Chapter 3

Evolving Fuzzy Systems — Fundamentals, Reliability, Interpretability, Useability, Applications

Edwin Lughofer

This chapter provides a round picture of the development and advances in the field of *evolving fuzzy systems* (EFS) made during the last decade since their first appearance in 2002. Their basic difference to conventional fuzzy systems (discussed in other chapters in this book) is that they can be learned from data on-the-fly (fast) during on-line processes in an incremental and mostly single-pass manner. Therefore, they stand for a very emerging topic in the field of soft computing for addressing modeling problems in the quickly increasing complexity of real-world applications, more and more implying a shift from batch off-line model design phases (as conducted since the 80s) to permanent on-line (active) model teaching and adaptation. The focus will be placed on the definition of various model architectures used in the context of EFS, on providing an overview about the basic learning concepts, on listing the most prominent EFS approaches (fundamentals), and on discussing advanced aspects toward an improved stability, reliability and useability (usually must-to-haves to guarantee robustness and user-friendliness) as well as an educated interpretability (usually a nice-to-have to offer insights into the systems' nature). It will be concluded with a list of real-world applications where various EFS approaches have been successfully applied with satisfactory accuracy, robustness and speed.

3.1. Introduction — Motivation

Due to the increasing complexity and permanent growth of data acquisition sites, in today's industrial systems there is an increasing demand of fast modeling algorithms from on-line data streams (Gama, 2010). Such algorithms are ensuring that models can be quickly adapted to the actual system situation and thus are able to provide reliable outputs at any time during on-line real-world processes. There, changing operating conditions, environmental influences and new unexplored system states may trigger a quite dynamic behavior, causing previously trained models to become inefficient or even inaccurate (Sayed-Mouchaweh and Lughofer, 2012). In this sense, conventional static models which are trained once in an off-line stage and are not able to adapt dynamically to the actual system states are not an adequate alternative for coping with these demands (severe downtrends in accuracy have been examined

in previous publications). A list of potential real-world application examples relying on on-line dynamic aspects and thus demanding flexible modeling techniques can be found in Table 3.3 (Section 3.6.5).

Another challenge which has recently become a very hot topic within the machine learning community and is given a specific attention in the new European framework programme Horizon 2020, is the processing and mining of so-called *Big Data*,[1] usually stemming from very large data bases (VLDB).[2] The occurrence of *Big Data* takes place in many areas such as meteorology, genomics, connectomics, complex physics simulations and biological and environmental research (Reichmann *et al.*, 2011). This data is that big (exabytes) such that it cannot be handled in a one-shot experience, such as exceeding virtual memory of nowadays conventional computers. Thus, standard batch modeling techniques are not applicable.

In order to tackle the aforementioned requirements, the field of "evolving intelligent systems (EIS)"[3] or, in a wider machine learning sense, the field of "learning in dynamic environments (LDE)" enjoyed an increasing attraction during the last years (Angelov *et al.*, 2010). This even lead to the emergence of their own journal in 2010, termed as "Evolving Systems" at Springer (Heidelberg).[4] Both fields support learning topologies which operate in single-pass manner and are able to update models and surrogate statistics on-the-fly and on demand. Single-pass nature and incrementality of the updates assure on-line and in most cases even real-time learning and model training capabilities. While EIS focus mainly on adaptive evolving models within the field of soft computing, LDE goes a step further and also joins incremental machine learning and data mining techniques, originally stemming from the area of "incremental heuristic search".[5] The update in these approaches concerns both, parameter adaptation and structural changes, depending on the degree of change required. The structural changes are usually enforced by evolution and pruning components, and finally responsible for the terminus *Evolving Systems*. In this context, *Evolving* should be not confused with *Evolutionary* (as sometimes happened in the past, unfortunately). Evolutionary approaches are usually applied in the context of complex optimization problems and learn parameters and structures based on genetic operators, but they do this by using all the data in an iterative optimization procedure rather than integrating new knowledge permanently on-the-fly.

[1] http://en.wikipedia.org/wiki/Big_data.
[2] http://en.wikipedia.org/wiki/Very_large_database.
[3] http://en.wikipedia.org/wiki/Evolving_intelligent_system.
[4] http://www.springer.com/physics/complexity/journal/12530.
[5] http://en.wikipedia.org/wiki/Incremental_heuristic_search.

