Int. J. Fuzzy Syst. (2017) 19(5):1645–1658
DOI 10.1007/s40815-016-0271-0

Top–Down Sparse Fuzzy Regression Modeling from Data with Improved Coverage

Edwin Lughofer[1] · **Stefan Kindermann**[2] · **Mahardhika Pratama**[3] · **Jose de Jesus Rubio**[4]

Received: 13 June 2016 / Revised: 22 August 2016 / Accepted: 21 October 2016 / Published online: 3 November 2016

Abstract We propose a new fuzzy modeling algorithm from data for regression problems. It acts in a top–down manner by allowing the user to specify an upper number of allowed rules in the rule base which is sparsed out with the usage of an iterative constrained numerical optimization procedure. It is based on the combination of the least squares error and the sum of rule weights over all rules to achieve minimal error with lowest possible number of significantly active rules. Two major novel concepts are integrated into the optimization process: the first respects a minimal coverage degree of the sample space in order to approach ϵ-completeness of the rule base (an important interpretability criterion) and the second optimizes the positioning and ranges of influence of the rules, which is done synchronously to the optimization of the rule weights within an intervened, homogeneous procedure. Based on empirical results achieved for several high-dimensional (partially noisy) data sets, it can be shown that our advanced, intervened optimization yields fuzzy systems with a better coverage and a higher degree of ϵ-completeness compared to the fuzzy models achieved by related data-driven fuzzy modeling methods. This is even achieved with a significantly lower or at least equal number of rules and with a similar model error on separate validation data.

Keywords Top–down fuzzy system modeling · Rule base out-sparseing · Constrained numerical optimization problem · Penalty term · Coverage · ϵ-completeness · Intervened · Homogeneous optimization

✉ Edwin Lughofer
edwin.lughofer@jku.at

1 Department of Knowledge-Based Mathematical Systems, Johannes Kepler University Linz, Linz, Austria

2 Industrial Mathematics Institute, Johannes Kepler University Linz, Linz, Austria

3 Latrobe University, Melbourne, Victoria, Australia

4 Seccion de Estudios de Posgrado e Investigacion, Esime Azcapotzalco, Instituto Politecnico Nacional, Av. de las Granjas, Mexico City, Mexico

1 Introduction

1.1 Motivation and State-of-the-Art

Nowadays, the data-driven design of fuzzy systems enjoy a wide attraction in various industrial domains, such as visual inspection systems [27, 29], quality control systems equipped with condition monitoring and predictive maintenance [40], human activity recognition [17], web news mining [18], medical and health care systems [44, 45], dynamic modeling of unmanned aerial vehicles (UAV) [37] or even software and systems engineering [35]. They are often used for automatically quantifying and predicting system states in order to achieve enriched supervision and monitoring of processes in industrial systems [5]. Opposed to classical knowledge-based fuzzy expert systems design (the old-school approach) [41], subjective experiences and impressions are part of the input [1]. In data-driven design, the input is data representing objective observations of the underlying mechanisms in the systems. They characterize various system states, behaviors and operating conditions. Another major advantage of data-driven fuzzy models is the generic applicability of their training algorithm: In fact, data-driven models can be trained solely from data without knowing any underlying physical, chemical, etc., laws. This is achieved within fully automatic runs of learning and evaluation algorithms.

The advantage of data-driven fuzzy systems over other types of soft computing-based model architectures is their joint characteristic of (1) *universal approximation property* [3], that is, to be able to model any implicitly contained nonlinear relationship with an arbitrary degree of accuracy, (2) *linguistic interpretability* [12, 23], which offers some insights into system dependencies and causal relations of variables, (3) *piecewise local approximation* which provides a kind of granulation (and thus partial local interpretation) of the input space and (4) the possibility to model and represent information and data uncertainty in natural, possibilistic way [4].

The most widely used architecture for (high-dimensional) regression/approximation problems are the so-called Takagi–Sugeno fuzzy systems [42], which meet the aforementioned properties. Several techniques for a fully automatic data-driven extraction of these systems have been proposed in the past. One of the most prominent techniques is the *genfis2* technique [6], which applies subtractive clustering [34] for eliciting the samples with highest density. These are then associated with cluster prototypes. It automatically finds the adequate number of clusters to sufficiently explain the data distribution over the input space. The clusters are projected onto the single axes to form the fuzzy sets and the fuzzy rules. Another well-known and widely used technique is the *FMCLUST* method by Babuska [2], which performs Gustafson–Kessel clustering [14] and applies a specific projection scheme to form the fuzzy sets. However, it requires the exact number of rules in advance. For a collection of further clustering techniques for fuzzy rule extraction, see [33]. Another prominent research field for data-driven fuzzy systems design is the so-called *genetic fuzzy systems* [8, 19], which try to optimize the structures and parameters of fuzzy rule bases based on evolutionary algorithms. These iteratively update and improve a whole population of fuzzy systems until convergence in terms of the (average or best) fitness is achieved. They produce solutions which are close to the global optimum by helping individuals (solutions) out of local optimality (based on mutation operations). On the other hand, they are non-deterministic due to implicit random effects, which means that they produce different solutions for different runs.

All the aforementioned methods either require the exact pre-defined number of rules in advance—which is typically hard to guess in advance and time intensive to tune within a validation phase [36]—or they extract as much rules as needed for an appropriate modeling of the distributed regression behavior. There is no control in terms of an upper limit on the number of rules to be extracted. This may become disadvantageous or even inapplicable when seeking for a compact fuzzy rule base due to interpretability reasons. Moreover, it increases the likelihood of over-fitting on new unseen data [39]. Furthermore, in the aforementioned approaches the rules are usually fit hard to the data (clouds) based on some error criteria, without respecting a minimal *sample coverage* (by rules) in order to avoid undefined or nearly undefined input states. Coverage, in mathematical context associated with *ϵ-completeness* criterion [32], is also an important and widely discussed property in the context of interpretability of fuzzy models [12, 23, 46] .

1.2 Our Approach

Our approach suggests a method which performs a top–down rule learning scheme :The operator/user/expert is able to define an acceptable upper number of rules, and the learning algorithm tries to *sparse out as many rules as possible* based on a constrained-based error criterion through the usage of rule weights in [0, 1]; please note that this concept is different to many other forms of regularized learning (such as ridge regression, Lasso, elastic net and spin-offs [15, 47]), which *sparse out as many inputs as possible*. It is based on the *SparseFIS* approach [25], but extends it with new concepts in order to meet the coverage and *ϵ-completeness*. It is thus termed *SparseFIS-Cover*. In particular, it integrates a specific penalty term in the optimization functional in order to punish rule bases with lower coverage. This is done within a (normalized) convex combination with the constrained least squares functional (by the usage of Lagrange multipliers). Thereby, it is possible to put more or less emphasis on the error or on the coverage, respectively.

The second major extension concerns the update of the nonlinear parameters in the rules antecedents, i.e., of the centers and spreads of all fuzzy sets occurring in the antecedents. They are synchronously optimized to the rule weights in an intervened numerical optimization procedure (rather than adjusted by clustering-based heuristics as is done in case of original *SparseFIS*). This leads to a better homogenization of the solutions among the complete parameter space and finally even to rule bases with lower complexities (empirically verified in Sect. 5). For rule consequents learning, it applies a regularized version of weighted least squares to achieve a localized learning scheme for each rule separately with improved properties [22].

The new learning approach is compared to original *SparseFIS* in terms of (1) percentual model errors on separate validation data, (2) average maximal rule membership degree over all samples in the training as well as in the separate validation data set (as a measure for *coverage*), (3) the number of samples meeting the *ϵ-completeness* criterion and (4) the complexity of the final rule base in terms of the number of rules. This is done for various noisy and high-dimensional real-world regression problems. The comparison also includes several other, widely used state-of-the-art (SoA) methods in fuzzy modeling for regression problems and highlights superior performance of the new approach *SparseFIS-Cover* in terms of coverage, ϵ-completeness and complexity of the rule bases while it produces just slightly higher model errors than state-of-the-art (SoA) methods, but lower ones than original *SparseFIS*.

The paper is organized in the following way: Sect. 2 provides a compact summary of the *SparseFIS* algorithm as being published before in [25], and upon which the new method builds. Section 3 denotes the methodological novelty in this paper and addresses the description of the extensions to *SparseFIS* newly proposed in this paper, which includes a pseudo-code of the whole algorithm at the end. Section 4 summarizes the applications and data sets used for empirical comparisons, and Sect. 5 presents the results.

