XFBPA: A TOOL FOR AUTOMATIC FUZZY SYSTEM LEARNING

F. J. Moreno†, D. R. López††, A. Barriga†, S. Sánchez Solano†

†Instituto de Microelectrónica de Sevilla - Centro Nacional de Microelectrónica
††Centro Informático Científico de Andalucía
Avda. Reina Mercedes, s/n (Edif. CICA)
E-41012, Sevilla, Spain

5th European Congress on Intelligent Techniques and Soft Computing (EUFIT’97),

This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author’s copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.
ABSTRACT: This paper introduces the main features of XFBPA, a tool that provides automatic learning capabilities for fuzzy systems, integrated into the fuzzy development environment Xfuzzy. Its main achievement is the support for the learning of complex systems, preserving the flexibility in system description provided by Xfuzzy.

1 INTRODUCTION

Since it was first introduced by Zadeh in 1965, fuzzy logic has been successfully employed in several application domains, such as control systems, pattern recognition, diagnostic systems or non-linear systems modelling [Schwartz 1994]. The success of its application to this wide range of fields is based on the ability of fuzzy logic to express system behavior in a simple way, close to the manner in which knowledge is expressed by natural language.

One of the main difficulties that arise when defining a fuzzy system is the tuning of its behavior. This behavior is described in terms of a rule base and a set of membership functions that represent the atoms employed in the propositions that rules define. The rulebase specifies the underlying logic of system behavior and, apart from unforeseen redesigns, is not usually intended to be changed by the tuning process. Fuzzy system tuning is performed basically on the description of membership functions and, more precisely, deals with the modification of the parameters defining these functions. Since it has to be performed simultaneously over a great number of parameters which play a role in the system definition, manual procedures are only applicable to very simple systems. The use of automated tuning tools is necessary for any system of a reasonable complexity.

Different learning algorithms for fuzzy systems, most of them originally developed for neural networks, have been described in the last few years [Lin 1991][Wang 1992][Khan 1993]. However, the implementations of these algorithms impose serious limits on the systems to be tuned and to the learning process itself. These limits include a very small number of system variables (in general, one single output and no more than four inputs), a reduced number of membership functions per variable (no more than seven in the best cases), the use of very few classes of membership functions, fuzzy connectives and defuzzification methods, and the lack of flexibility in the selection of those parameters to be learned.

When we started to work on the new version of our fuzzy system development tool Xfuzzy [Barriga 1996], one of our goals was the implementation of a learning method not constrained by the limits mentioned above. A classical supervised learning algorithm, backpropagation, was selected for this task. This paper presents the results of this effort: XFBPA, a tool for fuzzy system learning which preserves the flexibility in system definition supported by Xfuzzy and fully integrated in a graphical environment for the design, verification and synthesis of fuzzy systems.

The next section introduces the general characteristics of XFBPA and describes the features of the learning process that it performs. Section 3 includes a set of examples that show the capabilities of the tool in the study and evaluation of different fuzzy connectives and defuzzification methods. Finally, section 4 presents some conclusions and issues for future work.

2 A GENERAL DESCRIPTION OF XFBPA

XFBPA is a tool for the automatic learning of fuzzy systems described using XFL (the specification language used by Xfuzzy) [López 1997]. The tool is based in the backpropagation algorithm, which essentially attempts the minimization of an error function which is assumed to express the deviation of the system from its ideal behavior until a (possibly local) minimum is reached. In each iteration of the learning loop an increment is applied to any parameter under tuning, \( p \). This incre-
ment is defined by:

\[ \Delta p = -\eta \frac{\partial E}{\partial p} \]

where \( \eta \) is the learning rate and \( E \) is the error function. Several types of learning are supported by XFBPA. In the first (type 1) \( \eta \) is constant while in the second (type 2):

\[ \eta = \frac{S}{\sqrt{\sum_j \left( \frac{\partial E}{\partial p_j} \right)^2}} \]

where the constant \( S \) (step size) defines the modulus of the shift vector in the system parameters space [Jang 1992]. In order to speed-up the learning process several variants of backpropagation algorithms are implemented in the tool (quick-prop [Fahlman 1988], R-prop [Riedmiller 1992], and the manhattan algorithm).

