Can computers be more intelligent than humans?

Bogdan M. Wilamowski
Auburn University

***Abstract* - Humans are not perfect and they make mistakes. For example, humans without computer aided tools would not be able to design VLSI chips larger than 100 transistors. Computers are assisting humans in many aspects of their life. For many years already computers were used for number crunching, office related jobs, etc. More recently computers are used for making intelligent decisions. The methods of computational intelligence are relatively successful, but these methods have to be used with great care. In this presentation advantages and disadvantages of fuzzy and neural networks are presented. It turns out that often popular training algorithms are not capable of tuning neural networks to proper accuracy without losing generalization abilities. Also, abilities of neural networks strongly depend on the used architecture, and surprisingly the most popular architectures usually are least powerful.**

I. Introduction

Humans are capable not only of solving many scientific problems, but also in the process they are making mistakes. Without computer aided tools humans were not able to design chips larger than 100 transistors. By replacing humans by computer in the design process we can now successfully design chips with over 10 billion transistors. Humans are also the weakest links in communication control processes. Notice that an airplane may fly smoothly only when it is on autopilot. With the help of expert systems and computational intelligence, the role of humans are being steadily eliminated. The methods of computational intelligence are relatively successful, but they have to be used with great care [1][2]. In this presentation advantages and disadvantages of fuzzy and neural networks are presented. It turns out that often popular training algorithms are not capable of tuning neural networks to proper accuracy without losing generalization abilities [2][3]. Also, abilities of neural networks strongly depend on the used architecture, and surprisingly the most popular architectures are usually are least powerful [3][4].

II. Neural Networks

The fascination of artificial neural networks started in the middle of the previous century. First artificial neurons were proposed by McCulloch and Pitts [5] shown in Fig. 1. This very simplistic model of a neuron has tremendous computational ability. In the contrast traditional digital design, a different logic function can be obtained without changing the network topology but only by changing threshold and weights. Fig. 2 shows several of different logic functions implemented with a single neuron with different weights.

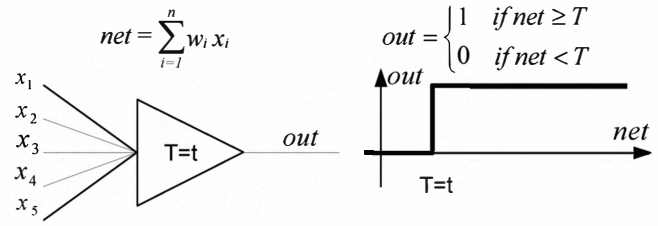


Fig. 1. Simple McCulloch and Pitts neuron with hard activation function

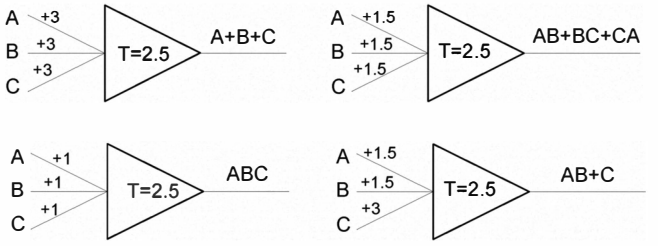


Fig. 2. Different logic functions obtained with the same topology and with different weights.

The McCulloch and Pitts neurons exhibited a incredible power, but for many years people were not able to design neural networks for required logic function. Today we can design very simple neural networks, but we still are not able to design more complex neural networks. Instead, to design these networks, we have developed many algorithms to train them. The most known is the Error-Back-Propagation (EBP) algorithm [6],[7]. the success of this algorithm was not that much due to its computation scheme, but to the assumption that for the training purpose the activation function cannot be as hard as in Fig. 1, but soft as shown in Fig. 3

Soft activation functions make neural network transparent for training [8]. In other words changes in weight values always produce changes on the network outputs. This would not be possible when hard activation functions are used. Having many learning algorithms, it was possible to train various Multi Layer Perceptron (MLP) neural network architectures shown in Fig. 4.

