LECTURE NOTICE. INTRODUCTION TO SOFT COMPUTING

Dmitry Bystrov

Lecture Notice "Introduction to Soft Computing" are based on Heikki Koivo "Soft Computing in Dynamical Systems" and Robert Fuller "Introduction to Neuro-Fuzzy Systems" books.

Fuzzy logic systems chapter describes the basic definitions of fuzzy set theory, i.e., the basic notions, the properties of fuzzy sets and operations on fuzzy sets. Some popular constructions of fuzzy systems are presented in sections 2.2.1 - 2.2.3. The approximation capability is described in section 2.3, and section 2.4 summarizes the different interpretations of fuzzy sets.

Neural networks chapter gives a short introduction to neural networks. The basic concepts are presented in 3.1. After that, multilayer perceptron, radial basis function network, neurofuzzy networks and, finally, ANFIS (Adaptive Neurofuzzy Inference System) are reviewed. The chapter ends with a discussion about approximation capability.

Unified form of soft computing methods chapter describes soft computing methods in neural networks, i.e., interpolation networks, models for non-linear mapping. Some actual relations as posterior probability in FLS and neural network is presented in section 4.4. Basic function selection is described in section 4.5.

Training of soft computing methods chapter gives more detail describes of soft computing training methods in neural networks. The basic concepts are presented in section 5.1. After that, network mapping, regularization, the bias/variance tradeoff and training methods in nutshell are reviewed.

Conclusion chapter summarized basic relations in soft computing.

Contents

1. Introduction 5

1.1 Soft Computing 5
1.1.1 Fuzzy Logic 7
1.1.2 Neural Networks 10
1.1.3 Probabilistic Reasoning 11

2. Fuzzy Logic Systems 14

2.1 Basic Concepts 14
2.1.1 Set-Theoretical Operations and Basic Definitions 14
2.1.2 Fuzzy Relations 23
2.1.3 The Extension Principle 26
2.1.4 Approximate Reasoning 26
2.1.5 Fuzzy Rules 29
2.1.6 Fuzzy Inference 31
2.1.7 Fuzzifier and Defuzzifier 35
2.1.8 The General Relation between Probability and Fuzziness. 37
2.2 Different Fuzzy Systems 38
2.2.1 Takagi and Sugeno's Fuzzy System 38
2.2.2 Mendel-Wang's Fuzzy System 39
2.2.3 Kosko's Standard Additive Model (SAM) 43
2.3 Approximation Capability 44
2.4 Different Interpretations of Fuzzy Sets. 46
2.5 Different Ways to form Fuzzy Sets 47

3. Neural Networks 49

3.1 Basic Concepts 49
3.1.1 Single-layer Feedforward Networks 52
3.2 Multilayer Perceptron 55
3.3 Functional Link Network 56
3.4 Radial Basis Function Network 56
3.5 Neuro-Fuzzy Networks 61
3.5.1 Different Neurofuzzy Approaches 61
3.6 ANFIS 68
3.7 Approximation Capability 71

4. Unified Form of Soft Computing Methods 73

4.1 Introduction 73
4.2 Interpolation Networks 77
4.3 Models for Non-linear Mapping 80
4.4 Posterior Probability in FLS and Neural Networks 88
4.5 Basis Function Selection 90

5. Training of Soft Computing Methods 93

5.1 The Basics 94
5.2 Network Mapping 97
5.3 Regularization 98
5.4 The Bias/Variance Tradeoff 102
5.5 Training Methods in Nutshell 104

6. Conclusion 105

References 107

Chapter 1

Introduction

Nowadays, *fuzzy logic*, *neurocomputing* and *probabilistic reasoning* have rooted in many application areas (expert systems, pattern recognition, system control, etc.). The first two methods have received a lot of critique during their existence (for example, see [Minsky & Papert, 1969; Cheeseman, 1986]). The critique has been particularly strong with fuzzy logic, but it has weakened as fuzzy logic has been successfully applied to practical problems and its universal approximation property has been proven [Kosko, 1992; Wang, 1994; Castro & Delgado, 1996].

Although these methodologies seem to be different, they have many common features - like the use of basis functions (fuzzy logic has membership functions, neural networks have activation functions and probability regression use probability density functions) and the aim to estimate functions from sample data or heuristics.

The purpose of this Lecture Notice is to illustrate the use of soft computing methods. The emphasis is put on the fuzzy logic and only the basics of neural networks and probabilistic methods will be described. The probabilistic methods are reviewed only in the end of this introductory chapter.

This first chapter describes the term *soft computing* and gives an introduction to its basic components. The following chapters go deeper into these components. The 2nd chapter provides a description of fuzzy logic systems and 3rd chapter describes neural networks. Unified approach to these methods is presented in chapter 4 and common basics of learning methods in chapter 5. Chapter 6 is devoted to conclusion.

1.1 Soft Computing

The term *soft computing* was proposed by the inventor of fuzzy logic, Lotfi A. Zadeh. He describes it as follows [Zadeh, 1994]:

"*Soft computing is a collection of methodologies that aim to exploit the tolerance for imprecision and uncertainty to achieve tractability, robustness, and low solution cost. Its principal constituents are fuzzy logic, neurocomputing, and probabilistic reasoning. Soft computing is likely to play an increasingly important role in many application areas, including software engineering. The role model for soft computing is the human mind*."

Soft computing is not precisely defined. It consists of distinct concepts and techniques which aim to overcome the difficulties encountered in real world problems. These problems result from the fact that our world seems to be imprecise, uncertain and difficult to categorize. For example, the uncertainty in a measured quantity is due to inherent variations in the measurement process itself. The uncertainty in a result is due to the combined and accumulated effects of these measurement uncertainties which were used in the calculation of that result [Kirkpatrick, 1992].

In many cases the increase in precision and certainty can be achieved by a lot of work and cost. Zadeh gives as an example the travel salesman problem, in which the computation time is a function of accuracy and it increases exponentially [Zadeh, 1994].

Another possible definition of soft computing is to consider it as an anti-thesis to the concept of computer we now have, which can be described with all the adjectives such as hard, crisp, rigid, inflexible and stupid. Along this track, one may see soft computing as an attempt to mimic natural creatures: plants, animals, human beings, which are soft, flexible, adaptive and clever. In this sense soft computing is the name of a family of problem-solving methods that have analogy with biological reasoning and problem solving (sometimes referred to as cognitive computing). The basic methods included in cognitive computing are fuzzy logic (FL), neural networks (NN) and *genetic algorithms* (GA) - the methods which do not derive from classical theories.

Fuzzy logic is mainly associated to imprecision, approximate reasoning and computing with words, neurocomputing to learning and curve fitting (also to classification), and probabilistic reasoning to uncertainty and belief propagation (belief networks). These methods have in common that they

1. are nonlinear,
2. have ability to deal with non-linearities,
3. follow more human-like reasoning paths than classical methods,
4. utilize self-learning,
5. utilize yet-to-be-proven theorems,
6. are robust in the presence of noise or errors.

Kosko lists the following similarities between fuzzy logic systems and neural networks [Kosko, 1992]:

- estimate functions from sample data
- do not require mathematical model
- are dynamic systems
- can be expressed as a graph which is made up of nodes and edges
- convert numerical inputs to numerical outputs
- process inexact information inexactly
- have the same state space
- produce bounded signals
- a set of n neurons defines n-dimensional fuzzy sets
- learn some unknown probability function $p(\mathbf{x})$
- can act as associative memories
- can model any system provided the number of nodes is sufficient.

The main dissimilarity between fuzzy logic system (FLS) and neural network is that FLS uses heuristic knowledge to form rules and tunes these rules using sample data, whereas NN forms "rules" based entirely on data.

Soft computing methods have been applied to many real-world problems. Applications can be found in signal processing, pattern recognition, quality assurance and industrial inspection, business forecasting, speech processing, credit rating, adaptive process control, robotics control, natural-language understanding, etc. Possible new application areas are programming languages, user-friendly application interfaces, automaticized programming, computer networks, database management, fault diagnostics and information security [Zadeh, 1994].

In many cases, good results have been achieved by combining different soft computing methods. The number of this kind of hybrid systems is growing. A very interesting combination is the neurofuzzy architecture, in which the good properties of both methods are attempted to bring together. Most neurofuzzy systems are fuzzy rulebased systems in which techniques of neural networks are used for rule induction and calibration. Fuzzy logic may also be employed to improve the performance of optimization methods used with neural networks. For example, it may control the vibration of direction for searching vector in quasi Newton method [Kawarada & Suito, 1996].

Soft computing can also be seen as a foundation for the growing field of computational intelligence (CI). The difference between traditional artificial intelligence (AI) and computational intelligence is that AI is based on hard computing whereas CI is based on soft computing.

"*Soft Computing is not just a mixture of these ingredients, but a discipline in which each constituent contributes a distinct methodology for addressing problems in its domain, in a complementary rather than competitive way.*" [Zadeh, 1994].

1.1.1 Fuzzy Logic

Fuzzy set theory was developed by Lotfi A. Zadeh [Zadeh, 1965], professor for computer science at the University of California in Berkeley, to provide a mathematical tool for dealing with the concepts used in natural language (*linguistic variables*). Fuzzy Logic is basically a multivalued logic that allows intermediate values to be defined between conventional evaluations.

However, the story of fuzzy logic started much more earlier . To devise a concise theory of logic, and later mathematics, *Aristotle* posited the so-called "Laws of Thought". One of these, the "Law of the Excluded Middle," states that every proposition must either be *True* (**T**) or *False* (**F**). Even when *Parminedes* proposed the first version of this law (around 400 Before Christ) there were strong and immediate objections: for example, *Heraclitus* proposed that things could be simultaneously *True* and *not True*. It was *Plato* who laid the foundation for what would become fuzzy logic, indicating that there was a third region (beyond **T** and **F**) where these opposites "tumbled about." A systematic alternative to the bi-valued logic of Aristotle was first proposed by *Łukasiewicz* around 1920, when he described a three-valued logic, along with the mathematics to accompany it. The third value he proposed can best be translated as the term "possible," and he assigned it a numeric value between **T** and **F**. Eventually, he proposed an entire notation and axiomatic system from which he hoped to derive modern mathematics.

Later, he explored four-valued logics, five-valued logics, and then declared that in principle there was nothing to prevent the derivation of an infinite-valued logic.£ukasiewicz felt that three- and infinite-valued logics were the most intriguing, but he ultimately settled on a four-valued logic because it seemed to be the most easily adaptable to Aristotelian logic. It should be noted that *Knuth* also proposed a threevalued logic similar to Lukasiewicz's, from which he speculated that mathematics would become even more elegant than in traditional bi-valued logic. The notion of an infinite-valued logic was introduced in Zadeh's seminal work "Fuzzy Sets" where he described the mathematics of fuzzy set theory, and by extension fuzzy logic. This theory proposed making the membership function (or the values **F** and **T**) operate over the range of real numbers [0, 1].

New operations for the calculus of logic were proposed, and showed to be in principle at least a generalization of classic logic. Fuzzy logic provides an inference morphology that enables approximate human reasoning capabilities to be applied to knowledge-based systems. The theory of fuzzy logic provides a mathematical strength to capture the uncertainties associated with human cognitive processes, such as thinking and reasoning. The conventional approaches to knowledge representation lack the means for representating the meaning of fuzzy concepts. As a consequence, the approaches based on first order logic and classical probability theory do not provide an appropriate conceptual framework for dealing with the representation of commonsense knowledge, since such knowledge is by its nature both lexically imprecise and noncategorical.

The developement of fuzzy logic was motivated in large measure by the need for a conceptual framework which can address the issue of uncertainty and lexical imprecision. Some of the essential characteristics of fuzzy logic relate to the following (Zadeh, 1992): In fuzzy logic, exact reasoning is viewed as a limiting case of approximate reasoning. In fuzzy logic, everything is a matter of degree. In fuzzy logic, knowledge is interpreted a collection of elastic or, equivalently, fuzzy constraint on a collection of variables. Inference is viewed as a process of propagation of elastic constraints. Any logical system can be fuzzified. There are two main characteristics of fuzzy systems that give them better performance for specific applications. Fuzzy systems are suitable for uncertain or approximate reasoning, especially for the system with a mathematical model that is difficult to derive. Fuzzy logic allows decision making with estimated values under incomplete or uncertain information.

Theory has been attacked several times during its existence. For example, in 1972 Zadeh's colleague R. E. Kalman (the inventor of Kalman filter) commented on the importance of fuzzy logic:
"...Zadeh's proposal could be severely, fericiously, even brutally criticized from a technical point of view. This would be out of place here. But a blunt question remains: Is Zadeh presenting important ideas or is he indulging in wishful thinking?..."

The heaviest critique has been presented by probability theoreticians and that is the reason why many fuzzy logic authors (Kosko, Zadeh and Klir) have included the comparison between probability and fuzzy logic in their publications. Fuzzy researchers try to separate fuzzy logic from probability theory, whereas some probability theoreticians consider fuzzy logic a probability in disguise.

Claim: Probability theory is the only correct way of dealing with uncertainty and that anything can be done with fuzzy logic can be done equally well through the use of probability-based methods. And so fuzzy sets are unnecessary for representing and reasoning about uncertainty and vagueness - probability theory is all that is required. "*Close examination shows that the fuzzy approaches have*

exactly the same representation as the corresponding probabilistic approach and include similar calculi." [Cheeseman, 1986].

Objection: Classical probability theory is not sufficient to express uncertainty encountered in expert systems. The main limitation is that it is based on two-valued logic. An event either occurs or does not occur; there is nothing between them. Another limitation is that in reality events are not known with sufficient precision to be represented as real numbers. As an example considers a case in which we have been given information:

An urn contains 20 balls of various sizes, several of which are large.

"One cannot express this within the framework of classical theory or, if it can be done, it cannot be done simply" (Zadeh to Cheeseman, in same book). General and more consistent treatment of this dispute can be found in Kosko's intelligible-to-all book [Kosko, 1993] and in [Cheeseman, 1986].

Term *fuzzy logic* has two meanings. According to the first interpretation (in narrow sense) it is seen as a multi-valued "imprecise" logic and as an extension to the more traditional multi-valued logic. Bart Kosko explains this point of view by emphasizing that in reality everything seems to occur or to be true to a degree. Facts are always fuzzy, vague or inaccurate to some extent. "*Only mathematics has black and white facts and it is only a collection of artificial rules and symbols. Science deals with gray or fuzzy facts as if they were black-and-white facts of mathematics. Nobody has presented a fact having to do with the real world, that is 100 per cent true or 100 per cent false. They are said to be*."

The first meaning was a some kind of model for human reasoning. The other interpretation (in wide sense) is that fuzzy logic = fuzzy set theory. According to this view any field X can be fuzzified by changing a set in X by a fuzzy set [Zadeh, 1994]. For example, set theory, arithmetic, topology, graph theory, probability theory and logic can be fuzzified. This has already been done in neurocomputing, pattern recognition, mathematical programming and in stability theory.

If the conventional techniques of system analysis cannot be successfully incorporated to the modeling or control problem, the use of heuristic linguistic rules may be the most reasonable solution to the problem. For example, there is no mathematical model for truck and trailer reversing problem, in which the truck must be guided from an arbitrary initial position to a desired final position. Humans and fuzzy systems can perform this nonlinear control task with relative ease by using practical and at the same time imprecise rules as "If the trailer turns slightly left, then turn the wheel slightly left."

The most significant application area of fuzzy logic has been in control field. It has been made a rough guess that 90% of applications are in control (the main part deals with rather simple applications, see Fig. 1.1). Fuzzy control includes fans, complex aircraft engines and control surfaces, helicopter control, missile guidance, automatic transmission, wheel slip control, industrial processes and so on. Commercially most significant have been various household and entertainment electronics, for example washing machine controllers and autofocus cameras. The most famous controller is the subway train controller in Sengai, Japan. Fuzzy system performs better (uses less fuel, drives smoother) when compared with a conventional PID controller. Companies that have fuzzy research are General Electric, Siemens, Nissan, Mitsubishi, Honda, Sharp, Hitachi, Canon, Samsung, Omron, Fuji, McDonnell Douglas, Rockwell, etc. For an introduction to the fuzzy control see [Driankov et al., 1993; Berenji, 1992; Jager, 1995].

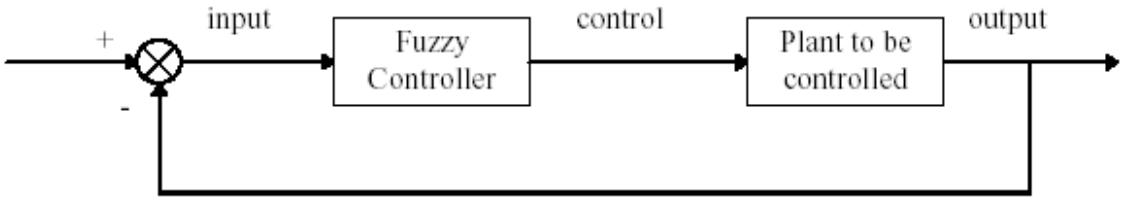


Figure 1.1 Example of a control problem

1.1.2 Neural Networks

The study of neural networks started by the publication of McCulloch and Pitts [1943]. The single-layer networks, with threshold activation functions, were introduced by Rosenblatt [1962]. These types of networks were called *perceptrons*. In the 1960s it was experimentally shown that perceptrons could solve many problems, but many problems which did not seem to be more difficult could not be solved. These limitations of one-layer perceptron were mathematically shown by Minsky and Papert in their book *Perceptron*. The result of this publication was that the neural networks lost their interestingness for almost two decades. In the mid-1980s, back-propagation algorithm was reported by Rumelhart, Hinton, and Williams [1986], which revived the study of neural networks. The significance of this new algorithm was that multilayer networks could be trained by using it.

Neural network makes an attempt to simulate human brain. The simulating is based on the present knowledge of brain function, and this knowledge is even at its best primitive. So, it is not absolutely wrong to claim that artificial neural networks probably have no close relationship to operation of human brains. The operation of brain is believed to be based on simple basic elements called neurons which are connected to each other with transmission lines called axons and receptive lines called dendrites (see Fig. 1.2). The learning may be based on two mechanisms: the creation of new connections, and the modification of connections. Each neuron has an activation level which, in contrast to Boolean logic, ranges between some minimum and maximum value.

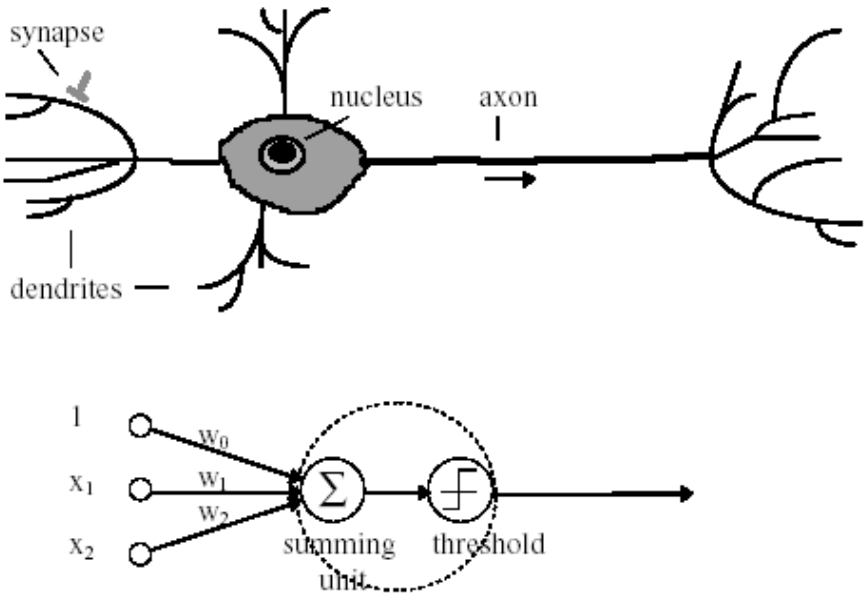


Figure 1.2 Simple illustration of biological and artificial neuron (perceptron).

In artificial neural networks the inputs of the neuron are combined in a linear way with different weights. The result of this combination is then fed into a non-linear activation unit (activation function), which can in its simplest form be a threshold unit (see Fig. 1.2).

Neural networks are often used to enhance and optimize fuzzy logic based systems, e.g., by giving them a learning ability. This learning ability is achieved by presenting a training set of different examples to the network and using learning algorithm which changes the weights (or the parameters of activation functions) in such a way that the network will reproduce a correct output with the correct input values. The difficulty is how to guarantee generalization and to determine when the network is sufficiently trained.

Neural networks offer nonlinearity, input-output mapping, adaptivity and fault tolerance. Nonlinearity is a desired property if the generator of input signal is inherently nonlinear [Haykin, 1994]. The high connectivity of the network ensures that the influence of errors in a few terms will be minor, which ideally gives a high fault tolerance. (Note that an ordinary sequential computation may be ruined by a single bit error).

1.1.3 Probabilistic Reasoning

As fuzzy set theory, the probability theory deals with the uncertainty, but usually the type of uncertainty is different. Stochastic uncertainty deals with the uncertainty toward the occurrence of certain event and this uncertainty is quantified by a degree of probability. Probability statements can be combined with other statements using stochastic methods. Most known is the Bayesian calculus of conditional probability.

Probabilistic reasoning includes genetic algorithms, belief networks, chaotic systems and parts of learning theory [Zadeh, 1994]. (Although, in this thesis, emphasis will be on fuzzy logic systems and neural networks, probabilistic reasoning is included because of the consistency).

Genetic algorithms:

Genetic algorithms optimize a given function by means of a random search. They are best suited for optimization and tuning problems in the cases where no prior information is available. As an optimization method genetic algorithms are much more effective than a random search.

They create a child generation from parent generation according to a set of rules that mimic the genetic reproduction in biology. Randomness plays an important role, since

- the parents are selected randomly, but the best parents have greater probability of being selected than the others
- the number of 'genes' to be muted is selected randomly
- all bits in new child string can be flipped with a small probability

Probability:

Random events are used to model uncertainty and they are measured by probabilities. A random event E is defined as a crisp subset of a sample space U. The probability of $E, P(E) \in [0,1]$, is the proportion of occurrences of E. The probability is supposed to fulfill the axioms of Kolmogorov:

- $P(E) \geq 0 \quad \forall E \subset U$
- $P(U) = 1$
- if E_i are distinct sets, then $P(\cup_i E_i) = \sum_i P(E_i)$.

A simple example of the classical probability is the throwing of the dice. Let U be the set of integers $\{1,2,3,4,5,6\}$. An event such as "$E = 6$" has a probability $P(E = 6) = \frac{1}{6}$.

Subjective probability (Bayesian interpretation of probability) is described as the amount of subjective belief that a certain event may occur, and is the most similar concept to fuzzy logic. Value of 1 is used to denote complete certainty that an event will occur, and 0 to denote complete certainty that the event will not occur. The values between 0 and 1 represent degrees of belief. This view differs from the frequentistic view of probability. Indeed, some authors, like Kosko in [Kosko, 1990], have proposed that the subjective probability theory is the subset of fuzzy logic.

Bayesian interpretation of probability is linked to *joint probability* and *conditional probability* through the *Bayes' theorem* which can be written as

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} = \frac{P(B,A)}{P(A)} \tag{1.1}$$

where $P(A|B)$ represents the conditional probability that specifies the probability of the event A given that the event B occurs. The quantity $P(B)$ is the probability of B with respect to the whole event space (*prior probability*). $P(B,A)$, is defined to be the probability that both the events A and B occur (joint probability).

A related approach to Bayesian probability is the *Dempster-Shafer belief theory* which is also referred to as the *Dempster-Shafer theory of evidence* (the latter is preferred terminology). A central concept is the *belief function*.

Definition 1.1. (*belief*) Let X be a finite set and let m be a function from subsets of X to nonnegative reals such that

$$m(\varnothing) = 0 \text{ and } \sum_{T:T\subseteq X} m(t) = 1 \tag{1.2}$$

The belief is defined by

$$Bel(S) = \sum_{T:T\subseteq S} m(T) \tag{1.3}$$

Different belief functions can be combined by the *Dempster's rule of combination*.

Belief network:

A belief network may also be referred to as a *Bayesian network* (preferred terminology), or as a causal network. Bayesian network is a model for representing uncertainty in our knowledge and it uses probability theory as a tool to manage this uncertainty. Network has a structure which reflects causal relationships, and a probability part which reflects the strengths of these relationships. User of network must provide observation based information about the value of a random variable. Network then gives the probability of another random variable.

Chapter 2

Fuzzy Logic Systems

2.1 Basic Concepts

Since set theory forms a base for logic, we begin with fuzzy set theory in order to "pave the way" to fuzzy logic and fuzzy logic systems.

2.1.1 Set-Theoretical Operations and Basic Definitions

In classical set theory the membership of element x of a set A (A is a crisp subset of universe X) is defined by

$$\mu_A(x) = \begin{cases} 0, \text{ if } x \notin A, \\ 1, \text{ if } x \in A \end{cases} \tag{2.1}$$

The element either belongs to the set or not. In fuzzy set theory, the element can belong to the set partially with a degree and the set does not have crisp boundaries. That leads to the following definition.

Definition 2.1.1 (*fuzzy set, membership function*) Let X be a nonempty set, for example $X = R^n$, and be called the universe of discourse. A fuzzy set $A \subset X$ is characterized by the membership function

$$\mu_A : X \to [0,1] \tag{2.2}$$

where $\mu_A(x)$ is a grade (degree) of membership of x x in set A.
From the definition we can see that the fuzzy set theory is a generalized set theory that includes the classical set theory as a special case. Since $\{0,1\} \in [0,1]$, crisp sets are fuzzy sets. Membership function (2.2) can also viewed as a distribution of truth of a variable. In literature fuzzy set A is often presented as a set of ordered pairs:

$$A = \{(x, \mu_A(x)) | x \in X\} \tag{2.3}$$

where the first part determines the element and the second part determines the grade of membership. Another way to describe fuzzy set has been presented by Zadeh, Dubois and Prade. If X is infinite, the fuzzy set A can be expressed by

$$A = \int_X \mu_A(x) / x \tag{2.4}$$

If X is finite, A can be expressed in the form

$$A = \mu_A(x_i) / x_1 + ... + \mu_A(x_n) / x_n = \sum_{i=1}^{n} \mu_A(x_i) / x_i \tag{2.5}$$

Note: Symbol $\int$ in (2.4) has nothing to do with integral (it denotes an uncountable enumeration) and / denotes a tuple. The plus sign represents the union. Also note that fuzzy sets are membership functions. Nevertheless, we may still use the set theoretic notations like $A \cup B$. This is the name of a fuzzy set given by $\mu_{A \cup B}$ (will be defined later).

Example 2.1.1

Discrete case: $\mu_A = 0.1/x_1 + 0.4/x_2 + 0.8/x_3 + 1.0/x_4 + 0.8/x_5 + 0.4/x_6 + 0.1/x_7$

Continuous case : $$\mu_A(x) = \begin{cases} 1 - \dfrac{x-c}{h} & , x \in [c-h, h] \\ \dfrac{x-c}{h} & , x \in [c, c+h] \\ 0 & , \text{otherwise} \end{cases}$$

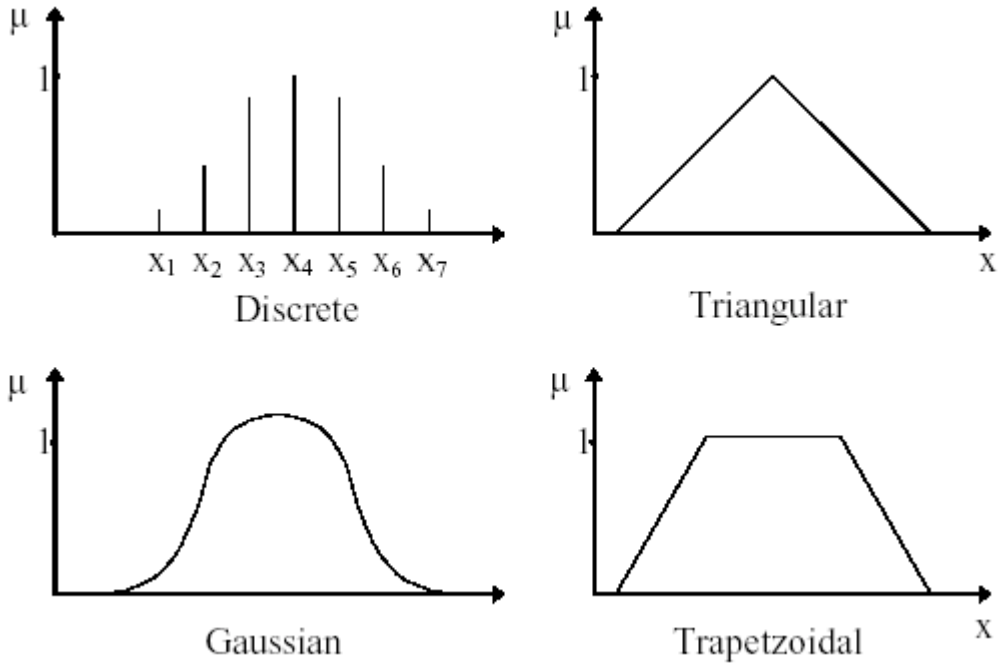


Figure 2.1 A discrete and continuous membership functions

Definition 2.1.2 (*support*) The support of a fuzzy set A is the crisp set that contains all elements of A with non-zero membership grade:

$$\mathrm{supp}(A) = \{x \in X \,|\, \mu_A(x) > 0\} \tag{2.6}$$

If the support is finite, it is called *compact support*. If the support of fuzzy set *A* consists of only one point, it is called a *fuzzy singleton*. If the membership grade of this fuzzy singleton is one, *A* is called a *crisp singleton* [Zimmermann, 1985].

Definition 2.1.3 (*core*) The core (nucleus, center) of a fuzzy set A is defined by

$$\mathrm{core}(A) = \{x \in X \,|\, \mu_A(x) = 1\} \tag{2.7}$$

Definition 2.1.4 (*height*) The height of a fuzzy set A on X is defined by

$$\mathrm{hgt}(A) = \sup_{x \in X} \mu_A(x) \tag{2.8}$$

and A is called *normal* if $\mathrm{hgt}(A) = 1$, and *subnormal* if $\mathrm{hgt}(A) < 1$.

Note: Non-empty fuzzy set can be normalized by dividing $\mu_A(x)$ by $\sup_x \mu_A(x)$. Normalizing of A can be regarded as a mapping from fuzzy sets to possibility distributions: $\text{Normal}(A) \mapsto \pi_A \equiv \mu_A$ [Joslyn, 1994].

