TASK QUARTERLY 7 No 1 (2003), 23–41

BETA NEURO-FUZZY SYSTEMS

ADEL M. ALIMI

Research Group on Intelligent Machines, University of Sfax,
National School of Engineers ENIS, Department of Electrical Engineering,
B.P. W–3038 Sfax, Tunisia
Adel.Alimi@ieee.org

(Received 23 January 2002; revised manuscript received 3 September 2002)

Abstract: In this paper we present the Beta function and its main properties. A key feature of the Beta function, which is given by the central-limit theorem, is also shown. We then introduce a new category of neural networks based on a new kernel: the Beta function. Next, we investigate the use of Beta fuzzy basis functions for the design of fuzzy logic systems. The functional equivalence between Beta-based function neural networks and Beta fuzzy logic systems is then shown with the introduction of Beta neuro-fuzzy systems. By using the Stone-Weierstrass theorem and expanding the output of the Beta neuro-fuzzy system into a series of Beta fuzzy-based functions, we prove that one can uniformly approximate any real continuous function on a compact set to any arbitrary accuracy. Finally, a learning algorithm of the Beta neuro-fuzzy system is described and illustrated with numerical examples.

Keywords: beta function, kernel based neural networks, Sugeno fuzzy model, neuro-fuzzy systems, universal approximation property, learning algorithms, incremental learning

1. Introduction

There are two principal approaches to solve the problem of construction of a suitable set of fuzzy rules: manual generation or automatic generation.

In the manual approach, several parameters of the fuzzy inference engine must be built by a fuzzy logic expert and is often based on heavy experimental and heuristic methods [1]. The development of these methods takes an excessive time to build and to adjust the fuzzy rules and it becomes even more difficult when the number of rules increases. This motivated the researchers to automate the process of extraction of fuzzy rules.

The basic idea of the automatic approach is to estimate the fuzzy rules by training starting from couples of input/output data. Fuzzy systems using the capacity of training of neural networks are currently used to successfully build input/output transformations in several applications [1, 2].

The two principal advantages of neuro-fuzzy systems are: first, their capacity to identify fuzzy rules and to adjust automatically and simultaneously membership functions and, second, the fact that the parameters of these systems of-

ten make physical sense, which is not always the case in the traditional neural networks.

In this paper, we introduce a new category of neuro-fuzzy systems based on a new kernel: the Beta function. Four theorems dealing with the main properties of Beta neuro-fuzzy system are presented (their proofs are given in appendixes). Main properties of this function are shown in Section 2. Section 3 is devoted to Beta-based function neural networks (BBFNN). In Section 4 we present the Beta fuzzy logic system (BFLS). The functional equivalence between Beta based function neural networks and Beta fuzzy logic systems is then shown in Section 5 with the introduction of Beta neuro-fuzzy systems. Section 6 deals with the universal approximation property of the BNFS. Finally, a learning algorithm of the BNFS is described in Section 7 illustrated with numerical examples.

2. Properties of the Beta function

2.1. Definition 1: One-dimensional Beta function

In the one-dimensional case, the Beta function [3] is defined by [4, 5]:

$$\beta(x;p,q,x_0,x_1)=\begin{cases}\left(\dfrac{x-x_0}{x_c-x_0}\right)^p\left(\dfrac{x_1-x}{x_1-x_c}\right)^q & \text{if } x\in]x_0,x_1[\\ 0 & \text{elsewhere}\end{cases} \tag{1}$$

with $p,q,x_0<x_1\in\mathbb{R}$, and

$$x_c=\frac{px_1+qx_0}{p+q}. \tag{2}$$

Figure 1 shows the different shapes that can be generated by the Beta function. One can note that this function is able to approximate many usual shapes such as triangular, trapezoidal or Gaussian shapes [6].

2.2. Basic properties of a one-dimensional Beta function

The Beta function may be characterized by the following properties:

$$\beta(x_0)=\beta(x_1)=0, \tag{3}$$

$$\beta(x_c)=1, \tag{4}$$

$$\frac{d\beta(x)}{dx}=\left[\frac{px_1+qx_0-(p+q)x)}{(x-x_0)(x_1-x)}\right]\beta(x), \tag{5}$$

$$\frac{d\beta(x_c)}{dx}=\frac{d\beta(x_0)}{dx}=\frac{d\beta(x_1)}{dx}=0, \tag{6}$$

$$\frac{p}{q}=\frac{x_c-x_0}{x_1-x_c}. \tag{7}$$

It should be noted here that the Beta function may be considered as a linear function of x if one takes $p=1$, $q=0$ or $p=0$, $q=1$.

2.3. Definition 2: A multi-dimensional Beta function

The Beta function in the multi-dimensional case is defined by [4]:

$$\beta(\underline{x})=\prod_{k=1}^{m}\beta(x_k,p_k,q_k,x_{0,k},x_{1,k}). \tag{8}$$

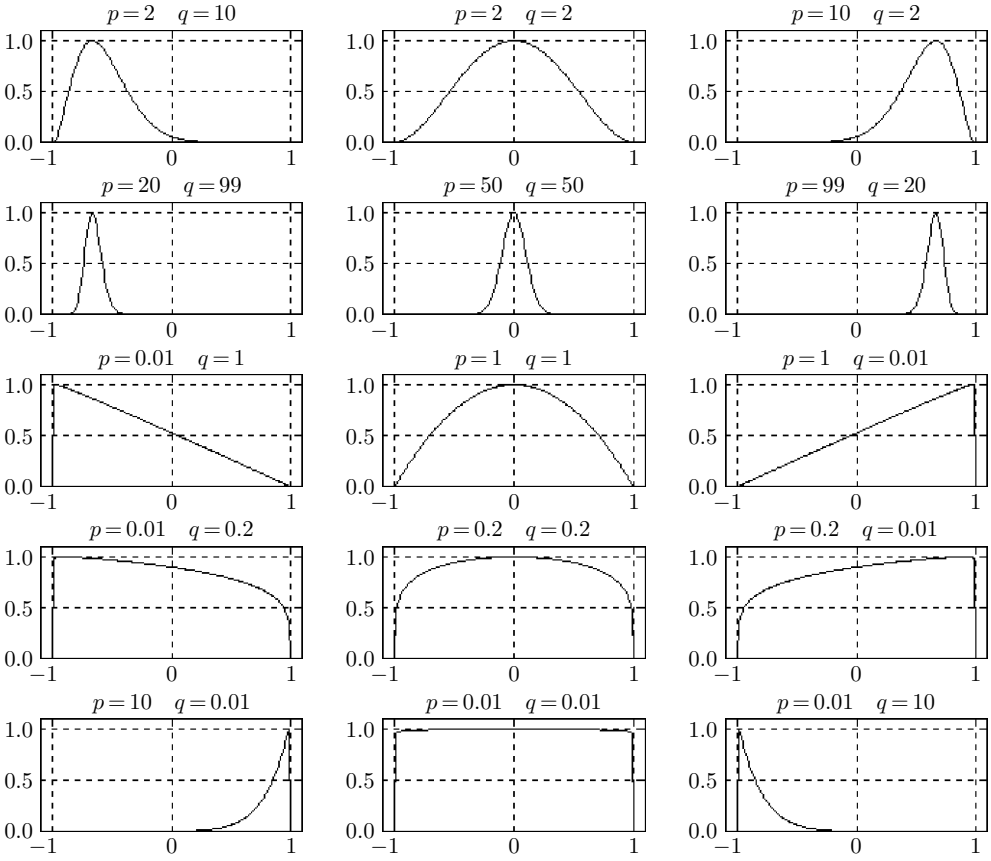

Figure 1. Different forms of the Beta function in the one-dimensional case ($x_0 = -1;\ x_1 = 1$)

It should be noted here that the multi-dimensional Beta function given by Equation (8) is the product of m one-dimensional Beta functions expressed by Equation (1). Below, we will use the same appellation (Beta function) when speaking about one-dimensional or multidimensional Beta functions.

