ATLANTIS PRESS

Atlantis Studies in Uncertainty Modelling, volume 1

11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019)

Granular Evolving Min-Max Fuzzy Modeling

Alisson Porto[a] **and Fernando Gomide**[b]
[a] [b]University of Campinas, Brazil, [alisport, gomide]@dca.fee.unicamp.br

Abstract

The paper addresses a novel evolving functional fuzzy modeling algorithm using hyperboxes and min-max fuzzy granulation. Data space granulation is done as data are input, and undergoes never ending adaptation using expansion and reduction operations to encompass new information. A fuzzy rule is assigned to each hyperbox using Gaussian membership functions in the rule antecedents, and affine functions in the rule consequents. The algorithm is fast, simple, and interpretable. Computational evaluation using time series modeling and nonlinear system identification experiments shows that the granular evolving min-max fuzzy modeling algorithm outperforms current state of the art evolving algorithms counterparts.

Keywords: evolving systems, fuzzy modeling, fuzzy min-max algorithms.

1 Introduction

Currently, modeling of real-world system dynamics online is of utmost importance in many applications domains. The need to learn and process complex nonlinear relationships and nonstationarity embedded in data is increasing rapidly. For instance, in economics there are evidences that portfolio returns (or log returns) are neither normal nor stationary. Market returns show structural shifts, time varying negative skewness, and excess kurtosis [7].

Evolving modeling has been shown to be a powerful approach to address challenging nonlinear, nonstationary data-driven modeling. Evolving fuzzy systems are an advanced form of adaptive systems that simultaneously learn the model structure and its functionality from data. An overview of existing approaches is found in [6].

Recently a parsimonious network-based fuzzy inference algorithm was developed to learn dynamic systems with shifts and drifts [10]. The algorithm is an advanced form of an evolving neuro-fuzzy like approach based on incremental learning. In the spirit of evolving modeling approaches, the algorithm starts from scratch and updates the existing rule base to accommodate uncovered time varying data distribution. A particular feature of the algorithm is the use of ellipsoids in arbitrary position of the data space to keep local correlation information of the variables. Projection of multidimensional membership functions forms the antecedent of the fuzzy rules, but fuzzy inference is done with higher dimensional ellipsoidal representations.

In this paper we show that the evolving min-max fuzzy granular modeling algorithm gives a simpler and competitive, yet compact alternative to current state of the art evolving modeling algorithms. The granular evolving min-max algorithm is fast, retains interpretability using classic functional rule base format, and does not need to build clusters in arbitrary positions because data orientation is captured by local affine models instead.

The paper is organized as follows. Next section reviews the evolving min-max fuzzy modeling approach addressed in this paper, and the ideas behind each of its modeling steps. The detailed algorithm steps are summarized in Section III. Section IV addresses the use of the evolving min-max fuzzy algorithm to forecast the S&P 500 index, to identify a nonlinear system, and to compare their performance with state of the art evolving algorithms. The result shows that the evolving min-max fuzzy modeling algorithm is a highly efficient and competitive alternative.

2 Evolving Fuzzy Min-Max Modeling

This section details the granular evolving min-max fuzzy modeling (eFMM). The algorithm granulates the

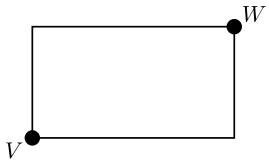


Figure 1: Hyperbox in I^2

input data space using hyperboxes and associates a functional fuzzy rule to each hyperbox. The input space granulation is an enhanced version of the one reported in [8]. A hyperbox reduction process is developed to improve rule robustness in the presence of concept drift, and to increase prediction performance. Once a rule base structure is constructed, the algorithm estimates the coefficients of the affine consequent functions using the recursive least squares with forgetting factor. The learning algorithm also evaluates the rule base quality to identify unnecessary and redundant rules. The purpose is to produce parsimonious, yet representative rule bases.

