lying in orthogonal subspace [13].

We will review here a few methods belonging to three approaches, namely spectral subtraction, RASTA-like filtering, adaptive interference canceling.

4.3.1. Spectral Subtraction

Spectral subtraction is a time-domain filtering. It was proposed by Boll [9] as a preprocessing technique for reducing the spectral effects of additive noise. It is largely used due to its relative simplicity and ease of implementation. When speech is corrupted with additive noise, the spectra of speech and noise are also added, that is

$$\begin{aligned} y(n) &= x(n) + \eta(n) \\ Y(\omega) &= X(\omega) + N(\omega) \end{aligned}$$

where $x(n)$, $y(n)$ are time samples of the clean and corrupted speech and $X(\omega)$, $Y(\omega)$ are their corresponding spectra. $\eta(n)$ and $N(\omega)$ are the time and frequency-domain representations of the corrupting noise source.

In doing spectral subtraction, the power spectrum $\widehat{P}_y$ (magnitude squared of the short-term Fourier transform) of the corrupted speech is computed and an estimate of noise spectrum is subtracted out to produce the estimated spectrum $\widehat{P}_x$ of the clean speech; i.e:

$$\widehat{P}_x = [\widehat{P}_y^{\gamma} - \alpha \widehat{P}_n^{\gamma}]^{1/\gamma}$$

where the scaling factor α allows for emphasis or deamphasis of the noise estimate, and allows for several variants, including power subtraction ($\gamma = 1$) and magnitude subtraction ($\gamma = 0.5$).

Spectral subtraction does not always improve the speaker recognition performance because it distorts the speech waveform introducing annoying musical noise [9, 72].

4.3.2. RASTA Filtering

Modulation frequency components of speech are mainly concentrated between 1 Hz and 16 Hz. Slowly-varying or fast-varying noises have components outside the speech range. The RelAtive SpeTrAl (RASTA) processing proposed by Hermansky et al. [33] can be considered as a technique capable of modifying the modulation frequency content of a noisy speech by filtering the time trajectories of a time-feature representation of speech [3, 33]. The RASTA filter is implemented as an autoregressive-moving average bandpass filter (IIR), with a magnitude frequency response which suppresses modulation frequencies below 1 Hz and above 16 Hz. When the modulation frequency range is between 1 and 16 Hz, RASTA filters for additive noise reduction yield to results similar to spectral subtraction [3].

4.3.3. Singular Value Decomposition

The idea with singular value decompostion is to consider the corrupted signal as a vector in an n-dimensional space and to separate it into a clean and a noise signal, lying in orthogonal subspace [14]. This is done by computing the SDV of the Hankel matrix of the data signal, reducing this matrix to a lower rank and restoring the Hankel structure [14]. The procedure is described as follows:
The noisy signal vector $y = [y[0], y[1], \cdots, y[N-1]]$ is assumed to be additive. So $y[k] = x[k] + n[k]$ where $x[k]$ corresponds to the clean signal and $n[k]$ to the noise. A Hankel matrix $Y \in \mathbb{R}^{l \times m}$

$$Y = \begin{bmatrix} y[0] & y[m-1] & \cdots & y[m-1] \\ y[1] & y[2] & \cdots & y[m] \\ \vdots & \vdots & \vdots & \vdots \\ y[l-1] & y[l] & \cdots & y[l+m-2] \end{bmatrix}$$

with $l \geq m$ is constructed from the signal y. The matrix Y is viewed as the sum of two Hankel matrices X and N constructed respectively from the clean signal x and the noise n. Assume that the clean signal x consists of a sum of p complex exponentials, then the matrix X has rank $p \leq m$. Taking the SVD of the Hankel matrix Y, can be assumed to be of the form:

$$Y = U \Sigma V^T = [U_1 U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}$$

where $U_1 \in \mathbb{R}^{l \times p}$, $U_2 \in \mathbb{R}^{l \times (m-p)}$, $\Sigma_1 \in \mathbb{R}^{p^2}$, $\Sigma_2 \in \mathbb{R}^{(m-p)^2}$, $V_1 \in \mathbb{R}^{m \times p}$, and $V_2 \in \mathbb{R}^{m \times (m-p)}$. Σ_1 contains the p largest singular values corresponding to the signal subspace whereas Σ_2 contains the (m-p) smallest singular values corresponding to the noise subspace. From the SVD of the Hankel matrix Y, it is possible to construct a least-squares estimate of X. The best rank-p approximation of the original matrix Y is

$$Y_p = [U_1 U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} = U_1 \Sigma_1 V_1^T$$

which has lost the Hankel structure [14]. A Hankel approximation of the Y_p is then made averaging along the anti-diagonal of the Y_p and putting each average value in the corresponding diagonal of a new Hankel-structured matrix $\widehat{X}$ of the same dimension,

$$\widehat{X} = \begin{bmatrix} \widehat{x}[0] & \widehat{x}[1] & \cdots & \widehat{x}[m-1] \\ \widehat{x}[1] & \widehat{x}[2] & \cdots & \widehat{x}[m] \\ \vdots & \vdots & \cdots & \vdots \\ \widehat{x}[l-1] & \widehat{x}[l] & \cdots & \widehat{x}[m+l-2] \end{bmatrix}$$

where

$$\widehat{x}[k] = \frac{1}{\beta - \alpha + 1} \sum_{i=\alpha}^{\beta} Y_p(k-i+2, i)$$
$$\alpha = \max(1, k-l+2) \;\; \beta = \min(m, k+1).$$

It has been shown that the signal $\widehat{x} = [\widehat{x}[0], \widehat{x}[1], \cdots, \widehat{x}[l+m-2]]$ is closed to the clean signal one [45]. The drawback of this procedure is the choice of the order p. When p is too low, the speech sounds low-pass filtered and the high frequencies lost. On the other hand when p is too high, annoying musical tones are introduced [14].

4.3.4. Adaptive Interference Canceling

Adaptive noise cancellation was introduced by Widrow and Glover [85]. The objective is to filter out an interference component by identifying a linear model between a measurable noise source and the corresponding unmeasurable interference. The general concept of adaptive noise canceling described by Widrow is as follows [84].

- A signal is transmitted over a channel to sensors that receive the signal plus an uncorrelated noise n_0. The combined signal and noise, $s + n_0$ forms the primary input to the canceler.

- A second sensor receives a noise n_1 which is uncorrelated with the signal but correlated in some unknown way with the noise n_0. This sensor provides the reference input to the canceler.

The noise n_1 is filtered to produce an output $\widehat{n}$ that is close replica of n_0. This output is subtracted from the primary input $s + n_0$ to produce the system output,

$$s + n_0 - \widehat{n}.$$

The main problem of this concept is that the characteristics of the channel over which the noise was transmitted to the primary and reference sensors are unknown. Otherwise the filter output could be subtracted from the primary input, and the system output would be the signal alone.

Adaptive noise cancellation using linear filters has been used successfully in real-world applications such as interference canceling in electrocardiograms (ECGs), echo elimination on long-distance telephone transmission lines, and antenna sidelobe interference canceling [84].

Linear adaptive noise cancellation concept could be expanded into the nonlinear realm by using nonlinear adaptive systems [84].

Summary

There are many techniques for speech noise cancellation. Filtering techniques in time-domain as spectral subtraction are easy to implement. However they do not always produce a good result since they distort the speech waveform. On the other side, frequency-domain filtering techniques are unuseful when the spectrum of the distorted noise and the spectrum of information signal overlap each other considerably. It appears that Adaptive Neuro-Fuzzy Inference Systems filtering that combines neural network capabilities and fuzzy logic potentialities are a promising technique to filter out noise from speech signals.

5. Neuro-Fuzzy Inference System for Speech Enhancement

A journey of thousand miles
must begin with a single step.
– **Lao-Tzu.**

5.1. Introduction

One of the major constraints for general applications of Automatic Speech Recognition is the unavoidable presence of background noise. In a pratical setting, the speaker cannot be isolated in order to have a clean acquisition of the uttered speech to be processed. Noise canceling from signal with adaptive interference filtering technique has shown to be highly advantageous over traditional filtering techniques in many applications where fixed filters are not efficient [3, 82, 84]. Neural networks have the capability to learn from scratch, and might have to do it any time if, substantial parameters of the dynamical system change for some reason whereas fuzzy systems have the potentialities to represent the vagueness and ambiguity in human thinking.

In this chapter, we present a neuro-fuzzy system that combines neural network potentialities and fuzzy system capabilities to cancel background noise from a speech signal. In the implementation the used Adaptive Network Fuzzy Inference System (ANFIS) architecture to identify the unknown nonlinear passage dynamics that transforms a noise source into an interference component in a detected signal. The hybrid learning algorithm of ANFIS that combines the least-squares estimator and the gradient descent is presented. The implemented system is tested quantitatively and qualitatively on a bilingual database. The results obtained are presented.

5.2. Passage Dynamics

Passage dynamics are mathematical models of the channels through which the noise waves undergo before corrupting a given signal such as a speech sentence. In general, how the environment modifies the noise source is unknown, but some hypotheses can be applied considering the channel over which the signal, together with the noise is transmitted. For example, if the channel is the air as in the case of cellular telephone, we must assume some reflection effects due to the presence of obstacle such as buildings, mountains, etc. One of the requirements of these models is that they should keep some statistical features of the noise source, such as the mean.

The use of such models is justified by the hypothesis that the noise is not correlated to the original signal and the distortion is simply added to it, to produce a corrupted signal.

Therefore, if such distortion can be modeled in the right way, the original signal can be obtained in subtracting such a distortion from the received signal (which is corrupted by the distortion noise).

Many passage dynamic functions have been suggested, among them those reported in equations 5.1, 5.2, 5.3. In such equations $n(t)$ is the noise source. In equation 5.3 α and β are free parameters introduced to improve the modeling as we will see in Chapter 8.

$$f[n(t), n(t-1)] = 4 \times \frac{(\sin n(t)) \cdot n(t-1)}{1 + [n(t-1)]^2} \tag{5.1}$$

Figures 5.1, 5.2, 5.3 show the shapes of the passage dynamics defined by equations 5.1, 5.2, 5.3 respectively.

$$g[n(t), n(t-1)] = 4 \times \frac{(sinc\ n(t)) \cdot n(t-1)}{[1 + n(t-1)^2]} \tag{5.2}$$

where $sinc\,(x) = \frac{\sin \pi x}{\pi x}$.

$$h[n(t), n(t-1)] = \frac{\alpha}{[2 - n(t)][2 - n(t-1)]} + \beta. \tag{5.3}$$

We defined and applied the above passage dynamic functions in our tentative to implement a system able to cancel noise form speech and we compared their performance through several experiments which are reported in the following chapters.

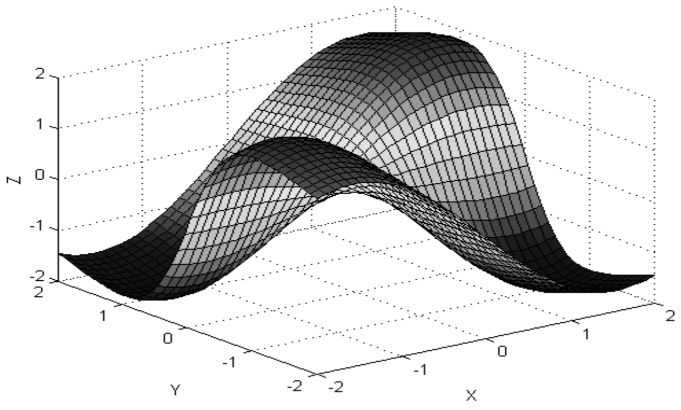


Figure 5.1.: Plot of the passage dynamic function given in equation 5.1.

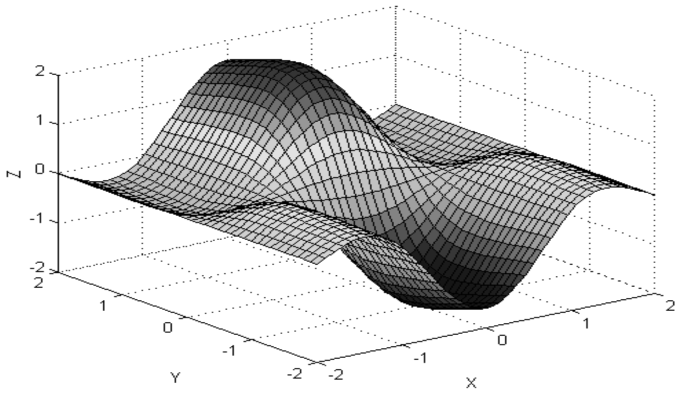


Figure 5.2.: Plot of the passage dynamic function given in equation 5.2.

5.3. ANFIS Architecture

ANFIS is a hybrid learning architecture, proposed by Jang [39, 40] which has proved its superiority over backpropagation neural networks. As an adaptive network, the parameters in some ANFIS nodes are adapted during training. Such nodes are called **adaptive nodes**. Moreover there are nodes whose parameters remain unchanged during training. They are called **fixed nodes**. The general architecture is composed of five layers as shown in Figure 5.4. The number of inputs to ANFIS for the training process depends upon of the order of the passage dynamic function which will be used.

- In our experiments, since we have always used a second order passage dynamic functions, the first layer of our configuration takes as inputs the noise wave $n(t)$

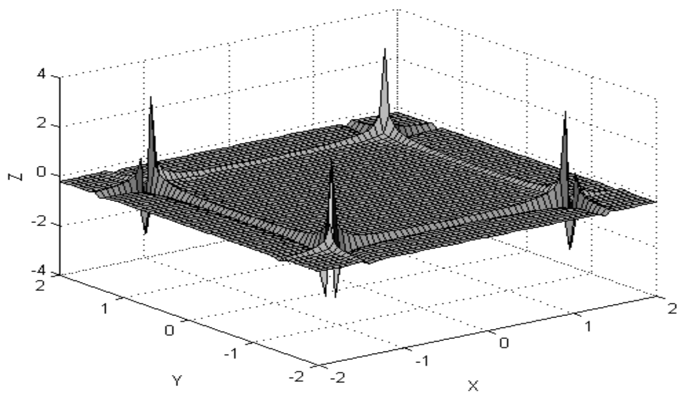


Figure 5.3.: Plot of the passage dynamic function given in equation 5.3.

