Comparison of a Neural Network Based on Fuzzy Flip-Flops and an MLP Robustness in Function Approximation

Robustness of Fuzzy Flip-Flop Based Neural Networks

R. Lovassy[1], L. T. Kóczy[2,3], L. Gál[2,4]

Abstract: In this paper two types of neural networks, namely the "traditional" *tansig* based neural networks and the multilayer perceptrons based on fuzzy flip-flops (F^3NN) trained by the Bacterial Memetic Algorithm with Modified Operator Execution Order (BMAM) are tested and compared on their robustness to test functions outliers. The robust design of the F^3NN is presented, and the best suitable fuzzy neuron type is emphasized. As our major motivation in these investigations was to construct a technology for the creation of real hardware MLPs and for this reason the fuzzy flip-flop based F^3NNs obviously offered much simpler and cheaper possibility for hardware implementation compared to a relatively complicated *tansig* type neural network.

Keywords: multilayer perceptrons based on fuzzy flip-flops, Bacterial Memetic Algorithm with Modified Operator Execution Order, fuzzy neural networks robustness to outliers

1. General remarks

Fast and good function approximation is a critical problem to solve in many applications ranging from system identification to pattern classification. It is also known, that the use of function approximation has an essential role in control system [26]. Investigations on the universal approximator properties of fuzzy systems and neural networks have been one of the most interesting aspects of these fields.

In the years 1990-92 papers by Dubois et al. [4], Kosko [16], furthermore Wang and Mendel [27], [28] proved almost simultaneously that fuzzy systems were universal approximators. In 1997 Klement et al. [15] argued that fuzzy systems could only be universal approximators in a rather restricted sense, because of the limits set by computational complexity. The authors also exemplified the main approaches to realize the idea of controlling real world processes by means of linguistic variables.

In the field of artificial neural networks mathematical function approximation using input-output data pairs from a set of examples is the object of study in different applications such as applied mathematics, and computer science. The paper of Hornik et al. [13] established that standard multilayer feedforward networks with a single hidden layer constituted a class of universal approximators. They gave a general investigation of the capabilities and properties of multilayer feedforward networks, without any suggestion to the number of hidden units needed to achieve a given accuracy of approximation. The function approximation capability of multilayered neural networks was studied in detail by Ciuca [2], Cybenko [3], Funahashi [5], Hecht-Nielsen [10] and Ito [14]. They proved that any continuous function could be approximated by a three-layered feedforward network with hidden sigmoid units and one linear output unit. The use of four-layered neural networks (with two sigmoid unit layers) as universal approximators of continuous functions was investigated by Funahashi [5], Girosi and Poggio [8] and Hecht-Nielsen [10]. Kurkova [17] studied also multilayer feedforward networks with sigmoid activation function approximation capabilities, analyzing also their computational complexity issues. Blum and Li showed [1] that four-layered feedforward networks with two hidden layers of semi linear units and with a single linear output unit were also universal approximators.

Hirota introduced the concept of fuzzy flip-flops (F^3) [11], as the extension of the binary counterparts. He started his investigations paralelly from the conjunctive and disjunctive minimal forms, thus defining set and reset type F^3s. As there exist infinite many different fuzzy logic operations (and this is true even for conjunctions, disjunctions and negation) obviously there are infinite many ways to define fuzzy flip-flops. These units obviously show a rather wide scope of behaviour. Hirota's group did various hardware implementations as well, investigating the behaviour of these fuzzy circuits [12]. A unified equation proposed in [24] led to a family of circuits with more symmetrical behaviour.

Fuzzy Flip-Flop based Neural Networks (F^3NN) were defined and their learning algorithm was proposed in [21], furthermore giving in this way a possible starting point for many researchers in the field of hybridization of fuzzy logic with artificial neural networks, and in function approximation for learning control.

This paper investigates the problem of stability analysis of F^3NNs. The sensitivity of fuzzy neural networks to the fuzzy neuron type is evaluated as approximator of different test functions. Performance of the F^3NNs and a simulated MLP is compared in order to evaluate robustness on the various test functions with respect to the outliers. When developing a robust function approximator, an important point is to maintain stability even in the presence of unexpected outliers or external disturbances applied to the network.

This paper is organized as follows. Section 2 summarizes some of our previous works and offers fundamental results for providing our results. In Section 3 the design of the F^3NN is presented. In Section 4 illustrative examples are given to study stability and to evaluate the robustness of simulated and F^3 based neural networks. It turned out that the Łukasiewicz type F^3NNs had a much more robust behaviour than the standard *tansig* based simulator. Finally, some concluding remarks are given.