2 SparseFIS: Basic Aspects

SparseFIS was proposed in [25] by Lughofer and Kindermann as a data-driven learning engine for Takagi–Sugeno (TS) fuzzy inference systems [42]. It acts in a *top–down* manner and thus is able to constrain the number of rules extracted from the data. The basic idea is to start with an upper number of (allowed) rules (e.g., pre-defined by experts or parameterized by a classical parameter grid search procedure) and to sparse out as many rules as possible with the help of a numerical optimization procedure.

Therefore, the classical definition of a TS fuzzy system is extended by integrating rule weights $\rho_i \in [0,1]$ which indicate importance levels of rules (0 means that the rule is unimportant and not respected when calculation the inference for new query points, 1 means that the rule should be fully active in the calculation of the inference). This leads to the extended functional definition:

$$\hat{f}(\mathbf{x}) = \hat{y} = \sum_{i=1}^{C} \Psi_i(\mathbf{x}) \cdot l_i(\mathbf{x}) \qquad \Psi_i(\mathbf{x}) = \frac{\rho_i \mu_i(\mathbf{x})}{\sum_{j=1}^{C} \rho_j \mu_j(\mathbf{x})} \tag{1}$$

with l_i the linear consequent functions (partial local linear hyperplane models). $\mu_i(\mathbf{x})$ denotes the activation degree of the i-th rule, so Ψ_i denotes the weighted normalized membership degree of the ith rule. $\mu_i(\mathbf{x})$ is defined by the conjunctive combination of the rule antecedent parts (through the usage of a t-norm [20]): $\mu_i(\mathbf{x}) = \mathrm{T}_{j=1}^{p} \mu_{ij}(x_j)$, where $\mu_{ij}(x_j)$ denotes the membership degree of the jth input coordinate in sample $\mathbf{x}$ to fuzzy set μ_{ij}, which appears in the jth antecedent part in Rule i; p denotes the input dimensionality. Per default, Gaussian fuzzy sets are applied for μ_{ij}, which have two parameters, the center $\mathbf{c}_{ij}$ and the spread $\boldsymbol{\sigma}_{ij}$.

By starting with a rule base in which all rules are fully activated, the goal of the optimization procedure is to drive as many rule weights as possible toward 0 while still achieving a low error in the least squares sense. This is achieved by minimizing the weights or by constraining the number of active rules (rules with weights significantly greater than 0) synchronously to the minimization of the classical least squares error. This leads to a constraint-based nonlinear optimization problem, because the rule weights appear as nonlinear parameters, which, by using the integration of classical Lagrange multipliers, can be re-formulated as a free optimization problem in the following way:

$$J(\boldsymbol{\rho}, \mathbf{c}, \boldsymbol{\sigma}, \mathbf{w}) + \boldsymbol{\alpha} \frac{\boldsymbol{1}}{\boldsymbol{C}} \left(\sum_{i=1}^{C} |\boldsymbol{\rho}_i| - \boldsymbol{K} \right) = \min_{\rho} ! \tag{2}$$

with J the conventional least squares problem (normalized by the number of samples N) and C the number of initially given rules. α plays the role of the Lagrange multiplier and K the role of the constraint on the sum of the rule weights. The parameter α is in one-to-one correspondence with the constraint K, so instead of choosing K someone can as well choose α as a parameter and consider K as implicitly defined by α. This is in accordance with the so-called compressed sensing idea [7].

The problem in (2) corresponds to a free optimization problem, where the main challenge is how to handle the second term appropriately, as the l^1-norm $\sum_{i=1}^{C} |\rho_i|$ is not differentiable. This abandons the application of any classical fast optimization methods [10, 11]. However, according to the derivations by Daubechies et al. [9], an iterative projected version of the so-called *steepest descent* method can solve the functional in (2), where a threshold operator T_α is applied on the rule weight update in each gradient descent iteration step. This leads to an iterative update formula for rule weight optimization from Step m to $m+1$ in the following form (here for the ith rule):

$$\rho_i^{m+1} = T_\alpha \left(\rho_i^m - \tau \sum_{n=1}^{C} \sum_{k=1}^{N} e_k l_n(x_k) \frac{\partial \Psi_n(x_k)}{\partial \rho_i} \right) \tag{3}$$

with τ a small constant defining the step size per iteration (usual setting $\tau = 0.5$), $e_k = y_k - \hat{y}_k$ the error between measured and predicted output in sample x_k and T_α the soft-thresholding operator. It is defined by:

$$T_\alpha = \max\left(\rho_i^{m+1}, \alpha\right) - \alpha \tag{4}$$

with α set to a small constant; according to [25], its default setting is $\alpha = 0.1$, but based on our past experience in various applications (see e.g., [40]), it turned out that it may be beneficial to tune it during the training phase through the cross-validation procedure. Higher values of α lead to a faster down-weighing of rule weights and thus may induce the risk of sparseing out the rule base too much. $\frac{\partial \Psi_n(x_k)}{\partial \rho_i}$ denotes the partial derivative of the normalized rule degree of Rule n subject to the rule weight of Rule i and is given by:

$$\begin{aligned}\frac{\partial \Psi_n}{\partial \rho_i}(x) &= \frac{\mu_n}{\sum_{j=1}^{C} \rho_j \mu_j(x)} \left(\delta_{n,i} - \frac{\rho_n \mu_i}{\sum_{j=1}^{C} \rho_j \mu_j(x)} \right) \delta_{n,i} \\ &= \begin{cases} 1 & \text{if } n = i \\ 0 & \text{if } n \neq i \end{cases}\end{aligned} \tag{5}$$

In the original *SparseFIS* method, there are two shortcomings, which we will try to overcome with the proposed extension described in the subsequent section:

- The rule centers $(\mathbf{c}_1, \ldots, \mathbf{c}_C)$ and spreads $(\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_C)$ are initially (roughly) estimated by vector quantization [13], but then kept fixed during the whole lifetime of the rule weight optimization process. This may lead to non-optimal positioning of the rules (centers) and especially to non-optimal ranges of influences whenever the corresponding rule weights are changing—some rules may be sparsed out, which, however, would require the extension of the ranges of influence of adjacent rules in order to assure sufficient coverage of the nearby lying samples.
- The algorithm tries to sparse out as much rules as possible during the optimization process while approaching a reasonable (low) model error. It thus takes care about a good tradeoff between compactness of the rule base and model error. However, it does not take into account how well the input space is still covered significantly active rules.

Both bottlenecks are obviously expected to affect the *coverage* property of the fuzzy rule base, which is (i) one essential property for (enhanced) interpretability aspects of fuzzy systems, see [12, 23, 46] and, often even more importantly, (ii) a necessary prerequisite for a well-posed and good predictive performance on new unseen data [22, 38]—a purely covered input space increases the likelihood of undefined or badly defined input states in the fuzzy partitions.

3 SparseFIS-Cover: The Novel Proposed Extension to SparseFIS

In order to overcome this unpleasant situation, we introduce a penalty term in the extended least squares functional as defined in (2). This penalty term should reflect the degree of coverage in terms of the famous ϵ-completeness concept [32]. The latter is used in fuzzy systems design for assuring a minimal coverage of the input space. In a data-driven learning context, the maximal membership degree of each of the available training samples to all rules is a reliable indicator of the coverage and ϵ-completeness [23]. So, each sample belongs to at least one rule with a minimal membership degree of ϵ.

In order to mathematically represent this demand, the idea is to relax the strict fitting criterion given by the least squares functional with an additional term which punishes those rule partitions which result in a lower degree of sample coverage. Therefore, we introduce the following term:

$$Pen = \sum_{k=1}^{N} \frac{\prod_{i=1}^{C} \max(0, \epsilon - \rho_i \mu_i(\mathbf{x}_k))}{\epsilon^C} \tag{6}$$

If $\mu_i(\mathbf{x}_k)$ is greater than ϵ for a specific sample $\mathbf{x}_k$ and the corresponding (the ith) rule is significantly active (i.e., its weight $\rho_i > > \epsilon$), a sufficient coverage in all antecedent parts is achieved, and thus the penalty term becomes 0 for $\mathbf{x}_k$, as $\max(0, \epsilon - \rho_i \mu_i(\mathbf{x}_k)) = 0$ which is multiplied with the same term obtained from all other rules. This is not the case when ρ_i approaches 0, as then the rule is not really active, and thus it does not contribute to the model prediction inference in (1). This means that the sample is not really covered by Rule i. If $\mu_i(\mathbf{x}_k)$ is close to 0 for all rules, the sample is not sufficiently covered and therefore (6) approaches 1 due to the normalization term ϵ^C, yielding a high impact of (6) in the joint optimization problem (7). In case of using Gaussian fuzzy sets, $\epsilon = 0.135$ is the most convenient choice, see [23, 38], and therefore also used in all our empirical tests.