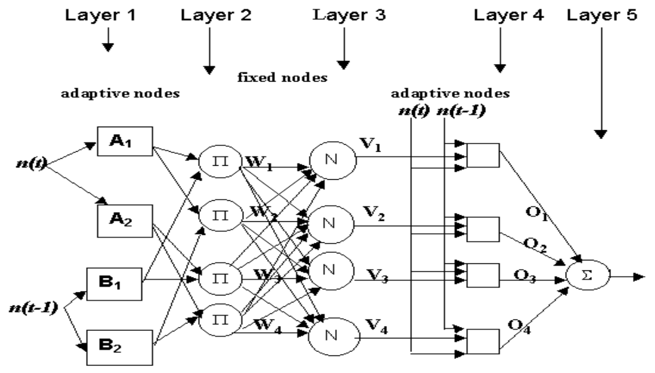


Figure 5.4.: ANFIS architecture.

and its delayed version $n(t-1)$. Membership functions are assigned to each input depending on the context. In [19, 20], generalized bell functions are used, where as trapezoidal membership functions are used in [22] for modeling the linguistic labels such as *High*, and *Low*. The use of linguistic rules constitutes the major benefits of fuzzy techniques description [50]. Membership function parameters are called **premise parameters**. Each node in this layer is an adaptive node with a node function described by the equation:

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

where A_i and B_{i-2} are linguistic labels associated to nodes. Therefore, $O_{1,i}$ specifies the degree to which the given input $n(t)$ or $n(t-1)$ satisfies the quantifier A_1, A_2, B_1, or B_2. For instance, when the membership function for the fuzzy set A_1 is a bell function (three parameters are required), one has

$$\mu_{A_1}(x) = \frac{1}{1 + |\frac{x-c}{a}|^{2b}},$$

where $\{a,\, b,\, c\}$ is a part of premise parameters set.

- The second layer is constituted of **fixed nodes** whose outputs are the product of all incoming signals

$$\begin{aligned} O_{2,i} &= w_i \\ &= prod\{\mu_{A_j(n(t-1)\}}, \mu_{B_k(n(t-1))}\},\ j,\, k = 1, 2. \\ &= \mu_{A_j(n(t))} \times \mu_{B_k(n(t-1))},\ j,\, k = 1, 2. \end{aligned}$$

Each node represents the firing strength of a rule. Any other T-norm operator that performs fuzzy AND can be used as the node function in this layer. The reader can refer to Chapter 2 for other T-norm operators.

- The third layer is also constituted of fixed nodes. The i-th node calculates the ratio of the i-th rules's firing strength to the sum of all rules firing strengths, i.e.

$$\begin{aligned} O_{3,i} &= \overline{w_i} \\ &= \tfrac{w_i}{w_1+w_2+w_3+w_4},\ i = 1, \cdots, 4. \end{aligned}$$

  Since $\sum_i O_{3,i} = 1$, the outputs of these nodes are called normalized firing strengths.

- The fourth layer is constituted of adaptive nodes whose node function is defined as:

$$O_{i,4} = \overline{w}_i[p_i n(t) + q_i n(t-1) + r_i]$$

  where $\overline{w}_i$ is a normalized firing strength from layer 3 and $\{p_i,\, q_i,\, r_i\}$ is the parameter set of this node. Parameters in this layer are referred to as *consequent parameters.*

- The output layer is constituted of only one node that computes the overall output as the sum of all incoming signals, i.e.

$$\begin{aligned} O_{i,5} &= \textstyle\sum_i \overline{w}_i \times [p_i n(t) + q_i n(t-1) + r_i] \\ &= \tfrac{\sum_i w_i[p_i n(t)+q_i n(t-1)+r_i]}{\sum_i w_i}. \end{aligned}$$

5.4. System Description

5.4.1. General Schematic Diagram

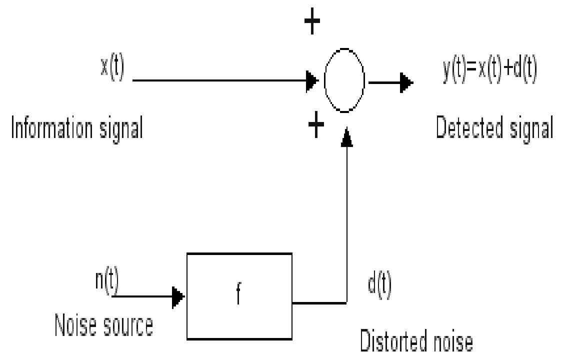


Figure 5.5.: Schematic diagram for noise cancellation.

Figure 5.5, shows an ideal situation to which adaptive noise cancellation can be applied. We have an information signal $x(t)$ and a measurable noise source $n(t)$. The noise source goes through unknown nonlinear dynamics to generate a distorted noise $d(t)$ which is then added to $x(t)$ to form the measurable output signal $y(t)$, which consists of the information signal $x(t)$ plus a distorted and delayed version of $n(t)$, which is $d(t)$.

$$y(t) = x(t) + d(t).$$

5.4.2. ANFIS Filtering Diagram

In practice, the distorted noise signal $d(t)$ cannot be access directly since it is an additive component of the overall measurable signal $y(t)$. Since the information signal $x(t)$ is zero mean and not correlated with the noise signal $n(t)$, we can use the detected signal $y(t)$ as the desired output for ANFIS training as shown in Figure 5.6.

5.4.3. Extended ANFIS Diagram

The filtering diagram we proposed in [19, 20] exploits both Widrow's approach [84, 85] in defining the primary input and ANFIS [39, 40, 41] filtering diagram. The primary input $s(t)$ of such an implemented system is the sum of the information signal $x(t)$ and

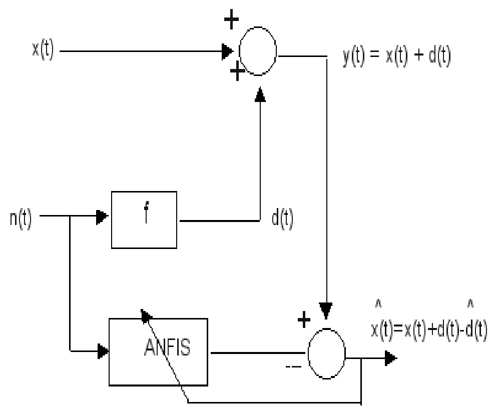


Figure 5.6.: Schematic Diagram for ANFIS.

the noise source $n(t)$.

$$s(t) = x(t) + n(t).$$

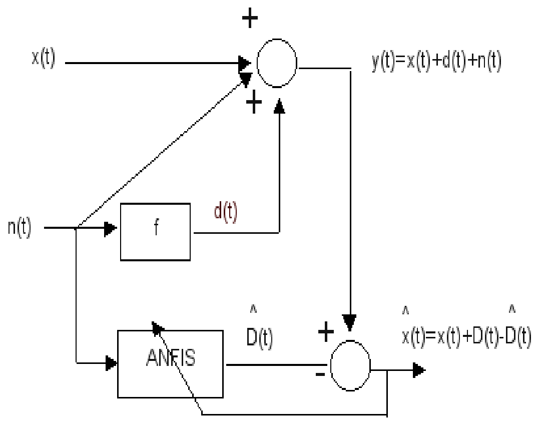


Figure 5.7.: Extension of ANFIS Diagram.

In this way, the detected and measurable signal $y(t)$ which was used as the target of the ANFIS system can be expressed as

$$\begin{aligned} y(t) &= s(t) + d(t) \\ &= x(t) + [n(t) + d(t)] \\ &= x(t) + D(t) \end{aligned}$$

where $D(t) = n(t) + d(t)$ is the noisy component of the measurable wave. The major characteristics of this model are:

- The speech sentence is completely distorted for listeners [19].
- ANFIS training process can even estimate the noisy component $D(t)$. Such an estimation $\widehat{D}(t)$ is subtracted from the detected signal $y(t)$ to obtain an estimation $\hat{x}(t)$ of the information signal $x(t)$.

5.4.4. Network Configuration

The architecture we used has two inputs (one is the noise wave $n(t)$ and the other is its delayed version $n(t-1)$). We assigned two membership functions to each input. The generalized bell function is used in the first experiment whereas the trapezoidal membership function is used in the second experiment. Therefore, the first layer has four adaptive nodes.

In the second layer there are four adaptive nodes whose output is the *min* (in our case) of all the incoming signals.

In the third layer, there are four fixed nodes, and their output is computed as the ratio between the output of the respective node in the previous layer and the sum over all the outputs in the previous layer. Each node in this layer is directly connected with only one node in the next layer.

In the fouth layer there are four adaptive nodes whose output is computed as the ratio between the output coming from the previous layer and a linear combination of the parameters p_i, q_i, r_i called **consequence parameters** weighted by the two inputs $n(t)$ and $n(t-1)$ i.e. $[p_i n(t) + q_i n(t-1) + r_i]$. The four fuzzy rules computed are of the form:

- if $n(t)$ is *Low* and $n(t-1)$ is *Low,* then $f_1 = p_1 n(t) + q_1 n(t-1) + r_1$
- if $n(t)$ is *Low* and $n(t-1)$ is *High,* then $f_1 = p_2(t) + q_2 n(t-1) + r_2$
- if $n(t)$ is *High* and $n(t-1)$ is *Low,* then $f_1 = p_3 n(t) + q_3 n(t-1) + r_3$
- if $n(t)$ is *High* and $n(t-1)$ is *High,* then $f_1 = p_4 n(t) + q_4 n(t-1) + r_4$.

The single node in the fifth layer is a fixed node that computed the overall output as the sum of all incoming signal. Figure 5.8 shows the architecture used. The complete network architecture has:

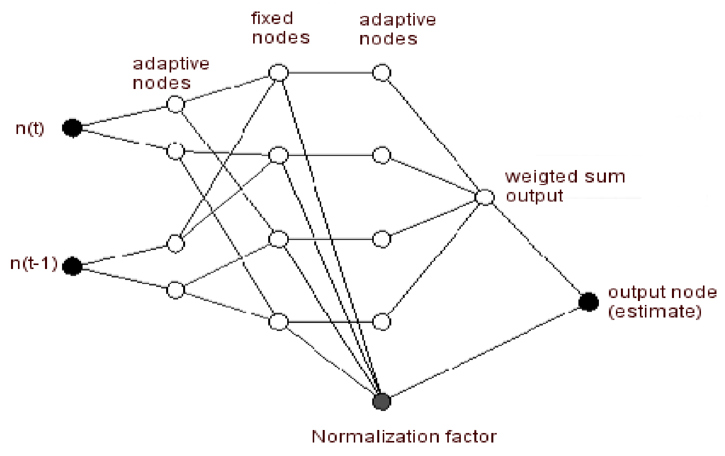


Figure 5.8.: ANFIS architecture used.

- 17 nodes
- 12 premise parameters
- 12 consequent parameters.

5.4.5. Learning of ANFIS

During the training, ANFIS outputs an estimation $\widehat{D}(t)$ of the distorted noise, which will be turned up by comparing it to $y(t)$, which contains the real distorted noise, together with the real speech signal. To update the system parameters, the learning procedure goes through two phases: a forward and a backward phase. During the forward phase, the data flow up to layer 4 and then consequent parameters are identified applying the least-squares estimator. Then the outputs of layer 4 are combined into the layer 5 and the squared error is calculated as the squared difference between the target $y(t)$ and the net output. During the backward phase the error is propagated from the net output to the input, and the premise parameters are updated using the backpropagation algorithm through the formula:

$$\Delta\alpha = \frac{-\kappa}{\sqrt{\sum_\alpha \frac{\partial E_p}{\partial \alpha}}}$$

where α represents a generic premise parameter and κ is the step size, which is updated dynamically using two heuristic rules [40] which are

1. If after 4 consecutive epochs the error function decreases, the step size is increased by 10% of its previous value.
2. If after 4 consecutive epochs the error function undergoes 2 consecutive combinations of one increase and one decrease, the step size is decrease by 10% of its previous value.

In all experiments we have done, κ is fixed to 0.01.

5.5. Database Description

Speech recognition systems need to be tested on real speech data. It is important to have a database which complies with the recognition system's intended use [48]. For example, a system for isolated word recognition should be trained with data from an isolated speech database. Furthermore, the language and even the accent within the language often restricts the use of a database [48]. For better generalization problem, the database should be composed of various types of contexts to reflect the learning capabilities independently of the implemented system. To test the nonlinear adaptive neuro-fuzzy inference system implemented for noise cancellation, we used a bilingual database composed of sentences produced by Italian native speakers and some selected sentences from the TIMIT database.

5.5.1. Italian Database

The Italian database collected by us, contains recordings of four speakers from different areas in Italy. Two females from Salerno and Leece, and two males from Pisa and Palermo. This database was so collected, in order to obtain dialectal accent variations in the produced speech. Each speaker answered to a selected set of four different questions. The recording was made in a quite room, to avoid the background noise as much as possible. The speech wave was sampled in mono-channel mode, at a frequency of 16 kHz and quantized with 16 bits for samples. In Table 5.1 the information related to each of the Italian speakers is summarized .

Speaker	Birthdate	Age	Sex	Accent	Place where they spent most of their life
sdf1	9/10/1974	26	F	Leccese	Lecce
vvf2	9/18/1970	30	F	Salernitano	Salerno
gsm1	1/24/1967	33	M	Palermitano	Palermo
lpm2	4/05/1971	29	M	Pisano	Pisa

Table 5.1.: Details on the speakers.

5.5.2. TIMIT Database

The TIMIT database was produced by **T**exas **I**nstrument and **M**assachusetts **I**nstitute of **T**echnology. This corpus database is composed of English sentences produced by speakers from 8 different demographic regions of United States of America. Each speaker read ten different English sentences namely, 2 dialectal sentences (i.e. sa-sentences), 5 phonetically-compact sentences (i.e. sx-sentences), and 3 phonetically-diverse sentences (i.e. si-sentences). The sentences are different from one speaker to another. Speakers were selected to be representative of different geographical dialectal regions [30]. Information on TIMIT data can be found in [30, 90].

Dialect Sentences (sa-sentences)

The dialect sentences were meant to expose dialectal variants of the speakers and were read by all speakers. There are two dialect sentences which are *"She had your dark suit in greasy wash water all year"* and *"Don't ask me to carry an oily rag like that"*.

Phonetically-Compact Sentences (sx-sentences)

They were hand designed to be comprehensive of all the phonemes of the language. The objective was to provide a good coverage of pairs of phones, with extra occurrences of phonetic contexts thought to be either difficult or of particular interest. Two examples of such sentences are *"Ralph prepared red snapper with fresh lemon sauce for dinner"* and *"Smash light bulbs and their cash value will diminish to nothing"*.

Phonetically-Diverse Sentences (si-sentences)

They were selected from existing text sources - the Brown corpus and a collection of dialogs from recent stage plays - so as to add diversity in sentence types and phonetic contexts. Two examples of such sentences are *"Boys and men go along the river bank or to the alcoves in the top arcade"* and *"Positive results start when it goes towards the hand you use to make your mark"*.

In our experiment, the database used is composed of two sa-sentences, two sx sentences, and two si sentences.

5.5.3. Noise Database

The noise database we used is exclusively background noise. Background noise includes both white noise and coloured noise. Since the noise is usually the result of many independent random events, there is a tendency for a noise signal to be considered Gaussian [47]. There is no modeling on coloured noise. The noise is called white if the power spectrum is independent of frequency [31]. For instance, a white noise signal denoted $n_W(t)$, has a power spectral density given by

$$P_{n_W}(\omega) = \begin{cases} N_0 \text{ (a constant)}, \ |\omega| < \omega_c \\ 0 \text{ if } |\omega| > \omega_c \end{cases}$$

and all frequency components with frequencies in the range $-\omega_c < \omega < \omega_c$, contribute equally to $n_W(t)$ [71].

5.6. Performance Functions

We intensively evaluated the implemented system qualitatively and quantitatively. Perceptual experiment is also performed.