Apart from the requirements and demands in industrial (production and control) systems, another important aspect about evolving models is that they provide the opportunity for self-learning computer systems and machines. In fact, evolving models are permanently updating their knowledge and understanding about diverse complex relationships and dependencies in real-world application scenarios by integrating new system behaviors and environmental influences (which are manifested in the captured data). Their learning follows a life-long learning context and is never really terminated, but lasts as long as new information arrives. Therefore, they can be seen as a valuable contribution within the field of computational intelligence (Angelov and Kasabov, 2005) or even in artificial intelligence (Lughofer, 2011a).

There are several possibilities for using an adequate model architecture within the context of an evolving system. This strongly depends on the learning problem at hand, in particular whether it is supervised or unsupervised. In case of the later, techniques from the field of clustering, usually termed as *incremental clustering* (Bouchachia, 2011) are a prominent choice. In case of classification and regression models, the architectures should support decision boundaries respectively approximation surfaces with an arbitrary non-linearity degree. Also, the choice may depend on past experience with some machine learning and data mining tools: for instance, it is well-known that SVMs are usually among the top-10 performers for many classification tasks (Wu *et al.*, 2006), thus are a reasonable choice to be used in an on-line classification setting as well [in form of incremental SVMs (Diehl and Cauwenberghs, 2003; Shilton *et al.*, 2005)]; whereas in a regression setting they are usually performing much weaker. Soft computing models such as neural networks (Haykin, 1999), fuzzy systems (Pedrycz and Gomide, 2007) or genetic programming (Affenzeller *et al.*, 2009) and any hybrid concepts of these [e.g., neuro-fuzzy systems (Jang, 1993)] are all known to be universal approximators (Balas *et al.*, 2009) and thus able to resolve non-linearities implicitly contained in the systems' behavior (and thus reflected in the data streams). Neural networks suffer from their black box nature, i.e., not allowing operators and users any insight into the models extracted from the streams. This may be essential in many context for the interpretation of model outputs to realize why certain decisions have been made etc. Genetic programming are a more promising choice in this direction, however they are often expanding unnecessarily complex formulas with many nested functional terms [suffering from the so-called bloating effect (Zavoianu, 2010)], which are again hard to interpret.

Fuzzy systems are specific mathematical models which build upon the concept of fuzzy logic, firstly introduced in 1965 by Lotfi A. Zadeh (Zadeh, 1965), are a very useful alternative, as they contain rules which are linguistically readable and interpretable. This mimicks the human thinking about relationships and structural

dependencies being present in a system. This will become more clear in the subsequent section when mathematically defining possible architecture variants within the field of fuzzy systems, which have been also used in the context of data stream mining and evolving systems. Furthermore, the reader may refer to Chapter 1 in this book, where the basic concepts of fuzzy sets and systems are introduced and described in detail.

3.2. Architectures for EFS

The first five subsections are dedicated to architectures for regression problems, for which evolving fuzzy systems (EFS) have been preliminary used. Then, various variants of fuzzy classification model structures are discussed, as have been recently introduced for representing of decision boundaries in various forms in evolving fuzzy classifiers (EFC).

3.2.1. *Mamdani*

Mamdani fuzzy systems (Mamdani, 1977) are the most common choice for coding expert knowledge/experience into a rule-based IF-THEN form, examples can be found in Holmblad and Ostergaard (1982); Leondes (1998) or Carr and Tah (2001); Reveiz and Len (2010).

