Rule Weight Optimization and Feature Selection in Fuzzy Systems with Sparsity Constraints [*]

Edwin Lughofer[1] Stefan Kindermann[2]

1. Department of Knowledge-Based Mathematical Systems, Johannes Kepler University
Altenbergerstrasse 69, A-4040 Linz, Austria
2. Department of RICAM, Johannes Kepler University
Altenbergerstrasse 69, A-4040 Linz, Austria
Email: edwin.lughofer@jku.at, stefan.kindermann@indmath.uni-linz.ac.at

Abstract— *In this paper, we are dealing with a novel data-driven learning method (*SparseFIS*) for Takagi-Sugeno fuzzy systems, extended by including rule weights. Our learning method consists of three phases: the first phase conducts a clustering process in the input/output feature space with iterative vector quantization. Hereby, the number of clusters = rules is pre-defined and denotes a kind of upper bound on a reasonable granularity. The second phase optimize the rule weights in the fuzzy systems with respect to least squares error measure by applying a sparsity-constrained steepest descent optimization procedure. This is done in a coherent optimization procedure together with elicitation of consequent parameters. Depending on the sparsity threshold, more or less rules weights can be forced towards 0, switching off some rules. In this sense, a rule selection is achieved. The third phase estimates the linear consequent parameters by a regularized sparsity constrained optimization procedure for each rule separately (local learning approach). Sparsity constraints are applied here in order to force linear parameters to be 0, triggering a feature selection mechanism per rule. In some cases, this may also yield a global feature selection, whenever the linear parameters of some features in each rule are near 0. The method is evaluated based on high-dimensional data from industrial processes and based on benchmark data sets from the internet and compared to well-known batch training methods in terms of accuracy and complexity of the fuzzy systems.*

Keywords— Takagi-Sugeno fuzzy systems, iterative vector quantization, rule weight optimization, sparsity constraints, rule selection, embedded feature selection

1 Introduction

1.1 Motivation and State of the Art

Nowadays data-driven learning of fuzzy systems enjoy a great attraction in many industrial applications, as, opposed to expert-based fuzzy systems, they can be fully automatically generated from process data such as measurements, images (features) or signal streams and hence save a lot of time and money for the development of system models. Their big advantage over other methods such as neural networks or genetic programming are that they are universal approximators [22] (i.e. being able to model a given problem to any degree of accuracy) and at the same time they still allow some insights in form of linguistically [5] and visually interpretable rules [19]. Due to these benefits, data-driven fuzzy systems are applied in many application areas such as fault detection, image classification or identification models in control systems and hence serve as important components in these systems.

During the recent 15 to 20 years a lot of data-driven fuzzy systems approaches were developed: some methods such as popular *genfis2* [23] (as implemented in MATLAB's fuzzy logic toolbox) and its successor *genfis3*, *FMCLUST* [2], exploit various clustering techniques (e.g. Gustafsson-Kessel [14] or variations of mountain and subtractive clustering [6]) for extraction clusters out of the data which are directly associated with the rules. A big family of design methods use genetic algorithms [9] for 1.) eliciting the optimal number of rules by achieving a reasonable tradeoff between accuracy and model complexity and 2.) for (fine-)tuning the parameters: these are also called genetic fuzzy systems [10] and enjoy a great attraction in the fuzzy community since more than 10 years [8].

Other methods use numerical optimization procedures for linear (consequent) and non-linear (antecedent) parameters in the fuzzy systems for minimizing the least squares errors (and variants) between estimated and predicted target values: a well-known method is the *ANFIS* approach [15] exploiting a neuro-fuzzy system architecture similar to a Takagi-Sugeno type system; another method demonstrated in [3] uses Levenberg-Marquardt algorithm for optimizing non-linear antecedent parameters in triangular and Gaussian fuzzy sets. *RENO* [4] exploits a generalized Gauss-Newton like method as a second-order approximation method with (regularized) smoothing constraints.

None of these methods perform an implicit feature selection approach, neither globally nor locally, which may be important in case of high-dimensional spaces for decreasing curse of dimensionality effect. Moreover, most of these methods do not have an integrated approach for a deterministic optimization of rule structure (number of rules) and consequent parts in a coherent procedure at the same time. Indeed, genetic fuzzy systems provide this aspect, however are non-deterministic and often lack efficiency in terms of computational performance.