978-1-4244-9640-2/11/$26.00 ©2011 IEEE

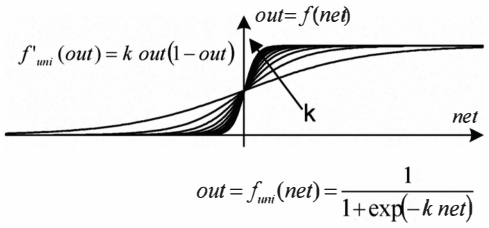


Fig. 3. Soft activation function of unipolar neuron

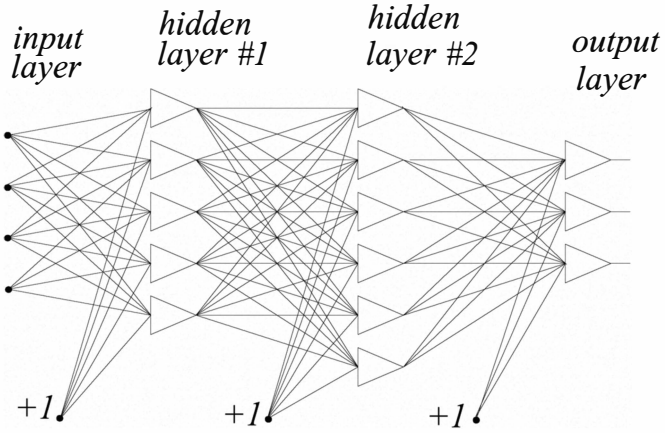


Fig. 4. Four layers MLP neural network architecture

The MPL neural networks were used, and the main neural network topologies for several decades and many difficult problems were solved with this approach.

Parity-N problems are difficult to be solved using traditional digital design and neural networks. Figure 5 shows a solution for Parity-3 problem using digital design, and Fig. 6 shows the Parity-3 problem solved with MLP neural network.

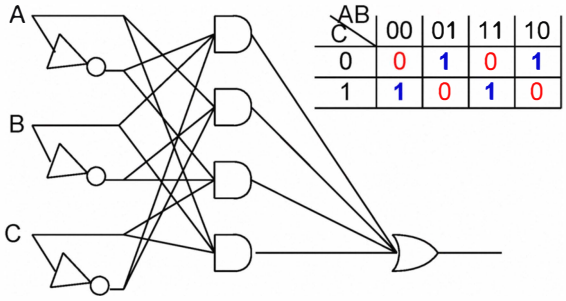


Fig. 5. Digital implementation of Parity-3 problem

It turns out that if instead of MLP architecture as shown in Figs. 4 and 6, another feed forward architecture is used, where connections across layers are allowed then much smaller neural network can be used to solve the same problem. Fig. 7 shows the Fully Connected Cascade (FCC) architecture for the solution of the Parity-3 problem.

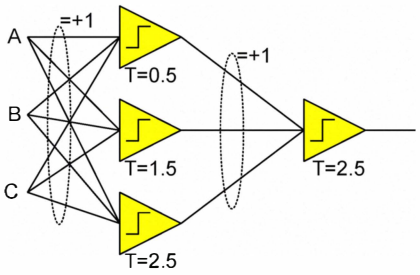


Fig. 6. MLP neural network implementation of Parity-3 problem

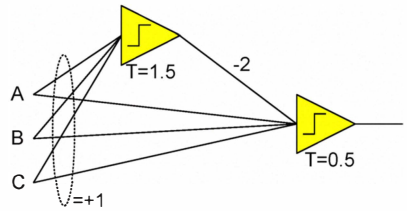


Fig. 7. FCC neural network implementation of Parity-3 problem.

If connections across layers are allowed in FCC or in Bridged Multi Layer Perceptron (BMLP), then neural networks are gaining a tremendous power. Fig. 8 shows how large Parity N problems can be solved with limited number of neurons. For example, with 10 neurons the MLP neural network with one hidden layer is the largest problem the network can solve in Parity-9. While with the same 10 neurons using FCC neural network it is possible to solve as large as Parity-1023 problem [9-12]. One may see that abilities of neural networks strongly depend on the used topology, and that with connections across layers neural networks are gaining incredible power.

Unfortunately neural networks with connections across layers are seldom used because until recently there were no efficient training software for such neural networks.

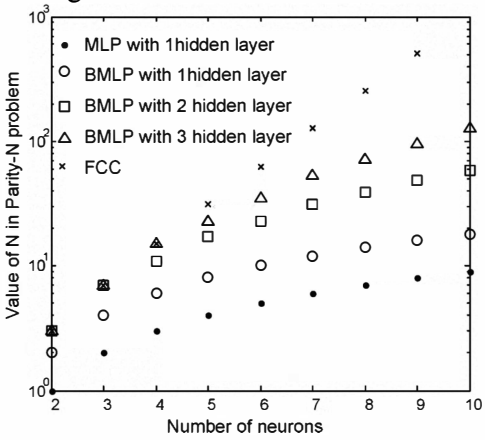


Fig. 8. Abilities of solving Parity-N problems as function of number of neurons.

Neural Networks can be used not only as classifiers but also as nonlinear approximators. For example, control surfaces, obtained with neural networks for the required surface shown in Fig. 9 are shown in Fig. 10.

