University of Ostrava

Institute for Research and Applications of Fuzzy Modeling

A Neural Network Approach to the Fuzzy Transform

Martin Štěpnička, Ondřej Polakovič

Research report No. 141

2009

Submitted/to appear:

Fuzzy Sets and Systems

Supported by:

Projects MSM6198898701 of the MŠMT ČR and DAR 1M0572 of the MŠMT ČR.

University of Ostrava
Institute for Research and Applications of Fuzzy Modeling
30. dubna 22, 701 03 Ostrava 1, Czech Republic

tel.: +420-59-7091401 fax: +420-59-6120478
e-mail: martin.stepnicka@osu.cz

A Neural Network Approach to the Fuzzy Transform ⋆

Martin Štěpnička and Ondřej Polakovič

Institute for Research and Applications of Fuzzy Modeling, University of Ostrava, 30. dubna 22, 701 03 Ostrava 1, Czech Republic

Abstract

The paper deals with the F-transform technique which was introduced as a method for an approximate representation of continuous functions. The same task is solved by many methods from different areas. Neural networks also belong to techniques which have been proved to be powerful for approximation objectives. They provide us with many advantages, especially incremental way of learning parameters. The paper inherits neural approaches to the F-transform method and presents experiments justifying the proposed approach. Incremental way of determination of certain parameters of the F-transform method which were up to now given by batch formula enriches possible areas of application of the method by fast on-line processes. Moreover, the neural approach is demonstrated to be an appropriate one for finding such fuzzy partition of a domain which respects better a given set of measured samples which are to be approximated by a continuous function with no predetermined shape.

Key words: Fuzzy approximation, Fuzzy transform, Neural network, RBF, Learning
1991 MSC: 68T30, 68T35, 68T37

1 Introduction

Fuzzy approximation is a quickly developing mathematical branch aiming at approximation of some dependencies by means of the fuzzy set theory and the fuzzy logic in broader sense. We can find its roots in the Takagi-Sugeno fuzzy

⋆ We gratefully acknowledge partial support of the projects MSM6198898701 and DAR 1M0572 of the MŠMT ČR.

rule based systems [16] and in works aiming at approximation capabilities of fuzzy rule based systems, see [1,6].

Motivation of the paper is to introduce the study of relationships between fuzzy approximation methods and other approximation techniques. It deals with a particular fuzzy approximation method called the *fuzzy transform* (F-transform) [10,11] and neural networks as another soft computing area which has been many times demonstrated to be an appropriate tool for approximation tasks.

By studying both approaches together we expect:

- development of new algorithms (known in neural networks) for fuzzy approximation
- enriching both branches by already done results from each other
- possible improvements
- answering natural question about similarities and similar problems in both branches
- inheriting theoretical results e.g. conditions of universal approximations etc.

At this first stage of our investigation, we try to look at the fuzzy transform from a neural network point of view to open this problematic, inherit neural network algorithms, investigate possible improvements, implement an incremental type of learning and build a bridge between both branches for next theoretical results and algorithmic improvements.

Structure of the paper is as follows. Section 2 focuses on the fuzzy transform, recalls basic definitions and facts about the fuzzy transform and its inversion and introduces some proved direct generalizations. Section **??** is devoted to radial basis function neural networks and presents a network performing the inverse fuzzy transform. The main aim of Section 3 is to inherit some algorithms from the neural network area and apply them to the F-transform. Good behaviour of the proposed algorithm is then justified by experiments in Section 4 and discussed in Section 5.

2 Fuzzy transform

This sections is devoted to the fuzzy transform (F-transform for short) [10,11] which is a method for an approximation of continuous functions.

We will assume an interval $[a, b]$ to be a common domain of all functions in the latter. This domain is partitioned by a *fuzzy partition* consisting of fuzzy sets. For purposes of further usage of the fuzzy transform we restrict our choice of

fuzzy partitions to those which are comprised of the so called *basic functions*.

Definition 1 Let $c_0 = c_1 < \cdots < c_n = c_{n+1}$ be fixed nodes within $[a, b]$ such that $c_1 = a$, $c_n = b$ and $n \geq 2$. We say that fuzzy sets $\mathbf{A}_1, \ldots, \mathbf{A}_n$ are *basic functions* and form a *fuzzy partition* of $[a, b]$ if the following conditions

