2006 IEEE International Conference on Fuzzy Systems
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Separated Antecedent and Consequent Learning for Takagi-Sugeno Fuzzy Systems

János Botzheim, Edwin Lughofer, Erich Peter Klement, László T. Kóczy, Tamás (Tom) D. Gedeon

Abstract— **In this paper a new algorithm for the learning of Takagi-Sugeno fuzzy systems is introduced. In the algorithm different learning techniques are applied for the antecedent and the consequent parameters of the fuzzy system. We propose a hybrid method for the antecedent parameters learning based on the combination of the *Bacterial Evolutionary Algorithm (BEA)* and the *Levenberg-Marquardt (LM)* method. For the linear parameters in fuzzy systems appearing in the rule consequents the *Least Squares (LS)* and the *Recursive Least Squares (RLS)* techniques are applied, which will lead to a global optimal solution of linear parameter vectors in the least squares sense. Therefore a better performance can be guaranteed than with a complete learning by BEA and LM. The paper is concluded by evaluation results based on high-dimensional test data. These evaluation results compare the new method with some conventional fuzzy training methods with respect to approximation accuracy and model complexity.**

Index Terms— **bacterial evolutionary algorithm, Levenberg-Marquardt method, Takagi-Sugeno fuzzy systems, (recursive) least squares**

I. Introduction

One of the crucial problems of fuzzy rule based modelling is how to find an optimal or at least a quasi-optimal rule base for a certain system. In most applications there is no human expert available, thus some automatic method to determine the fuzzy rule base must be deployed. Some of these methods inspired by evolutionary processes can be found in nature. The *Bacterial Evolutionary Algorithm (BEA)* [1] [2] is one of the most recent approaches in this field, other methods include constrained optimization techniques [3] or clustering methods [4]. Apart from these methods, in the area of neural networks there are training algorithms known and these could be applied to fuzzy systems as well. They prove to be useful when there are only quantitative data available. Neural networks can learn these data and have the capability to generalize. In training, the objective is to tune the membership functions in the fuzzy system so that the system approximates a desired dependency between some input and output variables in a quite accurate way. For this task, the *Levenberg-Marquardt* method [5] [6] [7] was proposed, which is a gradient-based training algorithm. The gradient-based algorithms are local searchers, thus, these methods often find only the local optimum. However, they can be used to improve the performance of the evolutionary algorithm, which may find the global optimum with sufficient precision in this way. The combination of the bacterial and the *Levenberg-Marquardt* technique was introduced in [8]. In that paper, the Mamdani fuzzy system was used for system modelling. Beside this model, the Takagi-Sugeno fuzzy systems are also widely used, as they possess some advantages compared with the Mamdani approach for instance, the universal approximation property [9], a fast inference mechanism as defuzzification is not needed and linear hyperplanes as consequent functions, entailing well interpretable insight for local control behaviors. Because the structure of the parameters is different in the antecedent and in the consequent parts, different methods can be applied for the two kinds of parameters. We propose the combination of the bacterial approach and the *Levenberg-Marquardt* method for the learning of the nonlinear antecedents' parameters, and least-square techniques for the linear rule consequents' parameters as shown in Figure 1. The latter is applied due to the fact that it delivers a global optimum of linear parameters in the least squares sense and hence is favorable among algorithms which can be stuck in local minima.

The paper is organized as follows. Section 2 describes the structure of the fuzzy system and explains our goals. The role of the bacterial evolutionary algorithm is described in Section 3. Section 4 presents the antecedent learning, while the consequent learning is introduced in Section 5. Simulation results are shown in Section 6. Section 7 draws some conclusions.

János Botzheim is with the Department of Telecommunication and Media Informatics, Budapest University of Technology and Economics, Hungary & Dept. of Computer Science, The Australian National University (email: botzheim@tmit.bme.hu)
Edwin Lughofer and Erich Peter Klement are with the Department of Knowledge-based Mathematical Systems, Johannes Kepler University of Linz, A-4040 Linz, Austria (e-mail: {edwin.lughofer,ep.klement}@jku.at).
László T. Kóczy is with the Department of Telecommunication and Media Informatics, Budapest University of Technology and Economics & Széchenyi István University, Győr, Hungary (email: koczy@tmit.bme.hu).
Tamás (Tom) D. Gedeon is with the Dept. of Computer Science, The Australian National University (email: tom.gedeon@anu.edu.au)