2.1 Modeling framework

The first step of eFMM granulates the input data space using hyperboxes. The input data space is the n-dimensional unit cube I^n. A hyperbox is a n-dimensional rectangle defined by a maximum (W) and a minimum (V) points, Fig.1. One appealing feature when using hyperboxes is that they are uniquely defined by these two points regardless of the input data space dimension. A hyperbox is defined as follows:

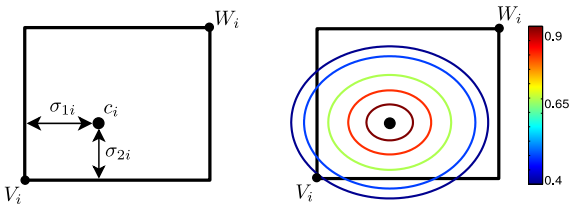
$$B_i = \{I^n, V_i, W_i, b_i(x, V_i, W_i, c_i)\} \tag{1}$$

where I^n denotes the input data space, b_i is the membership function associated with the i-th hyperbox, $x \in I^n$ is an input data point, and V_i, W_i, $c_i \in I^n$ are the minimum, maximum, and the center points, respectively.

The eFMM is a fuzzy rule-based modeling approach whose rules have local models forming their consequents, referred to as functional fuzzy models or Takagi-Sugeno models. The eFMM adopts affine functions in the rule consequents. A model is composed by a set of R fuzzy rules of the form:

$$R_i : \text{If } x \text{ is } B_i \text{ then } \bar{y}_i = \theta_{0i} + \sum_{j=1}^{n} \theta_{ji} x_j \tag{2}$$

where R_i is the i-th fuzzy rule, $\bar{y}_i$ is the rule output, θ_{ji}, $i = 1, ..., R$, $j = 1, ..., n$ are the consequent parameters, and R is the number of fuzzy rules in the rule base. The collection of the R fuzzy rules assembles the model as a weighted combination of the local



(a) Dispersions (b) Membership levels

Figure 2: Dispersions and membership levels

affine models. The contribution of each local model to the model output is proportional to the normalized firing degree of each rule. eFMM uses antecedent fuzzy sets built as the product of elementwise Gaussian membership functions:

$$b_i = \prod_{j=1}^{n} \bar{b}_{ji} \tag{3}$$

$$\bar{b}_{ji} = exp\left(\frac{-(x_j - c_{ji})^2}{2\sigma_{ji}^2}\right) \tag{4}$$

$$\sigma_{ji} = min(w_{ji} - c_{ji}, c_{ji} - v_{ji}) \tag{5}$$

where $\bar{b}_{ji}$ is the j-th component of the i-th rule membership function, σ_{ji} is the width, x_j is the j-th component of the n-dimensional input data x, and w_{ji}, v_{ji}, c_{ji} are the components of the i-th rule maximum, minimum, and center points, respectively. Fig. 2(a) shows the hyperbox B_i with the corresponding dispersions, and Fig. 2(b) illustrates the associated membership function levels.

Most of the min-max techniques use membership functions that assign maximum membership levels to any point confined within a hyperbox [3]. This approach creates inconsistencies when two different hyperboxes superpose each other because in this situation there are regions in the data space that are fully compatible with both boxes. An alternative to remedy the inconsistency is to identify and eliminate superposition regions using contraction methods. Even though contraction effectively eliminates superimposed regions, it adds errors in the prediction performance. The eFMM algorithm uses normal Gaussian membership functions instead, and hence only the rule center has full membership degree. This approach eliminates the need for contraction procedures, allows hyperboxes superpositions, and produces smooth transitions between the different local models. Furthermore, Gaussian membership functions have infinite support, preventing data points from having null membership level. Another interesting feature is that a linear combination of Gaussian functions can uniformly approximate any real continuous function on compact sets to arbi-

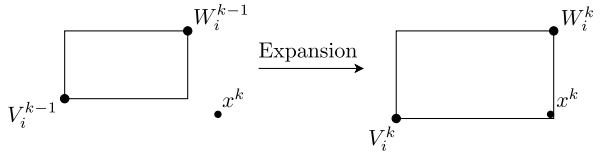
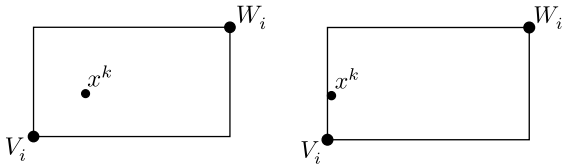


Figure 3: Expansion



(a) Hold (b) Do not hold

Figure 4: Reduction condition

trary accuracy. Likewise, fuzzy functional models with Gaussians are universal approximators [4].

The output of the eFMM model at step k is computed as the weighted average of the individual rule contributions, that is

$$\hat{y} = \sum_{i=1}^{R} \psi_i \theta_i^T \bar{x}^k, \quad \psi_i = \frac{b_i}{\sum_{l=1}^{R} b_l} \tag{6}$$

where ψ_i is the normalized firing degree of the i-th rule, y_k is the model output at step k, and $\bar{x}^k = [1, x_1^k, x_2^k, ..., x_n^k]^T$ is the extended input vector.