We integrate the term in (6) in the Lagrange least squares functional as defined in (2) in a way to achieve a convex combination between least squares error fit and coverage degree penalty. This gives us some degrees of freedom for putting more or less emphasis on one or the other criterion; thus, the optimization problem becomes:

$$\begin{aligned} J_{ext} = {} & \lambda \frac{1}{N * \mathrm{range}^2(y)} \left(\sum_{k=1}^{N} (y_k - \sum_{i=1}^{C} l_i(\mathbf{x}_k) \Psi_i(\Phi_{\mathrm{nonlin}})(\mathbf{x}_k))^2 \right) \\ & + \lambda \alpha \frac{1}{C} \left(\sum_{i=1}^{C} |\rho_i| - K \right) \\ & + (1 - \lambda) \frac{1}{N} \sum_{k=1}^{N} \left(\frac{\prod_{i=1}^{C} \max(0, \epsilon - \rho_i \mu_i(\mathbf{x}_k))}{\epsilon^C} \right) = \min_{\Phi_{\mathrm{nonlin}}} ! \end{aligned} \tag{7}$$

with λ a user-friendly parameter as it balances the importance level of precision versus completeness. It is thus able to tradeoff accuracy and interpretability in a continuous manner (default value is 0.5 inducing equal importance). $\mathrm{range}^2(y)$ denotes the squared range of the target variable.

This range normalization is indispensable in order to normalize the first term to [0, 1] and to achieve comparability with the other two terms (which also range in [0, 1])—a non-normalization would weight the first term differently for different learning problems and data sets with different target ranges, which is undesirable. Φ_{nonlin} denotes the set of nonlinear parameters, which in our case is now composed of $\{\boldsymbol{\rho}, \mathbf{c}, \boldsymbol{\sigma}\}$, as we also aim to optimize the centers and spreads over all rules, synchronously to the rule weights. This should assure an adaptation of rule spreads (and centers) toward optimality also in case when the rule weights are significantly changed during optimization.

The optimization function in (7) denotes a convex combination of least squares functional and the penalty term, which makes enhanced numerical optimization techniques such as Levenberg–Marquardt algorithm inapplicable (other terms vanish in the Jacobian matrix) [30]. Furthermore, the second derivatives for building up Hessian matrices are pretty complicated to establish and very time intensive to compute. Therefore, and due to the good experience we had in conventional *SparseFIS* with the projected gradient descent iteration in terms of converging rule weight sequences, we also apply the gradient descent algorithm for the enhanced functional proposed in this paper. Therefore, we add the gradient of the penalty term with respect to the rule weight ρ_i in the update equation (3), which (for the ith rule) leads to the following iteration step (from Step m to $m+1$).

$$\rho_i^{m+1} = T_\alpha\left(\rho_i^m - \tau\left(\lambda \sum_{n=1}^{C}\sum_{k=1}^{N} e_k l_n(x_k)\frac{\partial \Psi_n(x_k)}{\partial \rho_i} + \left(1-\lambda\right)\frac{\partial Pun}{\partial \rho_i}\right)\right) \tag{8}$$

where the partial derivative $\frac{\partial Pun}{\partial \rho_i}$ is given by (proof left to the reader):

$$\frac{\partial Pun}{\partial \rho_i} = \begin{cases} \sum_{k=1}^{N}\frac{1}{\epsilon^C}\mu_i(\mathbf{x}_k)\prod_{j=1,j\neq i}^{C}\max(0,\epsilon-\rho_j\mu_j(\mathbf{x}_k)) & \epsilon-\rho_i\mu_i(\mathbf{x}_k)>0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Furthermore, as mentioned above, we optimize the centers and spreads of all rules synchronously to the weights, in order to adjust them correctly toward optimality with respect to the current fuzzy model description (including the current weights). This is also done by applying a gradient descent step right after the rule weight update, but without using the projection operator T_α, as down-weighting centers and spreads toward 0 would not have any useful meaning. The partial derivatives of $\Psi_n, n=1,\ldots,C$ with respect to the center and spread coordinates of the ith rule, $c_{i,l}, \sigma_{i,l}$ with $l=1,\ldots,p$, are given by: If $n=i$ then we have

$$\frac{\partial \Psi_n(\mathbf{x})}{\partial c_{i,l}} = \frac{(1-\Psi_n)(-2\rho_n\mu_n)}{\sum_{j=1}^{C}\rho_j\mu_j(\mathbf{x})}\left(\frac{1}{2\sigma_{i,l}^2}(c_{i,l}-x_l)\right) \tag{10}$$

If $n \neq i$ then we get

$$\frac{\partial \Psi_n(\mathbf{x})}{\partial c_{i,l}} = \frac{(-\Psi_n)(-2\rho_i\mu_i)}{\sum_{j=1}^{C}\rho_j\mu_j(\mathbf{x})}\left(\frac{1}{2\sigma_{i,l}^2}(c_{i,l}-x_l)\right) \tag{11}$$

For σ we get similar equations, i.e., if $n=i$, then we have

$$\frac{\partial \Psi_n(\mathbf{x})}{\partial \sigma_{i,l}} = \frac{(1-\Psi_n)(\rho_n\mu_n)}{\sum_{j=1}^{C}\rho_j\mu_j(\mathbf{x})}\left(\frac{1}{\sigma_{i,l}^3}(c_{i,l}-x_l)^2\right) \tag{12}$$

If $n \neq i$ then we get

$$\frac{\partial \Psi_n(\mathbf{x})}{\partial \sigma_{i,l}} = \frac{(-\Psi_n)(\rho_i\mu_i)}{\sum_{j=1}^{C}\rho_j\mu_j(\mathbf{x})}\left(\frac{1}{\sigma_{i,l}^3}(c_{i,l}-x_l)^2\right). \tag{13}$$

The partial derivatives for the centers and spreads ($c_{i,l}$ the lth center coordinate of the ith rule) with respect to the penalty term are given by:

$$\frac{\partial Pun}{\partial c_{i,l}} = \begin{cases} \sum_{k=1}^{N}\frac{1}{\epsilon^C}\mu_i(\mathbf{x}_k)\frac{-(c_{i,l}-x_l)}{\sigma_{i,l}^2}\prod_{j=1,j\neq i}^{C}\max(0,\epsilon-\rho_j\mu_j(\mathbf{x}_k)) & \epsilon-\rho_i\mu_i(\mathbf{x}_k)>0 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$\frac{\partial Pun}{\partial \sigma_{i,l}} = \begin{cases} \sum_{k=1}^{N}\frac{1}{\epsilon^C}\mu_i(\mathbf{x}_k)\frac{(c_{i,l}-x_l)^2}{\sigma_{i,l}^3}\prod_{j=1,j\neq i}^{C}\max(0,\epsilon-\rho_j\mu_j(\mathbf{x}_k)) & \epsilon-\rho_i\mu_i(\mathbf{x}_k)>0 \\ 0 & \textit{otherwise} \end{cases} \tag{15}$$

Hence, the gradient descent step for the centers and spread is:

$$c_{i,l}^{m+1} = c_{i,l}^m - \tau\left(\lambda\sum_{k=1}^{N}e_k\sum_{n=1}^{C}l_n(\mathbf{x}_k)\frac{\partial \Psi_n(\mathbf{x}_k)}{\partial c_{i,l}} + (1-\lambda)\frac{\partial Pun}{\partial c_{i,l}}\right) \tag{16}$$

and

$$\sigma_{i,l}^{m+1} = \sigma_{i,l}^m - \tau\left(\lambda\sum_{k=1}^{N}e_k\sum_{n=1}^{C}l_n(\mathbf{x}_k)\frac{\partial \Psi_n(\mathbf{x}_k)}{\partial \sigma_{i,l}} + (1-\lambda)\frac{\partial Pun}{\partial \sigma_{i,l}}\right) \tag{17}$$

where τ (learning step) is a fixed number which is rather small in order to avoid fluctuations during the optimization process (based on our past experience with original SparseFIS method, we used 0.5 as default in all our experiments).

The whole algorithm of the new method *SparseFIS-Cover* is demonstrated in the pseudo-code provided in Algorithm 1.

