Incremental Learning of Fuzzy Basis Function Networks with a Modified Version of Vector Quantization

Edwin Lughofer
Fuzzy Logic Laboratorium Linz
Johannes Kepler University Linz
A-4040 Linz, Austria
E-mail: edwin.lughofer@jku.at

Ulrich Bodenhofer
Software Competence Center Hagenberg
A-4232 Hagenberg, Austria
E-mail: ulrich.bodenhofer@scch.at

Abstract

In this paper, an algorithm for data-driven incremental learning of fuzzy basis function networks is presented. A modified version of vector quantization is exploited for rule evolution and incremental learning of the rules' antecedent parts. Antecedent learning is connected in a stable manner with a recursive learning of rule consequent functions with linear parameters. The paper is concluded with an evaluation of the proposed algorithm on high-dimensional measurement data for engine test benches.

Keywords: incremental learning, fuzzy basis function networks, vector quantization, rule/cluster evolution

1 Introduction

Adaptive algorithms for data-driven models are often of fundamental importance in order to identify real-time processes that possess a time-variant behavior [17]. Beyond that, an insufficiency of amount, distribution or quality of actual recorded measurement data can occur, such that the model cannot meet the expectations at a particular time. In this case, the incorporation of newly recorded data can improve the model's accuracy and reduce the model error. This automatically yields an improvement in process security as extrapolation to new operating conditions or system states is prevented. For online identification tasks, this requires an adaptation of some model parameters in form of incremental learning steps. This is because a complete rebuilding of the models from time to time with all so far recorded measurements would yield an unacceptable computational effort. In order to meet these requirements algorithms for an incremental learning of fuzzy models are proposed in literature, such as *DENFIS* [12], *eTS* [2] or *FLEXFIS* [15]. In this paper a modified version of vector quantization is exploited (Section 3) for improving *FLEXFIS* with respect to the accuracy and complexity of the obtained models. This will be evaluated in Section 5 based on the fault detection performance at engine test benches. A special attention is given to a stable connection between antecedent and consequent learning in Section 4.

2 Definition of Fuzzy Basis Function Networks

A *fuzzy basis function network* with input variables $\mathbf{x} = (x_1, ..., x_p)$ and a single output variable y can be defined in the following way:

$$\hat{f}(\mathbf{x}) = \hat{y} = \sum_{i=1}^{C} l_i(\mathbf{x})\Psi_i(\mathbf{x}) \qquad (1)$$

with the normalized membership functions

$$\Psi_i(\mathbf{x}) = \frac{exp\left(-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j-c_{ij})^2}{\sigma_{ij}^2}\right)}{\sum_{k=1}^{C} exp\left(-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j-c_{kj})^2}{\sigma_{kj}^2}\right)} \qquad (2)$$

and consequent functions

$$l_i(\mathbf{x}) = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + ... + w_{ip}x_p \quad (3)$$

with c_{ij} the center and σ_{ij} the width of the Gaussian function appearing as fuzzy set in the j-th antecedent part (i.e. the antecedent part for the jth dimension) of the i-th rule.

When inspecting this formulation of a *fuzzy basis function network* we have to realize that principally three different kinds of components may be learned in incremental manner: linear rule consequent parameters ($w_{i0}, w_{i1}, ..., w_{ip}$), nonlinear antecedent parameters (c_{ij},σ_{ij}) and the rule base concerning the number of rules (C).

3 Antecedent Parameter Adaptation and Rule Learning

The incremental training process of the rules' antecedents is carried out with the help of incremental clustering. Whenever new clusters are born during the incremental learning process, new rules are born. Moreover, the clusters are projected onto the one-dimensional axes in order to form the fuzzy sets and rules. Hence, rules can be identified with clusters and a movement of clusters corresponds to a movement of the fuzzy sets in the rules antecedents. This is also carried out in approaches such as [4, 19]. In our approach a modified version of vector quantization is exploited for the incremental clustering process.

