Comparison of Fuzzy and Neural Systems for Implementation of Nonlinear Control Surfaces

T.T. Xie, H. Yu, and B.M. Wilamowski

Department of Electrical and Computer Engineering,
Auburn University, Auburn, AL, USA
tzx0004@auburn.edu, hzy0004@auburn.edu, wilam@ieee.org

Abstract. In this paper, a comparison between different fuzzy and neural systems is presented. Instead of using traditional membership functions, such as triangular, trapezoidal and Gaussian, in fuzzy systems, the monotonic pair-wire or sigmoidal activation function is used for each neuron. Based on the concept of area selection, the neural systems can be designed to implement the identical properties like fuzzy systems have. All parameters of the proposed neural architecture are directly obtained from the specified design and no training process is required. Comparing with traditional neuro-fuzzy systems, the proposed neural architecture is more flexible and simplifies the design process by removing division/normalization units.

1 Introduction

Traditional methods, such as PID (Proportion-Integration-Differentiation) algorithm, are relatively helpful to design linear control systems, but they are in trouble if the system has nonlinear properties [Farrell and Polycarpou 2008]. Unfortunately, most systems in practice are nonlinear.

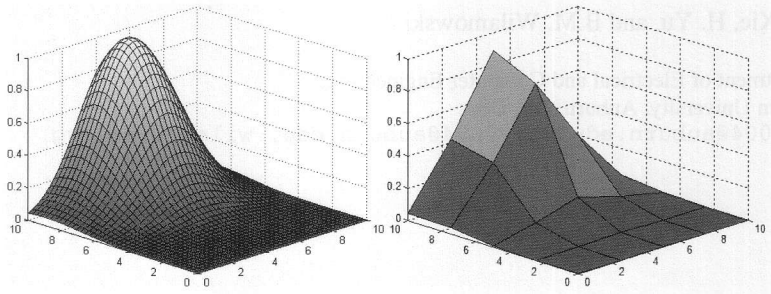
For some nonlinear systems, by adding a reverse nonlinear function to compensate for the nonlinear behavior of the system, the input-output relationship would become approximately linear. In those cases, the nonlinear problems can still be solved by the well developed linear control theory. Otherwise, it is necessary to apply an adaptive change to satisfy the nonlinear behavior of the systems. These adaptive systems are best handled with fuzzy systems and neural networks [Wilamowski 2002; Wilamowski and Binfet 2001].

In this paper, various fuzzy and neural systems are studied. The proposed neural architecture, using the concept of area selection in neural networks, is introduced and compared with classic fuzzy systems and traditional neuro-fuzzy systems. The comparison is based on the function approximation problem. The purpose of the problem is: using the given *25* points (Fig. 1b) to approximate the *1600* points (Fig. 1a) in the same range. All the required points satisfy the

Z.S. Hippe et al. (Eds.): Human – Computer Systems Interaction, AISC 99, Part II, pp. 313–324.
springerlink.com

relationship as described by equation (1) and the approximation will be evaluated using the SSE (sum-square-error) of the *1600* points.

$$f(x,y)=exp(-0.12(x-9)^2-0.12(y-5)^2) \quad (1)$$



(a) (b)

Fig. 1 Surfaces obtained from equation (1): (a) desired surface (*40×40=1600* points); (b) given surface (*5×5=25* points)

2 Fuzzy Systems

There are two commonly used architectures for fuzzy systems development. The one is proposed by Mamdani [Mamdani 1974; McKenna and Wilamowski 2001], as shown in Fig. 2 and the other in Fig. 5 is proposed by Takagi, Sugeno, and Kang (TSK)[Takagi and Sugeno 1985; Wilamowski and Binfet 1999].

2.1 Mamdani Fuzzy System

As shown in Fig. 2 below, Mamdani fuzzy systems mainly consist of three parts: fuzzifiers, fuzzy rules and defuzzifiers.