5.6.1. Quantitative Evaluation

Among functions that can be used for evaluating the system performance, we selected the mean, the mean square error and the signal to noise ratio since they are well adapted to the problem under examination.

Average or Mean Value

The average of a vector $x = (x_i)_{1 \leq i \leq n}$ of dimension n is defined by the formula

$$mean(x) = \frac{1}{n} \sum_{i}^{n} x_i.$$

In other words, $mean(x)$ is the mean value of the elements in the vector x. This definition could be easily extended to matrices. In this case, assuming that X is a matrix of dimension $n \times d$, $mean(X)$ is a row vector containing the mean value of each column [74].

Mean Square Error (MSE)

The MSE measures a system's performance according to the mean of squared errors [52]. Let N denote the number of samples in a given speech signal $x(t)$, and in its estimation $\widehat{x}(t)$. The MSE is defined as

$$MSE = \frac{1}{2} \sum_{n=1}^{N} \left[x(n) - \widehat{x}(n)\right]^2 .$$

Signal to Noise Ratio (SNR)

The signal to noise ratio (SNR) is a measure of the ratio between signal energy and noise energy. In speech analysis, SNR usually refers to the periodic energy relative to noise energy. The SNR is computed as

$$SNR(dB) = 10 \times \log_{10} \frac{\sigma^2[x(t)]}{\sigma^2[n(t)]}$$

where $\sigma^2[x(t)]$ and $\sigma^2[n(t)]$ represent the power of the signal and the noise respectively. In our experiments, SNR values were computed twice. Once as the ratio of the clean signal energy over the noise signal energy, and then as the ratio of the clean signal energy over the energy of the resisual noise (difference between $x(t)$ and the estimated clean signal $\widehat{x}(t)$). They were indicated as SNR_1 and SNR_2 and are reported in equations 5.4 and 5.5.

$$SNR_1(dB) = 10 \times \log_{10} \frac{\sigma^2[x(t)]}{\sigma^2[n(t)]} \tag{5.4}$$

$$SNR_2(dB) = 10 \times \log_{10} \frac{\sigma^2[x(t)]}{\sigma^2[x(t) - \widehat{x}(t)]}. \tag{5.5}$$

One could show that the improvement in signal to noise ratio is given as

$$\Delta SNR = 10 \times \log_{10} \frac{\sigma^2[n(t)]}{\sigma^2[x(t) - \widehat{x}(t)]}.$$

Experimental Results

First Experiment:

In this experiment, we assigned two generalized bell functions to each of the two inputs of the ANFIS architecture (see the extended diagram of ANFIS) and we used the passage dynamic function reported in equation 5.1. The number of training epochs was 100.

Figure 5.9, reports the signal to noise ratio improvement obtained.

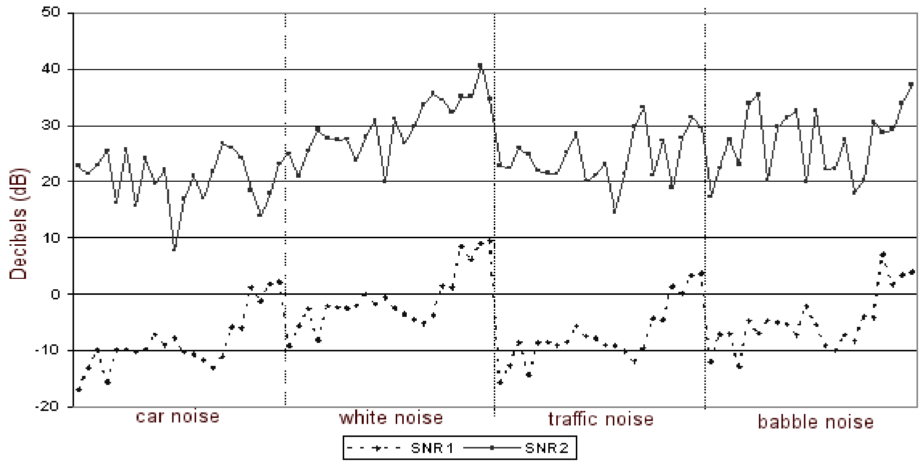


Figure 5.9.: Initial (dashed lines) and final (solid lines) signal to noise ratio computed over all the sentences used.

Table 5.2 summarizes the results obtained over the three quantitative measures defined above.

Second Experiment:

In this experiment, we applied two trapezoidal functions to each of the two inputs of the ANFIS architecture (see the extended diagram of ANFIS) whereas the passage

	Lenght	SNR Improvement	MSE	Aver. Error
Average	73090	30.73 dB		
Minimum	36781	15.05 dB	78.0×10^{-9}	21.4×10^{-5}
Maximum	114800	42.74 dB	67.8×10^{-5}	24.5×10^{-3}

Table 5.2.: Experimental results obtained when the system is trained with the passage dynamic function defined in equation 5.1. Bell functions are applied to each input.

	Lenght	SNR Improvement	MSE	Aver. Error
Average	73090	36.21 dB		
Minimum	36781	20.51 dB	11.0×10^{-9}	6.9×10^{-5}
Maximum	114800	45.89 dB	27.1×10^{-5}	16.38×10^{-3}

Table 5.3.: Experimental results obtained when the passage dynamic function defined in equation 5.2. Trapezoidal membership functions are applied to each input.

dynamic function used is the one reported in equation 5.2. The use of passage dynamic functions is for better modeling the environment through which the noise waves undergo before corrupting the speech signal. The use of trapezoidal function or more generally piece-wise linear functions is more appropriate to reduce the computational time [40]. The number of training epochs was 60. The results are reported in Table 5.3.

5.7. Qualitative Evaluation

For the qualitative evaluation of a system's performance, we used the speech waveform and the speech spectrogram as qualitative measures of the results obtained.

5.7.1. Speech Waveform

A speech waveform is a graph showing the varying amplitude of a speech signal over the time [47]. Figure 5.10 is the plot of three waveforms: the original speech sentence – the same speech sentence corrupted with babble noise – and its estimate after applying the noise canceling procedure. The sentence was extracted from the Italian database.

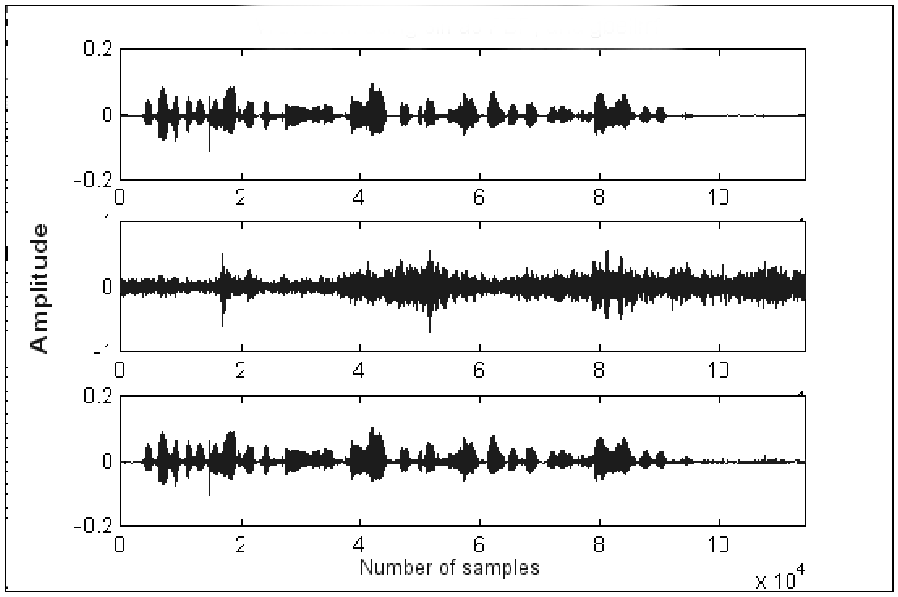


Figure 5.10.: From top to bottom are reported, the waveform of the original sentence, the waveform of the noisy sentence (babble noise is used), the waveform of the estimated sentence.

5.7.2. Speech Spectrogram

A speech spectrogram is a pattern for sound analysis containing information on time, frequency and intensity [47, 51].

- Time is shown on the horizontal axis.
- Frequency is shown on the vertical axis.
- Intensity on the gray scale i.e. the intensity of each frequency at a given instant is shown by the darkness of the mark. Figure 5.11 reports three spectrograms: the clean speech sentence – the same sentence corrupted with traffic noise – and its estimate once our algorithm of noise cancellation was applied on the noisy sentence. The signal was completely reconstructed (see bottom of the Figure 5.11.)

The sentence used, is extracted from the TIMIT database. It can be noticed that very little noise remains on the output wave, independently of the kind of noise used.

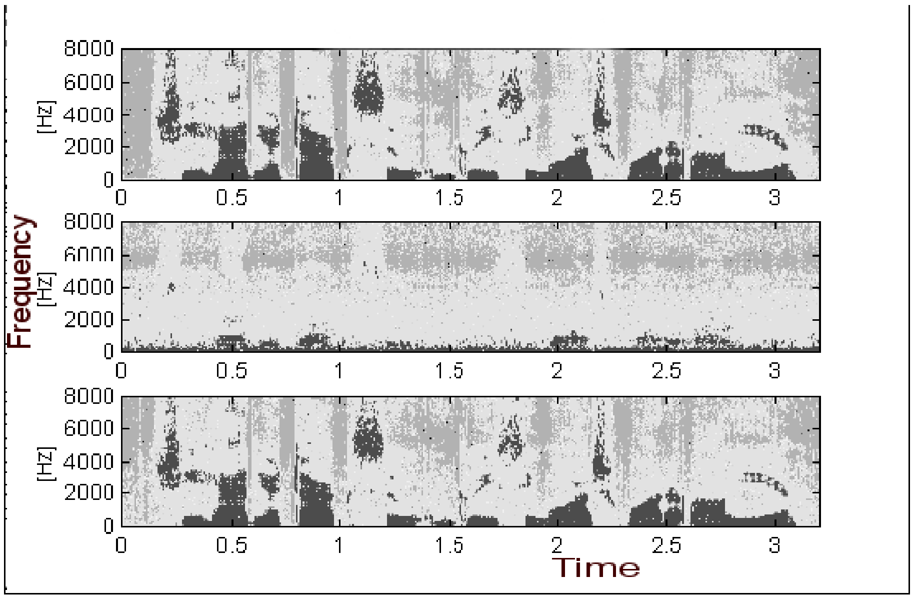


Figure 5.11.: From top to bottom are reported, the spectrogram of the original sentence, the spectrogram of the noisy sentence (babble noise is used), the spectrogram of the estimated sentence.

5.7.3. Perceptual Experiments

To further evaluate the effects of our algorithm of noise cancellation, we let some native speakers of Italian to listen the sentences both in the noisy and when the noise was removed form them. The results, which are conformed at this time, showed that in listening the original and the estimated speech sentences, it is mostly impossible to distinguish the clean from the estimated sentence.

5.7.4. Results Analysis

The improvement obtained in defining another passage dynamic function can be observed in Figure 5.12. The results obtained are very good compared to those in [3, 82] only 10 dB as improvement even though the database is not the same. Moreover, the passage dynamic function we proposed better models the environment through which the noise waves undergo before corrupting the speech signal.

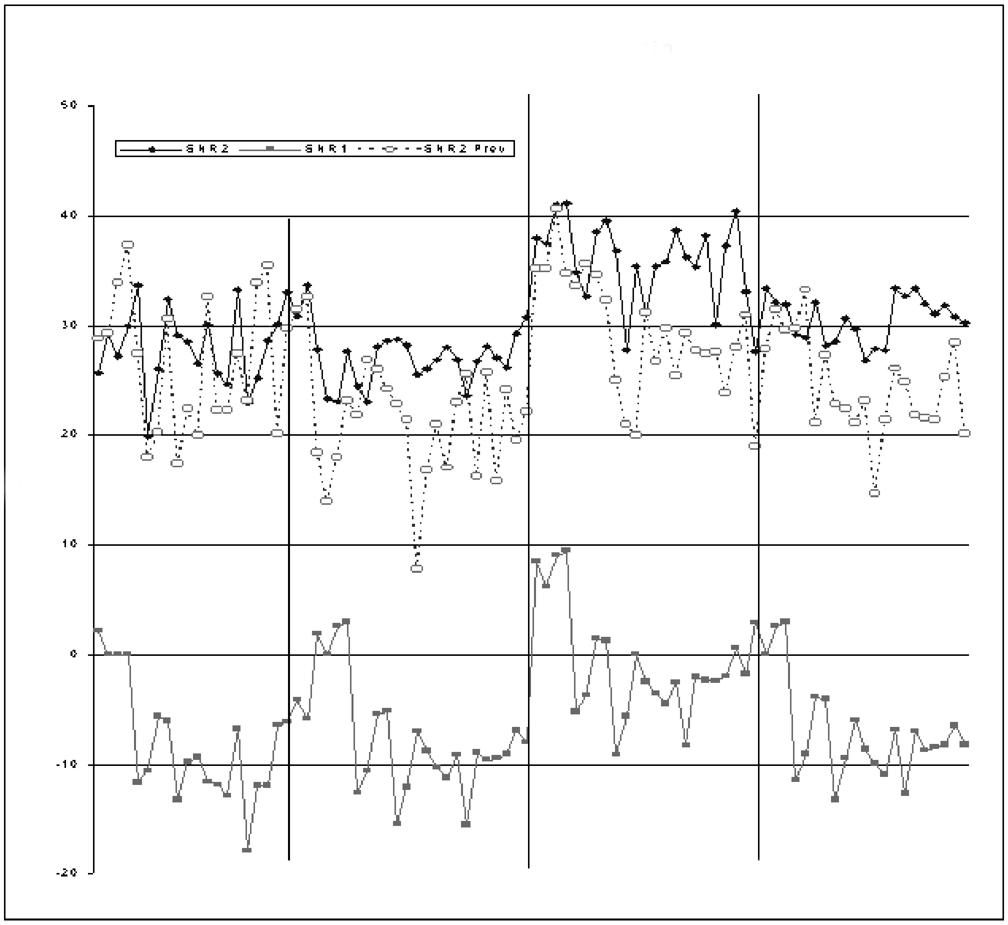


Figure 5.12.: Comparision of the signal to noise ratio improvement obtained in the first and second experiments.

Summary

In this chapter, we have presented an implementation of a neuro-fuzzy system for canceling noise from a speech signal. Two experiments were carried out with different setups. As the results showed, in both cases, the implemented system performs very well. However, one of the drawbacks of such a system is the computational time that increases with the length of the sentence and the size of the database. The next chapter presents an approach for handling this problem.

6. Designing a Fast Neuro-Fuzzy System for Noise Cancellation

It is with logic that one proves;
it is with intuition that one invents.
– Henri Poincaré.

6.1. Introduction

The real-time computation is a computation in which part of the problem specification involves the achievement of certain results within a certain elapsed time.