3.1 Vector Quantization in Incremental Mode

The purpose of vector quantization [8] originally stems from encoding discrete data vectors in order to compress data which has to be transferred quickly. It can be easily recognized that it is only applicable for offline clustering tasks as it processes through the entire data set more than one time. If this would be carried out for each incremental learning step, i.e., for each actual loaded data block separately, the cluster centers would only represent a reliable partition of this data block and forget the older data completely. Moreover, the number of clusters has to be known in advance. It should be noticed, that this problem can be indeed solved for the offline case by exploiting cluster validation indices [10] and applying them for different partitions obtained with different numbers of clusters. However, for the online case the drawback still remains as usually the data has to be sent into the algorithm as it is loaded or recorded.

Hence, for omitting these drawbacks the idea of adaptive resonance theory network [6] is exploited. In ART networks, especially in the ART-2 algorithm, the *stability/plasticity dilemma* is solved by the introduction of a *vigilance* parameter. It controls the tradeoff between adaptation of already learned clusters and generation of new clusters. In this sense, for each new data point the following condition is checked:

$$\|\mathbf{x} - \mathbf{c}_{\text{win}}\|_A \geq \rho \text{ and } \mathbf{x} \text{ is not faulty} \quad (4)$$

with $\mathbf{x}$ the actual data point, $\mathbf{c}_{\text{win}}$ the winning cluster and A the norm-inducing distance. If this condition is fulfilled, the prototype $\mathbf{c}_{C+1}$ of the new (the $C + 1$st) cluster is set to the actual data point. The second part of condition (4) assures that the new data point does not represent a faulty situation during data recordings. This is very hard to decide and further investigations needs to be done in this direction. In fact, it is true that the problem of a-priori defining the number of clusters C is shifted to finding a good value for the *vigilance* parameter ρ. However, a good guess for this parameter can be achieved much more easier, when clustering is applied onto data normalized into the hypercube $[0, 1]^p$: being far away from a new data point always can be explained with a certain distance value. In a trial and error tuning phase, it turned out that the following choice of this parameter should be preferred:

$$\rho \quad = \quad \text{fac} \cdot \frac{\sqrt{p}}{\sqrt{2}} \quad (5)$$

The dependency of ρ on the p-dimensional space diagonal can be explained with the so-called *curse of dimensionality* effect: the

higher the dimension, the greater the distance between two adjacent data points, see [11]. Therefore, the larger the parameter ρ should get in order to prevent the algorithm to generate too much clusters and causing overfitting effects. An appropriate value for fac strongly depends on the nature of the data. Usually, it can be set between 0.2 and 0.4 and is quite insensitive, but this is only a trial-and-error guess to our best knowledge. However, cluster merging and splitting strategies [5] can be modified and applied after each incremental learning step in order to compensate an inappropriate setting of fac. Therefore, after each sample-wise incremental step the updated cluster is first split into two parts and the resulting cluster structure is validated with a cluster validation index, which does not need any prior data (for instance PS-index [21]). This is compared with the validation without this split and the better structure is kept. The same procedure can be done for merging, where the cluster closest to the updated one is merged with the updated one.

With these notations and by assuming normalized data, vector quantization in incremental mode becomes:

Algorithm 1. VQ-INC

1. Initialize the number of clusters to 0
2. Take the next incoming data point $\mathbf{x}$
3. **If** the number of clusters is 0
   (a) Set $i = 1$, set the first center c_1 to the actual data point, hence $\mathbf{c}_1 = \mathbf{x}$
   (b) goto step 8
4. Calculate the distance of the selected data point to all cluster centers by using a predefined distance measure.
5. Elicit the cluster center which is closest to the data point by taking the minimum over all calculated distances $\rightarrow$ winner cluster c_{win}
6. **If** $\|\mathbf{x} - c_{\text{win}}\|_A \geq \rho$ and $\mathbf{x}$ is not faulty
   (a) Set $i = i + 1$, set $\mathbf{c}_i = \mathbf{x}$
   (b) goto step 8
7. Update the p components of the winner cluster by moving it towards the selected point $\mathbf{x}$:

$$\mathbf{c}_{\text{win}}^{(\text{new})} = \mathbf{c}_{\text{win}}^{(\text{old})} + \eta(\mathbf{x} - \mathbf{c}_{\text{win}}^{(\text{old})}) \quad (6)$$