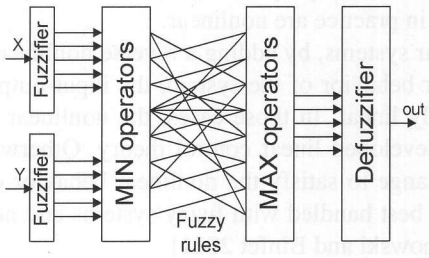
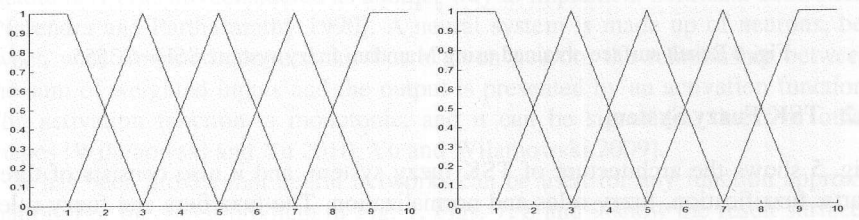


Fig. 2 Architecture of Mamdani fuzzy system

In order to design Mamdani fuzzy systems, the first step is to do fuzzification on inputs, which means to convert the analog inputs into sets of fuzzy variables using fuzzifiers. For each analog input, several fuzzy variables, with values between 0 and 1, are generated and the sum of them should be 1. There are various types of fuzzification methods, such as triangular, trapezoidal, Gaussian, sine, parabola, or any combination of them. Triangular and trapezoidal membership functions are the simplest and in most practical cases, acceptable results can be obtained with these two approaches.

More membership functions can be used for higher accuracy; however, too many membership functions causes frequent controller action (known as "hunting"), and may lead to system instability.

In the given problem in Fig. 1, we will use 10 membership functions (5 for each direction) and both triangular and trapezoidal membership functions are used, as shown in Fig. 3.



(a) (b)

Fig. 3 Membership functions used for fuzzification: (a) x-direction; (b) y-direction

After fuzzification, the following step is to perform fuzzy rules on fuzzy variables. Fuzzy logic rules have MIN and MAX operators, which can be treated as the extended Boolean logic. For binaries "0" and "1", the MIN and MAX operators in fuzzy logic rules perform the same calculation as the AND and OR operators in Boolean logic, respectively (Table 1); for fuzzy variables, the MIN operator is to get the minimum value and the MAX operator is to get the maximum value (Table 2).

Table 1 Fuzzy and Boolean logic rules for binaries

Binaries		MIN	AND	MAX	OR
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	1	1	1

Table 2 Fuzzy logic rules for fuzzy variables

Fuzzy variables		MIN	MAX
0.3	0.6	0.3	0.6
0.7	0.3	0.3	0.7
0.5	0.4	0.4	0.5
0.1	0.8	0.1	0.8

The last step is defuzzification, which converts the results of "MAX of MIN" operations to an analog output value. There are several defuzzification schemes used and the most common is the centroid type of defuzzification.

For Mamdani fuzzy system, the result surface of the given problem could be obtained as

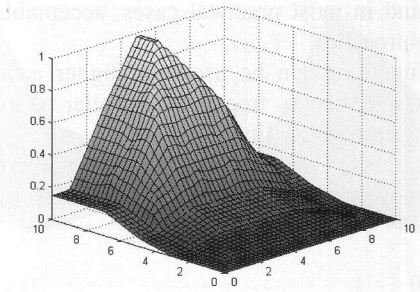


Fig. 4 Result surface obtained using Mamdani fuzzy system; SSE= 6.3555

2.2 TSK Fuzzy System

Fig. 5 shows the architecture of TSK fuzzy system, and it also consists of three parts: fuzzification, fuzzy rules and normalization. The fuzzifiers and fuzzy rules are almost the same as are used in Mamdani fuzzy system.

The difference is that, unlike the "MAX of MIN" defuzzification in Mamdani fuzzy systems, the TSK fuzzy systems do not require MAX operators; instead, a weighted average is applied directly to the results of MIN operators. The TSK fuzzy architecture is much simpler than Mamdani architecture, because the output weights are proportional to the average function values at the selected regions by MIN operators.

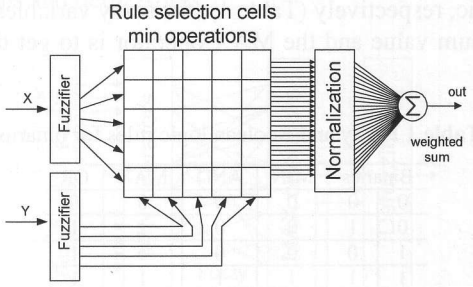


Fig. 5 Architecture of Mamdani fuzzy system

Fig. 6 shows the result surface using TSK fuzzy system; one may notice that it is more accurate than the result obtained by Mamdani architecture (Fig. 4).