In this chapter, we present an experimental neuro-fuzzy inference system which has been implemented with the objective to cancel noise from a speech signal in almost real-time. The novelty of this system consists in a setup that allows a better generalization in learning the noise features. The system was trained only once with a sample of babble noise during few epochs. The fuzzy inference system obtained has the capability to clean speech sentences corrupted not only by the same type of noise which was babble noise but also by the car, traffic, and white noise. The average improvement in terms of signal to noise ratio was 37 dB without further training, resulting a great reduction of the computational time.

6.2. Architecture Description

The architecture used is composed of five layers as described in the previous chapter. For this implementation, we applied *seven trapezoidal membership functions* to each of the two inputs. The motivation for the use of such a number of membership functions comes from the hypothesis that, a greater number of linguistic values will give a more detailed description about the noise waves. The lingusitic values assigned to each input variable are respectively *"Very Low (VL)"*, *"Low (L)"*, *"More or Less Low (MLL)"*, *"Medium (M)"*, *"More or Less High (MLH)"*, *"High (H)"*, *"Very High (VH)"*. Therefore the first layer has 14 adaptives nodes whose node functions are represented by

$$\begin{aligned}
O_{1,1} &= \mu_{VL_1}(n(t)) \\
O_{1,2} &= \mu_{L_1}(n(t)) \\
O_{1,3} &= \mu_{MLL_1}(n(t)) \\
O_{1,4} &= \mu_{M_1}(n(t)) \\
O_{1,5} &= \mu_{MLH_1}(n(t)) \\
O_{1,6} &= \mu_{H_1}(n(t)) \\
O_{1,7} &= \mu_{VH_1}(n(t)) \\
O_{1,8} &= \mu_{VL_2}(n(t-1)) \\
O_{1,9} &= \mu_{L_2}(n(t-1)) \\
O_{1,10} &= \mu_{MLL_2}(n(t-1)) \\
O_{1,11} &= \mu_{M_2}(n(t-1)) \\
O_{1,12} &= \mu_{MLH_2}(n(t-1)) \\
O_{1,13} &= \mu_{H_2}(n(t-1)) \\
O_{1,14} &= \mu_{VH_2}(n(t-1)).
\end{aligned}$$

The second layer has 49 fixed nodes whose outputs are the product of all incoming signals, according to the following expressions:

$$\begin{array}{ccccl}
O_{2,1} & = & w_1 & = & \mu_{VL_1}(n(t)) \times \mu_{VL_2}(n(t-1)) \\
O_{2,2} & = & w_2 & = & \mu_{VL_1}(n(t)) \times \mu_{L_2}(n(t-1)) \\
\vdots & \vdots & \vdots & \vdots & \\
O_{2,7} & = & w_7 & = & \mu_{VL_1}(n(t)) \times \mu_{VH_2}(n(t-1)) \\
O_{2,8} & = & w_8 & = & \mu_{L_1}(n(t)) \times \mu_{VL_2}(n(t-1)) \\
O_{2,9} & = & w_9 & = & \mu_{L_1}(n(t)) \times \mu_{L_2}(n(t-1)) \\
\vdots & \vdots & \vdots & \vdots & \\
O_{2,16} & = & w_{16} & = & \mu_{L_1}(n(t)) \times \mu_{VH_2}(n(t-1)) \\
O_{2,17} & = & w_{17} & = & \mu_{MLL_1}(n(t)) \times \mu_{VL_2}(n(t-1)) \\
O_{2,18} & = & w_{18} & = & \mu_{MLL_1}(n(t)) \times \mu_{L_2}(n(t-1)) \\
\vdots & \vdots & \vdots & \vdots & \\
O_{2,24} & = & w_{24} & = & \mu_{MLL_1}(n(t)) \times \mu_{VH_2}(n(t-1)) \\
O_{2,25} & = & w_{25} & = & \mu_{MLH_1}(n(t)) \times \mu_{VL_2}(n(t-1)) \\
O_{2,26} & = & w_{26} & = & \mu_{MLH_1}(n(t)) \times \mu_{L_2}(n(t-1)) \\
\vdots & \vdots & \vdots & \vdots & \\
O_{2,32} & = & w_{32} & = & \mu_{MLH_1}(n(t)) \times \mu_{VH_2}(n(t-1)) \\
O_{2,33} & = & w_{33} & = & \mu_{H_1}(n(t)) \times \mu_{VL_2}(n(t-1)) \\
O_{2,34} & = & w_{34} & = & \mu_{H_1}(n(t)) \times \mu_{L_2}(n(t-1)) \\
\vdots & \vdots & \vdots & \vdots & \\
O_{2,40} & = & w_{40} & = & \mu_{H_1}(n(t)) \times \mu_{VH_2}(n(t-1)) \\
O_{2,41} & = & w_{41} & = & \mu_{VH_1}(n(t)) \times \mu_{VL_2}(n(t-1)) \\
O_{2,42} & = & w_{42} & = & \mu_{VH_1}(n(t)) \times \mu_{L_2}(n(t-1)) \\
\vdots & \vdots & \vdots & \vdots & \\
O_{2,49} & = & w_{49} & = & \mu_{VH_1}(n(t)) \times \mu_{VH_2}(n(t-1)).
\end{array}$$

The indexes 1 and 2 of linguistic values refer to the first input $n(t)$ and the second input $n(t-1)$ respectively.

The third layer has 49 nodes, whose outputs are computed as the ratio of the respective node output in the previous layer and the sum over all the outputs in the previous layer, as in the following expressions:

$$\begin{array}{ccl}
O_{3,1} & = & \overline{w}_1 = \dfrac{w_1}{w_1 + w_2 + \cdots + w_{49}} \\
O_{3,2} & = & \overline{w}_2 = \dfrac{w_2}{w_1 + w_2 + \cdots + w_{49}} \\
\vdots & & \vdots
\end{array}$$

$$O_{3,49} = \overline{w}_{49} = \frac{w_{49}}{w_1 + w_2 + \cdots + w_{49}}.$$

Each one of these nodes is directly connected with only one node in the next layer.

In the fourth layer, there are 49 adaptive nodes whose activation functions are computed as

$$\begin{aligned}
O_{4,1} &= \overline{w}_1 \times [p_1 n(t) + q_1 n(t-1) + r_1] \\
O_{4,2} &= \overline{w}_2 \times [p_2 n(t) + q_2 n(t-1) + r_2] \\
\vdots &= \vdots \\
O_{4,49} &= \overline{w}_{49} \times [p_{49} n(t) + q_{49} n(t-1) + r_{49}].
\end{aligned}$$

The fifth layer has a single fixed node which computes the overall output as the sum of all incoming signals.

$$\begin{aligned}
O_{5,1} &= \sum_i \overline{w}_i [p_i n(t) + q_i n(t-1) + r_i] \\
&= \frac{\sum_i \overline{w}_i [p_i n(t) + q_i n(t-1) + r_i]}{\sum_i w_i.}
\end{aligned}$$

In Figure 6.1 is plotted the implemented acrchitecture.

6.3. Experimental Design

In Chapter 5, the system was trained on all the 88 noisy sentences in order to estimate $D(t)$ and then, the sentences were cleaned subtracting the estimated $\widehat{D}(t)$ from the noisy sentences. For those experiments, we used only two membership functions for each of the two inputs, which altogether generated a FIS of only 4 rules. Since the number of membership functions and the number of rules were small, the system had to be trained intensively to come up with a good performance. Due to the fact that the training was repeated for each of the 88 noisy sentences, the overall computational time was very high. To avoid retraining time and to improve the system's performance, we increased the number of membership functions applied to each input node, and, after training the system only once, we processed all the noisy sentences over the FIS obtained. Since Widrow's approach introduced has the aim to completely distort the sentence to the listener, we did not include this approach here for simplicity. Moreover

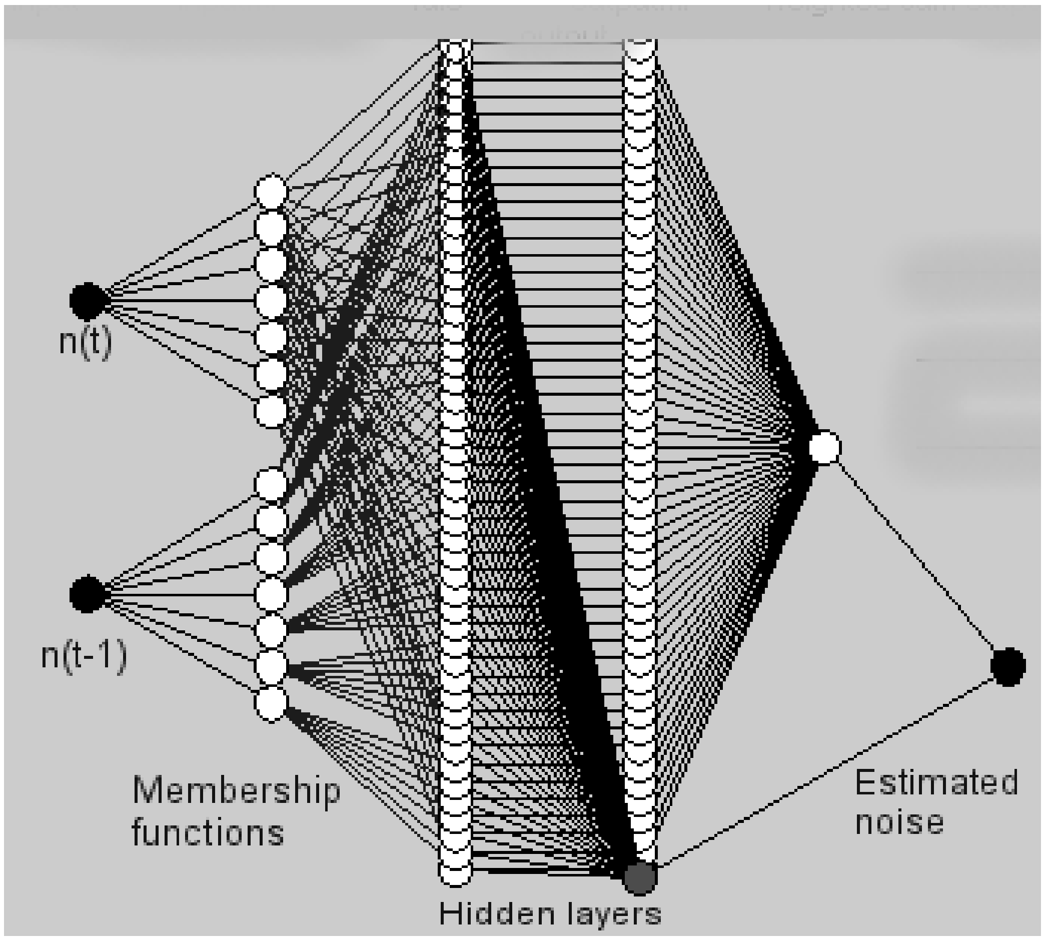


Figure 6.1.: Network architecture.

in a pratical situation, only the interference corrupts the speech signal. So we denoted $d(t)$ the distortion introduced into the speech signal.

We took the longest speech sentence from the database as the primary input of the implemented system. A sample of babble noise of the same duration is also collected. During the process, this noise sample is first delayed, and then distorted by means of the passage dynamic function reported below:

$$g[n(t), n(t-1)] = 4 \times \frac{n(t-1) \cdot sinc\, n(t)}{1 + [n(t-1)]^2}$$

whose output $d(t)$ was used to intercept the speech signal $x(t)$. In this way, we obtained the detected and measurable signal $y(t)$ expressed as

$$y(t) = x(t) + d(t).$$

	Length	SNR Improvement	MSE	AE
Average	73090	37.14 dB		
Minimum	36781	33.16 dB	54.23×10^{-6}	7.37×10^{-3}
Maximum	114800	38.45 dB	59.54×10^{-6}	$7.47\ \times10^{-3}$

Table 6.1.: Results obtained with the FIS generated with seven membership functions applied to each of the two inputs.

Since the passage dynamic function is a second order function, the system has two inputs which are $n(t)$ and $n(t-1)$ which come from babble noise. As target signal we used $y(t)$. The training process stops when the error goal reachs the 0.01 after three training epochs.

After the learning is completed, the output of ANFIS is a fuzzy inference system (FIS), with all parameters adapted to simulate the effects of the passage dynamic function over the noise waves. The final FIS is saved and applied to get an estimated $\widehat{d}(t)$ of the distorted noise. Then the estimated $\widehat{d}(t)$ is subtracted from the detected signal $y(t)$, to obtain an estimation $\widehat{x}(t)$ of the original speech signal $x(t)$. The process without new training, was applied to clean each of the 88 noisy sentences of the database described in Chapter 5.

6.4. Results

Table 6.1 summarizes the statistical results obtained after processing the 88 noisy sentences using the FIS generated. Figure 6.2 shows the initial signal to noise ratio (SNR_1) and the signal to noise ratio (SNR_2) after the 88 noisy sentences had been cleaned.

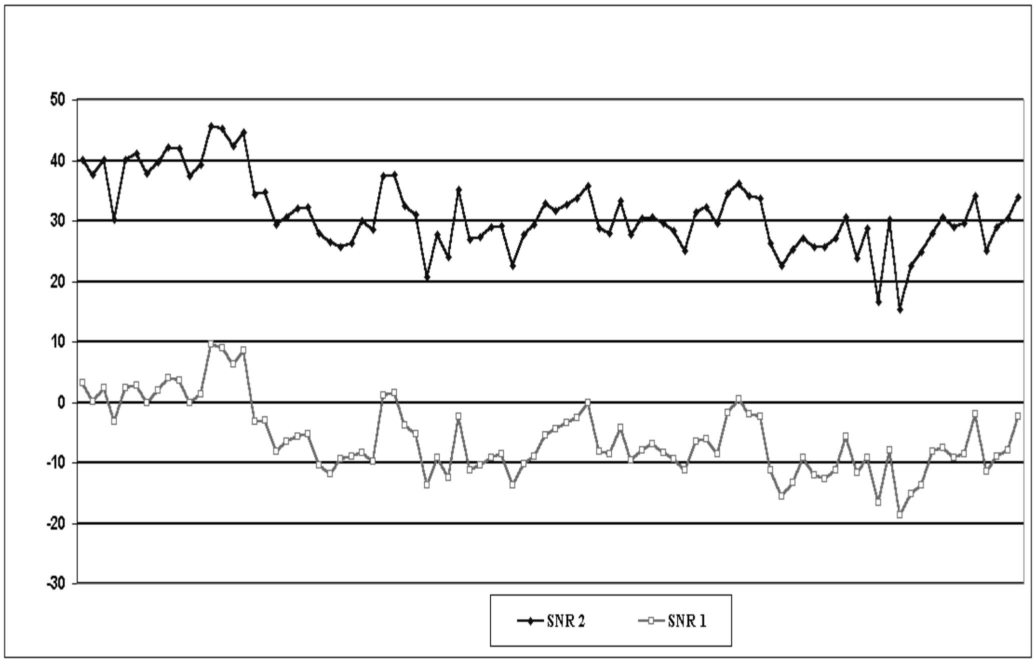


Figure 6.2.: Signal to noise ratio computed over the noisy sentences (SNR_1), and signal to noise ratio (SNR_2) computed after processing the 88 sentences through the implemented system. Each point in the graph corresponds to one sentence.