8. If new incoming data points are still available, goto step 2, otherwise stop

From this definition it can be realized that each (newly loaded) data point is processed only once through the update process. This is quite reliable (opposed to conventional vector quantization) as clusters centers are already initialized in new local data clouds and hence restricted to move therein (because of condition (5)). For this algorithm, the learning gain η_i (for the ith cluster center) should be decreased by the number of data points belonging to a cluster, i.e., for all i

$$\eta_i = \frac{0.5}{k_i} \quad (7)$$

with k_i the number of data points belonging to cluster i. This is because, as in k-means clustering algorithm [1], the step size is also normalized by this number, whereas original vector quantization is a simplified version of k-means clustering.

3.2 An Alternative Distance Strategy

The problem with Algorithm 1 is that, in the case of wider data clouds, it tends to generate more clusters than necessary and hence performs an 'overclustering' and incorrect partition of the input space. This fact is underlined in the left image of Figure 1, where obviously three clusters are the optimal case, but five cluster are generated. The reason for this unpleasant occurrence lies at hand: Algorithm 1 compares each new incoming point with all the cluster centers. This can be far away even if the data point is close to its spanned range of influence (represented by those data points already belonging to the cluster). Therefore, an obvious overcoming of this drawback can be achieved by calculating the range of influence during the incremental learning process and taking the distance of new points to these

ranges. Whenever the Euclidean distance is used as distance measure, axis-parallel ellipsoids are obtained as clusters, whose range of influence (as 2σ-area) can be calculated in incremental mode by exploiting recursive variance formula [18]:

$$k_{\text{win}}\sigma^2_{\text{win},j} = (k_{\text{win}} - 1)\sigma^2_{\text{win},j} + k_i \Delta c^2_{\text{win},j} + (c_{\text{win},j} - x_j)^2 \quad (8)$$

where $\Delta c_{\text{win},j}$ is the distance of the old prototype to the new prototype of the cluster nearest to the actual point $\mathbf{x}$ in the jth dimension and k_{win} is the number of data points lying nearest to cluster c_{win} For the distance of the new data point to the surface of the multidimensional ellipsoid spanned by a cluster we take the distance along the direction from the actual point towards the cluster center.

Lemma 1. *Let therefore* $\sum_{j=1}^{p} \frac{(x_j - c_{ij})^2}{\sigma_{ij}^2} = 1$ *be a multidimensional ellipsoid of the* i*th cluster in main position,* σ_{ij} *the variance of the data belonging to the* i*th cluster in dimension* j*, then the Euclidian distance of the new data point* $(q_1, ..., q_p)$ *to the surface along the direction towards the cluster center* c_{ij} *is given by*

$$dist = (1 - t)\sqrt{\sum_{j=1}^{p}(q_j - c_{ij})^2} \quad (9)$$

with

$$t = \frac{1}{\sqrt{\sum_{j=1}^{p} \frac{(q_j - c_{ij})^2}{\sigma_{ij}^2}}} \quad (10)$$

In fact, the distance (9) is only computed for the actual data point, if it is lying outside the ranges of influence of all clusters, i.e. there exists no i such that

$$\sum_{j=1}^{p} \frac{(q_j - c_{ij})^2}{\sigma_{ij}^2} \leq 1 \quad (11)$$

Otherwise, the usual distance strategy is applied for all clusters fulfilling (11).