(i) $\mathbf{A}_i : [a, b] \to [0, 1],\ \mathbf{A}_i(c_i) = 1,\ i = 1, \ldots, n$;
(ii) $\mathbf{A}_i(x) = 0$ if $x \notin (c_{i-1}, c_{i+1})$;
(iii) $\mathbf{A}_i$ is continuous;
(iv) $\mathbf{A}_i,\ i = 2, \ldots, n$, strictly increases on $[c_{i-1}, c_i]$ and $\mathbf{A}_i, i = 1, \ldots, n\ -\ 1$, strictly decreases on $[c_i, c_{i+1}]$;
(v) $\sum_{i=1}^{n} \mathbf{A}_i(x) = 1$, for all $x \in [a, b]$;

hold. If the nodes $c_1, \ldots, c_n$ are given equidistantly i.e. $c_i = a + h(i-1), i = 1, \ldots, n$ where $h = (b-a)/(n-1)$ and the following two additional properties are met:

(vi) $\mathbf{A}_i(c_i - x) = \mathbf{A}_i(c_i + x)$, for all $x \in [0, h],\ i = 2, \ldots, n-1,\ n > 2$,
(vii) $\mathbf{A}_{i+1}(x) = \mathbf{A}_i(x - h)$, for all $x \in [a + h, b],\ i = 2, \ldots, n-2,\ n > 2$,

we call the fuzzy partition *uniform*.

The following lemma claims that the definite integral of a basic function does not depend on its shape.

Lemma 2 *[11] Let a uniform fuzzy partition of $[a, b]$ be given by basic functions $\mathbf{A}_1, \ldots, \mathbf{A}_n, n > 2$. Then*

$$\int_a^b \mathbf{A}_1(x)dx = \int_a^b \mathbf{A}_n(x)dx = \frac{h}{2} \tag{1}$$

and for $i = 2, \ldots, n-1$

$$\int_a^b \mathbf{A}_i(x)dx = h. \tag{2}$$

Lemma 2 can be generalized for basic functions determining a fuzzy partition which is not uniform.

Lemma 3 *Let a fuzzy partition of $[a, b]$ be given by basic functions $\mathbf{A}_i, \ldots, \mathbf{A}_n, n > 2$. Denote $h_i = c_{i+1} - c_i$. If the following condition*

(vi') $\mathbf{A}_i(c_i + x) = \mathbf{A}_{i+1}(c_{i+1} - x),\ x \in [0, h_i]$

holds for $i = 1, \ldots, n-1$ then

$$\int_a^b \mathbf{A}_i(x)dx = \frac{(h_{i-1} + h_i)}{2}. \tag{3}$$

PROOF. First of all let us express the value h_i with help of integrals:

$$h_i = \int_{c_i}^{c_{i+1}} 1dx = \int_{c_i}^{c_{i+1}} \sum_{k=1}^{n} \mathbf{A}_k(x)dx = \int_{c_i}^{c_{i+1}} \mathbf{A}_i(x)dx + \int_{c_i}^{c_{i+1}} \mathbf{A}_{i+1}(x)dx. \quad (4)$$

From assumption (vi') we immediately get

$$\int_{c_i}^{c_{i+1}} \mathbf{A}_i(x)dx = \int_{c_i}^{c_{i+1}} \mathbf{A}_{i+1}(x)dx \quad (5)$$

and because of (4) we can state that both sides of equation (5) are equal to $h_i/2$ which proves the lemma. □

It is easy to see that Lemma 3 generalizes Lemma 2 which is a special case because conditions of uniformity of a fuzzy partition imply condition (vi') and obviously $h_0 = h_n = 0$. Let us consider the right hand side of (vi'). Then due to properties of uniform fuzzy partition we can write

$$\mathbf{A}_{i+1}(c_i - x) = \mathbf{A}_i(c_{i+1} - x - h) = \mathbf{A}_i(c_i - x) = \mathbf{A}_i(c_i + x)$$

which means that assumption (vi') is fulfilled.

Let $C([a, b])$ stands for the space of continuous functions on $[a, b]$ and let $f \in C([a, b])$. Let us recall the definition of the fuzzy transform of f.

Definition 4 [10] Let $\mathbf{A}_1, \ldots, \mathbf{A}_n$ be basic functions which form a fuzzy partition on $[a, b]$ and f be an arbitrary function from $C([a, b])$. We say that the n-tuple of real numbers $[F_1, \ldots, F_n]$ given by

$$F_i = \frac{\int_a^b f(x)\mathbf{A}_i(x)dx}{\int_a^b \mathbf{A}_i(x)dx},\ i = 1, \ldots, n \quad (6)$$

is the *fuzzy transform (F-transform)* of f w.r.t. the given fuzzy partition and $F_i, i = 1, \ldots, n$ are the *components* of the F-transform.

Remark 5 *Due to Lemma 3 we can simplify the computation of the F-transform given by formula (6) and write*

$$F_i = \frac{2}{(h_{i-1} + h_i)} \int_a^b f(x)\mathbf{A}_i(x)dx \quad (7)$$

Let us recall a lemma claiming that the definite integral of f can be computed with help of the F-transform which can be useful in further numerical methods based on this approximation model.