II. Problem Statement

A Takagi-Sugeno fuzzy system with multiple input variables $(x_1, ..., x_p)$ and a single output variable y can be generally defined in the following way:

$$\hat{f}(\vec{x}) = \hat{y} = \sum_{i=1}^{R} l_i \Psi_i(\vec{x}) \tag{1}$$

where

$$\Psi_i(\vec{x}) = \frac{\mu_i(\vec{x})}{\sum_{j=1}^{R} \mu_j(\vec{x})} \tag{2}$$

are called *normalized membership functions*, which normalize the degrees of rule fulfillment by using a t-norm, i.e.

$$\mu_i(\vec{x}) = \top_{j=1}^{p} \mu_{ij}(x_j) \tag{3}$$

where x_j is the j-th component in current data vector, hence reflecting the value of the j-th variable and μ_{ij} the membership degree of x_j to the fuzzy set describing the j-th premise part of the i-th rule. Of course, the j-th premise part always belongs to the j-th dimension, for don't care parts $\mu_{ij}(x_j)$ is simply set to 1. The symbol $\top$ denotes a t-norm in general. The l_i's are the so-called consequent functions of the R rules and are defined by:

$$l_i = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + ... + w_{ip}x_p \tag{4}$$

Arbitrary types of fuzzy sets can be chosen depending on the special task and behaviour of the fuzzy system. A widely used fuzzy set type are the Gaussian functions defined by:

$$\mu_{ij}(x_j) = e^{-\frac{1}{2}\frac{(x_j - c_{ij})^2}{\sigma_{ij}^2}}$$

where c_{ij} denotes the center and σ_{ij}^2 denotes the variance of the Gaussian function appearing in the j-th premise part of the i-th rule. The product t-norm together with Gaussian functions leads to some favorable properties, namely steady differentiable models, equivalency to radial basis function neural networks [10], [11], favorable interpolation properties due to infinite support and an easy extraction of the parameters for the Gaussian fuzzy sets (which is for instance not the case for sigmoidal functions, which also possess infinite support). However, due to its wide popularity trapezoidal fuzzy sets together with the minimum t-norm will also be used for all learning schemes and evaluation tests in the subsequent sections. These fuzzy sets are defined in the following way:

$$\mu_{ij}(x_j) = \begin{cases} \frac{x_j - a_{ij}}{b_{ij} - a_{ij}} & \text{if } a_{ij} < x_j < b_{ij} \\ 1 & \text{if } b_{ij} \le x_j < c_{ij} \\ \frac{d_{ij} - x_j}{d_{ij} - c_{ij}} & \text{if } c_{ij} \le x_j < d_{ij} \\ 0 & \text{otherwise} \end{cases}$$

where for the breakpoints the equational chain $a_{ij} \le b_{ij} \le c_{ij} \le d_{ij}$ holds. A comparison between these two choices of fuzzy set/t-norm connections will be also carried out in Section VI.

The goal is now to train non-linear antecedent parameters as well as linear consequent parameters in a hybrid and iterative manner, in order to gain good approximation accuracy with reasonable model complexity. The proposed learning scheme is demonstrated in Figure 1, whose components together with their role will be described in detail in the subsequent sections. Here it should be noticed that within the initialization component only the antecedent parameters of the fuzzy systems are initialized, i.e. in order to start the antecedent learning scheme, the consequent parameters have to be trained for this initial setting. This is because

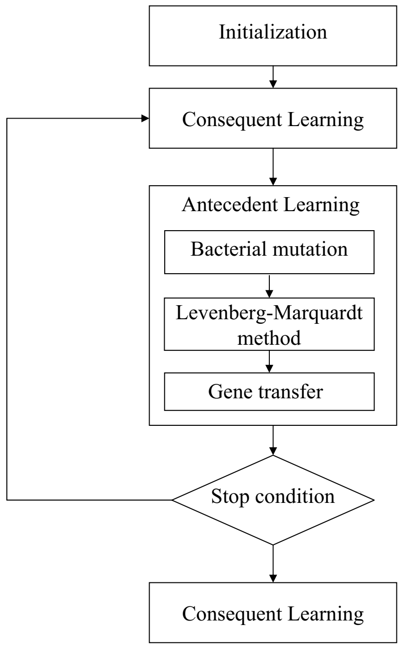

Fig. 1. Learning strategy by iterative hybrid training of antecedent and consequent parameters

in the *Bacterial Evolutionary Algorithm* and *Levenberg-Marquardt* method (which both represent the complete antecedent learning part) the consequent parameters are used for the evaluation of the optimization function as will be demonstrated in Section IV. After the stopping criterion is fulfilled, consequent learning is carried out once more, as an eventual last change in the antecedent parts should be compensated. This learning order is a well-known favorable one, no matter if it is applied for batch [12] or incremental learning [13]. The stopping criterion is reached, if the best fitted individual has a smaller fitness value than a pre-defined ϵ or the maximum number of generations N_{gen} is reached. The fitness value is calculated through an appropriate error measure defined as fitness function. An appropriate choice of an error measure will be discussed in subsection IV-C.