The intended (ideal) behavior of the system is described by means of a set of input-output data, thus the error function is obtained by comparing the actual output of the system for a set of inputs with the corresponding ideal values. The error function used by XFBPA is:

\[ E = \frac{1}{r} \sum_{i=1}^{r} \sum_{j=1}^{n} w_j \left( \frac{y_{ji} - Y_{ji}}{\text{range}_j} \right)^2 \]

where \( r \) is the number of training samples, \( n \) the number of outputs of the system (XFBPA allows a system to have an indefinite number of outputs), \( w_j \) is a weight assigned to the \( j \)th output (it defaults to \( 1/n \)), \( y_{ji} \) is the actual value generated by the system for the \( j \)th output when receiving the \( i \)th training input, \( Y_{ji} \) is the ideal value of the \( j \)th output for the \( i \)th input, and \( \text{range}_j \) is the range of values for the \( j \)th output. This error function represents the weighted quadratic deviation of the outputs normalized with respect to their respective ranges. Its value is always bounded between 0 and 1. It is possible to adjust the error function itself, giving higher weights to those output variables that are supposed to be more significant in system deviation. Moreover, XFBPA can prune those rules made obsolete by the learning process. A rule is considered obsolete (and therefore pruned) when its activation level never reaches a specific threshold, which can be specified by the user. If rule pruning is performed at least once, the main learning loop is repeated, since the modification introduced to the rulebase may imply changes in the final results of the learning process.

The tool does not impose any limit on the number of inputs or outputs of the system, nor in its rulebase complexity. Rule antecedents may contain any combination of terms, employing connectives of conjunction, disjunction and negation. XFBPA can be applied to systems using a wide range of fuzzy connectives, which can be arbitrarily selected by the user. Derivatives for these connectives are calculated by the runtime library of the tool. It supports all the classes of membership functions defined by XFL, from triangular to pointwise linear approximations of arbitrary functions, and the set of supported T-norms, T-conorms and implication functions goes far beyond sum-product and max-min combinations, covering many other functions described in the literature. With respect to defuzzification methods, XFBPA can be used either with systems using methods which sweep the universe of discourse of output variables (like the Center of Area method) or with those intended to reduce calculations by employing rule activation and membership function parameters (the so called simplified methods, like Fuzzy Mean).

XFBPA allows the user to select which system parameters are going to be tuned by the learning process. The specification of the parameters is done by means of a text file whose lines contain the tuning state for a parameter or a set of parameters. Each of these lines contains two fields: a specification of the parameter in terms of the XFL type used, the membership function class and the parameter identification inside the membership function definitions, and a tuning state, which defines if the referenced parameter(s) is(are) going to be tuned or not. By default, XFBPA tunes all parameters.

The learning algorithm requires the definition of an end condition to stop the tuning loop. The tool offers four different choices to define this end condition:

- maximum number of iterations
- minimum value for the mean deviation, \( m = \sqrt{E} \)
- minimum value for the maximum deviation, \( M = \max \left( \frac{y_{ji} - Y_{ji}}{\text{range}_j} \right) \)
- minimum value for the relative error increment, \( e = \frac{E(t) - E(t-1)}{E(t-1)} \)
The learning process can be performed by calling XFBPA from the command line or by means of a window integrated into the Xfuzzy graphical interface. Fig. 1 shows the appearance of this window, composed of four different areas: the upper left area allows the configuration of the learning algorithm parameters. The upper right shows information about the status of the learning process, while the third provides a graphical representation of the mean and maximum deviations with respect to the number of iterations of the learning loop. The last area is a menu bar with control commands. This interface not only allows the user to configure and execute the learning process, but also to dynamically change its configuration and to monitor its evolution.