Discussions

Comparing these results to the previous ones (refer to Chapter 5), it appears that the improvement is not significant. However, to obtain them, the computation is almost in real-time.

The use of more membership functions gives a more detailed description of the inputs. Although the number of rules increases, the system requires less training epochs to produce good results. Moreover, since the training was made off-line, the computational time required for this task is not a significant factor in further processing, and the resulting cleaned sentences are of the same quality of the ones preprocessed with the system reported in Chapter 5.

Summary

The system configuration and the experimental approach reported in this chapter, have yied a fast and efficient neuro-fuzzy system, with the capability of canceling noise from *speech in a computational time close to real. Moreover, since the training was made off-line, the computational time required for this task is not a significant factor in further processing. The drawback of this system is that it requires to have a clean sample of noise, both for training and for evaluation (which is a requirement for any adaptive system for canceling noise). This fact limits its application to situations where the sample of noise can be obtained in parallel with the noisy speech.*

7. A Neuro-fuzzy System for Noise Source Identification

I have yet to see any problem, however complicated, which, when you looked at it in the right way, did not become more complicated.
– **Paul Anderson.**

7.1. Introduction

Identification is the recognition of an individual object as a unique singleton class [56]. Adaptive systems for canceling noise from speech signals proved to perform very well with respect to standard techniques [3, 14, 82] due to their adaptive ability. The drawback of these systems is that they require to have a sample of noise for evaluating. This fact limits their application to situations where the sample of noise can be obtained in parallel with the noisy speech signal. However, in practical situation the kind of noise from which the speech signal has been corrupted is unknown and therefore it becomes hard to apply adaptive noise canceling systems.

In this chapter, we proposed a neuro-fuzzy inference system able to identify the noise source by which the speech signal has been corrupted. Once the noise source is identified, it should be possible to use adaptive noise canceling system to clean noisy speech signals.

7.2. Experimental Design

We artificially corrupted the recorded speech signal, respectively with babble, car, traffic, and white noise. Such corrupted speech signals constituted the primary inputs to the implemented system. Besides, a clean sample of each type of noise was collected. The noise samples were first delayed and then distorted by means of a passage dynamic function defined as

$$f(n(t), n(t-1)) = \frac{\alpha}{[2 - n(t)][2 - n(t-1)]} + \beta$$

where $n(t)$ and $n(t-1)$ are the actual and delayed noise sample respectively. Such a function is introduced to model the nonlinear process through which the noise waves undergo before corrupting the speech signal (i.e. the distortion produced on the noisy wave by the environment). The parameters α and β are introduced to adjust the distorted noise obtained by the passage dynamic function such that it could keep the statistical features of the original noise source. In our experiment, the values of α and β were set equal to 1.0 and -0.2 respectively. These values proved to be very powerful for the problem under examination. The highly nonlinear passage dynamic function is plotted as a surface in Figure 7.1 where babble noise and its delayed version are used as inputs.

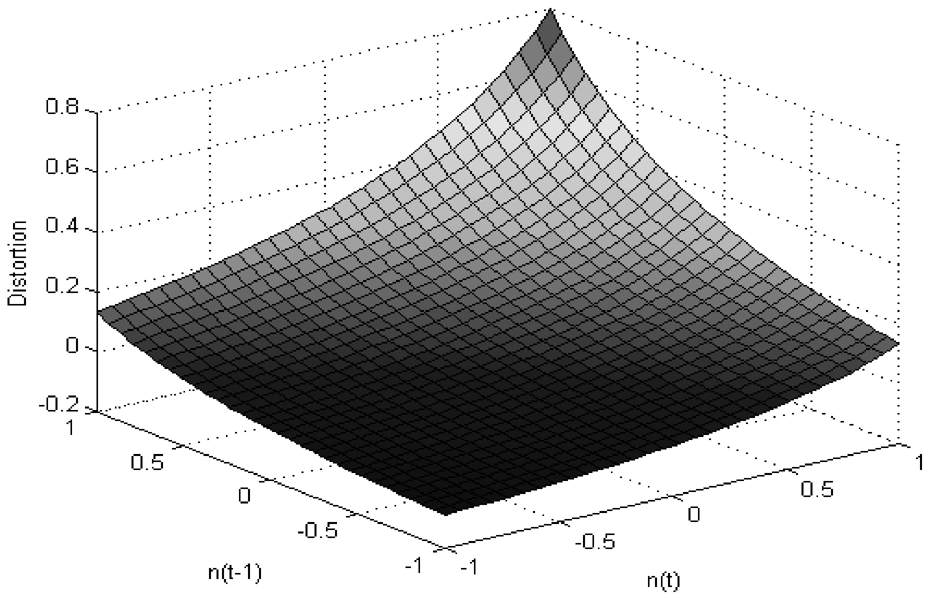


Figure 7.1.: Plot of passage dynamic function as a surface. Babble noise and its delay version are used as inputs.

Once the distortion to which the noise undergoes has been modeled, the received speech signal can be expressed as

$$y_k t) = x(t) + D_k(t)$$

where $D_k(t)$ represents the labelled noisy component of the measurable signal and k a generic index for babble, car, traffic, or white noise source. During the training, ANFIS is assumed to output an estimate $\widehat{D}_k(t)$ of the $D_k(t)$. $\widehat{D}_k(t)$ is subtracted from $y_k(t)$ producing an estimated signal $\hat{x}(t)$ of the original signal $x(t)$. The mean square error function was used to determine how much the estimated $\widehat{D}_k(t)$ was close to the noisy component $D_k(t)$ of the measured signal.

7.3. System and Algorithm Description

We are interested in the identification of babble, car, traffic and white noise in a corrupted speech signal. The system implemented for handling the problem under examination, is composed of four subsystems operating in parallel so that the output of each subsystem is independent from the others. The system takes as input the received noisy signal $y_k(t)$, the four noise sources (babble, car, traffic, white noise) and their delayed versions. The signal $y_k(t)$ is used as a target to each one of the subsystems whereas each noise source together with its delayed version are used as inputs to only one of the four FIS subsystems. Figure 7.2 shows the block-diagram of the four subsystems operating in parallel. Each fuzzy inference subsystem produced an estimate $\widehat{D}_k(t)$ of the modeled distortion $\widehat{D}_k(t)$, where $k = 1 \ldots 4$, is an index associated to babble, car, traffic and white noise respectively. Once the system has been trained, it outputs four estimates $\widehat{D_k}(t)$, $k = 1 \ldots 4$, and for each of them, the mean square error (MSE) e_k, between $\widehat{D}_k(t)$ and $D_k(t)$ is computed. The minimum among the e_k computed, identifies the index of the noise source which corrupts the clean speech signal. The criterion used for the noise source identification can be formulated in the following form:

The noise source labelled k is identified if the mean square error e_k between the distortion estimated by the subsytem k, i.e. $\widehat{D}_k(t)$, and the distortion modeled $D_k(t)$, is the minimum over all the other errrors e_i computed on the outputs $\widehat{D}_i(t)$ of each other subsystem, $i = 1 \cdots 4$.

The use of this criterion can be justified by the fact that the error between the original and the estimated sentences can be expressed as:

$$\begin{aligned}\epsilon &= x(t) - \widehat{x}(t) \\ &= x(t) - [x(t) + D_k(t) - \widehat{D}_k(t)] \\ &= x(t) - x(t) - D_k(t) + \widehat{D}_k(t) \\ &= \widehat{D}_k(t) - D_k(t).\end{aligned}$$

Figure 7.2 shows the block-diagram of the implemented system.

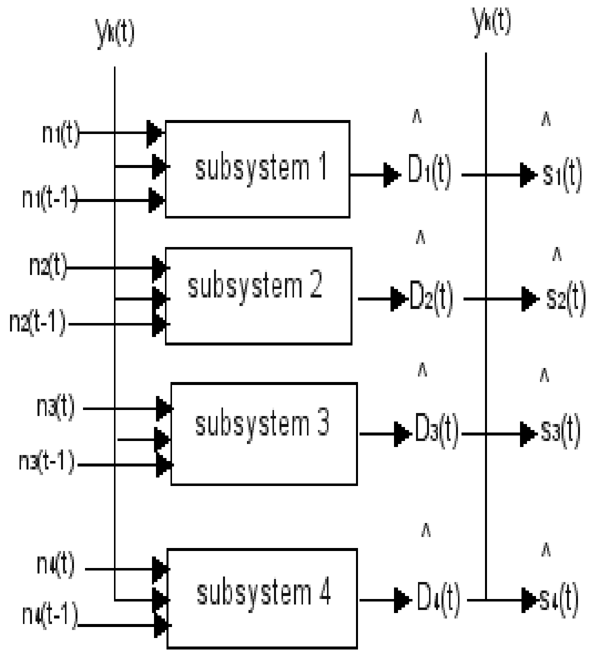


Figure 7.2.: Block-diagram of the implemented system.

7.4. System's Performance

Two experiments were carried out for testing the validity of the implemented system.

For the first one, two generalized bell functions were applied to each input of the subsystems to model the linguistic terms "*High*" and "*Low*". The full system was trained on each noisy sentence in the database. As expected the training process was computationally expensive. In Table 7.1, we report the results obtained in identifying the four types of noise sources. Each column in Table 7.1 describes the noise source, whereas in the first row is reported the percentage of correct identification, and in the second row the error percentage.

	Babble	Car	Traffic	White
Cor. Ident.	100%	91%	100%	100%
Unc. Ident.	0%	9%	0%	0%

Table 7.1.: Percentage of correct and uncorrect identification of the examined noise sources for the first experiment.

In the second experiment, we applied seven generalized bell functions to model the seven linguistic terms defined in Chapter 6, namely *Very Low, Low, More or Less Low, Medium, More or Less High, High, Very High* (refer to Chapter 6 for details). Such a system was then trained with the longest sentence in the database, which was corrupted with babble noise. The fuzzy inference system generated was saved for evaluating the four subsystems operating in parallel. Table 7.2, reports the results obtained in identifying the four types of noise sources. Each column in Table 7.1 describes the noise source, whereas the first row reports the percentage of correct identification, and the second row the error percentage.

	Babble	Car	Traffic	White
Cor. Ident.	100%	100%	100%	100%
Unc. Ident.	0%	0%	0%	0%

Table 7.2.: Percentage of correct and uncorrect identification of the examined noise sources for the second experiment.

7.5. Discussions

The use of more rules models better the linguistic values of the inputs to the fuzzy inference system. The implemented system can be used both for identifying and canceling the noise from the received speech signal. Once the distortion has been learnt, the clean signal can be obtained by subtracting the estimated distortion from the received noisy signal. A drawback of the implemented system is that for evaluating the estimated distortion, it is important, during training to use the same passage dynamic function that models the environment, otherwise the system is unable to identify the noise source correctly.

Summary

In this chapter, we proposed a neuro-fuzzy system constitued of four subsystems operating able to identify the nature of noise waves that corrupt the speech signal before canceling it from such a speech signal. We have shown that noise source identification is related to the mimimum mean square error measured between the noisy components of the measured signal and their estimates, which were the output of the implemented system.

8. Classification with Neural Networks

I have only tried, and cannot know whether sometimes I succeeded,
renouncing the fruit of action as old wisdom warns. Your gift is
the most gratifying I could receive: counting friends such as you will
add a dimension to my life until I have one.

– **Eduardo R. Caianiello.**

Introduction

The fundamental objective for pattern recognition is classification, that is the process of grouping objects together into classes according to their perceived likeness or similarities. A pattern recognition system can be considered as a two stage device. The first one is feature extraction or preprocessing phase. The second is classification.

In this chapter, two different experiments were carried out. We classified the English stops and four noise sources using respectively Time Delay Neural Network (TDNN) and Recurrent Neural Network (RNN) architectures. The preprocessing algorithms are described.

8.1. English Stops Classification with Time Delay Neural Networks

The work in the area of phoneme recognition using neural networks started with Waibel *et al.* [80] and was justified by the their potential of providing massive parallelism, adaptation and new algorithmic approaches to the problem. The initial studies demonstrated that multilayer neural nets, with time delay units, provide excellent discriminations on small samples of pre-segmented words (consonants and vowels) which are otherwise hard to discriminate. However, to obtain such results they used an elaborated network architecture and a specific set of data based on utterances produced by a small number of speakers. Other attempts to realize similar performance using a different database (like TIMIT) and even phonetic units (like vowels) which are much more easy to discriminate than consonants did not give the same results (see [5] for a survey on phoneme recognition using several preprocessing methods and neural network algorithms). Due to these difficulties researchers tried to recognize speech signals which are larger and less variable than phonemes (like words). For these speech signals it is easier to find some stable acoustic attributes which can be used to allow a good learning and classification process (see [75]). Phoneme recognition still remains a difficult problem to deal with. The present section approaches the phoneme recognition problem using data extracted from the multispeakers continuous TIMIT database. It exploits a new preprocessing algorithm based on the RASTA-PLP algorithm (see [33, 35]) and a neural classification approach based on TDNN. Moreover, this work also exploits the knowledge of the acoustic and perceptual features which significantly characterize the stop consonants (see the following subsection). Combining the three aspects mentioned above, we obtained classification percentages on the test data which are better than those obtained in the previous works of Zue *et al.*, Bengio *et al.*, and Flammia [6, 27, 90], which also tried to recognize stop consonants extracted from TIMIT. We will show how the modification of some parameters in the preprocessing RASTA-PLP algorithm will allows to capture acoustic and perceptual stop's features, and obtain significant improvements in their recognition. Moreover, we will report experimental results which allow to define an upper bound on the number of neurons in the first hidden layer of a TDNN. Above this bound the net performance decreases and the computational time for its training increases. Furthermore, we propose a learning rate function which avoids the trial and error processes needed to establish a good learning rate value to train the net on the current data.

8.1.1. Speech Database

The consonants [b, d, g, p, t, k] used to train and test the net were extracted from TIMIT-NIST[1] database. All data files in the two directories (TIMIT/TRAIN and

[1]National Institute of Standards and Technology.

Phoneme	Training	Testing
$[b]$	183	176
$[d]$	300	265
$[g]$	166	157
$[p]$	211	188
$[t]$	329	326
$[k]$	389	352
$Total$	1578	1464

Table 8.1.: Size of the training and testing data

TIMIT/TEST) are considered. The TIMIT-NIST database is composed of English sentences produced by speakers from different USA regions. Each speaker read ten English sentences. Each sentence was labeled phoneme by phoneme. A total of 730 different sentences were used to collect the consonants for the training and the testing set.

Our data (the stop consonants [b, d, g, p, t, k]) are extracted from such sentences. The sentences are different from one speaker to another. The training data used were produced by 38 speakers (24 males and 14 females). The testing data were produced by 35 speakers (22 males and 13 females). The training data were produced by speakers from the same USA region (dr1 in TIMIT). The testing data were produced by speakers from different USA regions (dr1-dr2-dr3 in TIMIT) and then, with a different accent. The sentences from which the consonants were extracted were different, both in the training and the testing data. Table 8.1 summarizes the size of the training and testing data for each stop consonant.