2. Fuzzy neuron activation functions, review of previous works

One of the common forms of the transfer function used in the construction of neural networks is the sigmoid activation function. The starting point of our own investigations was the fuzzy flip-flops [11]. In our previous papers [21], [22] the unified fuzzy J-K flip-flop based on various norms combined with the standard negation, was analyzed in order to investigate, whether and to what degree they present more or less sigmoid (s-shaped) $J \rightarrow Q_{out}$ characteristics in particular cases, when $K = 1 - Q$ (unified fuzzy J-K flip-flop with feedback), $K = 1 - J$ (new fuzzy D flip-flop derived from the unified fuzzy J-K one) with fixed values of Q. We conducted extensive investigations and found that the $J \rightarrow Q_{out}$ transfer characteristics of fuzzy J-K flip-flops with feedback based on Łukasiewicz, Yager, Dombi and Hamacher norms, further the $D \rightarrow Q_{out}$ characteristics of fuzzy D flip-flops of Łukasiewicz, Yager and Hamacher operations show quasi sigmoid curvature for selected Q and fuzzy operations parameter value pairs.

A new pair of conjunction and disjunction, the triangular t-norm and t-conorm was introduced in [7]. These new fuzzy operations, furthermore the Łukasiewicz and Dombi norms combined with the standard negation were applied in a practical problem, namely, they were proposed as suitable triangular norms for defining fuzzy flip-flop based neurons. Table 1 shows the above mentioned two widely used fuzzy operations, and the new triangular t-norm and t-conorm expressions.

Table 1. Some selected t-norms and t-conorms

Fuzzy Operation	t-norm $i\ (x, y)$	t-conorm $u\ (x, y)$
Łukasiewicz	$\max(0, x+y-1)$	$\min(1, x+y)$
Dombi	$\dfrac{1}{1+\left[(1/x-1)^{\alpha}+(1/y-1)^{\alpha}\right]^{1/\alpha}}$	$\dfrac{1}{1+\left[(1/x-1)^{-\alpha}+(1/y-1)^{-\alpha}\right]^{-1/\alpha}}$
Trigonometric	$\dfrac{2}{\pi}\cdot\arcsin\left(\sin\left(x\dfrac{\pi}{2}\right)\cdot\sin\left(y\dfrac{\pi}{2}\right)\right)$	$\dfrac{2}{\pi}\cdot\arccos\left(\cos\left(x\dfrac{\pi}{2}\right)\cdot\cos\left(y\dfrac{\pi}{2}\right)\right)$

The sigmoid activation functions are given by the expressions (1)-(3), which are the characteristic equations of the fuzzy J-K flip-flops with feedback based on Dombi and trigonometric norms, furthermore the fuzzy D flip-flop (derived from the J-K one) based on Łukasiewicz operations. For simplicity the t-norm is denoted by i (intersection), and the t-conorm by u (union), where the subscripts refer to the initial of the type of the norm: e.g., in case of the Dombi norms: $i_D\ (x,y) = x\ i_D\ y$ and $u_D\ (x,y) = x\ u_D\ y$.

The unified $Q(t+1)$ definition based on Dombi norms is

$$Q(t+1) = \left(J\ u_D\ Q\right)\ i_D \left(J\ u_D\ Q\right)\ i_D\ \left(Q\ u_D \left(1-Q\right)\right) = \tag{1}$$

$$= 1/\left(1+\left(\left(2^{1/\alpha}\left(\left(\left(\left(-1+\frac{1}{J}\right)^{-\alpha}+\left(-1+\frac{1}{Q}\right)^{-\alpha}\right)^{-1/\alpha}\right)^{\alpha}\right)^{1/\alpha}\right)^{\alpha}+\left(\left(\left(-1+\frac{1}{1-Q}\right)^{-\alpha}+\left(-1+\frac{1}{Q}\right)^{-\alpha}\right)^{-1/\alpha}\right)^{\alpha}\right)^{1/\alpha}\right)$$

The characteristic equation of the fuzzy J-K flip-flop with feedback based on trigonometric norms is

$$Q(t+1) = \left(J\ u_T\ Q\right)\ i_T \left(J\ u_T\ Q\right)\ i_T\ \left(Q\ u_T \left(1-Q\right)\right) = \tag{2}$$

$$= 2\,\mathrm{a}\sin\left(\sin\left(\mathrm{a}\cos\left(\frac{1}{4}\cos^2\pi^2\left(1-Q\right)Q\right)\right)\sin\left(\mathrm{a}\sin\left(\sin\left(\mathrm{a}\cos\left(\frac{1}{4}\cos^2\pi^2 JQ\right)\right)^2\right)\right)\right)/\,\pi$$