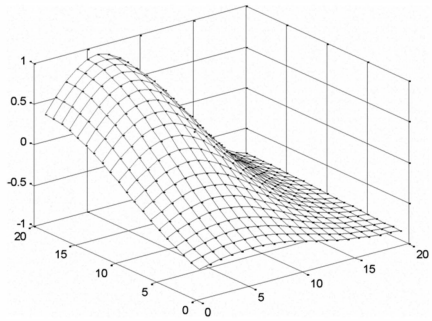
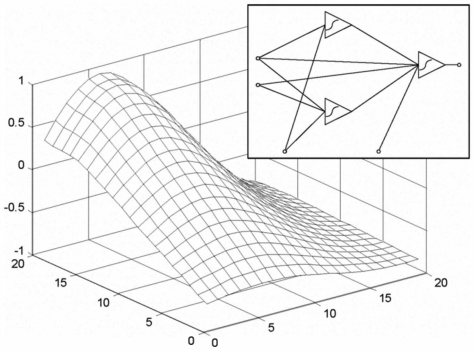
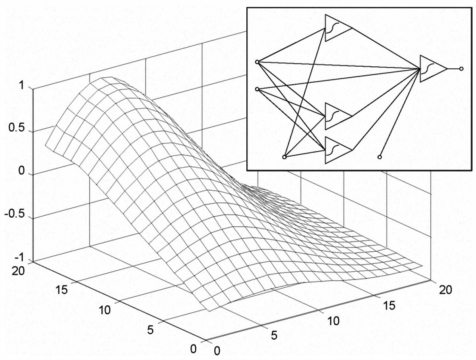


Fig. 9. Required nonlinear function



(a)

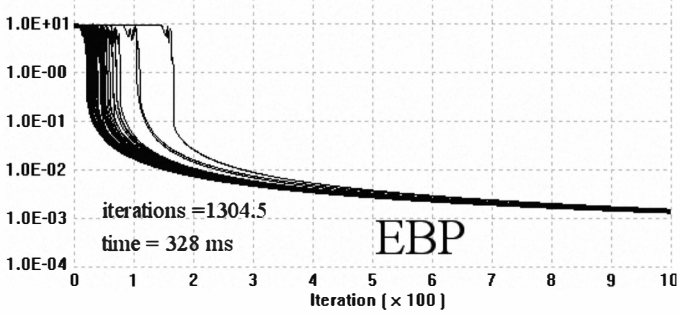


(b)

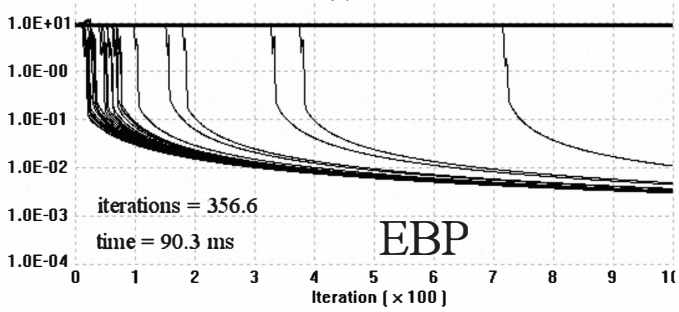
Fig. 10. Control surfaces obtained with neural controller using (a) 3 neuron network, (b) 4 neuron network

Many algorithms for training neural networks were already developed. The Error Back Propagation (EBP)[6][7] algorithm is the most popular algorithm, but it is very slow. The EBP training process requires significantly more iterations (Fig. 11) and more time than more advanced algorithms such as Levenberg- Marquardt (LM) [13][14] or Neuron by Neuron (NBN) [15-20] algorithms. Also it can handle much easier the MLP architecture than neural networks with arbitrarily connected neurons (Fig. 11). What is most important is that EBP is not only slow but it is has difficulties to find solutions for close to optimum architectures.

Second order algorithms are much faster (Fig. 12). Figs. 12 (a) and (b) show training results for MLP architectures using both LM (Fig. 12.a) and NBN (Fig. 12b) algorithms. Since the LM algorithm was not implemented for this optimal architectures like FCC, the training results for this architecture are shown only for NBN algorithm. One may notice that for MLP architectures, LM and NBN algorithms show similar results. In the contrast to the EBP the NBN algorithm can handle advanced neural network architectures even better than traditional MLP architectures (see Figs. 12b and 12c)



(a)

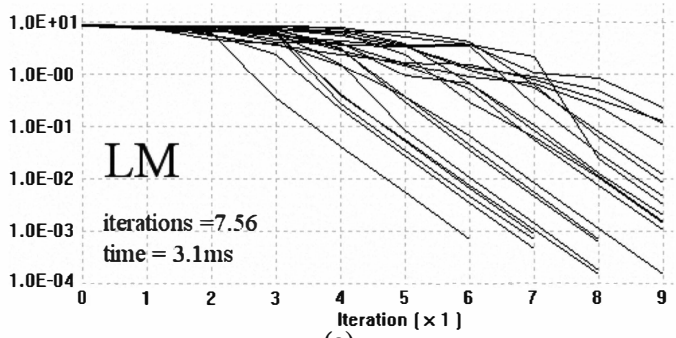


(b)

