Human factor and computational intelligence limitations in resilient control systems

Bogdan M. Wilamowski
Auburn University

***Abstract* - Humans are very capable of solving many scientific and engineering problems, but during the solution process they have a tendency to make mistakes. For example, humans without computer aided tools, would not be able to design VLSI chips larger than 100 transistors. This imperfection of humans make them very unreliable elements in resilient control systems. There is a tendency of replacing humans with computers using artificial intelligence, expert systems, or methods of computational intelligence. The methods of computational intelligence can be most successful but they have to be used with great care. Limitations of fuzzy and neural networks are presented and it is shown how to avoid these limitations so resilient control systems can be developed. It turns out that often popular training algorithms are not capable of tuning neural networks to proper accuracy without losing generalization abilities. As a consequence, such system of computational intelligence may not work properly for cases which were not used in training. The comparison of different neural network architectures follows and also it is shown how to develop and train close to optimal topologies, so resilient control systems can be developed.**

I. INTRODUCTION

Humans are capable not only of solving many scientific problems, but also in the process they are making mistakes. For example, even very good students are not able to solve long problems requiring complex calculations without making errors in the process. These mistakes can be eventually corrected but if a long problem with many calculations is given to a group of students then most likely initially each student would produce a different answer. In the early stage of development of VLSI chips, required mask was manually cut on a foil and when the number of transistors in the circuits was larger than 50, then chances of successful production of a chip was very slim. Even after several stages of corrections a chance of success was not that great because by correcting some mistakes humans generated other errors. As a consequence, without computer aided tools humans were not able to design chips larger than 100 transistors. Also, with usage of computers humans were the weakest link in the chip design process where many different software were used. For example, when humans were required to manually enter data obtained from one software to another software then again errors were generated. The problem was partially solved when the PERL language [1] was used to automatically convert data between different software. Eventually the human role was limited only to specific requirements and the rest of the process was done automatically by so called "silicon compilers". By taking humans out of the design process we can now successfully design chips with over 10 billion transistors. Humans are also the weakest links in communication control processes. Please notice that an airplane may fly smoothly only when it is on autopilot. The manual control of the airplane is used only when it is necessary, when autopilot is not designed for given flying conditions (like airborne and landing). With the help of expert systems and computational intelligence the role of humans are being steadily eliminated. There are, however, several areas where methods of computer intelligence have difficulty to replace humans. It is common knowledge that computers are much superior to humans in number crunching, but it is believed that humans are superior to computers in areas of image recognition.

Section II demonstrates that methods of computation intelligence can be much more efficient than humans in image recognition. Section III shows that humans are not as effective as methods of computational intelligence for data mining, to extract and to classify information from overwhelming data sets. Both sections II and III indicates superiority of method computational intelligence over humans, but these methods have their limitations and have to be used with caution. Section IV describes limitations of fuzzy systems and sections V and VI describe limitations of neural networks. Section VII introduces a recently developed efficient algorithm for neural network training.

II. COMPARISON OF HUMAN APPROACHES AND METHODS OF COMPUTATIONAL INTELLIGENCE FOR CHARACTER RECOGNITIONS

Let us compare humans and computers in recognition of noisy characters. In this experiment, 256 characters used in CGI displays were selected for the experiment. The CGI characters are organized in 8*7 arrays.

The experiment of character recognition was carried on in the following scheme. Original characters (left columns on Fig. 1 were distorted by six levels of noise). Then the neural network with the architecture of Fig. 2 was used to recognize

characters (left side of the architecture) and to retrieve the original character (right side of the architecture).

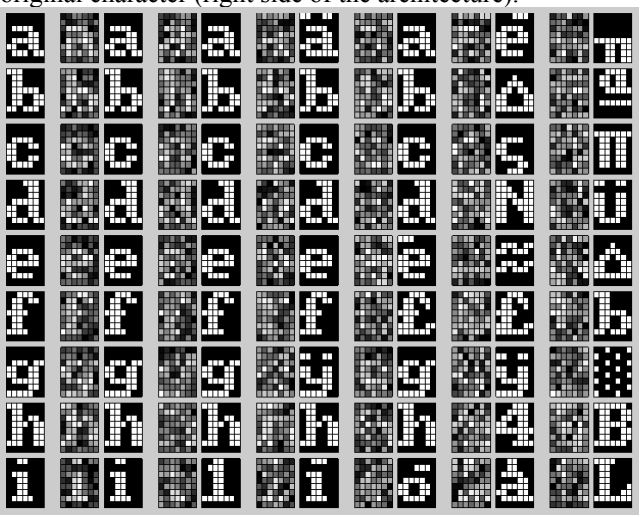


Fig. 1. Result of pattern retrieval for neural network architecture of Fig. 2 trained to recognize 256 patterns.