The next equation is defined by proposing the Łukasiewicz norms in the maxterm form of the new fuzzy D flip-flop formulae

$$Q(t+1) = \left(D\ u_L\ D\right)\ i_L\ \left(D\ u_L\ Q\right)\ i_L\ \left(D\ u_L\ \left(1-Q\right)\right) = \tag{3}$$

$$= \max\left(0,\ -1+\max\left(0,\ -1+\min\left(1,\ 2D\right)+\min\left(1,\ D+Q\right)\right)+\min\left(1,\ 1+D-Q\right)\right)$$

These sigmoid function generators can be implemented rather simply in hardware because they require only primitive mathematical operations, and therefore, they are inexpensive in terms of memory and computational requirements. The fuzzy flip-flop based neurons simple internal structure was given in [20]. The clocked fuzzy J-K and D flip-flop neuron circuits were built up using hardware blocks in order to realize various t-norms, t-conorms and fuzzy negations [29]. Since t-norms and t-conorms are functions from the unit square into the unit interval, the fuzzy J-K flip-flop block diagram differs from the binary J-K flip-flop structure. The input J is driven by a synchronized clock pulse in the sample-and- hold (S/H) circuit. FPGA technology was applied for hardware implementation of fuzzy D flip-flop neurons based on Łukasiewicz norms published in [19]. Simulation results, the hardware resources, number of logical gates, logic levels required, I/O used and timing delay consumed by Łukasiewicz operations and overall fuzzy D flip-flop neurons were presented. The proposed circuit was built up using core blocks which realize Łukasiewicz t-norm and t-conorm. The implementation was an 8 bit data width architecture and additionally two bit as selection for operations. These blocks can also be chained to form more complex structures such as Fuzzy Neural Network.

3. Multilayer perceptrons based on fuzzy flip-flops

The most commonly used network architecture is the multilayer feedforward network. The network topology consists of the connection between neurons, the input and the output structure. The neuron output signal is computed as the weighted sum of the input signals transformed by the transfer function. The deployment of the Levenberg-Marquardt algorithm (LM) [18], [23] and the Bacterial Memetic Algorithm with Modified Operator Execution Order Algorithm (BMAM) [6], was proposed and applied for F^3NN variables optimization and training. The four-layered F^3NN with hidden nodes defined by fuzzy flip-flop neurons (as sigmoid function generators) and a linear output node presented good function approximation properties. The network architecture built up from fuzzy D neurons (derived from the J-K one) is shown in Figure 1. In this approach the weighted input values are connected to inputs *J* and *K* of the new fuzzy flip-flop neuron based on a pair of t-norm and t-conorm, having quasi sigmoid transfer characteristics. The output signal is then computed as the weighted sum of the input signals transformed by the transfer function.

Input Layer Hidden Layer1 Hidden Layer2 Output Layer

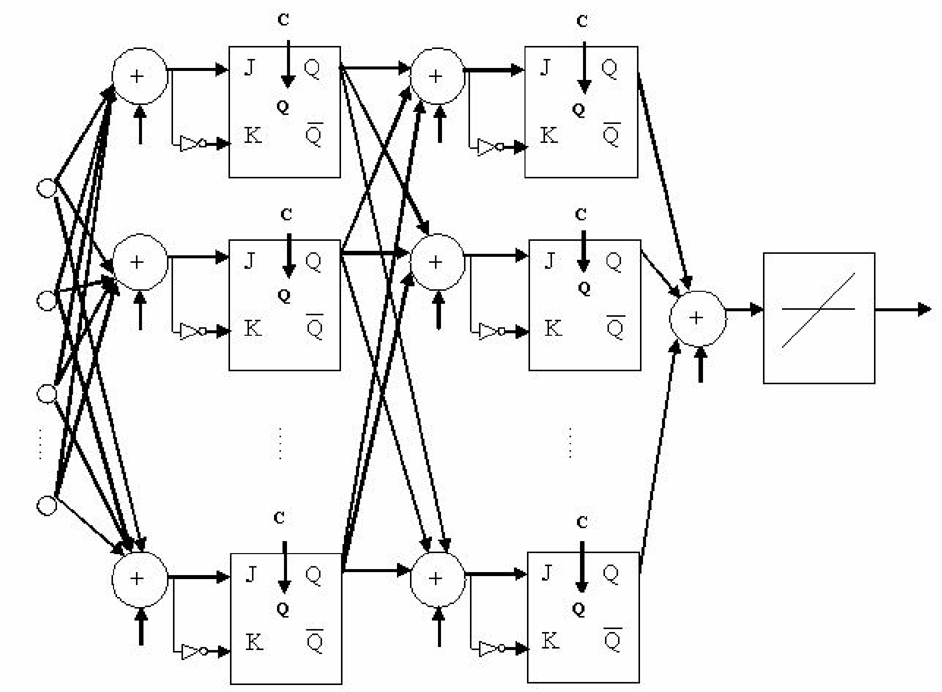

Figure 1. Fuzzy flip-flop based neural network

The most suitable types of F^3 neurons for constructing fuzzy neural networks were the Dombi and trigonometric type fuzzy J-K flip-flop neurons and Łukasiewicz type fuzzy D flip-flop neurons. Our investigations included comparative experiments using the

MATLAB Neural Networks Toolbox applying *tansig* based characteristic functions for the simulated neurons that we always used as reference [21], [22].