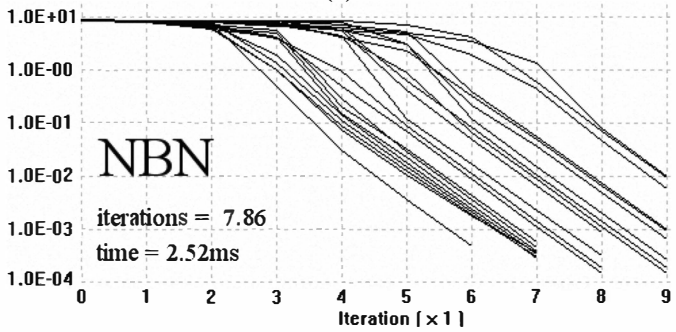
Fig. 11 Parity-3 problem solved with the Error-Back Propagation algorithms (a) using MLP architecture with 3 neurons in hidden layer (b) using FCC architecture with 2 neurons

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 [1]. 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 [1] should be used.

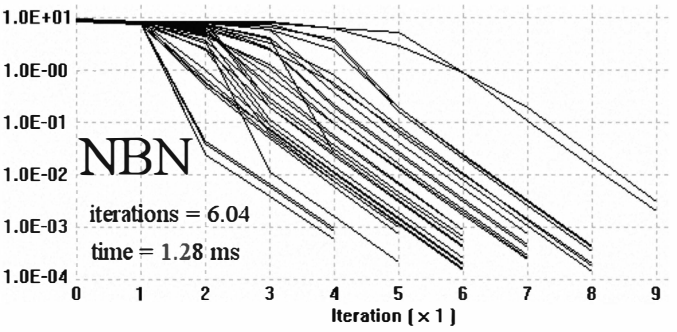
It is relatively easy to find neural network architectures, so they can be trained to very small errors. However, it is more important to find an architecture, which after training, will respond correctly to patterns which were not used for training. Let us illustrate this problem using an example with the peak surface [1][17]] shown in Fig. 9 as is required .



(a)

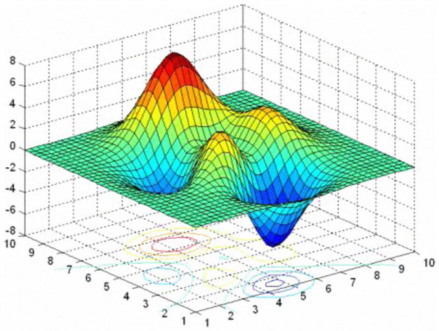


(b)



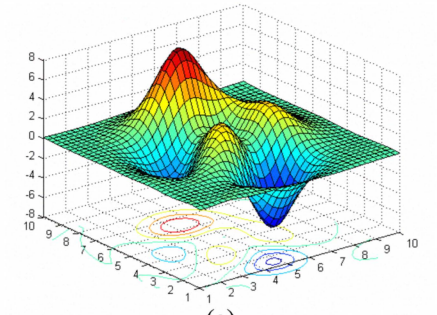
(c)

Fig. 12 Parity-3 problem solved with the second order algorithms (a) using LM algorithm and MLP architecture with 3 neurons in hidden layer, (b) using NBN algorithm and MLP architecture with 3 neurons in hidden layer, (c) using NBN algorithm and FCC architecture with 2 neurons

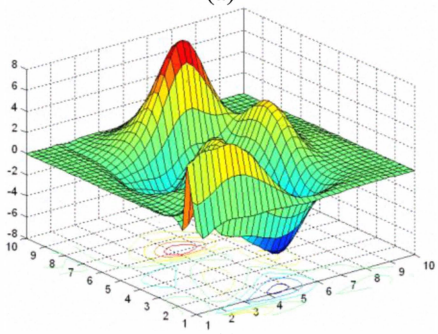


(a)

Fig. 13 Required control surface



(a)



(b)

Fig. 14 Training results using 100 trials with (a) NBN algorithm, 8 neurons in FCC network (52 weights); with maximum number of iterations of 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); with maximum number of iterations of 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[15-17], which can handle arbitrarily connected neural networks, it was possible to find the acceptable solution (Fig. 14a). The best result out of the 100 trials using EBP algorithm is shown in Fig. 14b. 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).