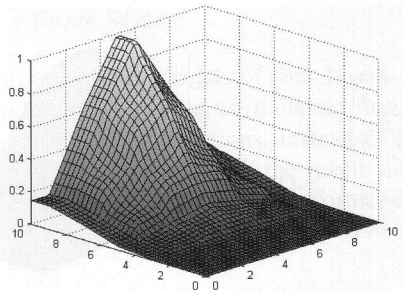
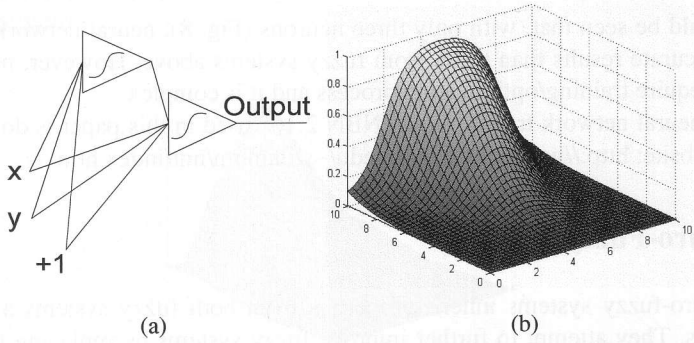


Fig. 6 Result surface obtained using TSK fuzzy system; SSE= 2.2864

3 Neural Networks

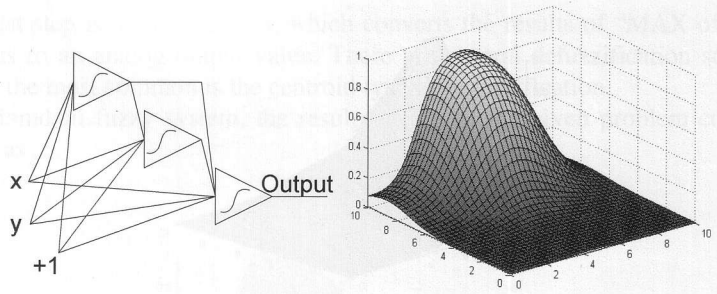
Neural networks are considered as another way to implement nonlinear controllers [Narendra and Parthasarathy 1990]. A neural system is made up of neurons, between with weighted connections. For a given neuron, the relationship between the sum of weighted inputs and the output is presented by an activation function. The activation function is monotonic, and it can be sigmoidal, linear or other shapes [Wilamowski and Yu 2010; Yu and Wilamowski 2009].

It has been proven that neural networks can be used for any function approximation. For the given problem in Fig. 1, Figs. 7, 8 and 9 show the result surfaces using different number of neurons with fully connected cascade (FCC) networks. Each neuron uses unipolar sigmoidal activation function and neuron-by-neuron (NBN) algorithm is used for training.



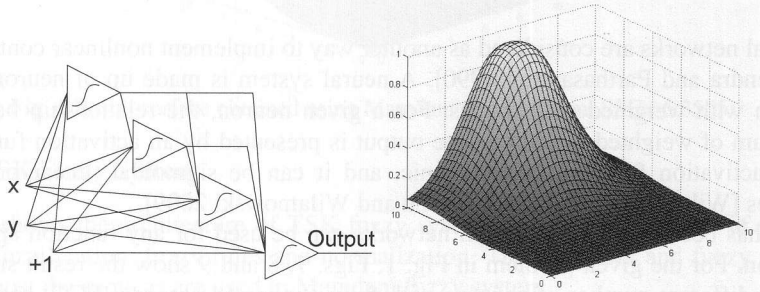
(a) (b)

Fig. 7 (a) Two neurons in FCC network; (b) Result surface with SSE= 5.1951



(a) (b)

Fig. 8 (a) Three neurons in FCC network; (b) Result surface with SSE= 0.9589



(a) (b)

Fig. 9 (a) Four neurons in FCC network; (b) Result surface with SSE= 0.0213

It could be seen that, with only three neurons (Fig. 8), neural networks can get more accurate results than those from fuzzy systems above. However, neural networks require training/optimization process and it is complex.

The neural network training tool "NBN 2.10" used in this paper is downloaded from website: http://www.eng.auburn.edu/~wilambm/nnt/index.htm.

4 Neuro-Fuzzy Systems

The neuro-fuzzy systems inherit properties from both fuzzy systems and neural networks. They attempt to further improve fuzzy systems by replacing fuzzifiers, MAX and MIN operators with weighted sum approaches [Masuoka et al 1990]. Compared with traditional neural networks, it has the advantage that all the parameters are designed and no training process is required.