The relation between fuzzy set membership function μ, possibility distribution π and probability distribution p: The definition $p_A(x) \equiv \mu_A(x)$ could hold, if μ is additively normal. Additively normal means here that the stochastic normalization

$$\int_X \mu_A(x)dx = 1$$

would have to be satisfied. So it can be concluded that any given fuzzy set could define either a probability distribution or a possibility distribution, depending on the properties of μ**.** Both probability distributions and possibility distributions are special cases of fuzzy sets. All general distributions are in fact fuzzy sets [Joslyn, 1994].

Definition 2.1.5 (*convex* fuzzy set) A fuzzy set A is convex if

$$\begin{aligned} &\forall x, y \in X \quad \text{and} \quad \forall \lambda \in [0,1] \\ &\mu_A(\lambda x + (1-\lambda)y) \geq \min(\mu_A(x), \mu_A(y)) \end{aligned} \tag{2.9}$$

Definition 2.1.6 (*width* of a convex fuzzy set) The width of a convex fuzzy set A is defined by

$$\text{width}(A) = \sup(\text{supp}(A)) - \inf(\text{supp}(A)) \tag{2.10}$$

Definition 2.1.7 (α *-cut*) The α -cut of a fuzzy set A is defined by

$$\alpha - cut(A) = \{x \in X \,|\, \mu_A(x) \geq \alpha\} \tag{2.11}$$

and it is called a *strong* $\alpha - cut$, if $\geq$ is replaced by >. $\alpha - cut$ is a generalized support.

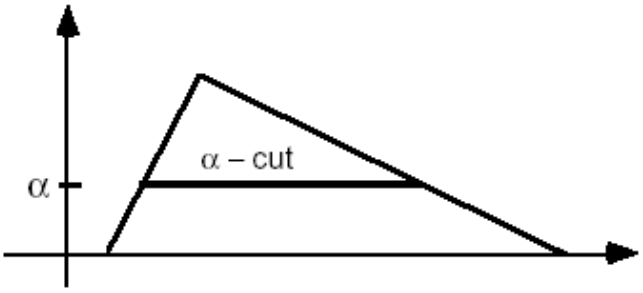


Figure 2.2 An $\alpha - cut$ of a triangular fuzzy number

Definition 2.1.8 (*fuzzy partition*) A set of fuzzy sets is called fuzzy partition if

$$\forall x \in X$$

$$\sum_{i=1}^{N_A} \mu_{A_i} = 1 \tag{2.12}$$

provided A_i are nonempty and subsets of X.

Definition 2.1.9 (*fuzzy number*) A fuzzy set (subset of a real line $\mathbf{R}$) is a fuzzy number, if the fuzzy set is convex, normal, membership function is piecewise continuous and the core consists of one value only. The family of fuzzy numbers is $\mathcal{F}$. In many situations people are only able to characterize numeric information imprecisely. For example, people use terms such as, about 5000, near zero, or essentially bigger than 5000. These are examples of what are called *fuzzy numbers*. Using the theory of fuzzy subsets we can represent these fuzzy numbers as fuzzy subsets of the set of real numbers.

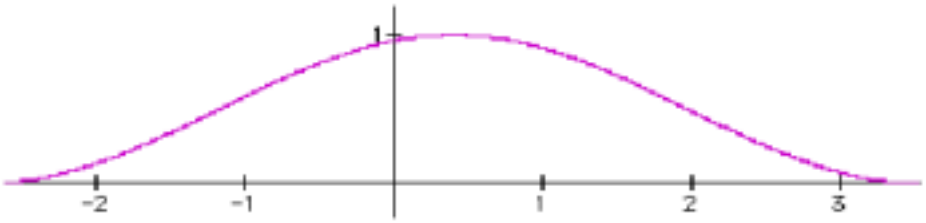


Figure 2.3 Fuzzy number

Figure 2.4 Non-fuzzy number

Note: Fuzzy number is always a fuzzy set, but a fuzzy set is not always a fuzzy number.

An example of fuzzy number is 'about 1' that is defined by $\mu_A(x) = \exp(-\beta(x-1)^2)$. It is also a *quasi fuzzy number* because $\lim_{x \to \pm\infty} \mu_A(x) = 0$.

Definition 2.1.10 (*fuzzy interval*) A fuzzy interval is a fuzzy set with the same restrictions as in definition 2.1.9, but the core does not need to be a one point only.

Fuzzy intervals are direct generalizations of crisp intervals $[a, b] \in \mathbf{R}$.

Definition 2.1.11 (*LR-representation of fuzzy numbers*) Any fuzzy number can be described by

$$\mu_A(x) = \begin{cases} L((a-x)/\alpha & , x \in [a-\alpha, a] \\ 1 & , x \in [a,b] \\ R((x-b)/\beta & , x \in [b, b+\beta] \\ 0 & , \text{otherwise} \end{cases} \tag{2.13}$$

where $[a,b]$ is the core of A, and $L:[0,1]\to[0,1], R:[0,1]\to[0,1]$. are shape functions (called briefly *s-functions*) that are continuous and non-increasing such that $L(0)=R(0)=1, L(1)=R(1)=0$, where L stands for left-hand side and R stands for right-hand side of membership function [Zimmermann, 1993].

Definition 2.1.12 (*LR-representation of quasi fuzzy numbers*) Any quasi fuzzy number can be described by

$$\mu_A(x) = \begin{cases} L((a-x)/\alpha & , x \le a \\ 1 & , x \in [a,b] \\ R((x-b)/\beta & , x \ge b \end{cases} \tag{2.14}$$

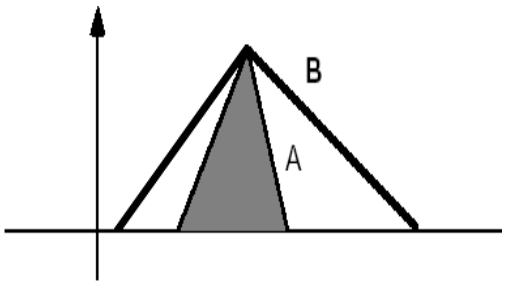
where $[a,b]$ is the core of A, and $L:[0,\infty)\to[0,1], R:[0,\infty)\to[0,1]$ are shape functions that are continuous and non-increasing such that $L(0)=R(0)=1$ and they approach zero: $\lim_{x\to\infty} L(x)=0$, $\lim_{x\to\infty} R(x)=0$

For example, $f(x)=e^{-x}$, $f(x)=e^{-x^2}$ and $f(x)=\max(0.1-x)$ are such shape functions. In the following the classical set theoretic operations are extended to fuzzy sets.

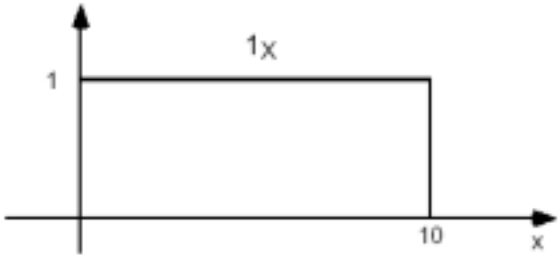
Definition 2.1.13 (*set theoretic operations*)

$\mu_\varnothing \equiv 0$	(empty set)
$\mu_X \equiv 1$	(basic set, universe)
$A=B \Leftrightarrow \mu_A(x)=\mu_B(x) \quad \forall x \in X$	(identity)
$A \subset B \Leftrightarrow \mu_A(x) \le \mu_B(x) \quad \forall x \in X$	(subsethood)
$\forall x \in X : \mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x))$	(union)
$\forall x \in X : \mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$	(intersection)
$\forall x \in X : \mu_{\bar{A}}(x) = 1-\mu_A(x)$	(complement)

Union could also be represented by $A \cup A = \{(x, \max(\mu_A(x), \mu_B(x))) | x \in X\}$ Same notation could also be used with intersection and complement.



A is a subset of B



The graph of the universal fuzzy subset in X = [0, 10].

Figure 2.5 Fuzzy sets

Theorem 2.1.1 The following properties of set theory are valid

$\overline{\overline{A}} = A$ (involution)

$A \cup B = B \cup A$
$A \cap B = B \cap A$ (commutativity)

$(A \cup B) \cup C = A \cup (B \cup C)$
$(A \cap B) \cap C = A \cap (B \cap C)$ (associativity)

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ (distributivity)

$A \cup A = A$
$A \cap A = A$ (idempotence)

$A \cup (A \cap B) = A$
$A \cap (A \cup B) = A$ (absorption)

$\overline{(A \cup B)} = \overline{A} \cap \overline{B}$
$\overline{(A \cap B)} = \overline{A} \cup \overline{B}$ (De Morgan's laws)

Proof: Above properties can be proved by simple direct calculations. For example, $\overline{\overline{A}} = 1 - (1 - A) = A$.

The laws of contradiction $A \cap \overline{A} = \emptyset$ (basis for many mathematical proofs) and excluded middle $A \cup \overline{A} = X$ are not valid for fuzzy sets. However, for the crisp sets they are valid. Bertrand Russell commented the law of excluded middle in the following way:

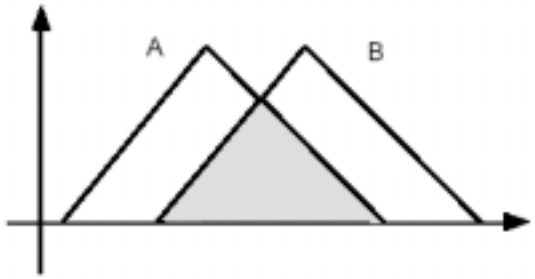
The law of excluded middle is true when precise symbols are employed but it is not true when symbols are vague, as, in fact, all symbols are.

Also other definitions for set operations are possible. An alternative well-known definition of union and intersection is given below. Operations are mostly selected relying on intuition or empirical results in particular problem.

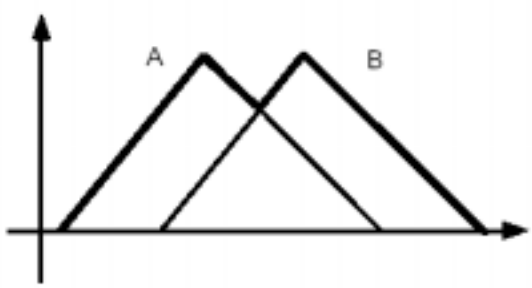
Definition 2.1.14

$\forall x \in X : \mu_{A \cup B}(x) = \min(1, \mu_A(x) + \mu_B(x))$ (union)

$\forall x \in X : \mu_{A \cap B}(x) = \mu_A(x) \bullet \mu_B(x)$ (intersection)



Intersection of two triangular fuzzy numbers



Union of two triangular numbers

Figure 2.6 Operations on fuzzy sets

More generally, the term *aggregation operator* is used to describe the union and intersection operators. Aggregation-operators can be divided into two groups: *compensatory* and *non-compensatory*. Non-compensatory intersection operators are called *t-norms* and union operators are called *t-conorms* or *s-norms*. Compensatory ones can perform operations which are somewhere between intersection and union operations.

Definition 2.1.15 (*t-norm*, "generalized intersection") Function $T : [0,1] \times [0,1] \to [0,1]$ is a t-norm, or triangular norm, if it satisfies the criteria:

(i) $T(x, y) = T(y, x)$ (symmetry)

(ii) $T(T(x, y), z) = T(x, T(y, z))$ (associativity)

(iii) $x \le z$ and $y \le v$ implies $T(x, y) \le T(z, v)$ (monotonicity)

(iv) $T(x,1) = x$ (border condition,one identy)

The basic t-norms are:

- minimum: $\min(a,b) = \min\{a,b\}$,
- Lukasiewisz: $T_L(a,b) = \max\{a+b-1,0\}$
- product: $T_p(a,b) = ab$
- weak:

$$T_w(a,b) = \begin{cases} \min\{a,b\} & \text{if } \max\{a,b\} = 1 \\ 0 & \text{otherwise} \end{cases}$$

- Hamacher $H_\gamma(a,b) = \dfrac{ab}{\gamma + (1-\gamma)(a+b-ab)}, \gamma \geq 0$
- Dubois and Prade

$$D_\alpha(a,b) = \frac{ab}{\max\{a,b,\alpha\}}, \alpha \in (0,1)$$

- Yager

$$Y_p(a,b) = 1 - \min\left\{1, \sqrt[p]{\left[(1-a)^p + (1+b)^p\right]}\right\}, p > 0$$

- Frank

$$F_\lambda(a,b) = \begin{cases} \min\{a,b\} & \text{if } \lambda = 0 \\ T_p(a,b) & \text{if } \lambda = 1 \\ T_L(a,b) & \text{if } \lambda = \infty \\ 1 - \log_\lambda\left[1 + \dfrac{(\lambda^a - 1)(\lambda^b - 1)}{\lambda - 1}\right] & \text{otherwise} \end{cases}$$

Definition 2.1.16 (*t-conorm*, "generalized union") Function $S : [0.1] \times [0.1] \to [0.1]$ is a t -conorm (s-norm) if it satisfies the criteria:

(i)-(iii) Same as in the definition of t-norms

(iv) $S(x,0) = x$

There is a special relation between t-norms and t-conorms if they are conjugate to each other: $S(x,y) = 1 - T(1-x,1-y)$. Among other, minimum and maximum operations satisfy this relation [Driankov et. al., 1993]. Relation is not essential, but it is sometimes used to select suitable pair of operations.

(i) $S(x, y) = S(y, x)$ (symmetry)

(ii) $S(x, S(y, z)) = S(S(x, y), z)$ (associativity)

(iii) $S(x, y) \le S(x', y') \ \ if \ \ x \le x' \ \ and \ \ y \le y'$ (monotonicity)

(iv) $S(x,0) = x, \quad \forall x \in [0,1]$ (zero identy)

The basic t-conorms are:

- maximum: $\max(a,b) = \max\{a,b\}$
- Lukasiewicz: $S_L(a,b) = \min\{a+b,1\}$
- probabilistic: $S_p(a,b) = a + b - ab$
- strong

$$STRONG(a,b) = \begin{cases} \max\{a,b\} & \text{if } \min\{a,b\} = 0 \\ 1 & \text{otherwise} \end{cases}$$

- Hamacher

$$HOR_\gamma(a,b) = \frac{a + b - (2-\gamma)ab}{1 - (1-\gamma)ab}, \gamma \ge 0$$

- Yager

$$YOR_p(a,b) = \min\left\{1, \sqrt[p]{a^p + b^p}\right\}, p > 0$$

Note: If the product (see def. 2.1.14) is used as a t-norm, the idempotence is not valid since $\mu_A(x) \cdot \mu_A(x) < \mu_A(x)$. Also the distributivity is not valid, which can be verified by substituting operations in def. 2.1.14 into distributivity law. However, the product function is gradually replacing the min operation in the real world applications. It has been shown to perform better in many situations [Driankov et. al., 1993; Brown & Harris, 1994].

2.1.2 Fuzzy Relations

Definition 2.1.17 (*fuzzy relation*) Fuzzy relation is characterized by a function $\mu_R : X_1 \times ... \times X_m \to [0,1]$ where X_i are the universes of discourse and $X_1 \times ... \times X_m$ is the product space. If we have two finite universes, the fuzzy relation can be presented as a matrix (*fuzzy matrix*) whose elements are the intensities of the relation and $\boldsymbol{R}$ has the membership function $\mu_R(u,v)$, where $u \in X_1, v \in X_2$. Two fuzzy relations are combined by a so called sup-*- or max-min composition, which will be given in the definition 2.1.19.

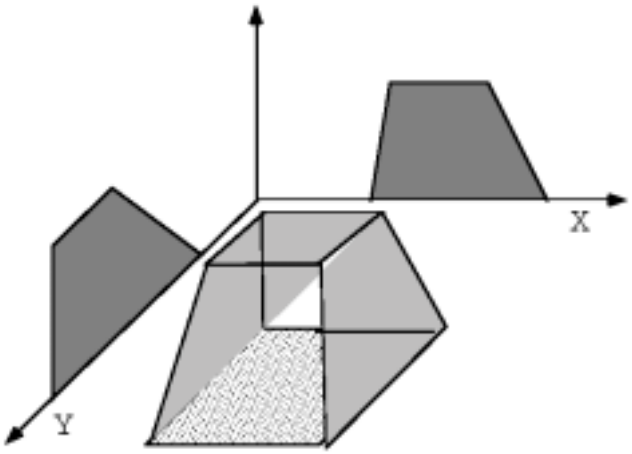


Figure 2.7 Shadows of a fuzzy relation

Note: Fuzzy relations are fuzzy sets, and so the operations of fuzzy sets (union, intersection, etc.) can be applied to them.

Example 2.1.2 Let the fuzzy relation $R =$ "approximately equal" correspond to the equality of two numbers. The intensity of cell $R(u,v)$ of the following matrix can be interpreted as the degree of membership of the ordered pair in *R*. The numbers to be compared are $\{1,2,3,4\}$ and $\{3,4,5,6\}$.

$$R(u,v) = \begin{array}{c|cccc} u \setminus v & 1 & 2 & 3 & 4 \\ \hline 3 & .6 & .8 & 1 & .8 \\ 4 & .4 & .6 & .8 & 1 \\ 5 & .2 & .4 & .6 & .8 \\ 6 & .1 & .2 & .4 & .6 \end{array}$$

The matrix shows, that the pair (4, 4) is approximately equal with intensity 1 and the pair (1, 6) is approximately equal with intensity 0.1.

Definition 2.1.18 (*Cartesian product*) Let $A_i \in X_i$ be fuzzy sets. Then the Cartesian product is defined

$$\mu_{A_1 \times \ldots \times A_n}(x_1,\ldots,x_n) = \min_{1 \le i \le n} \{\mu_{A_i}(x_i)\} \tag{2.15}$$

Note: min can be replaced by a more general t-norm.

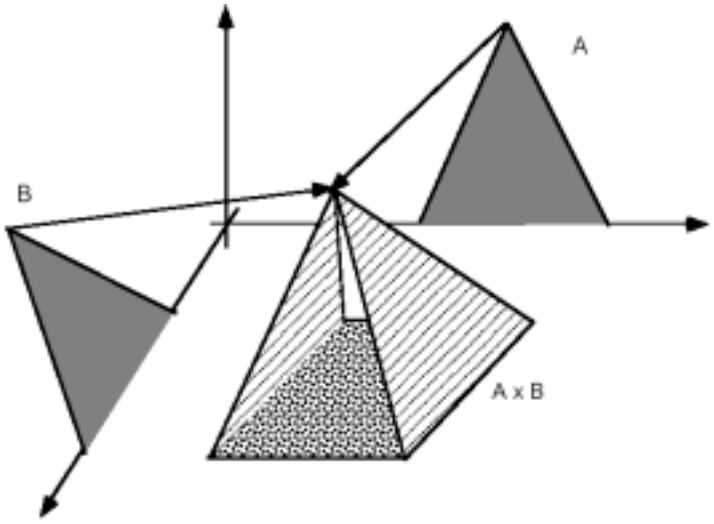


Figure 2.8 Cartesian product of two fuzzy sets is a fuzzy relation in $X \times Y$

Definition 2.1.19 (*sup-*-composition, max-min composition*) The composition of two relations $R \circ S$ is defined as a membership function in $X \times Y$

$$\mu_{R \circ S}(x, y) = \sup_v T(\mu_R(x, v), \mu_S(v, y)) \tag{2.16}$$

where R is relation in $X \times V$ and S is relation in $V \times Y, x \in X, y \in Y$ and T is a t-norm.

Composition corresponds to the product of matrices, as the t-norm is replaced by the product and sup is replaced by the sum. If S is just a fuzzy set (not a relation) in V, then (2.15) becomes

$$\mu_{R \circ S}(x) = \sup_v T(\mu_R(x, v), \mu_S(v)) \tag{2.17}$$

Example 2.1.3 Let $X = \{1,2,3,4\}$, fuzzy set $A = small = \{(1,1), (2,0.6), (3,0.2), (4.0)\}$ and fuzzy relation $R =$ "approximately equal".

$$R = \begin{pmatrix} 1 & .5 & 0 & 0 \\ .5 & 1 & .5 & 0 \\ 0 & .5 & 1 & .5 \\ 0 & 0 & .5 & 1 \end{pmatrix}$$

$$A \circ B = \max_x \min \{\mu_A(x), \mu_R(x, y)\}$$
$$= \{(1,1), (2,.6), (3,.5), (4,.2)\}$$

The interpretation of example: x is small. If x and y are approximately equal, y is more or less small.

2.1.3 The Extension Principle

The extension principle is said to be one of the most important tools in fuzzy logic. It gives means to generalize non-fuzzy concepts, e.g., mathematical operations, to fuzzy sets. Any fuzzifying generalization must be consistent with the crisp cases.

Definition 2.1.20 (*extension principle*) Let $A_1,\ldots,A_n$ be fuzzy sets, defined on $X_1 \ldots X_n$ and let f be a function $f : X_i \times \ldots \times X_n \to V$. The extension of f operating on $A_1,\ldots,A_n$ gives a membership function (fuzzy set F)

$$\mu_F(v) = \sup_{u_1,\ldots,u_n \in f^{-1}(v)} \min(\mu_{A_1}(u_1),\ldots,\mu_{A_n}(u_n)) \tag{2.18}$$

when the inverse of f exists. Otherwise define $\mu_F(v) = 0$. Function f is called *inducing mapping*.

If the domain is either discrete or compact, sup-min can be replaced by max-min. On continuous domains sup-operation and the operation that satisfies criterion

$$S_w(x,y) = \begin{cases} x & , y = 0 \\ y & , x = 0 \\ 1 & , \text{otherwise} \end{cases} \tag{2.19}$$

should be used [Driankov et. al., 1993].

2.1.4 Approximate Reasoning

Definition 2.1.21 (*linguistic variable, fuzzy variable*) A linguistic variable is characterized by a quintuple (V, T, X, G, M), where V denotes the symbolic name of a linguistic variable, e.g., *error*, *temperature*. T is the set of linguistic values that V can take (term set of V), e.g., {*negative*, *zero*, *positive*}. X is the domain over which the linguistic variable V takes its quantitative values, e.g., a continuous interval $[-10,10]$. Sometimes the universe of discourse is used instead of X. G is a syntactic rule which defines the elements (names of values) of V and M is a semantic rule which gives the interpretation of linguistic value in terms of quantitative elements.

The definition 2.1.21 says that linguistic variable may take words in natural language (or numbers) as its values. These words are labels of fuzzy sets. Fig. 2.9 describes the linguistic variable graphically.

Definition 2.1.22 (*completeness*) The fuzzy variable is said to be complete if for each $x \in X$ there exists a fuzzy set A such that

$$\mu_A(x) > 0 \tag{2.20}$$

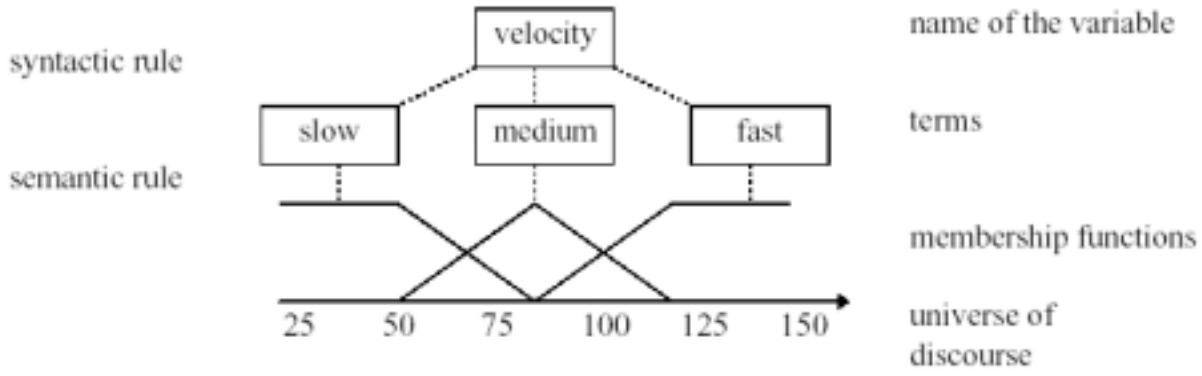


Figure 2.9 Linguistic variable 'velocity'

Definition 2.1.23 (*hedge*) A hedge is an operation which modifies the meaning of the term:

concentration (very): $$\mu_{con(A)}(x) = (\mu_A(x))^2 \tag{2.21}$$

dilation (more or less): $$\mu_{dil(A)}(x) = (\mu_A(x))^{1/2} \tag{2.21}$$

contrast intensification:

$$\mu_{\text{int}(A)}(x) = \begin{cases} 2(\mu_A(x))^2 & , \ 0 \le \mu_A(x) \le 0.5 \\ 1 - 2(1 - \mu_A(x))^2 & , \ 0.5 < \mu_A(x) \le 1 \end{cases} \tag{2.23}$$

If the term is not derived from some other term by using hedge, it is called a *primary term*. If the primary term 'cold' is modified by the concentration, we get a new term 'very cold'.

More generally, linguistic variable can be seen as a micro language, which includes context-free grammar and attributed-grammar semantics. Context-free grammar determines permissible values of linguistic variable and attributed-grammar semantics gives the possibility to add modifiers (hedges) [Zadeh, 1992].

In approximate reasoning, there are two important inference rules, a *generalized modus ponens* (GMP) which is a forward reasoning rule and *a generalized modus tollens* (GMT) which is a backward reasoning rule. They are defined in the following.

Definition 2.1.24 (*Generalized Modus Ponens*) GMP is defined as [Zadeh, 1992; Zimmermann, 1993]

premise:	x is A'
implication :	If x is A , then y is B
consequence :	y is B'

where A , A' , B , B' are fuzzy sets and x and y are symbolic names for objects.

Definition 2.1.25 (*Generalized Modus Tollens*) GMT is defined as the following inference [Zadeh, 1992; Zimmermann, 1993]

premise:	y is B'
implication:	If x is A , then y is B
consequence:	x is A'

where *A*, *A*', *B*, *B*' are again fuzzy sets.

Generalized modus ponens can be represented by a compositional rule of inference (a special case of GMP)

$$A' \circ R = B' \tag{2.24}$$

where $A' \circ R$ is a composition of the form (2.17). Similarly, Generalized modus tollens can be represented by

$$B' \circ R = A' \tag{2.25}$$

where the relation R can be formed by setting

$$\mu_R(y,x) = \min(\mu_{\bar{B}}(y), \mu_{\bar{A}}(x)) = \min(1-\mu_B(y), 1-\mu_A(x)) \tag{2.26}$$

Here the relation was created by min-operation. It could be any other operation that suits the problem.

Definition 2.1.26 (*fuzzy implication*) Let A and B be fuzzy sets in U and V , respectively. A fuzzy implication, $A \to B$, is a relation in $U \times V$ and it can be understood as a fuzzy IF-THEN rule.

The following interpretations for the fuzzy implication are possible:

Fuzzy conjunction: $\mu_{A\to B}(u,v) = \mu_A(u) * \mu_B(v)$ (2.27)

Fuzzy disjunction: $\mu_{A\to B}(u,v) = \mu_A(u) + \mu_B(v)$ (2.28)

Material implication: $\mu_{A\to B}(u,v) = \mu_{\bar{A}}(u) + \mu_B(v)$ (2.29)

Propositional calculus: $\mu_{A\to B}(u,v) = \mu_{\bar{A}}(u) + \mu_{A*B}(v)$ (2.30)

Generalization of modus ponens:

$$\mu_{A\to B}(u,v) = \sup\{c \in [0,1] \mid \mu_A(u) * c \le \mu_B(v)\} \quad (2.31)$$

Generalization of modus tollens:

$$\mu_{A\to B}(u,v) = \inf\{c \in [0,1] \mid \mu_B(v) + c \le \mu_A(u)\} \quad (2.32)$$

In each interpretation of fuzzy implication (2.27) - (2.32) we may use different types of t-norms or t-conorms.

Larsen $x \to y = xy$

Lukasiewicz $x \to y = \min\{1, 1 - x + y\}$

Mamdani $x \to y = \min\{x, y\}$

Standard Strict $x \to y = \begin{cases} 1, & \text{if } x \le y \\ 0, & \text{otherwise} \end{cases}$

G¨odel $x \to y = \begin{cases} 1, & \text{if } x \le y \\ y, & \text{otherwise} \end{cases}$

Gaines $x \to y = \begin{cases} 1, & \text{if } x \le y \\ y / x, & \text{otherwise} \end{cases}$

Kleene-Dienes $x \to y = \max\{1 - x, y\}$

Kleene-Dienes-Luk. $x \to y = 1 - x + xy$

Therefore there exists a number of different possible ways to interpret IF-THEN rules.

2.1.5 Fuzzy Rules

Fuzzy logic was originally meant to be a technique for modeling the human thinking and reasoning, which is done by fuzzy rules. This idea has been replaced by the thought that the fuzzy rules form an interface between humans and computers [Brown & Harris, 1994]. Humans explain their actions and knowledge using linguistic rules and fuzzy logic is used to represent this knowledge on computers. There are three principal ways to obtain these rules:

1. human experts provide rules
2. data driven: rules are formed by training methods
3. combination of 1. and 2.

The first way is the ideal case for fuzzy systems. Although the rules are not precise, they contain important information about the system. In practice human experts may not provide a sufficient number of rules and especially in the case of complex systems the amount of knowledge may be very small or even non-existent. Thus the second way must be used instead of the first one (provided the data is available). The third way is suited for the cases when some knowledge exists and sufficient amount of data for training is available. In this case fuzzy rules got from experts roughly approximate the behavior of the system, and by applying training this approximation is made more precise. Rules provided by the expert's form an initial point for the training and thus exclude the necessity of random initialization and diminish the risk of getting stuck in a local minimum (provided the expert knowledge is good enough).

It has been shown in [Mouzouris, 1996] that linguistic information is important in the absence of sufficient numerical data but it becomes less important as more data become available.

Fuzzy rules define the connection between input and output fuzzy linguistic variables and they can be seen to act as associative memories. Resembling inputs are converted to resembling outputs. Rules have a structure of the form:

IF (antecedent) **THEN** (consequent) (2.33)

In more detail, the structure is

IF (x_1 is A_1^i **AND** … **AND** x_d is A_d^i **THEN** (y is B^i) (2.34)

where A_i^j $_i$ and B^i are fuzzy sets (they define complete fuzzy partitions) in $U \subset \mathbf{R}$ $V \subset \mathbf{R}$, respectively. Linguistic variable **x** is a vector of dimension d in $U_1 \times ... \times U_d$ and linguistic variable $y \in V$. Vector **x** is an input to the fuzzy system and *y* is an output of the fuzzy system. Note that B^i can also be a singleton (consequence part becomes: ...**THEN** (y is z^i)). Further, if fuzzy system is used as a classifier, the consequence part becomes: ...**THEN** class is c.