8.1.2. Preprocessing Phase

Feature extraction, also known as preprocessing phase is rarely a trivial process and is often fundamental for any recognition problem. For most applications, it is important to transform the data into some representations before using them as input to some classification sytem, such as neural networks. Let us briefly review three feature extraction techniques in speech signal.

Linear prediction (LP) [59] analysis is a time-domain technique which attemps to predict "as well as possible" a speech sample through a linear combination of several previous signal samples. As the order of the LP model increases, more details on the power spectrum of speech can be approximated [33].

Perceptual Linear Prediction (PLP) [34] analysis uses concepts from psychophysics of hearing in order to derive an estimate of the auditory spectrum [33].

The RelAtive SpecTrAl (RASTA) technique was originally developed as a purely engineering technique for dealing with fixed or slowly varying non-linguistic components of speech features. Since the rate of extra-linguistic changes is often outside the typical rate of change of linguistic components, Hermansky et al. [34] proposed that filtering the temporal trajectories of speech parameters might alleviate the extra-lingusitic spectral components from the speech representation.

The speech signal was preprocessed using the RASTA speech processing algorithm due to its better performance over linear prediction (LP) and perceptual linear prediction (PLP) algorithms [35]. RASTA-PLP adds a spectral band-pass operation to PLP analysis in order to obtain acoustic features which are robust to distortions. Each speech segment was processed through several steps which include critical-band spectral resolution, equal loudness curve, intensity loudness conversion, inverse Discrete Fourier Transform, autoregressive coefficients, all-pole model.

We modified some parameter values of this algorithm in order to capture some of the invariant and transitional cues of the stop consonants. Indeed, the speech signal was sampled at 16 kHz and was weighted by a Hamming window 10 ms long and each frame of a phonetic segment overlapped with the precedent ones by 5 ms. This analysis provided nine acoustic features (8 coefficients plus gain) at the rate of 100 frames/sec. All stop consonants [b, d, g, p, t, k] were processed in the same way.

The modification of the preprocessing parameter values can be justified taking into account the acoustic features of stop consonants. Blumstein and Stevens [8], through a set of perceptual experiments, suggested that useful information about the stop consonant identity can be found mostly in the short time interval that follows the release. For this reason we used an analysis window and an overlapping step both shorter than their original values.

We used RASTA-PLP as preprocessing algorithm because, as said above, it uses concepts from psychophysics of hearing in order to derive an estimate of the auditory spectrum.

8.1.3. Acoustic Features of Stop Consonants

Which are the cues that our auditory apparatus uses to discriminate among speech sounds? Perceptual experiments, performed by Cole and Scott [11], suggested that our auditory apparatus uses *invariant* and *transitional* cues. An *invariant* cue, as it was defined by Cole and Scott, is an acoustic cue which accompanies a particular phoneme in any vowel environment. It is an acoustic feature that is present whenever a particular phoneme is produced during speech, and provides information about its phonetic identity. *Transitional* cues were defined as acoustic cues which accompany a particular phoneme in a specific phonetic environment. Both are used by listeners to discriminate among phonemes. The identification of a particular stop involves identification of either invariant or transitional cues, depending upon the position of the consonant in the syllable and the position of the syllable in the utterance.

The major cue which distinguishes stop consonants from the others in ongoing speech is silence. Stop consonants are always preceded by a closure interval prior to the release, and while voicing may continue throughout this interval in some environments there is a period of relative silence along the remaining of the energy spectrum which lasts at least 10-20 msec. The silent interval is an essential cue for the identification of a stop consonant (see [11]). Stop consonants are released with an explosive burst when produced before a vowel in a monosyllable. For voiced [b, d, g] the burst may be followed by a short open interval (5-10 msec in running speech), while for voiceless stops [p, t, k] the burst is followed (in initial syllable position) by a period of aspiration. The explosive burst is a transitional cue for stop consonants and it might help to discriminate them in some vowel environments (see [11]). Defined as the time interval between the burst onset and the voicing, Voice Onset Time (VOT) reliably discriminates between voiced and voiceless stops (see [57]) in initial position of isolated words and in short sentences. However, in other phonetic environments, VOT values were found to be compressed for both voiced and voiceless stops, and the separation between the two categories was generally less sharp (see [58]). The RASTA-PLP algorithm is capable of extracting such cues from the raw speech signal because it derives an estimate of the auditory spectrum, where such cues are enhanced. Cole and Scott's experiments also showed that some stop consonants occurring in the initial word position cannot be substituted by other

Cons./Epochs	200	1000	1800	2600
[b]	67.1	73.2	69.1	65.0
[d]	59.0	56.3	63.1	70.0
[g]	32.1	31.3	33.0	35.0
[p]	34.1	45.1	47.4	53.1
[t]	80.0	84.2	84.2	81.5
[k]	62.2	67.1	67.0	66.3
Overall	46.0	59.5	60.6	61.6

Table 8.2.: Net classification percentages (on the training data) for each consonant using the original RASTA-PLP algorithm and a 9–24–6–6 TDNN net architecture.

consonants if the perception has to be left unchanged, except for [g] which could be replaced by [b] whenever it occurs. When a stop consonant occurs in any other position, all the voiced stops could be replaced by [d] whereas all the voiceless stops could be replaced by [t] without modifying the perception of the word or the sentence in which they are embedded. The results of these experiments suggested that, in running speech, some speech segments could be changed by production errors or by phonological rules without changing the listener's perception. Moreover, the perception of [b, d] is much more stable than the perception of [g], and the perception of [t, k] is much more stable than the perception of [p]. Since neural networks can only extract features from their inputs in order to realize the recognition process, the phonemes that undergo changes like those mentioned above are not classified so well as the others. This means that extracting the acoustic features from the input patterns, the net will better generalize for [b, d] and [t, k] than for [g] and [p]. Our experimental results, reported through the Tables 8.2, 8.3, 8.4 confirm this hypothesis. Net recognition percentages for [g] and [p] are always lower than those for [b, d, t, k].

8.1.4. The Time Dependent Neural Classifier

To perform the classification task, we used Time-Delay Neural Networks, which have turned out to be very suitable for phoneme recognition [79]. TDNNs try to capture the relationships among sequences of acoustical events under translation in the time of the signal window that is being examinated.

Our net is constituted of an input layer of nine components, a first hidden layer (in

Cons./Epochs	200	1000	1800	2600
[*b*]	74.3	73.2	69.4	73.4
[*d*]	76.0	81.3	79.1	76.0
[*g*]	54.0	45.2	49.1	58.2
[*p*]	51.0	63.0	65.0	64.2
[*t*]	86.3	89.4	88.2	88.5
[*k*]	79.4	77.1	84.1	82.0
Overall	70.2	72.2	72.5	73.3

Table 8.3.: Net classification percentages on the training data for each consonant (reported on the columns as a function of the number of epochs) for each consonant using the modified RASTA-PLP algorithm and a 9–24–6–6 TDNN net architecture.

Consonants	*Classification Percentages*	
	Training data	*Testing data*
[*b*]	93.7	92.9
[*d*]	92.3	91.8
[*g*]	77.5	92.4
[*p*]	78.4	80.3
[*t*]	92.4	90.8
[*k*]	94.4	94.2
Overall	88.1	90.4

Table 8.4.: Classification percentages on the training and testing data respectively with a 9-256-6-6 TDNN net architecture.

which the number of neurons was varied through the set of the experiments performed), a second hidden layer of six neurons, and an output layer of six neurons. The basic unit of TDNNs is a neuron that has been modified by introducing delays for each connection. The input layer is activated by a vector of nine components. The input action on the first hidden layer is seen through two component delay units, which were passed over the input components. Likewise, in the second-hidden layer, each neuron sees the first hidden layer through two component delay units, stepped by one frame at a time. Each of the output units, corresponding to [b], [d], [g], [p], [t], [k], is connected to the second hidden layer. The output layer performs, over time, an integration, of the nodes' outputs in the second hidden layer. The delays are chosen short in order to take into account the way we processed the input data (short window analysis). This net operates similarly to that described by Waibel *et al.* [79, 80, 81] . However, the net is simpler in its structure compared to Waibel's net. Indeed, the net input is a vector of nine components while the Waibel net input was a matrix of 16×15 components. Moreover, to discriminate [b, d, g, p, t, k] Waibel [81] used two TDNNs, one for [b, d, g] and another for [p, t, k]. Classification performance was very good in such case, but the resulting net architecture was greatly complicated.

Our network was trained using an on-line back-propagation algorithm [73]. Bengio [5] showed that on-line learning is most appropriate for automatic speech recognition applications. The weights are updated after each pattern presentation without using the momentum term. The momentum does not seem to be useful in on-line learning algorithms [5, 10]. The learning rate, after a series of trial and error processes was set to $\eta = 0.03$. This value gave the better net performances in terms of computational time and classification percentage.

8.1.5. Recognition Experiments

A set of experiments was carried out in order to test different hypotheses. In all the experiments performed the input data were normalized so as to span over [-1 1].

Changes in the Preprocessing Parameter Values

The first experiment was carried out in order to test if our modifications of some RASTA-PLP parameters allowed to capture significant information about the phonemes we

Cons./Epochs	200	1000	1800	2600
[*b*]	67.1	73.2	69.1	65.0
[*d*]	59.0	56.3	63.1	70.0
[*g*]	32.1	31.3	33.0	35.0
[*p*]	34.1	45.1	47.4	53.1
[*t*]	80.0	84.2	84.2	81.5
[*k*]	62.2	67.1	67.0	66.3
Overall	46.0	59.5	60.6	61.6

Table 8.5.: Net classification percentages (on the training data) for each consonant using the original RASTA-PLP algorithm and a 9–24–6–6 TDNN architecture.

wanted to recognize. We run the net with an architecture of 9-24-6-6 (i.e. 9 units in the input layer, 24 units in the first hidden layer, 6 units in the second hidden layer, and 6 units in the output layer) for several epochs. Tables 8.5 and 8.6 report respectively the net performance with the original RASTA-PLP (i.e. with a sampling rate of 20 kHz, an analysis window duration of 20 msec, and a window step of 10 msec) and the modified RASTA-PLP algorithm (i.e. with a sampling rate of 16 kHz, an analysis window duration of 10 msec, and a window step of 5 msec). It is interesting to note that after a certain number of epochs, the overall net performance did not improve significantly even though the number of epochs was increased. For example, in Table 8.5 for [b, d, g] and in Table 8.6 for [p, t, k] the correct classification percentage after 1800 epochs is better than the correct classification percentage after 2600 epochs. However, it is evident from tables 8.5, and 8.6 that whathever was the number of epochs, the correct classification percentage always improved when the modified RASTA-PLP algorithm was used. Moreover, these results confirm that the way the data are processed plays an important role on the subsequent net performance [5]. As a consequence of these results, all the training and testing data for the subsequent experiments were processed with the modified RASTA-PLP algorithm.

Optimization of the Network Architecture

It is largely accepted that the net performance improves when the number of hidden neurons increases [12, 29, 38]. There is, however, an upper bound on the hidden neuron number which depends on the current task to which the net is devoted to [63]. The

Cons./Epochs	200	1000	1800	2600
$[b]$	74.3	73.2	69.4	73.4
$[d]$	76.0	81.3	79.1	76.0
$[g]$	54.0	45.2	49.1	58.2
$[p]$	51.0	63.0	65.0	64.2
$[t]$	86.3	89.4	88.2	88.5
$[k]$	79.4	77.4	84.1	82.0
Overall	70.2	72.2	72.5	73.7

Table 8.6.: Net classification percentages on the training data using the modified RAST-PLP algorithm and a 9–24–6–6 net architecture.

net gives, for such a bound, the minimum generalization error (the generalization error is the net error on an independent test set). Our goal in carrying out this set of experiments is to propose a net architecture that allows us to define (for our current task) an upper bound on the number of neurons in the first hidden layer. This bound should give the best compromise between the net's classification performance and the computational training time. We expect that the net performance with a number of hidden neurons above this bound should not pay for the increased computational training time. Therefore, in our experiments we only varied the number of neurons in the first hidden layer. Table 8.7 reports the overall net performance for this set of experiments. All the nets were trained on 1600 epochs. As it is possible to see, the net performance improved as the number of hidden neurons in the first hidden layer increased, together with the computational time required for the training. Our experimental results (see Table 8.7) also showed that the classification percentages reached a maximum value for a certain number of hidden neurons, and then started to decrease. In order to determine the performance function which fits our experimental data (that is the performance as a function of the number of hidden in the first hidden layer), we used the interpolating theory and Lagrange's formula [70] on our experimental data. Figure 8.1 shows that a 9–256–6–6 net architecture is optimal for our current task.

Finally, we used such a net architecture in the training and testing phase. The net was trained for 1600 epochs. The results are reported in Table 8.8 training and testing data, and for each consonant. Our generalization error on the independent test data set was 9.6%, which gave a significant improvement compared to that reported by Zue

Layer 1	24	34	64	128	256	350
Overall	72.1	72.8	81.5	85.0	88.1	85.6

Table 8.7.: Net classification percentages (on the training data) averaged over all the consonants for TDNN net architectures with a different number of hidden neurons in the first layer.

Consonants	*Classification Percentages*	
	Training data	*Testing data*
[*b*]	93.7	92.9
[*d*]	92.3	91.8
[*g*]	77.5	92.4
[*p*]	78.4	80.3
[*t*]	92.4	90.8
[*k*]	94.4	94.2
Overall	88.1	90.4

Table 8.8.: Classification percentages on the training and testing data respectively with a 9–256–6–6 TDNN net architecture.

et al., Bengio *et al.*, and Flammia [6, 27, 90]. These authors obtained a generalization error of 35%, 30.7 %, and 24.9% respectively using more sophisticated preprocessing features and more complicated network architectures. It is important to note that even though the number of epochs was increased the net performance did not improve (see Tables 8.2,and 8.3) due to the overtraining phenomenon. Indeed, it has been shown (see [78]) that such phenomenon appears after a certain number of epochs depending on the training data and the net architecture. The data reported in Tables 8.2 and 8.3 confirm this hypothesis. In Table 8.2 the net performance decreased after 2600 epochs whereas in Table 8.3 the training data have been changed and the net performance still increased after 2600 epochs.

Figure 8.1 shows the plot of the net performance as a function the number of neurons in the first hidden layer.