4.1 Traditional Neuro-Fuzzy System

The neuro-fuzzy system in Fig. 10 consists of four layers. The first layer is used for inputs fuzzification, the same process as in classic fuzzy systems. The second layer performs fuzzy variables multiplications, instead of fuzzy logic operations. The multiplication may be helpful to smooth the result surfaces, but also causes more computations. The third and fourth layers perform weighted averages which are similar to the normalization process in TSK fuzzy systems.

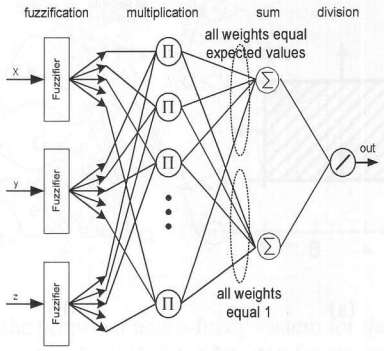


Fig. 10 Architecture of traditional neuro-fuzzy system

Fig. 11 shows the result surface using the traditional neuro-fuzzy system. Even though a smaller approximation error is obtained, the neuro-fuzzy architecture is not recommended because the computation becomes more complex than the classic fuzzy systems.

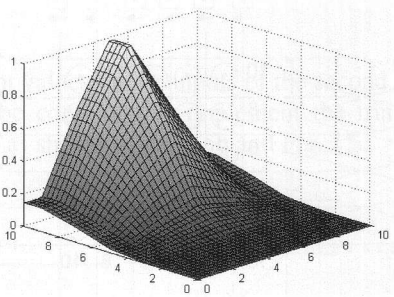


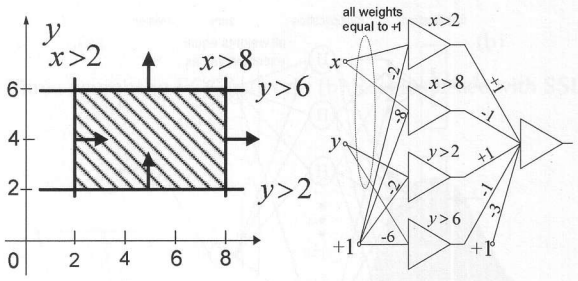
Fig. 11 Result surface obtained using traditional neuro-fuzzy system in Fig. 10; SSE= 1.9320

The architecture in Fig. 10 attempts to present a fuzzy system in a form of neural network. However, it is different from neural network, because the units inside perform signal multiplication or division, rather than activation functions as neurons do.

4.2 Neuro-Fuzzy System without Normalization

In neural systems, a single neuron can separate the input space by line, plane or hyper-plane, depending on the input dimensionality. In order to select a region in *n*-dimension space, more than *(n+1)* neurons should be used.

For example, in order to select a rectangle area in two dimensional space (Fig. 12a), at least *5* neurons are required and the neural network can be designed as shown in Fig. 12b.



(a) (b)

Fig. 12 Area selection using neural networks: (a) desired rectangular area; (b) neural network implementation with step function as activation functions

With this area selection concept, fuzzifiers and fuzzy logic rules (MIN and MAX operators) used for region selection can be replaced by simple neural network architecture, similar as shown in Fig. 12b.

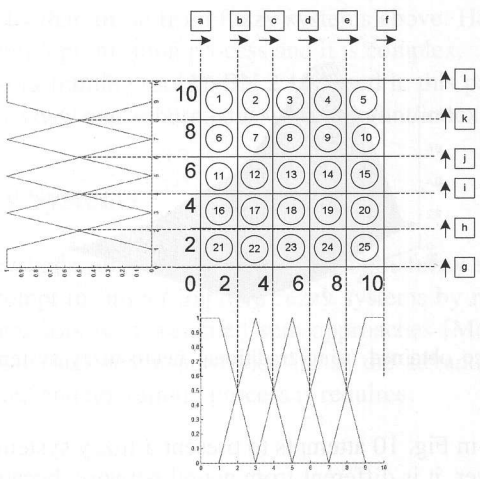


Fig. 13 Two-dimensional input plane separated vertically and horizontally by *6* neurons in each direction, obtained with *25* selection areas

For the given problem, there are two analog inputs and each input has 5 membership functions (Fig. 3). The two fuzzifiers and fuzzy logic rules can be represented by *12* neurons (line *a* to *l*) in the first layer and *25* neurons (area *1* to *25*) in the second layer, as shown in Fig. 13 and Fig. 14.