A fuzzy rule base consists of a collection of rules $\{R^1, R^2, ..., R^M\}$, where each rule R^i can be considered to be of the form (2.34). This does not cause a loss of generality, since multi-input-multi-output (MIMO) fuzzy logic system can always be decomposed into a group of multi-input-single-output (MISO) fuzzy logic systems. Furthermore (2.34) includes the following types of rules as a special case [Wang, 1994]:

- **IF** (x_1 is A_1^i **AND … AND** x_k is A_k^i) **THEN** (y is B^i), where $k < d$
- **IF** ((x_1 is A_1^i **AND …** **AND** x_k is A_k^i) **OR** (x_{k+1} is A_{k+1}^i **AND … AND** x_d is A_d^i)) **THEN** (y is B^i)
- y is B^i
- (y is B^i) **UNLESS** (x_1 is A_1^i **AND** ... **AND** x_d is A_d^i)
- non-fuzzy rules

In control systems the production rules are of the form

$$\textbf{IF} \text{ <process state> } \textbf{THEN} \text{ <control action>} \tag{2.35}$$

where the <process state> part contains a description of the process output at the kth sampling instant. Usually this description contains values of error and change-of-error. The <control action> part describes the control output (change-in-control) which should be produced given the particular <process state>. For example, a fuzzified PI- controller is of the type (2.35). Important properties for a set of rules are completeness, consistency and continuity. These are defined in the following.

Definition 2.1.27 (*completeness*) A rule base is said to be complete if any combination of input values results in an appropriate output value:

$$\forall \mathbf{x} \in \mathbf{X}: \ \text{hgt}(\text{out}(\mathbf{x})) > 0 \tag{2.36}$$

Definition 2.1.28 (*inconsistency*) A rule base is inconsistent if there are two rules with the same rule antecedent but different rule consequences.

This means that two rules that have the same antecedent map to two non-overlapping fuzzy output sets. When the output sets are non-overlapping, then there is something wrong with the output variables or the rules are inconsistent or discontinuous.

Definition 2.1.29 (*continuity*) A rule base is continuous if the neighboring rules do not have fuzzy output sets that have empty intersection.

2.1.6 Fuzzy Inference

As mentioned earlier, a fuzzy IF-THEN rule is interpreted as a fuzzy implication. The two most common operation rules of fuzzy implications are

- Mini-operation rule (Mamdani implication):

$$\mu_{A\to B}(\mathbf{x}, y) = \min\{\mu_A(\mathbf{x}), \mu_B(y)\} \tag{2.37}$$

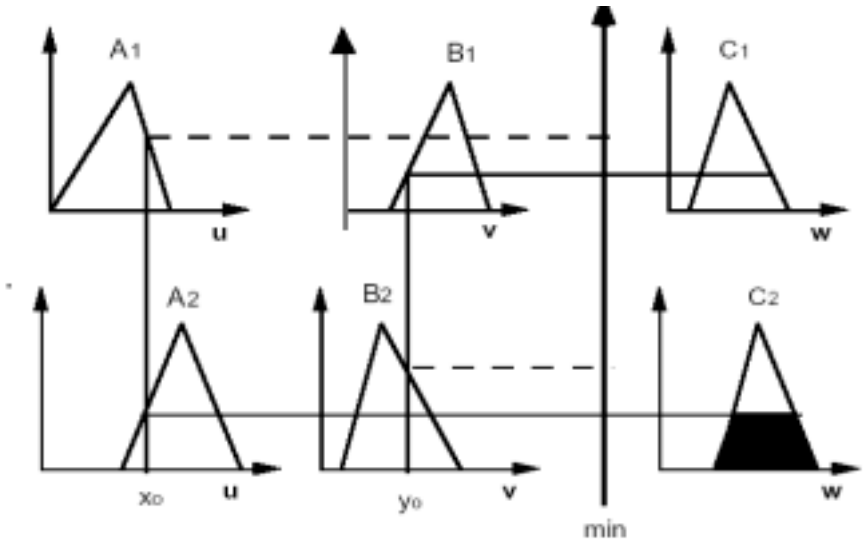


Figure 2.10 Inference with Mamdani's implication operator

- Product-operation rule (Larsen implication):

$$\mu_{A\to B}(\mathbf{x}, y) = \mu_A(\mathbf{x})\ \mu_B(y) \tag{2.38}$$

where

$$\mu_A(\mathbf{x}) = \mu_{A_1^i \times \ldots \times A_d^i}(\mathbf{x}) = \min\left\{\mu_{A_1^i}(x_1), \ldots, \mu_{A_d^i}(x_d)\right\} \tag{2.39}$$

or

$$\mu_A(\mathbf{x}) = \mu_{A_1^i \times \ldots \times A_d^i}(\mathbf{x}) = \mu A_1^i(x_1) \ldots \mu A_d^i(x_d) \tag{2.40}$$

The choice of operation rules (2.37) and (2.38) is justified by notion that they satisfy more intuitive criteria of GMP and GMT than their competitors [Wang, 1994].

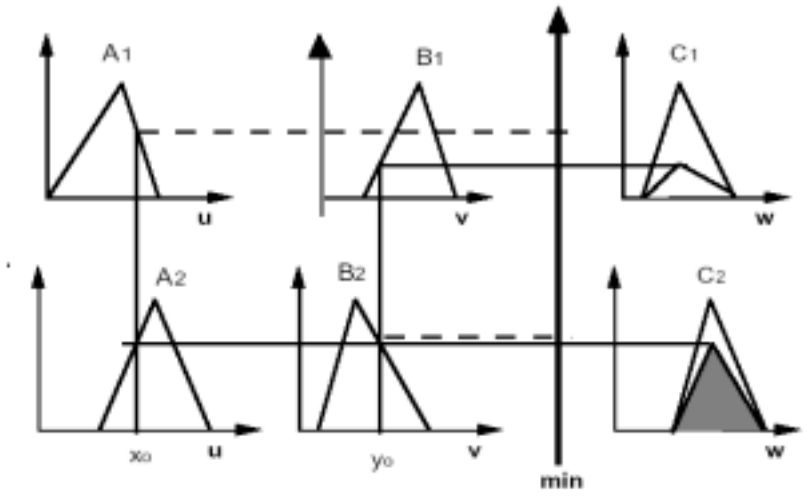


Figure 2.11 Inference with Larsen's product operation rule

Each fuzzy IF-THEN rule determines a fuzzy set in output space:

$$\mu_{B^{i'}}(y) = \sup_{x \in U} [\mu_{A^i \to B^i}(\mathbf{x}, y) * \mu_{A^i}(\mathbf{x})] \tag{2.41}$$

where A' is the input to the fuzzy inference, $B^{i'}$ is the set in the output space determined by the composition and * is t-norm. The output of inference engine is either M fuzzy sets in the form (2.41) or the union of M fuzzy sets

$$\mu_{B'}(y) = \mu_{B^{1'}}(y) + ... + \mu_{B^{M'}}(y) \tag{2.42}$$

- Tsukamoto implication. All linguistic terms are supposed to have monotonic membership functions. The firing levels of the rules, denoted by $\alpha_i, i = 1,2,$ are computed by

$$\alpha_1 = A_1(x_0) \wedge B_1(y_0), \alpha_2 = A_2(x_0) \wedge B_2(y_0)$$

In this mode of reasoning the individual crisp control actions z_1 and z_2 are computed from the equations $\alpha_1 = C_1(z_1), \quad \alpha_2 = C_2(z_2)$ and the overall crisp control action is expressed as

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2}$$

i.e. z_0 is computed by the discrete Center-of- Gravity method. If we have n rules in our rule-base then the crisp control action is computed as

$$z_0 = \frac{\sum_{i=1}^{n} \alpha_i z_i}{\sum_{i=1}^{n} \alpha_i}$$

where α_i is the firing level and z_i is the (crisp) output of the i-th rule, $i = 1,..., n$

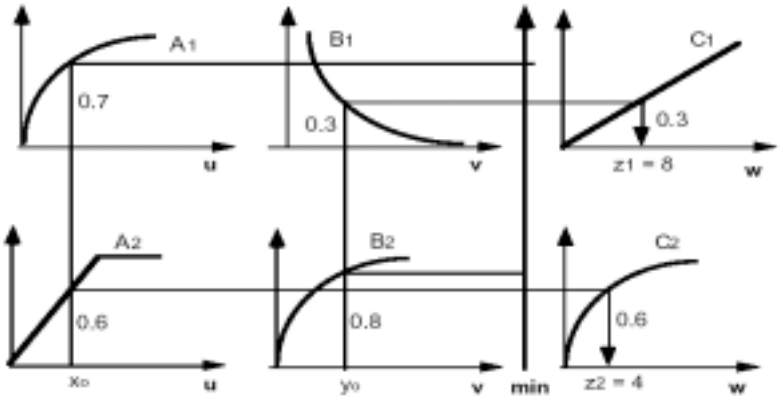


Figure 2.12 Sample of Tsukamoto's inference mechanisms

- Sugeno implication **.** Sugeno and Takagi use the following architecture

R$_1$**:** if x is A_1 and y is B_1 then $z_1 = \alpha_1 x + b_1 y$
also
R$_2$**:** if x is A_2 and y is B_2 then $z_2 = \alpha_2 x + b_2 y$
fact: x is $\overline{x}_0$ and y is $\overline{y}_0$

---

cons.: z_0

The firing levels of the rules are computed by $\alpha_1 = A_1(x_0) \wedge B_1(y_0)$, $\alpha_2 = A_2(x_0) \wedge B_2(y_0)$ then the individual rule outputs are derived from the relationships $z_1^* = \alpha_1 x_0 + b_1 y_0$, $z_2^* = \alpha_2 x_0 + b_2 y_0$

and the crisp control action is expressed as $$z_0 = \frac{\alpha_1 z_1^* + \alpha_2 z_2^*}{\alpha_1 + \alpha_2}$$

If we have n rules in our rule-base then the crisp control action is computed as

$$z_0 = \frac{\sum_{i=1}^{n} \alpha_i z_i^*}{\sum_{i=1}^{n} \alpha_i}$$

where α_i denotes the firing level of the i -th rule, $i = 1,...,n$

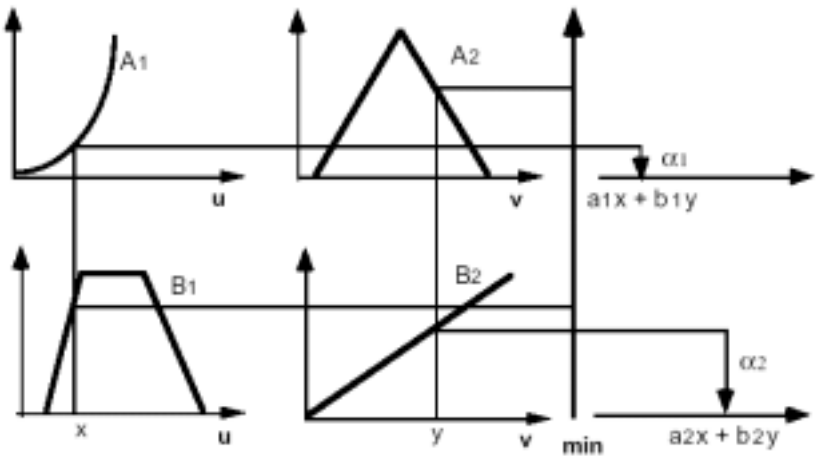
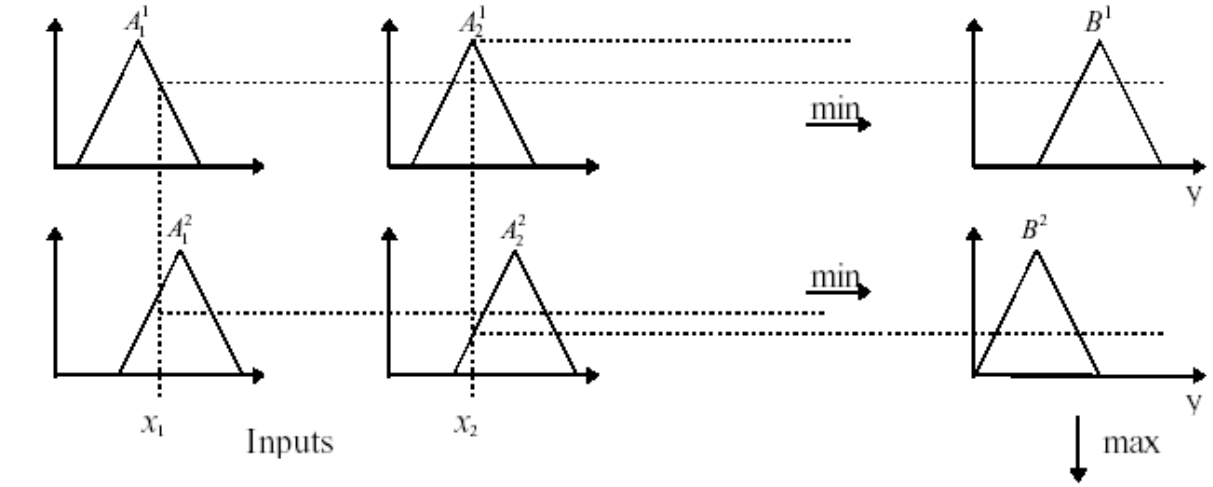


Figure 2.13 Sample of Sugeno's inference mechanism

Example 2.1.4 Let the input be a crisp value and let the mini-operation rule (2.37) be used with (2.39). When (2.42) is applied we get an output set illustrated in Fig. 2.14.



IF (x_1 is A_1^1 **AND** x_2 is A_2^1) **THEN** (y is B^1)

IF (x_1 is A_1^2 **AND** x_2 is A_2^2) **THEN** (y is B^2)

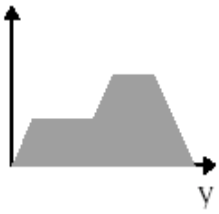


Figure 2.14 An example of fuzzy inference

2.1.7 Fuzzifier and Defuzzifier

The *fuzzifier* maps a crisp point **x** into a fuzzy set A'. When an input is a numerical value, the fuzzy set is simply given by a singleton:

$$\mu_{A'}(x) = \begin{cases} 1, & \text{if} \quad x = x' \\ 0, & \text{otherwise} \end{cases} \tag{2.43}$$

where x' is the input. Fuzzifier of the form (2.43) is called a *singleton fuzzifier*. If the input contains noise it can be modeled by using fuzzy number. Such fuzzifier could be called a *nonsingleton fuzzifier*. Because of simplicity, singleton fuzzifier is preferred to nonsingleton fuzzifier.

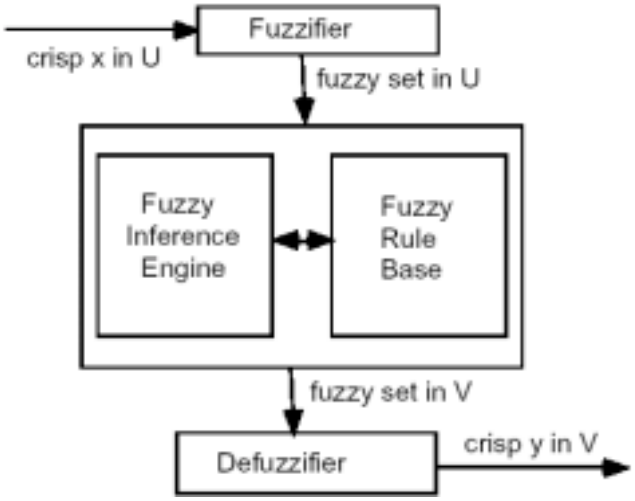


Figure 2.15 Fuzy logic controller

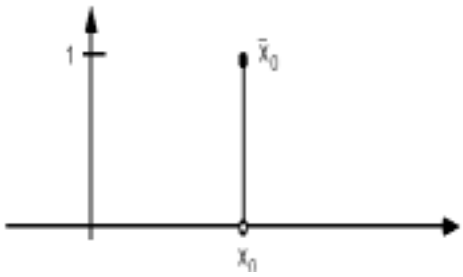


Figure 2.16 Fuzzy singleton as fuzzifier

The *defuzzifier* maps fuzzy sets to a crisp point. Several defuzzification methods have been suggested. The following five are the most common:

- **Center of Gravity** (CoG): In the case of 1-dimensional fuzzy sets it is often called the Center of Area (CoA) method. Some authors (for example, [Driankov et. al., 1993]) regard CoG and CoA as a same method, when other (for example, [Jager, 1995]) give them different forms. If CoA is calculated by dividing the area of combination of output membership functions by two and then taking from the left so much that we get an area equal to the right one, then it is clearly a distinct method. CoG determines center of gravity of the mass, which is formed as a combination of the clipped or scaled output fuzzy membership functions. The intersection part of these membership functions can be taken once or twice into calculation. Driankov [1993] separates the Center of Gravity methods such that, if the intersection is calculated once the method is Center of Area and if it is calculated twice the method is called Center of Sums (CoS). In Fig. 2.17 defuzzified value obtained by CoS is slightly smaller than obtained by the CoA -method.
- **Height method** (HM): Can be considered as a special case of CoG, whose output membership functions are singletons. If symmetric output sets are used in CoG, they have the same centroid no matter how wide the set is and CoG reduces to HM. HM calculates a normalized weighted sum of the clipped or scaled singletons. HM is sometimes called Center average defuzzifier or fuzzy-mean defuzzifier.
- **Middle of Maxima** (MoM): Calculates the center of maximum in clipped membership function. The rest of the distribution is considered unimportant.
- **First of Maxima** (FoM): As MoM, but takes the leftmost value instead of center value.

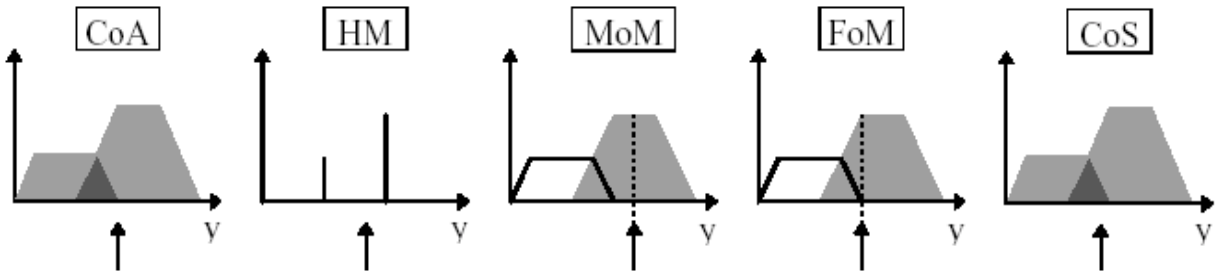


Figure 2.17 Defuzzification methods

Defuzzification methods can be compared by some criteria, which might be the continuity of the output and the computational complexity. HM, CoA and CoS produce continuous outputs. The simplest and quickest method of these is the HM method and for large problems it is the best choice. The fuzzy systems using it have a close relation to some well-known interpolation methods (we will return to this relation later).

The maximum methods (MoM, FoM) have been widely used. The underlying idea of MoM (with max-min inference) can be explained as follows. Each input variable is divided into a number of intervals, which means that the whole input space is divided into a large number of *d*-dimensional boxes. If a new input point is given, the corresponding value for *y* is determined by finding which box the point falls in and then returning the average value of the corresponding *y*-interval associated with that input box. Because of the piecewise constant output, MoM is inefficient for approximating nonlinear continuous functions. Kosko has shown [Kosko, 1997] that if there are many rules that fire simultaneously, the maximum function tends to approach a constant function. This may cause problems especially in control.

The property of CoS is that the shape of final membership function used as a basis for defuzzification will resemble more and more normal density function when the number of functions used in summation grows. Systems of this type that sum up the output membership functions to form final output set are called *additive fuzzy systems*.

2.1.8 The General Relation between Probability and Fuzziness

The relation between probability and fuzziness is perhaps best described by Kosko in [Kosko, 1990]. He uses *n*-dimensional hypercubes to illustrate this relation and shows that the probability distributions occupy points along the diagonal of $(n-1)$ - dimensional hyper-tetrahedron of the fuzzy set (Fig. 2.18). We notice that there are far more fuzzy sets which are not probability distributions, and which fill the hypercube. In fuzzy hypercube, each axis corresponds to the membership grade of a fuzzy set.

Joslyn has compared fuzzy logic, probability and possibility in the basis of possibility theory [Joslyn, 1994]. He claims, that although probability and possibility are logically independent, their measures both arise from Dempster-Shafer evidence theory as fuzzy sets defined on random sets and their distributions are fuzzy sets.

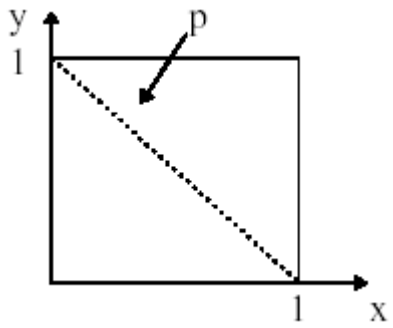


Figure 2.18 Probability in the fuzzy 2-dimensional hypercube for two sets. In the diagonal line are the points that's sum to unity. (If the possibility distributions had been drawn, they would be the upper and the right edge of the hypercube).

2.2 Different Fuzzy Systems

Fuzzy system is a set of rules, which converts inputs to outputs. By allowing partial memberships, it is possible to represent a smooth transition from one rule to another, i.e., to interpolate between the rules. In the following two commonly used fuzzy systems, which have importance in the later chapters, are presented.

2.2.1 Takagi-Sugeno's Fuzzy System

In fuzzy systems, two types of rules can be distinguished: rules of the type presented earlier (2.34) called Mamdani rules, and Takagi-Sugeno rules of the type (2.44).

$$\textbf{IF } (x_1 \text{ is } A_1^1 \textbf{ AND ... AND } x_d \text{ is } A_d^i) \textbf{ THEN } y^i = f^i(x_1,...,x_d) \tag{2.44}$$

where the consequences of the fuzzy rules are functions of the input vectors **x**. Index i means the ith rule and As are fuzzy sets. Sugeno used linear combinations of input values as output functions y

$$y^j = c_0^j + \sum_{k=1}^{d} c_k^j x_k \tag{2.45}$$

where c_{kj} are real-valued parameters and specific for each rule. Function can also be nonlinear with respect to inputs. A weighted sum is used as defuzzification method:

$$y\ (\mathbf{x}) = \frac{\sum_{j=1}^{k} y^j \prod_{i=1}^{d} \mu_{A_i^j}(x_i)}{\sum_{j=1}^{k} \prod_{i=1}^{d} \mu_{A_i^j}(x_i)} \tag{2.46}$$

since it provides a smooth interpolation of the local models. This type of fuzzy system has been successfully used in control problems, for example in the control of a model car [Sugeno & Murakami, 1984]. The output of the system is a weighted sum of functions. Thus (2.46) interpolates between different linear functions.

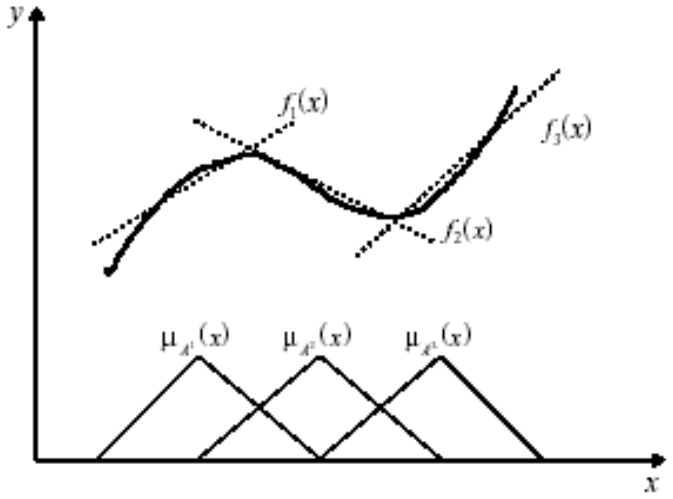


Figure 2.19 A function f defined by TSK -model. Dashed lines represent local models

2.2.2 Mendel-Wang's Fuzzy System

Mendel-Wang's fuzzy logic system is presented as

$$y\ (\mathbf{x}) = \frac{\sum_{l=1}^{k} w^l (\prod_{i=1}^{d} \mu_{A_i^l}(x_i)}{\sum_{l=1}^{k} (\prod_{i=1}^{d} \mu A_i^l(x_i)} \tag{2.47}$$

where w^l is a place of output singleton. The membership function $\mu_{A_i^l}(x_i)$ corresponds to the input x_i of the rule l . In short, the *and* connective in the premise is carried out by a product and

defuzzification by the Height Method (a special case of Center of Gravity method). The rules are of the Mamdani type. The fuzzy basis function is of the form

$$b_j(\mathbf{x}) = \frac{\prod_{i=1}^{d} \mu_{F_i^j}(x_i)}{\sum_{i=1}^{k} (\prod_{i=1}^{d} \mu_{F_i^j}(x_i))} \tag{2.48}$$

where the denominator normalizes the product of membership functions so that the sum of basis functions yields one at each point. Functions (2.48) are called basis functions even though they are not always orthogonal. The output of (2.47) equals a discrete expectation of the k then-part set centroids w_i. Basis functions could also be of the form

$$b_j(\mathbf{x}) = \frac{\min\left\{\mu_{F_i^j}(x_i) \middle| x_i \in X_i, i = 1,...,d\right\}}{\sum_{i=1}^{k} \min\left\{\mu_{F_i^j}(x_i) \middle| x_i \in X_i, i = 1,...,d\right\}} \tag{2.49}$$

if mini-inference rules were used. However, modern fuzzy systems dispense with min and max. They consist of only multiplies, additions, and divisions [Kosko, 1997]. System (2.47) can be expressed as a nonlinear mapping which is formed by a weighted sum

$$y(\mathbf{x}) = \sum_{i=1}^{k} w_i b_i(\mathbf{x}) \tag{2.50}$$

Although (2.50) does not look like a fuzzy system, it can be interpreted as such, because it is formed by using membership functions, t-norms and t-conorms. In the following, the use of fuzzy system of the form (2.47) is justified.

Why singleton fuzzifier and singleton output membership functions?

Singleton fuzzifier is simpler and more used than its nonsingleton counterpart. However, it is assumed that nonsingleton fuzzifier may work better in a noisy environment as it models uncertainty, imprecision or inaccuracy in the inputs [Wang, 194; Jager, 1995]. If the output membership functions were continuous, they would have to be discretized in defuzzification. As a consequence, t-norm (product) and t-conorm (sum) would have to be calculated at every discretizing point [Viljamaa, 1996].

Why product for and -operation?

The use of product-rule is justified by the axioms for aggregation operations [Klir, 1988]. With aggregation operation several fuzzy sets are combined to produce a single set as described earlier. Aggregation operation is defined by $h : [0,1] \to [0,1]$ and the axioms are

1. $h(\mathbf{0})=0$ and $h(\mathbf{1})=1$
2. h is monotonic nondecreasing in all its arguments: If $x_i \geq y_i$, then $h(x_i) \geq h(y_i)$

.

The following additional axioms are not essential, but they are usually taken into account

1. h is continuous
2. h is symmetric in all its arguments

The most significant advantage of product over minimum operator is the fact that product retains more information. This can be seen, for example, in the case of twodimensional input space. When the product operation is used, both inputs will have an effect on the output. If the min operation is used, only one input has effect on the output. It is extremely difficult to design a fuzzy system based on max-min operations such that the interpolation properties of the resulting system can be well understood. It can be only known that the value in the interior of an interval lies between the values at either end [Brown & Harris, 1994].

Why Height Method for defuzzification?

Driankov has compared in [Driankov et. al., 1993] six best known defuzzification methods. The result of this comparison is that Height Method, Center of Sums and Center of Area satisfy equal number of criteria and more than the other methods. Criteria are continuity, disambiguity (i.e., can produce a crisp output for every combination of output membership functions), plausibility (output lies approximately in the middle of support and has a high degree of membership), computational complexity and weight counting (weight information is not lost). They all are, in fact, variations of Center of Gravity method. These three "best methods" could be put in order by comparing the computational complexity. The order would be

1. Height Method
2. Center of Sums
3. Center of Area

This does not say that Height Method is the best choice for all applications, but it does say that it is a good general choice if there is no reason, i.e., criterion other than mentioned before, to use some other method. In practice, triangular (or trapezoidal) functions are more commonly chosen as membership functions than the Gaussians because of their low computational requirements. However, the Gaussians have a couple of important properties:

1. produces smooth mapping
2. universal approximation property can be easily proven
3. central limit theorem: distributions tend to normal (normal distributions can be approximated well by Gaussian -type basis functions)

One additional reason to use system of the form (2.47) is that it has a resemblance to other methods in approximation theory, interpolation theory (for example, finite element method interpolation) and neural networks.

Output as conditional average in scalar valued FLS (singleton outputs)

The convex functions $b_1(x),...,b_k(x)$ define a discrete probability density function $\mathbf{p}(X)$ for each input X, since

$$\begin{aligned} &1^\circ \quad b_i(x) \geq 0 \\ &2^\circ \quad \sum_i b_i(x) = 1 \end{aligned} \tag{2.51}$$

For each input the fuzzy system defines the expected value of the k output values w_i with respect to $\mathbf{p}(X)$:

$$y(x) = \sum_{i=1}^{k} w_i b_i(x) = \frac{\int w\mu_B(w)dw}{\int \mu_B(w)dw} = \int wp(w|x)dw \tag{2.52}$$

In training, weights in (2.52) can be replaced by targets. The output $y(x)$ defines a sort of conditional average. The uncertainty grows as the output must interpolate between many weights w_i. The uncertainty is minimal when $p_i(x) = 1$ for some i and $p_j(x) = 0$ for $j \neq i$. According to this view, FLS forms a final output set which can be interpreted as a probability density function from which the conditional average is calculated by a CoG -method. This is illustrated in Fig.2.20 for the symmetric output membership functions.