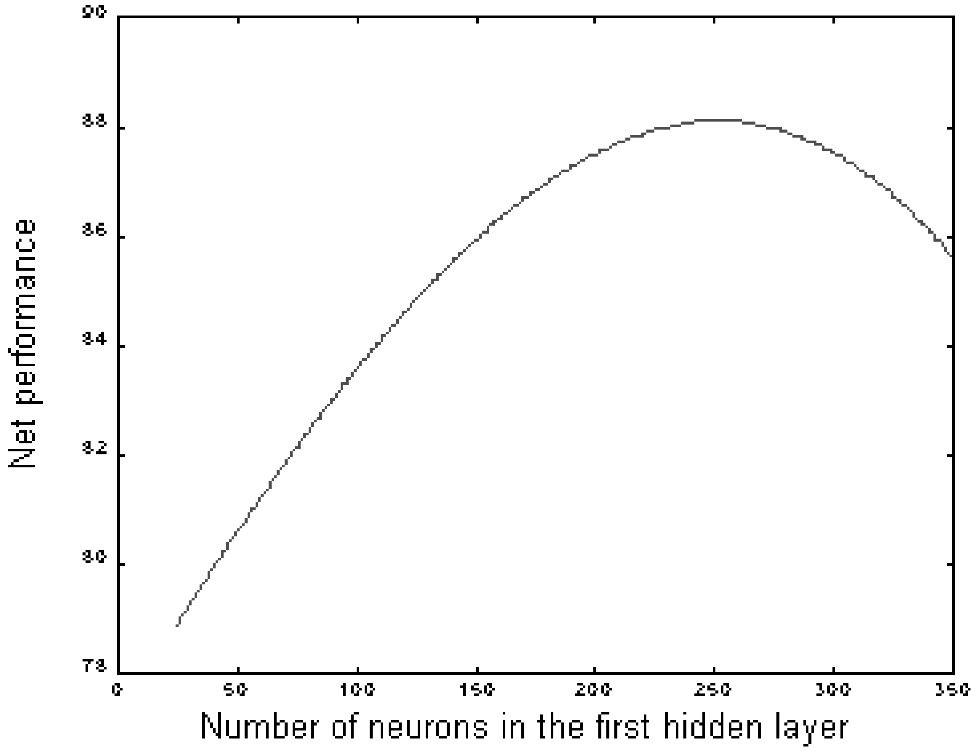


Figure 8.1.: Net performance as a function versus number of neurons in the hidden layer.

Optimal Learning Rate

Once we found the right way to preprocess the data and the right number of hidden neurons to achieve a good phoneme recognition task, we performed another set of experiments in order to test a general way to set up the learning rate value. The subsequent experiments were performed using the optimal net architecture found in the above section and data processed with the modified RASTA-PLP algorithm.

Generally, in the backpropagation training algorithm, the learning rate value is randomly changed until, through trial and error processes, an appropriate value for the current task was found. This is a hard process which relies only on the researchers' experience. We tried a general way to set up the learning rate value defining it as a function of the net output error on the current input. For a better understanding, let $\beta = \max_k \delta_k^{(p)}$ be valued on all output neurons, where $\delta_k^{(p)} = y_k(1 - y_k)(p_k - y_k), \quad k = 1, \ldots, 6$ is the error on the current pattern p computed over the output node k; p_k is the k-th component of the input pattern; y_k is the output of the node k. Then the learning rate $\eta^{(p)}$ for the current input p was set equal to $\eta^{(p)} = e^{-\beta}$ and it was changed after each epoch. The experimental data showed that with such a learning rate, the convergence was never reached and local minimum problems appeared. Mathematically, this effect could be explained by the fact that large and small errors appeared randomly during the training and cannot be predictable. Consequently, the MSE undergoes great oscillations from one training cycle to another because of the negative β value.

In the second experiment we define the learning rate as a decreasing function of the number of current epochs; i.e.

$$\eta = \frac{1}{M}\left(1 - \sum_{j=1}^{Ne} \frac{1}{2^j}\right) e^{-|\beta|}$$

where M is a free parameter. It could be used to decrease the initial learning rate value when it is too high for the classification task to which the net is devoted. In our case M is equal to 1. Ne is the current number of epochs (Ne defines the learning rate as a function of the current number of epochs), and $\beta = \max_k \delta_k^{(p)}$ was valued on all the output neurons. With such a learning rate the net MSE value after 1600 epochs was lower than that obtained with $\eta = 0.03$. However, the net classification performance was roughly the same (the classification percentage over all the consonants and after 1600 epochs was 88.06%) and the computational time increased. Nevertheless, such a choice avoids trial and error processes in defining the learning rate value and speeds up the net convergence.

8.2. Noise Sources Classification with Recurrent Neural Networks

In this section, we present a classification of noisy sentences (four different types of noise with recorded speech sentences) with recurrent neural networks as classifiers.

8.2.1. Generalization Process

One of the advantages of neural networks is their ability to generalize. This means that a trained neural network could classifiy data from the same class as the learning data that it has never seen before. The idea to split the data into three parts (see the next subsection) is to reach the best generalization. The generalization process is complex, subtle, often partial and rarely straightforward. One technique is to stop the learning at the minimum of the validation set error. After the learning phase, the network is checked with a third data set, the validation data. There are critical effects of corpus size, corpus structure and the time course of learning and many open questions remain [17].

8.2.2. Database Description

For this purpose, we artificially corrupted some selected sentences from TIMIT database [30] and Italian database. The resulting clean and noisy sentences are divided into three categories based on the generalization technique [77]. These categories are training, validation and testing sets. The training set is used to train a neural network. The validation set is used to determine the performance of a neural network on patterns that are not trained during learning. A test set is used for finally checking the overall performance of a neural net. The size of different categories of data used are summarized in Tables 8.9, 8.10, 8.11.

Sentences	Italian	English	Total
Clean	8	492	510
Babble	8	492	510
Car	8	492	510
Traffic	8	492	510
White	8	492	510

Table 8.9.: Training data set.

Sentences	Italian	English	Total
Clean	4	460	464
Babble	4	460	464
Car	4	460	464
Traffic	4	460	464
White	4	460	464

Table 8.10.: Validation data set.

8.2.3. Description of the Preprocessing Technique

We implemented an algorithm that derived from Linear Predictor Coding (LPC) theory [59]. The signal s_n is assumed to give as a linear combination of past values and some input u_n

$$s_n = -\sum_{j=1}^{p} a_k s_{n-k} + G u_n$$

where G is a gain factor, a_k the predictor coefficients and p the model order. The LPC algorithm finds the predictor coefficients a_k of an $p-$th order forward linear predictor

Sentences	Italian	English	Total
Clean	4	398	402
Babble	4	398	402
Car	4	398	402
Traffic	4	398	402
White	4	398	402

Table 8.11.: Testing data set.

and the gain factor G such that the sum of the squares of the error

$$e_n = s_n + \sum_{j=1}^{p} a_k s_{n-k}$$

is minimized.

The speech signal is segmented into 6 frames by applying a 200 msec as window's lenght every 100 msec. From each frame, 11 predictor coefficients plus the gain factor are extracted. The resulting coefficients are computed on a signal of 700 msec as duration and are arranged into a single observation vector of 72 coefficients.

Since the clean speech database is composed of 1376 sentences that we corrupted artificially with babble, car, traffic and white noise respectively, that leads to 2550 input patterns for the training set, 2320 input patterns for the validation set, and 2010 input patterns for the testing set.

8.2.4. Recurrent Neural Networks

Recurrent neural networks are logical candidates when identifying a nonlinear dynamical process [68, 87]. Such networks are becoming more and more attractive since they have capabilities to store information for later use and their ability to deal with time-varying input or output through their own natural temporal [54]. Many learning algorithms have been proposed in literature for fully recurrent networks [86] and partially recurrent networks [16]. Partially recurrent networks are basically back-propagation networks with proper feedback. The main function of partially recurrent networks is to deal with time varying explicitly. The recurrency in recurrent networks allows the network to remember cues from the recent past. In these networks, the nodes receiving feedback signals are called *context units*. At time t, the contex units have signals coming from part of the network state at time (t-1). Since partially recurrent networks are basically multilayer feedforward networks, feedback links can come from either the output nodes or the hidden layers nodes, and the destinations of the context units can be either input nodes or hidden nodes. In the following classification, we consider the Jordan sequential network, and the Elman recurrent network [16]. We trained the network on a workstation using the Stugartt Neural Network Simulator [77].

The Jordan sequential network is realized by adding recurrent links from the network's output to a context layer, and from the context units to themselves. Context units copy the activations of output nodes from the previous time step through the feedback links with unit weights. This model has some drawbacks. For instance, with sequences of increasing length, the network encounters difficulty in discriminating on the basis of the first cues presented [2].

Elman's recurrent network is a two layers' backpropagation network with the addition of a feedback connections from the output of the hidden layer to its input. This feedback path allows the Elman network to learn, recognize and generate temporal patterns as well as spatial patterns [16]. Elman network requires that the hidden layer must have enough neurons to perform well.

For both architectures used, we fixed the learning rate to 0.002 and the number of training epochs to 2000.

Classes	Jordan's Network			Elman's Network		
	Tra.	Val.	Tst.	Tra.	Val.	Tst
Clean	96.5	90.3	84.8	93.3	84.3	83.1
Babble	96.1	93.1	92.0	90.4	84.3	84.1
Car	99.0	95.9	96.0	97.3	92.0	93.5
Traffic	99.5	99.6	98.5	98.2	98.1	95.8
White	99.4	99.4	98.0	99.2	99.4	97.8

Table 8.12.: Experimental results with Elman's and Jordan's network architectures.

8.2.5. Results Analysis

As the results showed in Table 8.12, Jordan's network and Elman's network architectures are good classifiers. It also appears that the preprocessing technique used is optimal for the classification problem under examination, but neuro fuzzy systems can perform better.

Summary

In this chapter, we report on two classification problems which can be related with the work up to now reported since, to solve them, we still used learning techniques, even though without using fuzzy logic.

The results of these two works suggested that when the data are enough and well preprocessed it is not necessary to use a neuro-fuzzy system (which is more complex than a neural network, to obtain good results).

9. Conclusion and Further Topics

We believe that the new role of Science is to promote a more effective collaboration between East and West, and to contribute in a direct way to fill the ever-increasing gap between North and South.
– **Antonino Zichichi.**

9.1. Concluding Remarks

In this dissertation we showed that adaptive noise filtering from a speech signal with neuro-fuzzy systems is a good technique compared to those based on time-domain and frequency-domain since it combines the potentialities of neural networks to learn from example, and the capabitilies of fuzzy logic to model human concepts and thoughts.

The implemented systems have the advantage of canceling noise from the speech signal in almost real computational time. Moreover, the requirement of knowing the noise source (requirement of any adaptive noise canceling system) is overcome by the system (composed of four subsystems operating in parallel) we proposed in Chapter 7, since such a system is able to identify the nature of noise waves that corrupt the speech signal when recorded noise source samples are available.

We proposed some modelings of the environments with passage dynamics that are the mathematical descriptions of channels through which the noise waves undergo before corrupting the speech signal.

We also showed through classification experiments that neural networks are good classifiers when the input data are enough and well preprocessed.

9.2. Further Topics

This work provides a foundation for future expansion of studying the passage dynamics that model the environment through which the noise waves undergo before corrupting the speech signal. As underlined in Chapters 6 and 7, even though we have designed a fast neuro-fuzzy system for noise cancellation from speech signal, the fuzzy inference system generated takes into account during learning the passage dynamic function used. Its performance changes if the passage dynamic function changes. An interesting future work on this aspect should be to find a way to overcome this limitation. A possible, but not obvious solution, could be the introduction of wavelets into the implemented system for modeling the passage dynamics. Further work should be done on this topic.

Another drawback of this system is that we should know the noise source which corrupted the signal. We tried to overcome this limitation, implementing a system where different models of noise sources are learned (see Chapter 7). However in real world applications, it should be difficult to model the noise source appropriately, or to have a sample of it available. Therefore, further work should be done to avoid the dependence of the implemented system on the noisy environment.

Last but not least, it should be observed that the implemented systems were tested intensively on speech sentences artificially corrupted. This creates in some way a linear relationship between the corrupted signal and the noise source. In the real world, we only have a corrupted signal where the distortion created by the noise could be nonlinear. Assuming we may know this nonlinear distortion in advance, it should be interesting to test the system on data not corrupted artificially. This is an ongoing work which we are performing for forensic data analysis.

Bibliography

[1] **Abma R. L.,** *Least-Squares Separation of Signal and Noise Using Multimensional Filters,* Ph.D. Dissertation, Stanford University, 1995.

[2] **Anderson S. et al.,** *Dynamic Speech Categorization with Recurrent Networks,* in Touretzky D. S., Hinton G. E., and Sejnowski T. J., eds. Proc. Connectionist Models Summer School, pp. 398–406, San Mateo, CA, Morgan Kaufmann, 1998.

[3] **Avendano C.,** *Temporal Processing of Speech in a Time-Feature Space*, Ph.D. Dissertation, Oregon Graduate Institute of Science & Technology, April, 1997.

[4] **Beale R. and Jackson T.,** *Neural Computing: An Introduction,* Institute of Physics Publishing, Bristol and Philadelphia, IOP Publishing Ltd, 1992.

[5] **Bengio Y.,** *Artificial Neural Networks and their Application to Sequence Recognition,* Ph. D Thesis, McGill University, Montreal, Canada, 1991.

[6] **Bengio Y., et al.,** *Phonetically Motivated Acoustic Parameters for Continuous Speech Recognition using Artificial Neural Networks,* in Proceedings of EuroSpeech, Genova, Italy, pp. 551–554, 1991.

[7] **Bishop M. C.,** *Neural Networks for Pattern Recognition,* Oxford University Press Inc., New York, 1995.

[8] **Blumstein S. E., Stevens K. N.,** *Perceptual Invariance and Onset Spectra for the Stop Consonants in Different Vowel Environments,* in JASA, 67(2), pp. 648–662, 1980.

[9] **Boll S. F.,** *Suppression of Acoustic Noise in Speech using Spectral Subtraction*, IEEE ASSP - 27, pp. 113–120, April, 1979.

[10] **Bottou L., et al.** *Speaker-Independent Isolated Digit Recognition: Multilayer Perceptron vs. Dynamic Time Warping,* in Neural Networks, 3(4), pp. 453–456, 1990.

[11] **Cole R. A., Scott B.,** *Toward a Theory on Speech Perception,* in Psychologicial Review, 81(4), pp. 348–374, 1974.

[12] **Cybenko G.,** *Approximation by Superpositions of a Sigmoidal Function,* Technical Report n° 856, Urbana, IL: University of Illinois Urbana-Champaign, Department of Electrical and Computer Engineering, 1998.

[13] **Dendrinos M. et al.,** *Speech Enhancement from Noise: A Regenerative Approach,* Speech Comunication, vol. 10, n°2, pp. 45–57, February, 1991.

[14] **Doclo S., et al.,** *A Novel Iterative Signal Enhancement Algorithm for Noise Reduction in Speech,* in Proceedings of the 5th International Conference on Spoken Language Processing, ICSLP'98, Sydney, Australia, pp. 1435–1438, December 1-4, 1998.

[15] **Ellacott S. W.,** *Aspect of the Numerical Analysis of Neural Nwtworks,* Acta Numerica, 3, pp. 145–202, 1994.

[16] **Elman J. L.,** *Finding Structure in Time,* Cognitive Science, 14, pp. 179–211, 1990.

[17] **Elman J.,** *Generalization, Simple Recurrent Networks, and the Emergence of Structure,* in Genrsbacher M. A. and Derry S. (eds.), Proceedings of the Annual Conference of the Cognitive Science Society, Mahway, NJ, Lawrence Erlbaum Associates, 1998.