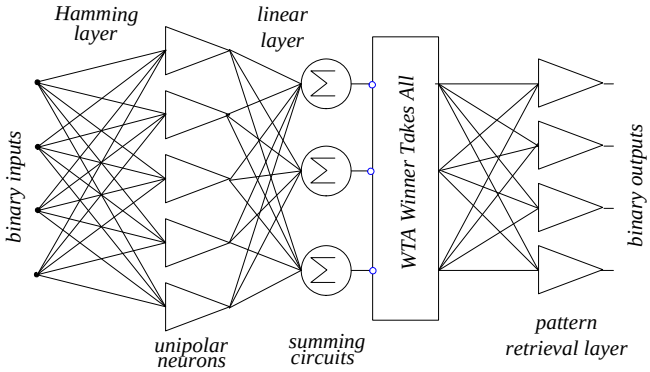


Fig 2. Artificial Neural Network architecture for classification of patterns.

The neural network of Fig. 2 is an enhanced version of the conterpropagation network proposed by Hecht-Nielsen [42] with Kohonen/Hamming input layer and with Grosberg/linear output layer. First layer computes Euclidean distances between input pattern and stored patterns. If inputs are binaries, for example **X**=[1, -1, 1, -1, -1] then the maximum value of *net*

$$net = \sum_{i=1}^{n} x_i w_i = \mathbf{X}\mathbf{W}^T = n = 5 \tag{1}$$

is when weights **W**=[1, -1, 1, -1, -1] are identical to the input pattern **X.** If input signal is different, for example **X**=[1, 1, 1, -1, -1] then

$$net = \sum_{i=1}^{n} x_i w_i = \mathbf{X}\mathbf{W}^T = 3 \tag{2}$$

or

$$net = n - 2 \cdot HD(\mathbf{X}, \mathbf{W}) = 3 \tag{3}$$

Winning "neuron" is the highest value of net and it is with the minimum of Euclidean distance from the stored pattern. This specific architecture of artificial neural network was selected because it is not computationally intensive because of binary patterns to find the Hamming distance the weight-signal multiplication process can be replaced by a subtraction:

$$\mathbf{D} = abs(\mathbf{X} - \mathbf{W}) \tag{4}$$

and calculate the sum of all elements of **D.** Then

$$net = n - 2 \cdot sum \tag{5}$$

Such a neural network can be easily implemented on very simple ($1 worth) microcontrollers.

For the experiment shown in Fig. 1 the neural network was trained to 256 patterns (all ASCII code). In this case the system failed for all 9 letters at the highest level of noise (levels 5 and 6). At level 4 five out of nine characters were classified correctly, while at level 3, seven out of nine characters were classified correctly [2].

One may notice that the artificial neural network of Fig 2 was able to correctly recognize noisy characters while humans may have difficulty in fulfilling the same task.

III. Computational Intelligence for Data Mining

With overwhelming information humans may have difficulty extracting proper information. For example, by observing price changes on different stock markets it is very difficult for humans to find correlations and interactions between various stock. Methods of computational intelligence are capable of solving such problems as long as some relations between stock exists. The problem is that humans have difficulty in analyzing data in more than 3 dimensions. Fig. 3 shows 4-dimensional data set with 76 patterns. These patterns are organized in clusters. Humans may have difficulty in finding how many clusters are there and where they are located.

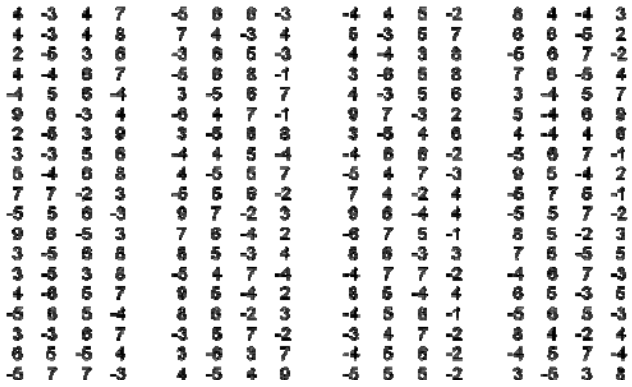


Fig. 3. 76 patterns in 4-dimensional space

Using methods of computational intelligence this problem can be easily solved following a simple scheme:

1. Assume threshold of attraction, which is distance where patterns can be grouped in a single cluster.

2. First pattern is applied and the cluster is formed
3. Next pattern is applied and then:
   a) If distance from all existing clusters is larger than threshold then a new cluster is formed
   b) Else weights of the closest cluster are updated

$$\mathbf{W} = \frac{m\mathbf{W} + \mathbf{X}}{m+1} \tag{6}$$

where **W** are coordinates of cluster and **X** are coordinates of the applied pattern, m is the number of previous patterns of a given set which were used to update this particular.