[*]This work was supported by the Upper Austrian Technology and Research Promotion. This publication reflects only the author's view.

1.2 Our Approach

In this paper, we present a novel technique for data-driven learning of Takagi-Sugeno fuzzy systems called *SparseFIS*, which is short for *Sparse Fuzzy Inference Systems*. This approach is characterized by mainly three issues (more details in Section 2):

1. it applies a top-down approach where the upper bound on the number of rules are allowed to be pre-defined (e.g. by an expert) and the non-linear parameters in the fuzzy sets of the antecedent parts are pre-estimated by an iterative vector quantization approach (Section 3).

2. it performs a rule selection strategy by optimizing rule weights together with the linear consequent parameters within one non-linear optimization procedure (Section 4). The application of a threshold operator therein (for forcing as much as possible rule weights towards zero) allows optimality in the least squares sense and readability at the same time.

3. it includes an embedded local (=for each rule separately) feature selection technique by applying a semi-smooth Newton method for a sparsity regularized estimation of the linear consequent parameters (Section 5). This may decrease the curse of dimensionality effect and improves interpretability of the achieved fuzzy systems.

Combining these three concepts together will guide us to the *SparseFIS* algorithm in Section 6, which is able to generate accurate and sparse fuzzy systems at the same time. Section 7 compares the novel method in terms of predictive accuracy and complexity with other well-known (and in MATLAB achievable) data-driven fuzzy system methods based on four data sets from the UCI respository and on noisy engine test bench data.

2 Problem Statement

Let us first define the Takagi-Sugeno fuzzy model with rule weights and by applying product t-norm with Gaussian fuzzy sets (in case of C rules):

$$y_{fuz} = \sum_{i=1}^{C} l_i(x)\Psi_i(x,\rho), \quad l_i(x) = \sum_{k=1}^{p} w_{i,k}x_k \tag{1}$$

with

$$\mu_i(x) = \Pi_{j=1}^{p}\mu_{i,j}(x_j), \quad \mu_{i,j}(x_j) = e^{-\frac{(x_j-c_{i,j})^2}{2\sigma_{i,j}^2}}$$

and

$$\Psi_i(x,\rho) = \frac{\rho_i\mu_i(x)}{\sum_{j=1}^{C}\rho_j\mu_j(x)} \tag{2}$$

Here $\rho = (\rho_1, \ldots \rho_C)$ are positive weights, which more or less steer the number of rules (as weights smaller than a low number $\epsilon > 0$ can be seen as not useful and hence omitted). For optimization purposes we use the least squares error measure, which, when applying TS fuzzy systems as above and in dependency of the unknown parameters, can be written as

$$J(w,c,\sigma,\rho) = \sum_{k=1}^{N}(y_k - \sum_{i=1}^{C} l_i(\vec{x}_k)\Psi_i(\vec{x}_k))^2 \tag{3}$$

The training goals (=steps in the algorithm) in this paper can be summarized as follows:

1. Estimating the non-linear antecedent parameters in a preliminary phase due to a pre-defined upper bound on the number of rules.

2. Minimizing the least squares error by keeping the model complexity (in terms of the number of rules) as low as possible (rule selection from the initial number). This is done in a coherent procedure with optimization of the consequent parameter synchronously.

3. Minimizing the least squares error by keeping the number of significant linear consequent parameters in each rule as low as possible. This has the effect of a kind of local feature selection (which can turn into a global one if certain parameters belonging to the same input variable are not significant in any rule) and hence serves as an additional interpretability aspect as well as regularization property.

The last two steps guarantee a reasonable tradeoff between accuracy and complexity resp. readability of the achieved models, as we optimize the rules resp. consequent parameters according to an optimization criterion by applying synchronously constraints on the number of significant rules resp. significant features in the consequents. This also means that this technique does not need any computational intensive post-processing techniques (such as e.g. [20]) for reducing complexity. In the next three sections we describe how to achieve these three goals.

