

Comparison of fuzzy function approximators

Andri Riid and Ennu Rüstern

Department of Computer Control, Tallinn Technical University, Ehitajate tee 5, EE0026 Tallinn, ESTONIA;
e-mail: andri@dcc.ttu.ee

ABSTRACT: This paper describes three universal fuzzy function approximators - fuzzy template modeling, ANFIS and NEFPFOX and compares them on the basis of their capability to model a fuzzy system.

Keywords: fuzzy modeling, identification, neuro-fuzzy systems.

1. Introduction

Most of the contemporary control methods assume the existence of the exact mathematical model of the process or system. In many cases such model is difficult or too expensive to obtain. On the other hand, fuzzy logic can be a convenient tool for solving a problem without having to analyze the control object in great detail. Such a solution is certainly rough, but it can be obtained sufficiently quickly, easily and at low cost.

The main advantage of fuzzy systems is their way of knowledge representation: the if-then rules with fuzzy antecedents and consequents express the nature of the process/system in the linguistic form and can be analyzed without effort to derive the control strategy.

2. Fuzzy systems

Fuzzy model of the process is usually constructed on the basis of human knowledge about the process/system and/or data gathered from the experiments made with control object. When both sources are used, human experience is usually employed to determine the structure and the initial settings of the model - variables, their membership functions, and fuzzy rules. The model is then tuned to make it match the experimental data.

There are basically two means of tuning the fuzzy systems: 1) adding and deleting the rules and 2) adjusting of the parameters of the membership functions of the input-output variables. While the former is usually used for rough tuning, the latter can be used for the fine tuning of the model.

Several authors have also introduced the rule weights (a rule weight is interpreted as the degree of its importance) for tuning the fuzzy model. Fuzzy template modeling, proposed by Yager and Filev in [1], uses this

feature. In [2] we used this method to extract the control algorithm for the fed-batch fermentation.

Although the results were successful, unfortunately this method has drawbacks - the modeling procedure is largely based on human intuition and knowledge about the process.

Human expert is required to define the input-output variables and their fuzzy partitions. The validity of the so-called elemental rules defined by these partitions is tested with experimental data and is expressed by rule weights.

The quality of the model is, on the other hand, strongly influenced by how well data selected for adjusting the rule weights reflects the nature of the process, but this problem is very general in the field of modeling [3] and can therefore be considered not method-specific.

3. Fuzzy template modeling

Let us now look closely at the fuzzy modeling method proposed by Yager and Filev in [1] expanded to MIMO models for [2].

A given input-output reading $(x_1^k, x_2^k, \dots, x_s^k, y_1^k, y_2^k, \dots, y_t^k)$, where $x_1^k, x_2^k, \dots, x_s^k$ are the values of the inputs U_1, U_2, \dots, U_s at the moment k ($k = (1, K)$), correspondingly and $y_1^k, y_2^k, \dots, y_t^k$ are the values of the outputs V_1, V_2, \dots, V_t at the same moment, matches the elemental rule

IF U_1 is $B_{i_1}^1$ **AND** ... **AND** U_s is $B_{i_s}^s$

THEN V_1 is $D_{j_1}^1$ **AND** ... **AND** V_t is $D_{j_t}^t$,

if all the firing strengths $\tau_{i_1}^1 = B_{i_1}^1(x_1^k), \dots, \tau_{i_s}^s = B_{i_s}^s(x_s^k)$

$(i_1 = (1, s_1), \dots, i_t = (1, s_s))$ and $\gamma_{j_1}^1 = D_{j_1}^1(y_1^k), \dots, \gamma_{j_t}^t =$

$D_{j_t}^t(y_t^k)$ ($j_1 = (1, t_1), \dots, j_t = (1, t_t)$) are nonzero. The degree of matching is defined as follows:

$$\delta_{i_1 i_2 \dots i_s j_1 j_2 \dots j_t}(k) = \tau_{i_1}^1 \cdot \tau_{i_2}^2 \cdot \dots \cdot \tau_{i_s}^s \cdot \gamma_{j_1}^1 \cdot \gamma_{j_2}^2 \cdot \dots \cdot \gamma_{j_t}^t. \quad (1)$$