One may notice that in order to find the location of all clusters each pattern has to be read only once. The only difficult part is a proper guess of the threshold of attraction. In this particular case if the threshold of attraction is selected from 3 to 12 the same three clusters are found as shown in Fig. 4.

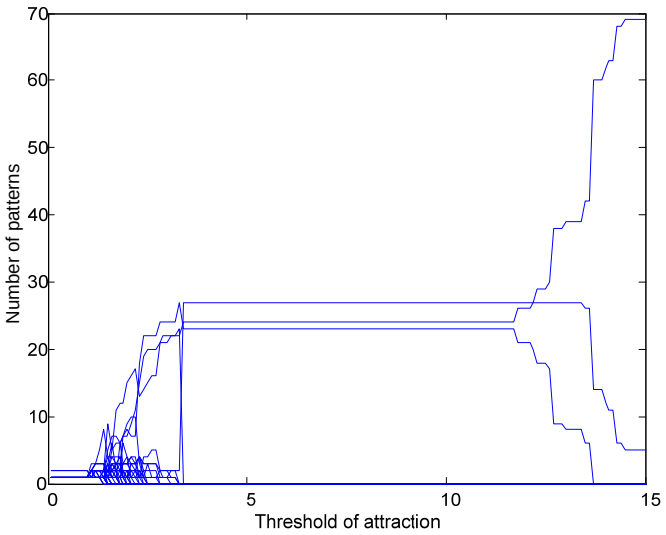
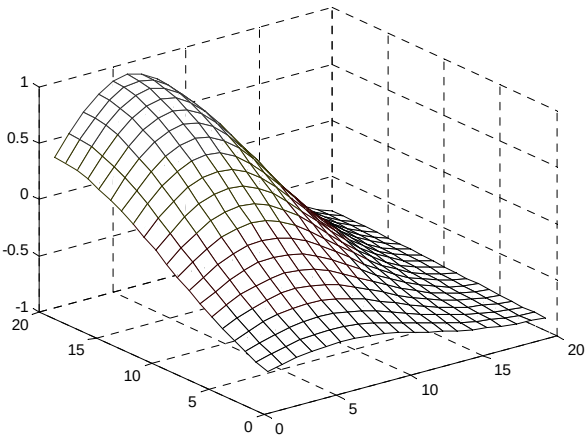


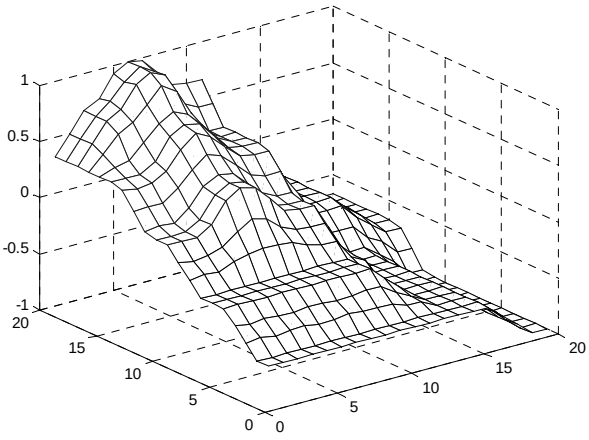
Fig. 4. Number of patterns in clusters depending on the attraction distance

IV. Limitations of Fuzzy Systems

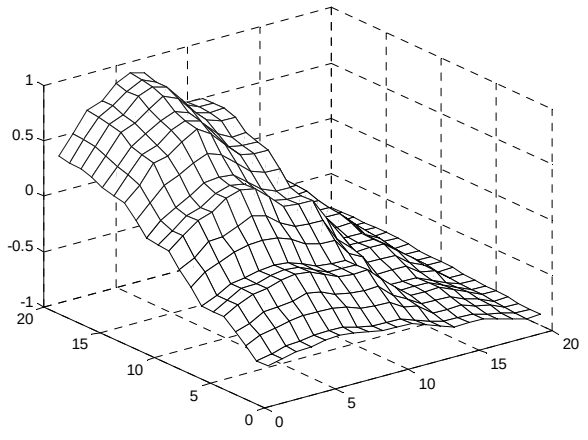
The major advantage of fuzzy logic based systems is their ability to utilize expert knowledge and perception based information. There are two major ways to design fuzzy controllers: a first was developed by Mamdani [3] and second by Tagagi, Sugeno and Kun [4,5]. Control surfaces of fuzzy controllers are relatively raw as is shown in Fig. 5. The rawness of control surface in fuzzy controllers leads to raw control and instabilities [6-8]. Therefore, for resilient control systems fuzzy controllers are not used directly in the control loop [9,10]. Instead traditional PID controllers are often used and fuzzy systems are just used to automatically adjust parameters of PID controllers.