3 Estimating Antecedent Parts of the Rules

The first step is to yield a proper estimation of the non-linear premise parameters c and σ in the Gaussian fuzzy sets of the rules antecedent parts. For doing so, we apply a clustering approach, which is called iterative vector quantization as iterating over the whole data set a multiple times, and associate each cluster to one rule. The fuzzy sets of the rules antecedent parts are obtained by projection onto the one-dimensional axis: the centers of the fuzzy sets are the center components of a cluster in each dimension, the widths are achieved by calculating the variance in each dimension. In principle, any clustering method requiring a fixed number of cluster as input and delivering cluster prototypes = centers of fuzzy sets and rules as well as the range of influence of the clusters in each dimension can be applied here. In fact, the projection concept is also carried out in many other approaches such as genfis2 [6] or FMCLUST [2] and hence not new. The difference in this paper is that we are applying vector quantization [12] where the a-forementioned methods apply other clustering algorithms (such as subtractive clustering and Gustafson-Kessel).

First, we estimate an initial number of rules (=clusters) and send these to the iterative vector quantization algorithm for fine tuning the non-linear parameters. The number of initial rules are

- Either pre-defined according to user knowledge, interpretability or computational aspects (usually $<$ 100 rules).
- Or automatically extracted from the data by using an evolving clustering approach (*eVQ* [16]) which starts with 0 clusters and evolve new ones according to the distribution and characteristics of new data.

Then, we apply the iterative vector quantization (VQ) algorithm (please refer to [12]) plus estimate the ranges of influence of each cluster in each dimension after the end of VQ.

4 Rule Selection by Optimizing the Rule Weights

Once having the centers and widths of the predefined number (upper bound) of rules estimated, it is a big challenge to reduce the rules in order to enhance interpretability by synchronously not loosing significant accuracy. In this sense, we propose to optimize the rule weights as defined in (2) within a iterative optimization process using sparsity constraints. In a sense, we try to find a minimal least squares error solution subject to the constraint that a significant number of rule weights should be zero. In an optimization formulation a goal would be to minimize the least squares error (3) subject to the constraint that the number of weights ρ_i should be bounded:

$$\min J(w,c,\sigma,\rho) \quad \text{such that}\#\{\rho_i \ / \tag{4}$$

where $\#$ denotes the cardinality of a set and K is an appropriate constant. A minimizer of such a problem is expected to achieve a small error together with a small number of nonzero rule weights. However, there is an intrinsic difficulty with this formulation, as this is a combinatorial optimization procedure (note that we only bound the number of nonzero elements, there is no restriction, where the nonzero elements are). A method that solves such a problem would have to test all subsets of $\{1,\dots N\}$ as candidates for the nonzero entry index of ρ and hence requires exponential complexity in N. However, recently the surprising fact came up, that an almost optimal solution to (4) can be computed by l^1-minimization [7]: That is, instead of (4) the problem

$$\min J(w,c,\sigma,\rho) \quad \text{such that} \sum_{m=1}^{N} |\rho_i| \le K, \tag{5}$$

is solved. This is a much more accessible approach, because the constraints are convex and there are efficient methods to solve it.

Our algorithm is a nonlinear version of the well-known iteration proposed by [11]. The iteration is basically a projected gradient descent algorithms. The gradient of least squares functional with respect to the jth rule weight ρ can be calculated to:

$$\frac{\partial y_{fuz}}{\partial \rho_j} = \sum_{n=1}^{C}\sum_{k=1}^{N} e_k l_n(x_k)\frac{\partial \Psi_n(x_k)}{\partial \rho_j} \tag{6}$$

with $l_n(x_k)$ the linear rule consequent functions in sample x_k, e_k the residual in sample x_k defined by:

$$e_k(c,\sigma,\rho) := y_k - \sum_{i=1}^{C} l_i(\vec{x}_k)\Psi_i(c,\sigma,\rho)(\vec{x}_k)) \tag{7}$$

and $\frac{\partial \Psi_n(x_k)}{\partial \rho_j}$ the partial derivative of the nth basis function with respect to ρ_j:

$$\frac{\partial \Psi_i}{\partial \rho_k} = \frac{\mu_i}{\sum_{j=1}^{C}\rho_j\mu_j(x)}\left(\delta_{i,k} - \frac{\rho_i\mu_k}{\sum_{j=1}^{C}\rho_j\mu_j(x)}\right) \tag{8}$$

where

$$\delta_{i,j} = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{if } i\neq j \end{cases}$$

The idea of the projected gradient method is to apply a threshold operator on the rule weight update in each gradient descent iteration step T_α. Additionally, we also have to optimize the consequent parameters in each update step as needed in $l_n(x_k)$ and e_k. This is done by the local learning approach, estimating the parameters for each rule separately and independently. This has some robustness [18], computational, flexibility [1] and interpretability [24] advantages over global learning approach. In this sense, the whole rule weight optimization procedure becomes as in algorithm 1.