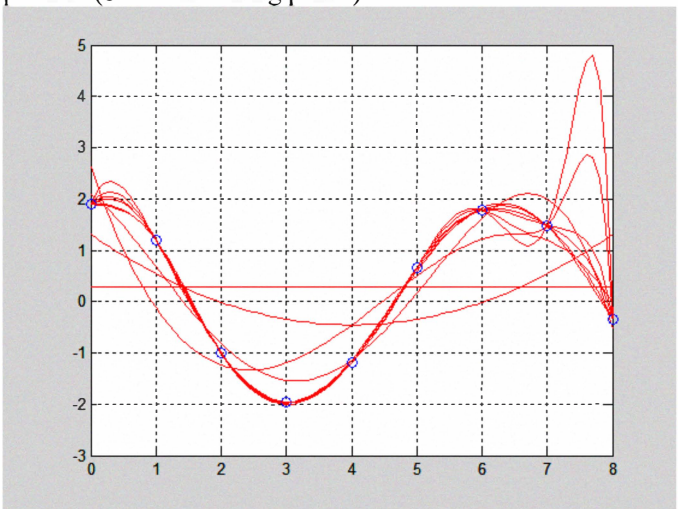


Fig. 15. Approximation of data by different orders of polynomials

One may notice that in order to sustain neural network generalization abilities the network should have as few neurons/weights as possible [1][21][22]. 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 5th, 6th, and 7th 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.

III. Fuzzy Systems

Interestingly, fuzzy logic has a more general nature and it works equally well as Boolean logic. Fig. 16 shows fuzzy logic operations on zero-one Boolean variables (Fig. 16.a) and on fuzzy variables (Fig. 16.b).

AB

0	0	0
0	1	0
1	0	0
1	1	1

A+B

0	0	0
0	1	1
1	0	1
1	1	1

(a)

MIN(A,B)

0.2	0.3	0.2
0.2	0.8	0.2
0.7	0.3	0.3
0.7	0.8	0.7

MAX(A,B)

0.2	0.3	0.3
0.2	0.8	0.8
0.7	0.3	0.7
0.7	0.8	0.8

(b)

Fig. 16. Comparison of (a) Boolean and (b) Fuzzy logic.

Fuzzy systems similar to neural networks may also process analog values in the range of 0 to 1, but the issue is how to handle analog inputs within much wider range (simple scaling will not work). Two slightly different approaches were proposed: one by Mamdani [23] and second by Takagi-Sugeno-Kang (TSK) [24][25].

The Mamdani concept follows the rule of ROM and PLA digital structures where AND operators are selecting specified addresses and then OR operators are used to find the output bits from the information stored at these addresses. Also, in the case of the fuzzy system, as presented in Fig. 17, first MIN and then MAX operators are used.

At the left side of the diagram, analog inputs are converted by fuzzifiers into sets of fuzzy variables. For each analog input several fuzzy variables typically are generated. Each fuzzy variable has an analog value between zero and one.

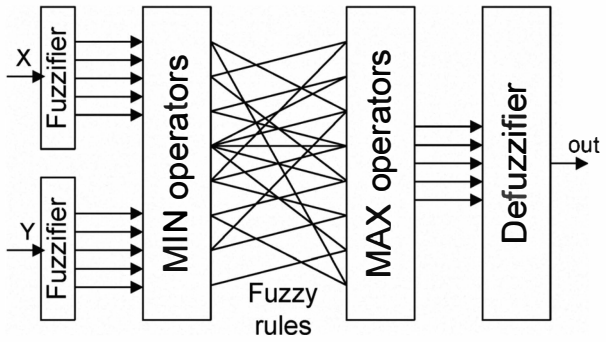


Fig. 17. Block diagram of a Mamdani type fuzzy controller.

More recently Mamdani architecture was replaced by TSK [19][20] (Takagi, Sugeno, Kang) architecture where the defuzzification block was replaced with normalization and weighted average [21][26][27]. The block diagram of TSK approach is shown in Fig. 18. The TSK architecture does not require MAX operators, but a weighted average is applied directly to regions selected by MIN operators. What makes the TSK system really simple is that the output weights are proportional to the average function values at the selected regions by MIN operators. The TSK fuzzy system works as a lookup table.

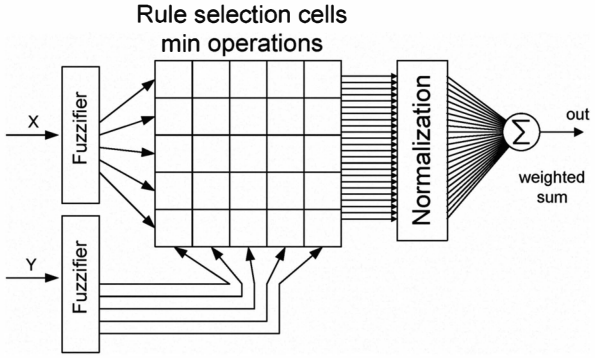


Fig. 18. TSK (Takagi-Sugeno-Kang) fuzzy architecture.