Algorithm 1 **Sparse Fuzzy Inference Systems Training for Improved Coverage (SparseFIS-Cover)**

```
1. Input: the upper number of allowed rules $C$, regularization parameter $\alpha$, $\tau$ (default 0.5), convex combination parameter $\lambda$ (default 0.5), $m = 0$
2. Estimate the initial position of the rule centers $\mathbf{c}_1^0, \ldots, \mathbf{c}_C^0$ through clustering (e.g., vector quantization as used in original SparseFIS) or through random assignment in the $p$ dimensional input space.
3. Estimate the initial spreads of the rules $\boldsymbol{\sigma}_1^0, \ldots, \boldsymbol{\sigma}_C^0$ through the standard deviation of data samples assigned to each rule (based on closest distance between samples and rule centers).
4. Initialization of $\mathbf{w}^0$ is done by using local learning with (18) on the initial rules (see Step 5 below); $\boldsymbol{\rho}^0$ is set to a vector of ones $(1, \ldots, 1) \in \mathbb{R}^C$ (all rules contribute equally at the beginning).
5. $\mathbf{w}^m, \boldsymbol{\rho}^m, \mathbf{c}^m, \boldsymbol{\sigma}^m$ are given in iteration $m$.
6. Calculate $\frac{\partial \Psi_n}{\partial \rho_i}$ based on (5).
7. Calculate $\frac{\partial Pun}{\partial \rho_i}$ based on (9).
8. Update the rule weights $i = 1, \ldots, C$ component-wise separately by applying (8) $\rightarrow \rho_i^{m+1}, i = 1, \ldots, C$.
9. For each dimension $l = 1, \ldots, p$, calculate $\frac{\partial Pun}{\partial c_{i,l}}$ based on (14) and calculate $\frac{\partial \Psi_n(\mathbf{x})}{\partial c_{i,l}}$ based on (10) and (11).
10. For each dimension $l = 1, \ldots, p$, calculate $\frac{\partial Pun}{\partial \sigma_{i,l}}$ based on (15) and calculate $\frac{\partial \Psi_n(\mathbf{x})}{\partial \sigma_{i,l}}$ based on (12) and (13).
11. Update all center coordinates $c_{i,l}$ component-wise separately by applying (16) $\rightarrow c_{i,l}^{m+1}$.
12. Update all spreads $\sigma_{i,l}$ component-wise separately by applying (17) $\rightarrow \sigma_{i,l}^{(m+1)}$.
13. Calculate $\Psi(\mathbf{c}^{m+1}, \boldsymbol{\sigma}^{m+1}, \boldsymbol{\rho}^{m+1})$ for the new $\rho_i^{m+1}, i = 1, \ldots, C$ and generate the regression matrix $\sqrt{Q_i} R_i, i = 1, \ldots, C$ for each rule separately with $R_i$ containing the original variables and a column of ones for the intercept, and the weighting matrix $Q_i$ defined by the diagonal matrix $Q_i = diag(\Psi_i(\mathbf{x}(1, \ldots, N)))$.
14. Perform the weighted least squares approach for estimating the linear consequent parameters for all rules $i = 1, \ldots, C$ separately. Its analytical solution in closed form is [22] (here for the $i$th rule):
        $$\hat{\mathbf{w}}_{\mathbf{i}} = (R_i^T Q_i R_i)^{-1} R_i^T Q_i \mathbf{y} \qquad (18)$$
    Here, we also apply Tikhonov regularization [43] if required, i.e., when the condition of the matrix $R_i^T Q_i R_i$ is very high. The regularization parameter $\alpha$ is set by applying an heuristic approach [25]:
    (a) Compute the condition of the matrix $R_i^T Q_i R_i$ by $cond(R_i^T Q_i R_i) = \frac{\lambda_{\max}}{\lambda_{\min}}$ with $\lambda_{\max}$ the largest and $\lambda_{min}$ the smallest eigenvalue.
    (b) If $cond(R_i^T Q_i R_i) > threshold$, the matrix is badly conditioned, so set $\alpha = \frac{2\lambda_{\max}}{threshold}$, with threshold a large value, typically $10^{15}$.
    (c) Else, set $\alpha = 0$
    (d) Apply (18) with $(R_i^T Q_i R_i + \alpha I)$ instead of $R_i^T Q_i R_i$.
15. If the difference of the rule weights between two iterations, i.e., $\|\rho^{m+1} - \rho^m\|$, is low enough, or a predefined maximal number of iterations is reached, then goto next step, otherwise goto Step 2.
16. Discard all rules with weights smaller than $\delta = 0.01$ from the rule base.
17. Perform Steps 9 to 15 again for the remaining rules $\rightarrow$ fine tuning step of rule centers and spreads after rule reduction.
18. Output: sparse fuzzy system with improved coverage.
```

Please note that the fine tuning step for the rule centers and spreads in Step 17 is beneficial, as the deletion of some rules (due to low weights) in Step 16 changed the structure of the rule base, whereas centers and spreads have been optimized before for the full rule base.

Steps 13 and 14 are for the purpose to optimize the linear consequent parameters synchronously to the nonlinear parameters; as neither the coverage problematic nor the rule sparsity aspect concerns the consequent parameters, they can be optimized based on the conventional least squares functional J. This is done in a local learning approach (per rule separately), which has several advantages over global learning as deeply analyzed in [22] (Chapter 2) and also in [23], such as faster computation time, higher stability and better interpretability. This leads to the closed loop formula in (18) (as the problem is parabolic for linear parameters). Tikhonov regularization as conducted in Step 14 d.) becomes indispensable when the matrix in (18) has low rank.

There are two sensitive parameters to tune from outside: the number of maximally allowed rules C and the regularization parameter α in the soft-shrinkage operator T; both can be varied within a grid search procedure.

Table 1 Summary of the characteristics of the data sets used

	# Training samples	Test samples	# Input variables	Source	Noise level
Auto-MPG	266	126	7	UCI	Low
NOx static	667	159	181	Engine test bench	Medium to high
NOx dynamic	16,936	5386	41	Engine test bench	Medium to high
Resistance value	6513	6662	132	Rolling mills	Medium to high
Premise prices	2068	653	5	Residence data base	None to Low
Jester	16,654	8329	9	Berkeley, rating of jokes	None

4 Experimental Setup

4.1 Data Sets Characteristics

We tested our new algorithm *SparseFIS-Cover* and performed a comparison with original *SparseFIS* as well as with other related fuzzy regression modeling methods, see Sect. 4.2. The comparison is based on the following high-dimensional real-world data sets:

– The Auto-MPG data set taken from the UCI repository[1]: it concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of three multi-valued discrete and five continuous attributes.
– Stationary data from engine test bench: it includes various measurement channels together with their time delay (up to 10) resulting in a high-dimensional problem; the task was to build a high-performance quantification model for the emission channel NOx for the purpose to save expenses for the sensor.
– Dynamic data from engine test bench: opposed to the previous data set, it contains dynamic data permanently recorded during different simulation cycles of the engine including the engine speed and torque profiles during an MVEG cycle, a sportive driving profile in a mountain road and two different synthetic profiles. Forty-two sensors at the engine test bench have been installed which measure various important engine characteristics such a pressure of the oil, various temperatures and emission gases. The final aim was to establish a k-step-ahead prediction model for NOx emission in order to save expenses [26] and also to perform early recognition of potential faults (pipe leakage, sensor overheating, interface defects, etc.) [40].
– Data from a cold rolling process at rolling mills: the task was to identify a prediction model for the resistance value of steel plates. This parameter is the most responsible one for a smooth and high-qualitative run-through of the process and thus is permanently supervised manually. An analytical model physically motivated by material laws was indeed already available [16] where some parameters were estimated through linear regression. However, its performance did not meet the company requirements, thus should be improved by nonlinear fuzzy models established from measurement data recorded at a multi-sensor network installed along the mill.
– Data set comprising the prices of residences in a Polish city: it was drawn from an unrefined data set referring to residential premises transactions accomplished in one Polish big city with the population of 640,000 within the years between 1998 and 2003. Nine features were pointed out as main drivers of premises prices: usable area of premises, age of a building, number of rooms in a flat, floor on which a flat is located, number of storeys in a building, geodetic coordinates Xc and Yc of a building, distance from the city center and distance from the nearest shopping center—for more details see also [28].
– Jester data set from the Berkeley data repository[2]: it contains the ratings of jokes by users on a continuous scale in the range $[-10, 10]$. The goal is to recognize and to predict the rating scores of new users. We used the dense subset of jokes $\{5, 7, 8, 13, 15, 16, 17, 18, 19, 20\}$ as partial problem: the rating of the last joke (#20) should be predicted based on the ratings of the other jokes.

Table 1 provides an overview about the properties of each data set in terms of the number of training samples, the number of separate test samples, the input dimensionality, the source and the expected noise level contained in the data.