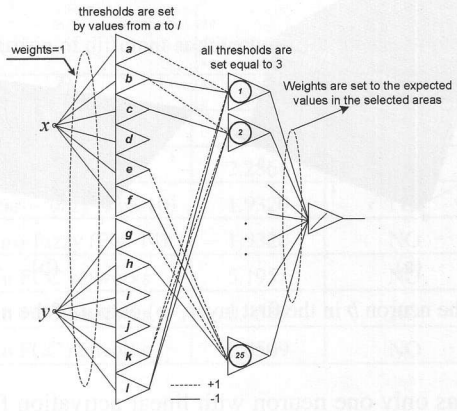


Fig. 14 Architecture of the proposed neuro-fuzzy system for the given problem in Fig. 1

Fig. 14 shows the architecture of the proposed neuro-fuzzy system [Xie et al 2010]. The first layer has *12* neurons, and each neuron presents a straight line, from *a* to *l*. The weight values on the inputs are all *1* and the thresholds of neurons depend on the intersection of the lines. The activation functions of the neurons in the first layer are shown in Fig. 15a, and can be mathematically described by

$$f(x)=\begin{cases} 1 & x \geq b \\ \dfrac{x-a}{b-a} & a < x < b \\ 0 & x \leq a \end{cases} \tag{2}$$

By setting the weight values on the inputs of the second layer to *1* or *-1*, the activation function can be combined to implement the triangular and trapezoidal membership functions, as shown in Fig. 15b and Fig. 15c.

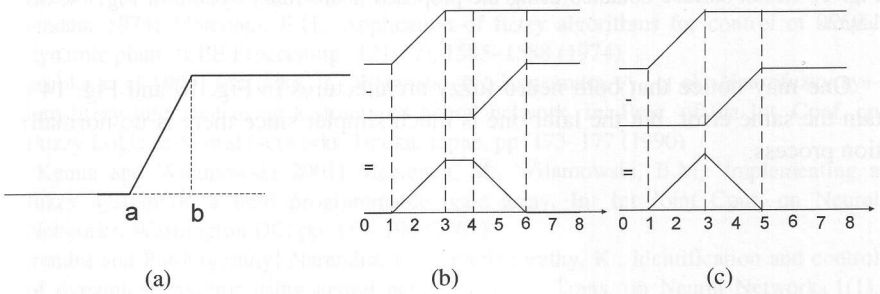
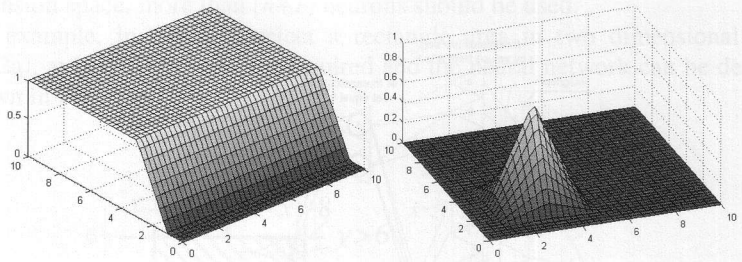


Fig. 15 (a) activation function of each neuron in the first layer; (b) implementation of trapezoidal membership function; (c) implementation of triangular membership function

The second layer has *25* neurons, and each of them presents a selected area as shown in Fig. 13. All the neurons have threshold *3* in the second layer. Fig. 16 below gives the outputs of particular neurons in the first and second layers.



(a) (b)

Fig. 16 (a) output of the neuron *b* in the first layer; (b) output of the neuron *17* in the second neuron

The third layer has only one neuron with linear activation function. The weight values on the inputs of the third layer are set as the expected values in the corresponding areas.

With this architecture, using the area selection strategy in Fig. 13, the result surface for the given problem can be obtained as shown in Fig. 17.

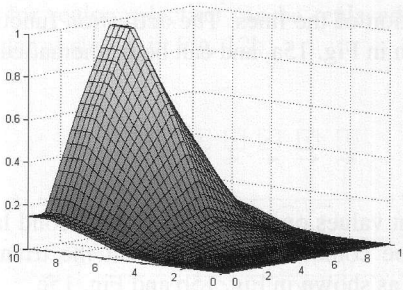


Fig. 17 Result surface obtained using the proposed neuro-fuzzy system in Fig. 14; SSE= 1.9320

One may notice that both neuro-fuzzy architectures in Fig. 10 and Fig. 14 obtain the same error, but the later one is much simpler since there is no normalization process.