Our experiments made it obvious that the F^3NNs based on even the best suitable fuzzy operations always led to greater Mean Square of Error – MSE when the "idealistic" *tansig* based simulator had exactly the same structure and number of neurons. However the best type F^3NNs produced MSEs in the same order of magnitude, thus we concluded that F^3NNs presented a realistic alternative to simulated MLPs.

As our major motivation in these investigations was to construct a technology for the creation of real hardware MLPs and F^3 based NNs obviously offered much simpler and cheaper possibility for hardware implementation compared to a rather complicated approximated *tansig* type neural network.

In the next section we will show that the fuzzy flip-flop based neural networks with a given structure and neuron type posses the ability to approximate robust the proposed test functions with and without outliers. It will guarantee that a certain level of approximation accuracy is possible, this is what the stability and performance of our systems typically depends on. Illustrative examples are presented, in order to demonstrate the success of this work in terms of the function approximation capability and the robustness of the proposed fuzzy neural network.

4. Comparison of the robustness F^3NNs and *tansig* based neural networks

In the next, the robustness of three different types of F^3NN and the *tansig* based neural networks is compared from the various test function approximation goodness points of view. The Dombi and trigonometric type fuzzy J-K flip-flop neurons, furthermore the Łukasiewicz type fuzzy D flip-flop neurons based fuzzy neural networks (DJKFNN, TJKFNN, LDFNN) and the simulated neural networks are studied. Here the robustness of the models can be defined in such a way that the set of selected models should be robust in the sense that they are indifferent to the hidden layers neuron numbers and also they do not overfit in case of outliers.

The one and two dimensional test functions used for NNs robustness comparison will be described in the next.

4.1. Data description

Modified *pH* benchmark problem. Originally, the one dimensional *pH* problem [25] is the task to approximate the following function:

$$f_1(x) = pH^{-1} = -\frac{\log\left(\sqrt{\frac{y^2}{4}+10^{-14}}-\frac{y}{2}\right)+6}{26}; \tag{4}$$

$$y = 2\cdot10^{-3}\cdot x - 10^{-3};\ x \in [0,\ 0.75]$$

The pattern set consists of 101 datasets, with very uneven distribution:

Domain: [0.034914, 0.743401]

Range: [0.0001, 1.0000]

No data in (0.19, 0.38); (0.39, 0.59); etc.

In our approach, we modified the pattern set by generating intentionally four outlier points at the beginning and at the end of the domain.

Superposition of two sine waves. A combination of two sine wave forms with different period lengths described with the equation

$$f_2(x) = \frac{\sin(c_1 \cdot x) \cdot \sin(c_2 \cdot x)}{2} + 0.5 \tag{5}$$

The values of constants c_1 and c_2 were selected to produce a frequency proportion of the two components 1:0.35. The test function is represented by 100 input/output data sets.

Two two - input trigonometric functions. The two dimensional test functions are

$$f_3(x_1, x_2) = \frac{\sin(c_1 \cdot x_1)^5 \cdot \sin(c_2 \cdot x_2)^3}{2} + 0.5 \tag{6}$$

and

$$f_4(x_1, x_2) = \left(\cos\left(\arctan^*\left(\frac{x_1}{x_2} \right) \right) + 1 \right) \cdot e^{-\frac{r^2}{50}} \cdot \sin^5\left(\frac{r}{10} \right) \tag{7}$$

$r = \sqrt{x_1^2 + x_2^2}$, $x_1, x_2 \in [-20, 20]$; $\arctan^*$ is the four-quadrant inverse tangent function. The two-input functions are both represented by 1600 input/output data sets.

4.2. Training algorithm

The Bacterial Memetic Algorithm with Modified Operator Execution Order Algorithm [6] was applied for network training. In this application according to the network size a population was initialized. During the algorithm each individual is selected one by one, and 30 generations of 5 individuals with 5 clones are chosen to obtain the best fitting variable values, with the lowest performance. The same part or parts are selected randomly from the clones and mutated. The LM method nested into the evolutionary algorithm is applied for 5 times for each clone. Several tests have shown that it is enough to run 3 to 5 of LM iterations per mutation to improve the performance of the whole algorithm. The best clone is selected and transferred with its all parts to the other clones. Choosing-mutation-LM-selection-transfer cycle is repeated until all the parts are mutated, improved and tested. The best individual is remaining in the population all other clones are deleted. This procedure is repeated until all the individuals are taking part in the modified bacterial mutation. As a second main step, the LM is applied 7 times for each individual executing several LM cycles during the bacterial mutation after each mutation step. In the applied algorithm the gene transfer operation is completely excluded.