2.4. Definition and properties of the Gaussian function

The Gaussian function is defined by:

$$\text{Gauss}(x;\,\mu,\sigma) = \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \tag{9}$$

$$\text{Gauss}(\mu) = 1, \tag{10}$$

$$\frac{d\,\text{Gauss}(x)}{dx} = -\frac{(x-\mu)}{\sigma^2}\,\text{Gauss}(x), \tag{11}$$

$$\text{Gauss}(\mu+n\sigma) = \text{Gauss}(\mu-n\sigma) = \exp^{-n^2/2} = \varepsilon, \tag{12}$$

$$\varepsilon \to 0 \ \text{ if } n \to 0. \tag{13}$$

2.5. Theorem 1: Approximation of the Gaussian function by a Beta function

For any given Gaussian function $\text{Gauss}(x;\,\mu,\sigma)$ and for any given precision ε, there exists a Beta function $\beta(x;\,p,q,x_0,x_1)$ that approximates the Gaussian function

Table 1. Parameters of the Beta functions illustrated in Figure 2

n	p	x_0	x_1	error(%)
3	4.5	−3	3	6.9871
4	8.0	−4	4	3.9726
5	12.5	−5	5	2.5505
6	18.0	−6	6	1.7738
7	24.5	−7	7	1.3043
8	32.0	−8	8	0.9997
100	5000.0	−100	100	0.0075
200	20000.0	−200	200	$7.5866 \cdot 10^{-5}$
1000	500000.0	−1000	1000	$7.673 \cdot 10^{-87}$

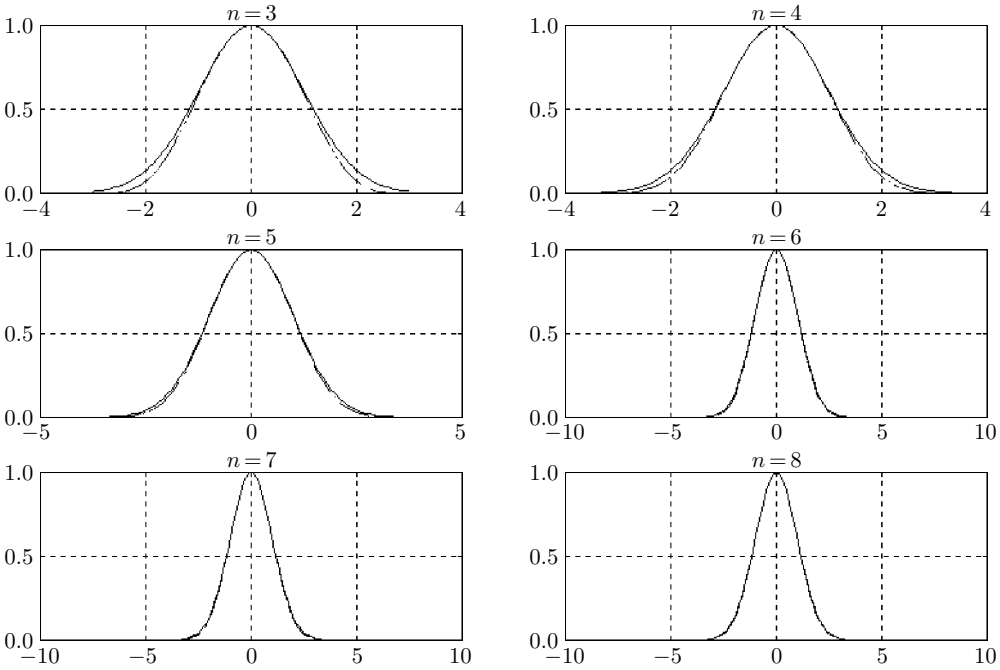

Figure 2. Approximation of the Gaussian function Gauss$(x; 0,1)$ with the Beta function

with an error of less than ε: $|\beta(x; p,q,x_0,x_1) - \text{Gauss}(x; \mu,\sigma)| \leq \varepsilon$ for any $x \in \mathbb{R}$. This Beta approximation is characterized by choosing a sufficiently great n and taking: $p = n^2/2$, $q = n^2/2$, $x_0 = \mu - n\sigma$ and $x_1 = \mu + n\sigma$ [6]. □

Figure 2 illustrates some examples of such approximation. As can be seen in this figure, for $n = 3, 4, 5$, a residual error is still apparent on the edges of the Gaussian function. However, for $n \geq 6$, it is not possible visually to distinguish the Gaussian function from its Beta approximation.

It should be noted that the reverse is not true, since the Beta function can have forms richer than the Gaussian function (asymmetry, linearity, *etc.*)

Moreover, the larger n, the better the approximation of the Gaussian function by the Beta function. Indeed, n represents to some extent the distance from the center of the Gaussian function at which this function is regarded as null. Table 1 illustrates the values of n, p, x_0, x_1 for the examples in Figure 2 and their error. Error is given

in % and represents the relative error between the Gaussian function and its Beta approximation.

2.6. The central limit theorem

The central-limit theorem [7, 8] is a fundamental theorem which is extensively treated in mathematical literature. Its formulation is based on probabilistic concepts. In a reformulation of the central limit theorem, Papoulis [9] showed that:

Papoulis theorem

Let's consider a set of n functions f_i that are zero outside a finite interval (a_i,b_i); then, under certain general conditions, their product tends to a Beta function as n tends to infinity. □

This theorem provides an important theoretical support in favor of the use of Beta functions. An illustration of the central-limit theorem is given in Figure 3, where it can be seen that only for $n=3$, the obtained functions product may be easily approximated by a Beta function.

3. Beta basis functions neural networks (BBFNN)

An artificial neural network [10] may be represented as a set of interconnected neurons; each neuron performs a given function and each connection specifies the direction of the passage of the signal from one neuron to another. The behavior of a neural network is governed by a certain number of adjustable parameters, which are distributed among the neurons. Each neuron has its own transfer function.

Among the existing neural networks, the most used in literature are radial basis function neural networks (RBFNN) [11]. In the remainder of this section, we will consider, for reasons of simplicity and clarity, multi-input single-output (MISO) networks.

3.1. Definition of a radial basis function neural network (RBFNN)

A radial basis function neural network is a three layered neural network such that: the first layer contains neurons corresponding to the input vector, the second layer includes kernel neurons whose output is a Gaussian function, and the last layer includes only one neuron which calculates the final output of the network given by [11]:

$$y = f(\underline{x}) = \sum_{j=1}^{N} w_j \,\mathrm{Gauss}_j(\underline{x}), \tag{14}$$

where

- $\underline{x} = (x_1, x_2, \ldots, x_n)^T \in \mathscr{I} \subset \mathbb{R}^n$ is the input variable;
- $y \in \mathscr{O} \subset \mathbb{R}$ is the output variable;
- w_j are the weights of the connections between the N hidden kernels and the output neuron.