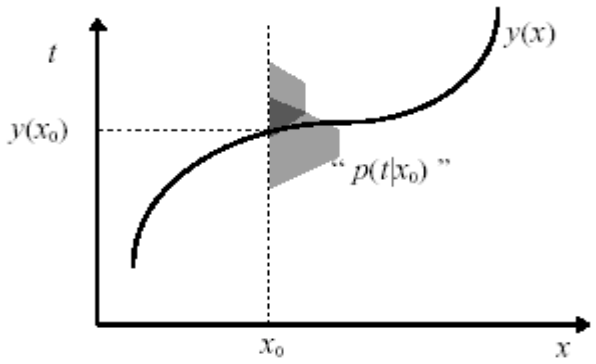


Figure 2.20 Output of FLS

2.2.3 Kosko's Standard Additive Model (SAM)

Kosko has presented an additive model SAM which tries to be as general fuzzy system as possible. It includes almost all fuzzy systems in practice [Kosko, 1997]. The form of SAM is

$$y(\mathbf{x}) = \frac{\sum_{i=1}^{m} d_i w_i a_i(\mathbf{x}) V_i}{\sum_{i=1}^{m} d_i a_i(\mathbf{x}) V_i} \tag{2.53}$$

The SAM stores m rules of the patch form $A_i \times B_i$. The input vector $\mathbf{x}$ belongs to the if-part fuzzy set $A_i \subset R^d$ to degree $a_i(\mathbf{x})$ in [0,1]. SAM uses the volume or area V_i of set B_i and its center of mass. d_i is a scalar rule weight. The larger V_i the more the global mapping acts like the local rule output w_i. The SAM reduces to the CoG model if the relative rule weights and volumes of the then part sets are ignored (e.g., set to some constant values, usually to unity). It can be easily seen that (2.47) is a special case of (2.53).

Kosko proved in [Kosko, 1997] that *all* center of gravity fuzzy systems are probabilistic systems and they compute a conditional expectation (conditional average) and thus compute a mean-squared optimal non-linear estimator. Fuzzy system can be regarded as a model-free statistical estimator since it does not use mathematical model but approximates the system with conditional average:

$$\begin{aligned} y(\mathbf{x}) &= \mathrm{CoG}(B(\mathbf{x})) \\ &= \frac{\int t b(\mathbf{x},t)\, dt}{\int b(\mathbf{x},t)\, dt} \\ &= \int t p(t|\mathbf{x})\, dt \\ &= \langle t|\mathbf{x} \rangle \end{aligned} \tag{2.54}$$

where B consists of the clipped or scaled output sets $B_j' = \prod_i \mu_{A_i^j}(x_i) B_j$, where B_j is the output set corresponding to (multidimensional) input set $\prod_i \mu_{A_i}$.

Each fuzzy output $y(\mathbf{x})$ has an uncertainty or conditional variance

$$\begin{aligned} \sigma^2(t|\mathbf{x}) &= \sum_{i=1}^{k} p_i(\mathbf{x}) \sigma_{B_i}^2 + \sum_{i=1}^{k} p_i(\mathbf{x})(c_i - y(\mathbf{x}))^2 \\ &= \frac{\int (t - \langle t|\mathbf{x} \rangle)^2 b(\mathbf{x}, t) dt}{\int b(\mathbf{x}, t) dt} \end{aligned} \tag{2.55}$$

where $\sigma_{B_i}^2 = \int (t-c)^2 p_{B_j}(t)dt$ and $p_i(\mathbf{x}) = \dfrac{V_i(\mathbf{x})}{\sum_j V_j} = \dfrac{\int b_i(\mathbf{x},t)dt}{\sum_j \int b_j(\mathbf{x},t)dt}$ V_i is the volume of output set $B_i^{'}$.

Figure 2.21 Volume of output set $B_i^{'}$.

2.3 Approximation capability

Universal approximator can approximate any continuous function on compact sets to any degree of accuracy. Wang proved in [Wang, 1992] the following important theorem:

Theorem 2.1.2 (*Universal Approximation Theorem*) For any given real continuous function *g* on compact set U $R_d \in$ and arbitrary $\varepsilon > 0$, there exists a fuzzy logic system f in the form of (2.47), where the membership functions are bell-shaped Gaussians, such that

$$\sup_{x \in U} \left| f(\mathbf{x}) - g(\mathbf{x}) \right| < \varepsilon \qquad (2.56)$$

If we regard the fuzzy system as a discretization of function *g* and consider it as an interpolation problem, we notice that the approximation can be made better by increasing "knots", i.e., rules. Conditions for interpolation are that 1) fuzzy basis functions sum to unity at every point, 2) if no more than two fuzzy sets overlap, the membership functions must be convex and normal, and 3) the rule base is complete [Jager, 1995].

The importance of this theorem is that it shows that the class of all fuzzy systems is a universal approximator. Wang also proved this theorem for square integrable functions. Thus these theorems justify the use of fuzzy logic systems to almost any modeling problem.

Guideline for the proof of Theorem 2.1.2: Wang used Stone-Weierstrass Theorem (Rudin, 1960) to prove the Universal Approximation Theorem. Let Y be the set of all fuzzy logic systems of the form (2.47) on compact set U . The Stone-Weierstrass Theorem can be used if it can be showed that 1) Y is algebra, 2) Y separates points in U , and 3) Y does not vanish at any point of U . Thus, the proof consists of the following steps

- Y is an algebra, if $f_1, f_2 \in Y$, then follows that $f_1(\mathbf{x}) + f_2(\mathbf{x}) \in Y$, $f_1(\mathbf{x}) f_2(\mathbf{x}) \in Y$ and $cf_1(\mathbf{x}) \in Y$, where c is an arbitrary real valued coefficient. This is easy to prove for the FLS of the form (2.47) with the Gaussian membership functions.
- Y separates points on U , if $\forall$ $\mathbf{x},\mathbf{y} \in U$, $\mathbf{x} \neq \mathbf{y}$ there exists $f \in y$ such that . It is shown that

$$\frac{w_1 + w_2 \prod_{i=1}^{d} \exp(-(x_i - y_i)^2 / 2)}{1 + \prod_{i=1}^{d} \exp(-(x_i - y_i)^2 / 2)} \neq \frac{w_2 + w_1 \prod_{i=1}^{d} \exp(-(x_i - y_i)^2 / 2)}{1 + \prod_{i=1}^{d} \exp(-(x_i - y_i)^2 / 2)}$$

- Y vanishes at no point of U . This is proved by choosing $y^j > 0$ for all j .

It is obvious that (2.47) is a real continuous function. Theorem 2.1.2 is a direct consequence of Stone-Weierstrass Theorem.

In the same year, 1992, also Kosko proved that fuzzy systems are universal approximators. He called the theorem as Fuzzy Approximation Theorem (FAT). His proof was for additive fuzzy rule systems and was based on the notion illustrated in the Fig 2.22. FAT -theorem says that a curve can always be covered by a finite number of fuzzy patches (fuzzy rules) which can overlap each other. All fuzzy systems give some type of patch covering. If more precise cover is needed, more patches are added. If crisp output values are required, fuzzy logic system calculates the average of overlapping patches. Although the principle is simple, the proof is not. Techniques of topology and measurement theory must be utilized to prove the theory [Kosko, 1992]. Note the resemblance with kernel regression methods, in which the curve is also covered by patches (Gaussian kernel functions centered on data points) and in which the curve is approximated by the conditional average. More on this relation later.

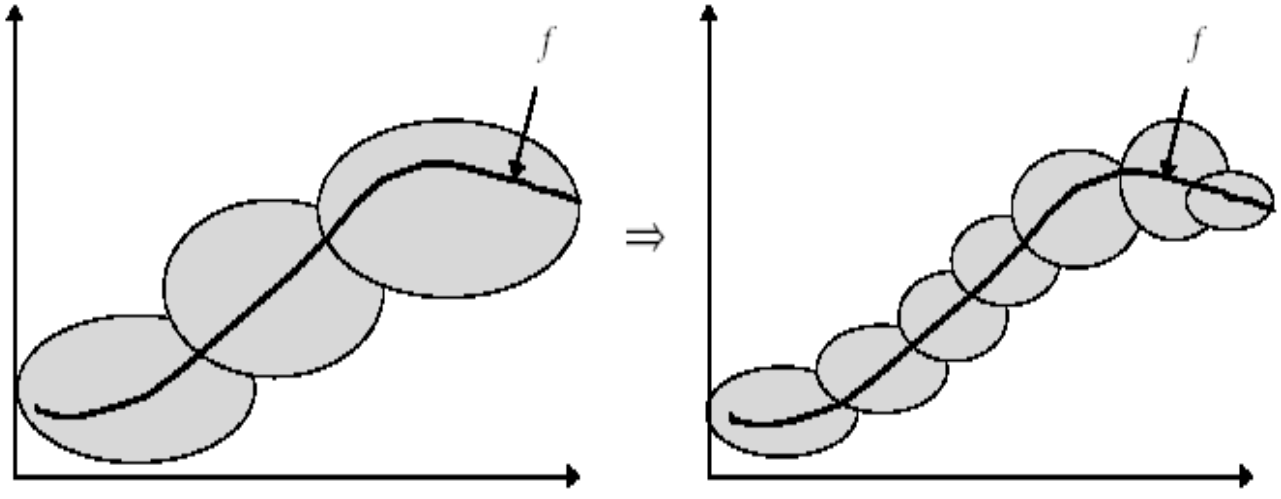


Figure 2.22 By increasing "rule patches" and making them smaller, FLS can approximate any continuous function to any accuracy. The more imprecise the rules the larger the patches. If neural or statistical clustering algorithms are used for learning, the large patches mean that data have been sparse or noisy.

Buckley proved the universal approximation theorem for Sugeno type of fuzzy controllers and Castro & Delgado proved it for fuzzy rule systems. Castro & Delgado showed that the fuzzy systems characterized by 1) a fixed fuzzy inference, 2) a certain kind of fuzzy relation that satisfies a smooth property, and 3) a defuzzification method, are universal approximators. They also showed that there exists an optimal fuzzy system for a wide variety of problems. This result does not, however, say, how we can design a fuzzy system for a particular problem. [Castro & Delgado, 1996].

2.4 Different Interpretations of Fuzzy Sets

One of the main difficulties of fuzzy logic has been with the meaning of membership functions. Lack of consensus on that meaning has created some confusion. Depending on the interpretation of fuzziness, the membership functions are chosen based on that interpretation. The more detailed description of the different interpretations can be found in [Bilgiç & Türksen, 1997].

- **The likelihood view**:

Sources of fuzziness are errors in measurement (statistical viewpoint), incomplete information and interpersonal contradictions. If the information is perfect, there cannot be any fuzziness and the membership functions are threshold functions. Likelihood view can be explained by a following illustrative example:

Example 2.1.5 We have some imprecise measurement of room temperature (which is within 20 2 ± degrees Celsius). Measurements construct an error curve around 20 degrees. If the temperature lies in 90% of the cases in the quantization interval, we would say that the likelihood that the label, for example ‘warm’, is assigned to the room temperature is 0.9:

$$\mu_{warm}(20) = P(warm | x = 20) = 0.9$$

Max and min as proposed by Zadeh are not always supported for the calculation of conjunction and disjunction.

This is not the only way to treat errors in measurement. More commonly they are treated by conventional techniques of statistics (probabilistic Bayesian inference is regarded as such a technique). Researchers, who represent likelihood viewpoint, are Hisdal, Mabuchi and Thomas. Their writings can be found in [Hisdal, 1995; Mabuchi, 1992; Thomas, 1995].

- **Random set view**

Membership functions are viewed horizontally as a family of level cuts, i.e., α -cuts defined in (2.11). Each level-set is a set in the classical sense. In this view the membership function can be viewed as a uniformly distributed random set. To illustrate this view the example 2.1.5 is modified and interpreted as follows: 90% of the population defined an interval in the universe of discourse for the set ‘warm’ which contained *x*, and 30% defined intervals which did not contain x .

- **Similarity view**

Membership measures the degree of similarity of an element to the set in question. There is some prototype or perfect set, which is used in comparison. The comparison is made by calculating the distance between the object and the prototype. In this viewpoint membership function could be of the form [Zysno, 1981]:

$$\mu(x) = \frac{1}{1 + f(d_x)} \tag{2.57}$$

where d_x is the distance of the object from the ideal and f is a function of distance. Similarity requires a metric space in which it can be defined. Thus this view has connection to Measurement Theory. The rules in fuzzy systems that represent similarity view can be described as follows:

$$\textbf{IF}\ (x_1 \text{ is } p_1^i\ \textbf{AND}\ \ldots\ \textbf{AND}\ x_d \text{ is } p_d^i)\ \textbf{THEN}\ (y \text{ is } z^i) \tag{2.58}$$

which differ from (2.34) by the fact that fuzzy sets are replaced by reference values p_j (prototypes) and the membership function is the distance between x_j and p_j.

Interpretation of example 2.1.5: Room temperature is away from the prototypical set ‘warm’ to the degree 0.1.

- **The measurement view**

The elements in set can be compared to each other. The relationship $A(x) > A(y)$ says that x is more representative of the set A than y. Relative values can be used to get information about the relative degree of A.

Interpretation of example 2.1.5: When compared to other temperatures in set ‘warm’, measured room temperature is warmer than some in that set. A value of 0.9 is attached to it on some scale.

As contrary to likelihood view, max and min are justified to be used for conjunction and disjunction [Bilgiç & Türksen, 1997].

2.5 Different Ways to form Fuzzy Sets

This is a summary of methods described in [Bilgiç & Türksen, 1997].

- **Polling:** The question “Do you agree that x is A?” is stated to different individuals. An average is taken to construct the membership function. Answers are typically of yes/no type.
- **Direct rating:** “How A is x?” This approach supposes that the fuzziness arises from individual subjective vagueness. The person is made to classify an object over and over again in time in such a

way that it is hard for he/she to remember the past answers. The membership function is constructed by estimating the density function.

- **Reverse rating:** The question "Identify x which is A to the degree $\mu_A(x)$" is stated to an individual or a group of individuals. The responses are recorded and normally distributed distributions are formed (mean and variance are estimated).
- **Interval estimation:** The person is asked to give an interval that describes best the Aness of x. This is suited to random set -view of membership functions.
- **Membership exemplification:** "To what degree x is A?" Person may be told to draw a membership function that best describes A.
- **Pairwise comparison:** "Which is a better example of A, x_1 or x_2 and by how much?" The results of comparisons could be used to fill a matrix of relative weights and the membership function is found by taking the components of the eigenvector corresponding to the maximum eigenvalue [Chameau & Santamarina, 1987; Saaty, 1974].
- **Clustering methods:** Membership functions are constructed given a set of data. Euclidean norm is used to form clusters on data.
- **Neurofuzzy techniques:** Neural networks are used to construct membership functions.

An essential part of forming membership functions is the input space partitioning. In Fig. 2.23 (a) is a grid, which is fixed beforehand and it does not change later. Grid in (b) is set to some "initial value", for example to (a), and later it is tuned. Fuzzy clusters are best suited for classification problems, because they implement a similarity measure.

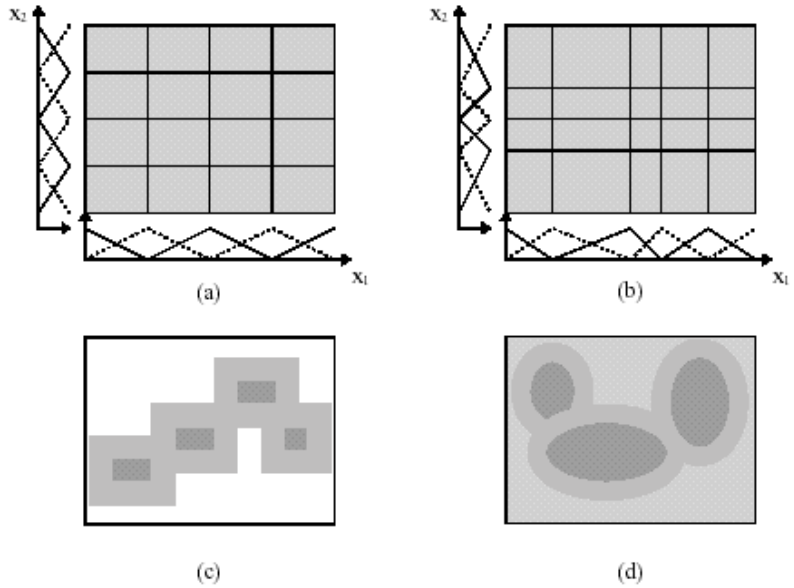


Figure 2.23 Input space partitioning: (a) fixed grid, (b) adaptive grid, (c), (d) fuzzy clusters.

Chapter 3

Neural Networks

3.1 Basic Concepts

Artificial neural networks (ANN) have been used for two main tasks: 1) function approximation and 2) classification problems. Neural networks offer a general framework for representing non-linear mappings

$$\hat{y} = f(\mathbf{x}, \theta) \tag{3.1}$$

or more generally

$$\hat{y}_k = f_k(\mathbf{x}, \theta) \tag{3.2}$$

where the form of the mapping is governed by a number of adjustable parameters θ ,which can be determined by using training methods explained in the chapter 5.

The classification problem can be regarded as a special case of function approximation, in which the mapping is done from the input space to a finite set of output classes. The arrangement of the network's nodes (neurons) and connections defines its architecture. Some of the network architectures are better suited for classification than for function approximation. Such networks are Self-organizing maps (SOM) developed by Kohonen [1990] and networks using learning vector quantization (LVQ) algorithms.

Artificial neural systems can be considered as simplified mathematical models of brain-like systems and they function as parallel distributed computing networks. However, in contrast to conventional computers, which are programmed to perform specific task, most neural networks must be taught, or trained. They can learn new associations, new functional dependencies and new patterns. Although computers outperform both biological and artificial neural systems for tasks based on precise and fast arithmetic operations, artificial neural systems represent the promising new generation of information processing networks.

The study of brain-style computation has its roots over 50 years ago in the work of McCulloch and Pitts (1943) and slightly later in Hebb's famous *Organization of Behavior* (1949). The early work in artificial intelligence was torn between those who believed that intelligent systems could best be built on computers modeled after brains, and those like Minsky and Papert (1969) who believed that intelligence was fundamentally symbol processing of the kind readily modeled on the *von Neumann* computer. For a variety of reasons, the symbol-processing approach became the dominant theme in *Artifcial Intelligence* in the 1970s. However, the 1980s showed a rebirth in interest in neural computing:

1982 Hopfield provided the mathematical foundation for understanding the dynamics of an mportant class of networks.
1984 Kohonen developed unsupervised learning networks for feature mapping into regular arrays of neurons.
1986 Rumelhart and McClelland introduced the backpropagation learning algorithm for complex, multilayer networks.

Beginning in 1986-87, many neural networks research programs were initiated. The list of applications that can be solved by neural networks has expanded from small test-size examples to large practical tasks. Very-large-scale integrated neural network chips have been fabricated.

In the long term, we could expect that artificial neural systems will be used in applications involving vision, speech, decision making, and reasoning, but also as signal processors such as filters, detectors, and quality control systems.

Definition. *Artificial neural systems, or neural networks, are physical cellular systems which can acquire, store, and utilize experiental knowledge.*

The knowledge is in the form of stable states or mappings embedded in networks that can be recalled in response to the presentation of cues.

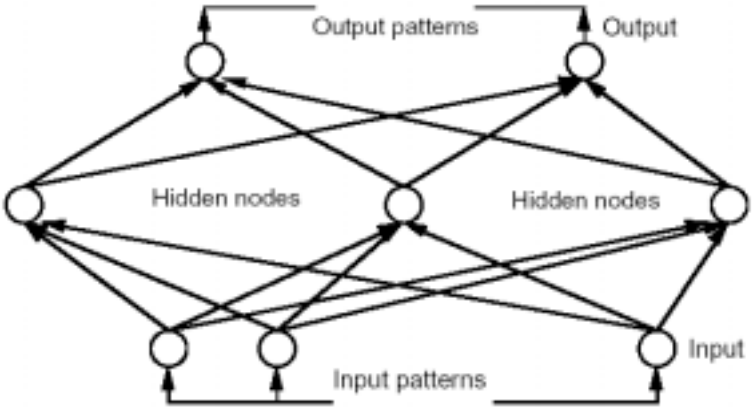


Figure 3.1 A multi-layer feedforward neural network.

The basic processing elements of neural networks are called *artificial neurons*, or simply *neurons* or *nodes*.

Each processing unit is characterized by an activity level (representing the state of polarization of a neuron), an output value (representing the firing rate of the neuron), a set of input connections, (representing synapses on the cell and its dendrite), a bias value (representing an internal resting level of the neuron), and a set of output connections (representing a neuron's axonal projections).

Each of these aspects of the unit are represented mathematically by real numbers.

Thus, each connection has an associated weight (synaptic strength) which determines the effect of the incoming input on the activation level of the unit.

The weights may be positive (excitatory) or negative (inhibitory).

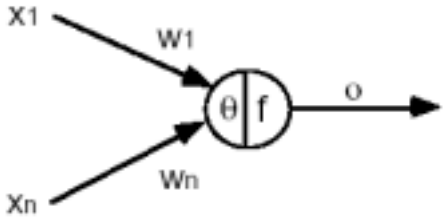


Figure 3.2 A processing element with single output connection

The signal flow from of neuron inputs, x_j, is considered to be unidirectionalas indicated by arrows, as is a neuron's output signal flow. The neuron output signal is given by the following relationship

$$o = f(\langle w, x \rangle) = f(w^T x) = f(\sum_{j=1}^{n} w_j x_j)$$

where $w = (w_1, ..., w_n)^T \in \mathbf{R}^n$ is the weight vector. The function $f(w^T x)$ is often referred to as an *activation (or transfer) function*. Its domain is the set of activation values, *net*, of the neuron model, we thus often use this function as $f(net)$. The variable *net* is defined as a scalar product of the weight and input vectors

$$net = \langle w, x \rangle = w^T x = w_1 x_1 + ... + w_n x_n$$

and in the simplest case the output value o is computed as

$$o = f(net) = \begin{cases} 1 & \text{if } w^T x \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where θ is called threshold-level and this type of node is called a *linear threshold unit*.

Network architectures can be categorized to three main types: *feedforward networks*, *recurrent networks* (*feedback networks*) and *self-organizing networks*. This classification of networks has been proposed by Kohonen [1990]. Network is feedforward if all of the hidden and output neurons receive inputs from the preceding layer only. The input is presented to the input layer and it is propagated forwards through the network. Output never forms a part of its own input. Recurrent network has at least one feedback loop, i.e., cyclic connection, which means that at least one of its

neurons feed its signal back to the inputs of all the other neurons. The behavior of such networks may be extremely complex.

Haykin divides networks into four classes [Haykin, 1994]: 1) *single-layer feedforward networks*, 2) *multilayer feedforward networks*, 3) *recurrent networks*, and 4) *lattice structures*. A lattice network is a feedforward network, which has output neurons arranged in rows and columns.

Layered networks are said to be *fully connected* if every node in each layer is connected to all the following layer nodes. If any of the connections is missing, then network is said to be *partially connected*. Partially connected networks can be formed if some prior information about the problem is available and this information supports the use of such a structure.

The following treatment of networks applies mainly to feedforward networks (single layer networks, MLP, RBF, etc.). The designation *n*-layer network refers to the number of computational nodes or the number of weight connection layers. Thus the input node layer is not taken into account.

3.1.1 Single-layer Feedforward Networks

The simplest choice of neural network is the following weighted sum

$$y(\mathbf{x}) = \sum_{i=0}^{d} w_i x_i \tag{3.3}$$

where d is the dimension of input space, $x_0 = 1$ and w_0 is the bias parameter. Input vector $\mathbf{x}$ can be considered as a set of activations of input layer. In classification problem (3.3) is called a discriminant function, because $y(\mathbf{x}) = 0$ can be interpreted as a decision boundary. Weight vector $\mathbf{w}$ determines the orientation of decision plane and bias parameter w_0 determines the distance from origin. In regression problems the use of this kind of network is limited; only $(d-1)$ -dimensional hyperplanes can be modeled. An example of single-layer networks is a linear associative memory, which associates an output vector with an input vector.

Extension of (3.3) to several classes is straightforward:

$$y_k(\mathbf{x}) = \sum_{i=0}^{d} w_{ki} x_i \tag{3.4}$$

where again $x_0 = 1$ and w_{k0} is the bias parameter. The connection from input i to output k is weighted by a weight parameter w_{ki} .

Figure 3.3 The simplest neural networks. Computation is done in the second layer of nodes

Functions of the form (3.3) can be generalized by using a (monotonic) linear or nonlinear activation function which acts on the weighted sum as

$$y(\mathbf{x}) = g(\sum_{i=0}^{d} w_i x_i) \tag{3.5}$$

where $g(v)$ is usually chosen to be a threshold function, piecewise linear, logistic sigmoid, sigmoidal or hyperbolic tangent function (tanh). The first neuron model was of this type and was proposed as early as in 1940's by McCulloch and Pitts.

- Threshold function (step function):

$$g(v) = \begin{cases} 1 & , \text{if } v \geq 0 \\ 0 & , \text{if } v < 0 \end{cases} \tag{3.6}$$

- Piecewise linear function (pseudolinear)

$$g(v) = \begin{cases} 1 & , \text{if } v \geq 0.5 \\ v & , \text{if } v > -0.5 \\ 0 & , \text{if } v \leq -0.5 \end{cases} \tag{3.7}$$

- Logistic sigmoid

$$g(v) = \frac{1}{1 + \exp(-v)} \tag{3.8}$$

Definition 3.1 A function $g : R \to R$ is called sigmoidal function, if the following criterion is satisfied

$$\begin{cases} \lim_{v \to -\infty} g(v) = 0 \\ \lim_{v \to \infty} g(v) = 1 \end{cases}$$

If $|v|$ is small then g approximates a linear function and (3.5) "contains approximately" a linear network as a special case. Functions of the form (3.5) which have logistic sigmoids as activation functions are used in statistics and are referred to as logistic discrimination functions. Sigmoid is a popular choice because it is monotone, bounded, nonlinear and has a simple derivative $g'(v) = g(v)(1 - g(v))$.

One way to generalize linear discriminants is to use fixed nonlinear basis functions $b_i(\mathbf{x})$ which transform the input vector $\mathbf{x}$ before it is multiplied by the weight vector:

$$y(\mathbf{x}) = \sum_{i=0}^{k} w_i b_i(\mathbf{x}) \qquad (3.9)$$

where the bias is included in the sum by defining $b_0(\mathbf{x})$. For suitable choice of basis functions (3.9) can approximate any continuous function to arbitrary accuracy (these basis functions could be, for example, radial basis functions, fuzzy basis functions, splines, etc.).

If (3.9) is used as input to the threshold activation function g, we get a perceptron. Rosenblatt proved for perceptrons that if the vectors used to train the perceptron are from linearly separable classes, then the perceptron learning algorithm converges [Rosenblatt, 1962].

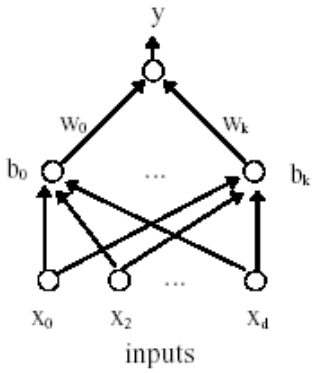


Figure 3.4 The perceptron network

3.2 Multilayer Perceptron

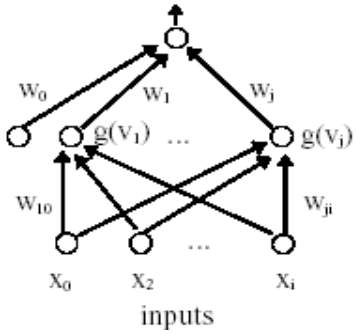


Figure 3.5 The multilayer perceptron network.

The summing junction of the hidden unit is obtained by the following weighted linear combination:

$$v_j = \sum_{i=0}^{d} w_{ij} x_i \tag{3.10}$$

where w_{ij} is a weight in the first layer (from input unit i to hidden unit j) and w_{j0} is the bias for hidden unit j. The activation (output) of hidden unit j is then obtained by

$$o_{hiddenj} = g(v_j) \tag{3.11}$$

For output of whole network the following activation is constructed

$$y = g(\sum_{i=0}^{k} w_{j\,ohiddenj}) \tag{3.12}$$

Two-layered multilayer perceptron in Fig. 3.5 can be represented as a function by combining the previous expressions in the form

$$y = g(\sum_{i=0}^{k} w_j g(\sum_{i=0}^{d} w_{ij} x_i)) \tag{3.13}$$

The activation function for the output unit can be linear. In that case (3.13) becomes a special case of (3.9), in which the basis functions are

$$b_j = g(\sum_{i=0}^{d} w_{ij} x_i) \tag{3.14}$$

If the activation functions in the hidden layer are linear, then such a network can be converted into an equivalent network without hidden units by forming a single linear transformation of two successive linear transformations [Bishop, 1996]. So the networks having non-linear hidden unit activation functions are preferred. Note: In literature the equation for output of neuron can also be seen written as follows

$$y_k = g(\sum_{i=1}^{d} w_{kj} x_j - \theta_k) \tag{3.15}$$

where θ_k is the bias parameter. Mathematically this is equivalent to the former equations where the bias was included in the summation. We can always set $w_{k0} = \theta_k$ and $x_0 = -1$. The sign '–' can, of course, be included in weight parameter and not in the input.

MLPs are mainly used for functional approximation rather than classification problems. They are generally unsuitable for modeling functions with significant local variations. The universal approximation theorem states that MLP can approximate any continuous function arbitrarily well, although it does not provide indication about the complexity of MLP. The Vapnik-Chervonenkis dimension d_{VC} gives a rough approximation about the complexity. According to this principle, the amount of training data should be approximately ten times the d_{VC}, or the number of weights in MLP.

MLPs are suitable for high-dimensional function approximation, if the desired function can be approximated by a low number of ridge functions (MLPs employ ridge functions in the hidden layer). They may perform well, although the training data have redundant inputs.

3.3 Functional Link Network

Functional link neural networks [Pao, 1994] have the same structure as MLPs but polynomial or trigonometric functions are used as activation functions in the hidden layer. Nodes in the input and output layer are linear and only the connections from hidden layer to output layer are weighted. Thus, functional link network can be regarded as a special case of (3.9). A successful use of the functional link networks requires a set of basis functions which can represent the desired function with a sufficient accuracy over the input space. The success of the network depends mainly on the basis functions used. The set of basis functions should not over-parametrize the network.