2.2 Data space granulation

The eFMM processes the input data stream and decides, for every new data point, if a new rule must be created, or an existing one should be modified. Rule modification is composed by three steps: expansion, center adjustment and reduction. This section details these three procedures.

Expansion

The expansion allows a hyperbox to encompass a new data point. This is done by displacing the maximum (W) and minimum (V) points to accommodate the data within the hyperbox borders. Fig.3 illustrates the expansion of the hyperbox B_i. Expansion is performed as follows:

$$W_i^k = max(W_i^{k-1}, x^k) \quad V_i^k = min(V_i^{k-1}, x^k) \tag{7}$$

$$M_i^k = M_i^{k-1} + 1 \tag{8}$$

where M_i^k is the i-th rule counter, which stores the number of points included by rule B_i up to the step k.

Center update

Every hyperbox has a center point formed by the average of all data samples encompassed by the hyperbox. The center point plays a key role in the determination of the membership function associated with the rule. The membership function spread in a given direction is calculated as the distance between the rule center and the rule boundary in that direction, as Fig. 2(a) illustrates. Whenever a hyperbox B_i is expanded, its center is updated as follows:

$$c_i^k = \frac{M_i^k - 1}{M_i^k} c_i^{k-1} + \frac{1}{M_i^k} x^k \tag{9}$$

Reduction

Reduction aims at decreasing the hyperbox size along a subset of the input coordinates. The procedure is important in some situations because excessively large hyperboxes may not be able to satisfy the expansion condition (detailed in Section 2.3), and therefore will not encompass new data points. Reduction is performed independently in each dimension of the input space. As during expansion, the hyperbox has to satisfy a specific condition before being reduced in a given dimension:

$$\textbf{IF} \ \left(v_{ji}^k < x_j^k \ \ \textbf{AND} \ \ x_j^k < w_{ji}^k\right), \tag{10}$$

$$\textbf{THEN:} \begin{cases} \textbf{IF} \ \left(w_{ji}^k - c_{ji}^k\right) > \left(c_{ji}^k - v_{ji}^k\right), \textbf{THEN:} \\ w_{ji}^k = (1-\alpha)\, w_{ji}^{k-1} + \alpha \left(c_{ji}^k + \sigma_{ji}^k\right) \\ v_{ji}^k = v_{ji}^{k-1} \\ \\ \textbf{IF} \ \left(w_{ji}^k - c_{ji}^k\right) < \left(c_{ji}^k - v_{ji}^k\right), \textbf{THEN:} \\ v_{ji}^k = (1-\alpha)\, v_{ji}^{k-1} + \alpha \left(c_{ji}^k - \sigma_{ji}^k\right) \\ w_{ji}^k = w_{ji}^{k-1} \end{cases} \tag{11}$$

The condition (10) evaluates the position of each x^k component within hyperbox B_i. If the component x_j is located in one of the B_i boundaries along dimension j, then reduction will not be performed in that dimension at step k. Otherwise, hyperbox B_i is decreased along dimension j. Fig.4 illustrates two scenarios: In Fig.4(a) reduction requirements hold along the horizontal axis, whereas in Fig. 4(b) they do not.

When the reduction condition is met, the maximum or the minimum point is displaced to reduce hyperbox B_i along dimension j (depending on condition (11)). Fig. 5 shows reduction along the horizontal axis. The reduction is performed only along the largest distance between the rule center and the hyperbox boundary. In Fig. 5, the distance from c_i^k to the left border is greater than the distance to the right border, so the

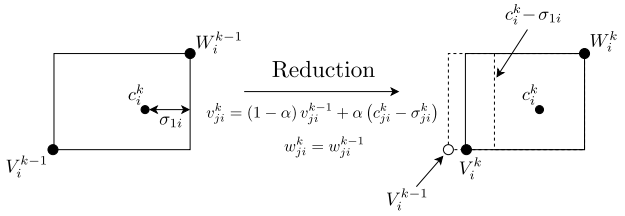


Figure 5: Reduction.

minimum point is displaced to decrease the distance between center and left border.

2.3 Online parameter adjustment

This section details the online adjustment of two learning parameters: The learning rate (α) and the maximum hyperbox size (δ).

