Multigrid-based fuzzy systems for time series prediction:
CATS competition

L.J. Herrera*, H. Pomares, I. Rojas, A. Guillén, J. González, M. Awad, A. Herrera

Department of Computer Architecture and Computer Technology, University of Granada, 18017 Granada, Spain

Available online 1 March 2007

Abstract

This paper presents a novel learning methodology for multigrid-based fuzzy system (MGFS), and its application to the CATS time series prediction benchmark. The MGFS model keeps the advantages of the traditional grid-based fuzzy systems (GBFS), and overcomes the problem inherent to all GBFSs when dealing with high dimensional input data. Thus the MGFS model keeps interpretability, low computational cost and high generalization. A novel architecture selection algorithm for MGFSs that allows performing input variable selection is proposed. It identifies the sub-optimal architecture, according to a provided data set of input/output data. The architecture selection algorithm is completed with a structure identification procedure, used to obtain the optimal input space partitioning of the different sub-grids of the model. The complete algorithm is used to obtain the MGFS models for the CATS series prediction problem, solved using a direct prediction-based approach.

1. Introduction

Several paradigms and methodologies have been applied to the problem of time series prediction. All the existing methodologies could be classified in two main blocks: methods dealing with linear models such as AR, MA, ARIMA, etc. [3,4] and methods dealing with nonlinear models such as artificial neural networks (ANN), RBF networks (RBFN), Fuzzy systems (FS), etc. [1,17,20].

In general, fuzzy systems are very powerful and convenient tools for function approximation and time series prediction problems, due to their efficiency and interpretability capabilities. Specifically, grid-based fuzzy systems (GBFSs) [19,26] perform a grid partitioning of the input space and generate a set of rules that covers the whole input space, providing a high generalization capability. Nevertheless, this approach suffers from the curse of dimensionality (exponential growth in the number of rules) [2] for problems with moderated complexity, specially when considering high-dimensional problems.

The multigrid-based fuzzy system (MGFS) model opens a door for the treatment of high-dimensional function approximation and time series prediction problems using grid-based models. A MGFS is a modified additive fuzzy model composed of a set of sub-grids; the output of the model is the weighted average of the outputs of the rules of the component sub-grids. This paper presents a novel architecture selection algorithm for MGFS. It allows taking into account any number of input variables, while it selects the most relevant interrelations among the input variables to construct the sub-grids of the model. This way it gets rid of the exponential growth in the number of rules that the GBFS model presents, and it keeps its advantages. The architecture selection methodology is completed after with a structure identification procedure based on [18], to find the optimal input space partitioning of the MGFS model. It identifies the optimal number of rules needed to perform the approximation, placing the appropriate number of membership functions along the input variables domains.

In this work, the MGFS model and the proposed learning algorithm is applied to the CATS time series prediction problem. The CATS time series [28] consists on...
5000 data points in which 100 values are missing. These missing values are divided in five blocks of 20 data points. The CATS benchmark was first used for competition in the IJCNN’2004 conference held in Budapest [13]. The MGFS learning algorithm is used to obtain a set of direct prediction models [27] for the present problem. For the first four gaps of the missing values, a set of forwards and backwards models are used to predict the data points. For the last gap, only forwards models are used to predict the 20 data points.

The paper is organized as follows. Section 2 introduces the basics of the multigrid fuzzy model. Section 3 presents the proposed learning algorithm for MGFS. It includes a first step of MGFS architecture selection, and is completed in a second step with a structure identification for the obtained MGFS architecture. Section 4 reviews the CATS time series prediction problem. Sections 5–7 show the application of the MGFS model to the CATS time series prediction problem, and some improvements performed with respect to the initial work [8]. Finally, Section 8 concludes the paper and discusses the further work.

2. Multigrid fuzzy systems

Fuzzy systems present in general a number of advantages and properties when they are applied to input/output (I/O) data modelling problems, such as: (a) the models obtained with this paradigm are understandable, avoiding the black-box curse of other paradigms; (b) the underlying model is simple, but can explain complex non-linear relations among variables; (c) it can be expressed in terms of linguistic rules, thus providing full interpretability to the model; (d) it can easily deal with imprecise data and (e) it has a high generalization capability [9].

This work proposes the use of a modified additive fuzzy-based approach based on GBFSs [19,18,21], which we call a MGFS model [9]. It keeps its philosophy and advantages and avoids its main drawback, the curse of dimensionality. We recall that a GBFs consists on a fuzzy system whose rules are distributed in a grid in the n-dimensional space. Takagi–Sugeno–Kang (TSK) [24] GBFSs comprises a set of \( K \) IF–THEN rules in the form

\[
\text{Rule}^k : \text{IF } x_1 \text{ is } \mu^k_1 \text{ AND } \ldots \text{ AND } x_n \text{ is } \mu^k_n \text{ THEN } y = R^k, \tag{1}
\]

where the \( \mu^k_i \) are fuzzy sets characterized by membership functions (MFs) \( \mu^k_i (x_i) \) in universes of discourse \( U_i \) (in which variables \( x_i \) take their values), and \( R^k \) are the consequents of the rules. The \( \mu^k_i \) in the GBFS rules all belong to a specified subset per input variable \( i \), and the rules centers (points in which the activation value is 1 for the rule) form a grid in the \( n \)-dimensional space as can be seen for a two-dimensional toy example in Fig. 1.

When dealing with a complex modelling problem with a high number of input variables, an \( n \)-dimensional grid runs into the curse of dimensionality problem [2]: the number of rules, that can be obtained in GBFSs as

\[
K = \prod_{i=1}^{n} mf_i, \tag{2}
\]

where \( mf_i \) is the number of MFs in variable \( i \), can be too high, reaching an efficiency bottleneck. Moreover, a model with a too large number of rules and with too many
antecedents on each rule, ends up in an incomprehensible model [5].