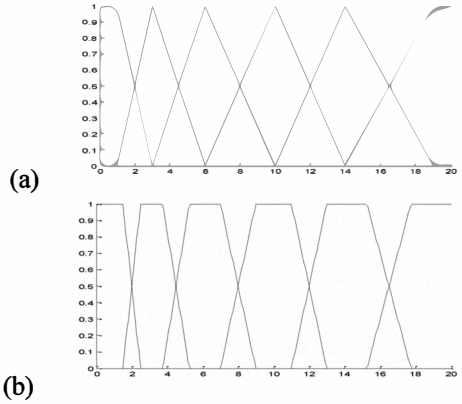
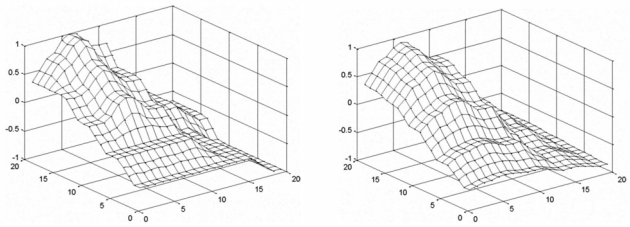


Fig. 19. Membership functions for various fuzzification methods: (a) triangular, (b) trapezoidal.

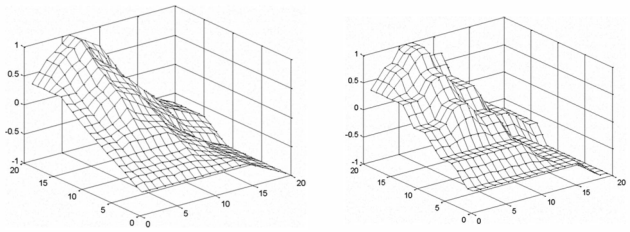
Various types of fuzzification methods can be used as shown in Fig. 19. Each point of the input analog variable should belong to at least one and preferably no more than two membership functions. For overlapping functions the sum of two membership functions must not be larger than one. This also means that overlap must not cross the points of maximum values (ones). For higher accuracy more membership functions should be used.

Let us use the concept of fuzzy systems to approximate function shown in Fig. 9. Fig. 20 shows results obtained with Mamdani and TSK approaches. Depending on if triangular or trapezoidal fuzzifiers are used, results are also slightly different as shown in Fig. 21. In most cases triangular fuzzifiers are giving better results than trapezoidal ones.



(a) (b)

Fig. 20. Control surface obtained with fuzzy controllers (a) required surface, (b) Mamdani controller with trapezoidal membership functions, (c) TSK controller with trapezoidal membership functions



(a) (b)

Fig. 21. Different control surfaces obtained with different fuzzification methods: (a) triangular, (b) trapezoidal using TSK approach

IV. Comparison of Neural and Fuzzy Systems

The major advantage of fuzzy logic based systems is their ability to utilize expert knowledge and perception based information. Currently, fuzzy controllers are the most popular choice for hardware implementation of complex control surfaces because they are easy to design [21].

Artificial neural networks are well known by their properties of complex nonlinear mappings and they are outperforming fuzzy systems. 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 [21].

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 [3]. Advantages and disadvantages of fuzzy and neural networks are compared in the Fig. 22

Comparison of Fuzzy Systems and Neural Networks

	Fuzzy	Neural
Number of inputs	Limited to 3 or 4	Unlimited
Analog implementation	Difficult	Easy
Microcontroller implementation	Easy	More difficult
FPGA implementation	Easy	More difficult
Speed	Resonable	Faster
Smoothens of the surface	Poor	Supperb
Errors	Large	Small
Design complexity	Simple	Difficult

Fig. 22. Comparison of neural and fuzzy systems

V. Supremacy of Computational Intelligence

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.

Let us consider a relatively complex problem of image recognition. In this problem there is 7*8=56 inputs and also 56 outputs. An intelligent system reads noisy left columns. On Fig. 23 pairs of columns have different levels of noise introduced, starting from 10% noise for the first pair of columns to the 60% of noise for the most right pair of columns. Since in this example there are only 9 letters to be recognized the system can be simplified and, instead of 56 outputs only 9 outputs can be used. This is possible because once the proper letter is recognized then it is trivial to produce an image for this letter.

Obviously 56 inputs is too many for a practical implementation of fuzzy systems, and neural networks can handle this problem relatively easily. Fig. 24 shows correctly recognized letters by neural networks. The same letter recognition was already tested on more than 1000 humans and in all cases humans failed a proper recognition.

There are many similar problems where neural networks can recognize complex patterns better than humans. These may include: recognition of stock market trends [27], prediction of seat occupancy for airliners [28], military target recognitions, etc.

The experiment of character recognition was carried on in the following scheme [29-31]. Original characters were distorted by six levels of noise. Then the neural networks were used to recognize noise characters (left columns) and to retrieve the original character (right columns).