III. Initialization

There are various successful evolutionary optimization algorithms known from the literature. The advantage of these algorithms is their ability to solve and quasi-optimize problems with non-linear, high-dimensional, multimodal, and discontinuous character. The original genetic algorithm (GA) was developed by Holland [14] and based on the process of evolution of biological organisms. These processes can be easily applied in optimization problems where one individual corresponds to one possible solution of the problem. A more recent evolutionary technique is called bacterial evolutionary algorithm [1] [2], based on the microbial evolution phenomenon. Bacteria can transfer genes to other bacteria. This

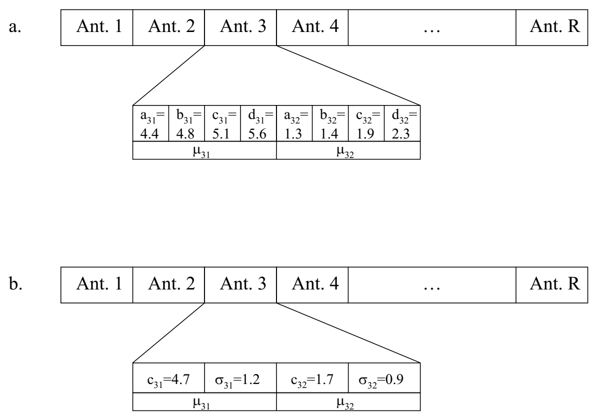

Fig. 2. Encoding scheme for nonlinear antecedent parameters in a Takagi-Sugeno fuzzy system

mechanism is used in the bacterial mutation and in the gene transfer operation. For the bacterial algorithm, the first step is to discover how the problem can be encoded in a bacterium (chromosome). Our task is to find the optimal fuzzy rule base for a pattern set. Thus, the parameters of the fuzzy rules must be encoded in the bacterium. The parameters of the rules' antecedent parts are the center and widths of the Gaussian fuzzy sets, respectively the breakpoints of the trapezoids. For example, the encoding method of a fuzzy system with two inputs and one output can be seen in Figure 2, where the upper image demonstrates the trapezoidal and the lower image the Gaussian case. As the consequent parameters are estimated by least squares approaches, see Section V, they are not encoded in the bacteria.

Before starting the antecedent learning scheme as demonstrated in Figure 1, the initial bacteria population is created. The population consists of N_{ind} chromosomes (bacteria), whereas the number of rules in the ith chromosome is $R(i)$. These two parameters have to be set in advance and have to be adjusted to the specific problem which should be solved in a trial and error phase. However, there exists some guidelines for an appropriate size of a population in GAs. Note that in this paper $R(i) = R(j)$ for all i and j, so just chromosomes with a fixed length are considered. Increasing the parameter $R(i)$ usually leads to more accurate models with respect to the error on the training data, but causes overfitting effects on fresh test data. This is also the reason why an extended error measure is used as fitness function, see Section IV-C. In total, $p \sum_{i=1}^{N_{ind}} R(i)$ membership functions are created, where p is the number of input variables in the given problem, and each membership function has either two (Gaussian case) or four (trapezoidal case) parameters. In the case of trapezoidal fuzzy sets a reliable initialization would be to start with a Ruspini partition [15] of fuzzy sets or at least a partition which ensures overlapping between every two adjacent fuzzy sets. This is to avoid holes between fuzzy sets and undefined input states, which improves the process security in an industrial application. It should be noticed that in the case of fuzzy sets with infinite support, for instance Gaussian sets, an overlap is always guaranteed, hence a randomization of the first partition is possible without violating the coverage property.

IV. Antecedent Learning

A. Bacterial Mutation

The bacterial mutation is applied to each chromosome. First, N_{clones} clones of the rule base are generated. Then a part of the chromosome is randomly selected and the parameters of this selected part are randomly changed in each clone (mutation). Next all the clones and the original bacterium are evaluated by an error criterion. The best individual transfers the mutated part into the other individuals. This cycle is repeated for the remaining parts, until all parts of the chromosome have been mutated and tested. At the end the best rule base is kept and the remaining N_{clones} are discharged. It is an important question to decide the length of chromosome (how many rules) to mutate at one time, and what is the degree of the mutation (expressed as the relative size in terms of the interval). This approach allows both selecting more than one membership function and fine-tuning. If selecting more than one membership function is allowed then the local minima in the optimization process can be avoided.