3.2. Definition 3: Beta basis function neural network (BBFNN)

A Beta basis function neural network is a three layered neural network such that: the first layer contains neurons corresponding to the input vector, the second

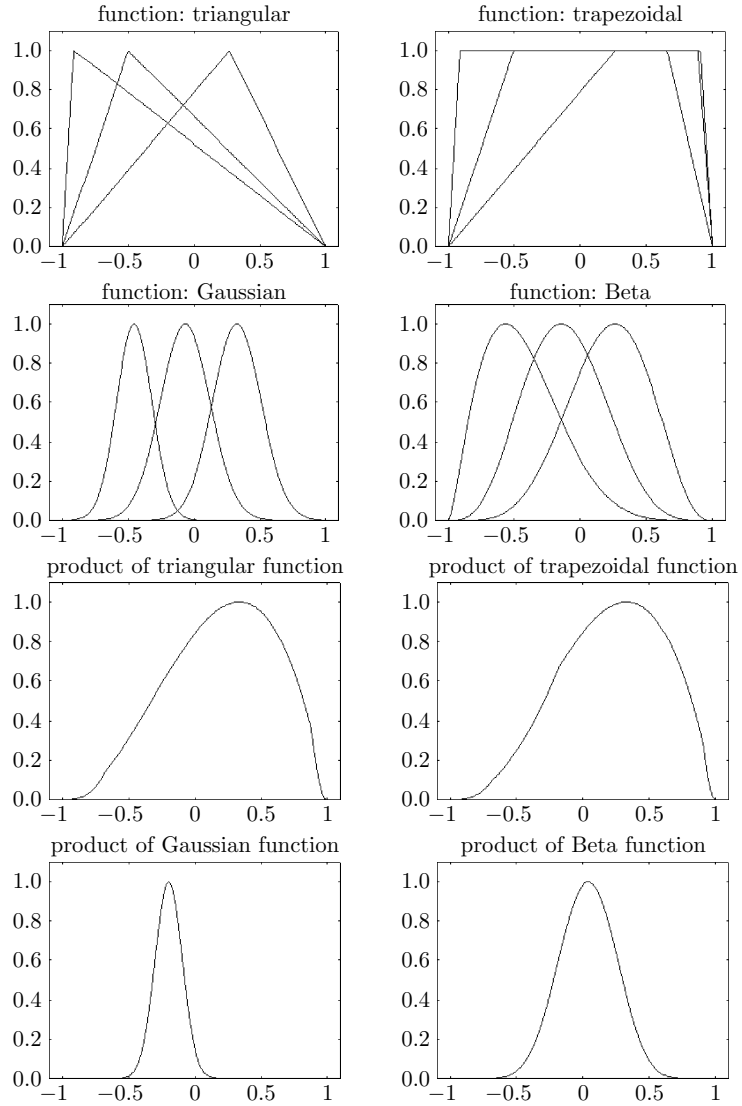

Figure 3. Illustration of the central limit theorem with $n=3$. The first four sub-figures represent three randomly generated functions each: triangular, trapezoidal, Gaussian, and Beta. The product of these three functions is presented in the last four sub-figures. The forms of the obtained product functions can simply be reproduced by Beta functions

layer includes kernel neurons whose output is a Beta function, and the last layer includes only one neuron which calculates the final output of the network given by:

$$y = f(\underline{x}) = \sum_{j=1}^{R} f_j(\underline{x})\,\beta_j(\underline{x}), \tag{15}$$

where

- $\underline{x} = (x_1, x_2, \ldots, x_n)^T \in \mathscr{I} \subset \mathbb{R}^n$ is the input variable;
- $y \in \mathscr{O} \subset \mathbb{R}$ is the output variable;

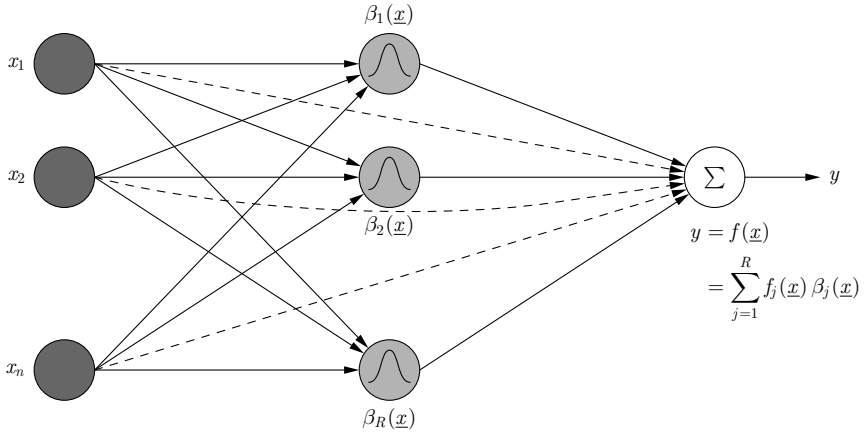

Figure 4. The architecture of a Beta-based functions neural network (BBFNN)

- f_j are the weights of the connections between the R hidden kernels and the output neuron. One of the differences with the standard RBFNN is that, one can have weights that depend on input vector $\underline{x}$.

The architecture of a BBFNN is illustrated in Figure 4.

3.3. Theorem 2: Approximation of a RBFNN by a BBFNN

Any radial basis function neural network (RBFNN) can be approximated to any precision by a Beta basis function neural network (BBFNN) with the same number of hidden kernels. □

It is important to note that the inverse is not true, since the Beta kernels of a BBFNN can generate more rich shapes than the Gaussian kernels of a RBFNN (asymmetry, linearity, *etc.*)

4. The Beta fuzzy logic system (BFLS)

4.1. The architecture of a fuzzy logic system (FLS)

Thirty years after the introduction of the state space theory by Kalman in 1960 [12], which initiated the modernization of the analysis methods of traditional systems, the fuzzy logic theory of Zadeh [13] seems to open new horizons in the vast field of data analysis and processing. The fuzzy approach is marked to a certain extent by a revolution in methodology while moving away from the heavy approaches based on the development of precise mathematical models and towards modeling and the concrete reproduction of the human processes of thinking and decision-making.

The fuzzy set theory is a theory of inaccuracy and uncertainty; it makes it possible to employ ill-defined concepts in ill-defined situations. It is a generalization of the conventional binary logic in the sense that, instead of using ambiguous membership functions such as 0 or 1, any degree of membership between 0.0 and 1.0 are allowed. This makes it possible to describe mathematically vague, ambiguous and qualitative information in terms of membership functions. A fuzzy logic system (FLS), as represented in Figure 5, characterizes a black box whose output is obtained by a reasoning elaborated from the observation of the state of the system and a list of rules describing how the system must evolve.

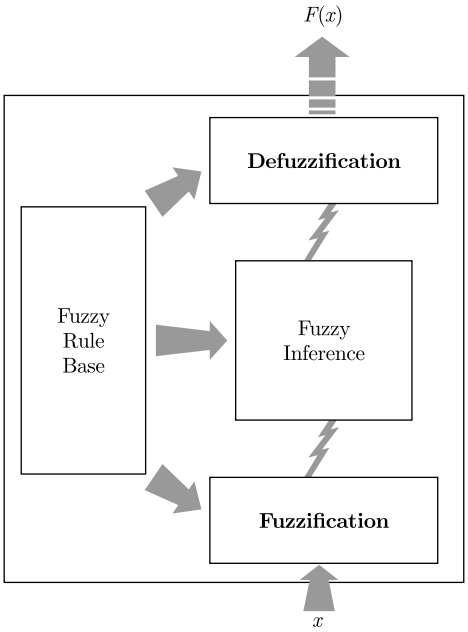

Figure 5. Architecture of a general fuzzy logic system (FLS)

The fuzzification (or fuzzy quantification) consists in changing the type of data representation: from a real number towards a fuzzy set of linguistic variables defined in a universe of discourse constituted by several qualitative terms by using coding generally expressed in linguistic terms.