4.2 Evaluation and Parametrization

According to the successful previous application of *SparseFIS* on several data sets in Table 1 (e.g., on Auto-MPG in [25] or on the residential premise price problem in [28]), one focus was sharpened to the comparison between original *SparseFIS* and its extension *SparseFIS-Cover*

[1] http://archive.ics.uci.edu/ml/.

[2] http://eigentaste.berkeley.edu/dataset/.

proposed in this paper, especially in terms of coverage and model complexity. *Moreover, we also examined the performance of other well-known and widely applied fuzzy modeling variants in order to use them as strong benchmarks against SparseFIS-Cover, such as FLEXFIS, short for Flexible Fuzzy Inference Systems* [21], *Gen-Smart-EFS, short for Generalized Smart Evolving Fuzzy Systems* [24], *LoLiMoT, short for Local Linear Model Trees* [31], *a modified version of genfis2* [6] (*as available in MATLAB's fuzzy logic toolbox*), *termed as genfis2 loc, FMCLUST, short for Fuzzy Modeling with Clustering* [2], *as available in a MATLAB toolbox.*[3] A short description of each method is given below:

- *FLEXFIS*, short for *Flexible Fuzzy Inference Systems* [21], employs classical TS fuzzy systems with axis-parallel rules and performs a regularization in consequent learning based on ridge regression (using an own developed fast heuristics for setting the regularization parameter). It evolves rules on demand in a streaming context based on a pre-defined vigilance parameter (the only real sensitive parameter).
- *Gen-Smart-EFS*, short for *Generalized Smart Evolving Fuzzy Systems* [24] an extension of FLEXFIS with the usage of arbitrarily rotated rules in the multi-dimensional space (achieved through multivariate Gaussians); it evolves rules on demand in a streaming context based on a pre-defined Mahalanobis distance-based tolerance range (deduced from the statistical concept of *prediction interval*). It comes with several extended functionalities regarding online rule merging and dynamic dimension reduction.
- *LoLiMoT*, short for *Local Linear Model Trees* [31], which is one of the most widely used fuzzy systems training methods in today's industrial processes. It performs successive splits along each axis to produce a top–down hierarchy of more and more fine-granulated fuzzy models in a tree-structured manner. It stops the splitting process whenever the error on separate validation data converges.
- A modified version of *genfis2* [6] (as available in MATLAB's fuzzy logic toolbox), termed as *genfis2 loc*, in order to substitute local learning of consequent parameters with global learning (thus, improving its stability, etc., as analyzed in [23]). For learning the rule structure and antecedent parts of the rules, it applies *subtractive clustering*, which searches for highest density points which are then assigned to cluster (=rule) centers. Depending on the parametrization of the maximal range of influence of each rule, more or less density points are assigned to rule centers.
- *FMCLUST*, short for *Fuzzy Modeling with Clustering* [2], is available in a MATLAB toolbox[4] and also one of the most widely used fuzzy training methods in today's industrial and control processes. It employs a Gustafson–Kessel clustering [14] to extract the rules, performs an enhanced projection to the input axes to obtain the shape of the fuzzy sets and finally estimates the rule consequent functions with a regularized least squares approach.

In order to achieve a fair comparison among all fuzzy modeling methods, cross-validation (CV) has been performed on the training data sets with the same fold partitioning for all methods. CV is run multiple times over a parameter grid defined for the most sensitive learning parameter(s) for each method. In particular, the following grids have been used for the various methods:

- *FLEXFIS* The vigilance parameter is varied from 0.1 to 0.9 in steps of 0.1.
- *Gen-Smart-EFS* The tolerance radius is varied from 0.5 to 4.0 in steps of 0.35.
- *LoLiMoT* The upper number of allowed splits is varied from 1 to 15 in steps of 1.
- Modified version of *genfis2* Range of influence of a cluster (percentage of feature range) from 0.1 to 0.9 in steps of 0.1.
- *FMCLUST* The number of clusters = rules is varied from 1 to 15 in steps of 1.
- *SparseFIS* and *SparseFIS-Cover* The initial number (upper allowed number) of rules is varied from 30 to 90 in steps of 10; α set to 0.3 for all problems.

Additionally, a filter variable selection method was employed in order to rank the variables according to their importance for the current learning problem at hand in advance. It is a modified nonlinear version of (classical) forward selection ready-made for fuzzy modeling as explained in more detail in [5]. The achieved rankings are used for successively adding variables and performing a full CV-based evaluation cycle for each subset. In this way, an additional dimension for the number of used inputs is added in the parameter grids defined above (a second one in all cases). It ranges from 1 to 20 (maximal dimensionality) in steps of 1 (so a value of five means to use the first five variables in the ranked list) and is applied in the same way for all modeling methods.

The minimal cross-validation error achieved over all parameter grid points in terms of the normalized mean absolute error (CV_NMAE) is used for selecting the optimal learning parameter configuration for each method, i.e., $argmin_{l,m}(\text{CV\_MAE}_{l,m})$ with l the first dimension of the grid (input dimensionality in all cases) and m the

[3] http://www.dcsc.tudelft.nl/~rbabuska/fmid-v40.zip.

[4] http://www.dcsc.tudelft.nl/~rbabuska/fmid-v40.zip.

second one (method dependent, see itemization above). This error is defined by

$$\text{CV\_NMAE} = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{N/K}\sum_{k=(i-1)*(N/K)+1}^{i*(N/K)}|y_k - \hat{y}_k|, \qquad (19)$$

with N the number of training samples in total and K the number of folds. The optimal configuration is finally used (1) to train a full model on the whole training data set and (2) to elicit the performance measures on the separate test data set: (a) the model error in terms of percentual mean absolute error (MAE):

$$\%(\text{MAE}) = \frac{1}{N}\sum_{k=1}^{N}\frac{|y_k - \hat{y}_k|}{\text{range}(y)}, \qquad (20)$$

with $\text{range}(y) = \max_{k=1,\ldots,N}(y_k) - \min_{k=1,\ldots,N}(y_k)$; (b) the average coverage degree of the test samples measured by

$$\text{cover} = \frac{1}{N}\sum_{k=1}^{N}\max_{i=1,\ldots,C}(\mu_i(\mathbf{x}_k)) \qquad (21)$$

and (c) the percentage of test samples fulfilling the ϵ-completeness criterion, i.e.,

$$\%\_eps = 100 * \left(\frac{1}{N}|\{\mathbf{x}_k | k = 1,\ldots,N \wedge max_{i=1,\ldots,C}\mu_i(\mathbf{x}_k) > \epsilon\}|\right), \qquad (22)$$

with $epsilon = 0.135$ the most convenient default value [23]. Furthermore, we will study the final model complexity in terms of the number of rules for examining the compactness and the likelihood of over-fitting of the rule base—a lower number with similar model error is always preferable.

5 Results

Table 2 provides an overview about the results achieved by *SparseFIS* (abbreviated with *SPF*), *SparseFIS-Cover* (abbreviated with *SPF-C*) and by the best of the state-of-the-art fuzzy modeling methods for each data set in terms of the percentual error (abbreviated with *(Best) SoA*)—for transparency reasons, we neglect the results of the other fuzzy modeling methods which produce worse results than the best; by comparing *SPC* and *SPF-C* with the *best SoA* method, we already see how much improvement over state-of-the-art fuzzy methods can be ideally achieved, which from our point of view leads to a sufficient comparison: if there is an improvement over the *best SoA* method, then automatically there is an improvement over all other *SoA* methods; if there is no improvement over the *best SoA* method, then someone could always do better when choosing this method. At first glance, it can be clearly recognized that SparseFIS-Cover produces lower or equal percentual errors on separate test data than SparseFIS, except for the resistance data set, while it is able to achieve a better or equal coverage and ϵ-completeness ratio in all cases. Interestingly, this can be achieved with a much lower number of rules (except for Jester and Premise prices) as can be seen from Table 3. The interpretation of this aspect is that the optimization of the rule spreads/centers synchronously to the rule weights (which is not done in conventional SparseFIS) leads to a more homogeneous partitioning in terms of wider rules which better cover the feature range. A strong point is that this is achieved by not

Table 3 Achieved fuzzy model complexities in terms of the number of rules

Data set	SPF	SPF-C	Best SoA
Auto-MPG	19	7	15
NOx static	30	14	17
NOx dynamic	30	5	45
Resistance	29	2	5
Premise	2	2	7
Jester	3	4	4

SPF SparseFIS, *SPF-C* SparseFIS-Cover, *Best SoA* Best State-of-the-Art fuzzy modeling method

Table 2 Summary of results for all data sets. The cell values of the first three columns reflect the percentual MAEs, of the next three columns they correspond to the coverage measure value *cover* and in the last three columns they represent the percentage of samples meeting \epsilon −*completeness*\ % \underscore eps -