One input-output reading can have a nonzero degree of matching to more than one elemental rule. This is taken into account by a normalized degree of matching $\nu_{i_1 i_2 \dots i_s j_1 j_2 \dots j_t}(k)$, obtained by the normalization of the fuzzy degree of matching (1) with respect to the total

degree of matching of the k^{th} data reading with all the elemental rules

$$v_{i_1 i_2 \dots i_s j_1 j_2 \dots j_t}(k) = \frac{\delta_{i_1 i_2 \dots i_s j_1 j_2 \dots j_t}(k)}{\sum_{i_1=1}^{s_1} \dots \sum_{i_s=1}^{s_s} \sum_{j_1=1}^{t_1} \dots \sum_{j_t=1}^{t_t} \delta_{i_1 i_2 \dots i_s j_1 j_2 \dots j_t}(k)}. \quad (2)$$

The next step will be computing the degree of matching of an elemental rule with respect to the whole input-output data set

$$v_{i_1 i_2 \dots i_s j_1 j_2 \dots j_t} = \sum_{k=1}^K v_{i_1 i_2 \dots i_s j_1 j_2 \dots j_t}(k). \quad (3)$$

The total normalized degree of matching of the $i_1 i_2 \dots i_r^{\text{th}}$ rule package that consists of rules with the equivalent antecedents is obtained by

$$v_{i_1 i_2 \dots i_s} = \sum_{j_1=1}^{t_1} \sum_{j_2=1}^{t_2} \dots \sum_{j_t=1}^{t_t} v_{i_1 i_2 \dots i_s j_1 j_2 \dots j_t}. \quad (4)$$

The outcome of modeling is the rulebase $R(B_{i_1}^1, \dots, B_{i_s}^s, D_{j_1}^1, \dots, D_{j_t}^t)$, consisting of $i_1 \cdot i_2 \cdot \dots \cdot i_s \cdot j_1 \cdot j_2 \cdot \dots \cdot j_t$ rules with the weights:

$$p_{i_1 i_2 \dots i_s j_1 j_2 \dots j_t} = \frac{v_{i_1 i_2 \dots i_s j_1 j_2 \dots j_t}}{v_{i_1 i_2 \dots i_s}}. \quad (5)$$

An implementation of this algorithm was prepared for our previous work by means of MATLAB.

4. Neuro-fuzzy systems

Besides the method developed by Yager and Filev [1] there exist several other ones. Most of these combine neural networks and fuzzy logic. For neural networks several learning algorithms are available - it can adapt its parameters by learning, but it is non-transparent to interpretation (black box) and learning must start from the scratch because prior knowledge cannot be incorporated into the structure of a neural network. On the other hand, fuzzy systems are easily interpreted and implemented, prior knowledge can be used but they cannot learn and tuning is often complicated.

Therefore, the most important reason for combining fuzzy systems with neural networks is their learning capability, but neural network learning algorithms are usually gradient descent methods and cannot be applied if the inference process is not differentiable. One solution to this problem is to replace the functions used in the fuzzy systems by differentiable functions, another is to use alternative learning algorithms.

4.1 ANFIS model

ANFIS introduced by Jang is defined with differentiable operators of the inference mechanism (product and sum), differentiable membership functions (Gaussian), and

weighted mean defuzzification (Takagi-Sugeno-Kang fuzzy model). The inference scheme of such systems can be represented as a special five-layer feedforward network (Fig. 1).

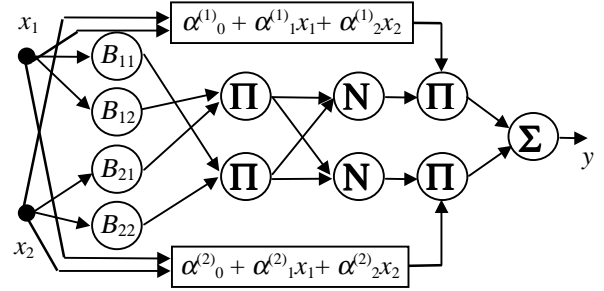


Fig. 1. ANFIS model.