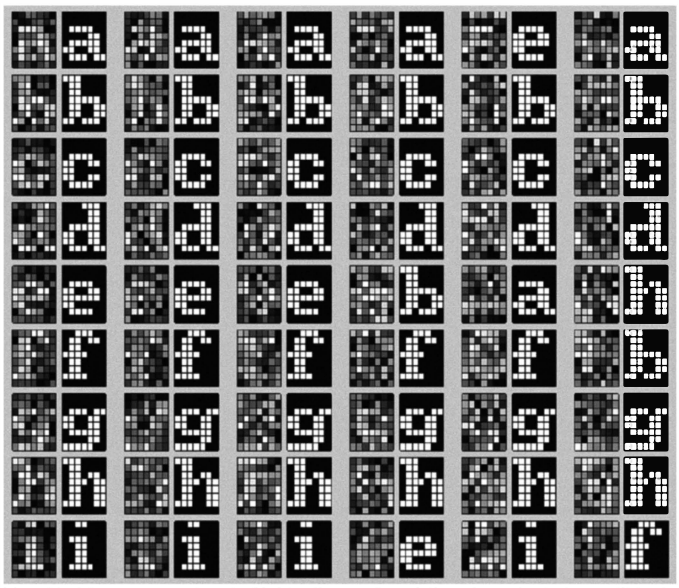


Fig. 23. Result of pattern retrieval using neural networks

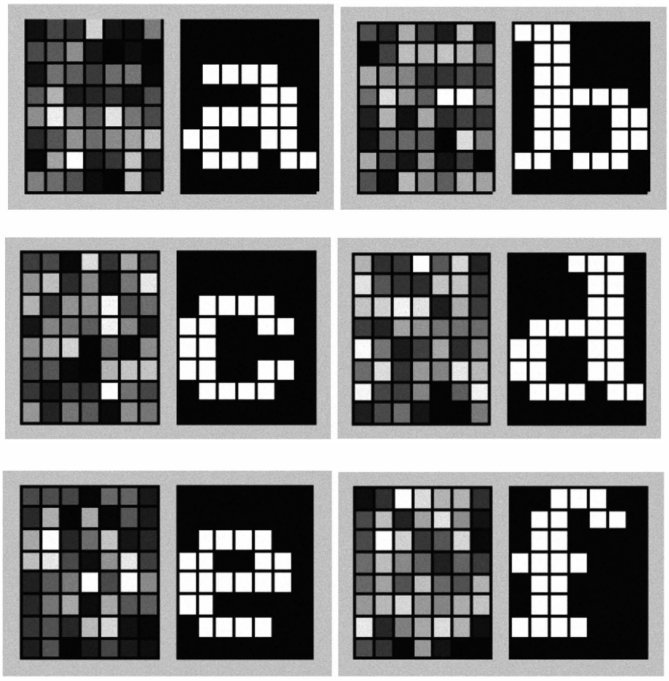


Fig. 24. Correctly recognized letters by neural networks.

Fig. 24 shows samples of correctly recognized letters by neural networks. One may notice that artificial neural networks were able to correctly recognize noisy characters while humans may have difficulty in fulfilling the same task.

With overwhelming information humans may have difficulty extracting proper information. The problem is that humans have difficulty in analyzing data in more than 3 dimensions. Fig. 25 shows a 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. Using methods of computational intelligence, this problem can be easily solved following a simple scheme [30][31].

4	-3	4	7	-5	6	6	-3	-4	4	5	-2	8	4	-4	3
4	-3	4	8	7	4	-3	4	5	-3	5	7	6	6	-5	2
2	-5	3	6	-3	6	5	-3	4	-4	3	8	-5	6	7	-2
4	-4	6	7	-5	6	8	-1	3	-6	5	8	7	6	-5	4
-4	5	6	-4	3	-5	6	7	4	-3	5	6	3	-4	5	7
9	6	-3	4	-6	4	7	-1	9	7	-3	2	5	-4	6	9
2	-5	3	9	3	-5	6	8	3	-5	4	6	4	-4	4	6
3	-3	5	6	-4	4	5	-4	-4	6	6	-2	-5	6	7	-1
5	-4	6	8	4	-5	5	7	-5	4	7	-3	9	5	-4	2
7	7	-2	3	-5	5	6	-2	7	4	-2	4	-5	7	5	-1
-5	5	6	-3	9	7	-2	3	9	6	-4	4	-5	5	7	-2
9	6	-5	3	7	6	-4	2	-6	7	5	-1	8	5	-2	3
3	-5	6	8	8	5	-3	4	8	6	-3	3	7	6	-5	5
3	-5	3	8	-5	4	7	-4	-4	7	7	-2	-4	6	7	-3
4	-6	5	7	9	5	-4	2	8	5	-4	4	6	5	-3	5
-5	6	5	-4	8	6	-2	3	-4	5	8	-1	-5	6	5	-3
3	-3	6	7	-3	5	7	-2	-3	4	7	-2	8	4	-2	4
6	5	-5	4	3	-6	3	7	-4	5	6	-2	-4	5	7	-4
-5	7	7	-3	4	-5	4	9	-5	5	5	-2	3	-5	3	8