3.4 Radial Basis Function Network

Interpolation

Radial basis functions (RBF) have originally been used in exact interpolation [Powell, 1987]. In interpolation we have n data points $x_i \in R^d$, and n real valued numbers $t_i \in R$, where

$i = 1,..., n$. The task is to determine a function s in linear space such that $s(x_i) = t_i \quad i = 1,..., n$. The interpolation function is a linear combination of basis functions

$$s(\mathbf{x}) = \sum_{i=1}^{n} w_i b_i(\mathbf{x}) \tag{3.16}$$

As basis functions b_i , radial basis functions of the form (3.17) are used.

$$b_i(\mathbf{x}) = \emptyset(\| \mathbf{x} - \mathbf{x}_i \|) \tag{3.17}$$

where $\emptyset$ is mapping $\mathbf{R}^+ \rightarrow \mathbf{R}$ and the norm is a Euclidean distance. The norm could also be a Mahalanobis distance. The following forms have been considered as radial basis functions [Powell, 1987]:

1. Multiquadric function: $\emptyset(r) = (r^2 + c)^{1/2}$, where c is positive constant (for example, a distance between neighboring knots) and $r \in \mathbf{R}$.
2. $\emptyset(r) = r$, where $r \geq 0$.
3. $\emptyset(r) = r^2$
4. $\emptyset(r) = r^3$
5. $\emptyset(r) = \exp(-r^2)$

It has been reported that the global basis functions may have slightly better interpolation properties than the local ones (Gaussians) [Buhmann & Powell, 1990]. If we next define a matrix $\mathbf{A}$ such that $A_{ij} = \emptyset(\| \mathbf{x}_i - \mathbf{x}_j \|)$, $i, j = 1,..., n,$ the following properties are valid for the basis functions in the list above (provided that the data points are distinct):

for 2: $\mathbf{A}$ is non-singular when $n \geq 2$.

for 3: $s(x)$ is quadratic polynomial in $\mathbf{x}$ and $\mathbf{A}$ is singular if the number of data points exceeds $\frac{1}{2}(d+1)(d+2)$.

for 4: $\mathbf{A}$ can be singular if $n \geq 2$, but is always non-singular if $n = 1$.

for 5: $\mathbf{A}$ is positive definite $\Rightarrow \forall \alpha >)$ the matrix $A(\alpha)_{ij} = \exp(-\alpha \| \mathbf{x}_i - \mathbf{x}_j \|)$ is also positive definite.

Detecting the singularity plays an important role, because the weights are obtained as a solution of system of linear equations. In matrix form the interpolation is simply

$$\mathbf{Aw} = \mathbf{t} \tag{3.18}$$

where **t** consists of *n* target values, **w** is an *n*-dimensional weight vector and **A** is a $n \times n$ -matrix as defined before. Depending on the singularity of **A**, (3.18) is solved either directly or by singular value decomposition methods.

Relation to splines

Also so called thin plate splines have been used as radial basis functions. In contrast to Gaussians, the learning problem is relatively insensitive to the data distribution. They have the property that they minimize the bending energy of infinite thin plate in two dimensional space [Poggio & Girosi, 1990]. This energy minimizing property is also characteristic to natural cubic splines. Next a short description of the relation between radial basis functions and splines is given.

If $\emptyset(r) = r^k$ where $r > 0$ and k is an odd, positive integer and if $m = \frac{1}{2}(k-1)$ and $m > n$ then

$$s(x) = \sum_{i=1}^{n} w_i \left| x - x_i \right|^k + \sum_{i=1}^{\hat{m}} v_i p_i(x) \tag{3.19}$$

becomes a natural spline. In (3.19) $\{p_i \,|\, i = 1,..., \hat{m}\}$ is a basis of algebraic polynomials of degree $\leq m$ and $\sum_{i=1}^{n} w_i p_i(x_i) = 0$, $j = 1,..., \hat{m}$. Then (3.19) can be written

$$s(x) = \sum_{i=1}^{n} w_i \left| x - x_i \right|^k + \sum_{i=1}^{m} v_i x^j \tag{3.20}$$

and the coefficients satisfy

$$\sum_{i=1}^{m} w_i x_i^j \ , \ j = 0,..., m \tag{3.21}$$

If we forget the second term in the right hand side of (3.19) then s becomes a spline of first order. Splines that satisfy the interpolation condition are identical on interpolation interval (they may differ outside this interval), which means that the function can be represented as a B -spline -function:

$$s(x) = \sum_{i=1}^{n} t_i b_i(x) \tag{3.22}$$

where b_i are B -spline basis functions.

RBF Network

Radial basis function network is a network of radial symmetric basis functions described above. Functions whose response increases monotonically away from a central point are also radial basis functions, but because of their globality they are not as commonly used as the local ones. The most used are the Gaussians – mainly because of their good analytical properties. RBF -network produces a mapping

$$y(\mathbf{x}) = \sum_{i=1}^{k} w_i b_i(\mathbf{x}) \tag{3.23}$$

where b_i are radial basis functions. Other common choices for radial basis functions are the Cauchy function, the inverse multiquadric and the non-local multiquadric. When the basis function is Gaussian (3.23) can be seen as approximating a probability density by a mixture of known densities.

RBF -network produces a local mapping. The extrapolation properties of the mapping can be very poor outside the optimization data set. However, the RBF network is an efficient method for low dimensional tasks when input vector dimension is low and very accurate approximations can be obtained. For higher dimensional tasks several difficulties are encountered. Distributing the basis function centers evenly on the input space results in a complex model. The number of hidden layer nodes depends exponentially on the size of input space. Especially, irrelevant inputs are problematic, since they do not add information but increase the number of basis functions. To overcome this problem, a *Gaussian bar network* has been proposed, where the product of univariate Gaussians (tensor product) is replaced by the sum [Hartman & Keeler, 1991].

$$y(\mathbf{x}) = \sum_{i=1}^{k} \sum_{j=1}^{d} w_{ij} \exp\left(-\frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2}\right) \tag{3.24}$$

The same idea of replacing the product by a sum has also been proposed to be used with fuzzy logic systems for control problems. This means that AND -operation is replaced by an OR -operation. However, the linguistic interpretation is then lost.

RBF -network differs from the RBF -interpolation such that

- Number of basis functions is not determined by the size of data.
- Centers of basis functions are not constrained to be input data vectors.
- Each basis function may have its own width.
- Biases and normalization may be included.

Relation to MLP

There is an interesting link between a special type of MLP -network and RBF - network (and FLS). Consider a MLP -network, which is partially connected and has three layers of weights. Input layer is fully connected to the first hidden layer, where the hidden nodes are arranged to the groups of $2d$ nodes, where d is the dimension of the input space. Sigmoidal functions are used as activation functions in both hidden layers. Each of the second hidden layer neuron receives inputs only from the particular preceding layer node group. Thus, the first and the second hidden layers are partially connected to each other. Bishop illustrates the approximation capability of this network by a two dimensional example [Bishop, 1996]:

Output of a neuron, for example of a neuron 1 in Fig. 3.6, is a function of two inputs and the shape of this function is illustrated in Fig. 3.7 (a). Adding two such outputs, for example outputs of neurons 1 and 2, can produce a ridge-shaped function (Fig. 3.7 (b)). Adding two ridges can give a

function of the form as in Fig. 3.7 (c). This is a result of summation in node 9. Transforming this bump function with sigmoid gives a localized "basis function" and the final output *y* is a linear combination of these basis functions (outputs of neurons 9 and 10). Dashed lines remind, that the number of groups in the first hidden layer and number of neurons in the second hidden layer could be different from 2.

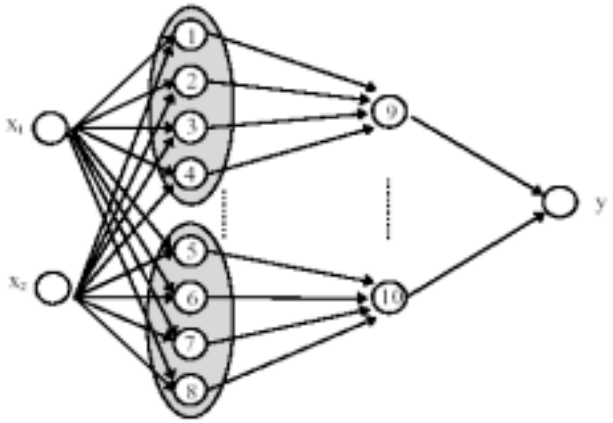


Figure 3.6 Three network layers

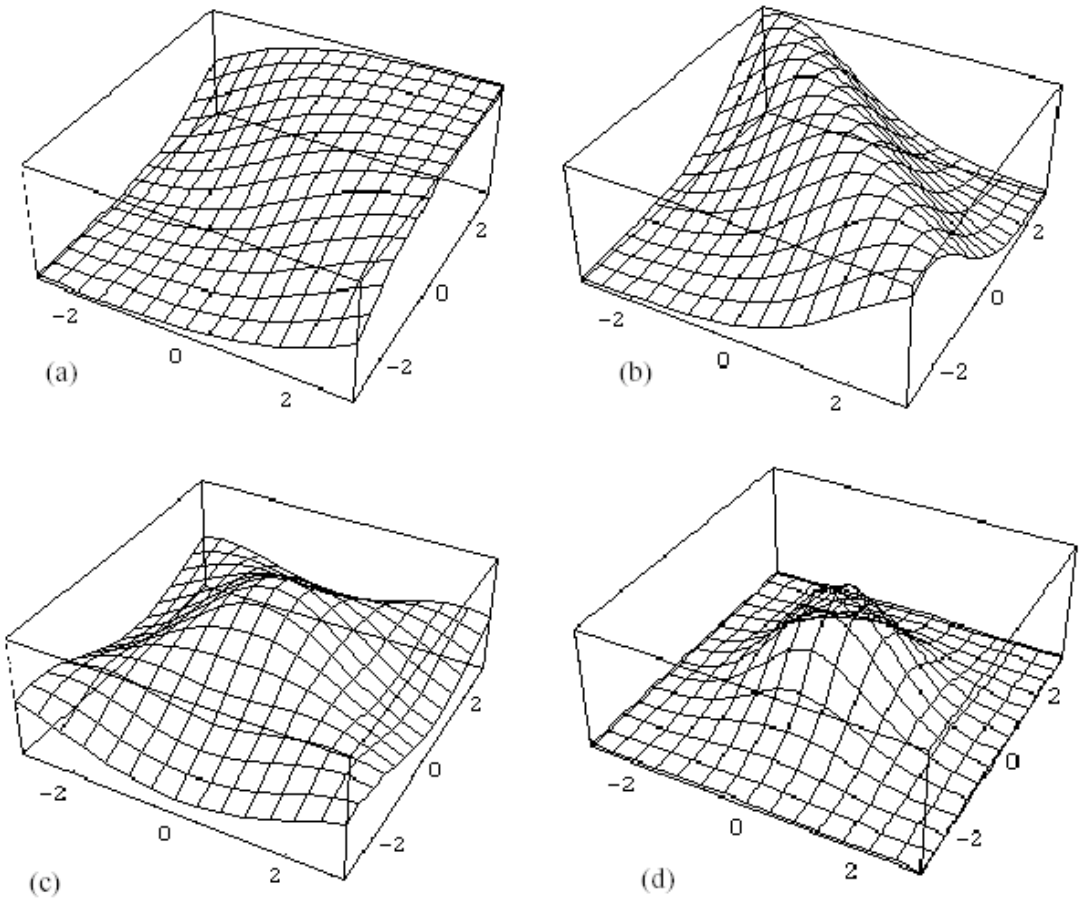


Figure 3.7 Three layer network constructs localized "basis functions", which resemble Gaussian radial basis functions. The final mapping of network is a weighted sum of those basis functions.

3.5 Neurofuzzy Networks

Motivation: Neural networks and fuzzy systems try to emulate the operation of human brain. Neural networks concentrate on the structure of human brain, i.e., on the "hardware" whereas fuzzy logic systems concentrate on "software". This can be viewed hierarchically as if the MLP were at the lowest level trying to simulate the basic functions and fuzzy logic at a higher level trying to simulate fuzzy and symbolic reasoning. [Wang, 1994]

Combining neural networks and fuzzy systems in one unified framework has become popular in the last few years. Although Lee & Lee worked on neurofuzzy as early as in 1975 [Lee & Lee, 1975], more effort was not put on the research until in early 1990's. Berenji & Khedkar, Jang J.-S.R., Takagi & Hayashi, Nauck & Kruse, Wang L.-X., Kosko, Harris & Brown are researchers who have had a strong influence on this field. They all have published their most significant research results (concerning neurofuzzy systems) in the 1990's.

Most of the approaches are difficult to compare because their architectures, learning methods, basis or activation functions differ strongly. Harris and Brown in [Brown & Harris, 1994] separate the neurofuzzy researchers into two schools: those who try to model human brain's reasoning and learning, and those who develop structures and algorithms for system modeling, control, that is, specialized applications. The following treatment of neurofuzzy systems adopts more the latter pproach than the former algorithms for system modeling, control, that is, specialized applications. The following treatment of neurofuzzy systems adopts more the latter approach than the former.

3.5.1 Different Neurofuzzy Approaches

Designation "neurofuzzy" has several different meanings. Sometimes "neurofuzzy" is associated to hybrid systems which act on two distinct subproblems. In that case, neural network is utilized in the first subproblem (e.g., in signal processing) and fuzzy logic is utilized in the second subproblem (e.g., in reasoning task). Normally, when talking about the neurofuzzy systems, the link between these two soft computing methods is understood to be stronger. In the following light is tried to shed on the most common different interpretations.

Fuzzy logic in learning algorithms

A common approach is to use fuzzy logic in neural networks to improve the learning ability. For example, fuzzy control of back-propagation has been proposed by Choi et al. [1992]. Fig. 3.8 explains the basic idea of this method. The purpose is to achieve a faster rate of convergence by controlling the learning rate parameter with fuzzy rules. Rules are of the type:

Rule 1: IF (GoE is NB) AND (CoGoE is NB) THEN CoLP is NS

:

Rule 13: IF (GoE is ZE) AND (CoGoE is ZE) THEN CoLP is PS

:

Rule 25: IF(GoE is PB) AND (CoGoE is PB) THEN CoLP is NS

where CoLP is change of learning parameter, GoE is the gradient of the error surface, CoGoE is change of GoE (approximation to second order gradient), NB, NS, ZE, PS and PB are fuzzy sets "negative big", "negative small", "zero equal", "positive small" and "positive big". (They also incorporated in rules information about the sign change of gradient and information about the momentum constant.)

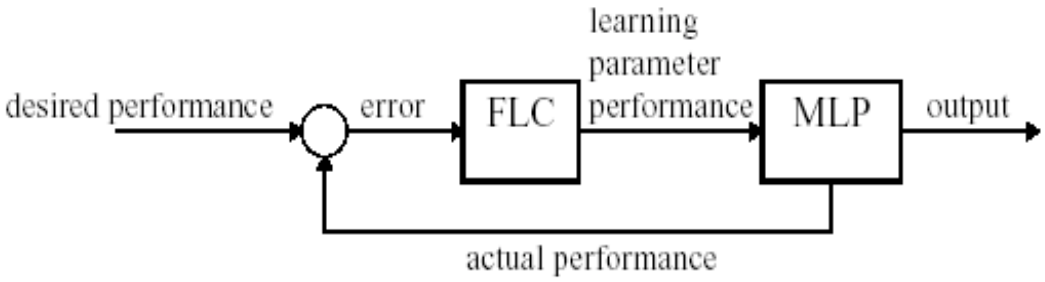


Figure 3.8 Learning rate control by fuzzy logic. FLC is fuzzy logic controller, MLP is multilayer perceptron.

Simulation results, presented by Choi et. al., show that the convergence of fuzzy backpropagation is, at least in their particular example problem (detecting constant signal in the presence of additive noise), faster than standard backpropagation. Another interesting example of using fuzzy logic in optimization is the control of direction of the searching vectors proposed by H. Kawarada & H. Suito (Kawarada, private communication). The objective is to get a minimizer (fuzzified Newton or quasi-Newton method) which has a low computing cost and a large convergence domain. They use an approximation of Hessian which is constructed by the following steps:

1. Interpolate target function f using sample data: Compute one dimensional spline interpolation with respect to each axis direction. Average the spline approximations obtained by the one dimensional interpolations.
2. Calculate Hessian based on spline approximation. If the determinant of Hessian is positive, then use this. Otherwise construct a diagonal Hessian (identity matrix is multiplied by *c*, where *c* is an element of Hessians with maximal absolute value or it is estimated by trial and error).

Fuzzy logic is then possibly used to control search direction

$$\mathbf{d}^{(\tau)} = - \frac{1}{\mathbf{H}^{(\tau)}} \nabla f(\mathbf{x}^{(\tau)}) \qquad (3.25)$$

If the searching vector obtained does not change much when compared to the preceding searching vector, then use it as it is. If the change is large, then use a modified vector which is averaged somewhat by fuzzy logic. Hence, the new searching vector is

$$\mathbf{d}^{(\tau)} = \beta\, \mathbf{d}^{(\tau-1)} + (1-\beta)\mathbf{d}_{cs}^{(\tau)} \qquad (3.26)$$

where β is obtained by fuzzy logic and $\mathbf{d}_{cs}^{(\tau)}$ is searching vector calculated based on Hessian approximation. In test example the convergence domain proved to be superior to that of other methods, but the computational cost was worse than with Newton method and slightly worse than with BFGS. Huntsberger and Ajjimarangsee [1991] have proposed the use of fuzzy logic with Kohonen's self organizing map. The learning rate coefficient is treated as a fuzzy membership value. Other ideas concerning SOMs can be found in [Vuorimaa, 1994], [Tsao et. al., 1994], [Chung & Lee, 1994].

Fuzzy neurons

A standard neuron as McCulloch-Pitts neuron in (3.5) is converted into fuzzy one by replacing multiplication and addition by t-norm and t-conorm. The resulting neuron is often referred to as *hybrid neural network.*

Definition 3.2 (*hybrid neural network*) A hybrid neural network is a network with real valued inputs $x \in [0,1]$, (usually membership degrees) and real valued weights $w_i \in [0,1]$. Input and weight are combined (i.e., product is replaced) using t-norm $T(x_i, w_i)$, t-conorm $S(x_i, w_i)$, or some other continuous operation. These combinations are again combined (i.e., addition is replaced) using t-norm, t-conorm, or some continuous operation. Activation function g can be any continuous function.

If we choose linear activation function, t-norm (min) for addition and t-conorm (max) for product, we get an *AND fuzzy neuron* (3.27), and if they are chosen on the contrary, we get an *OR fuzzy neuron* (3.28). Output of (3.27) corresponds to min-max composition and (3.28) corresponds to max-min composition known from the fuzzy logic.

$$y = T(S(x_1, w_1),...,S(x_d, w_d)) \tag{3.27}$$

$$y = S(T(x_1, w_1),...,T(x_d, w_d)) \tag{3.28}$$

Another way to implement fuzzy neuron is to extend weights and/or inputs and/or outputs (or targets) to fuzzy numbers. Buckley and Hayashi list three choices

1. crisp input, fuzzy weights, fuzzy output (or crisp output by defuzzification).
2. fuzzy input, crisp weights, fuzzy output
3. fuzzy input, fuzzy weights, fuzzy output

which can be used to implement fuzzy IF-THEN rules. In addition there exists a type of network where the weights and targets are crisp and the inputs are fuzzy. The networks of this type are used in classification problems to map fuzzy input vectors to crisp classes.

Definition 3.3 (*regular fuzzy neural network, RFNN*) A regular fuzzy neural network is a network with fuzzy signals and/or fuzzy weights, sigmoidal activation function and all the operations are defined by extension principle.

Example 3.1 Consider a simple network $y = g(w_1 x_1 + w_2 x_2)$, where the inputs and weights are fuzzy numbers. We use the extension principle to calculate $w_i x_i$. Output fuzzy set Y is computed by the Extension principle

$$Y(y) = \begin{cases} (w_1 x_1 + w_2 x_2) g^{-1}(y) & , 0 \le y \le 1 \\ 0 & otherwise \end{cases}$$

where $g^{-1}(y) = \ln y - \ln(1-y)$ is simply the inverse function of logistic sigmoidal $g(z) = 1/(1+e^{-z})$. [Fuller, 2001]

The problem of regular fuzzy neural networks is that they are monotonic, which means that the fuzzy neural nets based on the extension principle are universal approximators only for monotonic functions. Buckley and Hayashi [1994] were the first to prove this property.

Theorem 3.1 g is a increasing function of its arguments, i.e. if $x_1 \subset x_1^{'}$ and $x_2 \subset x_2^{'}$ ($x_i, x_i^{'}, w_i$ are fuzzy numbers) then

$$g(w_1 x_1 + w_2 x_2) \subset g(w_1 x_1^{'} + w_2 x_2^{'})$$

Proof. Because min and max are increasing functions, then

$$w_1 x_1 + w_2 x_2 \subset w_1 x_1^{'} + w_2 x_2^{'}$$

which means that regular fuzzy neural network is not a universal approximator. This is a serious drawback for the networks of this type.

Definition 3.4 (*hybrid fuzzy neural network, HFNN*) A hybrid fuzzy neural network is a network with fuzzy valued inputs $x_i \in [0,1]$ and/or fuzzy valued weights $w_i \in [0,1]$.

Input and weight are combined using t-norm $T(x_i, w_i)$, t-conorm $S(x_i, w_i)$, or some other continuous operation. These combinations are again combined using t-norm, t-conorm, or some continuous operation. Activation function *g* can be any continuous function.

The universal approximation property for hybrid fuzzy neural networks has been proven by Buckley and Hayashi [1994]. Thus the use of HFNN is more justified in practical applications than the use of RFNN.

Many researchers working with fuzzy neurons follow the basic principles described above, but there is no standard path to follow. This is evidenced by the various different approaches proposed [Kartalopoulos, 1996].

Cooperative approach: Neural networks as pre-processors or post-processors of data

One of the biggest problems with fuzzy systems is the curse of dimensionality. When the dimension of the problem increases the size of the fuzzy model (and the size of training set needed) grows exponentially. The use of more than 4 inputs may be impractical. The number of combinations of input terms (possibly also the number of rules) is

$$\prod_{i=1}^{d} m_i \tag{3.29}$$

where m_i is the number of fuzzy sets on axis i. For example, if we have five inputs and each input space is partitioned into seven fuzzy sets, the number of combinations is 16,807. Therefore, there is a strong need for data reduction. The smallest number of input variables should be used to explain a maximal amount of information [Bossley et. al., 1995].

The most common method to decrease the dimension of input space is the principal component analysis (PCA). The main goal of identifying principal components is to preserve as much relevant information as possible. Selecting M attributes from d is equivalent to selecting M basis vectors which span the new subspace, and projecting the data onto this M -dimensional subspace. Therefore, identifying principal components allows us to reduce the dimensionality of a data in which there are large number of correlated variables and at the same time retaining as much as possible of the variation present in the data. This reduction is achieved via a set of linear transformations which transform input variables to a new set of variables (uncorrelated principal components). The algorithm goes as follows [Press et. al., 1992; Bishop, 1996]:

1. compute the mean of inputs in data and subtract it off
2. calculate covariance matrix and its eigenvectors and eigen values
3. retain eigenvectors corresponding to the M largest eigen values
4. project input vectors onto the eigenvectors

Neural networks may be used to perform the dimensionality reduction. A two-layer perceptron with linear output units (number of hidden units is M , with $M < d$) which is trained to map input vectors onto themselves by minimization of sum-of-squares error is able to perform a linear principal component analysis. If two additional nonlinear hidden layers are allowed to put into the network, the network can be made to perform a non-linear principal component analysis.

Neural networks as tuners of fuzzy logic systems

The similarities between neural networks and fuzzy logic systems were noticed, which led to the development of neurofuzzy systems. The original purpose of neurofuzzy systems was to incorporate learning (and classification) capability to fuzzy systems or alternatively to achieve similar transparency (intelligibility) in neural networks as in fuzzy systems. Learning is assumed to reduce design costs, increase flexibility, improve performance and decrease human intervention. If prior knowledge is unavailable and/or the plant is time-varying then the most sensible (possibly the only) solution is to utilize learning capabilities.

In the 1980s, a computationally efficient training algorithm for multi-layer neural networks was discovered. It was named *error back-propagation*. The principle of the method is that it finds the derivatives of an error function with respect to the weights in the network. The error function can then be minimized by using gradient based optimization algorithms. Since back-propagation can be applied to any feedforward network, some researchers (L.-X. Wang among the first) began to represent fuzzy logic systems as feedforward networks. The idea was to use the training algorithm to adjust weights, centers and widths of membership functions.

In Fig. 3.9 fuzzy logic system (2.46), i.e., d -input 1 -output approximator, is represented as a network. From now on, the designation "neurofuzzy" is associated to the networks of this type (not to the fuzzy neurons).

The most common way to represent neurofuzzy architecture is shown in Fig 3.10. Although it looks different from the network in Fig. 3.9, it is basically the same network. Only the way to illustrate network differs.

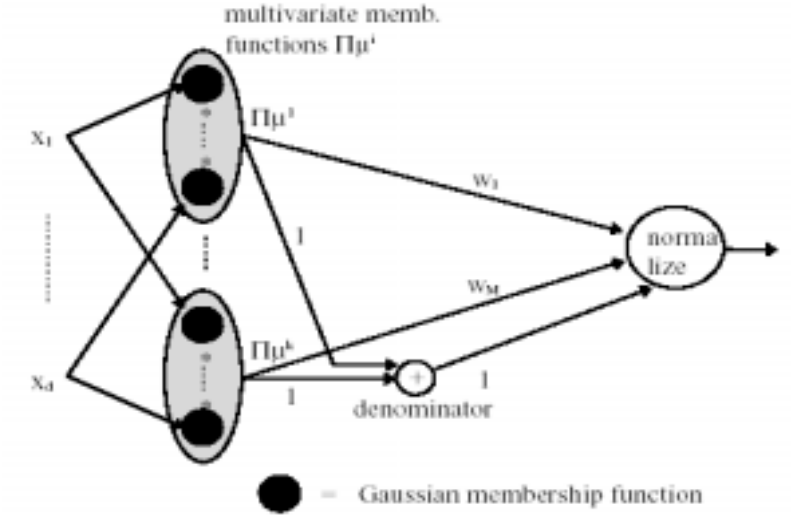


Figure 3.9 Neurofuzzy network for back-propagation

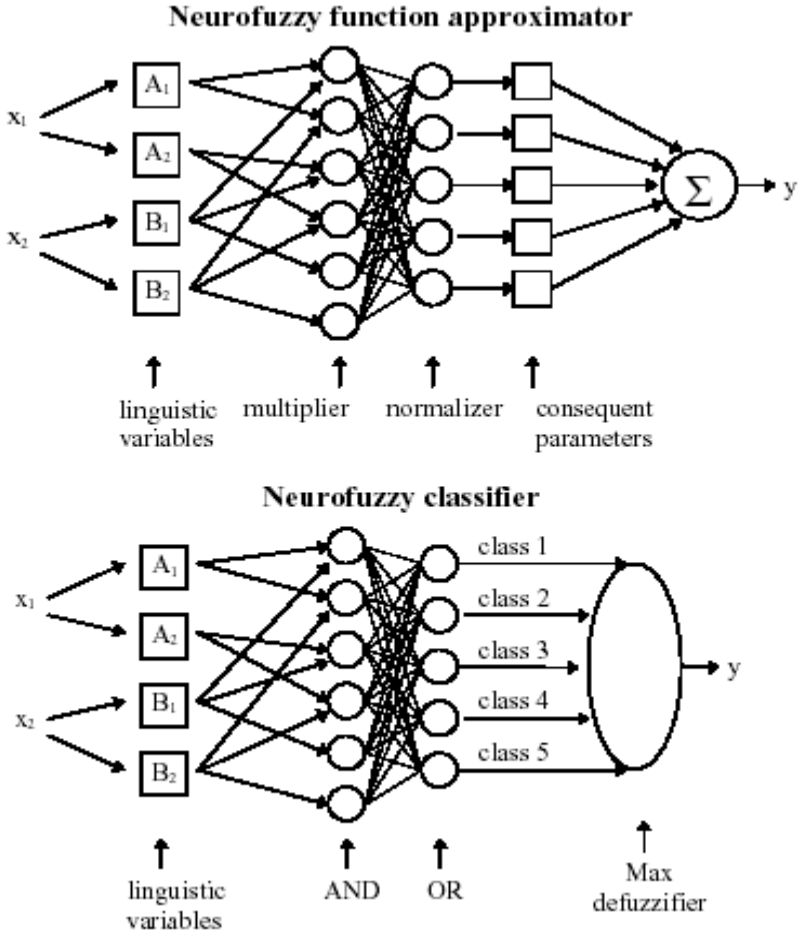


Figure 3.10 J.-H. R. Jang's neurofuzzy network for function approximation and classification problems.

Advantages and drawbacks of neurofuzzy systems

+ weights are the centers of THEN part fuzzy sets (clear meaning)
+ other parameters (width and position of membership functions) have clear meaning
+ initial weights can be chosen appropriately (linguistic rules)
− curse of dimensionality

Committee of networks

The method of combining networks to form a *committee* has been used to improve the generalization ability of the networks. The performance of committee can be better than the performance of isolated networks. It can consist of networks with different architectures, e.g., different types of neural networks, fuzzy logic systems and conventional models.

The committee prediction (output) is calculated as an average of the outputs of the q networks:

$$y_{COM}(\mathbf{x}) = \frac{1}{q} i=1 \sum^{q} y_i(\mathbf{x}) \tag{3.30}$$

The reduction of error arises from the reduced variance due the averaging. For more detailed description see [Bishop, 1996].

Kosko has proposed in [Kosko, 1997] the use of weighted average to combine different fuzzy systems that try to predict the same input-output relation. He does not restrict the form of fuzzy system to be additive or SAM system. The only difference with (3.30) is that Kosko weights fuzzy system outputs $y_i(\mathbf{x})$ by credibilities $w_i \in [0,1]$, such that at least one system has nonzero credibility. More information about combining fuzzy logic and neural networks can be found in the IEEE Communications Magazine special issue "Fuzzy and Neural Networks", September 1992 and [Carpenter et. al., 1991; Bulsari, 1992; Cox, 1992; Fullér, 2001; Brown & Harris, 1994; Kartalopoulos, 1996].

3.6. ANFIS (Adaptive Neuro-Fuzzy Inference System)

A fuzzy system can be considered to be a parameterized nonlinear map, called f . This point will be made clearer later on in the unified form of soft computing methods, but let's write here explicitly the expression of f .

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{m} y^l \left(\prod_{i=1}^{n} \mu_{A_i^l}(x_i)\right)}{\sum_{i=1}^{m} \left(\prod_{i=1}^{n} \mu_{A_i^l}(x_i)\right)}$$

where y^l is a place of output singleton if Mamdani reasoning is applied or a constant if Sugeno reasoning is applied. The membership function $\mu_{A_i^l}(x_i)$ corresponds to the input $\mathbf{x} = [x_1,...,x_n]$ of the rule l . The *and* connective in the premise is carried out by a product and defuzzification by the center-of-gravity method. This can be further written as

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i b_i(\mathbf{x})$$

where $w_i = y^i$ and

$$b_j(\mathbf{x}) = \frac{\prod_{i=1}^{n} \mu_{A_i^l}(x_i)}{\sum_{i=1}^{m} \left(\prod_{i=1}^{n} \mu_{A_i^l}(x_i) \right)}$$

If F is a continuous, nonlinear map on a compact set, then f can approximate F to any desired accuracy, i.e.

$$F \approx f_{FS}$$

The reader can skip this part without loss of continuity, if the mathematical machinery of Hilbert spaces is not familiar.