(a)



(b)

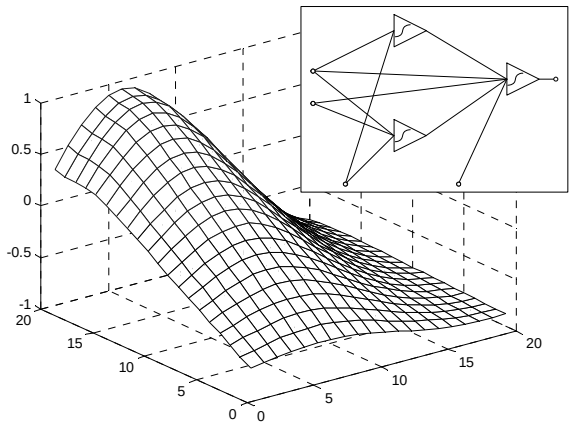


(c)

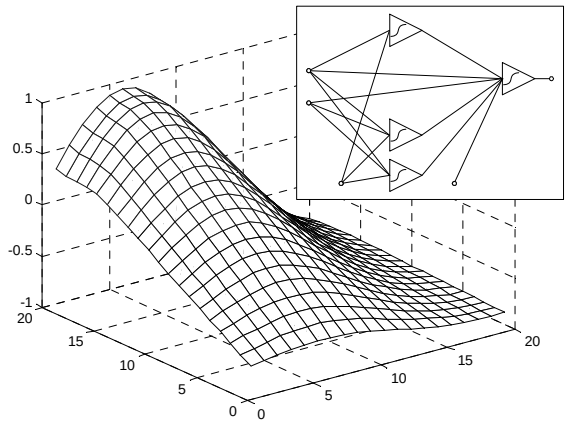
Fig. 5. Control surface obtained with fuzzy controllers (a) required surface, (b) Mamdani controller with trapezoidal membership functions, (c) TSK controller with trapezoidal membership functions

V. NEURAL NETWORKS

Artificial neural networks are well known by their properties of complex nonlinear mappings and they are outperforming fuzzy systems. For example control surface obtained with neural networks for the required surface shown in Fig. 5(a) are shown in Fig. 6.



(a)



(b)

Fig. 6. Control surfaces obtained with neural controller using (a) 3 neuron network, (b) 4 neuron network

Currently, fuzzy controllers are the most popular choice for hardware implementation of complex control surfaces because they are easy to design[12,13]. Neural controllers are more complex and harder to train, but provide an outstanding control surface with much less error than that of a fuzzy controller [8,11]. Figures 7 and 8 show a comparison of fuzzy and neural network based systems implemented in the Motorola HC11 microcontroller. Motorola's 68HC711E9 is a low cost, 8-bit microprocessor; the on-board features of which are 512 bytes of RAM and EEPROM and 12K bytes of UV erasable EPROM. The processor was used with an 8 MHz crystal, allowing an internal clock frequency of 2 MHz.

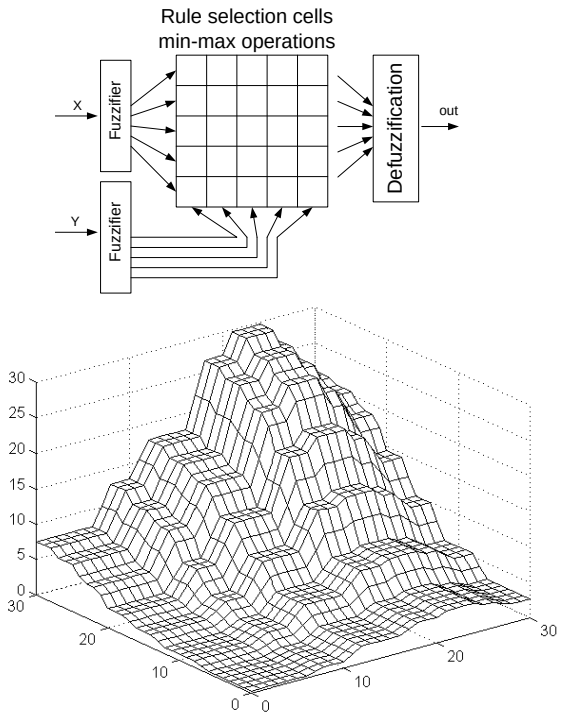


Fig. 7. Control surfaces obtained with Motorola microcontroller HC11 using fuzzy approach with trapezoidal membership functions (7 functions per input) and Tagagi-Sugeno defuzzification [8,11]