[18] **Ephraim Y.,** *Statistical-Model based Speech Enhancement System*, in Proceedings of the IEEE vol. 80, n°10, pp. 1526–1555, October, 1992.

[19] **Esposito A., Ezin C. E., Reyes-Garcia A. C.,** *A Nonlinear Adaptive System to Cancel Noise from Speech,* in Advances in Fuzzy Systems and Intelligent Technologies, F. Masulli, R. Parenti and G. Pasi (eds.), Shaker Publishing 2000, Genoa, Italy, pp. 129–138, June 28-29, 1999.

[20] **Esposito A., Ezin C. E., Reyes-Garcia A. C.,** *Speech Noise Cancellation Based on a Neuro-Fuzzy System: Experimental Results,* in Proceedings of IEEE International Workshop on Signal Processing, WISP'99, pp. 342-347, Budapest, Hungary, September 4-7, 1999, ISBN 963 420 607 7.

[21] **Esposito A., Ezin C. E., Reyes-Garcia A. C.,** *Speech Noise Cancellation Based on a Neuro-Fuzzy System: Further Improvements,* in the Journal of Advanced Computational Intelligence, edited by Prof. Kaoru Hirota in Tokyo, Japan, 2000.

[22] **Esposito A., Ezin C. E., Reyes-Garcia A. C.,** *Designing a Fast Neuro-Fuzzy System for Speech Noise Cancellation,* in Proceedings of the Mexican International Conference on Artificial Intelligence MICAI-2000, Acapulco, Mexico, pp. 482–492, April 10-14, 2000.

[23] **Esposito A., Ezin C. E., Reyes-Garcia A. C.,** *A Neuro-Fuzzy System for Source Identification,* in Proceedings of International Workshop, Speech and Computer, SPECOM, St.-Petersburg, Russia, pp. 53–56, September, 2000.

[24] **Esposito A., Ezin C. E.,** *Phoneme Classification using a Rasta-PLP Preprocessing Algorithm and a Time Delay Neural Network: Performance Studies,* in Proceedings of Workshop Italiano sui Reti Neurali, WIRN'98 edited by M. Marinaro and R. Tagliaferri, pp. 207–217, Springer-Verlag Publisher, 1998.

[25] **Esposito A., Ezin C. E.,** *A Study on the Optimal Number of Hidden Neurons and the Best Preprocessing Parameter Values for Recognizing Stop Consonants,* in Proceedings of Colloque Africain sur la Recherche en Informatique, CARI'98, edited by M. Tchuente INRIA-Universitaires de Dakar Press, pp. 365–375, October 12-15, 1998.

[26] **Esposito A., Ezin C. E., Ceccarelli M.,** *Preprocessing and Neural Classification of the English Stops [b, d, g, p, t, k],* in Proceedings of the International Conference on Spoken Language Processing, ICSLP'96 edited by H. T. Bunnell and W. Idsardi, vol. 2, pp. 1249–1252, October 3-6, 1996.

[27] **Flammia G.,** *Speaker Independent Consonant Recognition in Continuous Speech with Distinctive Phonetic Features,* Master of Science Thesis, McGill University, Montreal, Canada, 1991.

[28] **Fritsch J.,** *Modular Neural Networks for Speech Recognition*, Diploma Thesis, Pittsburg, PA, 1996.

[29] **Gallant A. R., White R.,** *There Exists a Neural Network That Does Not Make avoidable Mistakes,* in Proceedings of ICNN, vol. 1, pp. 657–664, 1988.

[30] **Garafolo S. J. et al.,** *DARPA TIMIT, Acoustic-Phonetic continuous Speech Corpus CDROM*, U.S. Department of Commerce, Ronald H. Brown Secretary, Documentation published, February, 1993.

[31] **Hartmann W. M.,** *Signals, Sounds and Sensation,* published by Library of Congress Cataloging-in-Publication Data, Springer-Verlag New York, Inc., ISBN 1-56396-283-7, 1998.

[32] **Haykin S.,** *Adaptive Filter Theory*, Third edition, Prentice Hall International Editions, Upper Saddle River, New Jersey 07458, 1996.

[33] **Hermansky H., and Morgan N.,** *RASTA Processing of Speech*, IEEE transaction on Speech and Audio Processing vol. 2, n°. 4 , pp. 578–589, October, 1994.

[34] **Hermansky H., Morgan N., and Hirsch N.-G.,** *Recognition of Speech in Additive and Convolutional Noise Based on Rasta Spectral Processing*, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, vol. II, pp. 83–86, April, 1995.

[35] **Hermansky H.,** *Perceptual Linear Predictive (PLP) Analysis of Speech,* JASA, 87(4), pp. 1738–1752, 1990.

[36] **Hermansky H.,** *Analysis in Automatic Recognition of Speech*, in Speech Processing, Recognition and Artificial Neural Networks, Proceedings of the 3th International School on Neural Nets "Eduardo R. Caianiello", edited by G. Chollet et al., Springer-Verlag, London limited, pp. 115–137, 1998.

[37] **Hirsch H. G.,** *Estimation of Noise Spectrum and its Application to SNR-Estimation and Speech Enhancement.* Technical Report TR-93-012, International Computer Science Institute, 1993.

[38] **Hornik K., et al.,** *Multilayer Feedforward Networks are Universal Approximators,* in Neural Networks, 2, pp. 358–366, 1989.

[39] **Jang J.-S. R., Sun C.-T.,** *Neuro-Fuzzy Modeling and Control*, in Proceedings of the IEEE, vol. 83, pp. 378–406, March, 1995.

[40] **Jang J.-S. R.,** *ANFIS: Adaptive Network-Based Fuzzy Inference Systems*, IEEE Transaction on Systems, Man, and Cybernetics, vol. 23, pp. 665–685, May, 1993.

[41] **Jang J.-S. R.,** *Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems,* IEEE Transaction on Neural Networks, vol. 4, pp. 156–159, January, 1993.

[42] **Jang J.-S. R.,** *Self Learning Fuzzy Controllers Based on Temporal Back-Propagation*, IEEE Transaction on Neural Networks, vol. 3, pp. 714–724, September, 1992.

[43] **Jang J.-S. R.,** *Neuro-Fuzzy Modeling: Architectures, Analyses, and Applications*, Ph.D. Dissertation, EECS Department, University of California at Berkeley, 1992.

[44] **Jang J.-S. R., Sun C.-T., Mizutani E.,** *Neuro-Fuzzy and Soft Computing : A Computational Approach to Learning and Machine Intelligence.* Prentice Hall, Upper Saddle River, New Jersey 07458, 1997.

[45] **Jensen S. H., et al.,** *Reduction of Broad-Band Noise in Speech by Truncated QSVD,* in IEEE Transaction, Speech, Audio, Processing, vol. 3, n°3, pp. 439–448, November, 1995.

[46] **Jordan M.,** *Attractor Dynamics and Parallelism in a Connectionist Sequential Machine,* in Proceedings of Eighth Annual Conference of the Cognitive Science Society, pp. 531–546, 1986.

[47] **Kent R. D., Read C.,** *The Acoustic Analysis of Speech*, Published by Singular Publishing Group, Inc., June, 1996.

[48] **Kilgour R.,** *Hybrid Fuzzy Systems and Neural Networks for Speech Recognition,* Master of Science Thesis, University of Otago, Dunedin, New Zealand, April, 1997.

[49] **Kosko B.,** *Neural Networks and Fuzzy Systems: A Dynamical System Approach to Machine Intelligence,* Englewood Cliffs, New Jersey, Prentice Hall.

[50] **Kruse R. and Nürnberger A.,** *Learning Methods for Fuzzy Systems,* in Proceedings of the 8th International Symposium Non-Linear Electromagnetic Systems, ISEM'97, pp. 367–372, IOS-Press, Amsterdam, Netherland, 1998.

[51] **Ladefoged P.,** *A Course in Phonetics,* Third Edition, Edited by Buchholz T., ISBN 0-15-500173-6, Harcourt Brace College Publishers, 1993.

[52] **Lauro N. C., Davino C., Vistocco D.,** *Neural Networks Applications in Economics: a Statistical Point of View,* in Proceedings of Neural Nets, WIRN Vietri-99, Edited by M. Marinaro and R. Tagliaferri, Vietri Sul Mare, Italy, pp. 357–375, May, 1999.

[53] **Lee C. S.,** *Fuzzy Neural Networks*, Mathematicial Biosciences, vol. 23, pp. 151–177, Copyright 1975 by Elsevier Science Publishing Company, Inc.

[54] **Lin C.-T., Lee C. S.,** *Neural Fuzzy System: A Neuro-Fuzzy Synergism to Intelligent Systems*, Prentice Hall, Upper Saddle River, New Jersey, 1996.

[55] **Lippmann R. P.,** *An Introduction to Computing with Neural Networks,* in Proceedings of IEEE, Acoustics, Speech, and Signal Processing Magazine, 4(2), pp. 207–224, 1991.

[56] **Looney C. G.,** *Pattern Recognition Using Neural Networks,* Oxford University Press Inc., 1997.

[57] **Lisker L., Abramson A. S.,** *A Cross Language Study of Voicing in Initial Stops: Acoustical Measurements,* in Word, 20, pp. 384–422, 1964.

[58] **Lisker L., Abramson A. S.,** *Some Effects of Context on Voice Onset Time in English Stops,* in Language and Speech, 10(3), pp. 1–28, 1964.

[59] **Makhoul J.,** *Linear Prediction: A Tutorial Review,* Proc. of the IEEE, vol.63, n°4, pp.561–579, April, 1975.

[60] **Luo F-L, Unbehauen R.,** *Applied Neural Networks for Signal Processing,* Cambridge University Press, 1997.

[61] **Markowitz J.,** *Using Speech Recognition*, Prentice Hall PTR, Upper Saddle River, New Jersey 07458, 1996.

[62] **Mendel J. M.,** *Fuzzy Logic Systems for Engineering: A tutorial,* in Proceedings of the IEEE, vol. 83, n°3, March, 1995.

[63] **Moody J. E.,** *Notes on Generalization, Regularization and Architecture Selection in Nonlinear Learning Systems,* Proc. of IEEE/NNSP, Los Alamitos, CA, pp. 29–32, 1991.

[64] **Moody J., and Darken C.,** *Fast Learning in Networks of Locally-tuned Processing Units,* in Neural Computation, vol. 1, pp. 281–294, 1989.

[65] **Nauck D.,** *Neuro-Fuzzy Systems: Review and Prospects,* in Proceedings of Fifth European Congress on Intelligent Techniques and Soft Computing, EUFIT'97, Aachen, pp. 1044–1053, September 8-11, 1997.

[66] **Nauck D. and Kruse R.,** *What are Neuro-Fuzzy Classifiers ?*, in Proceedings of Seventh International Fuzzy Systems Association World Congress IFSA'97, vol. IV, pp. 228-.233, Academia Prague, 1997.

[67] **Nauck D., Klawonn F., Kruse R.,** *Fuzzy Sets, Fuzzy Controllers, and Neural Networks,* in Proceedings of Scientific Journal of the Humboldt of Berlin, Series Medecine 41, n°4, pp. 99–120, 1992.

[68] **Nerrand O. et al.,** *Training Recurrent Neural Networks: Why and How ?,* An Illustration in Dynamical Process Modeling, IEEE Trans. on Neural Networks, 5, pp. 178–184, 1994.

[69] **Nokas G. et al.,** *Speech Recognition in Noisy Reverberant Rooms using Frequency Domain Adaptive Filtering,* in Proceedings Text, Speech, Dialogue, TSD'98, pp. 281–286, Brno, Czech Republic, September, 1998.

[70] **Press H. William et al.,** *The Art of Scientific Computing,* Second Edition, Cambridge University Press, 1999.

[71] **Priemer R.,** *Introductory Signal Processing,* Advanced Series in Electrical and Computer Science, vol. 6, World Scientific.

[72] **Ramanujam V., et al.,** *Robust Speaker Verification in Noisy Conditions by Modification of Spectral Time Trajectories,* in Proceedings of EuropSpeech vol. 2, pp. 791–794, 1999.

[73] **Rumelhart D. E., McClelland J. L.,** *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, 1986.

[74] **The MathWorks Inc.,** *Signal Processing Toolbox User's Guide,* Copyright 1988–1998.

[75] **Sakoe H., et al.,** *Speaker-Independent Word Recognition Using Dynamic Programming Neural Networks,* in Proceedings of International Conference on Acoustics, Speech and Signal Processing, pp. 29–32, 1989.

[76] **Speak C. E.,** *Introduction to Sound, Acoustic for the Hearing and Speech Sciences*, second edition, published by Singular Publishing Group, Inc., 1996.

[77] **SNNS Group,** *Stuttgart Neural Network Simulator,* Copyright 1990-1995, Institute for Parallel and Distributed High-Performance System, University of Stuttgart, Germany.

[78] **Sutton R. S.,** *Two Problems with Backpropagation and Other Steepest Descent Learning Procedures for Networks,* in Proceedings of 8th Conference of Cognitive Science Society, pp. 823–831, 1986.

[79] **Waibel A.,** *Connectionist Glue: Modular Design of Neural Speech Systems,* in Proceedings of the Connectionist Models Summer School, Morgan Kaufmann, 1988.

[80] **Waibel A., et al.,** *Phoneme Recognition Using Time Delay Neural Networks,* in IEEE Transaction on Acoustics, Speech and Signal Processing, ASSP – 37(3), pp. 328–339, 1989.

[81] **Waibel A., et al.,** *Consonant Recognition by Modular Construction of Large Phonetic Time-Delay Neural Networks,* in Proceedings of ICASSP, pp. 112–115, 1989.

[82] **Wan A. E., and Nelson T. A.,** *Networks for Speech Enhancement*, in Handbook of Neural Networks for Speech Processing, Edited by Shigeru Katagari, Artech House, Boston, 1998.

[83] **Wan A. E., and Nelson T. A.,** *Neural Dual Extended Kalman Filtering: Applications in Speech Enhancement and Monaural Blind Signal Separation*, in Proceedings of IEEE Conference on Neural Networks for Signal Processing, edited by Principe, Morgan, Giles, Wilson, 1997.

[84] **Widrow B., and Stearns D. S.,** *Adaptive Signal Processing*, Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632, 1985.

[85] **Widrow B. et al.,** *Adaptive Noise Canceling: Principles and Applications,* in Proceedings of IEEE, vol. 63, n12, pp. 1692–1712, December, 1975.

[86] **Williams R. J.,** *A Learning Algorithm for Continually Running Fully Recurrent Neural Networks,* in Neural Computation, 1, pp. 270–280, 1989.

[87] **Williams R. J., and Ziper D.,** *Technical Report NU–CCS–90–9,*, Boston, Northeastern University, College of Computer Science, April 12, 1990.

[88] **Tamura S.,** *An Analysis of a Noise Reduction Neural Network,* in ICASSP 87, pp. 2001-2004, 1987.

[89] **Zadeh A. L.,** *Fuzzy Sets*, Information and Control, vol. 8, pp. 338-353, 1965.

[90] **Zue S., Seneff J. G.,** *Speech Database Development: TIMIT and beyond,* Speech Communication, pp. 351–356, 1990.