Well-known theorems from Approximation theory for polynomials, can be extended to fuzzy systems (e.g. Wang: A course in fuzzy systems and control, Prentice Hall, 1997).

The following theorems are found from R.F. Curtain and A.J. Pritchard: Functional Analysis in Modern Applied Mathematics as Corollaries of Orthogonal Projection Theorem.

Theorem 3.4. Let F be a bounded function on $[a,b]$ and $E = \{x_1,...,x_k\}$ a set of points in $[a,b]$. Then there exists the least squares polynomial of degree $\leq n$, p_n^k which minimizes

$$\sum_{i=1}^{k} \left|F(x_i) - p(x_i)\right|^2$$

over all polynomials of degree $\leq n$.

Theorem 3.5. If $F \in C[a,b]$, then for any $n \geq 0$, there exists a best approximating polynomial π_n of degree $\leq n$ such that

$$\left\|F - u_n\right\|_\infty \leq \left\|F - p\right\|_\infty$$

over all polynomials p of degree $\leq n$.

We can also consider the simpler problem of approximating at finitely many points.

Theorem 3.6. If F is a bounded function on $[a,b]$ and $E = \{x_1,...,x_k\}$ a set of points in $[a,b]$, then there exists a best approximating polynomial π_n^k of degree $\leq n$, p_n^k which minimizes

$$\max_{0 \leq i \leq k} \left|F(x_i) - p(x_i)\right|$$

over all polynomials of degree $\leq n$.

The message of the Theorems can also be summarized by saying that polynomials are dense in the space of continuous functions. The same can also be said of trigonometric functions.

ANFIS structure

Consider a Sugeno type of fuzzy system having the rule base

1. If x is A_1 and y is B_1, then $f_1 = p_1 x + q_1 y + r_1$
2. If x is A_2 and y is B_2, then $f_2 = p_2 + q_2 y + r_2$

Let the membership functions of fuzzy sets $A_i, B_i, i = 1,2,$ be μ_{A_i}, μ_{B_i} .

In evaluating the rules, choose *product* for T-norm (logical *and*).

1. Evaluating the rule premises results in

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

2. Evaluating the implication and the rule consequences gives

$$f(x,y) = \frac{w_1(x,y) f_1(x,y) + w_2(x,y) f_2(x,y)}{w_1(x,y) + w_2(x,y)}$$

Or leaving the arguments out

$$f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2}$$

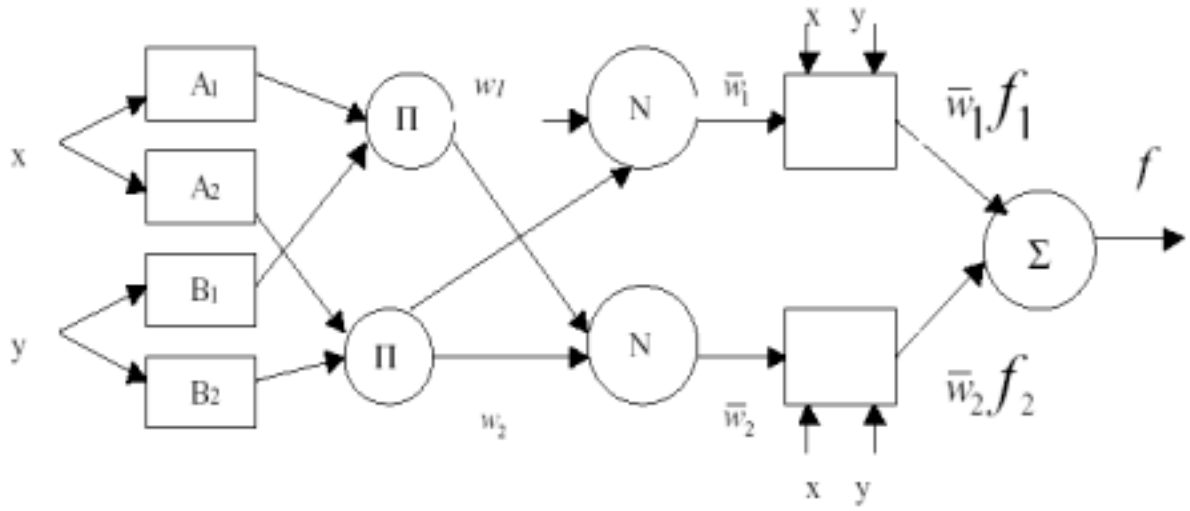
This can be separated to phases by first defining

$$\overline{w_i} = \frac{w_i}{w_1 + w_2}$$

Then f can be written as

$$f = \overline{w_1} f_1 + \overline{w_2} f_2$$

All computations can be presented in a diagram form.



3.7 Approximation capability

RBF –networks

The universal approximation property for RBF -network with Gaussian basis functions was proved by Hartman *et al*. in 1990. A generalization of this result was obtained by Park and Sandberg in 1991.

Theorem 3.1 Let $g : \mathbf{R}^d \rightarrow \mathbf{R}$ be a radial symmetric, integrable, bounded function such that b is continuous almost everywhere and $\int_{R^d} g(\mathbf{x})d(\mathbf{x}) \neq 0$, then the weighted sum

$$\sum_{i=1}^{k} w_i \emptyset \left(\frac{\| \mathbf{x} - \mathbf{x}_i \|}{\sigma} \right) \tag{3.31}$$

is dense in $L^P(\mathbf{R}^n)$, where $\emptyset(\| \mathbf{x} \|) = g(\mathbf{x})$.

Proof can be found in [Park & Sandberg, 1991]. Later they generalized this result such that the only conditions for $g : \mathbf{R}^d \rightarrow \mathbf{R}$ are: it is square-integrable and pointable.

Chen and Chen gave in [Chen & Chen, 1995] an answer to the question: What is the necessary and sufficient condition for a function to be qualified as an activation function in RBF -networks? They proved the following theorem:

Theorem 3.2 Let $g \in C(\mathbf{R}) \cap S'(\mathbf{R})$ i.e., g belongs to a set of all continuous functions which are also linear continuous functionals defined on the space of infinitely differetiable functions, which rapidly decrease at infinity. Then the linear combination

$$\sum_{i=1}^{k} w_i g(\lambda_i \| \mathbf{x}\text{-}\mathbf{y}_i \|_{R^d}) \tag{3.32}$$

is dense in $C(K)$, if g is not an even polynomial, where K is a compact set in $\mathbf{R}^d$, $y_i \in \mathbf{R}^d$, $w_i, \lambda_i \in \mathbf{R}$.

Two-layer feedforward networks

Two-layer neural network with sigmoidal activation functions in hidden layer can approximate any continuous functions defined on a compact set in $\mathbf{R}^d$. More formally:

Theorem 3.3 Let g be a bounded, increasing real function, K be a compact set in $\mathbf{R}^d$, and $f : K \to \mathbf{R}$ be a continuous function. Then for every $\varepsilon > 0$ there exists $k \in \mathbf{N}$ and $w_i, w_{ij}, \theta_i \in \mathbf{R}$ such that

$$\max_{x \in K} | f(\mathbf{x}) - y(\mathbf{x}) | < \varepsilon \tag{3.33}$$

where

$$y(\mathbf{x}) = \sum_{i=1}^{k} w_i g(\sum_{j=1}^{d} w_{ij} x_j - \theta_i) \tag{3.34}$$

Different proofs have been formulated (in these proofs sigmoidal functions are assumed to be continuous or monotone):

- Carrol and Dickinson [1989]: used the inverse Radon transformation.
- Cybenko [1989]: used functional analysis, Hahn-Banach theorem and Riesz representation theorem.
- Funahashi [1989]: used a kernel which can be expressed as a difference of two sigmoid functions.

The same Stone-Weierstrass theorem as with Mendel-Wang's fuzzy logic system earlier can be used to show universal approximation capability for certain networks. The requirement is that the activation function transforms multiplication into addition. Exponential functions, step functions and partial fractions accomplish this transformation. Fullér lists in [Fullér, 2001] networks that satisfy the Stone- Weierstrass theorem*: Decaying-exponential networks*, *Fourier networks*, *Exponentiated-function networks, Modified sigma-pi* and *polynomial networks*, *Step functions* and *Perceptron networks* and *Partial fraction networks*.

Chen & Chen completed the Cybenko's theorem by proving that instead of continuity the boundedness of the sigmoidal functions in hidden layer is sufficient to ensure the approximation ability [Chen & Chen, 1990; Cybenko, 1989].

Chapter 4

Unified Form of Soft Computing Methods

4.1 Introduction

The goal of this presentation is to study different soft computing methods for function approximation. It is also demonstrated here that when fuzzy system utilizes sum and product operators rather than max-min operators with the center of gravity defuzzification, there exists a close relationship between fuzzy systems and neural networks. Sometimes they are equivalent and thus they can be trained using the same training algorithms. Relationship can also be found between fuzzy systems, some conventional interpolation methods and probabilistic regression methods.

The basic functional mapping is of the form

$$y = F(\mathbf{x}) \tag{4.1}$$

where $F : \mathbf{R}^d \rightarrow \mathbf{R}$ is a mapping between an input vector $\mathbf{x}$ and an output y. The function F F (also referred to as target function) is approximated using parametrized non-linear mapping

$$\hat{y} = f(\mathbf{x}, \theta) \tag{4.2}$$

where $f : \mathbf{R}^d \rightarrow \mathbf{R}$ is some non-linear function parametrized by $\theta \in \mathbf{R}^{d_0}$. A neural network or fuzzy model can be regarded as a particular choice for the function f. More conventional choices for f are, for example, Fourier series, Wavelets, interpolation methods (including polynomials, Lagrange interpolation, splines, finite element methods, etc.) and probabilistic regressions. For example, the M th order polynomial

$$f(x, \theta) = w_0 + w_1 x + \ldots + w_M x^M = \sum_{i=0}^{M} w_i x^i \tag{4.3}$$

can be regarded as a non-linear mapping (one input, one output variable), where the parameter vector θ contains weights. There is analogy to the weights in neural network. Setting the number of terms large enough, (4.3) can approximate any continuous function to arbitrary accuracy. In higher dimensional cases input x can be replaced by a vector $\mathbf{x}$, but the curse of dimensionality limits the use of polynomials in practical applications.

The overall structure of the mapping under consideration can be presented as a weighted sum as follows

$$f(\mathbf{x},\theta)=\sum_{i=1}^{k} w_i(\theta) b_i(\mathbf{x},\theta) \tag{4.4}$$

The functions b_i are referred to as *basis functions*, since the role they play in (4.3) is similar to that of a functional space basis [Sjöberg et. al., 1995]. The weights w_i and the parameters of the basis functions are defined by the vector θ , which is determined with the help of data. Mapping (4.4) has basically the same form as the generalized linear discriminant in (3.9). The parameterization can also be biased

$$f(\mathbf{x},\theta)=\sum_{i=1}^{k} w_i(\theta) b_i(\mathbf{x},\theta)+w_0 \tag{4.5}$$

or normalized

$$f(\mathbf{x},\theta)=\frac{\sum_{i=1}^{k} w_i(\theta) b_i(\mathbf{x},\theta)}{\sum_{i=1}^{k} b_i(\mathbf{x},\theta)} \tag{4.6}$$

Basis function

Single-variable basis functions can be classified to different classes depending on their nature. The classification can be performed with regard to locality or globality of the function [Sjöberg et. al., 1995]:

- *Local basis functions* are functions having their gradient with bounded support, or at least vanishing rapidly at infinity.
- *Global basis functions* are functions having infinitely spreading gradient. The locality/globality of basis functions can be used to classify function approximation methods (regression methods) to the following categories:

Global methods:

- linear regression
- polynomial regression
- projection pursuit
- polynomial networks
- MLP

Local methods:

- kernel smoothers
- adaptive kernel methods
- splines
- piecewise regression

- RBF
- regularization networks
- FLS
- MARS
- partitioning methods

Support of the basis function is a set

$$S = \{ \mathbf{x} \in \mathbf{R}^d \mid b(\mathbf{x}) \neq 0 \} \tag{4.7}$$

The basis functions can be classified according to the property if its support is compact or not. Locality does not ensure that the function has a *compact support*.

Fuzzy logic example: Gaussian membership functions do not have compact support, but by using α-cut this property can be incorporated into them. Another way is to select *Gaussian-type* membership functions which have a compact support and are infinitely differentiable [Werntges, 1993].

$$\mu_A(x) = \begin{cases} \exp\left(-\frac{(b-a)^2}{(b-x)(x-a)}\right) / \exp(-1) & \mathbf{x} \in]a,b[\\ 0 & \text{, otherwise} \end{cases}$$

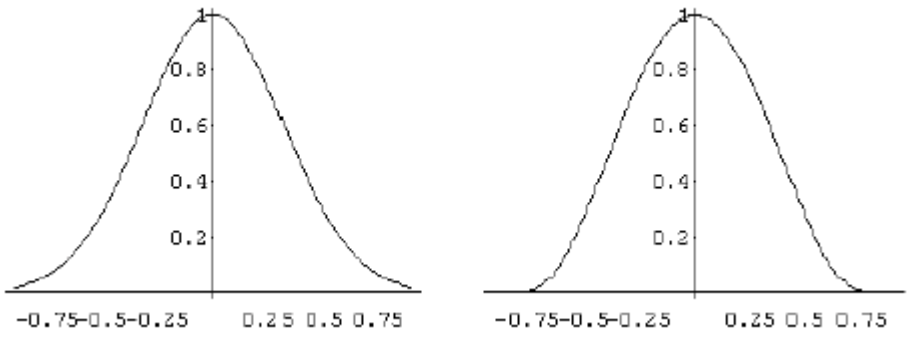


Figure 4.1 Gaussian and Gaussian type membership function.

The basis functions are *normal* if

$$\sum_{i=1}^{k} b_i(\mathbf{x}) = 1, \forall \mathbf{x} \in \mathbf{R}^d \tag{4.8}$$

and *orthogonal* if

$$\begin{cases} b_i(\mathbf{x}) b_j(\mathbf{x}) = 0 & i \neq j \\ b_i(\mathbf{x})\ b_i(\mathbf{x}) \neq 0 \end{cases} \tag{4.9}$$

Multi-variable basis function

The basis function is usually obtained by parameterizing a single mother basis function that can be generically denoted by $M(x)$ [Sjöberg et. al., 1995]. In the multi-variable case $(d>1)$, b_i are multi-variable functions. Usually they are constructed from the single-variable function M in some simple manner.

Tensor product construction is given by a product

$$b(\mathbf{x}) = M(x_1,\theta_1)M(x_2,\theta_2)...M(x_d,\theta_d) = \prod_{i=1}^{d} M(x_i,\theta_i) \tag{4.10}$$

where the single-variable basis functions can be identical or not (depends on the parameters θ_i). This construction principle can be found, for example, in fuzzy logic, finite element methods, multivariate splines.

Radial function construction has the form

$$b(\mathbf{x}) = M(\| \mathbf{x}\text{-}\mathbf{c} \|_D) \tag{4.11}$$

where $\|\mathbf{x}\|_D$ denotes any chosen norm, but usually quadratic norm $\|\mathbf{x}\text{-}\mathbf{c}\|_d^2 = (\mathbf{x}\text{-}\mathbf{c})^T \mathbf{D}(\mathbf{x}\text{-}\mathbf{c})$ is used, where $\mathbf{D}$ is a matrix of dilation parameters. In addition to radial basis functions, sometimes applied to membership function construction in fuzzy logic.

Ridge function construction is given by

$$b(\mathbf{x}) = M(\mathbf{d}^T\mathbf{x} + c), \quad \text{where } \mathbf{d} \in \mathbf{R}^d,\ c \in \mathbf{R} \tag{4.12}$$

See Fig. 3.5 (b) for a typical ridge function constructed by a MLP. In above, $\mathbf{x}$ is an input vector, $\mathbf{c}$ and c are location parameters, and $\mathbf{D}$ and $\mathbf{d}$ are scale parameters. A unified form for basis functions could then be written as

$$b_i(\mathbf{x}) = M(\mathbf{x},\mathbf{d}_i,\mathbf{c}_i) \tag{4.13}$$

which suggests that the basis functions could be characterized and indexed only by the parameter vectors, i.e.,

$$b_i(\mathbf{x}) = b(\mathbf{x},\theta_i) \tag{4.14}$$

Each approximation method previewed earlier and in this chapter establishes a set of predefined basis function. As the vector of parameters range over all of their allowed values, the entire set of basis functions

$$\{b(\mathbf{x}, \theta_i) | i = 1, ..., q\} \tag{4.15}$$

is created. The goal is to find a subset of (4.15)

$$\{b(\mathbf{x}, \theta_i) | i = 1, ..., d_0\} \subseteq \{b(\mathbf{x}, \theta_i) | i = 1, ..., q\} \tag{4.16}$$

such that the linear combination of basis functions best approximates the target function F. As the values of weights and basis function parameters range over all of their allowed values, all functions in the subspace are generated and the method is said to be a universal approximator for that function class (usually the class of all continuous functions). The approximating subspace can be defined as a set of all functions for which the bias is zero. The goal is to choose a set of basis functions such that the bias is kept low and the variance is also kept small.

The scale (or width) parameter of basis functions can be viewed as a smoothing parameter; basically the larger the width the smoother the output (which means larger bias, smaller variance). The number of entries *k*, which is either fixed beforehand or is taken to be a model selection parameter, has a similar effect because as the number of basis functions increases, their width decreases (provided that the linear independence of basis functions is satisfied) but variance increases. Regularization is an efficient method to control the bias-variance tradeoff and will be discussed later.

4.2 Interpolation Networks

Certain classes of neural networks and fuzzy systems share interpolation as a common feature. The strict interpolation can be stated:

Given a set of N vectors $\{\mathbf{x}^n | n = 1, ..., N\}$ and corresponding set of real numbers $\{t^n | n = 1, ..., N\}$, find a function *y* that satisfies $y(\mathbf{x}^n) = t^n$ for each n. The problem of knot placement is discussed under different names in various methods: center locations in RBF and FLS methods, partitioning strategy in recursive partitioning methods, etc. Typically knot positions in the input space are chosen as a subset of the training data set, or they are distributed uniformly. The corresponding y - values can be found from the training data, for example, by least squares techniques. Interpolation is eventually used to calculate the outputs corresponding to the input values between the knots.

Interpolation by fuzzy systems

x is A'
If x is A, then y is B
y is B'

can be viewed as a generalized interpolation as shown next. The value of y is calculated based on the relationship between x and y. We do not know the complete relationship; it is known only for

some particular values of x and it is represented through the use of linguistic rules. The data points $(\mathbf{x}^n, t^n)$ are replaced by "boxes" (A^n, B^n) as shown in Fig. 4.2 (1-dimensional input space).

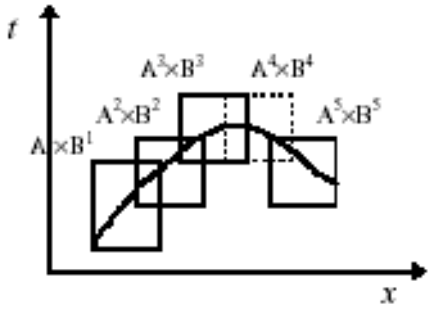


Figure 4.2 Fuzzy interpolation.

Let the implication be a product implication

$$\mu_R(\mathbf{x}, y) = \mu_A(\mathbf{x})\,\mu_R(y)$$

where $R = (\sum_i A_i \times B_i)$

Let μ_{O^l} be the final output membership function as a result of rule l :

$$\mu_{O^l}(y) = \sup_{x' \in X}[\mu_R(\mathbf{x'}, y) * \mu_{A'}(\mathbf{x'})]$$

$$= \sup_{x' \in X}[\prod_{i=1}^{d} \mu_{A_i^l}(x'_i)\mu_{R^l}(y^l)\mu_{A''}(\mathbf{x'}) \qquad (4.17)$$

Using singleton fuzzifier $\mu_{A''}(\mathbf{x'}) = \begin{cases} 1 & , \mathbf{x'}=\mathbf{x} \\ 0 & , \text{otherwise} \end{cases}$ and assuming that $\mu_{B^l}(y^l) = 1$

(4.17) becomes

$$\mu_{O^l}(y^l) = \prod_{i=1}^{d} \mu A_i^l(x_i) \qquad (4.18)$$

Substituting (4.18) into center of gravity defuzzifier we get a fuzzy system (2.47). Fuzzy interpolation can be regarded to a some extent as a generalized interpolation method:

Example 4.1 Let the membership functions be constructed by a *LR*-representation (2.13) as follows

$$\mu_A(x) = \begin{cases} L(\dfrac{m-x}{\alpha}) & , \ x < m \\ R(\dfrac{x-m}{\beta}) & , \ x \geq m \end{cases} = \begin{cases} \max(0, 1-\dfrac{m-x}{\alpha}) & , \ x < m \\ \max(0, 1-\dfrac{x-m}{\beta}) & , \ x \geq m \end{cases} \tag{4.19}$$

where $L(x)$ and $R(x)$ are replaced by the function $\max(0, 1-x)$. Then

$$y^{i-1}\mu_{O^{i-1}}(x) + y^i \mu_{O^i}(x) = y^{i-1} \max(0, \frac{\alpha - x + m_{i-1}}{\beta}) + y^i \max(0, \frac{\alpha - m_i + x}{\alpha})$$

$$= y^{i-1}(\frac{m_i - x}{m_i - m_{i-1}}) + y^i(\frac{x - m_{i-1}}{m_i - m_{i-1}})$$

$$= y^{i-1}(\frac{x - m_i}{m_{i-1} - m_i}) + y^i(\frac{x - m_{i-1}}{m_i - m_{i-1}})$$

is a Lagrangian interpolation polynomial of 1st degree (or a B-spline of order 1). Otherwise, if Mean of Maximum defuzzification method is used then we get an interpolation which resembles B^0-spline interpolation. In Fuzzy interpolation the noncontinuity point lies in the intersection of the basis functions, not on the knots as with B^0-spline interpolation (Fig 4.3).

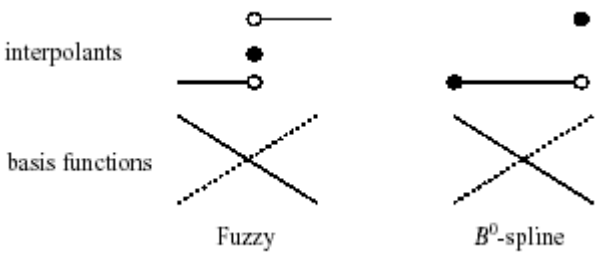


Figure 4.3 Fuzzy interpolation and B^0-spline interpolation.

Interpolation of noisy data

Here we consider a mapping from single input variable to single output variable as presented in [Webb, 1994; Bishop, 1996]. Target data are generated from a noise-free function f but input data have noise ε . The error function can be written

$$E = \frac{1}{2}\int\int \{y(x+\varepsilon) - f(x)\}^2 \tilde{p}(\varepsilon)\, p(x) d\varepsilon dx \tag{4.20}$$

and it is reformulated by changing variables using $z = x + \varepsilon$

$$E = \frac{1}{2}\int\int \{y(z) - f(x)\}^2 \tilde{p}(z-x)\, p(x) dz dx \tag{4.21}$$

By setting the derivative of E with respect to function y to zero, we get

$$y(z) = \frac{\int f(x)\tilde{p}(z-x)\, p(x) dx}{\int \tilde{p}(z-x)\, p(x) dx} \tag{4.22}$$

which can be approximated in the case of finite data points by

$$y(x) = \frac{\sum_n f(x^a)\tilde{p}(x-x^n)}{\sum_n \tilde{p}(x-x^n)} = \frac{\sum_n t^n \tilde{p}(x-x^n)}{\sum_n \tilde{p}(x-x^n)} \tag{4.23}$$

which is a weighted sum of normalized basis functions. The form of basis functions depends on the distribution of noise. Gaussian basis functions are chosen if the distribution is normal.

4.3 Models for Non-linear Mapping

Here some popular models for performing non-linear mapping are described. They all have the basic structure of (4.4) and can be represented as a network and some of them have close relationships to soft computing methods. The modeling capabilities of (4.4) depend on the properties of the basis functions (their shape, size and distribution). The basis functions can be bounded ridge functions (MLP), wavelets, trigonometric and polynomial functions (Functional Link Networks), Gaussian and Gaussian bar functions (RBF, Fuzzy) and piecewise polynomial functions (splines, Fuzzy, etc.).

Polynomials and Splines

Polynomials are global, i.e., they have basis functions that spread their influence everywhere. Polynomials can be used locally by means of regression splines, which represent the fit as a piecewise polynomial. For any given set of knots the mapping is formed by multiple regression on

the appropriate set of basis vectors. These vectors are the basis functions (piecewise polynomials) evaluated at $y^n, n = 1,..., N$.

A variant of the polynomial splines are the *natural splines*. They are constrained to produce mapping, which has linear parts beyond the boundary knots. This can reduce the variance at the boundaries.

The simplest form of (4.4) is the B^0 -spline. The output is w_i on small area A_i and the whole output is a piecewise constant and (piecewise continuous) function (Fig. 4.3).
The B-splines are local basis functions which are computationally simple and their smoothness can be easily varied. They are mainly used in graphical applications, but they are also appropriate for function approximation. There is an invertible relationship between fuzzy and B -spline networks and, in fact, some researchers (for example, Harris & Brown) have used B -splines as basis functions in their neurofuzzy systems.

The *cubic smoothing spline* derives from

$$\sum_{n=1} \left\{ t^n - y(x^n) \right\}^2 + \lambda \int_a^b \left\{ \frac{d^2 y}{dx^2} \right\} dx \tag{4.24}$$

where $a \le x_1 \le ... \le x_n \le b$. λ is a constant. The penalized residual sum of squares has a unique minimizer which is a natural cubic spline with knots at the unique values of x_n . This minimizing spline is called cubic smoothing spline.

The generalization of the cubic smoothing spline to two or higher dimensions is difficult. However, it can be done by a thin-plate spline which derives from the generalization of the second derivative penalty to a two-dimensional Laplacian penalty. Usually splines are generalized by a tensor product.

Multivariate adaptive regression splines (MARS)

Friedman [1991] generalized the adaptive regression spline procedure to include multivariate tensor-spline bases. The mapping function of MARS can be written

$$y = \sum_{i=1}^{k} w_i \prod_{j=1}^{d} b_{ij}(x_{v(ij)}) + w_0 \tag{4.25}$$

where the ith basis function is formed by a product of one-dimensional spline functions b_{ij} . In x_v the v determines which input is used in each case. More generally, MARS can be explained as follows: The output of a MARS model is a combination of multivariate basis functions, i.e., of the form (4.4). The multivariate basis functions in turn are produced by multiplying univariate truncated polynomial basis. A truncated polynomial is a function which is zero when its argument is negative and is polynomial mapping when it is positive.

Wavelets

Wavelets are functions that are used in representing other functions or data. They cut up data into frequency components and analyze data in components according to scale. This means that data is processed at different resolutions (basis functions vary in scale). In wavelet construction the so called *mother wavelet* $b(x)$ is dilated and translated to form an orthogonal basis, which consists of compactly supported local basis functions (wavelets):

$$b_{jk}(x) = 2^{-\frac{j}{2}} b(2^{-j}x - k) \tag{4.26}$$

where *j* and *k* are integers that scale (change the wavelets width) and dilate (change the wavelets location) the mother function. Mother functions are scaled by powers of two, and translated by integers, which makes wavelet bases self-similar. Orthonormality makes it easy to compute the weights **w**. [Chui, 1992; Graps, 1995]Sometimes the wavelets may be described as a frame instead of a basis [Daubechies, 1992; Sjöberg et. al., 1995]. Also the basis functions in fuzzy logic systems of the form (2.48) have resemblance with frames, because they can be almost linearly dependent

Kernel estimators

Usually the kernel methods use kernel functions that decrease in a smooth fashion as the distance from the data point **x** is increased. The mapping is defined by

$$y(\mathbf{x}) = \sum_{n=1}^{N} w_n K\left(\frac{\|\mathbf{x}-\mathbf{x}^n\|}{\lambda}\right) \tag{4.27}$$

where $K(z)$ is called a kernel if it satisfies

$$\lim_{\|z\|\to\infty} |z\| K(z)| = 0,$$
$$\int |K(z)| dz < \infty$$
$$\sup_{z\in R} |K(z)| < \infty,$$
$$\int K(z) dz = 1$$

The parameter λ is the window-width (bandwidth, scaling factor), but can also be regarded as a variance. Research to date suggests that the choice of the kernel is relatively unimportant compared to the choice of λ [Hastie & Tibshirani, 1990]. The norm in (4.27) is usually a squared distance, but also the Mahalanobis distance, which incorporates covariance information, has been proposed. Typically the kernel mapping (kernel smooth) is computed as

$$y(\mathbf{x}) = \frac{\sum_{n=1}^{N} y_n K\left(\frac{\|\mathbf{x}-\mathbf{x}^n\|}{\lambda}\right)}{\sum_{n=1}^{N} K\left(\frac{\|\mathbf{x}-\mathbf{x}^n\|}{\lambda}\right)} \tag{4.28}$$

Kernel regression can be used to estimate regression functions from noisy data. The generator of the training data is described by a probability density function in the joint input-target space. If using Parzen kernel estimator with Gaussian kernel functions, the estimator is

$$\tilde{p}(\mathbf{x},\mathbf{t}) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{(2\pi h^2)^{(d+\varepsilon)/2}} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}^n\|^2}{2h^2} - \frac{\|\mathbf{t}-\mathbf{t}^n\|^2}{2h^2}\right) \tag{4.29}$$

where N is the number of data points, d and c are dimensions of input and output spaces.

The optimal mapping (in the presence of noise) is given by forming conditional average $\langle \mathbf{t} \mid \mathbf{x} \rangle$ of the target data conditioned on the input variables. Conditional average can be formed by using joint density as follows

$$\langle \mathbf{t} \mid \mathbf{x} \rangle = \frac{\int \mathbf{t}\, p(\mathbf{x},\mathbf{t})\, d\mathbf{t}}{\int p(\mathbf{x},\mathbf{t})\, d\mathbf{t}} \tag{4.30}$$

Nadaraya-Watson estimator (4.31) is formed by substituting (4.29) into (4.30) [Nadaraya, 1964].

$$y(\mathbf{x}) = \frac{\sum_{n=1}^{N} \mathbf{t}^n \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}^n\|^2}{2h^2}\right)}{\sum_{n=1}^{N} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}^n\|^2}{2h^2}\right)} \tag{4.31}$$

Estimator (4.31) can be viewed as a normalized radial basis function expansion (or as an FLS with Gaussian membership functions) where basis functions are placed on each data point. If we want to reduce the number of basis functions, we may replace the estimator with an adaptive mixture model. The result is a normalized radial basis function expansion in which the centers **c** of basis functions are not constrained to be the data points:

$$\mathbf{y(x)=}\frac{\sum_{i=1}^{M}\mathbf{w}_i \exp\left[-\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{2h^2}\right]}{\sum_{i=1}^{M}\exp\left[-\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{2h^2}\right]} \tag{4.32}$$

When considering this result, both fuzzy logic system of the form (2.47) with Gaussian membership functions and normalized RBF expansion turn out to have good statistical properties.