B. Gene Transfer

The gene transfer operation allows the recombination of genetic information between two bacteria. First the population must be divided into two halves. The better bacteria are called the superior half, the other bacteria are called the inferior half. One bacterium is randomly chosen from the superior half, this will be the source bacterium, and another is randomly chosen from the inferior half, this will be the destination bacterium. A "good" part from the source bacterium is chosen and this part will overwrite a "not-so-good" part of the destination bacterium. A good part can be a fuzzy rule with a high degree of activation value. The activation value of a fuzzy rule can be calculated for example:

$$\bar{v_i} = \frac{1}{N} \sum_{k=1}^{N} v_i^{(k)} \tag{5}$$

where $\bar{v}_i$ is the mean activation value of the i-th rule, $v_i^{(k)}$ is the activation value of the i-th rule for the k-th pattern, N is the number of patterns. So the best part of the source bacterium is the rule which has the greatest mean activation value. This cycle is repeated for N_{inf} times, where N_{inf} is the number of "infections" per generation.

C. Fitness Value Calculation

The fitness value calculation plays a central role for selecting the best individuals for gene transfer and the winning individual among the clones in the bacterial mutation. As a fitting method typically adapts to the training data and hence *mean root squared error (MSE)* on the training data alone is an overly optimistic estimate of the generalized prediction error, an optimism term is added to the MSE [16], obtaining the so-called *in-sample* error as fitness function:

$$Fit = Err_{in} = \frac{1}{N}\sum_{k=1}^{N}(\hat{y}_k - y_k)^2 + \frac{2}{N}\sum_{k=1}^{N} Cov(\hat{y}_k, y_k) \quad (6)$$

where $Cov(\hat{y}_k, y_k) = (\hat{y}_k - \bar{\hat{y}})(y_k - \bar{y})$ with $\bar{\hat{y}} = \frac{1}{N}\sum_{k=1}^{N}\hat{y}_k$ and $\bar{y} = \frac{1}{N}\sum_{k=1}^{N} y_k$ denotes the covariance between all measured and estimated output values over the training set. In Section VI we demonstrate the impact of the choice of the fitness function for individuum selection.

D. Levenberg-Marquardt

After bacterial operations, the Levenberg-Marquardt algorithm [5] is applied for optimization the antecedent fuzzy sets in the fuzzy rule base. In this method a minimization criterion has to be employed, that is related to the quality of the fitting. The training criterion that will be employed is the usual *Mean Squared Error (MSE)*:

$$\Omega = \frac{\|e(k)\|^2}{N} = \frac{1}{N}\sum_{k=1}^{N}(\hat{y_k} - y_k)^2 \quad (7)$$

where $\hat{y}$ stands for the estimated output vector, y for the measured output vector, e for the error vector and N the number of training samples. This is a feasible choice here, as Levenberg-Marquardt only changes the positions and widths of the already obtained fuzzy sets for steering the fuzzy systems in the actual population towards the nearest local and global minima. The most used method to minimize (7) is the Error-Back-Propagation (BP) [17] algorithm, which is a steepest descent algorithm. The BP algorithm is a first-order method as it only uses derivatives of the first order. If no line-search is used, then it has no guarantee of convergence and the convergence rate obtained is usually very slow. If a second-order method is to be employed, the best to use is the Levenberg-Marquardt (LM) algorithm [6], which explicitly exploits the underlying structure (sum-of-squares) of the optimization problem on hand. Denoting by J the Jacobian matrix:

$$J[k] = \frac{\partial \hat{y}(x^{(m)})[k]}{\partial par[k]} \quad (8)$$

where the vector par contains all membership functions' parameters of the antecedents, $x^{(m)}$ denotes the mth data point in the training matrix and k is the iteration variable. The derivatives after the non-linear parameters are the following for the Gaussian case with product t-norm:

$$\frac{\partial \hat{y}}{\partial c_{kl}} = \sum_{i=1}^{R} e^{-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j-c_{ij})^2}{\sigma_{ij}^2}} \left[(w_{k0}-w_{i0}) + ... + (w_{kp}-w_{ip})x_p\right] . \frac{e^{-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j-c_{kj})^2}{\sigma_{kj}^2}}(x_l - c_{kl})}{\sigma_{kl}^2\left(\sum_{i=1}^{R} e^{-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j-c_{ij})^2}{\sigma_{ij}^2}}\right)^2}$$

$$\frac{\partial \hat{y}}{\partial \sigma_{kl}} = \sum_{i=1}^{R} e^{-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j-c_{ij})^2}{\sigma_{ij}^2}} \left[(w_{k0}-w_{i0}) + ... + (w_{kp}-w_{ip})x_p\right] . \frac{e^{-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j-c_{kj})^2}{\sigma_{kj}^2}}(x_l - c_{kl})^2}{\sigma_{kl}^3\left(\sum_{i=1}^{R} e^{-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j-c_{ij})^2}{\sigma_{ij}^2}}\right)^2}$$