4.3. Results and discussion

The outlier points introduced in our data sets are numerically distant from the rest of data. Usually, this clear deviations from other data sample indicates that they do not fit the model under study, can occur by change in any distribution, input/output data pairs with very uneven distribution, or an error in measurement.

Next it is tested how well the simulated and the F^3 based neural networks function approximation handle the test functions outlier points. This section analyses the function approximation performance of these types of network in terms of accuracy, hidden layer neuron number, stability and robustness.

During the simulations the *tansig* based neural network is compared with all combinations of fuzzy J-K and D type F^3NNs which had been covered with all three fuzzy operation pairs to approximate the test functions listed in subsection 4.1, eq. 4-7.

Table 2. MSE median values, one dimensional test functions

	1D-1			
NN type	Neural network size			
	1-2-2-1	1-3-3-1	1-4-4-1	1-6-6-1
tansig	$3.161x10^{-3}$	$2.996x10^{-3}$	$2.243x10^{-3}$	$1.242x10^{-3}$
DJKFNN	$3.171x10^{-3}$	$3.154x10^{-3}$	$3.080x10^{-3}$	$3.019x10^{-3}$
TrigJKFNN	$3.305x10^{-3}$	$3.222x10^{-3}$	$3.062x10^{-3}$	$3.029x10^{-3}$
LDFNN	$3.225x10^{-3}$	$3.067x10^{-3}$	$2.879x10^{-3}$	$2.298x10^{-3}$
	1D-2			
NN type	Neural network size			
	1-2-2-1	1-4-4-1	1-6-6-1	1-8-8-1
tansig	$7.112x10^{-3}$	$7.394x10^{-7}$	$2.141x10^{-7}$	$6.032x10^{-9}$
DJKFNN	$2.464x10^{-2}$	$1.563x10^{-3}$	$9.665x10^{-4}$	$3.764x10^{-6}$
TrigJKFNN	$6.626x10^{-3}$	$1.798x10^{-3}$	$3.025x10^{-4}$	$2.498x10^{-6}$
LDFNN	*$6.888x10^{-3}$*	*$2.043x10^{-3}$*	*$1.969x10^{-3}$*	*$6.858x10^{-4}$*

Table 2 presents the 5 runs average approximation goodness, by indicating the median MSEs (mean squared error) of various networks training values. During evaluation we compared the median MSE values, considering them as the most important indicators of trainability. The median is a robust estimate of the centre of a data sample, since outliers have little effect on it. The median represents robust statistic, while the mean does not. In the next, we study how the network function approximation goodness depends on the number of hidden layers neurons, furthermore we show how much the simulator is better than the emulated hardware, and how many more F^3 neurons should be applied in order to achieve similar training properties. We denoted by 1D-1 the modified *pH* problem, by 1D-2 the one dimensional sine waveforms, by 2D-1 the $f_3(x_1, x_2)$, and finally by 2D-2 the $f_4(x_1, x_2)$ test functions.

Comparing the simulation results we concluded that increasing the neuron numbers in the hidden layers the networks function approximation accuracy had been increased. This research was concluded as an essentially successful one, especially so because one of the reference problem (modified *pH* problem) showed that F^3NNs had a much more stabile behaviour than the *tansig* based simulator: "Rogue" data (outliers) caused overfitting with the MATLAB simulator while F^3NNs kept the approximated function in a relatively smooth form, (see Figure 2.) even so the different networks function approximation goodness are in the same order of magnitude. In this case the simulator it turned out to be the same like the emulated hardware, furthermore the same number of F^3 neurons should be applied in order to achieve similar training properties.