Bin smoother (regressogram)

Bin smoother can also be described as expansion in basis functions. For example, consider one input one output -case. Bin smoother partitions the x -values into number of disjoint regions, and then averages the y -values in each region. Those regions are defined such that they each have the same number of points. Let the region be

$$R_k = \left\{n; c_k \le x^n < c_{k+1}\right\} \tag{4.33}$$

where $k = 0,..., K-1$ and c_k are cut points. The mapping can then be given

$$y(x) = \sum_{i=1}^{k} \bar{y}_i b_i(x) \tag{4.34}$$

where $\bar{y}_i$ is the average of the y -values in region R_i and

$$b_i(x) = \begin{cases} 0 \quad , \; x \notin R_i \\ \\ 1 \quad , \; x \in R_i \end{cases} \tag{4.35}$$

The regions (and thus the basis functions) do not overlap which is the reason tot the non-smooth nature of this mapping. If the number of points chosen to be included in each region is set to one, bin smoother reduces to B_0 -interpolation.

The nearest neighborhood methods

The nearest neighborhood methods differ from the bin smoother such that the k closest data points to each x_i are picked to be included in the hypercube, which means that basis functions formed will overlap. If the averages are calculated in each neighborhood (region) the resulting smoother is called a *moving average*. It has a tendency to flatten out the trends near endpoints. This can be solved by computing a least-squares line instead of mean in each region. This gives rise to a *running-line smoother*. It can be easily seen that also the nearest neighborhood models can be described by the expansion (4.4). The size parameter of the cubes controls the smoothness of the mapping.

Projection pursuit regression

Projection pursuit regression [Friedman & Stuetzle, 1981]

$$y(\mathbf{x}) = \sum_{i=1}^{k} w_i b_i(\mathbf{x}, \theta) + w_0 \tag{4.36}$$

where

$$b_i(\mathbf{x}, \theta) = g_i(\mathbf{u}_i^T \mathbf{x} + u_{i0})$$

can be represented as a two-layer feed-forward neural network with b_i s as non-linear activation functions (i.e., ridge type basis functions). The difference with ordinary twolayer feedforward neural network is that each hidden unit may use different activation functions and they are determined from the data. For the same number of entries into the approximation, k, projection pursuit has less bias but more variance than the ordinary neural networks. Projection pursuit regression of the form (4.36) can be regarded as a generalization of the MLP.

The convergence rate of $\boldsymbol{O}(\sqrt{k})$ is achieved if the approximation is done in a function space

$$\int_{R^d} \left| \hat{f}(s) \right| ds < \infty,$$

and sin function is selected for the activation function and the norm is chosen to be a L_2 -norm [Jones, 1992]. The same rate of convergence for MLPs with sigmoid activation functions and with L_2 -norm is achieved in a function space [Barron, 1993]

$$\int_{R^d} \|s\| \left| \hat{f}(s) \right| ds < \infty$$

For comparison, the corresponding norm and function space for RBF are $L_2(\mathbf{R}^2)$ and

$$\mathbf{H}^{2m,1}(\mathbf{R}^d)$$

where $2m > d$ [Girosi, 1993].

Additive models

An additive model is defined by

$$y(\mathbf{x}) = \sum_{i=1}^{d} y_i(x_i) + w_0 + \varepsilon \tag{4.37}$$

where ε is noise and $E(\varepsilon) = 0$ and $\text{var}(\varepsilon) = \sigma^2$ $y_i s$ are (univariate) functions, one for each input dimension. Additive model (4.37) may also have component functions defined on two or more dimensional input spaces.

Once the additive model is fitted to data, d coordinate functions can be plotted separately to examine the roles of the inputs in modeling the response. The difficulty with these models is that the interactions between the input variables cannot be modeled.

In fuzzy logic systems, the additive modeling can be incorporated by replacing the product (AND-operation) by a sum (OR). Additive modeling has also been done in RBF -networks by utilizing Gaussian bar basis functions. Mills and Harris [1995] have made an effort to alleviate the curse of dimensionality in neurofuzzy systems by splitting an d dimensional function y (**x**) into Analysis Of Variance (ANOVA) decomposition

$$y(\mathbf{x}) = \sum_{i=1}^{d} y_i(x_i) + \sum_{i=1}^{d} \sum_{j=i+1}^{d} y_{ij}(x_i, x_j) + \ldots + y_{1,\ldots,d}(x_1, \ldots, x_d) + w_0 \tag{4.38}$$

where w_0 is the bias and the other terms represent the combinations of univariate, bivariate, etc., subfunctions that additively decompose the function y. The advantage of ANOVA representation is that each subfunction can be a neurofuzzy system and the network transparency can be retained. The output is a linear function of the weights. Mills and Harris illustrate the reduction in the number of fuzzy rules by approximating Powell's function

$$y(\mathbf{x}) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

The ANOVA system will use approximately 200 rules, when the conventional neurofuzzy system uses over 2000. An algorithm (ASMOD) for searching ANOVA has been proposed by Kavli [1994] and the result of applying it to the modeling the pitch of six degrees of freedom autonomous underwater vehicle is given in [Mills & Harris, 1995].

Hastie and Tibshirani [1990] have generalized additive models to take the form

$$y(\mathbf{x}) = g\left(\sum_{i=1}^{d} y_i(x_i) + w_0\right) \tag{4.39}$$

where $y_i s$ are non-linear functions and g is a logistic sigmoid function. Mapping (4.39) has the same problem as (4.37); functions of the form $x_1 x_2$ cannot be modeled.

Hinging hyperplane

Sjöberg and Pucas show in [1995] that the hinging hyperplane is over-parametrized in its original form introduced by Breiman [1992]. By letting the basis functions be

$$b(x) = \begin{cases} 0 & , x < 0 \\ \pm x & , x > 0 \end{cases} \tag{4.40}$$

they write hinging hyperplanes model as

$$\sum_{i=1}^{k} b_i \left(\mathbf{w}_i^T \mathbf{x} + w_{i0}^{(1)}\right) + \mu^T \mathbf{x} + w_0^{(2)} \tag{4.41}$$

where μ is a parameter vector. Hinging hyperplane is a ridge construction with an additional linear term. The convergence rate of $\boldsymbol{O}(\sqrt{k})$ is achieved if the function space is selected as

$$\int_{R^d} \|s\|^2 \left| f(s) \right| ds < \infty$$

and the norm is a L_2 -norm [Breiman, 1992].

CMAC

The Cerebellar Model Articulation Controller (CMAC) was first proposed by Albus [1975]. CMACs inputs and outputs operate with binary-valued signals. In its simplest form it is a distributed look-up table which generalizes locally. It can be represented as a weighted sum of (basis) functions where these functions can take various forms. They can be functions that act on norm (like RBF) or they can be formed by combining d univariate basis functions using a product or min operator (direct relationship to fuzzy systems). The specialty of CMAC is that the number of basis functions that contribute to the output is not a function of the input space dimension. The generalization parameter determines the number of nonzero basis functions for any input. One of the characteristics of CMAC is its fast learning speed. A good description of CMAC can be found in [Brown & Harris, 1994].

4.4 Posterior Probability in FLS and Neural Networks

Posterior probability (Bayes theorem) ⇔ Fuzzy logic (Center of Gravity)

As shown in Chapter 2, the fuzzy basis function can be interpreted as conditional probability

$$P(C_k|\mathbf{x}) = \frac{C_k \quad)}{\sum_{=1} \quad C \quad)} = b_{C_k}(\mathbf{x}) \tag{4.42}$$

where represents the membership degree of **x** (subjective probability) being a member

of th class. Ripley [1996] sees also the link to the mixture models where the (**x**) can be interpreted as a posterior probability of having been generated by a component (class) of the mixture.

Also the posterior probability gives the probability of class membership:

$$\mathbf{x}) = \frac{\mathbf{x}}{\mathbf{x}} \tag{4.43}$$

Let

$$\mathbf{x}, \quad \mathbf{x}, \quad \mathbf{x}, \quad \mathbf{x} \tag{4.44}$$

be the probability of occurrence of input-output pair (**x**,) . The above decomposition is possible since the prior probabilities sum to unity. The optimal output of network can be predicted such that the expected error is minimized by

$$\mathbf{x} \quad \mathbf{x}, \quad \frac{\mathbf{x}}{\mathbf{x}} \tag{4.45}$$

Replacing the posterior probability part of (4.45) by (**x**) and **x,** by we get

$$\mathbf{x} = \mathbf{(x)} \tag{4.46}$$

Similarities of posterior probability and fuzzy basis function:
- they describe the class membership
- they sum to unity

Posterior probability (Bayes theorem) ⇔ Neural networks (Classification)

The representation here follows [Bishop, 1996]. An important characteristic of neural network classifier is that network outputs provide estimates of posterior probabilities. When posterior probabilities are estimated accurately classification error rate is minimized and outputs can be regarded as probabilities (sum to one) [Lippmann, 1993].

In classification problems vector **x** is assigned to class if the discriminant function

$$(\mathbf{x}) > (\mathbf{x}) \tag{4.47}$$

The discriminant function can be chosen to be

$$(\mathbf{x}) \quad \mathbf{x}) \tag{4.48}$$

Because the denominator of (4.43) does not depend on class label , (4.48) can be rewritten

$$(\mathbf{x}) = \mathbf{x} \tag{4.49}$$

By taking a logarithm of (4.49) (could be replaced by any monotonic function) we can write discriminant function in form

$$(\mathbf{x}) = \ln \mathbf{x} + \ln \tag{4.50}$$

If the class-conditional density functions are independent normal distributions and the covariance matrices are taken to be equal for various classes, the discriminant functions can be written in the form (details can be found in [Bishop, 1996])

$$(\mathbf{x}) = \mathbf{w} \mathbf{x} + \tag{4.51}$$

where

w

Also the logistic sigmoid activation function acting on the linear sum allows the outputs to be interpreted as posterior probabilities, i.e., the outputs of

(4.52)

4.5 Basis Function Selection

It is very difficult to decide, which set of basis functions (and thus which approximation method) should be used in approximation at hand. The problem is to choose the basis in the sense that the "worst case" will be minimal. This means that we choose a set of basis functions that does not give a worse result than the other sets [Pinkus, 1985].

Definition 4.1 Let be a normed linear space and any -dimensional subspace of . If there exists a for which the distance

then is a *best approximation* to from .

Definition 4.2 Let be a subset of . The quality of subspace in approximating is measured by

and is called the *deviation of from* . Definition 4.2 means that for each in we look for a best approximation in and then we choose the worst of those best approximations and the distance between it and the point in *A*. This gives the *worst case*. The measure tells how well the "worst element" can be approximated from .

Definition 4.3 A subspace of for which -width

an - dim subspace

is an *optimal subspace* for .

In practice it is usually impossible to determine optimal subspaces for . However, there exist choices of and for which optimal subspaces have been found. For example, classes of periodic functions have been heavily studied and some results concerning their optimal subspaces can be found in [Pinkus, 1985].

Hilbert spaces have an advantage compared to other functional spaces: the best approximant can be found by orthogonal projection from into a subspace. If is a compact -dimensional subspace of Hilbert space (-norm, Sobolev-space) and is an orthonormal basis for , then the best approximation to is

where is the set of non-increasing positive numbers. The best approximation operator in a Hilbert space is a linear projection. If the norm used in Sobolev function class is -norm where , these classes are not convex (they consist of functions with sparse singularities or functions that are not uniformly smooth), and can not necessarily be approximated using linear subspaces [Juditsky et. al., 1995]. Juditsky et. al. [1995] recommend the use of Besov function spaces (with Besov norm) and spatially adaptive basis functions (free knot splines [Petrushev&Popov, 1987], Wavelets) for approximating such functions. Any function from Besov space can be approximated by splines and any function which have a good spline approximant belongs to a certain Besov space [Juditsky et. al., 1995]. However, such spline approximations are hard to compute because of the free knot placements: optimal positions are difficult to find. Wavelets are as good as splines but are much more easily constructed [Juditsky et. al., 1995].

Simple Comparisons

MLP / Projection pursuit regression (more variance)

+ high dimensional mapping
+ can learn to ignore redundant inputs
(-) adjusting any weight affects the output of the network for every input
- network not transparent (each input can appear many times in the model)
- behaves badly on a class of spherical symmetric functions
  [Juditsky et. al., 1995].
- training times long

Polynomial / trigonometric networks

- if too many terms are included, the output can be very oscillatory
- if any of the terms is left away, the output is globally biased
( ) success depends on the identification of the nonlinear hidden nodes
+ strong theoretical background

RBF

+ proven successful for modeling highly nonlinear data
( ) success depends on the identification of the nonlinear hidden nodes
- if globally supported basis functions used, more advanced training algorithms have to be used (each basis function contributes to the output)
+ if globally supported basis functions used, the approximation capability may be better than functions with local support

Truly local basis functions (FLS with triangular membership functions, B-splines, etc.)

( ) forces the network to generalize in a prespecified manner (needs a priori knowledge)
+ piecewise nature prevents oscillations and overfitting
- curse of dimensionality

Regression splines

- choosing the appropriate number and position of the knots difficult
- if small number of knots used, it may have some disturbing nonlocal behavior
+ computationally attractive (provided knots are given)
+ standard linear model estimation can be applied
- smoothness of the estimate cannot easily be varied (no single smoothing parameter)

Additive models

+/- work well when the underlying structure is additive or has mild lower order product interactions

Conclusion: The choice of models based on the properties of basis functions

- If the dimension of input space is large and data are available, then basis functions obtained by ridge construction should be used to avoid the curse of dimensionality.
Neurofuzzy network with input space reduction method (Principal Component Analysis, additive models based on ANOVA, shrinking by regularization) could be tried if the transparency of the network is required.
- If the dimension of input space is large and no data but linguistic information is available, then fuzzy logic system should be used. The curse of dimensionality can be avoided by appropriate rules (no redundant information).
- If the dimension of input space is small and data available, RBF and Wavelets (coefficients can be estimated efficiently) are good choices. If and no noise: spline interpolation.
- If the dimension of input space is small and only linguistic information available, the only solution is to use fuzzy logic systems.
- If the function is not uniformly smooth (includes spikes and jumps), spatially adaptive methods should be used. For wavelets, a complete analysis both for approximation and estimation do exist [Juditsky et. al., 1995].

Chapter 5

ra t C p t eth

Intelligent control and modelling requires algorithms which are capable of operating in time-varying environment. By these algorithms soft computing methods can learn significant information and adapt to dynamical changes. Many of the learning rules can be applied to a wide range of different neural and fuzzy systems. New algorithms can be developed independently of the model to which they are applied [Harris & Brown, 1994]. However, their suitability for a particular problem should be determined.

Learning required:

- to improve the performance
- reduce design costs
- prior knowledge about the plant is unavailable or only partially known
- time-varying plant

Important in learning:

- generalization from a limited number of training examples
- algorithm has to extract the relevant knowledge from the data
- model structure: the chosen model structure influences the rate of convergence of the learning and determines the type of learning that should be used. Each task requires an appropriate learning system, as the problem structures are generally different. Therefore learning systems should be aimed at particular problems and should always use the maximum amount of relevant *a priori* information.

Supervised learning (active learning):

- the desired network output must be available (requires the availability of an external teacher).
- reduces the output error by minimizing the multidimensional error-performance surface
- can be performed in an on-line or off-line manner

Important in on-line learning:

- learn in real-time
- algorithm must be proved to converge and to be stable

If algorithms do not satisfy these properties, the off-line training algorithms should be used.

The solution of minimization problem, which is based entirely on the training data, is not unique. Nothing is known about the function between the observations. There is a infinite number of functions that can interpolate the data points such that the optimization criterion (sum of squares) yield to zero. If there is no noise, one of the infinite number of functions will be the target function . But int the presence of noise, none of them will be . Thus the optimization problem of this kind is ill-posed. The finite data set does not sufficiently constrain the function space.

Usually the set of solutions is restricted to a smaller set of possible functions.This can be done by a choice of training method. Applying a particular restriction may give a unique solution, but different restriction may lead to a different unique solution [Friedman 1993]. The prior knowledge can be used to provide the necessary constraints. Usually, this knowledge consist of information about the smoothness of the function, and it is incorporated in the minimization problem with the aid of regularization term (penalty) [Tikhonov & Arsenin, 1977]. All the available prior knowledge (constraints) and empirical data should be employed to make the identification problem better conditioned.

Unsupervised learning:

- organizes the network's structure
- is based only on the training inputs, desired output values are not available.
- clustering (dimensionality reduction)

5.1 The Basics

The data used in training will be labeled with the index so that each data point consists of pair (**x** ,), where **x** is the input and is the desired output, ie. a target. Our model, i.e., nonlinear mapping is

x ,) (5.1)

where **w**,) consists of weights and parameters of basis functions. All the parameters are estimated from the data. Training in standard curve fitting problem is done by minimizing an error between the output of the mapping and the targets:

x , (5.2)

If is a linear function of the parameters , then is a quadratic function of . A fuzzy system of the form (2.47), which has Gaussian membership functions with fixed parameters (centers, variances) and weights are to be determined, is a linear function of the weights. This means that the minimum of *E* can be found by using linear optimization methods, which yield to a solution of linear algebraic equations. More generally, this apply to all mappings of the form (4.4). (Note:

Although system may be a linear function of the parameters, it doesn't mean that it is a linear function of the input variables).

For example, differentiating $E(\mathbf{w}) =$ with respect to gives

(5.3)

In matrix notation (5.3) is

$$(\mathbf{B}\ \mathbf{B})\mathbf{w} = \mathbf{B}\ \mathbf{t} \tag{5.4}$$

where **B** is a matrix with as its elements, vector **w** consists of elements and **t** is a vector of target values . Methods for solving (e.g., SVD) $\mathbf{Aw} = \mathbf{b}$ type of linear systems can be found in [Press, Golub] and in many other books dealing with matrix computations. If the matrix $\mathbf{B}\ \mathbf{B}$ is not ill-conditioned (5.4) could be solved directly, for example in Matlab:

```
w =
```

If the parameters of basis functions that minimize the error function are searched, or the basis function is a non-linear function of the weights, then a closed form solution is not possible. If the basis functions are differentiable (excludes the use of triangles, trapezoids, or functions with corners), the derivatives of the error function with respect to parameters can be evaluated and various gradient-based optimization methods (e.g., backpropagation in network structures) can be utilized. In addition to basic gradient method, more sophisticated optimization methods do exist. Most used are Newton, Quasi-Newton, Conjugate gradient and Levenberg-Marquardt algorithms.

A root-mean-square (RMS) error can be used to assess the capability of the network to generalize to new data. It is given by an equation

(5.5)

where $*$ is the vector of parameters in the minimum of the error function .

Local Minima

Researchers have proposed hundreds of schemes to lessen the possibility to get stuck in local minima, but none has removed it. This means for example that, if neural network training algorithms are used to tune fuzzy rules, there is no guarantee that the final result will be much better than the first set of rules. That resembles a trial and error method. Local optimality may be the best neural networks and fuzzy systems can ever achieve [Kosko, 1997].

The best thing is to have initial weights such that they are close enough to the optimal ones (for example, FLS can be used to incorporate prior information). Another alternative is to choose them by random. The parameters of basis functions should be selected such that their supports cover the important regions of the input space. To avoid getting stuck in local minima, training algorithms based on randomness could be used (e.g. genetic algorithms, random serch directions).

Basis function selection in linear networks

Non-linear networks have advantage of being able to adapt their basis functions. Same kind of adaptivity can be incorporated in linear networks by selecting a subset of basis functions from a larger set of candidates. That way the linearity of the network can be retained and flexibility can be added to it. The necessity of using non-linear optimization methods for finding the parameters of basis functions is avoided and replaced by a search in the space of possible basis function subsets [Orr, 1995, 1997].

Subset selection is used in linear regression theory and is also the principal problem in forming the linear combination of the basis functions as stated in chapter 4. One variant is the forward selection in which basis function are selected one at a time and if the basis function under examination reduces the sum of squared errors the most it is added to the network. The selection is stopped when the predicted average error (calculated by using Bayesian information criterion, unbiased estimate of variance etc.) on new data starts to increase [Efron et. al., 1993; Orr, 1997]. For example, a stopping criterion

$$\frac{\mathbf{w}^t \mathbf{B}^t \mathbf{B}\mathbf{w}}{} \qquad (5.6)$$

where , could be used.

A more efficient approach to perform forward selection is to use *orthogonal least squares* (OLS) [Chen & Cowan, 1991], which can efficiently be applied also to fuzzy logic systems [Wang, 1994]. In fuzzy logic systems the OLS transforms the set of fuzzy basis functions into a set of orthogonal basis vectors, and uses only significant ones to form the final fuzzy basis function expansion. For the algorithm and an example of controlling the non-linear ball-and-beam system using fuzzy logic system in which the basis functions are selected by OLS see [Wang, 1994].

When not using regularization it is worth using OLS because it makes no difference to the results but speeds up the computations [Orr, 1997]. However, when regularization is utilized, OLS is not appropriate since there is a difference which has to do with the nature of regularization term in the cost [Efron et. al., 1993].

Unsupervised training

Unsupervised clustering techniques are appropriate for finding the basis function centers from the data. Clusters approximate the distribution of the data. Some of the methods are listen in the following table. More information about clustering data can be found in [Kaufman & Rousseeuw, 1990].

K-means clustering	Moody& Darken (1989)
Batch version of K-means	Lloyd (1982)
Self-organizing feature map	Kohonen (1982)
Adaptive Vector Quantization (fuzzy)	Kong & Kosko (1992)
modified SOM (fuzzy, RBF)	Nie & Lionkens (1993)
modified SOM (fuzzy)	Lin & Lee (1991)
modified K-means (fuzzy)	Kwon & Zervakis (1994)

Table 5.1 Clustering techniques

Parameters of the basis functions can be optimized by maximum likelihood if the density of the input data is modeled by a mixture model

$$\mathbf{(x)} \qquad \mathbf{(x)} \tag{5.7}$$

where are the mixing coefficients [Bishop, 1996]. The likelihood function

$$\mathbf{x} \ )$$

is maximized with respect to and the parameters of the basis functions. Nonlinear optimization algorithms or EM (expectation-maximization) algorithm may be used in maximization. After optimization mixing coefficients can be discarded. Details are given in [Bishop, 1996].

5.2 Network Mapping

Sum-of-squares error in the case of infinite data becomes

$$\mathbf{x,} \qquad \mathbf{,x)} \quad \mathbf{x}$$

$$\mathbf{x,} \qquad \mathbf{x)} \quad \mathbf{x)} \quad \mathbf{x} \tag{5.8}$$

After substituting conditional average

$$\mathbf{x} \qquad \mathbf{x}$$

$$\mathbf{x} \qquad \mathbf{x} \tag{5.9}$$

into (5.8) the sum-of-squares error can be written

$$\mathbf{x}, \quad \mathbf{x} \quad \mathbf{x} \ \mathbf{x} \quad \mathbf{x} - \mathbf{x} \quad \mathbf{x} \ \mathbf{x} \tag{5.10}$$

For determining network parameters, the second term can be neglected. The absolute minimum of occurs when

$$\mathbf{x}, \quad \mathbf{x} \tag{5.11}$$

where is the parameter vector at the minimum of error function .

The requirement for the nonlinear mapping is that it is sufficiently general to be able to make the first term of sufficiently small, i.e., it must satisfy the universal approximation property. In classification problems, (5.11) becomes

$$\mathbf{x}) \quad \mathbf{x}) \tag{5.12}$$

where is th class and pattern should be classified to the class with largest posterior probability. This is also done in fuzzy logic system that uses maximum method as defuzzification. Suppose that the target data is given by

$$\mathbf{x} \tag{5.13}$$

where g is some unknown generator of the data and e is zero-mean noise. The network output, given by (5.11)

$$\mathbf{x} \quad \mathbf{x} \quad \mathbf{x} \quad \mathbf{x} \quad (\mathbf{x}) \tag{5.14}$$

discovers the underlying generator of data. The second term of can be written

$$\mathbf{x}) \quad \mathbf{x}) \ \mathbf{x} \tag{5.15}$$

where $(\mathbf{x})$ is the variance of the target data. If the first term of vanishes the residual error is given by (5.15)

5.3 Regularization

The lack of sufficient amount of data makes the model identification problem often illconditioned. This results in that some basis functions may not have sample points on their supports which means that the local parameters can be chosen arbitrarily and thus good estimates cannot be calculated (in Fig. 5.1 RBF network is used to approximate function). Other causes for unsatisfactory performance

may be the non-optimal choice of model structure (no sufficient amount of a priori information) or collinear ties. In the case of Gaussian basis functions or other basis functions that do not have compact support the resulting model may have oscillations - especially when the data is approximately uniform but some discontinuities do exist (Fig. 5.2).

Figure 5.1 Function and its approximation when missing data

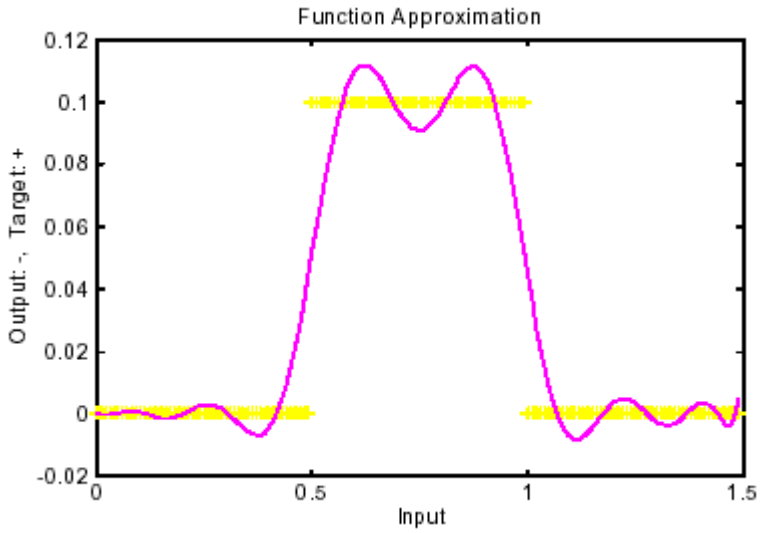


Figure 5.2 Oscillations in function approximation

One effective way to improve the robustness of standard parameter identification algorithms and to avoid overfitting is to use regularization. Regularization controls the effective complexity of the model by adding a penalty term (regularization term) to the error function. When the regularization is applied, the error function becomes

(5.16)

where controls the influence of the regularization term. As a result, we hope to get a model which has a optimum complexity such that the combination of bias and variance is minimized; neither of them becomes dominating. Regularization has been applied (or proposed to be used) with neural networks [Bishop, 1996; Girosi et. al., 1995], statistics, spline modeling, fuzzy logic systems and system identification [Sjöberg et. al., 1995].

Regularization can be utilized in the determining of the important parameters in (4.4). The model is offered an amount of parameters that is surely known to be enough for the modeling task and then the redundant ones are pruned by regularization. By using *weight decay* as a regularizer, i.e.

(5.17)

where is the number of all parameters in the model, it can be shown [Sjöb,Moody] that when minimizing (5.17) the number reduces to

(5.18)

where are eager values of the Hessian of the error function . A small eager value of Hessian means that the corresponding parameter is not very essential in the model and can thus be neglected. The parameter in (5.18) controls the efficient number of parameters. By increasing its value the model becomes smaller, which yield to smaller variance but larger bias. On the contrary, by making smaller the variance becomes larger and bias smaller.

By replacing the regularization term in (5.17) by

(5.19)

where is a mean, this regularizer corresponds to a prior Gaussian distribution for the parameters [Sjöberg et. al., 1995; MacKay, 1992]. Redundant parameters can be found by first associating with them a large prior and then minimizing the error function. If the bias stays small they can be excluded [MacKay, 1992].

Fixed or variable regularization parameter can be combined with forward selection (selection of basis functions) and can lead to improved fits [Orr, 1995].

Regularization and default model

Another way to utilize regularization is to penalize the model, when it differs from some nominal model (default model) based on some prior information [Suykens et. al., 1994]. This nominal model could be a partially or completely known model (e.g., a fuzzy model that has been derived from the expert knowledge). In this case the penalized error function takes the form

$$\mathbf{x}\ )-\ \mathbf{x} \tag{5.20}$$

where $(\mathbf{x}\ )$ is output of a (fuzzy) model and $(\mathbf{x}\ )$ is output of a model which is tried to be optimized. See chapter 6 for details about using default models in modeling of dynamical systems. More about this method in chapter 6.

Different stabilizers

A relatively new suggestion is to use an *approximated version of Tikhonov stabilizer* in identifying the parameters of fuzzy systems [Johansen, 1997a]

$$\mathbf{w} \tag{5.21}$$

where **W** is a positive definite diagonal weighting matrix, :s are the parameters of fuzzy sets, is a measure of how close the fuzzy sets and are. is close to one for adjacent sets and close to zero for distant sets. One choice to calculate :s is to use multivariate Gaussian function, the width of which is the average deviation of the fuzzy membership functions. The purpose of (5.21) is to attract the parameters of adjacent sets towards each other. *Tikhonov regularization*: In one-dimensional case, if the smoothing functional is defined by

(5.22)

The formula above is a special case of the more general Tikhonov regularization term [Haykin, 1994]

$$\mathbf{x}\ )\ \ \mathbf{x} \tag{5.23}$$

which contains a set of derivatives of different orders. Computing integral and derivatives in the (5.23) may be too difficult or needs a lot of computation. A simpler method is to use *Levenberg-Marquardt Regularization*

(5.24)

where is an attraction point. In *weight decay* stabilizer is set to zero.

Linear networks

In the case of linear networks, the optimal weight vector for weight decay regression can be solved by calculating

$$\mathbf{w} = (\mathbf{B}\ \mathbf{B} \quad \mathbf{I} \quad \mathbf{B}\ \mathbf{t} \tag{5.25}$$

where $\mathbf{I}$ is an identity matrix of dimension (number of weights). A generalization of weight decay is done by attaching a separate (local) regularization parameter to each basis function:

(5.26)

which leads to an optimal weight of

$$\mathbf{w} = (\mathbf{B}\ \mathbf{B} + \mathbf{V}) \quad \mathbf{B}\ \mathbf{t} \tag{5.27}$$

where $\mathbf{V}$. Local weight decay (5.26) is supposed to be more suitable than the global weight decay for the target functions with inhomogeneous structure scales.