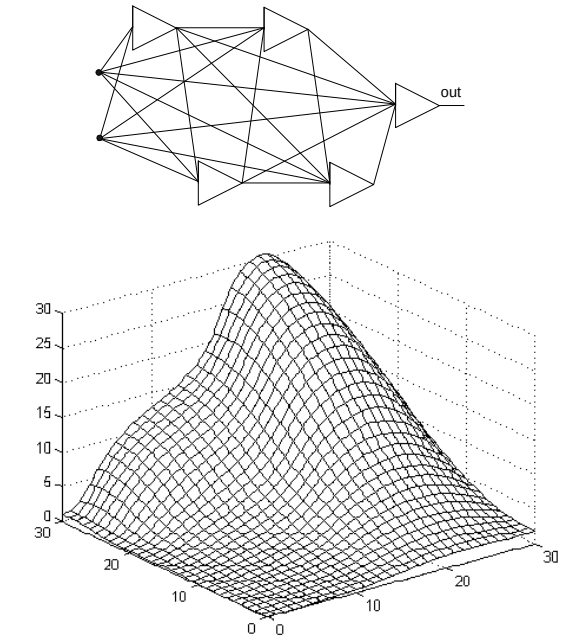


Fig. 8. Control surfaces obtained with Motorola microcontroller HC11 using fuzzy approach with six neurons 2-1-1-1-1-1 architecture and Elliot activation function. [8,11]

A drawback of neural controllers is that the design process is more complicated than that of fuzzy controllers. However, this difficulty can be easily overcome with proper design tools. One severe disadvantage of a fuzzy system is its limited ability of handling problems with multiple inputs. In the case of neural networks such a limitation does not exist.

VI. Limitations of Neural Networks

The most important feature of neural networks is their generalization abilities. This means that neural networks should correctly respond to new patterns which were never used in the training [13]. The number of neurons in such networks should be as small as possible. Unfortunately it is very difficult to train neural networks with good generalization abilities. In order to reduce the number of neurons special network architectures have to be used. Also, more advanced learning algorithms than popular EBP algorithm [14] should be used.

It is relatively easy to find neural network architectures so they can be trained to very small errors. However, it is more important to find an architecture which after training will respond correctly to patterns which were not used for training. Let us illustrate this problem using an example with the peak surface [13,16] shown in Fig. 9(a) as the required surface and let us use equally spaced *10×10=100* patterns (Fig. 9(b)) in training neural networks. The quality of trained networks is evaluated using errors computed for equally spaced *37×37=1,369* patterns. In order to make a valid comparison between training and verification error, the SSE, as defined in (1), is divided by 100 and 1,369 respectively.

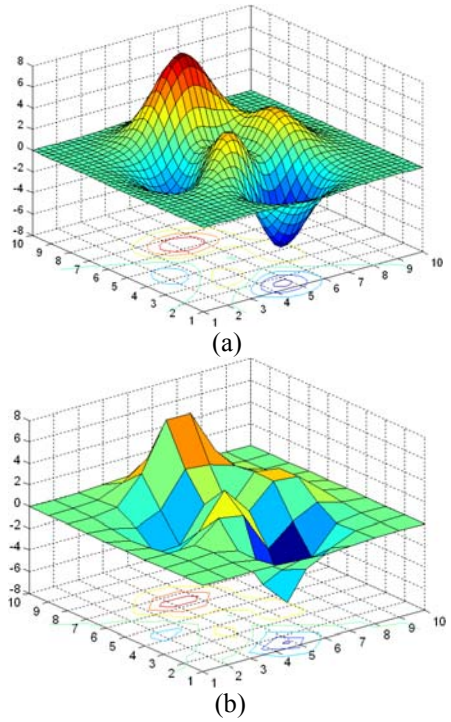


Fig. 9 Surface matching problem: (a) Required 2-D surface with *37×37=1,369* points, used for verification; (b) *10×10=100* training patterns extracted in equal space from (a), used for training.