This leads us to the modified version of vector quantization in incremental mode:

Algorithm 2. VQ-INC-MOD

1. Initialize the number of clusters to 0
2. Take the next incoming data point $\mathbf{x}$
3. **If** the number of clusters is 0
   (a) Set $i = 1$, set the first center c_1 to the actual data point, hence $\mathbf{c}_1 = \mathbf{x}$, set $\sigma_1 = \epsilon$, ϵ a vector of small numbers > 0
   (b) goto step 8
4. **If** condition (11) is fulfilled for at least one i
   (a) Calculate the distance of the selected data point to all those cluster centers fulfilling (11) by using Euclidian distance measure.
   (b) Elicit the cluster center which is closest to the data point by taking the minimum over all calculated distances $\rightarrow$ winner cluster c_{win}
   (c) Set $\min_{dist} = 0$
5. **Else If** condition (11) is not fulfilled for any i
   (a) Calculate (9) for all clusters
   (b) Elicit the cluster center which is closest to the data point by taking the minimum over all calculated distances $\rightarrow$ winner cluster c_{win}
   (c) Set min_{dist} as the minimum over all distances
6. **If** $min_{dist} \geq \rho$ and $\mathbf{x}$ is not faulty
   (a) Set $i = i + 1$
   (b) $\mathbf{c}_i = \mathbf{x}$, $\sigma_i = \mathbf{0}$
7. **Else** Update the p components of the winner cluster by moving it towards the selected point $\mathbf{x}$ as in (6), update the variance in each direction by using (8).
8. If new incoming data points are still available goto step 2, otherwise stop

Figure 1 demonstrates the impact of this modified version of vector quantization. While Algorithm 1 generates new clusters for data points lying near the range of influence of another cluster (left image), the modified version performs better and extends the range of

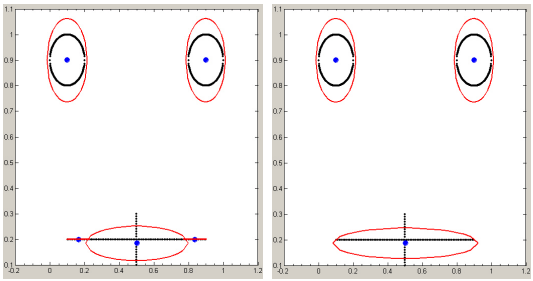

Figure 1: Left Image: clustering obtained by Algorithm 1 → 'over-clustering' (three clusters are generated instead of one correct one for the large data cloud at the bottom); right image: clustering obtained by its modified version with new distance strategy (Algorithm 2) → clusters are correct

influence of the nearby lying cluster (right image). The cluster centers are visualized as big dark dots.

4 Connecting Rule Antecedent Learning with Rule Consequent Learning

Principally two different approaches for learning linear rule consequents can be applied [22]:

- Global Learning: all C normalized membership functions (each one belonging exactly to one rule) are joined together in one regression matrix. Hence, the complete parameter vector $\mathbf{w}$ representing all linear consequent parameters in all rules is optimized.
- Local Learning: Each rule is treated separately and the corresponding linear consequent parameters are optimized in a weighted least squares approach, where the weights contain the normalized membership functions.

A local approach is necessary for a complete incremental learning of Takagi-Sugeno fuzzy systems, as it yields a higher flexibility for adjoining rules when new operating conditions for the measurement process occur. This is because for each rule an independent linear consequent parameter estimation is carried out (which is not the case for global learning). Furthermore, local learning possesses some other advantages compared to global learning such as numerical stability (as dealing with inversion of smaller matrices), computational performance and transparency of the consequent functions (hyper-planes). The latter aspect is based on an observed snuggling of the linear hyper-planes along the approximating surface [14].