Fig. 2 shows the proposed MGFS architecture [9] to deal with high dimensional input spaces. Each group of variables is used to define a GBFs from which a set of rules is obtained in the form

IF $x_{i_1}^p$ is $\mu_{i_1}^{k_p}$ AND ... AND $x_{i_n}^p$ is $\mu_{i_n}^{k_p}$
THEN $y = R^{k_p}$

(3)

$R^{k_p}$ being the consequent of the $k$th rule of the $p$th GBFS. This work will use constant rule consequents (linear or second order rules [11] are not analyzed here). All the rules from all the GBFSs make up the whole MGFS, whose output is obtained by weighted average aggregation. Therefore, the final output of the system for any input value $\tilde{x} = (x_1, x_2, \ldots, x_N)$, can be expressed as

$$F(\tilde{x}, MF, R, A) = \frac{\sum_{p=1}^P \sum_{k=1}^{K_p} R^{k_p} \prod_{m=1}^{n_p} \mu_{i_m}^{k_p}(x_m)}{\sum_{p=1}^P \sum_{k=1}^{K_p} \prod_{m=1}^{n_p} \mu_{i_m}^{k_p}(x_m)},$$

(4)

where an explicit statement is made on the dependency of the output function with the structure of membership functions $MF$ of the system, with the consequents of the whole set of $K$ rules $R$, and with the architecture of the MGFS $A = \{x_1, x_2, \ldots, x_N\}, \{x_1^1, x_2^1, \ldots, x_2^2\}, \ldots, \{x_1^P, x_2^P, \ldots, x_P^p\}$. The total number of rules $K$ equals the sum of $\{K_1, K_2, \ldots, K_P\}$, i.e., the input variables entering each individual grid according to the structure $MF$.

The MGFS model reduces the computational complexity of GBFSs when dealing with complex problems, and can obtain interpretable grid-based systems for I/O modelling problems with a higher number of input variables. MGFSs open a door for the expression of complex nonlinear functions of a high number of variables, as a summation of simpler functions; the conceptual complexity of complex problems is reduced, and the input space is covered using grid partitioning. Moreover, thanks to the learning approach next presented, it is possible to select the most relevant variables (and their interrelations) that influence the output variable.

However, MGFSs, as a grid-based model, still might be not useful for very complex problems in which there is a very big number of relevant input variables (and complex interrelations among them). The optimal MGFS architecture needed to perform the I/O modelling could be unfeasible (see Eq. (2)). From our point of view, this would be the case of modelling problems without a grid-based interpretable solution; for problems with more than 10–15 relevant variables, the chances of applicability might be reduced. For those problems clustering-based fuzzy models (or similarly RBFNs [20], or other paradigms such as SVMs [22,23] or ANNs [1] could be used.

In cases in which an interpretable model is needed, the MGFS model extends the applicability of traditional GBFS. The MGFS model together with the learning algorithm proposed, as it will be shown, obtains an interpretable model, providing high generalization and good performance.

### 3. Multigrid learning algorithm

This section presents an effective and automatic algorithm to learn a MGFS model from a set of I/O data.
Learning a MGFS first implies to determine the groups of variables that will form each sub-grid; this process will be referred to as architecture selection. All the sub-grids together with the input variables entering each of them, will make up the system architecture, as shown in Fig. 2. Second, a structure identification step can be performed, to determine the optimal input space partitioning within each sub-grid. Parameter adjustment (finding the optimum MFs parameters) should also be performed during the structure identification process [18].

This section is structured as follows: Section 3.1 presents the proposed architecture selection algorithm. Afterwards, Section 3.2 introduces a complementary structure identification and parameter adjustment algorithm for MGFSs. Two significant examples of application are briefly discussed in Sections 3.3 and 3.4. Finally Section 3.5 argues some final remarks on the methodology proposed.

3.1. MGFS architecture selection

The objective of the MGFS architecture selection process is to select the MGFS architecture that performs best given an I/O data set. Obtaining the optimal architecture is a very complex task. In principle, every possible architecture should be evaluated, but the number of possible architectures for a given number of input variables is a combinatorial number. A procedure that tries to check every possible architecture would be unfeasible. This section presents an effective incremental algorithm that avoids a complete scanning of the search space, but that obtains an appropriate good performing architecture for any given I/O modelling problem.

A top-down algorithm was presented in [9] that, from a complete GBFS (a MGFS with one single n-dimensional sub-grid) discards more complex structures in favor of simpler ones, while keeping a certain error threshold and keeping fixed the number of MFs per variable. This approach, though very powerful to discover intrinsic relations in the input variables, has one main disadvantage that makes it unsuitable for complex problems: the starting point is a whole GBFS system, which can be computationally too expensive when having a high number of input variables (see Eq. (2)). This starting point is thus unfeasible for medium-high complexity problems.

However, in this paper we will use a different bottom-up approach, first taken in [10]. It starts with the least complex possible architecture, having one sub-grid per input variable, and performs a breadth-first search in increasing groups-complexity. A different idea surrounds this second alternative: instead of looking for keeping a certain error tolerance while reducing the whole-system complexity, this approach keeps a certain acceptable complexity limit (number of parameters to be optimized = number of rules) while it searches for the system that performs best, given this number of rules. The limit in the number of rules is an important parameter of the algorithm and its obtention will be discussed afterwards in this subsection.

In order to equitably compare different MGFS architectures throughout the proposed incremental algorithm, for a given limit in the number of parameters, the following heuristics will be used (in the lack of further information about the specific problem) for the sake of efficiency and simplicity: the rules will be homogeneously split among all the sub-grids and, inside each sub-grid, the number of MFs per variable will be homogeneously distributed in order to approximate the number of rules assigned. Note that this limit in the number of rules might not fit the number of resulting rules assigned to all the sub-grids according to this heuristics; nevertheless as it was mentioned, the algorithm tries to reach the best rules and MFs distribution to approximate this number. The rule consequents will be optimally obtained in every case using least squares (see the discussion about least squares in Section 3.2). The MGFS model will use triangular-partitioned MFs [19] in this work, so that the optimal rule consequents can be obtained by linear methods [9,19].

The algorithm works in a greedy manner (see Fig. 3). It starts from the simple MGFS architecture having n sub-grids with one input variable each, where n is the initial number of input variables. It will sequentially search for additions of one variable to one sub-grid, but without searching for component sub-grids of order m + 1 until all the possible combinations of order m have been explored. Among all the possibilities of adding one variable to one sub-grid (forming a grid of order m), the algorithm will choose in each step the one that reduces more the training error. It will repeat the process until the error does not decrease and then will try sub-grids of order m + 1.