The stage of inference uses two sets of data: observations of the state of the system and a set of rules describing the system functionality, in order to infer the system's output depending on its current state.

Defuzzification relates to the production of a non-fuzzy value of the output, which represents as accurately as possible the system's output inferred by the fuzzy inference engine.

Fuzzy logic becomes even more tempting when one notes the capacity of humans to understand complex interrelationships and to react in consequence while observing, reasoning and learning, and this with incomplete and partly inaccurate information. It should be noted that in spite of the popularity of fuzzy logic, it is surprising to notice that a fuzzy system is nothing more than a static, nonlinear and multidimensional transformation of the input space towards the output space.

The principal objective of a fuzzy system is to model the human process of decision-making by using the concepts of fuzzy logic and approximate reasoning.

A multi-input multi-output fuzzy system (MIMO) can be regarded as being a function $f : \mathscr{T} \subset \mathbb{R}^n \to \mathscr{O} \subset \mathbb{R}^{n'}$, where $\mathscr{T}$ is the input space and $\mathscr{O}$ is the output space. As shown by Zeng and Singh [14] and by Lee [15], a MIMO fuzzy system can always be separated into groups of multi-input simple-output (MISO) fuzzy systems. Thus, all MIMO versions of the results in this paper can be easily obtained by simple algebraic operations.

4.2. Definition of a MISO Sugeno fuzzy system

According to the Sugeno model [16, 17], the output of a MISO fuzzy system may be expressed by:

$$y = f(\underline{x}) = \left\{ \sum_{j=1}^{R} f_j(\underline{x}) \prod_{i=1}^{n} \mu_{X_i^j}(x_i) \right\} / \left\{ \sum_{k=1}^{R} \prod_{l=1}^{n} \mu_{X_l^k}(x_l) \right\} \tag{16}$$

$$= \sum_{j=1}^{R} f_j(\underline{x}) w_j(\underline{x}), \tag{17}$$

where

- $\underline{x} = (x_1, x_2, \ldots, x_n)^T \in \mathscr{I}$ is the input variable: $x_{i,\min} \le x_i \le x_{i,\max}$,
- $y \in \mathscr{O}$ is the output variable,
- R is the number of fuzzy rules in the form $\mathscr{R}_j$: IF ($\underline{x}$ is $\underline{X}^j$) THEN (y is $f_j(\underline{x})$),
- $\underline{X}^j = (X_1^j, X_2^j, \ldots, X_n^j)^T$ are linguistic terms characterized by fuzzy membership functions $\mu_{X_i^j}(x_i)$ (these functions are in general triangular, trapezoidal, or Gaussian functions),
- f_j are functions: $\mathscr{I} \subset \mathbb{R}^n \to \mathscr{O} \subset \mathbb{R}$ (these functions are usually polynomials in input variables x_i, but can be any functions that describe the output of the system within the fuzzy region specified by the antecedent of the fuzzy rule). When f_j is a constant, the fuzzy system is known as a Sugeno FLS of order 0,
- w_j are R fuzzy basis functions (FBF), representing the firing intensity of each fuzzy rule given by:

$$w_j(\underline{x}) = \prod_{i=1}^{n} \left\{ \mu_{X_i^j}(x_i) / \left[\sum_{k=1}^{R} \prod_{l=1}^{n} \mu_{X_l^k}(x_l) \right]^{1/n} \right\}. \tag{18}$$

Because of the great flexibility and interesting approximation properties of the Beta function, we proposed [4, 5] the use of one-dimensional Beta functions as membership functions and the use of multi-dimensional Beta functions as fuzzy basis functions. This leads to the definition of a Beta fuzzy logic system.

4.3. Definition 4: Beta fuzzy logic system (BFLS)

A Beta fuzzy logic system (BFLS) is a Sugeno fuzzy system for which the fuzzy basis functions are Beta functions. The output of a BFLS is given by [4, 5]:

$$f(\underline{x}) = \sum_{j=1}^{R} f_j(\underline{x}) \beta_j(\underline{x}). \tag{19}$$

This definition is based on the central limit theorem. Indeed, we have proposed [4, 5] to approximate the fuzzy basic functions (FBFs) given by Equation (18) by Beta functions (since FBFs are products of n functions) and that this product tends, according to the central limit theorem, towards a Beta function when n tends towards infinity. It should be noticed that the use of Beta functions as FBFs does not require normalization, which is the case in traditional FBFs given by Equation (18). Thus, one may characterize the intensity of activation of each rule $\mathscr{R}_j$: by a Beta function [4, 5]:

$$w_j(\underline{x}) = \beta_j(\underline{x}). \tag{20}$$

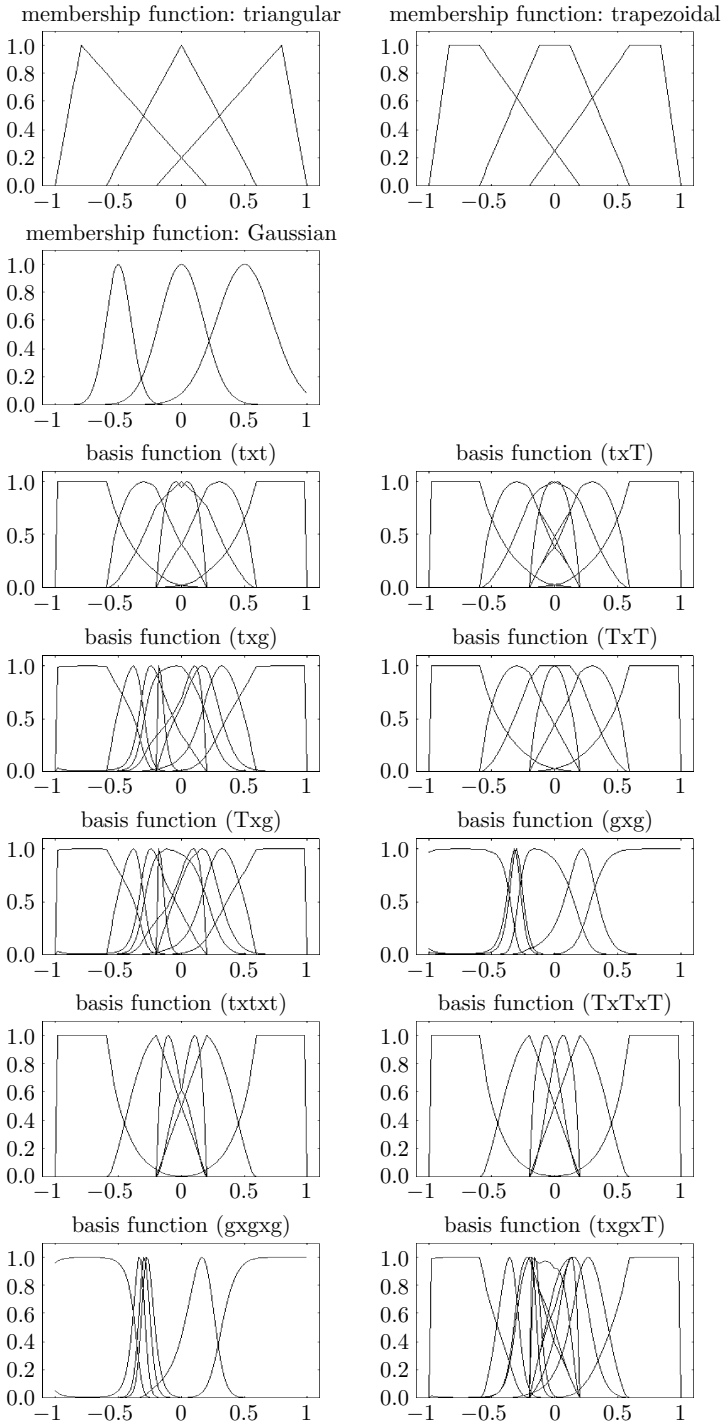

Figure 6. Illustration of approximation by Beta functions (Equation (8)) of some FBFs (Equation (18)) for different membership functions used in a fuzzy system. The first three sub-figures represent three membership functions each: triangular, trapezoidal and Gaussian. The remaining sub-figures show FBFs of the different FLS using different combinations of these membership functions

An illustration of approximation by Beta functions (Equation (8)) of some fuzzy basis functions (Equation (18)) for various membership functions used in a fuzzy system is given in Figure 6.

5. Beta neuro-fuzzy system (BNFS)

Radial basis function neural networks (RBFNN) were shown [18] to be functionally equivalent to Gaussian FBF based FLS without normalization. An extension of this functional equivalence to generalized RBFNN, when using normalized FBF as in Equation (18), has also been made [19].

5.1. Theorem 3: Functional equivalence between BBFNN and BFLS

Beta basis function neural networks are functionally equivalent to Beta fuzzy logic systems [4, 5]. □

This theorem allows the simple and effective design of neuro-fuzzy models of complex systems starting from the knowledge of couples of input/output data as it will be presented in the next section.

5.2. Definition 5: Beta neuro-fuzzy system (BNFS)

A Beta neuro-fuzzy system (BNFS) is a system whose output is described by Equation (19) and who, while being based on the preceding theorem, can be considered at the same time as a Beta fuzzy logic system (BFLS) or as a Beta-based function neural network (BBFNN).

With this definition, we are able to benefit at the same time from the expressional richness of fuzzy systems and the training flexibility of neural networks.

6. Universal approximation properties of Beta neuro-fuzzy systems

It is with the universal approximation theorems that fuzzy systems and neural networks gained the recognition of the scientific community as useful tools with robust mathematical bases.

6.1. Stone-Weierstrass theorem

Let X be a non-empty compact metric space, $C[X]$ – a set of continuous functions defined on X, and A – a subalgebra of $C[X]$ with the following two properties:

1. A contains the function (1),
2. A separates points in X, *i.e.* for any two distinct points $a,b \in X$, there is a function $f \in A$ such that $f(a) \neq f(b)$.

Then A is a dense subalgebra of $C[X]$ in the topology induced by the uniform metric [20, 21]. □

Based on the theorem of Stone-Weierstrass [20, 21], Wang and Mendel [22] have shown that, in the case of Gaussian membership functions, the set of all FBF expansions given by Equation (17) can approximate, to any given accuracy, any real continuous function.

6.2. Wang and Mendel theorem

In the case of Gaussian membership functions, the set of all expansions of FBF given by Equation (17) can approximate, with any precision, any real function continuous [22]. □

In the preceding section, based on the central-limit theorem, we have shown that Beta functions given by Equation (8) used as Beta FBF can approximate any FBF given by Equation (18) for any membership function used in the FLS. Using the Stone-Weierstrass theorem [20, 21], we have proven a theorem concerning Beta neuro-fuzzy systems [4–6] equivalent to that of Wang and Mendel [22].

6.3. Theorem 4: A Beta neuro-fuzzy system is a universal approximator

For any given real continuous function g on the compact $\mathscr{T} \subset \mathbb{R}^n$ and any arbitrary $\varepsilon > 0$, there exists a Beta neuro-fuzzy system f such that:

$$\sup_{\underline{x} \in \mathscr{T}} |g(\underline{x}) - f(\underline{x})| < \varepsilon. \tag{21}$$

In other words, a Beta neuro-fuzzy system (BNFS), by its two components BFLS or BBFNN, can approximate any continuous real function on a compact to any precision [4, 5]. □

We have recently proven the same theorem with an entirely different approach based on pseudo-trapezoidal functions, which Beta functions are a particular case of [23, 24].

7. A learning algorithm for a Beta neuro-fuzzy system

To carry out the training of a Beta neuro-fuzzy system, we developed many learning algorithms based on genetic algorithms [25, 26], iterative approaches [27], or incremental gradient approaches [28, 29]. In this paper we will describe a learning algorithm based on a modified version of the hierarchically self-organizing learning algorithm [30, 31].

Our algorithm considers unknown fuzzy rules starting from a data set. The training of a Beta neuro-fuzzy system consists in determining the minimum necessary number R of rules and adjusting the parameters of the Beta functions of each hidden node as well as their weights. The algorithm starts with 0 rules and generates the suitable fuzzy rules as long as it is necessary. The fuzzy rules are generated by recruiting in an incremental way the units which are Beta functions (hidden nodes) and by adjusting parameters until a desired precision is reached. Incremental addition of nodes is based simultaneously on control and adjustment of parameters x_0, x_1, p and q.

A general procedure of the learning algorithm is as follows:

1. Initialize the number of training cycles $l = 1$, the number of hidden nodes $h = 0$, and the number of patterns presented to the Beta neuro-fuzzy system $n = 1$;
2. At the l^{th} training cycle, evaluate $|t_{nk} - y_{nk}|$ by using the n^{th} pattern to be learned (t_{nk} and y_{nk} are respectively the desired value and the current value of the k^{th} output unit);
3. If $|t_{nk} - y_{nk}| > e_m$, where e_m represents the error margin, then go to (4); if not go to (5);

4. If there is a hidden node such as the patterns to be learned x_n are in the hypersphere ($\beta(x_n) \neq 0$), then go to (5); if not create a hidden node and go to (6) (The parameters of the new hidden node are determined initially by: set $h = h+1$, set $x_{c,h} =$ input of the pattern to be learned x_n, set $x_{1,h} - x_{0,h} = r_{init}$, set $p_h = q_h = p_{init}$.);
5. Apply the training of the parameters for all the hidden nodes;
6. If all the patterns to be learned are presented, go to (7); if not, set $n = n+1$ and go to (2);
7. Set $l = l+1$;
8. If the network presents satisfactory performances ($l > l_{\max}$); if not, set $n = 1$ and go to (2).

The learning algorithm updates the parameters in an incremental way while being based on the current presented pattern to be learned. The parameters of the Beta neuro-fuzzy network $v_h = [x_{0,h}^T, x_{1,h}^T, p_h^T, q_h^T, w_h^T]^T$ (w_h being the consequence part of the fuzzy rule of a zero-order Sugeno type) are updated by applying the traditional method of Levenberg-Marquardt [32] in order to minimize the error. The error function of the n^{th} pattern to be learned is defined by:

$$E_n = \frac{1}{2}\|t_n - y_n\|^2, \tag{22}$$

with t_n and y_n being respectively the desired output and the current output of the Beta neuro-fuzzy network for the n^{th} pattern to be learned.