Lemma 6 *[10] Let a uniform fuzzy partition of $[a, b]$ be given by basic functions $\mathbf{A}_i, \ldots, \mathbf{A}_n, n > 2$ and let $f \in C([a, b])$. Then*

$$\int_a^b f(x)dx = h(\frac{1}{2}F_1 + F_2 + \cdots + F_{n-1} + \frac{1}{2}F_n) \tag{8}$$

where $F_i, i = 1, \ldots, n$ are the components of the F-transform w.r.t. the given fuzzy partition.

Due to Lemma 3 we can formulate the following generalization.

Lemma 7 *Let a fuzzy partition of $[a, b]$ be given by basic functions $\mathbf{A}_i, \ldots, \mathbf{A}_n, n > 2$ which fulfill condition (vi') and let $f \in C([a, b])$. Then*

$$\int_a^b f(x)dx = \frac{1}{2}\sum_{i=1}^{n}((h_{i-1} + h_i)F_i) \tag{9}$$

where $F_i, i = 1, \ldots, n$ are the components of the F-transform w.r.t. the given fuzzy partition.

PROOF. By direct computation one gets

$$\int_a^b f(x)dx = \int_a^b f(x)\sum_{i=1}^{n}\mathbf{A}_i(x)dx = \sum_{i=1}^{n}\int_a^b f(x)\mathbf{A}_i(x)dx$$

which due to Lemma 3 (see Remark 5) equals to

$$\sum_{i=1}^{n}\left(\frac{(h_{i-1} + h_i)}{2}F_i\right)$$

which proves the lemma. □

The F-transform serves as a discrete approximate representation of a given function. It has been shown that it is an appropriate approximate representation by proving the fact that its components approximately equal to function values at respective nodes.

Lemma 8 *[10] Let a uniform fuzzy partition of $[a, b]$ be given by basic functions $\mathbf{A}_i, \ldots, \mathbf{A}_n, n > 2$ and let $f \in C([a, b])$. Then*

$$F_i = f(c_i) + O(h^2),\ i = 1, \ldots, n \tag{10}$$

where $F_i, i = 1, \ldots, n$ are the components of the F-transform w.r.t. the given fuzzy partition.

Analogously to the previous results, we can state the following lemma generalizing this property even for the F-transform w.r.t. basic functions comprising a non-uniform fuzzy partition.

Lemma 9 *Let a fuzzy partition of* $[a,b]$ *be given by basic functions* $\mathbf{A}_i,\dots,\mathbf{A}_n, n > 2$ *which fulfill condition (vi') and let* $f \in C([a,b])$. *Then*

$$F_i = f(c_i) + O(h_{i-1}^2 + h_i^2),\ i = 1,\dots,n \tag{11}$$

where $F_i, i = 1,\dots,n$ *are the components of the F-transform w.r.t. the given fuzzy partition.*

PROOF. To prove the lemma we use the technique already used in [10] to prove Lemma 8.

$$F_i = \frac{2}{(h_{i-1}+h_i)}\left(\int_{c_{i-1}}^{c_i} f(x)\mathbf{A}_i(x)dx + \int_{c_i}^{c_{i+1}} f(x)\mathbf{A}_i(x)dx\right)$$

and to both integrals on the right hand side we apply the trapezium formula i.e.

$$F_i = \frac{2}{(h_{i-1}+h_i)}\frac{h_{i-1}}{2}\left(f(c_{i-1})\mathbf{A}_i(c_{i-1}) + f(c_i)\mathbf{A}_i(c_i)\right) + O(h_{i-1}^2)$$

$$+\frac{2}{(h_{i-1}+h_i)}\frac{h_i}{2}\left(f(c_i)\mathbf{A}_i(c_i) + f(c_{i+1})\mathbf{A}_i(c_{i+1})\right) + O(h_i^2) =$$

$$\frac{h_{i-1}}{(h_{i-1}+h_i)}f(c_i) + \frac{h_i}{(h_{i-1}+h_i)}f(c_i) + O(h_{i-1}^2 + h_i^2) =$$

$$= f(c_i) + O(h_{i-1}^2 + h_i^2).$$

□

The F-transform is a discrete approximate representation of f which can replace it in complex numerical computations. Let us recall the following definition of the inverse F-transform which maps the discrete representation back to the space of continuous functions.

Definition 10 Let $\mathbf{A}_1,\dots,\mathbf{A}_n$ be basic functions and $[F_1,\dots,F_n]$ be the F-transform of a function f w.r.t. the given fuzzy partition. Then function

$$f_{F,n}(x) = \sum_{i=1}^{n} F_i\mathbf{A}_i(x) \tag{12}$$

is called the *inverse F-transform*

The universal approximation property and the uniform convergence of a sequence of the F-transforms to the original approximated function have been proven as well, see [11].