Algorithm 1 Rule Weight Optimization with Sparsity Constraints

1. Initialization of w^0 is done by using local learning with (11) on the initial number of rules (see Step 5 below); ρ^0 is set to a vector of ones (all rules contribute equally at the beginning).
2. w^k, ρ^k are given in iteration k.
3. Calculate $\frac{\partial \Psi_n}{\partial \rho_j}$ as given in (8).
4. Update the rule weights $j = 1,\dots,C$ componentwise separately by:

$$\rho_j^{k+1} = T_\alpha\left(\rho_j^k - \tau\sum_{n=1}^{C}\sum_{k=1}^{N} e_k l_n(x_k)\frac{\partial \Psi_n(x_k)}{\partial \rho_j}\right) \tag{9}$$

with τ a small constant defining the step size per iteration (usual setting $\tau = 0.1$) and T_α the soft-thresholding operator defined by:

$$T_\alpha = \max(\rho_j^{k+1}, \alpha) - \alpha \tag{10}$$

with α set to a small constant (usual guess is 0.1).

5. Calculate $\Psi(c, \sigma, \rho^{k+1})$ for the new $\rho_j^{k+1}, j = 1, ..., C$ and generate the regression matrix $\sqrt{Q_i}R_i, i = 1, ..., C$ 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 $diag(\Psi_i(\vec{x}(k))), k = 1, ..., N$.

6. Perform the weighted least squares approach for the linear consequent parameters whose analytical solution in closed form is:

$$\hat{\vec{w}}_i = (R_i^T Q_i R_i)^{-1} R_i^T Q_i \vec{y} \tag{11}$$

   Here, we also apply Tikhonov regularization [21] if required, i.e. when the condition of the matrix $R_i^T Q_i R_i$ is very high, indicating a matrix close to singular and hence an unstable solution in (11).

7. If the difference between two iterations $\|\rho^{k+1} - \rho^k\|$ is low enough or a pre-defined maximal number of iterations is reached, then stop, otherwise perform next iteration (goto Step 2).

After this optimization procedure, the rules with weights close to 0 are eliminated and the remaining rules C_{rem} are kept for further processing through the last step (described in the subsequent section). Note the in view of our derivation of this iteration, the output of this method should be on the one hand close to the data (this is done by minimizing the least squares functional), and have many rule weights as zero, (which can be discarded afterwards).

In order to visualize the update progress of the rule weights through Algorithm 1, Figure 1 shows the rule weights from three iterations (1, 7 and 13). The weights are sorted in order to get an idea how they are reduced through the optimization procedure by applying T_α. Clearly, after the first iteration almost all rule weights are bigger than zero, whereas after the 13th iterations more than 60 rules can be deleted.

5 Feature Selection by Learning of Consequent Parameters with Sparsity Constraints

Now the final step in our algorithm is to perform a final estimation of the linear consequent parameters $\vec{w}_i, i = 1, ...C_{rem}$ for all rules separately. Indeed, the outcome of Algorithm 1 contains not only the rule weights but also estimates the consequent parameters: these are optimized with standard weighted least squares method (and eventually plus a regularization term) in order to minimize least squares error. Here, we want to go a step further and make the consequent parameters also sparse, i.e. forcing as many of these as possible towards zero, again without loosing (significant) accuracy as optimized in an iterative optimization procedure called Semi-smooth Newton method. This evoke a local feature selection, as variables with low weights may be ignored. Furthermore, if through this selection scenario it turns out that certain variables have very low weights in all rules, it can be concluded that these ones are not necessary for the whole global fuzzy model at all.