In order to test our learning algorithm and to analyze performance of Beta neuro-fuzzy systems, we have carried out three series of simulations.

In the first series of simulations, we considered the work of Mitaim and Kosko [33], in which they carried out a comparative study between different membership functions of fuzzy systems. They considered the following 6 functions:

$$f_1(x) = 3x(x-1)(x-1.9)(x-0.7)(x+1.8), \quad -2 \leq x \leq 2; \tag{23}$$

$$f_2(x) = 10\tan^{-1}\left[\frac{(x-0.2)(x-0.7)(x+0.8)}{x+1.4}\right], \quad -1 \leq x \leq 1; \tag{24}$$

$$f_3(x) = \frac{100(x+0.95)(x+0.6)(x+0.4)(x-0.1)(x-0.4)(x-0.8)(x-0.9)}{(x+1.7)(x-2)^2}, \quad -1 \leq x \leq 1; \tag{25}$$

$$f_4(x) = 8\sin(10x^2+5x+1), \quad -1 \leq x \leq 1; \tag{26}$$

$$f_5(x) = 10\tan^{-1}\left[\frac{(x-0.2)(x-0.7)(x+0.8)}{(x+1.4)(x-1.1)x+0.7}\right], \quad -1 \leq x \leq 1; \tag{27}$$

$$f_6(x) = 10\left[\exp(-5|x|) + \exp(-3|x-0.8|/10) + \exp(-10|x+0.6|)\right], \quad -1 \leq x \leq 1. \tag{28}$$

Then, they sought the error of approximation of these functions (200 samples) by fuzzy systems with 12 rules and different membership functions by using non-linear methods of regression to seek the minimal error. They found that the Gaussian and sinc functions ($\text{sinc}(x) = \sin(x)/x$) have the best performances.

We continued their work by introducing the Beta function and found the relative errors of approximation shown in Table 2 (where *error* is given in % and represents

Table 2. Relative approximation error of Mitaim and Kosko's functions with FLS of different membership functions

Function	BNFS	Gaussian-based FLS	Sinc-based FLS
f_1	$5.5589 \cdot 10^{-5}\%$	$1.3229 \cdot 10^{-4}\%$	$4.6308 \cdot 10^{-2}\%$
f_2	$1.7984 \cdot 10^{-4}\%$	$1.8675 \cdot 10^{-4}\%$	$8.1124 \cdot 10^{-3}\%$
f_3	$1.3350 \cdot 10^{-4}\%$	$6.6626 \cdot 10^{-4}\%$	$2.9980 \cdot 10^{-4}\%$
f_4	$5.2980 \cdot 10^{-3}\%$	$2.6678 \cdot 10^{-1}\%$	$1.5688 \cdot 10^{-1}\%$
f_5	$2.8628 \cdot 10^{-4}\%$	$3.3936 \cdot 10^{-4}\%$	$1.0076 \cdot 10^{-2}\%$
f_6	$2.1288 \cdot 10^{-3}\%$	$1.2755 \cdot 10^{-2}\%$	$3.2081 \cdot 10^{-2}\%$

the relative error between the function to be approximated and the output of the fuzzy system), which show beyond doubt a reduction in the error of approximation due to the adoption of the Beta function.

In the other two series of simulations, we considered the following 5 new functions:

$$\text{square}(x) = \text{sign}(\cos(x)), \quad -3\pi \leq x \leq 3\pi; \tag{29}$$

$$\text{parabola}(x) = x^2, \quad 0 \leq x \leq 3; \tag{30}$$

$$\text{sine}(x) = \sin(x), \quad -3\pi \leq x \leq 3\pi; \tag{31}$$

$$\text{deadenedsine}(x) = \sin(x)\exp(-0.1x), \quad -3\pi \leq x \leq 3\pi; \tag{32}$$

$$\text{logarithm}(x) = \log(x), \quad 0.2 \leq x \leq 3. \tag{33}$$

In the second series of simulations, we fixed the number of rules (or the neurons in the hidden layer) and we sought to minimize the relative error of approximation by Gaussian and Beta functions to find the results shown in Table 3, once again confirming the superiority of the Beta function over the Gaussian function. For the square function, we have obtained excellent performances with just 3 Beta functions, which is far from being the case of the Gaussian function.

Table 3. Relative approximation error with FLS of the same number of rules

Function	Number of rules	BNFS	Gaussian-based FLS
square	3	$8.5122 \cdot 10^{-2}\%$	$3.6484 \cdot 10^{-1}\%$
parabola	3	$3.4133 \cdot 10^{-4}\%$	$3.4133 \cdot 10^{-4}\%$
sine	6	$3.9851 \cdot 10^{-2}\%$	$4.0113 \cdot 10^{-2}\%$
deadened sine	6	$9.4938 \cdot 10^{-3}\%$	$9.4965 \cdot 10^{-3}\%$
logarithm	4	$5.3821 \cdot 10^{-3}\%$	$7.1865 \cdot 10^{-3}\%$

In the third series of simulations, we left free the number of rules (or the neurons in the hidden layer) and we sought to minimize, with a common error threshold, the relative error of approximation by Gaussian functions and Beta functions to find the results (see Table 4) which clearly demonstrate the flexibility of the Beta function in the design of powerful neuro-fuzzy systems with a low number of rules or hidden neurons.

Table 4. Number of rules for different FLS with the same error goal

Function	Error goal	BNFS	Gaussian-based FLS
square	$1.0 \cdot 10^{-2}\%$	4 rules	59 rules
parabola	$1.0 \cdot 10^{-4}\%$	6 rules	9 rules
sine	$1.0 \cdot 10^{-2}\%$	10 rules	17 rules
deadened sine	$1.0 \cdot 10^{-3}\%$	11 rules	15 rules
logarithm	$1.0 \cdot 10^{-3}\%$	5 rules	7 rules

8. Conclusions

We have presented the Beta function and its main properties. Based on the central-limit theorem, we have shown that Beta functions given by Equation (8) used as Beta fuzzy basis functions can approximate any fuzzy basis function given by Equation (18) for any membership function used in fuzzy logic systems.

We have investigated the use of Beta fuzzy basis functions for the design of fuzzy logic systems and Beta-based function neural networks. A functional equivalence between Beta fuzzy logic systems and Beta based function neural networks has also been demonstrated to obtain a new kernel-based architecture known as Beta neuro-fuzzy systems.

By using the Stone-Weierstrass theorem and expanding the output of Beta neuro-fuzzy systems in series of Beta fuzzy basis functions, we have proven that any real continuous function on a compact set may be uniformly approximated to any arbitrary accuracy. Thus, Beta neuro-fuzzy systems have been shown to be universal approximators.

Recently, other theoretical works on Beta neuro-fuzzy systems have proven a theorem concerning better approximation of Beta neuro-fuzzy systems (it should be noticed that this property is not verified by the multi-layer perceptrons neural networks [11]), as well as another theorem, concerning the unicity of the best approximation of Beta neuro-fuzzy systems [34].

The Beta neuro-fuzzy system was successfully implemented as a VLSI circuit [35, 36] and applied in the following areas:

- control of non-linear systems [37],
- control of robotic systems [38, 39],
- robot trajectory planning [40],
- recognition of isolated spoken Arabic words [41],
- recognition of printed numerals [42],
- recognition of on-line handwritten characters [43],
- recognition of Arabic handwritten characters [44],
- recognition of on-line cursive handwriting [45],
- recognition of on-line Arabic handwriting [46].