Regularization and Fuzzy Logic

It has been claimed (based on experince) [Johansen, 1997a] that without regularization a unuform partitioning in the fuzzy model rarely leads to an accurate and robust model. If one do not want or is unable to put a lot of effort on finding an appropriate fuzzy model structure, regularization may be the way to improve the robustness and to get a better model. More information about applying regularization to fuzzy logic system can be found in [Johansen, 1997a].

5.4 The Bias/Variance Tradeoff

As with the conventional regression methods (polynomials, bins, running means, Gaussian kernels, smoothing and regression splines) the fundamental point in solving the regression problem is adjusting the tradeoff between bias and variance such that the network gives good predictions for new data (i.e. network has a good generalization ability). There exists some parameter that governs the smoothing (or model complexity) or at least it can be added by means of regularization. As the amount of smoothing is increased many methods approach the linear regression line and thus smooth out the underlying structure of data. As the amount of smoothing is decreased they approach an interpolation function which does not necessarily give good predictions.The bias-variance tradeoff tells that the number of estimated parameters (number of basis functions) should not be increased above a certain level (which depends on the application). The parameter that is not important for the model fit contributes as much as an important parameter and thus increases the variance error.

The mean squared error of an identified model is defined as

x , (5.28)

where is the parameter vector corresponding to the minimum of the error function. In the case of infinite data set we can write the error function as

x, (5.29)

where the expectation E is with respect to the joint distribution x) of the identification data. The data is assumed to be generated by

x (5.30)

where is a function (generator of the data) and is a stochastic variable (noise) with variance .

The error can be decomposed into bias and variance (as represented in [Bishop, 1996])

(bias) x x x x
(5.31)
variance x x x x

and in the presence of noise the third term is . With a regularized least squares method we get

(5.32)

where is bias, is the variance. For details see [Johansen, 1997a]. If the length of data is kept fixed and regularization parameter *v* decreases, then bias will decrease while the variance will increase. There exists that gives an optimal balance between bias and variance within the model structure. One approach to find , i.e., to minimize with respect to , is the Final Prediction Error Criterion for Regularized models [Larsen & Hansen, 1994]. Other approaches to estimate are crossvalidation techniques (part of the data is used for identification and the remaining for estimating) and the use of separate data sets (independent of the identification data).

5.5 Training Methods in Nutshell

The general consensus is that a damped (= step size optimized) Gauss-Newton algorithm with regularization features for ill-conditioned Hessians should be used in an off-line manner (batch version of training) [Sjöberg et. al., 1995]. Generally, the parameters are updated in the following manner

$$\mathbf{D} \quad \mathbf{g} \tag{5.33}$$

where is step size, **g** is the gradient of error function (derivatives taken with respect to), **D** is a matrix that modifies the search direction. The choice of direction matrix depends on the method used:

Gradient direction: $\mathbf{D} = \mathbf{I}$

Gauss-Newton direction: $\mathbf{D} = \mathbf{H}$

Levenberg-Marquardt direction: $\mathbf{D} = \mathbf{H}$ $\mathbf{I}$ (= step size, = 1)

Conjugate gradient: Newton direction constructed from a sequence of gradient estimates.

Quasi-Newton: Hessian **H** replaced by an approximation.

In networks the back-propagation is used to evaluate derivatives (can also be used to evaluate Jacobians and Hessians).

The back-propagation algorithm has also been criticized. The biological neurons do not seem to work backward to adjust their synaptic weights. Therefore, many do not view back-propagation as a learning process that emulates the biological world [Kartalopoulos, 1996]. It also suffers from slow training speed.

Chapter 6

C

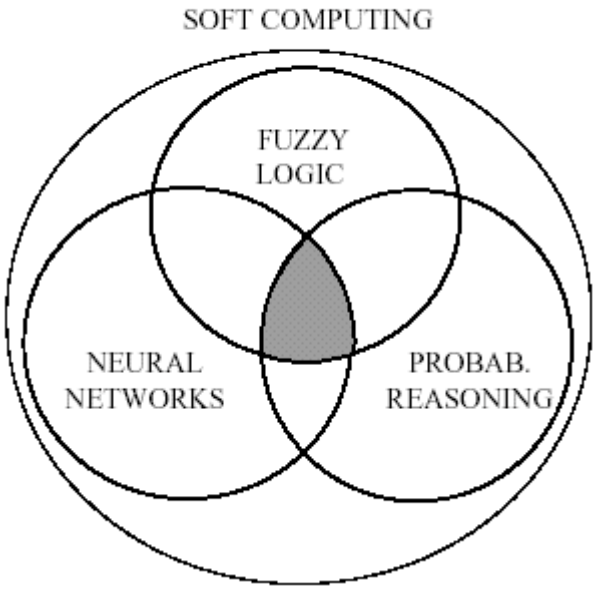


Figure 6.1 Soft computing as a union of fuzzy logic, neural networks and probabilistic reasoning. Intersections include neurofuzzy techniques, probabilistic view on neural networks (especially classification networks) and similar structures of fuzzy logic systems and Bayesian reasoning.

The strengthnesses of each method are summarized in the following table:

Fuzzy Logic
- management of uncertainty
- knowledge representation (as rules)
- approximate reasonings

Neural Networks
- learning from examples (supervised, unsupervised)
- optimization

Probabilistic logic
- management of uncertainty (belief networks)

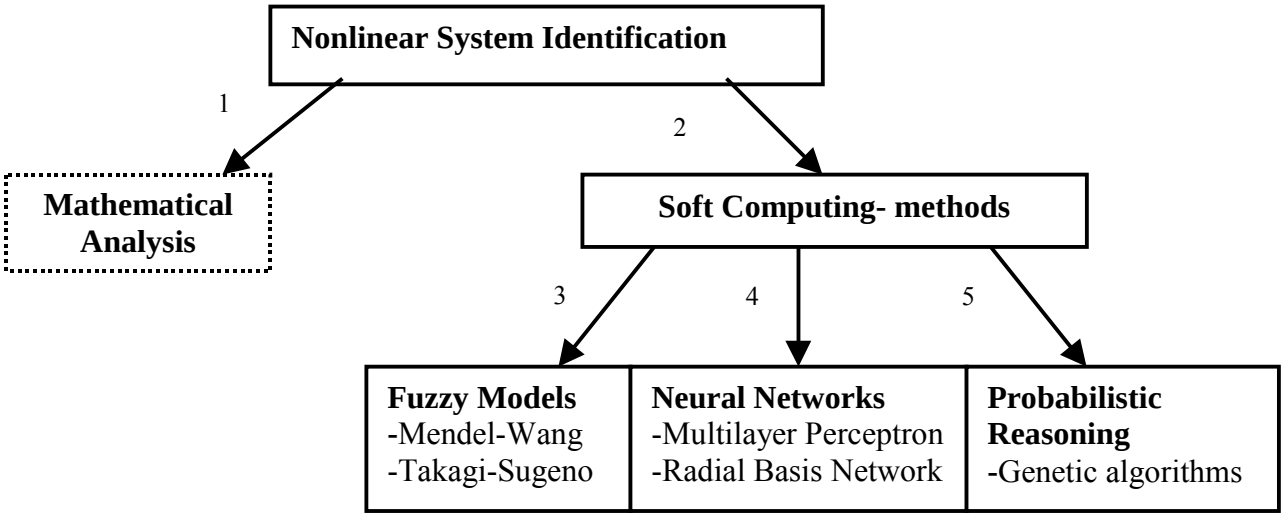


Figure 6.2 The choice of methods

Explanation of arrows in Fig. 6.2:

1. Mathematics gives calculation methods as: differential equations, difference equations, matrices, linear and nonlinear mappings, etc. This direction is preferable, if the dependencies are well-defined and numerical analysis can be used.
2. If the dependencies are not well-defined or difficult. If the dependencies are welldefined, but we want more tolerance for the imprecision.
3. If-then rules can be formulated. If rules need calibration, neural net methods could be used.
4. Rules can not be formulated or the dimension of input space is high.
5. Probabilistic prior information is available.

Soft computing employs techniques which can reason about their environment in a flexible and robust manner. The main difference between AMNs and a certain class of fuzzy systems is the linguistic interpretation of the basis functions. There exists, however, fuzzy networks that do not use linguistic terms to describe their membership functions. In fact, fuzzy logic forms a unifying link between a number of approximation methods in soft computing and in function approximation in general.

References

1. Albus, J.S. (1975). A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC). *Trans. ASME. Journal of Dyn. Sys. Meas. And Control*, vol. 63, no. 3, 220-227.
2. Aliev R. (2001). Soft Computing and its Applications. Azerbaidjan Oil Academy. Baku.2001.
3. Barron, A.R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inf. Theory*, vol. 39, no. 3, 930-945.
4. Berenji, H.R. (1992). Fuzzy Logic Controllers. In R.R.Yager, L.A. Zadeh (eds.) *An Introduction to Fuzzy Logic Applications and Intelligent Systems*. Kluwer Academic Publisher, Boston, MA, Chapter 4.
5. Bilgiç, T.; Türksen, I.B. (1997). Measurement of Membership Functions: Theoretical and Empirical Work. In Dubois and Prade (eds.*): Handbook of Fuzzy Sets and Systems* Vol. 1, Foundations.
6. Bishop, C.M. (1991). Improving the generalization properties of radial basis function neural networks. *Neural Computation*, vol. 3, 579-588.
7. Bishop, C.M. (1996). Neural Networks for Pattern Recognition. Clarendon Press, Oxford.
8. Bossley, K.M.; Brown, M.; Harris, C.J. (1995). Parsimonious Neurofuzzy Modelling. Technical report, University of Southampton, Department of Electronics and Computer Science.
9. Breiman, L. (1993). Hinging hyperplanes for regression, classification and function approximation. *IEEE Trans. Information Theory*, 39 (3), 999-1013.
10. Brown, M. and Harris C. (1994). Neurofuzzy adaptive modelling and control. Prentice Hall, New York.
11. Buckley, J.J; Hayashi, Y. (1993). Numerical relationships between neural networks, continuous functions, and fuzzy systems. *Fuzzy Sets and Systems*, 60, 1-8.
12. Buckley, J.J; Hayashi, Y. (1994). Fuzzy neural networks. In Zadeh; Yager (eds.): Neural Networks and Soft Computing. Van Nostrand Reinhold, New York, 233-249.
13. Buhmann and Powell, M.J.D. (1990). Radial Basis Function Interpolation on an Infinite Grid. In J.C..Mason, M.G. Cox (eds.): *Algorithms for Approximation II*, Chapman and Hall, London, 146-169.
14. Bulsari, A. (1992). Fuzzy Simulation by an Artificial Neural Network. *Eng. Appl. Artificial Intelligence*, vol. 5, no. 2, 401-406.
15. Carpenter, G.A.; Grossberg, S.; Rosen, D.B. (1991). Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an adaptive Resonance System. *Neural Networks*, vol. 1, 759-771.
16. Carroll, S.M.; Dickinson, B.W. (1989). Construction of neural nets using radon transform. In *Proc. IJCNN I*, 607-611.
17. Castro J. L. and Delgado M. (1996). Fuzzy Systems with Defuzzification are Universal Approximators. In *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 26, no. 1.
18. Chameau, J.L.; Santamarina, J.C. (1987). Membership functions part I: Comparing method of measurement, *Int. Journal of Approximate Reasoning*, vol.1, 287-301, part II: 303-317.
19. Cheeseman P. (1986). Probabilistic versus Fuzzy Reasoning. In Kanal L. N. And Lemmer J. F. (eds.): *Uncertainty in Artificial Intelligence*. Elsevier Science Publishers, Amsterdam, Netherlands.
20. Chen S., Cowan F.N., Grant P.M (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, vol. 2, no. 2, March 1991, 302-309.
21. Chen, T.; Chen, H.; Liu, R. (1990). A constructive proof of Cybenko's approximation theorem and its extension. In LePage; Page (eds.), *Computer Science and Statistics*, 163-168.

22. Chen T., Chen H. (1995). Universal Approximation to Nonlinear Operators by Neural Networks with Arbitrary Activation Functions and Its Application to Dynamical Systems. *IEEE Transactions on Neural Networks*, vol. 6, no. 4, July, 911-917.
23. Chen T., Chen H. (1995). Approximation Capability to Functions of Several Variables, Nonlinear Functionals, and Operators by Radial Basis Function Neural Networks. *IEEE Transactions on Neural Networks*, vol. 6, no. 4, July 1995, 904-910.
24. Chen T., Chen H. (1993). Approximation of Continuous Functionals by Neural Networks with Application to Dynamical Systems. *IEEE Transactions on Neural Networks*, vol. 4, no. 6, November 1993, 910-918.
25. Choi, J.J.; Arabhashi, P.; Marks, R.J.; Claudell T.P. (1992). Fuzzy parameter adaptation in neural systems. *Int. Joint Conference on Neural Networks*, vol. 1, 232-238.
26. Chui, C. (1992). Wavelets: a tutorial in theory and applications. Academic Press, San Diego.
27. Chung, F.L.; Lee, T. (1994). Fuzzy competitive learning. *Neural Networks*, vol. 7, no.3, 539-551.
28. Cox, E. (1992). Integrating Fuzzy Logic into Neural Nets, *AI expert*, June, 43- 47.
29. Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Contr., Signals Syst*., vol. 2, no. 4, 303-314.
30. Daubechies, I. (1992). Ten lectures on wavelets. CBMS-NSF regional conference series in applied mathematics. Society for Industrial and Applied Mathematics, Philadelphia.
31. Driankov, D.; Hellendoorn, H.; Reinfrank, M. (1993). An introduction to fuzzy control. Springer, Berlin.
32. Dyn, N.. (1987). Interpolation of scattered data by radial functions. In Chui; Schumaker; Utreras (eds.): Topics in multivariate approximation. Academic Press, New York.
33. Efron, B.; Tibshirani, R.J. (1993). An introduction to the Bootstrap. Chapman and Hall.
34. Friedman, J.H. (1991). Multivariate adaptive regression splines. *Annals of Statistics* 19.
35. Friedman, J.H. (1993). An Overview of Predictive Learning and Function Approximation. In Cherkassky, V.; Friedman, J.H.; Wechsler, H. (eds*): From Statistics to Neural Networks*. Springer-Verlag.
36. Friedman, J.H.; Stuetzle, W. (1981). Projection pursuit regression. *Journal of the Amirican of Statistical Ass.*76.
37. Fullér, R. (1995). Neural fuzzy systems. Åbo Akademi, Institute for Advanced Managemen Systems Research.
38. Fullér, R. (2001). Introduction to Neuro-Fuzzy Systems. Advances in Soft Computing Series, Springer-Verlag, Berlin/Heildelberg
39. Funahashi, K. (1989). On the approximate realization of continous mapping by neural networks. *Neural Networks*, vol. 2,183-192.
40. Girosi, F. (1993). Regularization Theory, Radial Basis Functions and Networks. In Cherkassky, V.; Friedman, J.H.; Wechsler, H. (eds*): From Statistics to Neural Networks*. Springer-Verlag.
41. Girosi, F.; Jones, M; Poggio, T. (1995). Regularization theory and neural network architectures. *Neural Computation* 7, 219-269.
42. Golub, Gene H. (1989). Matrix computations. Johns Hopkins University Press, Baltimore.
43. Graps. (1995). An introduction to wavelets. IEEE Computational Science and Engineering, vol. 2, no. 2.
44. Hartman, E.J.; Keeler, J.D.; Kowalski, J.M. (1990). Layered neural networks with Gaussian hidden units as universal approximations. *Neural Computation,* vol. 2, no. 2, 210-215.
45. Hartman, E.J.; Keeler, J.D. (1991). Predicting the Future: Advantages of Semilocal Units. *Neural Computation*, vol. 3, no. 4, 566-578.
46. Hastie, T.J.; Tibshirani, R.J. (1990). Generalized Additive Models. Chapman & Hall, London.

47. Haykin, S. (1994). Neural Networks - A Comprehensive Foundation. Macmillan College Publishing Company, New York.
48. Hisdal, E. (1995). Are grades of membership probabilities? *Fuzzy Sets and Systems*, vol. 25, 325-348.
49. Hiew, H.,L.; Tsang, C.,P. (1997). Fuzzy chaos and recursive partitioning. In Kosko, B.: *Fuzzy Engineering*. Prentice-Hall, Upper Saddle River, New Jersey.
50. Hubbard, J.H.; West, B.H. (1991). Differential Equations - A Dynamical Systems Approach, Part 1: Ordinary Differential Equations. Springer-Verlag.
51. Huntsberger, T.L.; Ajjimarangsee, P. (1991). Neural networks for nonlinear internal model control. *IEEE Proc.-D*, vol. 138, no.5, 431-438.
52. Jager, R. (1995). Fuzzy logic in control. Diss., Delft University of Technology, The Netherlands.
53. Johansen, T.A. (1996). Identification of Non-linear Systems using Empirical Data and Prior Knowledge - An Optimization Approach. *Automatica*, vol. 32, 337-356.
54. Johansen, T.A. (1997a). On Tikhonov regularization, bias and variance in nonlinear system identification. *Automatica*, vol. 33, 441-446.
55. Johansen, T.A. (1996). Stability, Robustness, and Performance of Fuzzy Model Based Control. IEEE Conf. Decision and Control, Kobe, December 1996.
56. Johansen, T.A.; Foss B.A. (1997). Operating Regime Based Process Modeling and Identification. *Computers and Chemical Engineering*, vol. 21, 159-176.
57. Johnson, R.C. (1990). Fuzzy-Neural Hybrid Born. *Electr. Eng. Times*, Aug. 27, 29-33.
58. Jones, L.K. (1992). A simple lemma on greedy approximation in Hilbert space and convergence rates for Projection Pursuit Regression and neural network training. *Annals of Statistics*, vol. 20, no. 1, 608-613.
59. Joslyn, C. (1994). Possibilistic Processes for Complex Systems Modeling. UMI Dissertation Services, Ann Arbor MI: PhD dissertation, SUNY-Binghampton.
60. Juditsky, A.; Hjalmarsson, H.; Benveniste, A., Delyon, B.; Ljung, L.; Sjöberg, J.; Zhang, Q. (1995). Nonlinear Black-Box Models in System Identification: Mathematical Foundations. Linköping University, Sweden.
61. Kanal, L.N.; Lemmer, J.F. (1986). Uncertainty in Artificial Intelligence. Elsevier, New York.
62. Kartalopoulos, S. V. (1996). Understanding neural networks and fuzzy logic: basic concepts and applications. IEEE Press, Piscataway, New York.
63. Kaufman, L.; Rousseeuw, P.J. (1990). Finding Groups in Data. An introduction to Cluster Analysis. Wiley, New York.
64. Kavli, T. (1994). ASMOD - an Algorithm for Adaptive Spline Modelling of Observation Data. In Harris, Taylor and Francis (eds*): Advances in Intelligent Control*. London.
65. Kawarada, H.; Suito, H. (1996). Fuzzy Optimization Method. Institute of Computational Fluid Dynamics, Chiba University, Japan.
66. Kirkpatrick; Wheeler (1992). Physic. A World View, Saunders.
67. Klir, G. (1988). Fuzzy sets, uncertainty, and information. Prentice Hall, Englewood Cliffs, New York.
68. Kohonen T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, vol. 43, 59-69.
69. Kohonen T. (1990). Self-organizing map. *Proc. IEEE*, vol. 78, no. 9, 1464- 1480.
70. Koivisto. (1995). A practical approach to model based neural network control. Dissertation, Tampere University of Technology.
71 Koivo H. (2002). Soft Computing in Dynamical Systems. Tampere University of Technology.
72. Kong, S.-G.; Kosko, B. (1992). Adaptive fuzzy systems for backing up a truckand- trailer. *IEEE Trans. Neural Networks*, vol. 3, no. 2, 211-223.

73. Kosko, B. (1990). Fuzziness vs. Probability. *Int. Journal of General Systems*, vol. 17:2-3, 211-240.
74. Kosko, B. (1992). Fuzzy Systems as Universal Approximators. *Proc. of the First IEEE Conference on Fuzzy Systems*, San Diego, March, 1153-1162.
75. Kosko, B. (1997). Fuzzy Engineering. Prentice-Hall, Upper Saddle River, New Jersey.
76. Kosko, B. (1993). Fuzzy thinking : the new science of fuzzy logic. Art House.
77. Kosko, B. (1995). Fuzzy thinking : the new science of fuzzy logic. London, Flamingo.
78. Kosko, B. (1992). Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence. Englewood Cliffs (N.J.): Prentice-Hall International.
79. Kwon, T.M.; Zervakis, M.E. (1994). A self-organizing KNN fuzzy controller and its neural network structure. *Int. Journal of Adapt. Cont. and Signal Processing*, vol. 8, 407-431.
80. Larsen, J.; Hansen, L.K. (1994). Generalization performance of regularized neural network models. *Proc. IEEE Workshop on Neural Networks for Signal Processing*, Ermioni, Greece.
81. Lee, S.C.; Lee, E.T. (1975). Fuzzy neural networks. *Mathematical Biosciences*, vol. 23, 151-177.
82. Lin, C.-T.; Lee, C.S.G. (1991). Neural-network-based fuzzy logic control and decision system. *IEEE Trans. Computers*, vol. 40, n0. 12, 1320-1336.
83. Lippmann, R.P. (1993). Neural Networks, Bayesian a posteriori Probabilities, and Pettern Classification. In Cherkassky, V.; Friedman, J.H.; Wechsler, H. (eds*): From Statistics to Neural Networks*. Springer-Verlag.
84. Lloyd, S.P. (1982). Least squares quantization in PCM. *IEEE Trans. Information Theory*, vol. 28, no. 2, 129-137.
85. Mabuchi, S. (1992). An interpretation of membership functions and the properties of general probabilistic operators as fuzzy set operators. 1. case of type-1 fuzzy-sets. *Fuzzy Sets and Systems*, vol. 49, no. 3, 271-283.
86. MacKay, D. (1992) Bayesian interpolation. *Neural Computation*, vol. 4, no. 3. 415-447.
87. Mc Culloch, W.S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, 115-133.
88. Mills, D.J.; Harris, C.J. (1995). Neurofuzzy modelling and control of a six degree of freedom AUV. Technical report, University of Southampton,
Department of Electronics and Computer Science, 1995/6 Research Journal. 86. Minsky, M. L. and Papert (1969). Perceptrons. Cambridge, MA: MIT Press.
89. Moody, J.; Darken, C.J. (1989). Fast learning in networks of lacally-tuned processing units. *Neural Computation*, vol. 1, no. 2, 281-294.
90. Mouzouris and Mendel. (1996). Designing fuzzy logic systems for uncertain environments. In *Fuzz-IEEE '96*, New Orleans, vol.1, 295-301.
91. Nadaraya, É. A. (1964). On estimating regression. *Theory of Probability and its Applications*. vol. 9, no. 1, 141-142.
92. Narendra, K.S.; Mukhopadhyay, S. (1997). Adaptive Control Using Neural Networks and Approximate Models. *IEEE Trans. Neural Networks*, vol. 8, no. 3, 475-485.
93. Narendra, K.S.; Parthasarathy, K. (1990). Identification and Control of Dynamic Systems Using Neural Networks. *IEEE Trans. Neural Networks*, vol. 1, no. 1, 4- 27.
94. Nauck, D. (1994). A Fuzzy Perceptron as a Generic Model for Neuro-Fuzzy Approaches. In *Proc. Fuzzy Systeme'94*, Munich.
95. Nie, J.; Linkens, D.A. (1993). Learning control using fuzzified self-organizing radial basis function network. *IEEE Trans. Fuzzy Systems*, vol. 1, no. 4, 280- 287.
96. Orr, M.J.L. (1995). Regularisation in the selection of radial basis function centres. *Neural Computation*, vol. 7, no. 3, 606-623.

95. Orr, M.J.L. (1997). Matlab Routines for Subset Selection and Ridge Regression in Linear Neural Networks. Edinburgh University. Available in World Wide Web: http://www.cns.ed.ac.uk/people/mark.html.
97. Pao, Y.H.; Phillips, S.M.; Sobajic, D.J. (1994). Neural-net Computing and Intelligent Control of Systems. In C.J. Harris, Taylor and Francis (eds.): *Advances in Intelligent Control*. London.
98. Park, J.; Sandberg I.W. (1991). Universal approximation using radial-basisfunction networks. *Neural Computation*, vol. 3, 246-257.
99. Petrushev, P.; Popov, V. (1987). Rational Approximation of Real Functions. Cambridge University Press, Cambridge.
100. Pinkus, Allan. (1985). n-Widths in Approximation Theory. Springer-Verlag.
101. Poggio, T.; Girosi, F. (1990). Networks for Approximation and Learning. *Proc. IEEE*, vol. 78, no. 9, 1481-1497.
102. Powell, M.J.D. (1981). Approximation theory and methods. Cambridge University Press.
103. Powell, M.J.D. (1987). Radial Basis Functions. In J.C.Mason, M.G.Cox (eds.): *Algorithms for Approximation*. Oxford University Press, New York.
104. Press, W.H,; Teukolsky; Vetterling; Flannery. (1992). Numerical Recipes in C: The Art of Scientific Computing (2nd edition). Cambridge University Press.
105. Pucar, P.; Sjöberg, J. (1995). On the parameterization of hinging hyperplane models. Technical report, Dep. of Electrical Engineering, Linköping University, Sweden.
106. Raitamäki, J.; Viljamaa, P.; Koivo, H.; Neittaanmäki, P. (1996). Basis Functions in Soft Computing and in Finite Element Method. EUFIT'96 - Fourth European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, September 2-5, *Proceedings*, vol 1, 115-119.
107. Rasband, S. N. (1990). Chaotic Dynamics of Nonlinear Systems. John Wiley & Sons, New York.
108. Ripley, B. (1996). Pattern Recognition and Neural Networks. Cambridge University Press.
109. Roberts, J.M.; Mills, D.J.; Charnley, D.; Harris, C.J. (1995). Improved Kalman filter initialisation using neurofuzzy estimation. In *4th Int. Conf. on Artificial Neural Networks*, Cambridge, U.K. , 329-334.
110. Rosenblatt, F. (1962). Principles of Neurodynamics: Perceptrons and the theory of Brain Mechanism. Washington DC: Spartan.
111. Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol.1: Foundations, Cambridge, MA: MIT Press, 318-362.
112. Saaty, T.L. (1974). Measuring the fuzziness of sets. *Journal of Cybernetics*, vol. 4, 53-61.
113. Sjöberg, Zhang, Ljung, Benveniste, Deloyn, Glorennec, Hjalmarsson,Juditsky. (1995). Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, vol. 31, no. 12, December, 1691-1724.
114. Sugeno, M.; Murakami M. (1984). Fuzzy parking control of model car. In *Proceedings IEEE 23rd Conference on Decision and Control*, Las Vegas, USA.
115. Suykens, J.A.K.; De Moor, B.L.R.; Vandewalle, J. (1994). Static and dynamic stabilizing neural controllers, applicable to transition between equilibrium points. *Neural Networks*, vol. 7, 819-831.
116. Södeström, T.; Stoica, P. (1988). System Identification. Prentice Hall, Englewood Cliffs, NJ.
117. Takagi, T.; Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Trans. Systems, Man, and Cybernetics*, 15, 116-132.
118. Thomas, S.F. (1995). Fuzziness and Probability. ACG Press, Wichita, USA.
119. Thompson, M.L.; Kramer, M.A. (1994). Modeling chemical processes using prior knowledge and neural networks. *AIChe*, vol. 40, 1328-1340.

120. Tikhonov, A.N.; Arsenin, V.Y. (1977). Solutions of Ill-Posed Problems. Washington, DC, Winston.
121. Tsao, E.C.-K.; Bezdek, J.C.; Pal, N.R. (1994). Fuzzy Kohonen clustering networks. *Pattern recognition*, vol.27, no.5,757-764.
122. Verhulst, F. (1990). Nonlinear Differential Equations and Dynamical Systems. Springer-Verlag, Berlin, Heidelberg.
123. Viljamaa, P. (1996). Fuzzy-säätäjän viritysmenetelmiä. Report 1. Tampere University of Technology, Control engineering laboratory, Tampere.
124. Viljamaa, P.; Raitamäki, J.; Neittaanmäki, P.; Koivo, H. (1996). Basis Functions in Soft Computing. WAC '96 - Second World Automation Congress, *Proceedings*, Montpellier, France.
125. Vuorimaa, P. (1994). Fuzzy self-organizing map, *Fuzzy Sets and Systems*, vol. 66, 223-231.
126. Wahba, G. (1990). Spline Models for Observational Data. SIAM, Philadelphia.
127. Wang, L.-X. (1994). Adaptive Fuzzy systems and control Design and stability analysis. Prentice Hall, New Jersey.
128. Wang, L.-X. (1992). Fuzzy systems are universal approximators. *IEEE Trans. Syst., Man, Cybern*, vol. SMC-7, no. 10, 1163-1170.
129. Webb, A.R. (1994). Functional approximation by feed-forward networks: a least squares approach to generalisation. *IEEE Trans. Neural Networks*, vol. 5., no. 3, 363-371.
130. Werntges, H.W. (1993). Partitions of Unity Improve Neural Function Approximation, *Proc. IEEE Int. Conf. Neural Networks*, San Francisco, vol. 2, 914-918.
131. White, D.; Bowers, A.; Iliff, K.; Noffz, G.; Gonda, M.; Menousek, J. (1992). Flight, Propulsion and Thermal Control of Advanced Aircraft and Hypersonic Vehicles. In White, D.; Sofge, D.A. (eds.): *Handbook of Intelligent Control*. Van Nostrand Reinhold, NY.
132. Zadeh, Lotfi A. (1992). Fuzzy logic: advanced concepts and structures. IEEE, Piscataway, New York.
133. Zadeh, Lotfi A. (1965). Fuzzy Sets. *Information and Control*, June 1965, 338- 354.
134. Zadeh, Lotfi. A. (1994). Soft Computing and Fuzzy Logic. *IEEE Software*, vol. 11, no. 6, 48-56.
135. Zimmermann, H. J. (1993). Fuzzy sets, decision making, and expert systems. Kluwer, Boston.
136. Zysno, P. (1981). Modeling membership functions. In Rieger (ed.*): Empirical Semantics I, Vol. 1 of Quantitative Semantics*, vol. 12, Studienverlag Brockmeyer, Bochum, 350-375.