Data set	SPF%	SPF-C%	Best SoA%	SPF cover	SPF-C cover	SoA cover	SPF %_eps	SPF-C %_eps	SoA %_eps
Auto-MPG	9.99	9.66	9.28	0.04/0.51	0.09/0.51	0.14/0.65	7.14/93.6	25.4/93.6	29.4/96.6
NOx static	3.76	3.50	3.71	0.19/0.23	0.23/0.26	0.04/0.04	61.6/71.4	61.0/72.2	5.66/7.80
NOx dynamic	6.00	5.80	5.47	0.45/0.53	0.52/0.58	0.47/0.63	90.0/96.5	91.7/97.4	85.8/99.2
Resistance	3.03	3.77	2.94	0.30/0.38	0.51/0.60	0.45/0.58	68.2/84.4	89.0/94.1	84.1/93.8
Premise	4.81	4.81	4.76	0.37/0.38	0.37/0.38	0.34/0.35	86.7/88.3	86.7/88.3	88.5/88.8
Jester	3.34	3.34	3.32	0.39/0.41	0.51/0.52	0.13/0.13	90.8/91.7	99.9/99.9	80.2/84.4

SPF SparseFIS, *SPF-C* SparseFIS-Cover, *Best SoA* Best State-of-the-Art fuzzy modeling method (in terms of the error) loosing any model accuracy, and in most cases even accuracy on separate test data is gained. This is in accordance with the expected over-fitting effect, which is tendentially more severe in case of a higher number of rules, especially their coverage of the input sample space is low. In this sense, SparseFIS-Cover clearly produces more reliable and further useable fuzzy models than SparseFIS, except in case of premise prices where both ended up in exactly the same fuzzy models. For this data, there was obviously no real need for a penalty function during optimization to improve sample coverage.

Regarding the comparison with other state-of-the-art (SoA) fuzzy modeling methods, the situation is somewhat similar, especially when comparing the model complexities from Table 3—clearly higher for the best SoA method (elicited for each data set separately) except for Jester data set. So, obviously the numerically funded sparseing out procedure in combination with the optimization of the centers/spreads does the job pretty well to form homogenously optimized rule partitions with low generalization errors. Indeed, in all cases, the error of the best SoA method is slightly (but no significantly!) higher than the

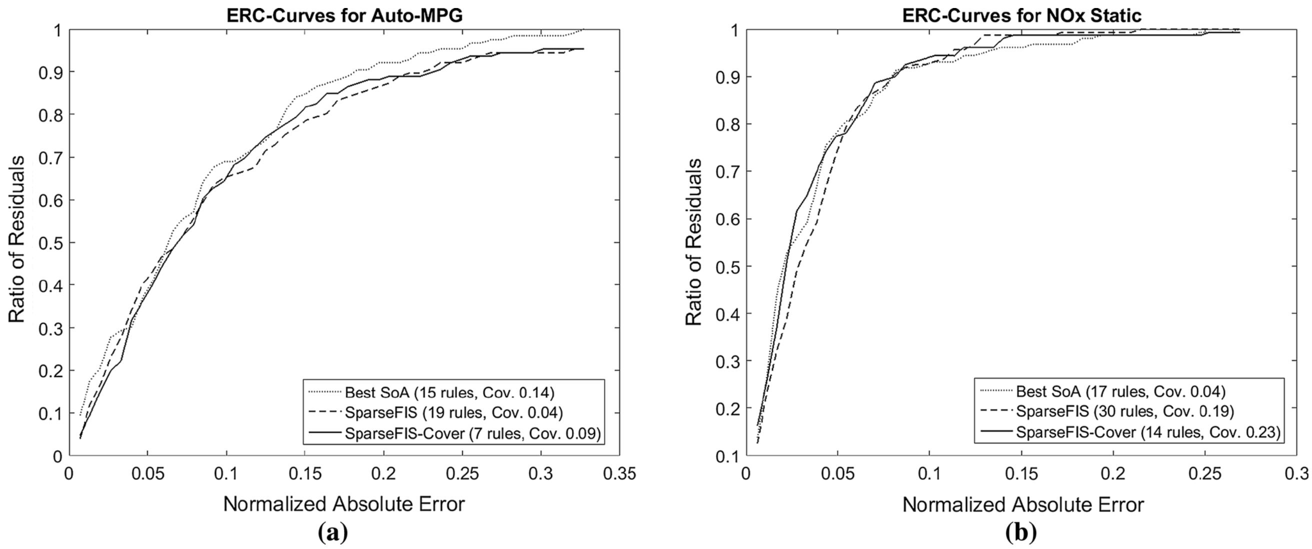

Fig. 1 **a** ERC curves for the Auto-MPG data set as achieved by original SparseFIS (*dashed line*), the best related SoA method (*dotted line*) and SparseFIS-Cover (*solid line*), the legend explains the number of rules and the coverage degree, **b** the same for static NOx modeling

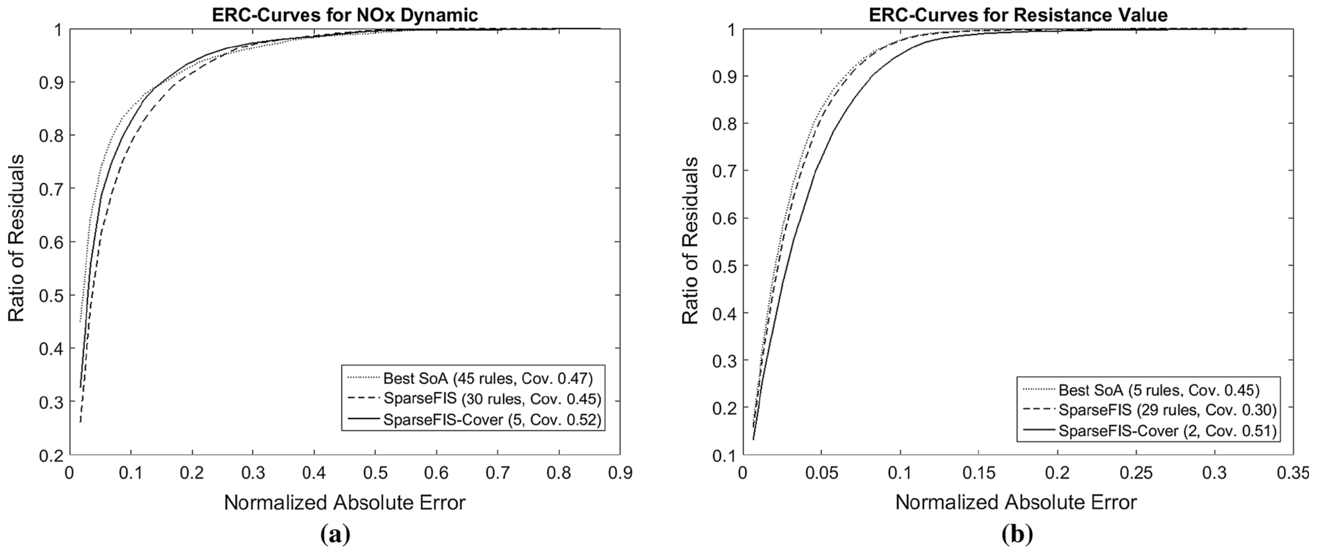

Fig. 2 **a** ERC curves for the dynamic NOx data set as achieved by original SparseFIS (*dashed line*), the best related SoA method (*dotted line*) and SparseFIS-Cover (*solid line*), the legend explains the number of rules and the coverage degree, **b** the same for the resistance value modeling