Parameter α

The parameter α influences the rules updating pace, and therefore, it is commonly referred to as the learning rate. One usual approach is to require the user to determine a suitable value for α, and then keep this value unchanged during all the processing steps. Alternatively, the eFMM algorithm continuously updates α upon auxiliary parameters provided by the user. The value of α at step k is estimated as follows:

$$\alpha^k = \left(max\left(1 - \frac{|\bar{y}^k - y^k|}{max_{err}}, 0\right)\right)^t max_\alpha \qquad (12)$$

This approach produces a value for α accordingly to a exponentially decaying function of the local prediction error. When the error is low, α has higher values, and the learning is expedited. The value of max_{err} determines the maximum local error which produces $\alpha > 0$. The maximum possible value for α, which is generated when $\hat{y} - y^k = 0$, is determined by the max_α parameter. The value of t sets the exponential decay rate.

Even though this approach needs three new parameters to be chosen by the user, there is a set of empirically specified values that produce reasonable prediction performance in all experiments performed in the current and previous [9] versions of the proposed algorithm. These values are: $max_\alpha = 0.03$, $max_{err} = 0.3$, $t = 10$.

Parameter δ

The parameter δ controls the maximum length a hyperbox may reach in each dimension using the condition:

$$max(w_{ji}^k, x_j^k) - min(v_{ji}^k, x_j^k) \leq \delta_{ji} \qquad (13)$$

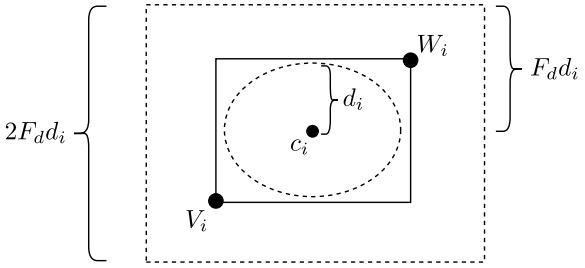


Figure 6: Maximum hyperbox size.

A hyperbox must first satisfy condition (13) to be expanded. Distinct dimensions may have different maximum lengths, and δ_i is a vector of n components ($j = 1, 2, ..., n$). Initially, the user chooses a value δ_0 valid for all dimensions. As data are input, the algorithm estimates the local dispersion and adjusts the maximum size as follows:

$$d_{ji}^k = \sqrt{\frac{M_i - 1}{M_i}\left(d_{ji}^{k-1}\right)^2 + \frac{1}{M_i}\left(x_j - c_{ji}\right)^2} \qquad (14)$$

$$\delta_{ji}^k = (1-\alpha)\,\delta_{ji}^{k-1} + 2\alpha F_d d_{ji}^k \qquad (15)$$

where d_{ji}^k is the local data dispersion for the j-th dimension. The value chosen for F_d is 2, so the maximum size will gradually converge to a value which covers the distance of 2 data dispersions from the center. The factor 2 on the left size of α is used because the hyperbox should be able to expand in two ways i.e. left and right from the center, up and down from the center, and so forth. Fig. 6 illustrates d_i, $F_d d_i$ and $2F_d d_i$ for the vertical coordinate.

2.4 Local parameter estimation

Once the rules antecedents are constructed, the eFMM estimates the parameters of the local affine models of the rule consequents using the recursive least squares with forgetting factor [2]. Forgetting factor emphasizes recent over old data when estimating the local models to enhance adaptability in nonstationary environments.

Let $\theta_i = [\theta_{0i}, \theta_{1i}, ..., \theta_{ni}]^T$ be the local model parameters for the i-th rule consequent. Then, θ_i is recursively updated using:

$$K = \frac{P_i^{k-1}}{\gamma + (\bar{x})^T P_i^{k-1} \bar{x}} \bar{x} \qquad (16)$$

$$\theta_i^k = \theta_i^{k-1} + K(y^k - \bar{y}_i^k) \qquad (17)$$

$$P_i^k = \frac{1}{\gamma}(I - K(\bar{x}^k)^T) P_i^{k-1} \qquad (18)$$

where γ is the forgetting factor, $\bar{y}_i^k$ is the i-th rule output at k, I is the identity matrix, and P is initially set as $P^0 = \omega I$, $\omega = [100, 10000]$.

2.5 Rule base redundancy

The eFMM continuously adapts its rule base upon new information present in the data. In this process, new rules are created and existing ones are constantly modified. In such a scenario a subset of rules, once useful and representative, may become obsolete. Keeping these obsolete rules in the rule base will unnecessarily increase rule base complexity, and eventually deteriorate prediction performance.