For the trapezoidal fuzzy sets in connection with minimum t-norm the following derivatives are obtained if $\mu_{kl} = \min_{j=1}^{p}\mu_{kj}(x_j)$:

$$\frac{\partial \hat{y}}{\partial par_{kl}} = \frac{\partial \mu_{kl}}{\partial par_{kl}} . \frac{\sum_{i=1}^{R}\min_{j=1}^{p}\left(\mu_{ij}(x_j)\left[(w_{k0}-w_{i0}) + ... + (w_{kp}-w_{ip})x_p\right]\right)}{\left(\sum_{i=1}^{R}\min_{j=1}^{p}\mu_{ij}(x_j)\right)^2}$$

with $par = \{a; b; c; d\}$ and

$$\frac{\partial \mu_{kl}}{\partial a_{kl}} = \frac{x_l - b_{kl}}{(b_{kl}-a_{kl})^2} \text{ if } a_{kl} \le x_l \le b_{kl} \text{ otherwise } 0$$

$$\frac{\partial \mu_{kl}}{\partial b_{kl}} = \frac{a_{kl} - x_l}{(b_{kl}-a_{kl})^2} \text{ if } a_{kl} \le x_l \le b_{kl} \text{ otherwise } 0$$

$$\frac{\partial \mu_{kl}}{\partial c_{kl}} = \frac{d_{kl} - x_l}{(d_{kl}-c_{kl})^2} \text{ if } c_{kl} \le x_l \le d_{kl} \text{ otherwise } 0$$

$$\frac{\partial \mu_{kl}}{\partial d_{kl}} = \frac{x_l - c_{kl}}{(d_{kl}-c_{kl})^2} \text{ if } c_{kl} \le x_l \le d_{kl} \text{ otherwise } 0$$

If $\mu_{kl} \neq \min_{j=1}^{p}\mu_{kj}(x_j)$, then all derivatives are 0.

With these definitions, the LM update is given as the solution of

$$(J^T[k]J[k] + \alpha I)s[k] = -J^T[k]e[k] \quad (9)$$

where $s[k] = par[k] - par[k-1]$ denotes the parameter change in the k iteration step and α the regularization parameter, which controls both, the search direction and the magnitude of the update. The search direction varies between the Gauss-Newton direction and the steepest direction, according to the value of α. This is dependent on how well the actual criterion agrees with a quadratic function in a particular neighborhood. The good results presented by the LM method (compared with other second-order methods such as the quasi-Newton and conjugate gradient methods) are due to the explicit exploitation of the underlying characteristics of the optimization problem (a sum-of-square of errors) by the training algorithm. Notice that (9) can be recast as:

$$s[k] = -\begin{bmatrix} J[k] \\ \sqrt{\alpha} I \end{bmatrix}^{+} \begin{bmatrix} e[k] \\ 0 \end{bmatrix} \tag{10}$$

The complexity of this operation is of $O(n^3)$, where n is the number of columns of J, i.e. the number of antecedent parameters in the fuzzy system.

V. Consequent Learning

For consequent learning we exploit least squares and alternatively, in the case of an incremental online learning scheme we use the recursive least squares [18] approach. This is feasible as the consequent parameters in (4) are linear ones and therefore the mean squared error optimization problem can be solved analytically and globally. Principally there are two possibilities for linear consequent learning [19]:

- Global Learning: all C normalized membership functions (each one belonging exactly to one rule) are joined together in one regression matrix, hence the complete parameter vector $\vec{w}$ representing all linear consequent parameters in all rules is optimized. Thus, the optimization problem is defined as

  $$J = \sum_{k=1}^{N} (y(k) - \sum_{i=1}^{C} l_i \Psi_i(\vec{x}(k)))^2 = \min_{\vec{w}}! \tag{11}$$

  and the solution given by

  $$\hat{\vec{w}} = (R^T R)^{-1} R^T \vec{y} \tag{12}$$

  where R the regression matrix containing the regressors

  $$\vec{r_i}(k) = [\Psi_i(\vec{x}(k)) x_1(k) \Psi_i(\vec{x}(k)) \dots x_p(k) \Psi_i(\vec{x}(k))]$$