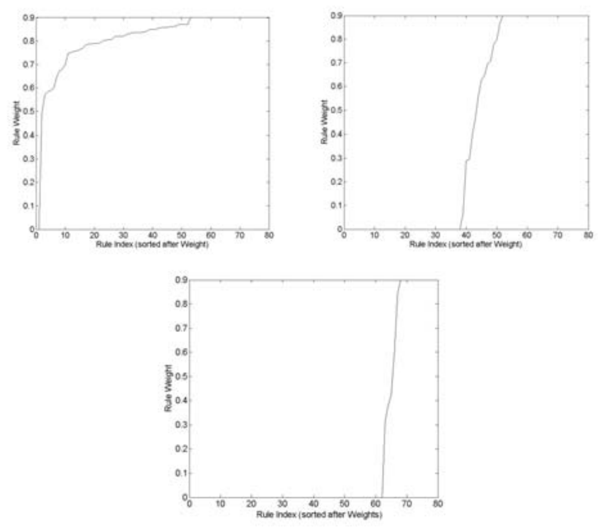

Figure 1: From left to right: sorted rule weights after iteration step 1, 7 and 13; note how the rule base is sparsed out from iteration to iteration (more and more rule weights become 0)

The consequent parameter $\vec{w}_i$ appear linearly in the Function F, in contrast to the rule weights. Although adding sparsity constraints make the optimization a nonlinear problem, because of the linear structure it seems favorable to use a Newton-type method for this problem. Despite the fact that a sparsity constraint problem is usually not differentiable, a generalized version of derivative (the slant derivative) can be defined and subsequently a Newton-type iteration (the Semi-smooth Newton method, [13]) can be applied. The main steps are a soft shrinkage operation on the residual and a Newton step, which makes use of active and inactive sets, see Algorithm 2. For the sake of simplicity, we define this algorithm for the ith rule. Local convergence for this iteration has been shown e.g. in [13].

Algorithm 2 Semi-Smooth Newton with Sparsity Constraints

Let $\sigma > 0$ and parameter k be an iteration index, and M_i the following covariance matrix for local learning

$$M_i = R_i^T Q_i R_i$$

then we apply the following steps:

1. Initialization is done by simply taking the latest estimated $\vec{w}$ (in the last iteration of Algorithm 1).

2. For the kth step we have already $\vec{w}_i^k$ evaluated

3. Step $k \rightarrow k+1$:
   Calculate

$$\vec{u}_{ik} := \vec{w}_i^k - \sigma(R_i^T Q_i(R_i \vec{w}_i^k - \vec{y}))$$

4. Elicit in $\vec{u}_{ik}$ those indices which are greater or smaller than σ (with σ a small value, usually set to 0.1 and not sensitive):

$$\mathcal{A} := \{j : |(\vec{u}_i^k)_j| > \sigma\} \quad \mathcal{I} := \{j : |(\vec{u}_i^k)_j| \leq \sigma\}$$

ISBN: 978-989-95079-6-8

5. Calculate the residual by applying the sparsity constraint operator T_α as defined in (10):

$$\vec{r}_i^k = T_\alpha(\vec{w}_i^k - \sigma(R_i^T Q_i(R_i \vec{w}_i^k - \vec{y})))$$

6. Calculate the update: for all indices on the inactive set:
$$(update_I)_j = -(\vec{r}_i^k)_j \quad \text{für all } j \in \mathcal{I}$$

7. Divide matrix M_i in active and inactive sets:

$$M_{\mathcal{A},\mathcal{A}} := (M_i)_{k,l}, \quad k,l \in calA$$

$$M_{\mathcal{A},\mathcal{I}} := (M_i)_{k,l}, \quad k \in calA, l \in calI$$

8. Calculate update on the inactive set:

$$(update_A)_j = (M_{\mathcal{A},\mathcal{A}})^{-1} \left(\frac{1}{\sigma}(-\vec{r}_i^k)) - M_{\mathcal{A},\mathcal{I}} update_I \right)$$

9. Assign the consequent vector the current update vector one times for the active, one times for the inactive set:
$$\vec{w}_i^{k+1} = update$$

10. **If** the residual $\|\vec{r}_i^k\|$ is large and the iteration smaller than the maximal # of iterations, goto Step 2

An example of the impact of this algorithm on the development of the sparseness of the consequent parameters and hence local/global feature selection will be demonstrated in Section 7.

6 SparseFIS - The Algorithm

The whole *SpareFIS* method is simply obtained by combining the three algorithms described in the previous three sections to one, so applying iterative vector quantization first, then rule weight optimization and finally semi-smooth Newton method for sparse learning of consequent parameters. The explicit formulation is not made here due to space restrictions.