On the other hand, the dynamical behaviour of the learning algorithm can bring a rule pair close together to the point where they could be replaced by a unique rule without losing prediction performance and knowledge of the underlying process. In this situation, it is desirable to merge the similar rules to avoid needless complexity. The eFMM has deleting and merging mechanisms to manage the aforementioned issues to enhance flexibility and avoid redundant rules.

Deleting rules

The eFMM uses the utility measure [1] to identify representative rules. The utility, a measure of how often a rule is activated, is computed using:

$$U_i^k = \frac{\sum_{p=1}^{k} \psi_i^p}{k - I_i^*} \tag{19}$$

where U_i^k is the i-th rule utility at k, I_i^* is the step at which the i-th rule was created and ψ_i^p is the i-th rule activation level at step p.

Rules with low activation levels are candidates for exclusion. Exclusion is done when the following condition holds:

$$U_i^k = \frac{\sum_{p=1}^{k} \psi_i^p}{k - I_i^*}, \;\; \bar{U}^k = mean\{U_i^k\} \tag{20}$$

$$\textbf{IF } U_i^k < \epsilon \bar{U}^k, \textbf{ THEN } delete\{B_i\} \tag{21}$$

where $\bar{U}^k$ is the mean utility for all the rules in the rule base at k, and ϵ is a user defined parameter.

Merging rules

The eFMM has an automatic mechanism to merge rules using the centers, maximum and minimum points to decide when they should be merged. The method evaluates three conditions, and if at least one condition is fulfilled, then the rules are merged. The three conditions are:

$$\text{condition 1}: v_{ji} \leq v_{jl} \text{ AND } w_{ji} \geq w_{jl}, \text{ for } j = 1, 2, ..., n$$
$$\text{condition 2}: v_{jl} \leq v_{ji} \text{ AND } w_{jl} \geq w_{ji}, \text{ for } j = 1, 2, ..., n$$
$$\text{condition 3}: v_{ji} \leq c_{jl} \leq w_{ji} \text{ AND } v_{jl} \leq c_{ji} \leq w_{jl} \text{ AND } vol_{il} < vol_i + vol_l$$

$$vol_l = \prod_{j=1}^{n} (w_{jl} - v_{jl}), \; vol_i = \prod_{j=1}^{n} (w_{ji} - v_{ji}) \tag{22}$$

$$vol_{il} = \prod_{j=1}^{n} (max(w_{ji}, w_{jl}) - min(v_{ji}, v_{jl})) \tag{23}$$

$$\textbf{IF } \text{condition 1 OR condition 2 OR condition 3} \tag{24}$$
$$\text{THEN merge}\{B_i, B_l\}$$

where vol_i, vol_l and vol_{il} are the volumes of the hyperboxes i, l, and the hyperbox il formed by the union of hyperboxes i and l. Fig. 7 illustrates the union hyperbox il, delimited by the dashed lines, in two distinct cases. In the first case, $vol_{il} < vol_i + vol_l$ and hyperboxes i and l include each other centers. These are the requirements to fulfill the merging condition 3, and therefore, hyperboxes i and l should be merged. In the second case, however, $vol_{il} > vol_i + vol_l$, the condition is not met, and hence the hyperboxes are not modified.

The merging conditions 1 and 2 mean that if one of the hyperboxes completely comprises the other, then they should be merged.

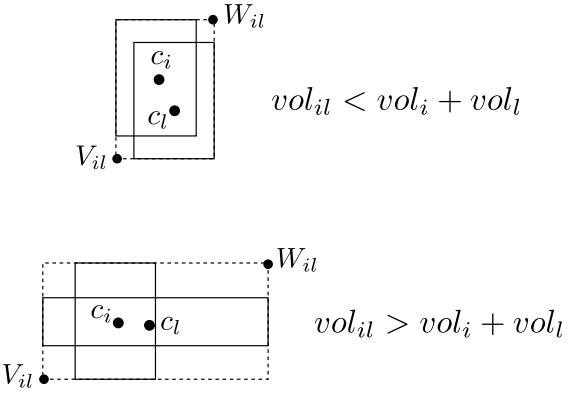


Figure 7: Merging condition satisfied (first case) and not satisfied (second case)

3 Evolving Fuzzy Min-Max Algorithm

The eFMM processes input data in an online fashion to learn the underlying input/output data relationship. The model starts from scratch, with an empty rule base, and gradually inserts new rules or modifies existing ones. The first data point becomes the first rule center, maximum and minimum points: $c_1 = W_1 = V_1 = x^1$. Whenever a new data point is received, the algorithm computes the activation level of all rules and then verifies if any existing rule can expand to include the current data point, in descending activation order. If no rule can expand, a new rule is created as follows:

$$R = R + 1, c_R = W_R = V_R = x^k \tag{25}$$

After processing the current data point, the algorithm assesses and adapts the rule base when necessary. The model output is computed at the beginning of every processing step. The Algorithm 1 details the eFMM learning steps.