If we deal with an approximation of a function it is necessary to distinguish the approximation among other possible ones. This is guaranteed by a minimization of a certain criterion which defines a closeness of an original function to its approximation.

Assume, that the basic functions are fixed. In [10], there was an error function $E : \mathbb{R}^n \to \mathbb{R}^+$ given as follows

$$E(Q_1, \ldots, Q_n) = \int_a^b \left(\sum_{i=1}^{n} (f(x) - Q_i)^2 \mathbf{A}_i(x) \right) dx, \quad Q_i \in \mathbb{R} \tag{13}$$

proposed to measure the quality of a discrete approximation of a function. By direct computation, one can check that the error function E called *piecewise integral least square criterion*[10] is minimized by the components of the F-transform F_i i.e. $E(Q_1, \ldots, Q_n) \leq E(F_1, \ldots, F_n)$ for F_i given by (6) and arbitrary $Q_i \in \mathbb{R}$. Advantages of criterion (13) were discussed, e.g. in [12,15].

Usually in practical situations, function f is not given analytically and we are provided only with a set of, say measured, samples $(x_k, f(x_k))$ where $k = 1, \ldots, K$ and $n << K$, in principle. For this case, formula (6) defining the components is modified as follows

$$F_i = \frac{\sum_{k=1}^{K} f(x_k)\mathbf{A}_i(x_k)}{\sum_{k=1}^{K} \mathbf{A}_i(x_k)}, \quad i = 1, \ldots, n \tag{14}$$

and sometimes we talk about so called *discrete F-transform*.

In the case of the discrete F-transform, criterion (13) is modified to the following one

$$E(Q_1, \ldots, Q_n) = \sum_{k=1}^{K} E_k(Q_1, \ldots, Q_n) \tag{15}$$

where

$$E_k(Q_1, \ldots, Q_n) = \sum_{i=1}^{n} (f(x_k) - Q_i)^2 \mathbf{A}_i(x_k), \quad Q_i \in \mathbb{R}. \tag{16}$$

for $k = 1, \ldots, K$.

And again, it has been proved that the (discrete) F-transform components given by (14) minimize criterion (15)-(16). For proofs and details we refer to [11].

3 Fuzzy transform as an RBF neural network

Compared to fuzzy techniques, neural nets are usually implemented as black boxes but they also have advantages such as e.g. an algorithmic approach to an identification of a model or *incremental* (on-line) learning algorithms which can be very useful for approximation tasks in various applications [8].

Basic functions $\mathbf{A}_i$, $i = 1, \ldots, n$ partitioning the domain $[a, b]$ can be viewed (in the neural network terminology) as local units. Let us consider RBF (*Radial Basis Function*) one hidden layer neural nets with local units in the hidden layer and with one linear unit (performing the identity activation function) in the output layer.

Usually, RBF neural networks deal with a continuous non-increasing activation function $\mathbf{A} : \mathbb{R}^+ \to [0, 1]$ and the inner potential is given by the following formula

$$\xi_i(\mathbf{x}) = \frac{||\mathbf{x} - \mathbf{c}_i||}{h_i} \tag{17}$$

where $\mathbf{x} \in \mathbb{R}^m$ is an input vector, $\mathbf{c}_i \in \mathbb{R}^m$ is a vector determining *center* of the i-th unit and finally $h_i \in \mathbb{R}^+$ is a parameter determining the width of the i-th unit. For more details we refer to [2,3,5,14].

In the latter, we restrict our focus on the case $m = 1$ for a simplified visualization. The above displayed RBF neural network is depicted on Figure 1.

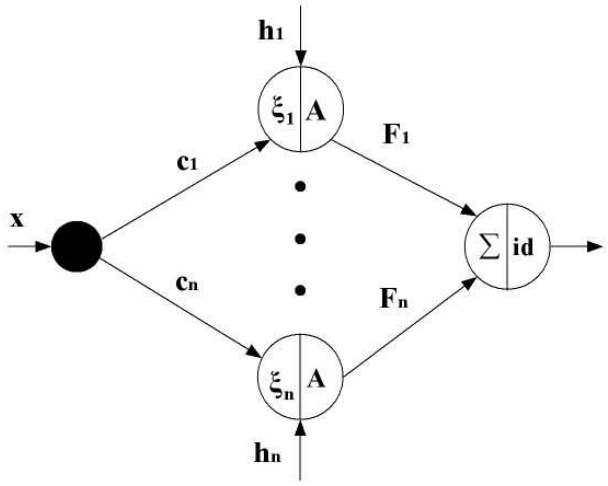


Fig. 1. One hidden layer RBF neural network with a linear unit in the output layer