The fixed number of rules is always kept, splitting this number as explained previously. That is, adding a variable to a sub-grid implies a reduction in the number of MFs per variable in that sub-grid. The rule splitting proposed carries out a trade-off between higher dimensionality and lower number of MFs per variable, in the current sub-grid. This implies that only significant variables interrelations will reflect a reduction in the training error. The MGFS architecture that provides the lowest training error given the data set D, will be chosen as the candidate.

According to Fig. 3, a leave-one-out sub-grid elimination procedure is performed after the initialization of the model, to check that all the selected sub-grids are relevant for the modelling problem. If leaving a sub-grid out of the model leaves a similar or better training error, then this sub-grid can be eliminated from the architecture. We recall that eliminating an irrelevant sub-grid, due to the rule splitting used, implies that other relevant sub-grids will get more MFs. This could even imply an increase in the performance. The leave-one-out phase will assure that irrelevant and redundant variables are discarded from the model, by discarding the respective single-variable sub-grids. In this way, the search space would be limited, with a low risk (variables that in principle do not provide information to the output are eliminated; however, they
could be recovered at posterior sub-grid extensions), but with a reduction in computational cost.

The limit in the number of rules for which the best MGFS structure will be searched, can be selected taking into account the trade-off between the interpretability and precision of the system to be designed, the number of training points available (considering that usually it is not worth having a higher number of parameters than data), the number of input variables and the computational cost of the algorithm. To obtain a final value for the limit in the number of rules, in this work cross-validation will be performed to tune the trade-off between precision and generalization of the architecture to be selected. Thus, the training data set will be divided into a number of blocks. For a candidate limit, the architecture selection algorithm (see Fig. 3) will be launched using all the data blocks but one, which is used to evaluate the obtained architecture using the obtained rule splitting. This process is repeated for all the blocks, leaving one out at a time, and the average validation error is considered for the selection of the optimal limit in the number of rules. This procedure will be launched using a range of candidate values of the limit. The limit selected will be that for which the mean validation error was lowest.

The final selected architecture will be obtained by executing the architecture selection algorithm using the complete training data set, for the optimal limit in the number of rules selected. Due to the non-linear optimization process being discussed, it cannot be claimed that the obtained architecture will be the optimum in all cases, but in any case it will be a sub-optimal architecture.

Once the architecture has been selected, it is interesting to identify the optimal number of MFs per input variable, with the MF parameters also optimized. The rule splitting used in the architecture selection algorithm might be inefficient in some cases leading to a deficient performance. An optimization procedure is proposed in the next section.

3.2. Optimization of a MGFS model with a given architecture

This section deals with finding the optimal input space partitioning to perform the I/O modelling for a given architecture. A correct distribution of the MFs in the input variables helps to control the trade-off between the precision and the generalization capabilities of the model. For the MGFS model, it implies partitioning the input space in each of the sub-grids so that the global model performs optimally. The process of optimizing a MGFS with a given architecture is divided into two intertwined sub-processes, structure identification and parameter adjustment (see Fig. 4).

3.2.1. Structure identification

Structure identification deals with finding the optimum number of MFs per input variable, or similarly the
It is now proposed an automatic structure identification algorithm based on a well-known structure identification approach for GBFSs [18]. It starts with the simplest structure possible, having 1 MF per input variable. Then an iterative process adds in each iteration a new MF to a selected variable, until a certain threshold is reached or whenever the validation error stops decreasing.

The variable in which the MF is added, is selected evaluating an addition of \( l \) MFs to each variable of every sub-grid in the MGFS. The variable that showed that adding \( l \) MFs reduces most the training error is selected and an addition of 1 MF is eventually performed. The value of \( l \) is proportional (200\% or 150\%) to the highest number of MFs in a variable in the current step (see [18] for more details); it is used to estimate the need of each input variable of having more MFs to improve the overall performance. Once the variable is selected and the new MF is added, cross-validation is used to check if the new addition of an MF improves the model’s efficiency and generalization. In each of the executions of the cross-validation process, the parameter adjustment phase (explained in the next subsection) is performed. The final MGFS structure, is the one for which the cross-validation error was lowest. The MF parameters of the model (with the obtained architecture and structure), can be finally optimized using all the training data to obtain the definitive model.

It must be noted that one input variable \( x_i \) can belong to several different sub-grids in the selected MGFS architecture. Each input variable \( x_i \) could have as many different representations of its domain as sub-grids it is present in. However, for the structure optimization approach presented, one input variable \( x_i \) present in different sub-grids is considered as a single variable. The reason for that is keeping the interpretability of the whole system. The flexibility of the model could be higher if those input variables were to be considered as different variables of the MGFS model. Moreover, the methodology presented could easily consider this option. But we chose that it is important to keep a single input partitioning of each input variable of the problem, so that the interpretability of the model (comparing to GBFSs, in which there is a single set of membership functions per variable) is kept.

### 3.2.2. Parameter adjustment

Parameter adjustment refers here to the optimization of the location of the membership functions and the rule consequents. With respect to the rule consequents, they can always be optimally obtained using least squares minimization (LSE). This technique is used in every step of the architecture selection and structure identification algorithms, to obtain the optimal rule consequents and evaluate the current configuration. LSE leads to a linear equation system that can be solved using any mathematical method. This work uses singular value decomposition (SVD) due to its efficiency and thanks to the possibility of eliminating unneeded coefficients [11].

With respect to the MF parameter adjustment, due to the use of triangular partition (see Fig. 1), only the rule

![MGFS optimization](image-url)
centers of the MF non-situated in the extremes of the domain need to be adjusted. Parameter adjustment is a necessary step, since there are some cases in which there is a higher variability of the function to approximate in some zones of the input space; a higher number of MFs should be assigned to those areas [7]. This work uses a gradient-based approach to properly place the rule centers of every input variable in the sub-grids of the MGFS. The gradient-based approach used in this work is the Levenberg–Marquardt algorithm [16], which has been chosen among other gradient-based approaches for its robustness and computational efficiency. But as every local search algorithm, it has the risk of falling into local minimum. A modified approach adapted from [19] for GBFS, called “Error Equidistribution”, is performed before the local search, so that a good initial position of the MF centers in the MGFS is taken. This centers initialization technique gets its name, because it places all the MF centers, in such a way that in both sides of every MF, there is the same amount of error; according to the data set D and the least squares approach. The use of the Levenberg–Marquardt approach, together with the initialization of the centers using error equidistribution, guarantees that a good local minimum for the MF parameters will be found [19].