The adaptive formulation of local learning results in well-known recursive weighted least squares [3,13], which calculates a new update for the linear parameters $\mathbf{w}$ each time a new data point comes in. It is remarkable, that it is a recursive formulation of weighted least squares, meaning that for each time instant the algorithm leads to the optimum in the least squares sense within each iteration step.

$$\begin{aligned}\hat{\mathbf{w}}_\mathbf{i}(k+1) =& \hat{\mathbf{w}}_\mathbf{i}(k) + \gamma(k)(y(k+1) \\ & - \mathbf{r}^T(k+1)\hat{\mathbf{w}}_\mathbf{i}(k))\end{aligned} \tag{12}$$

$$\gamma(k) = \frac{P_i(k)\mathbf{r}(k+1)}{\frac{1}{\Psi_i(\mathbf{x}(k+1))} + \mathbf{r}^T(k+1)P_i(k)\mathbf{r}(k+1)} \tag{13}$$

$$P_i(k+1) = (I - \gamma(k)\mathbf{r}^T(k+1))P_i(k) \tag{14}$$

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

This recursive learning of linear consequent parameters is connected with the antecedent parameter and rule learning strategy in Section 3. An obvious and straightforward way for doing so would be, either first to update the antecedent part and then second to adapt the consequent parameters of that rule corresponding to the winning cluster, or vice versa.

However, we want the incremental learning variant to be as close as possible to the solution obtained by the batch learning case, as this leads to the optimal solution in the least squares sense. Indeed, *recursive weighted least squares* applied to the linear consequent parameters of each rule separately as in (12),

(13) and (14) possesses the property of convergence within each iteration step. However, a change of fuzzy sets in the antecedent parts and furthermore in the normalized membership functions (2) makes the prior estimated linear consequent parameters always non-optimal for this changed fuzzy model (as these were optimized for the old fuzzy set positions). Hence, we introduce correction terms for the linear parameter update in order to balance out this non-optimal situation towards the optimal one with respect to the degree of change in the antecedent part. In this sense, before updating rule consequents, they are added to linear parameters as well as to the inverse Hessian matrix P.

The incremental learning of fuzzy basis function networks with a modified version of vector quantization becomes:

Algorithm 3. FLEXFIS-MOD

1. Collect k data points sufficient for the actual dimensionality of the problem
2. From these collected k data points generate an initial fuzzy model by using VQ-INC-MOD and local learning of rules consequents in batch mode (with least squares)
3. Take the next incoming data point $\mathbf{x}$
4. Normalize cluster centers and widths as well as the current data point due to the ranges from the previous cycle. This has the effect that new incoming (fault-free) data points lying significantly outside the already estimated range cause certainly a new rule.
5. Perform steps 4, 5 and 6 of Algorithm 2
6. **If** the condition in step 6 of Algorithm 2 is fulfilled, set a new rule: $C = C + 1$, project cluster to the axes to obtain the Gaussian fuzzy sets, set the linear consequent parameters $\hat{\mathbf{w}}_C$ of the new rule C to $\mathbf{0}$, set the inverse Hessian matrix $(R_C^T Q_C R_C)^{-1}$ of the new rule to αI.
7. **Else** Perform step 7 in Algorithm 2, project modified clusters onto the axes $\rightarrow$ modified sets; correct the linear parameter vector of consequent functions and the inverse Hessian matrix of the rule corresponding to the winning cluster by

$$\hat{\mathbf{w}}_{\text{win}}(k) = \hat{\mathbf{w}}_{\text{win}}(k) + \delta_{\text{win}}(k)$$
$$P_{\text{win}}(k) = P_{\text{win}}(k) + \Delta_{\text{win}}(k)$$

8. Update the ranges of all input and output variables
9. Perform recursive weighted least squares as in (12) to (14) for all C rules, achieving parameter vectors $\hat{\mathbf{w}}_i(k+1)$ and inverse Hessian matrices $P_i(k+1)$ for all i rules.
10. If new incoming data points are still available set $k = k + 1$ goto step 3, otherwise stop

The correction terms $\delta_{\text{win}}(k)$ and $\Delta_{\text{win}}(k)$ cannot be calculated explicitly within each incremental learning step without using prior loaded data points. However, a bound to these correction terms can be achieved, ensuring an almost-optimality (close to the optimality) in the least squares sense, when the correction terms are simply set to 0. The key factors of this achievement are the decreasing learning gain η (see (7)) and the bounded change of the rules' antecedents in each incremental learning step (because of condition in Step 6 of Algorithm 2).