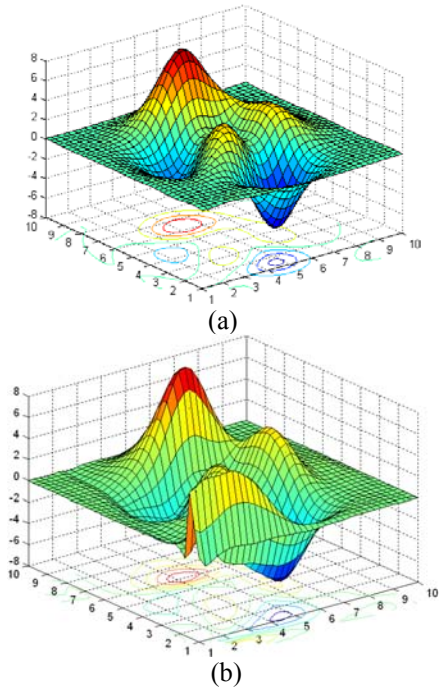


Fig. 10 Training results using 100 trials with (a) NBN algorithm, 8 neurons in FCC network (52 weights); maximum training iteration is 1,000; SSE_{Train}=0.0044, SSE_{Verify}=0.0080 and training time=0.37 s, (b) EBP algorithm, 13 neurons in FCC network (117 weights); maximum training iteration is 1,000,000; SSE_{Train}=0.0018, SSE_{Verify}=0.4909 and training time=635.72 s,

As the training results are shown using the NBN algorithm[16-18], which can handle arbitrarily connected neural networks, it was possible to find the acceptable solution (Fig. 10(a)), SSE_{Train}=0.0044 and SSE_{Verify}=0.0080) with 8 neurons (52 weights). Unfortunately, with the EBP algorithm, it was not possible to find acceptable solutions in 100 trials within 1,000,000 iterations each. The best result out of the 100 trials with 1,000,000 iterations each was SSE_{Train}=0.0764 and SSE_{Verify}=0.1271. When the network size was significantly increased from 8 to 13 neurons (117 weights), the EBP algorithm was able to reach a similar training error as with NBN algorithm, but the network lost its ability to respond correctly for new patterns (between training points). Please notice that indeed with enlarged number of neurons, the EBP algorithm was able to train the network to small error SSE_{Train}=0.0018, but as one can see from Fig. 10(b), the result is unacceptable with verification error SSE_{Verify}=0.4909. When reduced number of neurons are used the EBP algorithm can't converge to required training error. When the size of networks increase, the EBP algorithm can reach the required training error, but trained networks lose their generalization ability and can't process new patterns well (Fig. 10(b)). On the other hand, second order algorithms, such as a NBN algorithm [16,19], works not only significantly faster but it can find good solutions with close to optimal networks (Fig. 10(a)).

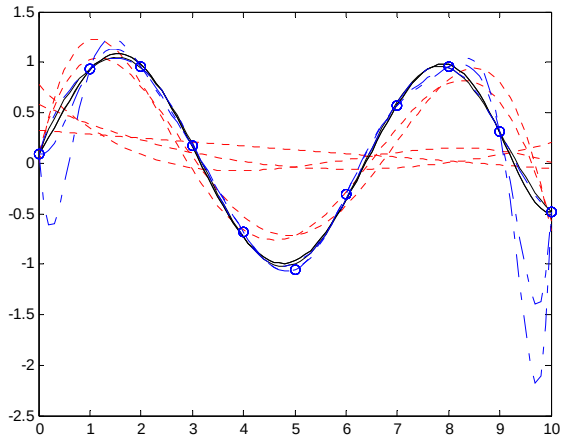


Fig. 11. Approximation of data by different orders of polynomials

From the above analysis, one may notice that in order to sustain neural network generalization abilities the network should have as few neurons/weights as possible [13,20]. This problem is very similar to function approximation by polynomials. If too high order of polynomial is used then errors for training points and values between points cannot be evaluated correctly. In the example on Fig. 11 only 5-th, 6-th, and 7-th order of polynomials are giving adequate results, while higher order polynomials can be tuned to smaller errors for given points; they are useless to predict evaluated new points which were not used for training. We are facing a similar problem with neural network training. If more neurons are used then actually a worse result can be obtained if number of training patterns are limited. In other words for larger neural networks larger number of training patterns must be used. The problem is that efficient learning algorithms like LM - Levenberg-Marquardt [21,22] cannot handle problems with large number of patterns. The solution to this problem is shown in the next section.