The basic functions from Definition 1 can be constructed in the presented RBF neural network way. For instance, if we take $\mathbf{A}(z) = (1 - z) \wedge 0,\ z \in \mathbb{R}$ then it is easy to check that

$$\mathbf{A}_i(x) = \mathbf{A}\left(\xi_i(x)\right) \tag{18}$$

where $\mathbf{A}_i$, $i = 1, \ldots, n$ are triangular shaped basic functions determining a uniform fuzzy partition i.e. $h = h_i$ for $i = 1, \ldots, n-1$. Analogously, if we take

$$\mathbf{A}(z) = \begin{cases} \frac{1}{2}\left(\cos\left(\Pi z\right) + 1\right) & z \leq 1, \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

which is for $z \in \mathbb{R}^+$ obviously a non-increasing function then one can again check that equality (18) holds for sinusoidal shaped basic functions $\mathbf{A}_i$, $i = 1, \ldots, n$. So, the neural network depicted on Figure 1 may perform the inverse F-transform function for in case of appropriately chosen parameters.

3.1 Learning algorithm

The most important feature of neural networks is hidden in the possibility to learn or tune distinguish parameters, especially incrementally. Otherwise the neural network would serve just as a visualization, as in case of the F-transform and the neural network on Figure 1

In the terminology of the neural nets, the computation of the components of the F-transform F_i according to (14) is called *off-line* (or batch) learning. However, for certain applications incremental learning algorithms have to be used [8], especially for on-line identification problems where we have to avoid complete rebuilding of a model because of new measurements which could yield high computational efforts.

From the original definitions we keep only the inverse F-transform formula which is performed by the RBF neural net displayed on Figure 1 and criterion (15)-(16) which is to be minimized. Formula (14) defining the F-transform components will be replaced by an on-line algorithm. For this purpose, we adopt the *delta rule* which modifies weights (components F_i in this case) after each new sample $(x_k, f(x_k))$ is involved. The gradient descent method as a standard tool for finding the delta is used.

To minimize the error function $E(Q_1, \ldots, Q_n) = \sum_{k=1}^{K} E_k(Q_1, \ldots, Q_n)$ where E_k are given by (16), we differentiate

$$\frac{\partial E_k}{\partial Q_i}, for \quad i = 1, \ldots, n \tag{20}$$

which is

$$\frac{\partial E_k}{\partial Q_i} = 2\mathbf{A}\left(\xi_i(x_k)\right)\left(f(x_k) - Q_i\right) \tag{21}$$

where

$$\xi_i(x_k) = \frac{|x_k - c_i|}{h} \tag{22}$$

and hence the delta rule is as given follows

$$F_i^{(k)} = F_i^{(k-1)} + \theta_1(f(x_k) - F_i^{(k-1)})\mathbf{A}\left(\xi_i(x_k)\right) \tag{23}$$

where $0 < \theta_1 \leq 1$ is a learning coefficient and $F_i^{(k)}$ is the i-th component of the F-transform after k samples involved where $k = 1, \ldots, K$.

Remark 11 *Notice, that although we use standard RBF neural network and standard neural tools like the gradient descent method together with the delta rule, the error function which is minimized is different compared to usual approaches. We do not compare function values $f(x_k)$ with the outputs of the network but with its weights $F_i^{(k-1)}$. This is a significant difference which is inherited from the F-transform to keep its properties.*

3.2 Learning of other parameters

The construction of the basic functions can be the key issue for the results of the approximation. In general, one can hardly expect that the uniform distribution of the basic functions of the same width would provide us with the best results but on the other hand, the basic functions cannot be chosen arbitrarily and some cluster analysis would have to be used. Therefore, in most applications, the uniform fuzzy partition has been chosen. We will discuss the possibility of the neural approach to the fuzzy partition construction.

The nonsymmetric basic functions are functions of one variable x and three parameters c_{i-1}, c_i, c_{i+1}. Therefore, in the latter, we will again use the notation from the F-transform since it is shorter:

$$\mathbf{A}_i(x) = \mathbf{A}(x, c_{i-1}, c_i, c_{i+1}) \tag{24}$$

for $i = 1, \ldots, n$.

For instance, nonuniform triangular shaped basic functions are given by

$$\mathbf{A}_i(x) = \begin{cases} \frac{(x-c_{i-1})}{c_i - c_{i-1}} & x \in [c_{i-1}, c_i] \\ \frac{(c_{i+1}-x)}{c_{i+1}-c_i} & x \in [c_i, c_{i+1}] \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

where $i = 0, \ldots, n+1$ and $c_0 = c_1$, $c_{n+1} = c_n$ and nonuniform sinusoidal

shaped basic functions are given by

$$\mathbf{A}_i(x) = \begin{cases} \frac{1}{2}\left(\cos\left(\frac{\Pi(x-c_i)}{(c_i-c_{i-1})}\right)+1\right) & x \in [c_{i-1}, c_i] \\ \frac{1}{2}\left(\cos\left(\frac{\Pi(x-c_i)}{(c_{i+1}-c_i)}\right)+1\right) & x \in [c_i, c_{i+1}] \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

where $i = 0, \ldots, n+1$ and $c_0 = c_1,\ c_{n+1} = c_n$.