5 Evaluation

In order to demonstrate practical feasibility, FLEXFIS-MOD is applied for generating high-dimensional fuzzy models from measurement data coming from engine test benches. Real faults were simulated at the test bench while recording the data. These faults should be detected by calculating the deviation of actual measurements to the online trained fuzzy models as done in [16]. There, it was pointed out that the fault detection performance of data-driven models is directly related to their prediction performance. Hence, the detection rate, i.e. the number of correctly detected faults is stated as quality measure in Table 1. These values were achieved by setting the

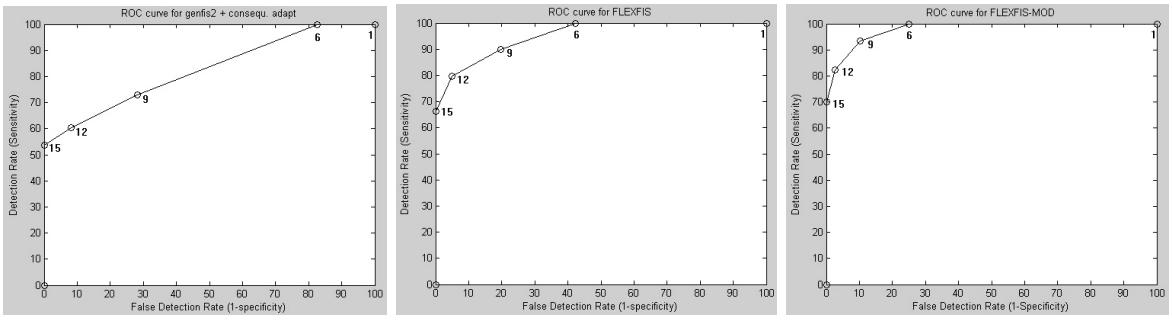

Figure 2: ROC curves for genfis2 with consequent adaptation (left), *FLEXFIS* (middle) and *FLEXFIS-MOD* (right)

Table 1: Comparison of incremental training variants for fuzzy basis function networks

Method	**Comp.**	**Det. Rate**
genfis2 + consequ. adapt	9.43	53.75% / 50%
FLEXFIS [15]	8.24	66.25% / 75%
FLEXFIS-MOD	7.12	70% / 75%

threshold in a way, such that no false detections occurred. A zero false detection rate is quite often a requirement in industrial systems, as with a high number of false alarms the confidence of the operators in the software gets weakened, until finally the fault detection component is ignored. Furthermore, for comparing the methods with respect to specificity vs. sensitivity, the ROC curve is shown in Figure 2. Note that the closer the curve follows the left-hand border and then the top border of the ROC space, the better the fault detection performance of the method. In sum, 56 up to five-dimensional reasonable models could be extracted automatically from the data with the help of variable selection methods [9], where 70 channels were measured in sum. This gives a good coverage of channels, when taking into account, that some of the remaining 14 appear at the input side of the models.

The table and the ROC curve are shown for the online case, where fuzzy models have to be adaptively trained with new incoming data. This was accomplished in an online simulation framework, where the data was loaded sample-wise into the memory. A high-frequented re-building with conventional batch modelling methods of all the 56 models was not possible, as this slowed down the whole process significantly, such that real time performance could not be achieved. Therefore, in the case of genfis2 [20] (as batch mode modeling variant) an adaptation of linear rule consequent parameter alone (with recursive least squares as stated in Section 4) was tried, as it is often proposed in literature, e.g. [4,7]. From Table 1 and Figure 2 it can be recognized that this incremental fuzzy model training option cannot compete with both, *FLEXFIS* and *FLEXFIS-MOD*. This is quite intuitive as different operating conditions at the engine test bench occurred, for which the models needs to be extended and flexibly adapted. This cannot be achieved with adaptations of the linear hyper-planes alone. The complexity is measured by the average number of rules over the 56 fuzzy models and stated in Table 1. In this table the detection rate and the false detection rate are measured on two bases: the first number corresponds to the measurement basis i.e. all measurements affected by faults are counted (in sum 80), the second one corresponds to the fault bases, i.e. all different kind of faults are counted (in sum eight). The ROC curves in Figure 2 only show the rates on the measurement basis.