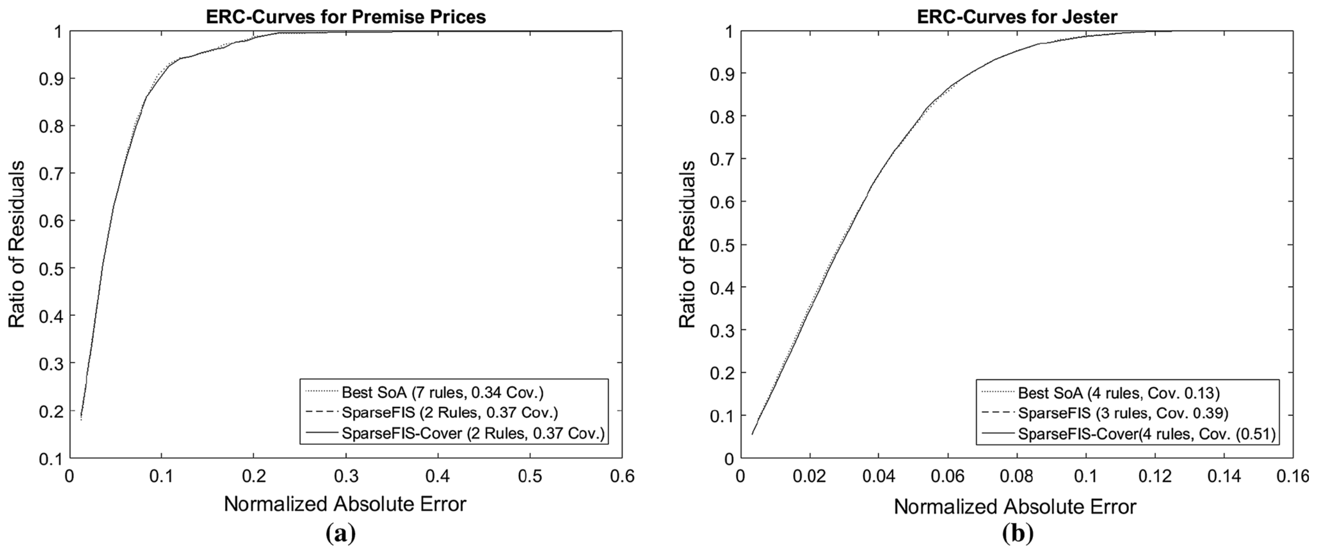

Fig. 3 **a** ERC curves for the premise price data set as achieved by original SparseFIS (*dashed line*), the best related SoA method (*dotted line*) and SparseFIS-Cover (*solid line*), the legend explains the number of rules and the coverage degree, **b** the same for the jester data; the three ERC curves are almost identical in both cases

error achieved by SparseFIS-Cover, but in all but one case (i.e., for Auto-MPG) the coverage is significantly worse (compare Columns #6 and #7, or #9 and #10 for ϵ-completeness).

The error characteristic curves (ERCs) in the figures (Figs. 1, 2 and 3) summarize these findings by representing the three measures in one plot (per data set) in a compact form. Moreover, the curves show the distribution of the residuals (ratio of residuals) in an ascending sorted manner—steeper curves point to more residuals in the lower range, thus to models with better error distributions. SparseFIS-cover tendentially shows slightly better ERC curves than SparseFIS, but slightly worse ones than the best SoA method, whereas its induced sample coverage is significantly higher and the extracted number of rules significantly lower in most cases (both mentioned in the legends of the figures).

6 Conclusion

In this paper, we presented a new fuzzy modeling method termed SparseFIS-Cover, which is able to perform a top–down learning scheme from a pre-defined upper (allowed) number of rules to find an appropriate number for the current problem at hand. This is achieved within an intervened numerical optimization procedure (Algorithm 1) (1) while respecting a minimal coverage of the sample space by the rules in terms of approaching ϵ-completeness and (2) by consecutively optimizing the rule centers and spreads synchronously to the rule weights reflecting the importance of rules. To our best knowledge, current SoA techniques are not respecting the coverage and ϵ-completeness aspects (which are important criteria in terms interpretability assurance of fuzzy systems, see [12, 23, 46]), neither within the optimization of parameters nor within structural learning phases. The second point above assures more homogenization between rule weight learning and rule spread adaptation, leading to less complex models than its predecessor SparseFIS with even lower model errors and better coverage. In this way, also several other fuzzy modeling methods could be outperformed in terms of coverage and model complexity. Among all examined data sets, the average coverage degree is 0.29 for SparseFIS, 0.26 for the best related SoA method and 0.37 for SparseFIS-Cover, while the average number of rules is 18.8 for SparseFIS, 15.5 for the best related SoA method and only 5.7! for SparseFIS-Cover (so one-third of the others two). This is remarkable as the average percentual error (MAE) is 5.16 for SparseFIS, 5.01 for the best related SoA method and 5.15 for SparseFIS-Cover, which does not imply a clear, significant difference among the methods.

Future work will address the extension of SparseFIS-Cover to the data stream mining case by developing incremental optimization procedures for solving (7) in incremental manner with high flexibility but still good convergence properties of the parameters (ideally converging to the batch solution). This will be connected in homogeneous manner with an appropriate rule reactivation or even with a rule evolution concept in order to account for significant system dynamics and non-stationary environments. Finally, we will then investigate and develop the

ϵ-completeness and coverage assurance for *evolving fuzzy systems* as defined and positioned in [23].

Acknowledgements The first author acknowledges the support of the Austrian COMET-K2 programme of the Linz Center of Mechatronics (LCM), funded by the Austrian federal government and the federal state of Upper Austria, and the support of the COMET Project 'Heuristic Optimization in Production and Logistics' (HOPL), #843532 funded by the Austrian Research Promotion Agency (FFG). This publication reflects only the authors' views.

References

1. Akerkar, R., Sajja, P.: Knowledge-Based Systems. Jones & Bartlett Learning, Sudbury (2009)
2. Babuska, R.: Fuzzy Modeling for Control. Kluwer Academic Publishers, Norwell (1998)
3. Castro, J., Delgado, M.: Fuzzy systems with defuzzification are universal approximators. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **26**(1), 149–152 (1996)
4. Celikyilmaz, A., Türksen, I.: Modeling Uncertainty with Fuzzy Logic: With Recent Theory and Applications. Springer, Berlin (2009)
5. Cernuda, C., Lughofer, E., Röder, T., Märzinger, W., Reischer, T., Pawliczek, M., Brandstetter, M.: Self-adaptive non-linear methods for improved multivariate calibration in chemical processes. Lenzing. Ber. **92**, 12–32 (2015)
6. Chiu, S.: Fuzzy model identification based on cluster estimation. J. Intell. Fuzzy Syst. **2**(3), 267–278 (1994)
7. Cohen, A., Dahmen, W., DeVore, R.: Compressed sensing and best *k*-term approximation. J. Am. Math. Soc. **22**(1), 211–231 (2009)
8. Cordon, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L.: Ten years of genetic fuzzy systems: current framework and new trends. Fuzzy Sets Syst. **141**(1), 5–31 (2004)
9. Daubechies, I., Defrise, M., Mol, C.D.: An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Commun. Pure Appl. Math. **57**(11), 1413–1457 (2004)
10. Dennis, J.E., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Prentice-Hall Series in Computational Mathematics, Englewood Cliffs (1983)
11. Fletcher, R.: Practical Methods of Optimization. Wiley, New York (2000)
12. Gacto, M., Alcala, R., Herrera, F.: Interpretability of linguistic fuzzy rule-based systems: an overview of interpretability measures. Inf. Sci. **181**(20), 4340–4360 (2011)
13. Gray, R.: Vector quantization. IEEE ASSP Mag. **1**(2), 4–29 (1984)
14. Gustafson, D., Kessel, W.: Fuzzy clustering with a fuzzy covariance matrix. In: Proceedings of the IEEE CDC Conference, pp. 761–766. San Diego, CA (1979)
15. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction, 2nd edn. Springer, New York, Berlin Heidelberg (2009)
16. Hensel, A., Spittel, T.: Kraft- und Arbeitsbedarf bildsamer Formgebungsverfahren. VEB Deutscher Verlag für Grundstoffindustrie (1978)
17. Iglesias, J., Angelov, P., Ledezma, A., Sanchis, A.: Evolving classification of agent's behaviors: a general approach. Evol. Syst. **1**(3), 161–172 (2010)
18. Iglesias, J., Tiemblo, A., Ledezma, A., Sanchis, A.: Web news mining in an evolving framework. Inf. Fusion **28**, 90–98 (2016)
19. J. Casillas, F.H., Pereza, R., Jesus, M.D., Villar, P.: Special issue on genetic fuzzy systems and the interpretability-accuracy trade-off. Int. J. Approx. Reason. **44**(1), 1–3 (2007)
20. Klement, E., Mesiar, R., Pap, E.: Triangular Norms. Kluwer Academic Publishers, Dordrecht, Norwell, New York, London (2000)
21. Lughofer, E.: FLEXFIS: a robust incremental learning approach for evolving TS fuzzy models. IEEE Trans. Fuzzy Syst. **16**(6), 1393–1410 (2008)
22. Lughofer, E.: Evolving Fuzzy Systems—Methodologies, Advanced Concepts and Applications. Springer, Berlin Heidelberg (2011)
23. Lughofer, E.: On-line assurance of interpretability criteria in evolving fuzzy systems—achievements, new concepts and open issues. Inf. Sci. **251**, 22–46 (2013)
24. Lughofer, E., Cernuda, C., Kindermann, S., Pratama, M.: Generalized smart evolving fuzzy systems. Evol. Syst. **6**(4), 269–292 (2015)
25. Lughofer, E., Kindermann, S.: SparseFIS: data-driven learning of fuzzy systems with sparsity constraints. IEEE Trans. Fuzzy Syst. **18**(2), 396–411 (2010)
26. Lughofer, E., Macian, V., Guardiola, C., Klement, E.: Identifying static and dynamic prediction models for nox emissions with evolving fuzzy systems. Appl. Soft Comput. **11**(2), 2487–2500 (2011)
27. Lughofer, E., Smith, J.E., Caleb-Solly, P., Tahir, M., Eitzinger, C., Sannen, D., Nuttin, M.: Human-machine interaction issues in quality control based on on-line image classification. IEEE Trans. Syst. Man Cybern. Part A Syst. Humans **39**(5), 960–971 (2009)
28. Lughofer, E., Trawinski, B., Trawinski, K., Kempa, O., Lasota, T.: On employing fuzzy modeling algorithms for the valuation of residential premises. Inf. Sci. **181**(23), 5123–5142 (2011)
29. Lughofer, E., Weigl, E., Heidl, W., Eitzinger, C., Radauer, T.: Integrating new classes on the fly in evolving fuzzy classifier designs and its application in visual inspection. Appl. Soft Comput. **35**, 558–582 (2015)
30. Marquardt, D.: An algorithm for least-squares estimation of nonlinear parameters. SIAM J. Appl. Math. **11**(2), 431–441 (1963)
31. Nelles, O.: Nonlinear Syst. Identif. Springer, Berlin (2001)
32. Nguyen, H., Sugeno, M., Tong, R., Yager, R.: Theor. Asp. Fuzzy Control. Wiley, New York (1995)
33. Oliveira, J.V.D., Pedrycz, W.: Advances in Fuzzy Clustering and its Applications. Wiley, Hoboken (2007)
34. Pal, N., Chakraborty, D.: Mountain and subtractive clustering method: improvement and generalizations. Int. J. Intell. Syst. **15**(4), 329–341 (2000)
35. Pedrycz, W., Gomide, F.: Fuzzy Systems Engineering: Toward Human-Centric Computing. Wiley, Hoboken (2007)
36. Pratama, M., Anavatti, S., Angelov, P., Lughofer, E.: PANFIS: a novel incremental learning machine. IEEE Trans. Neural Netw. Learn. Syst. **25**(1), 55–68 (2014)
37. Pratama, M., Anavatti, S., Garret, M., Lughofer, E.: Online identification of complex multi-input-multi-output system based on generic evolving neuro-fuzzy inference system. In: Proceedings of the IEEE EAIS 2013 workshop (SSCI 2013 conference), pp. 106–113. Singapore (2013)
38. Rong, H.J., Sundararajan, N., Huang, G.B., Saratchandran, P.: Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. Fuzzy Sets Syst. **157**(9), 1260–1275 (2006)
39. Schaffer, C.: Overfitting avoidance as bias. Mach. Learn. **10**(2), 153–178 (1993)