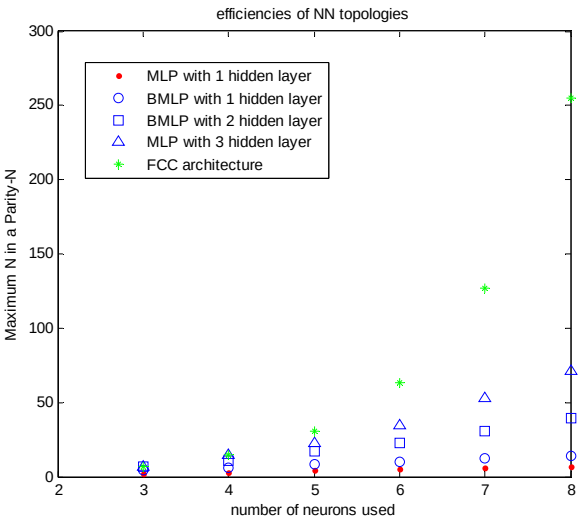


Fig.12. Abilities of solving Parity-N problems as function of number of neurons.

VII. Efficient Algorithm for Training Neural Networks

The most common training algorithm is EBP – Error Back Propagation [15]. It is relatively simple and it does not require a lot of computer resources. This algorithm however seldom leads to a good solution and is extremely slow. Much better results can be obtained with second order algorithms such as LM – Levenberg-Marquardt algorithm [21] or NBN – Neuron by Neuron algorithm [16,18].

Please notice that the capability of neural networks strongly depends on their architecture. Fig. 12 shows ability of solving parity-N problems using different neural network architectures [15,23]. For example if popular MLP (Multi Layer Perceptron) architecture with one hidden layer is used then with 10 neurons only parity-9 problem can be solved. Once the FCC (Fully Connected Cascade) architecture is used then with the same 10 neurons as big a problem as parity-1023 can be solved. Unfortunately, it is very difficult to find software to train these advanced neural network architecture. One exception is SNNS [24] which can train arbitrarily connected neural networks, but only first order algorithms are implemented there so training process is not very efficient. Moreover, as it was shown in the previous section these algorithms may converge to small errors only if an excessive number of neurons are used and the neural network is losing its generalization abilities. Very powerful Levenberg-Marquardt algorithm was unfortunately developed only for MLP networks [21] and it is not suitable to train optimal neural network architectures.

Recently developed Neuron by Neuron (NBN) algorithm [25] implemented in NNT software package [25], was developed in order to eliminate most disadvantages of the LM algorithm and can handle arbitrarily connected neural networks. The NBN algorithm is at least as fast as the LM algorithm []. It can efficiently train arbitrarily connected neural networks. What is also very important is that it can solve with a second order algorithm large problems with basically an unlimited number of patterns.

VIII. CONCLUSION

Neural networks exhibit superior performance in comparison to fuzzy systems but there are several reasons for frustration of people trying to adapt neural networks for their research:

- Neural networks can be over-trained if excessive number of neurons are used. This way the network is losing its ability for generalization and it is not able to correctly process new patterns which were not used for training.
- In most cases the relatively inefficient MLP architecture is used instead of more powerful topologies with connections across layers. As a result, full power of neural networks is not utilized.
- In order to find solutions for close to optimal architectures, second order algorithms such as NBN or LM should be used. Unfortunately, the LM algorithm adopted in popular MATLAB NN Toolbox can handle only MLP topology without connections across layers and these topologies are far from optimal.

- Newly developed NBN algorithm is very fast, it can train any neural network architectures and it has no limitations for the number of patterns used in training. Additional feature of this algorithm is that individual patterns can be added or subtracted from the training set without necessity of training network with entire set of patterns.

The only current limitation of the NBN algorithm is that neural networks should not be too big and practically it can train networks with up to 500 weights.

REFERENCES

[1] J. Kolbusz, S. Paszczyński and B. M. Wilamowski, "Network traffic model for industrial environment", *IEEE Transaction on Industrial Informatics,* vol. 2, No. 4, pp. 213-220, 2006.

[2] B. M. Wilamowski, Vitaly J. Vodyanoy, "Neural Network Architectures for Artificial Noses, " HIS'08, May 25-27, Krakow, Poland, pp. 731-736

[3] H. Mamdani, "Application of Fuzzy Algorithms for Control of Simple Dynamic Plant," *IEEE Proceedings*, Vol. 121, No. 12, pp. 1585-1588, 1974.

[4] M. Sugeno and G. T. Kang, "Structure Identification of Fuzzy Model," *Fuzzy Sets and Systems*, Vol. 28, No. 1, pp. 15-33, 1988.

[5] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Application to Modeling and Control," *IEEE Transactions on System, Man, Cybernetics*, Vol. 15, No. 1, pp. 116-132, 1985.