3.3. Example 1

This subsection presents an application of the complete MGFS methodology to the nine-dimensional function defined by the following mathematical expression:

\[
F(x_1, \ldots, x_9) = \sin(\pi x_1 x_2) + \exp x_3 x_4 + 2x_5^2 + 2x_6 x_3 x_8 + 0x_9 + \xi, \tag{5}
\]

where \(x_1, \ldots, x_9 \in [0, 1]\), and \(\xi\) is an additive gaussian error with standard deviation 0.1. In this artificial example the variables are grouped in three well-differentiated associations, that the MGFS learning algorithm should identify as sub-grids. It is a simple artificial example, but it will help to verify the operation and robustness of the proposed learning algorithm. Note that with these problem conditions, the traditional GBFS model cannot be applied; considering for example three MFs per variable an unmanageable number of rules is obtained according to Eq. (2).

For this proposed example, 1000 data points were generated. 800 of them were used for training and the remaining 200 for testing the learning methodology. First the architecture selection algorithm was launched. The limit in the number of rules was obtained using 5-fold cross-validation over the training data. Fig. 5 shows the average validation errors obtained for the selection of the limit in the number of rules.

Thus the limit selected is 70 rules that is the lowest validation error value found. However, for this particular example, from the value 40 the validation error is stabilized. This implies that there is a wide range of values of the limit in the number of rules, for which the architecture obtained by the algorithm is optimal. Table 1 shows the evolution of the architecture selection algorithm using the limit 70 with the 800 training data (using as error measure the normalized root mean-square error, NRMSE [19]). See how first the leave-one-out sub-grid elimination process is carried out; the variable \(x_6\) is eliminated from the current model. After, sub-grids of two variables are evaluated. Note that when a bigger sub-grid is added to the architecture, subsumed sub-grids are eventually eliminated from the model.

Next, sub-grids of three variables, coming from the addition of a new variable to a two variables sub-grid, are evaluated. After four variables sub-grids are evaluated, and the algorithm stops since there are no more interrelations detected.

The structure identification procedure is performed next. Table 2 shows the trace of the iterative process. As explained in Section 3.2, when the number of centers in a variable is higher than 2, the Levenberg–Marquardt algorithm (with a previous error equidistribution process) locates the intermediate MF centers optimally. The optimal structure found by the optimization algorithm is \{3, 3, \{5, 4\}, \{2, 2, 2\}, \{5\}\}, with a total number of rules of 42.

Thus the final returned MGFS model has four sub-grids, two of two variables ((\(x_3, x_4\), \(x_1, x_2\)), one of three (\(x_6, x_7, x_8\)) and one of one (\(x_5\)), with the corresponding MF distribution and MF centers position obtained by the optimization algorithm. The final test error obtained was NRMSE = 0.0737.

3.4. Example 2: Mackey–Glass time series

A well-known benchmark for time series prediction is the Mackey–Glass time series. It is an artificial time series without noise, that has been widely used in the literature
for neuro-fuzzy and other nonlinear models comparisons.
This time series is described by the following delay equation:
\[
\frac{dx(t)}{dt} = \frac{a x(t-\tau)}{1 + x(t-6)} - b x(t).
\]  
(6)

A sequence of 1200 data points were generated with an initial condition \(x(0) = 1.2\) and \(\tau = 17\), using the fourth order Runge–Kutta method. From the 1200 data points generated, the last 1000 were taken, using 500 data for training, and the final 500 for testing. To make the comparisons with earlier works fair, the parameters were chosen so that the training vectors for the model have the following format:
\[
[x(t-18), x(t-12), x(t-6), x(t); x(t+6)].
\]  
(7)

For this problem, in opposition to the previous one, the optimal grouping of the input variables that should make up the MGFS architecture is unknown. Again, cross-validation was used to select the optimal number of rules for the architecture selection algorithm. The limit number of rules selected was 190. For this limit, the algorithm was launched using all the training data. Starting from the simplest configuration having four sub-grids with one variable each, the algorithm reached a sub-optimal architecture of three sub-grids of two variables each, discarding any other more complex structure. The sub-grids forming the MGFS are: \(\{x(t-12), x(t)\}\), \(\{x(t-12), x(t-6)\}\), \(\{x(t-12), x(t-18)\}\) each one having a \(8 \times 8\) MFs configuration. The test root mean-square error (RMSE) [20], that is the error measure normally used for this example in the literature) obtained with this architecture is 0.0039. The structure identification process shortly improved the performance to RMSE of 0.0033, with structure \(\{6, 6\}\), \(\{6, 6\}\), \(\{6, 8\}\).See Table 3 for a comparison of the MGFS methodology and other paradigms and interpretable methodologies presented in the literature. See how using the traditional GBFS model, the obtained optimal system has a higher number of parameters (a \(2, 3, 8, 4\) membership function structure) and obtains a worse performance.

### Table 1
MGFS architecture selection algorithm trace for Example 1

<table>
<thead>
<tr>
<th>Architecture</th>
<th>MF distribution</th>
<th>Training NRMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>{{x_1}, {x_2}, {x_3}, {x_4}, {x_5}, {x_6}, {x_7}, {x_8}}</td>
<td>{8, 8, 8, 8, 8, 8, 8}</td>
<td>0.278</td>
</tr>
</tbody>
</table>

### Table 2
Evolution of the structure identification algorithm for the architecture \(\{x_1, x_2\}, \{x_1, x_2\}, \{x_6, x_7, x_8\}\) (see Eq. (5))