For a given shape of basic functions the centroids c_i for $i = 1, \ldots, n$ already completely specify the fuzzy partition. The task is to tune the centroids. Again, let us use the advantage of incremental self-organizing (unsupervised) algorithms already developed for neural networks and adopt the *k-means clustering* for RBF neural networks, see [9,14].

The resulting algorithm using both, the self-organizing method for determining a distribution of the nodes c_i and the gradient descent method for adapting the components F_i will be as follows.

Algorithm:

(27)

FOR $k := 1$ **TO** K **DO BEGIN**
  $j = \mathtt{arg}\min_{i=1,\ldots,n}\{|x_k - c_i^{(k-1)}|\}$;
  FOR $i := 1$ **TO** n **DO BEGIN**
   IF $i = j$ **AND** $j \notin \{1, n\}$ **THEN**
   $c_i^{(k)} := c_i^{(k-1)} + \theta_2(x_k - c_i^{(k-1)})$
   ELSE
   $c_i^{(k)} := c_i^{(k-1)}$;
   $F_i^{(k)} = F_i^{(k-1)} + \theta_1(f(x_k) - F_i^{(k)})\mathbf{A}_i(x_k)$;
  END;
END.

The inputs $F_i^{(0)}$ for $i = 1, \ldots, n$ to the algorithm described above are small random numbers and $c_i^{(0)}$ for $i = 0, \ldots, n+1$ are distributed equidistantly on the domain and keeping the conditions $c_0^{(0)} = c_1^{(0)} = a$ and $c_n^{(0)} = c_{n+1}^{(0)} = b$.

The algorithm is independent on the shape of the basic functions. In its first part, it searches for the closest centroid to an actual incoming value x_k. The chosen centroid is then shifted unless it is a corner centroid $c_1^{(k-1)}$ or $c_n^{(k-1)}$. Then the delta rule formula is applied to each component of the F-transform but because of the influence of the basic function $\mathbf{A}_i$ weighting the formula only two neighboring components are modified.

4 Experiments

Let us consider a function f given by

$$f(x) = 2e^{(-40(x-0.5))} - 1 \tag{28}$$

on a domain $[a, b] = [0, 1]$. Function (28) has been sampled to get a training set $(x_k, f(x_k))$ at randomly chosen nodes x_k where $k = 1, \ldots, K = 100$. For simplicity, only one learning coefficient $\theta = \theta_1 = \theta_2$ has been considered.

Obviously, incremental learning (23) cannot reach the accuracy obtained in case when the components are given by original formula (14). The components given by the delta rule only tend to the optimal ones given by (14).

On the other hand, by resulting algorithm (27) which besides the components also modifies the distribution of the nodes c_i significantly better results have been achieved. It is impossible to measure the accuracy of the approximations by criterion (15)-(16) since particular errors are weighted by the basic functions which are different for both approximations. So, the *simple normed least square criterion* let

$$Error = 100\frac{1}{K}\sum_{k=1}^{K}\frac{(\hat{f}(x_k) - f(x_k))^2}{(\max f(x_k) - \min f(x_k))} \tag{29}$$

where $\hat{f}$ is the approximate output, have been used to measure the accuracy.