5 Discussion and Conclusions

This paper studied the design process of both fuzzy and neural systems, and gave a practical function approximation problem as an example. The comparison results are presented in Table 3.

Table 3 Comparison of different architectures for the given problem in Fig. 1

Architectures	SSE	Normalization	Training
Mamdani fuzzy	6.3555	NO	NO
TSK fuzzy	2.2864	YES	NO
Traditional neuro-fuzzy (Fig. 10)	1.9320	YES	NO
Proposed neuro-fuzzy (Fig. 14)	1.9320	NO	NO
2 neurons in FCC networks	5.1951	NO	YES
3 neurons in FCC networks	0.9589	NO	YES
4 neurons in FCC networks	0.0509	NO	YES

From the comparison results, it can be concluded that fuzzy systems get rough results, but are easier to design; while neural networks can get very precise approximation, but require training process (optimization).

Both neuro-fuzzy architectures (Fig. 10 and Fig. 14) got the same errors in the function approximation problem; however, the architecture in Fig. 14 does not require normalization process and is much simpler than the traditional architecture in Fig. 10.

In the case when only design rules have to be used and optimization is not desired, the neuro-fuzzy architecture in Fig. 14 can replace the classic fuzzy systems and traditional neuro-fuzzy architecture in Fig. 10.

References

[Farrell and Polycarpou 2008] Farrell, J.A., Polycarpou, M.M.: Adaptive approximation based control: unifying neural, fuzzy and traditional adaptive approximation approaches. IEEE Trans. on Neural Networks 19(4), 731–732 (2008)

[Mamdani 1974] Mamdani, E.H.: Application of fuzzy algorithms for control of simple dynamic plant. IEEE Proceedings 121(12), 1585–1588 (1974)

[Masuoka et al 1990] Masuoka, R., Watanabe, N., Kawamura, A., et al.: Neurofuzzy system-fuzzy inference using a structured neural network. In: Proc. of the Int. Conf. on Fuzzy Logic & Neural Networks, Hzuka, Japan, pp. 173–177 (1990)

[McKenna and Wilamowski 2001] McKenna, M., Wilamowski, B.M.: Implementing a fuzzy system on a field programmable gate array. In: Int Joint Conf. on Neural Networks, Washington DC, pp. 189–194 (2001)

[Narendra and Parthasarathy] Narendra, K.S., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. IEEE Trans. on Neural Networks 1(1), 4–27 (1990)

[Takagi and Sugeno 1985] Takagi, T., Sugeno, M.: Fuzzy identification of systems and its application to modeling and control. IEEE Trans. on System, Man, Cybernetics 15(1), 116–132 (1985)

[Wilamowski 2002] Wilamowski, B.M.: Neural networks and fuzzy systems. In: Bishop, R.R. (ed.) Mechatronics Handbook, vol. 33(1), pp. 32–26. CRC Press, Boca Raton (2002)

[Wilamowski and Binfet 1999] Wilamowski, B.M., Binfet, J.: Do fuzzy controllers have advantages over neural controllers in microprocessor implementation. In: Proc. of 2nd Conf. on Recent Advances in Mechatronics, Istanbul, Turkey, pp. 342–347 (1999)

[Wilamowski and Binfet 2001] Wilamowski, B.M., Binfet, J.: Microprocessor Implementation of fuzzy systems and neural networks. In: Int. Joint Conf. on Neural Networks, Washington DC, pp. 234–239 (2001)

[Wilamowski and Yu 2010] Wilamowski, B.M., Yu, H.: Improved computation for Levenberg Marquardt training. IEEE Trans. on Neural Networks 21(6), 930–937 (2010)

[Xie et al 2010] Xie, T.T., Yu, H., Wilamowski, B.M.: Replacing fuzzy systems with neural networks. In: Proc. IEEE Conf. on Human System Interaction, Rzeszow, Poland, pp. 189–193 (2010)

[Yu and Wilamowski 2009] Yu, H., Wilamowski, B.M.: Efficient and reliable training of neural networks. In: Proc. 2nd IEEE Human System Interaction, Catania, Italy, pp. 109–115 (2009)