Fig. 25. 76 patterns in 4-dimensional space

VI. CONCLUSION

With time there are more and more cases where computers exhibits superior to human performance. Recently they are able to make many intelligent decisions. This is primarily to advanced neural networks architectures and learning algorithms. 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 easily 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 the 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. Additionally, a 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] B. M. Wilamowski, "Neural Network Architectures and Learning Algorithms," *IEEE Industrial Electronics Magazine,* vol. 3, no. 4, pp. 56-63, Dec. 2009

[2] B. M. Wilamowski "Silicon implementation of computational intelligence for mechatronics", ICM'04 IEEE International Conference on Mechatronics 2004, Istanbul, Turkey, pp.1-8. June 3-5, 2004.

[3] 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.

[4] B.M. Wilamowski "Neural Network Architectures and Learning", ICIT'03 - International Conference on Industrial Technology, Maribor, Slovenia, December 10-12, 2003

[5] W.S. McCulloch and W.H. Pitts., A logical calculus of the ideas imminent in nervous activity. Bull. Math. Biophy. 5 pp. 115-133, 1943.

[6] P. Werbos, Beyond regression: new tools for prediction and analysis in behavioral sciences. Ph.D. diss., Harvard University, 1974.

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

[8] B.M. Wilamowski and L. Torvik, " Modification of Gradient Computation in the Back-Propagation Algorithm", *ANNIE'93 - Artificial Neural Networks in Engineering*, St. Louis, Missouri, November 14-17, 1993, pp. 175-180.

[9] B. M. Wilamowski, N. J. Cotton, O. Kaynak,, and G. Dundar, "Method of computing gradient vector and Jacobean matrix in arbitrarily connected neural networks" *ISIE 2007- IEEE International Symposium on Industrial Electronics*, Vigo, Spain, 4-7 June 2007, pp. 3298-3303.

[10] B.M. Wilamowski "Efficient Neural Network Architectures and Advanced Training Algorithms", *Gdańsk University of Technology Faculty of ETI Annals,* Vol 18, pp. 345-352, 2010

[11] B. 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, 2000.

[12] B.M. Wilamowski, Hao Yu, and Kun Tao Chung "Parity-*N* Problems as a Vehicle to Compare Efficiency of Neural Network Architectures" *Industrial Electronics Handbook, vol. 5 – Intelligent Systems,* 2nd Edition, chapter 10, pp. 10-1 to 10-8, CRC Press 2011.

[13] K. Levenberg, "A method for the solution of certain problems in least squares". *Quarterly of Applied Machematics*, 5, pp. 164-168, 1944.

[14] Hagan, M. T. and Menhaj, M., "Training feedforward networks with the Marquardt algorithm", IEEE Transactions on Neural Networks, vol. 5, no. 6, pp. 989-993, 1994

[15] 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

[16] B. 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

[17] 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.

[18] B. M. Wilamowski and H. Yu, "Neural Network Learning Without Backpropagation," *IEEE Trans. on Neural Networks*, vol. 21, no.11, pp. 1793 - 1803 Nov. 2010.

[19] Yu and B. M. Wilamowski, "Fast and efficient and training of neural networks," in *Proc. 3nd IEEE Human System Interaction Conf. HSI 2010*, Rzeszow, Poland, May 13-15, 2010, pp. 175-181.

[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] 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.

[22] Andersen, Thomas J. and B.M. Wilamowski, " A Modified Regression Algorithm for Fast One Layer Neural Network Training", *World Congress of Neural Network*s, vol. 1, pp. 687-690, Washington DC, USA, July 17-21, 1995.

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

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

[25] 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.

[26] Safer A. and B. M. Wilamowski, "Using neural networks to predict abnormal returns of quarterly earnings" presented at *1999 International Joint Conference on Neural Networks - IJCNN'99*, pp. 3840-3843, Washington, DC, July 10-16, 1999.

[27] Lawrence R. Weatherford, Travis Gentry, and Bogdan Wilamowski, "Neural Network Forecasting for Airlines: A Comparative Analysis" *Journal of Revenue and Pricing Management*. vol.1, no 4, pp.319-331, 2003.

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

[29] Katarzyna Wilamowska, Milos Manic, "Unsupervised pattern clustering for data mining", *IECON'01 - The 27th Annual Conference of the IEEE Industrial Electronics Society*, pp. 1862-1867.

[30] Aleksander Malinowski, Katarzyna Wilamowska, "Unsupervised pattern clustering for data mining", *IECON'99 - The 25th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, pp. 111-115.

[31] 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.