  for all C rules and $k = 1, \dots, N$ data points, where $x_i(k)$ is the ith column of the row vector $\vec{x}$ in point k.
- Local Learning: Each rule is treated separately and the corresponding linear consequent parameters are optimized in a weighted least squares approach, where the weights contain the normalized membership functions. Thus, the optimization problem for the ith rule is defined as:

  $$J_i = \sum_{k=1}^{N} \Psi_i(\vec{x}(k)) e_i^2(k) = \min_{w_i}! \tag{13}$$

  where $e_i(k) = y(k) - \hat{y_i}(k)$ represents the error of the local linear model in the kth point as $\hat{\vec{y_i}} = R_i \vec{w_i}$ with R_i the original data matrix with only ones in the first column. The solution is given by:

  $$\hat{\vec{w_i}} = (R_i^T Q_i R_i)^{-1} R_i^T Q_i \vec{y} \tag{14}$$

  with Q_i the weighting matrix

  $$Q_i = \begin{bmatrix} \Psi_i(\vec{x}(1)) & 0 & ... & 0 \\ 0 & \Psi_i(\vec{x}(2)) & ... & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & ... & \Psi_i(\vec{x}(N)) \end{bmatrix}$$

From various analytical and empirical examinations local learning turned out to be superior to global one in a lot of aspects such as numerical stability (as dealing with inversion of smaller matrices), computational performance, the bias error when approximating from data with medium and high noise levels and transparency of the consequent functions (hyper-planes). The latter aspect is due to an observed snuggling of the linear hyper-planes along the approximating surface [20]. This entails well interpretable insight for local control behaviors and opens the possibility to gain reasonable error bars and confidence intervals directly from the hyper-planes.

In the case of adaptation of all the fuzzy systems in the population to new incoming points, consequent learning in incremental manner can be performed by using recursive least squares [18], which possesses the favorable property that it converges to the global optimum within each iteration step. This is carried out for each rule separately as the local learning approach is applied:

$$\hat{\vec{w_i}}(k+1) = \hat{\vec{w_i}}(k) + \gamma(k)(y(k+1) - \vec{r}^T(k+1)\hat{\vec{w_i}}(k)) \tag{15}$$

$$\gamma(k) = \frac{P_i(k)\vec{r}(k+1)}{\frac{1}{\Psi_i(\vec{x}(k+1))} + \vec{r}^T(k+1)P_i(k)\vec{r}(k+1)} \tag{16}$$

$$P_i(k+1) = (I - \gamma(k)\vec{r}^T(k+1))P_i(k) \tag{17}$$

with $P_i(k) = (R_i(k)^T Q_i(k) R_i(k))^{-1}$ the inverse weighted inverse Hesse matrix and $\vec{r}(k+1) = [1 \;\; x_1(k+1) \;\; x_2(k+1) \;\; \dots \;\; x_p(k+1)]^T$ the regressor values of the $k+1$th data point, which is the same for all i rules.

In order to ensure stable matrix inversion a regularization parameter $\alpha > 0$ is incorporated in the case of badly conditioned matrices into the optimization function:

$$J_i = \sum_{k=1}^{N} \Psi_i(\vec{x}(k)) e_i^2(k) + \alpha \vec{w_i}^T \vec{w_i} = \min_{\vec{w_i}}! \tag{18}$$

The regularization parameter $\alpha > 0$ controls the balance between fitting the data and avoiding the badly conditioned Hesse matrices $R_i^T Q_i R_i$ leading to the following solution for the local learning approach:

$$\hat{\vec{w_i}} = (R_i^T Q_i R_i + \alpha I)^{-1} R_i^T Q_i \vec{y} \tag{19}$$

where I is a $(p+1) \times (p+1)$ identity matrix. From these two estimation formulas it is obvious that a large α leads to numerically stable estimation but to incorrect parameter estimations resulting in a high bias error. Hence, in practice the following strategy for estimation of linear consequents for the ith rule is a good choice:

1) Compute $R_i^T Q_i R_i$

2) Compute the condition of $R_i^T Q_i R_i$, by applying a singular value decomposition and using the well-known formula $cond(R_i^T Q_i R_i) = \frac{\lambda_{max}}{\lambda_{min}}$, where λ_{min} and λ_{max} denote the minimal and maximal eigenvalue of $R_i^T Q_i R_i$.
3) If $cond(R_i^T Q_i R_i) > threshold$, the matrix is badly conditioned, hence do the following
   a) Choose α in a way, such that the condition of $R_i^T Q_i R_i$ gets smaller than the threshold but not too small due to the considerations above. This can be accomplished by exploiting the fact, that the addition of αI to $R_i^T Q_i R_i$ influences the small eigenvalues strongly leading to the approximation formula $cond(R_i^T Q_i R_i) \approx \frac{\lambda_{max}}{\alpha}$, hence if desiring a condition of $threshold/2$, α can be approximated via

$$\alpha \approx \frac{2\lambda_{max}}{threshold}$$

   b) Perform the ridge least squares as in (19)
4) Else, apply weighted least squares approach as in (14)

Note: Setting the threshold can be carried out from experience with badly conditioned matrices or simply by stepwise trying out from which condition level on the inverse leads to instable results. The same procedure can be applied for recursive least squares, whenever the condition of $P_i(k)$ gets too high.

VI. Evaluation

In this section an example has been used to test the performance of the proposed algorithm. We refer to our hybrid method as *Bac-LM-CL*, where for the different variants (i.e. different choices of fuzzy sets, t-norms and fitness functions for individuum selection) we use the following notation:
Bac-LM-CL var1a = *Bac-LM-CL* with Gaussian sets, product t-norm and MSE (7) as fitness function
Bac-LM-CL var1b = *Bac-LM-CL* with Gaussian sets, product t-norm and in-sample error (6) as fitness function
Bac-LM-CL var2a = *Bac-LM-CL* with trapezoidal sets, minimum t-norm and MSE (7) as fitness function
Bac-LM-CL var2b = *Bac-LM-CL* with trapezoidal sets, minimum t-norm and in-sample error (6) as fitness function

The test is based on an artificial data set generated by the following non-linear six-dimensional function:

$$f(\vec{x}) = x_1 + x_2^{\frac{1}{2}} + x_3 x_4 + 2e^{2(x_5 - x_6)} \qquad (20)$$

where $x_1 \in [1,5]$, $x_2 \in [1,5]$, $x_3 \in [0,4]$, $x_4 \in [0,0.6]$, $x_5 \in [0,1]$, $x_6 \in [0,1.2]$. The task is to approximate this function as close as possible while not to overfit. Hence, the complete data set of 400 data points were randomly split into a training and a test data set. The latter was used to calculate the mean squared error (MSE) as approximation of the generalized prediction error. In fact, a 10-fold cross-validation [21] leads to even more accurate values for the generalized prediction error, but quite often a unique split is sufficient for a performance comparison between modelling methods. Table I compares the well-known and in MATLAB toolboxes available methods *ANFIS* [22], *FMCLUST* [12] and *genfis2* [4] with the extended version of *genfis2* by using a modified version of vector quantization as clustering method [23] and local learning for estimating the rule consequent functions and with *FLEXFIS* [13], an incremental version of *genfis2 extended*, which builds up the fuzzy system on a sample per sample basis (and hence applicable for fast online identification tasks). For each method different settings of the most essential parameters were carried out and the best settings were taken and stated in Table I. In our algorithm the number of generations was set to 40, the number of individuals is 10. The parameters of the bacterial operators were 8 clones and 4 infections. The length of the bacterium (so, the number of rules) was set to 10, thus we can compare which one of the four variants gives the best result. From Table I it can be seen that our approach provides a good solution for the problem. If we compare our method with the results obtained by the other approaches we can see that we can get lower MSE values with fewer rules in most of the cases. The genfis2 extended method gives a low MSE value, however it uses more rules. Our technique converges to the lowest possible solution. If we increase the parameter values then better results can be obtained. Comparing our 4 variants we can see that the fuzzy rules with Gaussian membership functions are better. The first reason for this is that when Gaussian sets are applied then there is no gap in the rule base. Because we do not use interpolative reasoning, in the trapezoidal case there can be gaps in the rules causing some abnormal inference output on this way. The other reason for the better outcome by the Gaussian sets is that in the Levenberg-Marquardt step the gradient values carry more information than the trapezoids, because the latter have constant parts in the memberships. We can also see that for fitness value calculation the classical MSE (7) provides a better outcome than the so-called in-sample error (6).

TABLE I
Comparison of fuzzy modelling methods on non-linear 6-dimensional data

Method	MSE Test	No. of Rules
FMCLUST	1.31	10
ANFIS	0.71	64
genfis2	1.38	18
genfis2 ext.	0.68	14
FLEXFIS	1.09	17
Bac-LM-CL var1a	1.06	10
Bac-LM-CL var1b	1.15	10
Bac-LM-CL var2a	1.33	10
Bac-LM-CL var2b	1.38	10

VII. Conclusion

A new technique for Takagi-Sugeno fuzzy rule base optimization was introduced in this paper. The bacterial algorithm was combined with the Levenberg-Marquardt method to optimize the nonlinear antecedents' parameters. Separately for the consequents parameters, a least-square method was proposed.