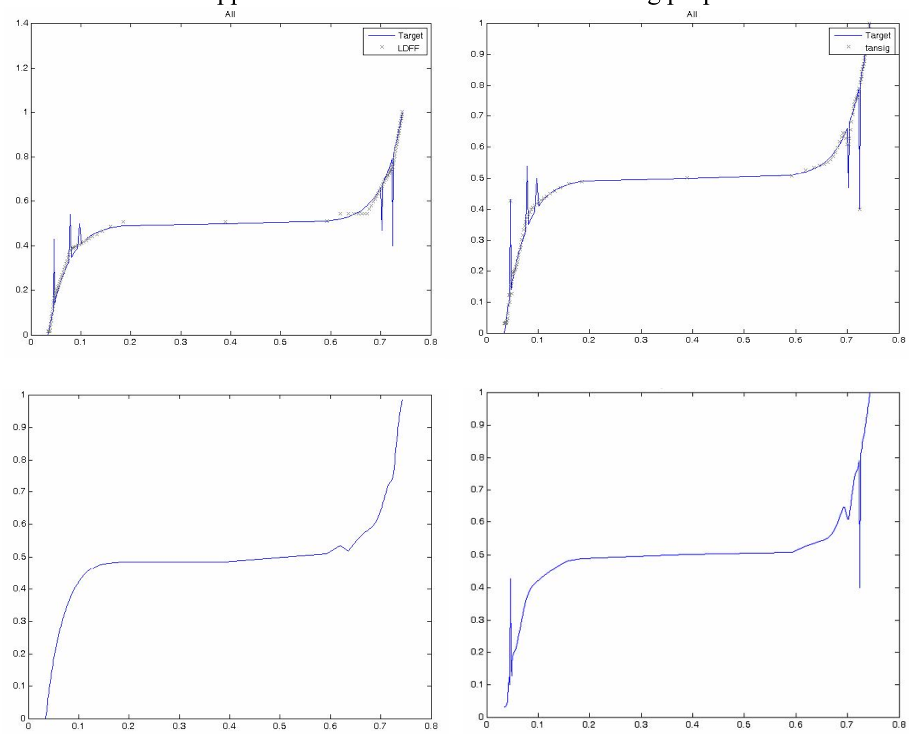

Figure 2. Łukasiewicz type F^3NN and tansig based neural network pH benchmark problem, 1-6-6-1 network size

When instead of the benchmark data set a more complex wave form (eq. 5) was used as test function the training MSE median values in case of 2-2 neurons in the hidden layers present almost the same values. Extending the network size, we have to apply approximately twice more F^3 neurons in order to obtain the same function approximation goodness as the *tansig* based neural network. The exception is the Łukasiewicz type fuzzy D flip-flop neuron based neural network, where several times more neurons have to be used. The simulation results ensure that this type of F^3NNs possesses a stabile property – the MSE median values are very close to each other. Figure 3 shows the simulation results.

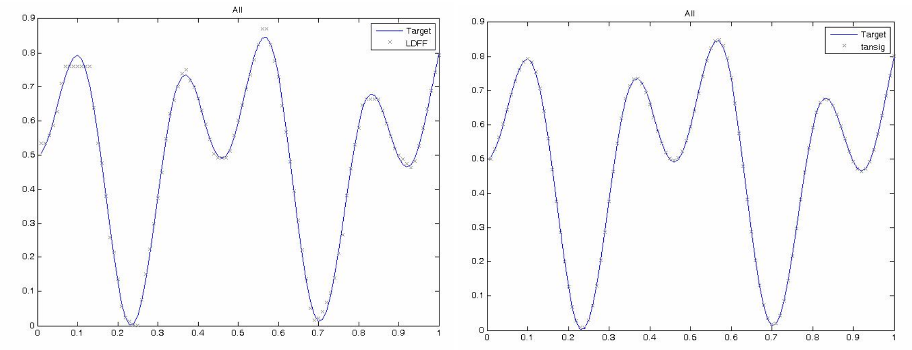

Figure 3. Łukasiewicz type F^3NN and tansig based neural network two sine waves, 1-4-4-1 network size

Table 3 summarizes the simulation results in case of the two dimensional test functions (eqs. 6 and 7).

Table 3. MSE median values, two dimensional test functions

	2D-1			
NN type	Neural network size			
	1-4-4-1	1-8-8-1	1-15-15-1	1-20-20-1
tansig	1.782×10^{-3}	9.077×10^{-5}	2.228×10^{-6}	9.071×10^{-7}
DJKFNN	5.519×10^{-3}	6.785×10^{-5}	2.131×10^{-5}	8.754×10^{-6}
TrigJKFNN	1.664×10^{-3}	1.557×10^{-4}	7.921×10^{-5}	1.033×10^{-5}
LDFNN	*4.038×10^{-3}*	*1.496×10^{-3}*	*9.672×10^{-4}*	*7.481×10^{-4}*
	2D-2			
NN type	Neural network size			
	1-4-4-1	1-8-8-1	1-15-15-1	1-20-20-1
tansig	3.735×10^{-7}	4.344×10^{-8}	6.229×10^{-9}	3.889×10^{-9}
DJKFNN	7.097×10^{-6}	1.501×10^{-6}	2.672×10^{-7}	5.732×10^{-8}
TrigJKFNN	1.241×10^{-5}	1.234×10^{-6}	5.343×10^{-7}	1.879×10^{-7}
LDFNN	*4.246×10^{-5}*	*1.714×10^{-5}*	*3.702×10^{-6}*	*1.046×10^{-6}*

Independently from the input function complexity, in these cases also the function approximation accuracy is directly proportional to the neuron numbers. Only the Łukasiewicz type F^3NNs are robust in the sense that that the network output does not change significantly under the influence of changes in the hidden layers. These types of F^3NNs are more robust to variations of neuron numbers. Figure 4 shows the function approximation goodness of Łukasiewicz type F^3NNs and the *tansig* based neural network in case of two dimensional test functions (eqs. 6, 7).