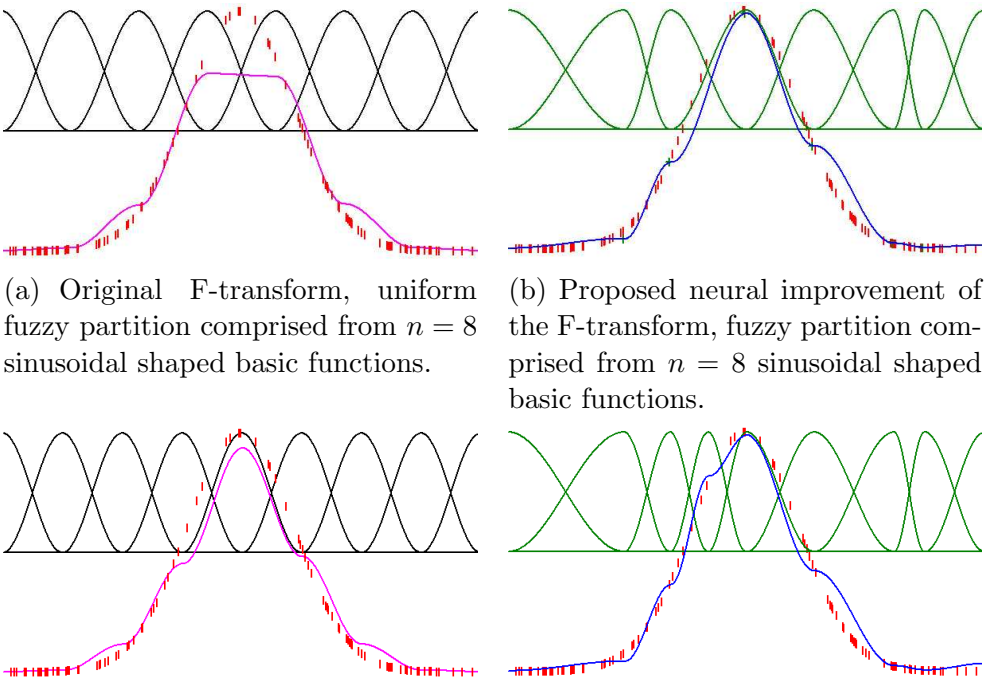
The proposed neural approach provided very often even better than the original batch formula. For instance, for $n = 10$ the original approach gives results with 0.523 error for the triangular shaped basic functions and 0.462 for the sinusoidal shaped basic functions. The neural approach gives always different errors depending on random generation of $F_i^{(0)}$ and the choice of θ but in general, oscillating between 0.457 and 0.966 depending on different θ coefficient or methods varying the learning coefficient.

The advantage of shifting centroid will play the more important role the more basic functions we use. For the case of $n = 7$ and sinusoidal shaped basic functions, in which the original batch formula gives error 1.227, the neural approach returns much better results, see Table 1.

Similar result were obtained for the other combinations of number n and θ. In general, it can be stated, that for smaller numbers of the basic functions the advantage of the neural network approach of shifting the centroids can compensate the the higher impreciseness caused by the incremental character of the algorithm.

Learning coefficient θ	**Error**
0.6	0.669
0.7	0.618
0.8	0.586

Table 1
Table of errors of the proposed neural approach, $n = 7$.



(a) Original F-transform, uniform fuzzy partition comprised from $n = 8$ sinusoidal shaped basic functions.

(b) Proposed neural improvement of the F-transform, fuzzy partition comprised from $n = 8$ sinusoidal shaped basic functions.

(c) Original F-transform, uniform fuzzy partition comprised from $n = 9$ sinusoidal shaped basic functions.

(d) Proposed neural improvement of the F-transform, fuzzy partition comprised from $n = 8$ sinusoidal shaped basic functions.

Fig. 2. Samples of function f given by (28) and its approximations by the inverse F-transform and by the proposed neural improvement of the fuzzy transform with learning coefficient $\theta = 0.8$.

Due to Lemma 6 we can use the F-transform technique to very fast and computationally simple numerical evaluation of the definite integral of f. Due to Lemma 7 we can analogously use the F-transform w.r.t. non-uniform fuzzy partitions so, even the incremental variant with centroids shifting can be considered.

Again, function (28) has been considered. Its definite integral is according to adaptive recursive Simpson's method implemented in MATLAB® [4] equal to -0.440 (with the *tolerance responsiveness toll* $= 10^{-6}$ certifying a very high accuracy, for details see [4]). The F-transform gives numerical definite integral equal to -0.455 for $n = 10$ sinusoidal shaped basic functions and equal to

-0.479 for $n = 7$ sinusoidal shaped basic functions. The neural improvement of the F-transform returns again always a bit different integral because of random start setting but of a very high preciseness, see Table 2.

Remark 12 *Note, that the numerical integral of function f has been computed by MATLAB using its analytical description (28) while the F-transform either in batch or neural incremental version used only a set of* 100 *random samples.*

Learning coefficient θ	**MATLAB**	**F-transform**	**Neural F-transform**
0.6	-0.440	-0.479	in (-0.459, -0.457)
0.7	-0.440	-0.479	in (-0.456, -0.455)
0.8	-0.440	-0.479	in (-0.454, -0.454)

Table 2
Numerical integrals by MATLAB, the fuzzy transform and the proposed neural approach to the fuzzy transform, $n = 7$.

Even for the case of $n = 10$ sinusoidal shaped basic functions, neural algorithm (27) gives again numerical integral based on Lemma 7 which is always closer to the value -0.440, than the integral computed with help of the original formula for the fuzzy transform and based on Lemma 6, see Table 3.