40. Serdio, F., Lughofer, E., Pichler, K., Pichler, M., Buchegger, T., Efendic, H.: Fault detection in multi-sensor networks based on multivariate time-series models and orthogonal transformations. Inf. Fusion **20**, 272–291 (2014)
41. Siler, W., Buckley, J.: Fuzzy Expert Systems and Fuzzy Reasoning: Theory and Applications. Wiley, Chichester, West Sussex (2005)
42. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. **15**(1), 116–132 (1985)
43. Tikhonov, A., Arsenin, V.: Solutions of Ill-Posed Problems. Winston & Sons, Washington (1977)
44. Vetterlein, T., Ciabattoni, A.: On the (fuzzy) logical content of cadiag-2. Fuzzy Sets and Syst. **161**, 1941–1958 (2010)
45. Vetterlein, T., Mandl, H., Adlassnig, K.P.: Fuzzy arden syntax: a fuzzy programming language for medicine. Artif. Intell. Med. **49**, 1–10 (2010)
46. Zhou, S., Gan, J.: Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy systems modelling. Fuzzy Sets Syst. **159**(23), 3091–3131 (2008)
47. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. Royal Stat. Soc, Series B 301–320 (2005)

Edwin Lughofer received his Ph.D. degree from the Johannes Kepler University Linz (JKU) in 2005. He is currently Key Researcher with the Fuzzy Logic Laboratorium Linz / Department of Knowledge-Based Mathematical Systems (JKU) in the Softwarepark Hagenberg, see www.flll.jku.at/staff/edwin/. He has participated in several basic and applied research projects on European and national level, with a specific focus on topics of *Industry 4.0* and *FoF (Factories of the Future)*. He has published around 160 publications in the fields of evolving fuzzy systems, machine learning and vision, data stream mining, active learning, classification and clustering, fault detection and diagnosis, including 60 journals papers in SCI-expanded impact journals, a monograph on 'Evolving Fuzzy Systems' (Springer) and an edited book on 'Learning in Non-stationary Environments' (Springer). He is associate editor of the international journals IEEE Transactions on Fuzzy Systems, Evolving Systems, Information Fusion, Soft Computing and Complex and Intelligent Systems, the general chair of the IEEE Conference on EAIS 2014 in Linz, the publication chair of IEEE EAIS 2015, 2016 and 2017, and the Area chair of the FUZZ-IEEE 2015 conference in Istanbul. He co-organized around 20 special issues and special sessions in international journals and conferences. In 2006 he received the best paper award at the International Symposium on Evolving Fuzzy Systems, in 2013 the best paper award at the IFAC conference in Manufacturing Modeling, Management and Control (800 participants) and in 2016 the best paper award at the IEEE Intelligent Systems Conference.

Stefan Kindermann was born in 1972 in Freistadt, Austria. He studied industrial mathematics at the Johannes Kepler University (JKU) in Linz, Austria, where he also received his Master's degree in 1996 and his Ph.D. degree in 2001. From 2001 to 2004, he worked as research assistant at the Industrial Mathematics Institute at the JKU in Linz. From 2004 to 2005, he was CAM assistant professor at the UCLA, Los Angeles and from 2005 to 2006 at the Johann Radon Institute for Computational and Applied Mathematics (RICAM) in Linz. Since 2006 he is assistant professor at the Industrial Mathematics Institute at the JKU, where he was habilitated in 2010. His main field of research is on the regularization of inverse problems and applications thereof. He is the co-editor of one proceedings volume and the (co)author of more than 45 publications in refereed journals.

Dr. Mahardhika Pratama received his Ph.D. degree from the University of New South Wales, Australia in 2014. He completed his Ph.D. in 2.5 years with a special approval of the UNSW higher degree committee due to his outstanding Ph.D. research achievement. Dr. Pratama is currently working at the Department of Computer Science and IT, La Trobe University, Melbourne, Australia as Lecturer. Prior to joining La Trobe University, he was with the Centre of Quantum Computation and Intelligent System, University of Technology, Sydney as a postdoctoral research fellow of Australian Research Council Discovery Project. Dr. Pratama received various competitive research awards in the past 5 years, namely the Institution of Engineers, Singapore (IES) Prestigious Engineering Achievement Award in 2011, the UNSW high impact publication award in 2013 and 2014. Dr. Pratama has published over 50 high-quality papers in journals and conferences and has been invited to deliver keynote speeches in international conferences. Dr. Pratama serves as an Editor In Chief of International Journal of Business Intelligence and Data Mining. Dr. Pratama is a member of IEEE, IEEE Computational Intelligent Society (CIS) and IEEE System, Man and Cybernetic Society (SMCS), and Indonesian Soft Computing Society (ISC-INA) and severs as a reviewer in some top tier journals such as: IEEE Transactions on Fuzzy Systems, IEEE Transactions on Cybernetics, Neurocomputing, Soft Computing, and Evolving Systems. His research interests involve machine learning, computational intelligent, evolutionary computation, fuzzy logic, neural network and evolving adaptive systems.

José de Jesús Rubio is a full-time professor of the Sección de Estudios de Posgrado e Investigación, ESIME Azcapotzalco, Instituto Politécnico Nacional. He has published 97 papers in International Journals, 1 International Book, 8 chapters in International Books, and he has presented 31 papers in International Conferences with 800 citations. He is a member of the IEEE AFS Adaptive Fuzzy Systems. He is member of the National Systems of Researchers with level II. He has been the tutor of 4 P.Ph.D. students, 5 Ph.D. students, 31 M.S. students, 4 S. students, and 17 B.S. students.