One issue worth discussing is the activation level of the brand new rules in the rule base. When a rule is generated, its center, maximum and minimum points are coincident (see (25)), and therefore, the rule has zero initial width. If expression (3) were used, every rule with zero width would have null activation level, regardless of its distance to the current data point. The proposed solution to this bottleneck is composed of the following steps: First, calculate the element wise distance from the zero activation rules center to the new data point:

$$\textbf{For}\{l = 1, 2, ..., Z\}\ \textbf{Do:} \qquad (26)$$
$$dist_l = \sum_{j=1}^{n} |c_{jl} - x_j^k|$$
$$\textbf{EndFor}$$

where Z is the number of zero activation rules currently in the rule base. The second step is testing the zero activation rules for expansion condition (13), one at a time, in an ascending element wise distance order. If one rule satisfies the condition (13), then the test is interrupted, and the rule is updated. If no rule fulfills the expansion condition, a new rule is created.

Algorithm 1 eFMM

Choose M_{min}, δ_0 and ϵ
For $k = 1, 2, 3, ...$ **Do**
  Read input data x^k
  For $i = 1, 2, 3, ..., R$ **Do**
    Compute i-th rule activation level using (3).
    Generate system output with (6).
    Find the rule i with the highest activation level and still not tested for condition (13).
    Test rule B_i for condition (13).
    If Condition (13) holds **Then**
      Expand B_i to include x^k using (7).
      Update B_i counter and center using (8) and (9).
      If $M_i > M_{min}$ **Then**
        Update B_i data dispersion and maximum size using (14) and (15).
      End If
      For $j = 1, 2, ..., n$ **Do**
        If Reduction condition (10) holds for the j-th dimension **Then**
          Reduce hyperbox B_i along dimension j using (11)
        End If
      End For
      Update i-th rule consequent parameters with (16), (17) and (18).
    Else If $b_i > 0$ **Then**
      Find the next hyperbox with highest membership value; go to line 8.
    Else
      Check if there exist rules for which $b_i = 0$.
      If there are, use (26) to compute their element wise distance to the current data point.
      Test the zero activation rules for condition (13), one at a time, on ascending distance order.
    End If
    If no existing hyperbox fulfills condition (13), then create a new hyperbox:
    $R = R + 1,\ \ W_R = V_R = c_R = x^k$
    Compute utility measure and find obsolete rules using (20) and (21).
    Verify for redundant rule pairs using (24).
  End For
End For

4 Computational Results

This section evaluates the performance of the eFMM in two problems: The S&P-500 Index Forecasting, and a nonlinear system identification. Comparisons with alternative evolving and batch modeling approaches are reported considering root mean squared error and the non-dimensional error indexes as performance measures. The expressions for these two measures are:

$$RMSE = \left(\frac{1}{N} \sum_{k=1}^{N} \left(\hat{y}^k - y^k \right)^2 \right)^{\frac{1}{2}}, \quad NDEI = \frac{RMSE}{std(y)} \qquad (27)$$

where N is the size of the test dataset, y^k and $\hat{y}^k$ are the target and the model output, respectively, and $std()$ is the standard deviation function. The number of rules, for fuzzy rule-based methods, or the number of neurons, for neural-based approaches, gives model complexity.