<table>
<thead>
<tr>
<th>MGFS distribution</th>
<th>Number of rules</th>
<th>Cross-validation NRMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>{{1, 1, 1}, {1, 1, 1}, {1, 1, 1}, {1, 1}}</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>{{1, 1, 1}, {1, 1, 1}, {1, 1, 1}, {1, 2}}</td>
<td>5</td>
<td>0.665</td>
</tr>
<tr>
<td>{{1, 1, 1}, {1, 2, 1}, {1, 1, 1}, {1, 2}}</td>
<td>6</td>
<td>0.600</td>
</tr>
<tr>
<td>{{1, 1, 2}, {1, 2, 1}, {1, 1, 1}, {1, 2}}</td>
<td>8</td>
<td>0.549</td>
</tr>
<tr>
<td>{{1, 2, 2}, {1, 2, 1}, {1, 1, 1}, {1, 2}}</td>
<td>9</td>
<td>0.481</td>
</tr>
<tr>
<td>{{2, 2, 2}, {2, 2, 1}, {1, 1, 1}, {1, 2}}</td>
<td>11</td>
<td>0.371</td>
</tr>
<tr>
<td>{{2, 2, 2}, {2, 2, 1}, {1, 1, 1}, {1, 3}}</td>
<td>12</td>
<td>0.323</td>
</tr>
<tr>
<td>{{2, 2, 3}, {2, 2, 1}, {1, 1, 1}, {1, 3}}</td>
<td>14</td>
<td>0.275</td>
</tr>
<tr>
<td>{{2, 2, 3}, {2, 3, 1}, {1, 1, 1}, {1, 3}}</td>
<td>17</td>
<td>0.220</td>
</tr>
<tr>
<td>{{2, 2, 3}, {2, 3, 1}, {1, 2, 1}, {1, 3}}</td>
<td>18</td>
<td>0.191</td>
</tr>
<tr>
<td>{{2, 2, 3}, {2, 3, 2}, {1, 1, 1}, {1, 3}}</td>
<td>20</td>
<td>0.166</td>
</tr>
<tr>
<td>{{2, 2, 3}, {2, 3, 2}, {1, 2, 1}, {1, 3}}</td>
<td>24</td>
<td>0.113</td>
</tr>
<tr>
<td>{{2, 2, 3}, {2, 3, 2}, {2, 2, 1}, {1, 3}}</td>
<td>25</td>
<td>0.104</td>
</tr>
<tr>
<td>{{2, 2, 4}, {2, 3, 2}, {2, 2, 1}, {1, 3}}</td>
<td>28</td>
<td>0.0985</td>
</tr>
<tr>
<td>{{2, 2, 4}, {2, 3, 2}, {2, 2, 1}, {3, 3}}</td>
<td>32</td>
<td>0.0937</td>
</tr>
<tr>
<td>{{2, 2, 4}, {2, 3, 2}, {2, 2, 2}, {1, 3}}</td>
<td>34</td>
<td>0.0881</td>
</tr>
<tr>
<td>{{2, 3, 4}, {2, 3, 2}, {2, 2, 2}, {1, 3}}</td>
<td>37</td>
<td>0.0822</td>
</tr>
<tr>
<td>{{3, 3, 4}, {2, 3, 2}, {2, 2, 2}, {3, 3}}</td>
<td>38</td>
<td>0.0785</td>
</tr>
<tr>
<td>{{3, 3, 5}, {2, 3, 2}, {2, 2, 2}, {3, 3}}</td>
<td>42</td>
<td>0.0795</td>
</tr>
</tbody>
</table>

### Table 3
Comparison results of the prediction error of different methods for prediction step equal to 6 (500 training points)

<table>
<thead>
<tr>
<th>Method</th>
<th>Test RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto regression model</td>
<td>0.19</td>
</tr>
<tr>
<td>Cascade correlation NN</td>
<td>0.06</td>
</tr>
<tr>
<td>Back-prop. NN</td>
<td>0.02</td>
</tr>
<tr>
<td>6th-order polynomial</td>
<td>0.04</td>
</tr>
<tr>
<td>Linear predictive method</td>
<td>0.55</td>
</tr>
<tr>
<td>Kim and Kim (Genetic algorithm and fuzzy system)</td>
<td>5 MFs 0.049</td>
</tr>
<tr>
<td>7 MFs</td>
<td>0.042</td>
</tr>
<tr>
<td>9 MFs</td>
<td>0.038</td>
</tr>
<tr>
<td>Classical RBF with 23 neurons ((\geq 200) params)</td>
<td>0.0114</td>
</tr>
<tr>
<td>PG-RBF ((\geq 200) params)</td>
<td>0.0030</td>
</tr>
<tr>
<td>MGFS with 192 rules (136 params)</td>
<td>0.0031</td>
</tr>
<tr>
<td>GBFS with 192 rules (201 params)</td>
<td>0.0041</td>
</tr>
<tr>
<td>Neuro-fuzzy network [17] 9 rules (92 params)</td>
<td>0.0239</td>
</tr>
</tbody>
</table>

References were taken from [20].
3.5. Remarks on the effectiveness and computational efficiency of the learning algorithm

The MGFS model as seen presents a good performance, with additional advantages comparing to GBFSs. First the number of rules obtained for a similar performance can be much lower, and with a much lower number of antecedents. Furthermore, from the interpretability point of view, a data point, for which the output is to be estimated, needs to perform weighted average over \(2^n\) rules when using GBFSs. Using MGFSs, this is translated to a lower number of rules equal to \(\frac{1}{2^n}\) (where \(n_p\) is the number of input variables in the sub-grid \(p\)); moreover, the rules in MGFS have a lower number of antecedents, which makes interpretability easier.

With respect to the training computational cost, given a certain architecture, the MGFS methodology presents a similar training comparing to what traditional GBFS would carry out. A certain fixed set of MFs has to be obtained per input variable. This implies finding the optimal number of MFs per variable, and their optimal parameters. The linear parameters of the model (rule consequents) are similarly obtained too; however, the number of rules in MGFS can be much lower that in GBFSs for a similar objective performance as it was shown.

Additionally, an architecture selection process has to be performed when training MGFSs. It is a very complex problem, but thanks to the proposed incremental methodology, the search space is highly reduced. Moreover, thanks to the control of the search using a limit in the estimated number of rules of the model, computational bottlenecks are avoided. The total number of rules is automatically split among the sub-grids, and the training of each current architecture for its evaluation is performed using LSE (solving a linear equation system). As it was argued this limit in the number of rules is obtained using cross-validation, to obtain the expected trade-off between precision and generalization capability of the model as this limit increases. The computational effort for the architecture selection procedure will then depend mostly on the number of data points available, and the number of relevant input variables.