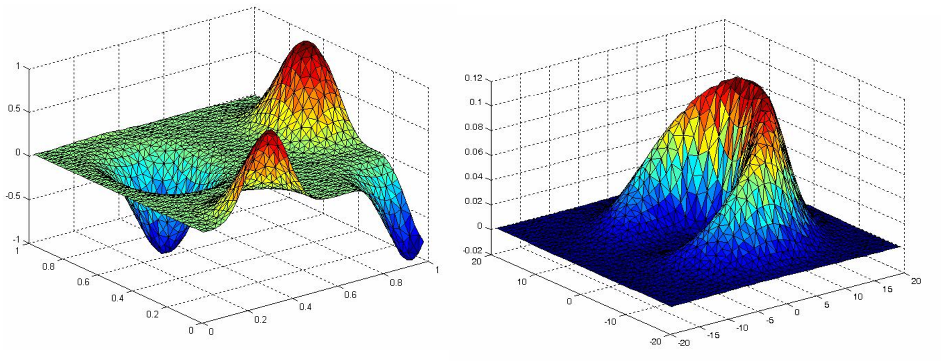

Figure 4. Łukasiewicz type F^3NN and tansig based neural network two dimensional test functions, 1-15-15-1 network size

Conclusion

In this paper we proposed the design of a robust function approximator for a general class of multi-input and one output systems. One of our test data sets contained a number of outlaying items. Taking both theory and practice of the Evolutionary Algorithms (EAs) [9] into account, it is safe to anticipate that the solutions obtained by the application of a conventional evolutionary optimization algorithm will be the most performing, not the most robust, that is the reason why we proposed the BMAM algorithm for the NNs training. In addition, selecting the most relevant rules (maxim of relation) the Łukasiewicz type F^3NNs yield more compact and robust systems. This kind of neural networks are robust enough for hardware implementation with general purpose (unknown application) and are more suitable to avoid overfitting than customary (e.g. *tansig* based) neural networks.

Further research will be dedicated to the mathematical analyses of why Łukasiewicz type F^3NNs present a better stability in terms of the number of neurons than the reference simulator. Next steps would be to prove stability and evaluate the robustness of these approaches. The idea is to avoid unnecessary retraining, and as consequence, we would like also to avoid any particular overtraining problem. Additionally, besides the often mentioned advantage of complexity reduction when using fuzzy logic in practical applications we found robustness being also present.

References

[1] Blum, E. K., Li, L. K.: *Approximation theory and feedfordward neural networks* Neural Networks, 4(4), 1991, pp. 511-515.

[2] Ciuca, I., Ware, J. A.: *Layered Neural Networks as Universal Approximators*, Fuzzy Days Dortmund, Germany, 1997, pp. 411-415.

[3] Cybenko, G.: *Approximation by superposition of sigmoidal functions*, Math. Contr. Signals. Syst. 2, 1989, pp. 303-314.

[4] Dubois, D., Grabisch, M., Prade, H.: *Gradual rules and the approximation of functions*, in Proc. of the 2nd International Fuzzy Systems Association Congress Iizuka, Japan, 1992, pp. 629-632.

[5] Funahashi, K. I.: *On the approximate realization of continuous mapping by neural networks*, Neural Networks, 2, 1989, pp. 183-192.

[6] Gál, L., Botzheim, J., Kóczy, L. T.: *Improvements to the Bacterial Memetic Algorithm used for Fuzzy Rule Base Extraction*, in Proc. of Computational Intelligence for Measurement Systems and Applications, CIMSA, Istanbul, Turkey, 2008, pp. 38-43.

[7] Gál, L., Lovassy, R., Kóczy, L. T.: *Function Approximation Performance of Fuzzy Neural Networks Based on Frequently Used Fuzzy Operations and a Pair of New Trigonometric Norms*, Proc. of IEEE World Congress on Computational Intelligence, WCCI 2010, Barcelona, Spain, pp.1514-1521.

[8] Girosi, F., Poggio, T.: *Representation properties of network: Kolmogorov's theorem is irrelevant*, Neural Computation 1, 1989, pp. 456-469.

[9] Goldberg, D. E.: *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison-Wesley, Reading (1989).