Each unit in the first layer (inputs are not considered as a separate layer) stores the parameters of a membership function B_{ij} and performs the operation

$$B_{ij}(x_i) = \frac{1}{1 + \left(\frac{x_i - c}{a} \right)^2}^b. \quad (6)$$

In the second layer, the rule antecedents will be aggregated using the product operation: $\tau_r = B_{i1}(x_1) \cdot B_{i2}(x_2) \cdot \dots \cdot B_{in}(x_n)$.

In the third layer, the relative degree of fulfilment of a rule will be computed by

$$\bar{\tau}_r = \frac{\tau_r}{\sum_{i=1}^R \tau_i}. \quad (7)$$

Each unit in layer 4 computes the output of a rule R_r by $o_r = \bar{\tau}_r \cdot (\alpha_0^{(r)} + \alpha_1^{(r)} x_1 + \dots + \alpha_n^{(r)} x_n)$. By summing all the outputs of the previous layer, we obtain the output y in the output layer. The expansion of ANFIS to MIMO systems is straightforward.

Because ANFIS uses only differentiable functions, it is possible to apply standard neural network learning procedures. Here, a mixture of backpropagation (gradient descent) and least squares estimation (LSE) is used. Backpropagation is used to learn the antecedent parameters, and LSE is used to determine the coefficients of the linear combinations in the rule consequents. ANFIS appears to be one of the well-known methods described in detail in [1,4,5].

ANFIS requires the antecedent membership functions and fuzzy rules to be defined prior to the training. It is also possible to use fuzzy clustering methods (fuzzy c-means clustering, mountain clustering), to initialize the neuro-fuzzy system. In this case, only the number of antecedent membership functions should be specified before training.

A version of ANFIS is implemented through the fuzzy toolbox available for MATLAB.

4.2 NEFPROX model

The NEFPROX model (introduced by Nauck et al. in [4]) is based on the generic fuzzy perceptron (Fig. 7). The units in this network use t-norms or s-norms instead of the activation functions normally used in neural networks. The first layer represents the input variables, the second (or hidden) layer represents fuzzy rules and the third layer represents output variables. Fuzzy sets are encoded as connection weights $W(x, R)/W(R, y)$. The weights that represent the same membership functions have identical values.

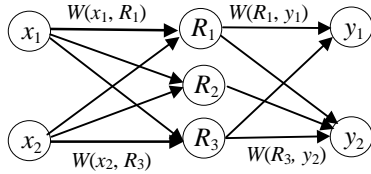


Fig. 2. Generic fuzzy perceptron.

This fuzzy system has very few restrictions - practically all s- and t-norms and types of membership functions, either mean of maxima or center of gravity defuzzification can be used and the system can be initialized with prior knowledge. Gradient descent methods cannot be used here, and the learning algorithm is therefore a simple heuristic procedure, divided into two stages - learning fuzzy rules (structure learning) and learning fuzzy sets (parameter learning).

The incremental rule learning algorithm creates a rulebase from scratch if necessary by adding rule after rule. From each input-output reading, the fuzzy sets that provide the highest membership degree are taken in order to construct a rule. In the second phase, the rulebase is optimized by changing the consequent to an adjacent membership function if necessary.

Parameter learning results in shifting the membership functions and making their supports larger or narrower. On the basis of the rule fulfilment, the algorithm determines whether the activation of a rule unit must be increased or decreased and identifies the fuzzy sets that are responsible for the current rule activation.

Output fuzzy sets will be shifted to the target and made narrower if the target values have a high membership degree. Depending on the rule error, an input fuzzy set is moved towards the current input and is made wider or is shifted away and made narrower. The rule error is larger for a rule fulfilment near 0.5.