Acknowledgements

The author wishes to thank Professor Lotfi A. Zadeh for his advice and the fruitful discussions at the International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial

Applications: ACIDCA'2000, (Monastir, Tunisia). The author also wishes to thank Professor Nabil Derbel for his continuous help and support and would like to acknowledge the financial support of this work by grants from the Franco-Tunisian cooperation project CMCU no. 99 F 14 07 and the General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program.

Appendix A: Proof of theorem 1

We wish to find the parameters of the Beta function $\beta(x;\, p,q,x_0,x_1)$ which approximates the Gaussian function as well as possible $\text{Gauss}(x;\, \mu,\sigma)$, that is for all given μ and σ, find p, Q, x_0 and x_1 such that:

$$\beta(x;\, p,q,x_0,x_1) \simeq \text{Gauss}(x;\, \mu,\sigma), \tag{34}$$

$$\frac{d\beta(x;\, p,q,x_0,x_1)}{dx} \simeq \frac{d\text{Gauss}(x;\, \mu,\sigma)}{dx}. \tag{35}$$

The Gaussian function being symmetrical and centered in μ, we must thus choose a symmetrical Beta function centered in μ. One can thus choose:

$$p = q, \tag{36}$$

$$x_c = \mu, \tag{37}$$

$$x_0 = x_c - \Delta, \tag{38}$$

$$x_1 = x_c + \Delta, \tag{39}$$

One thus obtains:

$$|\beta(\mu;\, p,q,x_0,x_1) - \text{Gauss}(\mu;\, \mu,\sigma)| = |1-1| = 0,$$
$$|\beta(x_0;\, p,q,x_0,x_1) - \text{Gauss}(x_0;\, \mu,\sigma)| = |0 - \exp^{-n^2/2}|,$$

and

$$|\beta(x_1;\, p,q,x_0,x_1) - \text{Gauss}(x_1;\, \mu,\sigma)| = |0 - \exp^{-n^2/2}|,$$

that tends towards zero if n tends towards infinity.

In order to respect Equation (12) for the Beta function, one can choose:

$$\Delta = n\sigma. \tag{40}$$

Finally, Equations (5), (11), (34), (35) give:

$$\left[\frac{px_1 + qx_0 - (p+q)x)}{(x-x_0)(x_1-x)}\right] \simeq -\frac{(x-\mu)}{\sigma^2} \tag{41}$$

and since $p = q$ and $x_0 + x_1 = 2\mu$, one obtains easily:

$$p = \frac{(x-x_0)(x_1-x)}{2\sigma^2}. \tag{42}$$

Equation (42) is valid for all $x \in]x_0,x_1[$, and in particular for $x = x_c$, which gives:

$$p = \frac{n^2}{2}. \tag{43}$$

Let us now show that the Beta function which we have just built verifies the inequality $|\beta(x;\, p,q,x_0,x_1) - \text{Gauss}(x;\, \mu,\sigma)| \leq \varepsilon$ for all $x \in \mathbb{R}$.

$$\beta(x;\, p,q,x_0,x_1) = \left[\frac{x-\mu+n\sigma}{n\sigma}\right]^{n^2/2} \left[\frac{\mu+n\sigma-x}{n\sigma}\right]^{n^2/2}, \tag{44}$$

$$= \left[\frac{1}{n\sigma}\right]^{2n^2/2} \left[(n\sigma)^2 - (x-\mu)^2\right]^{n^2/2}, \tag{45}$$

$$= \left[\frac{1}{n\sigma}\right]^{n^2} \left[(n\sigma)^2\right]^{n^2/2} \left[1 - \left(\frac{x-\mu}{n\sigma}\right)^2\right]^{n^2/2}, \tag{46}$$

$$= \exp\left\{\frac{n^2}{2}\log\left[1 - \left(\frac{x-\mu}{n\sigma}\right)^2\right]\right\}, \tag{47}$$

$$= \exp\left\{\frac{n^2}{2}\left[-\left(\frac{x-\mu}{n\sigma}\right)^2 + O(1/n^2)\right]\right\}, \tag{48}$$

$$= \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2} + \frac{n^2}{2}O(1/n^2)\right\}. \tag{49}$$

It is well known that for all given ε, there exists n sufficiently large such that:

$$\left|\exp\left\{-\frac{(x-\mu)^2}{2\sigma^2} + \frac{n^2}{2}O(1/n^2)\right\} - \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}\right| \leq \varepsilon$$

and thus for all given ε, there exists n sufficiently large such that:

$$|\beta(x; n^2/2, n^2/2, \mu - n\sigma, \mu + n\sigma) - \text{Gauss}(x; \mu, \sigma)| \leq \varepsilon.$$

Appendix B: Proof of theorem 2

Based on theorem 1, one can approximate to any precision any Gaussian function by a Beta function. Theorem 2 is then automatically deduced.

Appendix C: Proof of theorem 3

The proof easily results from the observation that the output of a Beta based function neural network, given by Equation (15), is identical to the output of the Beta fuzzy logic system, given by Equation (19).

Appendix D: Proof of theorem 4

We will use the Stone-Weierstrass theorem [20, 21] to prove theorem 4.

First, we will prove that the set of Beta neuro-fuzzy systems (BNFS) is a subalgebra of $C[\mathscr{I}]$. Let f_1, f_2 be two BNFS, so we can write these functions as:

$$f_1(\underline{x}) = \sum_{j=1}^{R_1} f_{1j}(\underline{x})\beta_{1j}(\underline{x}), \tag{50}$$

$$f_2(\underline{x}) = \sum_{j=1}^{R_2} f_{2j}(\underline{x})\beta_{2j}(\underline{x}), \tag{51}$$

$$f_1(\underline{x}) + f_2(\underline{x}) = \sum_{k=1}^{R_1+R_2} \{\delta_{1k} f_{1k}(\underline{x})\beta_{1k}(\underline{x}) + \delta_{2k} f_{2k}(\underline{x})\beta_{2k}(\underline{x})\}, \tag{52}$$

$$\delta_{1k} = 1 \text{ if } k \leq R_1, \text{ 0 otherwise}, \tag{53}$$

$$\delta_{2k} = 1 \text{ if } k > R_1, \text{ 0 otherwise}, \tag{54}$$

$$f_1(\underline{x})f_2(\underline{x}) = \sum_{j=1}^{R_1}\sum_{k=1}^{R_2}[f_{1j}(\underline{x})f_{2k}(\underline{x})][\beta_{1j}(\underline{x})\beta_{2k}(\underline{x})]. \tag{55}$$

Equation (52) is in the same form as Equation (19), so that $f_1 + f_2$ is a BNFS.

Similarly, we have Equation (55), which is also in the same form of Equation (19), since the product of two Beta functions given by Equation (8) is a Beta function; hence $f_1 \times f_2$ is a BNFS.

Finally, for any $\alpha \in \mathbb{R}$:

$$\alpha f_1(\underline{x}) = \sum_{j=1}^{R_1}[\alpha f_{1j}(\underline{x})]\beta_{1j}(\underline{x}), \tag{56}$$

which is again in the form of Equation (19); hence, αf_1 is a BNFS.

Therefore, the set of BNFS is a subalgebra of $C[\mathscr{I}]$.

Next, we will prove that the set of BNFS verifies condition 1 of the Stone-Weierstrass theorem.