7 Evaluation

7.1 Experimental Setup

The evaluation of the novel method *SparseFIS* includes a comparison with other famous fuzzy batch learning algorithms such as *genfis2* [6] [23], *ANFIS* [15], *FMCLUST* [2] and the batch off-line version of the evolving fuzzy system approach *FLEXFIS* [17], which are all available as implementations in MATLAB. The evaluation will be based on five different high-dimensional data sets for regression: 1.) four data sets from the UCI repository [1] (clean, noise-free): auto-mpg, housing, forrest fires and concrete and 2.) one data set from an engine test bench, where measurement data were collected on-line and is affected by some white noise and disturbances.

The engine test bench data set includes various measurement channels together with their time delay (up to 10), the task was to build a high-performance k-step ahead prediction model for the emission channel NOX for two purposes: 1.) in order to save expenses for the sensor channel and 2.) to perform early recognition of faults (pipe leakage, sensor overheating, interface defects etc.) when using this model in a multi-channel fault detection approach.

A summary of the characteristics of all the data sets used in this evaluation is shown in Table 1. In all data set cases, the evaluation will be based on the accuracy of the models obtained by applying 10-fold cross-validation with a coupled best parameter grid search scenario. The later means that a parameter grid for the most essential parameter(s) in each method is defined consisting of a lower bound, an upper bound and a step-size and for each grid point cross-validation is performed. The best solution with respect to the CV-error (averaged mean absolute error over all folds or averaged maximal error over all folds) will be reported. For *SparseFIS* we will observe the impact of the sparsity constraints on local variable selection (per rule separately). The parameter grids were defined in the following way:

- *genfis2*: radius cluster is varied from 0.1 to 0.9 in steps of 0.05
- *ANFIS*: an initial model is constructed with genfis1 by specifying the number of fuzzy sets per input dimension varied from 2 to 5 in steps of 1
- *FMCLUST*: the number of rules is varied from 1 to 50 in steps of 1
- *FLEXFIS*: the vigilance parameter responsible for the number of clusters = rules is varied from 0.1 to 0.9 in steps of 0.05

For *SparseFIS*, we set the number of maximal allowed (initial) rules to 50 and τ to 0.05 for all experiments, requiring no parameter grid at all. All other parameters are set as mentioned in above algorithms.

7.2 Results

The obtained results are reported in Table 2. The rows represent the different methods and the columns the different data sets: the first number is the CV-error, i.e. the minimal averaged mean absolute error (averaged over the CV-folds, minimal over all parameter settings) plus-minus the standard deviation over these folds. The second number denotes the maximal error over all samples for the minimal averaged mean absolute error (so worst-case prediction error). The third number reports the complexity of the fuzzy models in terms of the number of rules (belonging to the model with minimal CV-error). The results on *SparseFIS* are reported two times, one time by using the full algorithm, one time without applying the sparsity constraint threshold T_α, so without forcing rule weights and consequent parameters towards zero (denoted as *SparseFIS uncon.*). From this table it can be realized that the full *SparseFIS* approach (with constraints) can outperform all other methods in three data sets (forrest fires, housing and NOX) in terms of mean absolute error (MAE), in the case of forest fires

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

Table 1: Some data sets from the UCI repository and their characteristics

	# Training Samples	# Input Variables	Source	Noise Level
Auto-MPG	398	8	UCI	None
Concrete	1030	8	UCI	None
Forrest Fires	517	12	UCI	None
Housing	506	13	UCI	None
NOX	667	181	Engine Test Bench	Medium to high

Table 2: Comparison of fuzzy modelling variants on four data sets from UCI repository and (noisy) NOX data from an engine test bench

Method	**Auto-MPG**			**Concrete**			**Forest Fires**		
	MAE±STD	Max	#rules	MAE±STD	Max	# rules	MAE±STD	Max	# rules
genfis2	2.23± 0.85	3.88	6	8.37± 1.70	11.52	3	19.64± 15.50	58.36	3
ANFIS	2.41± 0.84	4.07	16	11.25± 9.98	39.13	8	20.27± 15.60	58.38	2
FMCLUST	2.35± 0.91	3.99	20	7.75± 2.06	12.11	3	19.83± 15.52	58.93	2
FLEXFIS	**2.17± 0.73**	3.59	11	**7.73± 1.97**	11.96	8	19.69± 15.32	58.35	3
SparseFIS	2.20± 0.30	2.53	8	8.91± 4.55	21.01	6	**15.13± 6.76**	30.77	5
SparseFIS uncon.	2.48± 0.80	4.19	21	11.02± 5.95	25.55	14	20.13± 15.50	59.53	9