In general, assuming p input variables (features), the definition of the ith rule in a single output Mamdani fuzzy system is as follows:

$$\text{Rule}_i : \text{IF } (x_1 \text{ IS } \mu_{i1}) \text{ AND} \ldots \text{AND } (x_p \text{ IS } \mu_{ip}) \text{ THEN } l_i(\vec{x}) \text{ IS } \Phi_i,$$

with Φ_i the consequent fuzzy set in the fuzzy partition of the output variable used in the consequent $l_i(\vec{x})$ of the ith rule, and $\mu_{i1}, \ldots, \mu_{ip}$ are the fuzzy sets appearing in the rule antecedents. The *rule firing degree* (also called *rule activation level*) for a concrete input vector $\vec{x} = (x_1, \ldots, x_p)$ is then defined by:

$$\mu_i(\vec{x}) = \mathop{\mathsf{T}}_{j=1}^{p} \mu_{ij}(x_j), \tag{1}$$

with T a specific conjunction operator, denoted as t-norm Klement *et al.* (2000) — most frequently, minimum or product are used, i.e.,

$$\mu_i(\vec{x}) = \min_{j=1}^{p}(\mu_{ij}(x_j)) \quad \mu_i(\vec{x}) = \prod_{j=1}^{p}(\mu_{ij}(x_j)). \tag{2}$$

It may happen, that $\Phi_i = \Phi_j$ for some $i \neq j$. Hence, a t-conorm (Klement *et al.*, 2000) is applied which combines the rule firing levels of those rules having the same consequents to one output set. The most common choice for the t-conorm is

the maximum operator. In this case, the consequent fuzzy set is cut at the alpha-level:

$$\alpha_i = max_{j_i=1,\ldots,C_i}(\mu_{j_i}(\vec{x})), \tag{3}$$

with C_i the number of rules whose consequent fuzzy set is the same as for Rule i, and j_i the indices of these rules. This is done for the whole fuzzy rule base and the various α-cut output sets are joined to one fuzzy area employing the *supremum operator*. An example of such a fuzzy output area is shown in Figure 3.1.

In order to obtain a crisp output value, a defuzzification method is applied, most commonly used are the mean of maximum (MOM) over the whole area, the center of gravity (COG) or the bisector (Leekwijck and Kerre, 1999) which is the vertical line that will divide the whole area into two sub-regions of equal areas. For concrete formulas of the defuzzification operators, please refer to Piegat (2001) and Nguyen *et al.* (1995). MOM and COG are exemplarily shown in Figure 3.1.

Due to the defuzzification process, it is quite intuitive that the inference in Mamdani fuzzy systems looses some accuracy, as an output fuzzy number is reduced to a crisp number. Therefore, they have been hardly applied within the context of on-line modeling and data stream mining, where the main purpose is to obtain an accurate evolving fuzzy model in the context of *precise evolving fuzzy modeling* [an exception can be found in Rubio (2009), termed as the SOFMLS approach]. On the other hand, they are able to provide linguistic interpretability on the output level, thus may be preferable in the context of interpretability demands for knowledge gaining and reasoning (see also Section 3.5). The approach in Ho *et al.*, 2010, tries

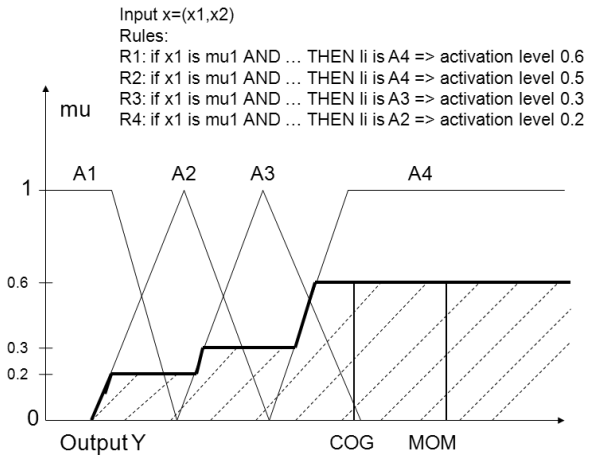


Figure 3.1: Mean of maximum (MOM) and center of gravity (COG) defuzzification for a rule consequent partition (fuzzy partition in output variable Y) in a Mamdani fuzzy system, the shaded area indicates the joint consequents (fuzzy sets) in the active rules (applying supremum operator); the cutoff points (alpha cuts) are according to maximal membership degrees obtained from the rule antecedent parts of the active rules.

to benefit from this while keeping the precise modeling spirit by applying a switched architecture, joining Mamdani and Takagi–Sugeno type consequents in form of a convex combination.