This can be trivially obtained by noting that for $p = q = 0$:

$$1 = 1\prod_{k=1}^{n}(x_k - x_{k,\min})^0(x_{k,\max} - x_k)^0, \tag{57}$$

where $x_{k,\min}$ and $x_{k,\max}$ are the bounds of $\mathscr{I}$ in each dimension. This equation has the same form as that of a Beta function given by Equation (8) and for which $x_{k,\min} < x_{0,k}$ and $x_{k,\max} < x_{1,k}$.

Finally, we will prove that the set of BNFS separates points on $\mathscr{I}$ (condition 2).

Let's consider two $\underline{a}, \underline{b} \in \mathscr{I}$ such that $\underline{a} \neq \underline{b}$. Suppose for example that $a_i \neq b_i$. Let's construct an $f \in BFBFe$ such that $f(\underline{b}) \neq f(\underline{a})$. Let's choose f as following: $f(\underline{x}) = (x_i - a_i)$. One can easily see that $f(\underline{a}) = 0$ and $f(\underline{b}) = (b_i - a_i) \neq 0$, since $a_i \neq b_i$.

References

[1] Takagi H 1990 *Proc. Int. Conf. Fuzzy Logic and Neural Networks*, New Orleans, USA, pp. 13–26

[2] Wang L X 1994 *Adaptive Fuzzy Systems and Control*, Prentice-Hall, Englewood Cliffs, NJ

[3] Johnson N I 1970 *Continuous Univariate Distributions*, Houghton Mifflin Company, Boston

[4] Alimi A M 1998 *Proc. IEEE/IMACS Multiconference on Computational Engineering in Systems Applications: CESA'98*, Hammamet, Tunisia **2** 339

[5] Alimi A M 1997 *Proc. Séminaire sur la Commande Robuste et Applications: SCRA'97*, Nabeul, Tunisia, pp. C1–C5

[6] Alimi A M 2000 *Int. J. Management* Invited Paper no. 15–19

[7] Gnedenko B V and Kolmogorov A 1954 *Limit Distributions for Sums of Independent Random Variables*, Addison-Wesley Publ. Co., Reading, MA

[8] Uspensky J V 1937 *Introduction to Mathematical Probability*, McGraw-Hill Book Co., New York

[9] Papoulis A 1962 *The Fourier Integral and Its Applications*, McGraw-Hill Book Co., New York

[10] Haykin S 1994 *Neural Networks: A Comprehensive Foundation*, MacMillan, New York

[11] Pogio T and Girosi F 1990 *Proc. IEEE* **78** (9) 1481

[12] Kalman R E 1960 *Proc. 1st Int. Congress of Automatic Control*, Moscow, London: Butterworths **1** 481

[13] Zadeh L A 1965 *Information Control* **8** 338

[14] Zeng X J and Singh M G 1995 *IEEE Trans. Fuzzy Systems* **3** (2) 219

[15] Lee C C 1990 *IEEE Trans. Syst., Man, and Cybern.* **20** (2) 404

[16] Sugeno M and Kang G T 1988 *Fuzzy Sets and Systems* **28** 15
[17] Takagi T and Sugeno M 1985 *IEEE Trans. Syst., Man, and Cybern.* **15** 116
[18] Jang J S R and Sun C T 1993 *IEEE Trans. Neural Networks* **4** (1) 156
[19] Hunt K J, Haas R and Murray-Smith R 1996 *IEEE Trans. Neural Networks* **7** (3) 776
[20] Stone M H 1948 *Math. Mag.* **21** 167; *ibid.* 237
[21] Stone M H 1937 *AMS Trans.* **41** 375
[22] Wang L X and Mendel J M 1992 *IEEE Trans. Neural Networks* **3** (5) 807
[23] Alimi A M, Hassine R and Selmi M 2000 *Int. J. Appl. Math. Comput. Sci.* **10** (4) 101
[24] Alimi A M, Hassine R and Selmi M 2002 *Int. J. Appl. Math. Comput. Sci.* (accepted)
[25] Aouiti C, Alimi A M and Maalej A 2002 *Systems Analysis, Modeling, and Simulation, Special Issue on "Advances in Control and Computer Engineering"* (in print)
[26] Aouiti C, Alimi A M and Maalej A 2000 *Proc. Int. Conf. Computational and Artificial Intelligence for Decision, Control and Automation: ACIDCA'2000*, Monastir, Tunisia **IM** 88
[27] Hassine R, Alimi A M and Selmi M 2000 *Proc. Int. Conf. Computational and Artificial Intelligence for Decision, Control and Automation: ACIDCA'2000*, Monastir, Tunisia **IM** 72
[28] Njah M, Alimi A M, Chtourou M and Tourki R 2002 *NeuroComputing* (accepted)
[29] Njah M, Alimi A M and Chtourou M 2000 *Proc. Int. Conf. Computational and Artificial Intelligence for Decision, Control and Automation: ACIDCA'2000*, Monastir, Tunisia **IM** 76
[30] Cho K B and Wang B H 1996 *Fuzzy Sets and Systems* **83** 325
[31] Lee S and Kil R M 1991 *Neural Networks* **4** 207
[32] Press W H, Flannery B P, Teukolsky S A and Vetterling W T 1989 *Numerical Recipes in C*, Cambridge University Press, Cambridge, England
[33] Mitaim S and Kosko B 1996 *Proc. IEEE Int. Conf. Fuzzy Systems*, New Orleans, pp. 1237–1243
[34] Hassine R, Alimi A M and Selmi M 2000 *New Frontiers in Computational Intelligence and its Applications* (Mohammadian M, Ed.), IOS Press, The Netherlands, pp. 62–67
[35] Masmoudi M, Samet M and Alimi A M 2000 *Int. J. Electronics* **87** (6) 675
[36] Samet M, Masmoudi M and Alimi A M 2001 *Int. J. Electronics* **88** (6) 645
[37] Derbel N and Alimi A M 1997 *Proc. Journées Tunisiennes d'Electrotechnique et d'Automatique: JTEA'97*, Nabeul, Tunisia **1** 66
[38] Bezine H, Derbel N and Alimi A M 2002 *Engineering Applications of Artificial Intelligence* (accepted)
[39] Bezine H, Derbel N and Alimi A M 2000 *Proc. Int. Conf. Computational and Artificial Intelligence for Decision, Control and Automation: ACIDCA'2000*, Monastir, Tunisia **SAAC** 135
[40] Ouezri A, Derbel N and Alimi A M 2002 *Systems Analysis, Modeling, and Simulation, Special Issue on "Advances in Control and Computer Engineering"* (in print)
[41] Alimi A M and Ben Jemaa M 2002 *J. Control and Intelligent Systems* **30** (2) 47
[42] Charfi M and Alimi A M 1998 *Proc. IEEE/IMACS Multiconference on Computational Engineering in Systems Applications: CESA'98*, Hammamet, Tunisia **4** 439
[43] Alimi A M 1998 *Proc. IEEE/IMACS Multiconference on Computational Engineering in Systems Applications: CESA'98*, Hammamet, Tunisia **2** 335
[44] Alimi A M 1997 *Proc. Int. Conf. Neural Networks: ICNN'97*, Houston, TX, USA, pp. 1397–1400
[45] Alimi A M 2002 *IETE J. Research, Special Issue on "Evolutionary Computation in Engineering Sciences"* (Pal S K *et al.*, Eds.) (in print)
[46] Alimi A M 1997 *Proc. Int. Conf. Document Analysis and Recognition: ICDAR'97*, Ulm, Germany, pp. 382–386