There is already an implementation of NEFPROX (although in beta stadium) freely available via the website at <http://fuzzy.cs.uni-magdeburg.de>

5. System

We defined a simple fuzzy SISO system to compare and evaluate the fuzzy modeling approaches described above. The input variable $x = [0, 10]$ and output variable $y = [-5, 5]$ are partitioned into 6 and 5 membership functions mf_i ,

respectively. Figure 3 presents these membership functions, fuzzy rules and the relationship between input and output of the resulting system. The purpose of selecting such membership functions and rulebase was to create a function that would not be too easy to approximate.

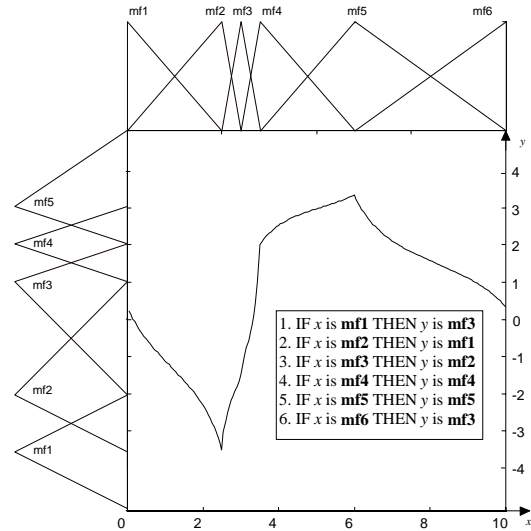


Fig. 3. Sample fuzzy system.

6. Modeling results.

First, we model the system with the Yager-Filev method by using a) symmetrical triangular membership functions with equal supports and overlap degrees (0.5) and b) the fuzzy partition of the original system.

Selection of the training data is simple, because whole operating range of the system can easily be covered. The resulting models (with rules with weights less than 0.1 removed for the sake of clarity) are given in Table 1 and 2, correspondingly.

Table 1

| Input/output | mf1 | mf2 | mf3 | mf4 | mf5 |
|--------------|-------|-------|-------|-------|-------|
| mf1 | | 0.414 | 0.568 | | |
| mf2 | 0.248 | 0.562 | 0.149 | | |
| mf3 | | 0.111 | 0.241 | 0.309 | 0.437 |
| mf4 | | | | 0.331 | 0.669 |
| mf5 | | | 0.124 | 0.789 | |
| mf6 | | | 0.487 | 0.513 | |

Table 2

| Input/output | mf1 | mf2 | mf3 | mf4 | mf5 |
|--------------|-------|-------|-------|-------|-------|
| mf1 | | 0.243 | 0.377 | | |
| mf2 | 0.603 | 0.658 | | | |
| mf3 | | 0.717 | 0.241 | | |
| mf4 | | | 0.103 | 0.330 | 0.520 |
| mf5 | | | 0.140 | 0.342 | 0.516 |
| mf6 | | | 0.644 | 0.290 | |

The model response to the input in the range $[0, 10]$ compared to the original system is depicted in Fig. 4. We can see that the performance in both cases is quite poor. Model 2 (normal line) is closer to the original system

(dashed line) than model 1 (bold line), however not remarkably, and the fact that even using the partition that reflects the nature of the process well, does not result in a good model, makes the applicability of the whole method questionable. Another problem of this method is rule weights themselves that destroy the semantics of a fuzzy system and make the interpretation of the system difficult. By using a smaller overlap degree or trapezoid membership functions instead of triangular should lead to higher overall rule weights. When analyzing the model, only the rules with high weights can be taken into account, rules with low weights may lead to wrong decisions.

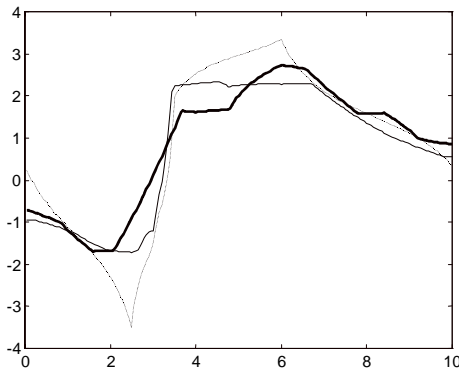


Fig. 4. Modeling with the Yager-Filev method.