[10] Hecht-Nielsen, R.: *Theory of the backpropagation neural network*, in Proc. of the Neural Networks, IJCNN, International Joint Conference on Neural Networks, 1989, pp. 593-605.

[11] Hirota, K., Ozawa, K.: *Concept of fuzzy flip-flop*, Preprints of 2nd IFSA Congress, Tokyo, 1987, pp. 556-559.

[12] Hirota, K., Ozawa, K.: *Fuzzy flip-flop as a basis of fuzzy memory modules*, In: M. M. Gupta et al., eds., Fuzzy Computing. Theory, Hardware and Applications, North Holland, Amsterdam, 1988, pp. 173-183.

[13] Hornik, K., Stinchcombe, M., White, H.: *Multilayer Feedforward Networks are Universal Approximators*, Neural Networks, 2, 1989, pp. 359-366.

[14] Ito, Y.: *Approximation Capability of Layered Neural Networks with Sigmoid Units on Two Layers*, Neural Computation, 6(6), 1994, pp. 1233-1243.

[15] Klement, E. P., Kóczy, L. T., Moser, B.: *Are fuzzy systems universal approximators?* International Journal of General Systems 28 (2–3), 1999, pp. 259 – 282.

[16] Kosko, B.: *Fuzzy Systems are Universal Approximators*, in Proc. of the IEEE International Conference on Fuzzy Systems, San Diego, CA, 1992, pp.1153-1162.

[17] Kurkova, V.: *Kolmogorov's theorem and multilayer neural networks*, Neural Networks, 5, 1992, pp. 501-506.

[18] Levenberg, K.: *A method for the solution of certain non-linear problems in least squares*, Quart Appl. Math. 2(2), 1944, pp. 164-168.
[19] Lovassy, R., Zavala, A. H., Gál, L., Nieto, O. C., Kóczy, L. T., Batyrshin, I.: *Hardware Implementation of Fuzzy D Flip-Flop Neurons Based on Łukasiewicz Norms*, in Proc. of the 9th WSEAS Int. Conference on Applied Computer and Applied Computational Science, Penang, Malaysia, 2010, pp.196-201.
[20] Lovassy, R., Kóczy, L. T., Gál, L.: *Optimization in Fuzzy Flip-Flop Neural Networks*, Computational Intelligence in Engineering; Studies in Computational Intelligence (eds.: I. J. Rudas, J. Fodor, J. Kacprzyk), Springer, Vol. 313, 2010, pp.337-348.
[21] Lovassy, R., Kóczy, L. T., Gál, L.: *Function Approximation Capability of a Novel Fuzzy Flip-Flop Based Neural Network*, Proc. of International Joint Conference on Neural Networks, IJCNN 2009, Atlanta, USA, 2009, pp. 1900-1907.
[22] Lovassy, R., Kóczy, L. T., Gál, L.: *Applicability of Fuzzy Flip-Flops in the Implementation of Neural Networks*; Proc. of CINTI 2008, 9th International Symposium of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, 2008, pp. 333-344.
[23] Marquardt, D.: *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*, SIAM J. Appl. Math. 11, 1963, pp. 431-441.
[24] Ozawa, K., Hirota, K., Kóczy, L. T.: *Fuzzy flip-flop*, In: M. J. Patyra, D. M. Mlynek, eds., Fuzzy Logic. Implementation and Applications, Wiley, Chichester, 1996, pp. 197-236.
[25] Ruano, A E., Cabrita, C., Oliveira, J. V., Tikk, D., Kóczy, L. T.: *Supervised training algorithms for B-spline neural networks and fuzzy systems*, 9th IFSA World Congress and 20th NAFIPS International Conference (IFSA NAFIPS) Vancouver, 2001, pp. 2830-2835.
[26] Spooner, J. T., Maggiore, M., Ordonez, R., Passino, K. M.: *Stable adaptive control and estimation for nonlinear systems: neural and fuzzy approximator techniques*, John Wiley and Sons, 2002.
[27] Wang, L. X.: *Fuzzy systems are universal approximators*, in Proc. of the IEEE International Conference on Fuzzy Systems, San Diego, CA, 1992, pp. 1163-1169.
[28] Wang, L. X., Mendel, J. M.: *Fuzzy basis functions, universal approximations and orthogonal least-squares learning*, IEEE Transactions on Neural Nets, 3, 1992, pp. 807-814.
[29] Zavala, A. H., Nieto, O. C., Batyrshin, I., Vargas, L. V.: *VLSI Implementation of a Module for Realization of Basic t-norms on Fuzzy Hardware*, Proc. of FUZZ-IEEE 2009, IEEE Conference on Fuzzy Systems, Jeju Island, Korea, August 20-24, 2009, pp. 655-659.