4.1 S&P-500 Index Forecasting

This section analyzes the prediction performance of eFMM using the S&P-500 market index dataset. The Standard & Poor's 500 Index (S&P-500) is an index of 505 stocks issued by 500 large companies with market capitalizations of at least $6.1 billion. It is seen as

a leading indicator of U.S. equities and a reflection of the performance of the large-cap universe. The dataset encompasses 60 year daily index value, which can be found in the Yahoo! Finance website. A total of 14 893 data were acquired from January 3, 1950 to March 12, 2009. This is the same dataset used in [10], so the performances could be properly compared. The eFMM is compared with other algorithms such as Parsimonious Network based on Fuzzy Inference System (PANFIS), Dynamic Evolving Neural-Fuzzy Inference System (DENFIS), evolving Takagi-Sugeno (eTS), simplified eTS (simpl_eTS), Adaptive Network Based on Fuzzy Inference System (ANFIS), Linear Regression (LR), evolving Fuzzy Neural Network (EFuFNN), and Self organizing Fuzzy Associative Machine (SeroFAM). The input and output relationship of the system is expressed as:

$$y^k = f(y^{k-1}, y^{k-2}) \tag{28}$$

In [10], the prediction model used up to five lagged observations of the series to predict the output (i.e. $y^k = f(y^{k-1}, ..., y^{k-5})$). In our experiments, however, we note that only two lagged observations are needed to achieve prediction performance, and additionally, lower execution time and memory usage.

The parameters for eFMM in this simulation were: $M_0 = 3.5(n+1) = 10.5, \epsilon = 0.5, \delta_0 = 0.4$. Fig. 8 shows prediction performance, and Table 1 compares eFMM against other evolving and batch methods using the NDEI. Table 1 also exhibits the number of rules at the end of the simulation, and the total number of rule base parameters stored by the algorithms. Except for eFMM, these are the values reported in [10]. The total number of parameters of eFMM is computed as follows:

$$N_{par} = (3n + (n+n) + (n+1))\, R = (6n+1)R \tag{29}$$

where the first term $3n$ corresponds to the minimum, maximum and center vectors (i.e. V, W, and c), the second term $(n+n)$ corresponds to the maximum size and data dispersion (i.e. δ_i and d_i), and the last term $(n+1)$ is associated with the parameters of the consequent. The terms embraced by the parenthesis are the total number of parameters stored in memory associated with a single rule, and therefore, the total number of parameters is found multiplying it by the total number of rules R.

4.2 Nonlinear System Identification

This section evaluates the eFMM performance to identify a high dimensional nonlinear system. The system to be identified is:

$$y^k = \frac{\sum_{l=1}^{m} y^{k-l}}{1 + \sum_{l=1}^{m} \left(y^{k-l}\right)^2} + u^{k-1} \tag{30}$$

Table 1: S&P-500 Index Forecasting

Model	Total par.	RB par.	Nº of rules	NDEI
PANFIS	144	144	4	0.014
LR	14899	6	-	0.157
DENFIS	126	126	6	0.02
ANFIS	320	15213	32	0.015
EFuNN	1143	1143	114	0.154
SeroFAM	290	290	29	0.027
eTS	75	75	14	0.015
Simpl_eTS	39	39	7	0.045
eFMM	39	39	3	0.016

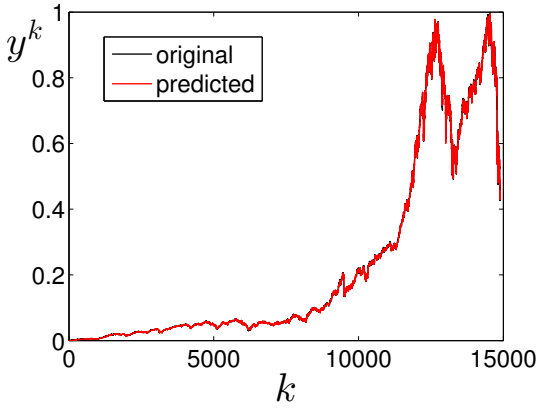


Figure 8: S&P-500

where $u^k = sin(2\pi k/20)$ and $y^j = 0$ for $j = 1, 2, ..., m$ and $m = 10$.

The goal is to predict the current output using the last input, and lagged outputs:

$$\hat{y}^k = f\left(y^{k-1}, y^{k-2}, ..., y^{k-10}, u^{k-1}\right) \tag{31}$$

where $\hat{y}^k$ is the model output.

The first 3000 data points were used for training, and the remaining 300 data points to test the model, keeping the model structure and parameters fixed after training. This is the same test setup used in [5]. The data are not normalized in this experiment, and yet the performance was not affected. The parameters of eFMM in this experiment are: $\gamma = 0.95, M_0 = 3.5(n+1), \epsilon = 0.05$ and $\delta_0 = 0.8$. Table 2 compares the eFMM against alternative evolving methods using the RMSE. Figure 9 illustrates the prediction performance on test data. The number of rules and RMSE values for all alternative evolving methods were taken from [5]. This experiment shows that eFMM considerably outperforms the competing techniques, and suggests that eFMM has higher performance in high dimensional systems than state of the art evolving modeling methods.