Method	**Housing**			**NOX**		
	MAE±STD	Max	#rules	MAE±STD	Max	# rules
genfis2	3.14± 1.31	6.53	4	12.97± 1.08	15.98	5
ANFIS	3.59± 1.39	6.56	4	13.72± 1.01	14.72	8
FMCLUST	2.84± 1.08	5.38	6	13.99± 1.05	15.55	6
FLEXFIS	2.98± 1.27	6.09	6	12.96± 0.98	14.31	8
SparseFIS	**2.84± 0.98**	5.65	6	**12.95± 0.97**	15.22	7
SparseFIS uncon.	3.24± 1.25	6.37	20	13.59± 0.83	14.76	27

even significantly. In the case of the auto-mpg data set, the new method performs similar than the other approaches, whereas for the concrete data set the *SparseFIS* is worse than the others. A similar interpretation of the results can be reported when inspecting the maximal error rate. The standard deviations shows us how sensitive the method is with respect to variations in the training data samples. Lower values represent less variations and hence more robust methods. The best performing methods for each data set with respect to the mean absolute error are shown in bold face. It is also worth mentioning that *SparseFIS* in its unconstrained version performs (mostly) weaker as the full approach; an explanation of this occurrence is that the short version tends to over-fit, as it does not reduce any rules or consequent parameters (and hence complexity).

Another important aspect when training fuzzy systems from data is a reasonable complexity of the obtained models in order to guide the models to more interpretable power [5]. Otherwise, the benefit of fuzzy models over other modelling techniques (such as neural networks, genetic programming etc.) is diminished. We measure the complexity of the final fuzzy models by the number of generated rules by the learning procedure applied onto the whole data set with the optimal parameter setting as obtained in the best parameter grid search and CV procedure. This number is reported in the third columns of each single part in Table 2. Here, we can realize that *SparseFIS* can compete with the other methods, although *genfis2* is mostly the best performer with respect to the number of generated rules. It is also remarkable that the numerical rule weight optimization concept as described in Algorithm 1 is able to reduce the number of rules from 50 to below 10 for all data sets (and this with a reasonable accuracy), confirming Figure 1.

Finally, in Figure 2 we inspect the impact of the sparsity constraints onto the linear consequent parameters as applied in Algorithm 2 (Step 4). Therefore, we show a plot of consequent parameters for all extracted rules (=rows) for the second (upper row) and the third features (lower row) in case of auto-mpg and using all input variables in *SparseFIS* method. While for the second feature four rules are sparsed out (compare left and right image in upper row), for the third feature all consequent parameters are forced to a small value around 0 which is not significant (note the range of this feature is 184). This means that feature #3 is not significant for modelling the target in the auto-mpg data set.

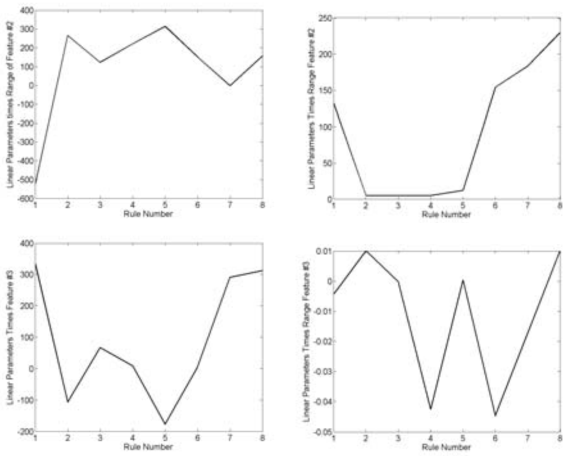

Figure 2: Left: Linear parameters times the ranges for features #2 (upper) and #3 (lower) when not applying sparsity constraints in consequent learning; right: the same as in left, but applying sparsity constraints

8 Conclusion

In this paper, we presented a novel data-driven learning technique for Takagi-Sugeno type fuzzy systems, called *SparseFIS*, which is able 1.) to perform rule selection together with consequent parameter estimation in a coherent optimization procedure and 2.) to gain more insight into the fuzzy models by yielding a local feature selection through sparsing out unimportant features in each rule. As a top-down method, it is possible to elicit an initial upper bound of number of (allowed) rules by expert-knowledge or by initial guess from the data. Tests on various multi-dimensional data sets from UCI repository and real-world applications show that the novel method can compete with other well-known learning methods in terms of both, accuracy and complexity and synchronously is able to put in more interpretation in from of local importance of the features.