The control of overfitting in the learning methodology is twofold. On the one hand, the identification of the limit in the number of rules for the architecture selection algorithm is performed using cross-validation. This limit is used to get the most appropriate architecture for the given I/O data set. On the other hand the optimal model structure is calculated, given the model architecture, using an incremental process whose stop-criteria depends on the decrease of the cross-validation error. This twofold methodology assures the effectiveness and generalization of the final model obtained.

Next, the rest of the paper deals with the CATS time series prediction problem, and its solution using a set of MGFS direct prediction models.

4. Prediction of the CATS time series

4.1. The CATS time series

The CATS time series consists on 5000 data points in which 100 of these values are missing. These missing values are divided in five blocks:

1. elements 981 to 1000;
2. elements 1981 to 2000;
3. elements 2981 to 3000;
4. elements 3981 to 4000;
5. elements 4981 to 5000.

The task in this problem is to predict these 100 values, and the quality of the obtained prediction is evaluated computing the mean square error \(E\) on the 100 missing values using thus the formula

\[
E = \frac{\sum_{m=0}^{4} \sum_{j=1000m+981}^{1000m+991} (y_j - \hat{y}_j)^2}{100}.
\]

A complete plot of the first 5000 data provided to the competitors, with the final solutions for the five gaps, is shown in Fig. 6. A second sequence of the time series with 5000 new elements and five new gaps to be predicted, that was provided after the IJCNN competition, is shown in Fig. 7.

4.2. Data analysis

For this CATS time series prediction problem, a single series of unidimensional data samples has been provided. No additional information of the underlying data model has been given.

Taking a first look to the original time series it can be noted that, in principle, there is not any repetitive pattern. The data follows a certain noisy path, but with no identifiable pattern (see Figs. 6 and 7), in opposite to other well-known time series benchmarks [6,14,15].

![Fig. 6. First sequence of 5000 data samples of the CATS time series, with the solution for the five gaps.](image-url)
In general, a predictive model that takes into account previous outputs as input variables has the form
\[ y(t + h) = F(y(t - i_1), y(t - i_2), \ldots, y(t - i_m)), \] (9)
where \( h \) is the prediction horizon and typically the indices \( i_1, i_2, \ldots, i_m \) have the form \( 0, \tau, 2 \times \tau, \ldots, (m - 1) \times \tau \), where \( \tau \) is the sampling time.

### 4.3. Differentiated time series

As noted in [15], instead of considering the original time series, the prediction problem can be analyzed using the ‘differentiated’ time series \( u(t) = y(t) - y(t - 1) \). New information is taken into account, considering the differences from one data point with respect to the previous one. The original time series can be expressed therefore as
\[ y(t) = y(t - 1) + u(t). \] (10)

The whole ‘differentiated’ time series is shown in Fig. 8, and a partial detail is shown in Fig. 9. The advantage of using the differentiated time series instead of the original one, is that all the data are inside a certain shorter domain that allows a better coverage of the input domain. The application of the MGFS model to the CATS time series prediction problem will use the differentiated time series. Sections 2 and 3 presented the proposed MGFS model and learning methodology. Next, Section 5 shows the application of the MGFS model to the CATS time series prediction problem using direct prediction. Section 6 introduces some improvements performed with respect to the initial work [8], and Section 7 shows the results obtained for the new data set of the CATS series [28].

### 5. Modelling CATS through the differentiated time series

The sample partial autocorrelation function (PACF) of the differentiated time series \( u(t) \) (see Fig. 10), shows that only the first 20–25 previous data might influence the prediction of each data \( \hat{u}(t) \) using a linear model.

For the MGFS model, several tests were performed taking even up to 100 previous values to predict the \( \hat{u}(t + 1) \) (see Eq. (9)), but the results obtained showed that considering \( i_m \) higher that 20–25 does not provide better performance. Note in any case that in Fig. 10, a lag \( l \) corresponds to the effect of variable \( u(t - l + 1) \) over the
desired \( \hat{u}(t + 1) \). The previous 20 values \( u(t - 0) \ldots u(t - 20) \) were finally selected to launch the MGFS architecture selection algorithm with the initial objective of predicting \( \hat{u}(t + 1) \),

\[
\hat{u}(t + 1) = F(u(t - 0), u(t - 1), \ldots, u(t - 20)).
\]  

The I/O training data \( [u(t + 1); u(t - 0), u(t - 1), \ldots, u(t - 20)] \) was randomly divided into training (4000 data samples approx) and test (1000 data samples). This data set obviously does not contain the five gaps in the sequence of 5000 data samples.

The first step is to select the limit in the number of rules for the MGFS architecture selection algorithm. A 5-fold cross-validation procedure was performed as it was discussed in Section 3.1, checking the range 50–150. The selected limit was 60; from that value, the cross-validation error was stabilized. This cross-validation procedure took 4 h in a P-2.8 Ghz PC.

The MGFS architecture selection algorithm starts from an architecture having \( n \) sub-grids (\( n = 20 \) in this case) of 1 variable. The leave-one-out sub-grid elimination procedure was done after this initialization (as it is explained in Section 3.1). That is, the starting architecture had \( p \) sub-grids of 1 variable, that were those showing to be relevant to the output of the model. The MGFS architecture obtained after the leave-one-out elimination procedure is:

\[
A = \{u(t - 2), [u(t - 3), \{u(t - 13), \{u(t - 15)\}]\}
\]

The number of rules is distributed as it was mentioned in Section 3.1 (i.e. 15 for each sub-grid). The first iteration of the algorithm tries to find second order sub-grids, obtained by adding any of the 20 variables to the first order sub-grids. Further iterations (\( m \)) would do similarly with \( m \)th order sub-grids of the architecture. The algorithm ends up selecting the following MGFS architecture: \( A = \{u(t - 2), [u(t - 3), [u(t - 15), u(t - 1), [u(t - 13), u(t - 12)]\}] \) (the computing time needed for this case was about 10 min in a P-2.8 Ghz PC)

\[
\hat{u}(t + 1) = F^1(u(t - 2)) + F^2(u(t - 3)) + F^3(u(t - 15), u(t - 1)) + F^4(u(t - 13), u(t - 12)).
\]  

The structure identification algorithm is later performed to obtain the optimal structure for the given architecture. Table 4 shows the evolution of the structure identification process for the current problem. Further iterations of the algorithm did not bring an improvement of the cross-validation error. The algorithm obtained an optimal MF distribution that, with only 37 rules obtains a similar approximation than with the 60 rules distribution obtained after the architecture selection algorithm. The MF parameter adjustment procedure was performed from the seventh iteration (when the amount of MFs of any variable reaches 3; see Section 3.2).