Acknowledgment

Research supported by the Australian Research Council, the Széchenyi University Main Research Direction Grant 2005, and a National Scientific Research Fund Grant OTKA T048832.

References

[1] J. Botzheim, B. Hamori, L. Kóczy, and A. Ruano, "Bacterial algorithm applied for fuzzy rule extraction," in *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems, IPMU 2002*, Annecy, France, 2002, pp. 1021–1026.

[2] N. E. Nawa and T. Furuhashi, "Fuzzy system parameters discovery by bacterial evolutionary algorithm," *IEEE Trans. on Fuzzy Systems*, vol. 7, pp. 608–616, 1999.

[3] M. Burger, J. Haslinger, and U. Bodenhofer, "Tuning of fuzzy systems as an ill-posed problem," in *Progress in Industrial Mathematics at ECMI 2000*, ser. Mathematics in Industry, M. Anile, V. Capasso, and A. Greco, Eds. Springer, 2002, vol. 1, pp. 493–498.

[4] R. Yager and D. Filev, "Generation of fuzzy rules by mountain clustering," Machine Intelligence Institute, Iona College, New Rochelle, NY 10801, Tech. Rep. MII-1318R, 1994.

[5] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM J. Appl. Math.*, vol. 11, pp. 431–441, 1963.

[6] A. Ruano, C. Cabrita, J. Oliveira, L. Kóczy, and D. Tikk, "Supervised training algorithms for B-spline neural networks and fuzzy systems," in *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, 2001.

[7] J. Botzheim, C. Cabrita, L. Kóczy, and A. Ruano, "Estimating fuzzy membership functions parameters by the Levenberg-Marquardt algorithm," in *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2004*, Budapest, Hungary, 2004, pp. 1667–1672.

[8] ——, "Fuzzy rule extraction by bacterial memetic algorithm," in *Proceedings of IFSA 2005*, Bejing, China, July 2005, pp. 1563–1568.

[9] L. Wang, "Fuzzy systems are universal approximators," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, 1992, pp. 1163–1169.

[10] J.-S. Jang and C.-T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE Trans. on Neural Networks*, vol. 4, pp. 156–159, 1993.

[11] L. Kóczy, D.Tikk, and T. Gedeon, "On functional equivalence of certain fuzzy controllers and RBF type approximation schemes," *International Journal of Fuzzy Systems*, 2000.

[12] R. Babuska, *Fuzzy Modeling for Control*. Boston: Kluwer Academic Publishers, 1998.

[13] E. Lughofer and E. Klement, "FLEXFIS: A variant for incremental learning of Takagi-Sugeno fuzzy systems," in *Proceedings of FUZZ-IEEE 2005*, Reno, Nevada, U.S.A., 2005, pp. 915–920.

[14] J. Holland, *Adaption in Natural and Artificial Systems*. Cambridge, Massachusetts: The MIT Press, 1992.

[15] E. Ruspini, "A new approach to clustering," *Information and Control*, vol. 15, pp. 22–32, 1969.

[16] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. New York, Berlin, Heidelberg, Germany: Springer Verlag, 2001.

[17] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Appl. Math., Harvard University, USA, 1974.

[18] L. Ljung, *System Identification: Theory for the User*. Upper Saddle River, New Jersey 07458: Prentice Hall PTR, Prentic Hall Inc., 1999.

[19] J. Yen, L. Wang, and C. Gillespie, "Improving the interpretability of TSK fuzzy models by combining global learning and local learning," *IEEE Trans. on Fuzzy Systems*, vol. 6, no. 4, pp. 530–537, 1998.

[20] E. Lughofer, E. Hüllermeier, and E. Klement, "Improving the interpretability of data-driven evolving fuzzy systems," in *Proceedings of EUSFLAT 2005*, Barcelona, Spain, 2005, pp. 28–33.

[21] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society*, vol. 36, pp. 111–147, 1974.

[22] J.-S. Jang, "ANFIS: Adaptive-network-based fuzzy inference systems," *IEEE Trans. Syst. Man Cybern.*, vol. 23, pp. 665–685, 1993.

[23] E. Lughofer and U. Bodenhofer, "Incremental learning of fuzzy basis function networks with a modified version of vector quantization," in *to appear in Proceedings of IPMU 2006*, Paris, France, 2006.