Learning coefficient θ	**MATLAB**	**F-transform**	**Neural F-transform**
0.6	-0.440	-0.455	in (-0.447, -0.445)
0.7	-0.440	-0.455	in (-0.442, -0.441)
0.8	-0.440	-0.455	in (-0.438, -0.437)

Table 3
Numerical integrals by MATLAB, the fuzzy transform and the proposed neural approach to the fuzzy transform, $n = 10$.

It can be stated, that the advantage of shifted centroids is even stronger in numerical integration no matter that the approximation was obtained incrementally.

5 Discussion

We have recalled the F-transform technique as a robust fuzzy approximation method as well as the RBF neural networks as appropriate tools for an approximation of functions. We have shown that the inverse F-transform mapping can be realized by a certain RBF neural network. The difference in both methods is in learning of parameters especially the components F_i.

We have introduced a computationally cheap delta rule based on the gradient

descent method which determines the components incrementally. It is an obvious computational advantage and for some (especially real-time) applications even a necessary condition [8].

Moreover, we can analogously adopt other approaches and techniques already developed in the neural network area. One of them, unsupervised k-means clustering for determining an appropriate distribution of nodes c_i for non-uniform fuzzy partition, has been implemented to increase the approximation accuracy of the model.

Better results could be obtained by an off-line cluster analysis and afterwards by applying the batch formula for the components. Incremental algorithms can hardly compete with batch ones in accuracy. On the other hand, from the computational complexity point of view it does not have to be always efficient and therefore only uniform fuzzy partitions have been used so far.

Experimental part of the paper justified the suggested approach. For lower numbers of basic functions, the proposed algorithm provided better results then the original batch F-transform due to the advantage of shifting the centroids. Moreover, due to Lemma 7 introduced in Section 2, definite integral of functions can be computed with help of the F-transform w.r.t non-uniform fuzzy partitions. Sections 4 demonstrated that the usefulness of the suggested approach which provides us with a powerful incremental tool for an accurate approximation of a function as well as for a precise and computationally cheap numerical integration of a sampled function.

Acknowledgements

The authors would like to express their thanks to anonymous referees for their valuable comments which helped to improve the paper.

References

[1] J. Buckley and Y. Hayashi, Fuzzy input-output controllers are universal approximators, Fuzzy Sets and Systems 58 (1993) 273-278.

[2] D. Coufal, Radial implicative fuzzy systems, in: Proceedings of the FUZZ-IEEE'05, Nevada, Reno, 2005 963-968.

[3] D. Coufal, Radial implicative fuzzy inference systems, PhD-thesis, University of Pardubice, 2003.

[4] W. Gander and W. Gautschi, Adaptive Quadrature - Revisited, BIT 40(1) (2000) 84-101.

[5] S. S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd Edition. Prentice Hall, Upper Saddle River, 1998.

[6] B. Kosko, Fuzzy systems as universal approximators, in: Proceedings of the FUZZ-IEEE'92, California, San-Diego, 1992 1153-1162.

[7] V. Kurková, Approximation of functions by neural networks, in: V. Mařík, O. Štěpánková and J. Lažanský, (Ed.), Artificial Intelligence vol. 4, Academia, Prague, in Czech 2003 254-273.

[8] E. Lughofer, Data-driven incremental learning of Takagi-Sugeno fuzzy models, PhD-Thesis, University Linz, Department of Knowledge-Based Mathematical Systems 2005.

[9] J. Moody and C. Darken, Fats adaptive k-means clustering: some empirical results, in: Proceedings of the IJCNN'90 Volume 2, California, San Diego, pp. 233-238.

[10] I. Perfilieva, Fuzzy approach to solution of differential equations with imprecise data: Application to reef growth problem, in: R.V. Demicco and G. J. Klir, (Ed.), Fuzzy Logic in Geology, Academic Press, Amsterdam, 2003 275-300.

[11] I. Perfilieva, Fuzzy transforms: Theory and applications, Fuzzy Sets and Systems 157 (2006) 993-1023.

[12] I. Perfilieva and R. Valášek, Fuzzy transforms in removing noise, in: B. Reusch, (Ed.), Computational Intelligence, Theory and Applications (Advances in Soft Computing), Springer-Verlag, Berlin, 2005 225-236.

[13] E.H. Ruspini, A new approach to clustering, Inform. and Control 15 (1969) 22-32.

[14] J. Šíma and R. Neruda, Theoretical Problems of Neural Networks, Matfyzpres, Prague, in Czech, 1996.

[15] M. Štěpnička and R. Valášek, Numerical solution of partial differential equations with help of fuzzy transform, in: Proceedings of the FUZZ-IEEE'05, Nevada, Reno, 2005 1104-1109.

[16] T. Takagi and M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Transactions on Systems, Man and Cybernetics 15 (1985) 116-132.