For ANFIS, six membership antecedent functions were defined, using subtractive clustering (Fig. 5).

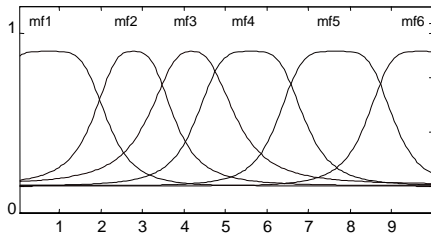


Fig. 5. ANFIS antecedent membership functions.

The results of the training are depicted in Fig. 6.

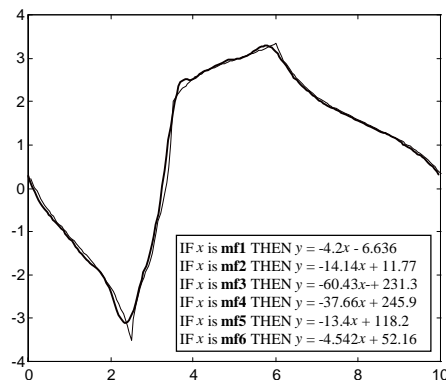


Fig. 6 Modeling with ANFIS.

The performance of the ANFIS system, combined with subtractive clustering, is nearly perfect in our case. That is due to the large number of adjustable parameters and

analytical learning rules. Interpretation of Sugeno rules is, however, complicated and ANFIS can therefore be used successfully only when the performance of the model rather than understanding the nature of the process is important.

Finally, applying NEFPROX results in the model shown in Fig.7. Similarly to the template modeling, this method is unable to reconstruct the original model, but now we are free of rule weights and the model response is much closer to the original system.

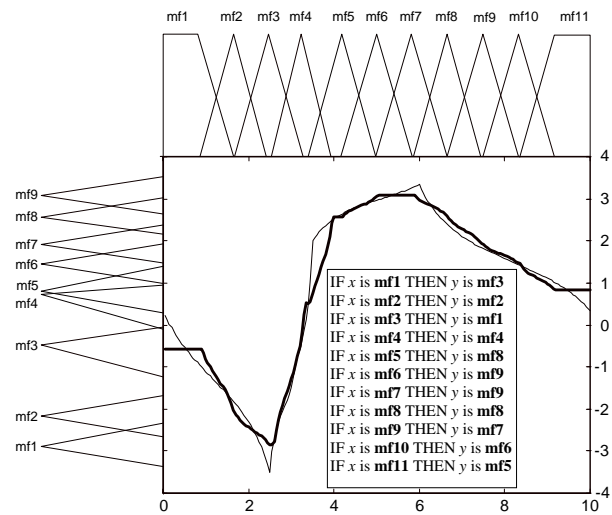


Fig. 7. Modeling with NEFPROX.

Similarly to ANFIS, the performance of NEFPROX is strongly influenced by the number of membership functions that should be specified prior to the training. In contrast to ANFIS, where increasing the number of membership functions generally leads to better results, here only a certain combination gave reasonable modeling error. If that feature is truly in the nature of NEFPROX or just a property of the current software application is not yet clear.

However, the NEFPROX-like approach is very promising in every sense and can be seriously considered when modeling for control is the topic.

References

- [1] R.Yager and D.Filev, *Essentials of Fuzzy Modeling and Control*. John Wiley and Sons, New York, 1994.
- [2] A.Riid and E.Rüstern, "Fuzzy modeling and control of fed-batch fermentation". Proc. 9th Int. Symp. on System-Modelling-Control, Zakopane, Poland, 1998 (to be published).
- [3] Y-Z. Lu, *Industrial Intelligent Control*. John Wiley and Sons, New York, 1996.
- [4] D.Nauck, F.Klawonn and R.Kruse, *Foundations of Neuro-fuzzy Systems*. John Wiley and Sons, New York, 1997.
- [5] L.Tsoukalas and R. Uhrig, *Fuzzy and Neural Approaches in Engineering*. John Wiley and Sons, New York, 1997.