References

[1] P.P. Angelov, E. Lughofer, and X. Zhou. Evolving fuzzy classifiers using different model architectures. *Fuzzy Sets and Systems*, 159(23):3160–3182, 2008.

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

[3] J. Botzheim, E. Lughofer, E.P. Klement, L. Kóczy, and T.D. Gedeon. Separated antecedent and consequent learning for takagi-sugeno fuzzy systems. In *Proceedings of FUZZ-IEEE 2006*, pages 2263–2269, Vancouver, Canada, 2006.

[4] M. Burger, J. Haslinger, U. Bodenhofer, and H. W. Engl. Regularized data-driven construction of fuzzy controllers. *J. Inverse Ill-Posed Probl.*, 10(4):319–344, 2002.

[5] J. Casillas, O. Cordon, F. Herrera, and L. Magdalena. *Interpretability Issues in Fuzzy Modeling*. Springer Verlag, Berlin Heidelberg, 2003.

[6] S. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2(3):267–278, 1994.

[7] A. Cohen, W. Dahmen, and R. DeVore. Compressed sensing and best k-term approximation. *J. Amer. Math. Soc.*, 22(1):211–231, 2009.

[8] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena. Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, 141(1):5–31, 2004.

[9] O. Cordon and F. Herrera. Identification of linguistic fuzzy models by means of genetic algorithms. *Fuzzy Model Identification. Selected Approaches*, pages 215–250, 2003.

[10] O. Cordon, F. Herrera, F. Hoffmann, and L. Magdalena. *Genetic Fuzzy Systems — Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. World Scientific, 2001.

[11] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. Pure Appl. Math.*, 57(11):1413–1457, 2004.

[12] R.M. Gray. Vector quantization. *IEEE ASSP Magazine*, pages 4–29, 1984.

[13] R. Griesse and D.A. Lorenz. A semismooth Newton method for Tikhonov functionals with sparsity constraints. *Inverse Problems*, 24(3):19 p., 2008.

[14] D. Gustafson and W. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Proc. IEEE CDC*, pages 761–766, San Diego, CA, USA, 1979.

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

[16] E. Lughofer. Extensions of vector quantization for incremental clustering. *Pattern Recognition*, 41(3):995–1011, 2008.

[17] E. Lughofer. FLEXFIS: A robust incremental learning approach for evolving TS fuzzy models. *IEEE Trans. on Fuzzy Systems (special issue on evolving fuzzy systems)*, 16(6):1393–1410, 2008.

[18] E. Lughofer. Towards robust evolving fuzzy systems. In P. Angelov, D. Filev, and N. Kasabov, editors, *Evolving Intelligent Systems: Methodology and Applications*, page to appear. John Wiley & Sons, New York, 2009.

[19] D. Nauck and R. Kruse. Nefclass-x – a soft computing tool to build readable fuzzy classifiers. *BT Technology Journal*, 16(3):180–190, 1998.

[20] M. Setnes. Simplification and reduction of fuzzy rules. In J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, editors, *Interpretability Issues in Fuzzy Modeling*, volume 128 of *Studies in Fuzziness and Soft Computing*, pages 278–302. Springer, Berlin, 2003.

[21] A.N. Tikhonov and V.Y. Arsenin. *Solutions of ill-posed problems*. Winston & Sonst, Washington D.C., 1977.

[22] L.X. Wang. Fuzzy systems are universal approximators. In *Proc. 1st IEEE Conf. Fuzzy Systems*, pages 1163–1169, San Diego, CA, 1992.

[23] R.R. Yager and D.P. Filev. Learning of fuzzy rules by mountain clustering. In *Proc. of SPIE Conf. on Application of Fuzzy Logic Technology*, pages 246–254. Boston, MA, U.S.A., 1993.

[24] J. Yen, L. Wang, and C.W. Gillespie. Improving the interpretability of TSK fuzzy models by combining global learning and local learning. *IEEE Trans. on Fuzzy Systems*, 6(4):530–537, 1998.