6 Conclusion

A new method for incremental learning of fuzzy systems was demonstrated. It exploits a modified variant of vector quantization and recursive learning of linear rule consequent parameters. A special attention is given to stability and process safety when combining these two approaches. The evaluation in Section 5 underlines the applicability of the method for online identification and fault detection tasks. Promising future directions will include process safety enhancements with respect to extrapolation behavior, a concept for omitting the incorporation of faulty occasions and calculating error bars incrementally for improved fault detection.

Acknowledgements

Ulrich Bodenhofer gratefully acknowledges support by the Austrian Government, the State of Upper Austria, and the Johannes Kepler University Linz in the framework of the K*plus* Competence Center Program.

References

[1] K. Alsabti, S. Ranka, and V. Singh. An efficient k-means clustering algorithm. In *IPPS: 11th International Parallel Processing Symposium*. IEEE Computer Society Press, 1998.

[2] P.P. Angelov and D.P. Filev. Flexible models with evolving structure. *International Journal of Intelligent Systems*, 19(4):327–340, 2004.

[3] K.J. Aström and B. Wittenmark. *Adaptive Control - Second Edition*. Addison-Wesley ISBN-0-201-55866-1, 1995.

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

[5] J. Beringer and E. Hüllermeier. Online clustering of parallel data streams. *To appear in Data & Knowledge Engineering*, 2006.

[6] G. A. Carpenter and S. Grossberg. Adaptive resonance theory (art). In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 79–82. MIT Press, Cambridge, MA, 1995.

[7] A. Fink, M. Fischer, O. Nelles, and R. Isermann. Supervision of nonlinear adaptive controllers based on fuzzy models. *Journal of Control Engineering Practice*, 8:1093–1105, 2000.

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

[9] W. Groißböck, E. Lughofer, and E.P. Klement. A comparison of variable selection methods with the main focus on orthogonalization. In M. Lopéz-Díaz, M.Á. Gil, P. Grzegorzewski, O. Hryniewicz, and J. Lawry, editors, *Soft Methodology and Random Information Systems*, Advances in Soft Computing, pages 479–486. Springer, Berlin, Heidelberg, New York, 2004.

[10] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2/3):107–145, 2001.

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

[12] N. K. Kasabov and Q. Song. DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Trans. on Fuzzy Systems*, 10(2):144–154, 2002.

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

[14] E. Lughofer, E. Hüllermeier, and E.P. Klement. Improving the interpretability of data-driven evolving fuzzy systems. In *Proceedings of EUSFLAT 2005*, pages 28–33, Barcelona, Spain, 2005.

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

[16] E. Lughofer, E.P. Klement, J.M. Lujan, and C. Guardiola. Model-based fault detection in multi-sensor measurement systems. In *Proceedings of IEEE IS 2004*, pages 184–189, Varna, Bulgaria, 2004.

[17] O. Nelles. *Nonlinear System Identification*. Springer Verlag Berlin, Germany, 2001.

[18] S.J. Qin, W. Li, and H.H. Yue. Recursive PCA for adaptive process monitoring. *Journal of Process Control*, 10:471–486, 2000.

[19] R. Yager and D. Filev. Approximate clustering via the mountain method. *IEEE Trans. on Systems and Cybernetics*, 24(8), 1994.

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

[21] M.S. Yang and K.L. Wu. A new validity index for fuzzy clustering. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, pages 89–92, Melbourne, Australia, 2001.

[22] 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.