The test error obtained with the final MGFS model with the architecture specified in Eq. (12), and with structure

\[
\{[2], [3], [4, 3], [5, 5]\}
\]

is 0.815 NRMSE (with a similar value for the cross-validation error 0.805).

To predict the remaining points of the gaps \( (u(t + 2), u(t + 3), \ldots, u(t + 20)) \), this work uses a direct prediction approach [27]. Direct prediction presents the drawback that several models need to be trained, one for each prediction horizon. Nevertheless each model can be trained separately in order to get a better performance. The alternative is to use recursive prediction, that uses the same model \( \hat{u}(t + 1) \) to predict all the possible horizons. However, the error obtained to predict \( \hat{u}(t + 1) \) is too high, and this error would be dragged to the prediction of the following points in the gaps.

Table 4 shows the evolution of the structure identification algorithm for the MGFS with architecture in Eq. (12).

<table>
<thead>
<tr>
<th>MGFS Structure</th>
<th>Number of Rules</th>
<th>Cross-validation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>{[1], [1], [1], [1]}</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>{[1], [1], [1], [2]}</td>
<td>5</td>
<td>0.910</td>
</tr>
<tr>
<td>{[2], [1], [1], [1]}</td>
<td>6</td>
<td>0.871</td>
</tr>
<tr>
<td>{[2], [2], [1], [1]}</td>
<td>7</td>
<td>0.850</td>
</tr>
<tr>
<td>{[2], [2], [2], [1]}</td>
<td>8</td>
<td>0.841</td>
</tr>
<tr>
<td>{[2], [2], [2], [2]}</td>
<td>10</td>
<td>0.831</td>
</tr>
<tr>
<td>{[2], [2], [2], [2]}</td>
<td>12</td>
<td>0.821</td>
</tr>
<tr>
<td>{[2], [2], [2], [2]}</td>
<td>13</td>
<td>0.815</td>
</tr>
<tr>
<td>{[2], [2], [3], [3]}</td>
<td>15</td>
<td>0.813</td>
</tr>
<tr>
<td>{[2], [2], [3], [3]}</td>
<td>18</td>
<td>0.812</td>
</tr>
<tr>
<td>{[2], [3], [2], [3]}</td>
<td>20</td>
<td>0.810</td>
</tr>
<tr>
<td>{[2], [3], [2], [3]}</td>
<td>23</td>
<td>0.809</td>
</tr>
<tr>
<td>{[2], [3], [3], [4]}</td>
<td>26</td>
<td>0.808</td>
</tr>
<tr>
<td>{[2], [3], [3], [5]}</td>
<td>29</td>
<td>0.807</td>
</tr>
<tr>
<td>{[2], [4], [3], [5]}</td>
<td>32</td>
<td>0.806</td>
</tr>
<tr>
<td>{[2], [3], [4], [5]}</td>
<td>37</td>
<td>0.805</td>
</tr>
</tbody>
</table>

The time needed to obtain all the models was approximately 3 h.
in a P-800 MHz PC for the architecture selection algorithm, and a similar computing time for the additional optimization procedure (structure selection and parameter adjustment). It must be mentioned about these results, that the error increases as the horizon increases. Before it was argued that the previous 20 values showed to give more information about \( u(t + 1) \); this explains why as the horizon increases, the available information decreases. Note also how the algorithm obtains sometimes similar sub-optimal models for close horizons \( H \) in \( \hat{y}(t + H) \) (for instance \( \hat{y}(t + 1) \) and \( \hat{y}(t + 2) \), \( \hat{y}(t + 5) \) and \( \hat{y}(t + 6) \), etc.). This supports the robustness of the architecture selection algorithm. The second possibility, i.e. Eq. (14), showed to provide a worse prediction error both for training and test than the first alternative, i.e. Eq. (13).

An example of the approximation of a complete section of 20 data points using the models in Table 5 is shown in Fig. 11. The twenty values of the gap are predicted using the models in 5 for the differentiated time series, and those predicted difference values are used to recover the expected original time series by inverting 10. The overall estimated MSE is 1000 for this forward prediction complex model.

However, for the first four gaps (first 80 points) to be predicted, the continuation of the time series is known. The same methodology can be applied, reversing the time series to predict the four sections of 20 data points (taking into account the following data as well as the previous data).

A new set of 20 similar models have been obtained using this reversed prediction model.

\[
\hat{u}(t - H) = F(u(t + 0), u(t + 1), \ldots, u(t + 20)).
\] (15)

A weighted average strategy has been applied to combine the output of the two sets of direct-models (forwards and backwards) to predict the first four gaps of 20 unknown data points are available. The weights of the two model outputs have been obtained by considering their error performance, both for \( \hat{y}(t + H) \) and \( \hat{y}(t - H) \).

The final expected MSE for this combined methodology taking into account forward and backward prediction for the unknown data points is approximately 330. This expected MSE was calculated in 500 executions, by considering artificial gaps of 20 samples in the whole series of 5000 samples, and trying to predict them. For the last 20 data points to be predicted, i.e. from 4980 to 5000, only the forward model \( \hat{u}(t + 1) \ldots \hat{u}(t + 20) \) would be used.

The final predictions for the five gaps (100 data) for the first 5000 series data are shown in Figs. 12–16. The mean-square error (MSE) obtained for the prediction of the first 80 data points (first four gaps) is 352, for the last 20 data points (last gap) the MSE is 1186 and finally, the MSE for
the total 100 data points is 518 (note that there are differences with the results published in [8,13] due to a wrong implementation of the weighted average to combine the output of the forward and backward models).