[6] B. M. Wilamowski "Silicon implementation of computational intelligence for mechatronics", ICM'04 IEEE International Conference on Mechatronics 2004, Istanbul, Turkey, June 3-5, 2004

[7] B. M. Wilamowski, "Methods of Computational Intelligence for Nonlinear Control Systems" *ICCAE' 05 International Conference on Control, Automation and System,* June 2-5, 2005, Gyeonggi-Do, Korea, pp. P1-P8

[8] B.M. Wilamowski and J. Binfet, " Do Fuzzy Controllers Have Advantages over Neural Controllers in Microprocessor Implementation" *Proc of.2-nd International Conference on Recent Advances in Mechatronics - ICRAM'99*, Istanbul, Turkey, pp. 342-347, May 24-26, 1999

[9] Bogdan Wilamowski and Xiangli Li, "Fuzzy System Based Maximum Power Tracking for PV System" *Proc. of the 28th Annual Conference of the IEEE Industrial Electronics Society*, pp. 1990-1994, Sevilla, Spain, Nov 5-8, 2002

[10] Ota, Yasuhiro and Bogdan M. Wilamowski, " Current-Mode CMOS Implementation of a Fuzzy Min-Max Network", *World Congress of Neural Networks*, vol. 2, pp. 480-483, Washington DC, USA, July 17-21, 1995

[11] Bogdan Wilamowski and Jeremy Binfet " Microprocessor Implementation of Fuzzy Systems and Neural Networks ", *International Joint Conference on Neural Networks (IJCNN'01),* pp. 234-239, Washington DC, July 15-19, 2001.

[12] B. K. Bose, "Neural Network Applications in Power Electronics and Motor Drives—An Introduction and Perspective," *IEEE Trans. on Industrial Electronics,* vol. 54, no. 1, pp. 14-33, Feb 2007.

[13] M. Wilamowski, "Neural Network Architectures and Learning Algorithms," *IEEE Industrial Electronics Magazine,* vol. 3, no. 4, pp. 56-63, Dec. 2009

[14] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning representations by back-propagating errors". *Nature*, vol. 323, pp. 533-536, 1986.

[15] B. M. Wilamowski, "Challenges in Applications of Computational Intelligence in Industrials Electronics," (keynote) *ISIE' 10 IEEE International Symposium on Industrial Electronics*, Bari, Italy, July 5-7, 2010, pp. 15-22.

[16] B. M. Wilamowski, H. Yu, "Improved Computation for Levenberg Marquardt Training," *IEEE Trans. on Neural Networks*, vol. 21, no. 6, pp. 930-937, June 2010.

[17] M. Wilamowski, N. J. Cotton, O. Kaynak, G. Dundar, "Computing Gradient Vector and Jacobian Matrix in Arbitrarily Connected Neural Networks," *IEEE Trans. on Industrial Electronics,* vol. 55, no. 10, pp. 3784-3790, Oct 2008

[18] B. M. Wilamowski, N. Cotton, J. Hewlett, and O. Kaynak, "Neural Network Trainer with Second Order Learning Algorithms", 11th *INES 2007 -International Conference on Intelligent Engineering Systems,* Budapest, Hungary, June 29 2007-July 1 2007, pp. 127-132

[19] B. M. Wilamowski " VLSI Analog Multiplier/divider Circuit"(5,6,7) *ISIE'98 International Symposium on Industrial Electronics*, July, 7-10, 1998, Pretoria, South Africa, pp. 493-496

[20] B. M. Wilamowski and O. Kaynak, "Oil Well Diagnosis by sensing Terminal Characteristics of the Induction Motor," *IEEE Transactions on Industrial Electronics,* Vol 47, No 5, pp. 1100-1107, October 2000.

[21] M. T. Hagan, M.B. Menhaj, "Training feedforward networks with the Marquardt algorithm". *IEEE Trans. on Neural Networks*, vol. 5, no. 6, pp. 989-993, Nov. 1994.

[22] MATLAB Neural Network ToolBox http://www.mathworks.com/products/neuralnet/

[23] B. M. Wilamowski, D. Hunter, A. Malinowski, "Solving Parity-n Problems with Feedforward Neural Network," Proc. of the IJCNN'03 International Joint Conference on Neural Networks, pp. 2546-2551, Portland, Oregon, July 20-23, 2003

[24] Stuttgart Neural Network Simulator SNNS http://www.ra.cs.uni-tuebingen.de/SNNS/

[25] NNT - Neural Network Trainer http://www.eng.auburn.edu/~wilambm/nnt/