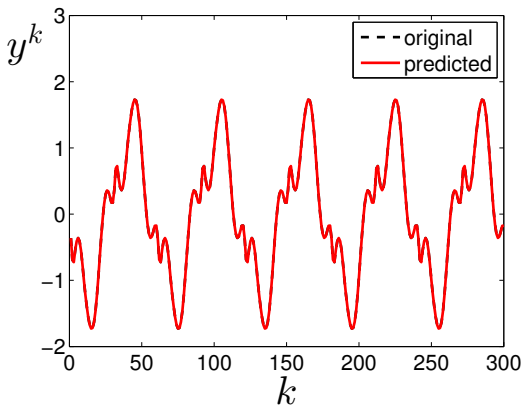


Figure 9: Nonlinear System Identification

Table 2: Nonlinear system identification

Model	Number of Rules	RMSE
xTS	9	3.31 x 10^{-2}
FLEXFIS	15	8.50 x 10^{-3}
eTS	14	7.50 x 10^{-3}
eMG	13	5.00 x 10^{-3}
eFMM	17	5.37 x 10^{-7}

5 Conclusion

The paper has introduced a new evolving functional fuzzy modeling algorithm based on hyperbox min-max granulation and recursive least square parameter estimation. The algorithm process stream data online, may start with an empty rule base, updates the rule base to accommodate the data distribution, features which are important to model nonstationary systems. Computational results with time series and nonlinear system identification problems show that the granular evolving min-max fuzzy modeling algorithm is very effective and competes with evolving algorithms representative of the current state of the art. Future work shall analyse how sensitive the model performance is to changes in the parameters, and also address the issue of how to automatically select the remaining learning parameters.

Acknowledgement

The authors would like to thank the anonymous reviewers for their constructive suggestions, and the Brazilian National Council for Scientific and Technological Development (CNPq) for a fellowship, and grant 305906/2014-3, respectively.

References

[1] P. Angelov, Evolving Takagi-Sugeno fuzzy systems from streaming data (eTS+), in: P. Angelov, D. P. Filev, N. Kasabov (Eds.), Evolving Intelligent Systems: Methodology and Applications, Wiley & IEEE Press, Hoboken, NJ, USA, 2010, pp. 21–50.

[2] K. J. Aström, B. Wittenmark, Computer-Controlled Systems: Theory and Design, 3rd Edition, Prentice Hall, Upper Saddle River, NJ, USA, 1996.

[3] B. Gabrys, A. Bargiela, General fuzzy min-max neural network for clustering and classification, IEEE Transactions on Neural Networks 11 (3) (2000) 769–783.

[4] V. Kreinovich, G. C. Mouzouris, H. T. Nguyen, Fuzzy rule based modeling as a universal approximation tool, in: H. T. Nguyen, M. Sugeno (Eds.), Fuzzy Systems: Modeling and Control, Springer US, Boston, MA, 1998, pp. 135–195.

[5] A. Lemos, W. Caminhas, F. Gomide, Multivariable gaussian evolving fuzzy modeling system, IEEE Transactions on Fuzzy Systems 19 (1) (2011) 91–104.

[6] A. Lemos, W. Caminhas, F. Gomide, Evolving intelligent systems: Methods, algorithms and applications, in: S. Ramanna, L. C. Jain, R. J. Howlett (Eds.), Emerging Paradigms in Machine Learning, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 117–159.

[7] I. Luna, R. Ballini, Online estimation of stochastic volatility for asset returns, IEEE Conference on Computational Intelligence for Financial Engineering & Economics (2012) 1–7.

[8] A. Porto, F. Gomide, Evolving granular fuzzy min-max modeling, in: G. A. Barreto, R. Coelho (Eds.), Fuzzy Information Processing, Springer International Publishing, Cham, 2018, pp. 37–48.

[9] A. Porto, F. Gomide, Evolving granular fuzzy min-max regression, in: P. Melin, O. Castillo, J. Kacprzyk, M. Reformat, W. Melek (Eds.), Fuzzy Logic in Intelligent System Design: Theory and Applications, Springer International Publishing, Cham, 2018, pp. 162–171.

[10] M. Pratama, S. G. Anavatti, P. P. Angelov, E. Lughofer, PANFIS: A novel incremental learning machine, IEEE Transactions on Neural Networks and Learning Systems 25 (1) (2014) 55–68.