6. Improvements on the presented model: reflecting the trend of the time series in the forward–backward prediction

The last section discussed how for the first four gaps (in which the previous data and the following data of the series are available), a combined forwards-backwards direct prediction has been performed. The weighted average of both predictions showed to perform well, but it is not optimal, in the sense that it might not reflect completely the trend of the time series.

For the forward prediction, the $20 \dot{y}(t+i), i = 1 : 20$ predicted data have been stretched so that the last predicted data $\dot{y}(t + 20)$ coincides with the last point of a line that joins the known data $y(t)$ and $y(t + 21)$. This modification in the forward prediction leads to obtain an MSE of 314 by itself. The same procedure has been performed for the backward prediction, stretching the predicted values so that the first predicted data $\dot{y}(t + 1)$ coincides with the first point of the line that joins $y(t)$ and $y(t + 21)$. The backward prediction with this modification obtains an MSE of 318.
Finally the two predictions obtained for the first four gaps have been joined using weighted average to reach an expected MSE of 284. See [13] for a comparison with other approaches. See Figs. 12–15 (in dotted line) to check the new predictions for the four gaps of the first 5000 data samples of the CATS time series.

7. New results obtained on the new values of the benchmark

According to the procedure explained in Sections 5 and 6, Figs. 17–20 show the prediction for the four first gaps for the second 5000 data samples provided shown in Fig. 7 that have a similar expected MSE of 284.

And finally Fig. 21 shows the prediction for the final gap of the second 5000 data samples provided, using the 20 forward prediction model explained in Section 5.

8. Conclusions

This paper presents a novel learning algorithm for multigrid-based fuzzy systems, including architecture selection and structure identification. This type of additive grid-based fuzzy system allows keeping a low computational cost while being able to consider a high number of input variables. It keeps the advantages of traditional grid-based fuzzy systems, but it can be applied to high-dimensional problems. It selects the most relevant interrelations among the input variables with respect to the output variable to perform the approximation. This approach is therefore very suitable for function approximation and time series forecasting problems in which the
input variables involved are not determined, and an effective interpretable solution is needed.

The MGFS model has been applied to the CATS time series prediction using a modified direct approach [27]. For the first four gaps to be predicted, in which the continuation of the time series is available, a set of forwards-backwards predictors has been obtained. The global predictor also takes into account the trend in the time series to reach a better solution. For the last gap of the series, only the forwards set of models was used. All the MGFS models obtained are composed of a reduced set of simple rules with a low number of antecedents. The results obtained show that this modified direct approach based on MGFSs presents a good performance for this time series prediction problem.

As further work, on the one hand it is intended to improve the robustness of the architecture selection algorithm, by better adapting the partitioning of the rules along the MGFS architecture to each specific problem. Also it is intended to study the use of higher order consequents (not only constant), which could lead to obtaining models with a lower number of rules without loss in interpretability [11].

Acknowledgement

This work has been supported by the Spanish CICYT Project TIN2004-01419.

References


Luis Javier Herrera was born in 1978. He received the M.Sc. degree in Computer Engineering in 1995 from the University of Granada, Spain. He is currently an associate lecturer within the Department of Computer Architecture and Technology in the University of Granada. His current areas of research interest are in the fields of function approximation, self-organizing fuzzy interpretable models and time series prediction.

Hector Pomares was born in 1972. He received the M.A.Sc. degree in Electronic Engineering in 1995, the M.Sc. degree in Physics in 1997, and his Ph.D. degree in 2000 from the University of Granada, Spain. He is currently a Fellowship holder within the Department of Computer Architecture and Technology in the University of Granada. His current areas of research interest are in the fields of function approximation and on-line control using adaptive and self-organizing fuzzy systems.
Ignacio Rojas received the B.Sc. degree in electronic physics in 1992 and the Ph.D. degree in Intelligent Systems with honors in 1996, both from the University of Granada, Spain. He has been visiting professor at the University of Dortmund (Germany) at the Department of Electrical Engineering (1993–1995, 2001), as a visiting researcher with the BISC Group of Prof. L. Zadeh, University of California, Berkeley (1998) and as visiting professor at the University of Genova (Italy), Department of Computer Science (2003). He is currently an Associate Professor with the Department of Computer Architecture and Computer Technology, University of Granada, member of the IEEE Computational Intelligence Society and Secretary of the IEEE Spanish Regional Interest Group of the Neural Networks Council. His research interests include hybrid system, hardware-software implementation, combination of intelligent system for adaptive control, self-organizing neuro-fuzzy systems, neural networks, time series forecasting, data mining and architectures for complex optimization problems.

Alberto Guillén was born in 1979. He received the M.A.Sc. degree in Computer Science in 2002, and the M.Sc. degree in 2004 from the University of Granada, Spain. He is currently an assistant lecturer at the Department of Computer Science in the University of Jaén. His current areas of research interest are in the fields of clustering, function approximation, and parallel genetic algorithms.

Jesús González was born in 1974. He received the M.Sc. degree in Computer Science in 1997 and the Ph.D. degree in 2001, both from the University of Granada, Spain. He is currently an Associate Professor in the Department of Computer Architecture and Computer Technology at the same University. His current areas of research interest are in the fields of embedded systems and function approximation using radial basis function neural networks, fuzzy systems, and evolutionary computation.

Mohammed Awad was born in 1976. He received the B.Sc. degree in Industrial Automation Engineering in 2000 from the Palestine Polytechnic University and the Ph.D. degree in 2005, from the University of Granada, Spain. He is currently Assistant Professor in the Faculty of Information Technology and Director of the Department of computer information Technology at the Arab American University, Palestine. His current areas of research interest include artificial neural networks and evolutionary computation, function approximation using radial basis function neural networks and fuzzy systems.

Ana M. Herrera was born in 1968. She received the M.A.Sc. degree Mathematics in 1990 from the University of Granada, Spain. She is currently working as a high school teacher, and she’s actively collaborating with the Department of Computer Architecture and Technology in the University of Granada. Her current areas of research interest are in the fields of function approximation, self-organizing fuzzy interpretable models and time series prediction.