3 APPLICATIONS

The wide range of connectives, implication functions and defuzzification methods supported by XFBPA permits a great number of combinations among them. This implies that its results are reliable when used in comparisons between different approaches, since only optimal systems are employed, thus avoiding errors introduced by misconfiguration of system parameters. To show the results that can be obtained in this area using XFBPA we have chosen two reference functions widely employed in the literature [Rovatti 1996]:

\[
\begin{align*}
\text{example 1: } f(x, y) &= \frac{1}{1 + e^{10(x-y)}}; \\
\text{example 2: } f(x, y) &= \frac{1}{2} \cdot (1 + \sin(2\pi x) \cdot \cos(2\pi y));
\end{align*}
\]

The tests were carried out using systems with seven membership functions per input variable, initially distributed in an uniform coverage of the range of each variable. Two classes of membership functions were considered: triangular and gaussian bell functions, while the conjunction operators selected were minimum and product, and the defuzzification methods evaluated were Fuzzy Mean and Weighted Fuzzy Mean. To specify the ideal behavior of both systems sets of 625 training samples were employed. In both cases, system rulebases were composed of 49 rules, whose consequent values were initially assigned to an intermediate value, i.e., with a plain surface. We have chosen in all cases the same learning strategies, based in three steps. In the first one, learning was performed on consequent parameters. The second step carried out a tuning of antecedent parameters with the values obtained for consequents in the previous step fixed. In the third step a further consequent parameters refinement was enforced.

Table 1 shows the results obtained for the RMSE and maximum deviation in the two examples described above for each combination of membership function classes, fuzzy connectives and defuzzification methods. Fig. 2 shows some examples of the membership functions obtained after the learning process. For each case we include the results obtained for both inputs and the output of the system. It is worth noting that the initial regular distribution for the antecedents is (approximately) preserved by the learning process. In the case of consequents, the initial 49 conclusion singletons tend to be comprised into five well defined groups, suggesting a reduction of rulebase complexity. Fig. 3 provides a graphical representation of the examples, including the target surfaces, some learned surfaces for both examples and a plot of the corresponding error surfaces.

![Fig. 1: Xfuzzy learning window](image-url)
4 CONCLUSIONS

We have introduced here the main characteristics of XFBPA, a tool for automatic fuzzy system learning, based on back-propagation algorithms. Its main features are the ability to deal with complex systems, the wide range of fuzzy functions that can be used with and its integration into a development environment (Xfuzzy) which supports the definition, tuning, simulation and synthesis of fuzzy systems. We have also shown the applicability of this tool with some examples that demonstrate its use in the evaluation of optimal systems using different choices for fuzzy connectives and defuzzification methods.

The results obtained with XFBPA have allowed our group to open new tasks focused in the implementation of more powerful learning algorithms and in the support for new fuzzy operations in the learning process. XFBPA constitutes a first step in the development of learning tools for the Xfuzzy environment.

REFERENCES


<table>
<thead>
<tr>
<th></th>
<th>example 1</th>
<th>example 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FM</td>
<td>WFM</td>
</tr>
<tr>
<td>min Bell</td>
<td>0.96% // 5.11%</td>
<td>0.85% // 2.54%</td>
</tr>
<tr>
<td>Triangle</td>
<td>1.24% // 4.16%</td>
<td>1.62% // 7.92%</td>
</tr>
<tr>
<td>prod Bell</td>
<td>1.15% // 6.06%</td>
<td>0.76% // 2.37%</td>
</tr>
<tr>
<td>Triangle</td>
<td>0.85% // 3.09%</td>
<td>0.70% // 9.45%</td>
</tr>
</tbody>
</table>

Table 1: RMSE // Maximum deviation obtained using min and product conjunction operators, Fuzzy Mean (FM) and Weighted Fuzzy Mean (WFM) defuzzification, and bell and triangle membership shapes.

Example 1: Bell-Product-WFM

Example 2: Triangle-Product-WFM

Fig. 2: Input and output membership functions for some learning results.


<table>
<thead>
<tr>
<th>Target surface</th>
<th>Learned surface</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Target surface" /></td>
<td><img src="image2" alt="Learned surface" /></td>
<td><img src="image3" alt="Errors" /></td>
</tr>
</tbody>
</table>

**Bell-Product-FM**

**Triangle-Product-WFM**

**example 1:** \( f(x, y) = \frac{1}{1 + e^{10(x-y)}} \)

**example 2:**
\[
 f(x, y) = \frac{1}{2} \cdot (1 + \sin(2\pi x) \cdot \cos(2\pi y))
\]

**Fig. 3:** Target, learned and error surfaces.