3.2.2. *Takagi–Sugeno*

Opposed to Mamdani fuzzy systems, Takagi–Sugeno (TS) fuzzy systems (Takagi and Sugeno, 1985) are the most common architectorial choice in evolving fuzzy systems approaches (Lughofer, 2011b). This has several reasons. First of all, they enjoy a large attraction in many fields of real-world applications and systems engineering (Pedrycz and Gomide, 2007), ranging from process control (Babuska, 1998; Karer and Skrjanc, 2013; Piegat, 2001), system identification (Abonyi, 2003; Nelles, 2001), through condition monitoring (Serdio *et al.*, 2014a, 2014b) and chemometric calibration (Cernuda *et al.*, 2013; Skrjanc, 2009) to machine vision and texture processing approaches (Lughofer, 2011b; Riaz and Ghafoor, 2013). Thus, their robustness and applicability for the standard batch modeling case has been already proven since several decades. Second, they are known to be universal approximators (Castro and Delgado, 1996), i.e., being able to model any implicitly contained non-linearity with a sufficient degree of accuracy, while their interpretable capabilities are still intact or may offer even advantages: while the antecedent parts remain linguistic, the consequent parts can be interpreted either in a more physical sense (see Bikdash, 1999; Herrera *et al.*, 2005) or as local functional tendencies (Lughofer, 2013) (see also Section 3.5). Finally, parts of their architecture (the consequents) can be updated exactly by recursive procedures, as will be described in Section 3.3.1. This is a strong point as they are converging to the same solution as when (hypothetically) sending all data samples at once into the optimization process (*true optimal incremental solutions*).

3.2.2.1. *Takagi–Sugeno standard*

A single rule in a (single output) standard TS fuzzy system is of the form

$$\text{Rule } i : \text{IF} \quad (x_1 \text{ IS } \mu_{i1}) \quad \text{AND} \ \ldots \ \text{AND} \quad (x_p \text{ IS } \mu_{ip}) \tag{4}$$

$$\text{THEN} \quad l_i(\vec{x}) = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + \cdots + w_{ip}x_p \tag{5}$$

where $\vec{x} = (x_1, \ldots, x_p)$ is the p-dimensional input vector and μ_{ij} the fuzzy set describing the j-th antecedent of the rule. Typically, these fuzzy sets are associated with a linguistic label. As in case of Mamdani fuzzy systems, the AND connective is modeled in terms of a t-norm, i.e., a generalized logical conjunction (Klement *et al.*, 2000). Again, the output $l_i = l_i(\vec{x})$ is the so-called consequent function of the rule.

The output of a TS system consisting of C rules is a linear combination of the outputs produced by the individual rules (through the l_i's), where the contribution of each rule is given by its normalized degree of activation Ψ_i, thus:

$$\hat{f}(\vec{x}) = \hat{y} = \sum_{i=1}^{C} \Psi_i(\vec{x}) \cdot l_i(\vec{x}) \quad \text{with } \Psi_i(\vec{x}) = \frac{\mu_i(\vec{x})}{\sum_{j=1}^{C} \mu_j(\vec{x})}, \tag{6}$$

with $\mu_i(\vec{x})$ as in Equation (1). From statistical point of view, a TS fuzzy model can be interpreted as a collection of piecewise local linear predictors by a smooth (normalized) kernel, thus in its local parts (rules) having some synergies with local weighted regression (LWR) (Cleveland and Devlin, 1988). The difference is that in LWR the model is extracted on demand based on the nearest data samples [also termed as the reference base in an instance-based learning context for data streams (Shaker and Hüllermeier, 2012)] while TS fuzzy systems are providing a global model defined over the whole feature space (thus preferable in the context of interpretability issues and on-line prediction speed).

The most convenient choice for fuzzy sets in EFS and fuzzy systems design in general are Gaussian functions, which lead to so-called *fuzzy basis function networks* (Wang and Mendel, 1992) and multi-variate kernels following normal distributions are achieved for representing the rules' antecedent parts:

$$\mu_i(\vec{x}) = \prod_{i=1}^{p} exp\left(-\frac{1}{2}\frac{(x_i - c_i)^2}{\sigma_i^2}\right). \tag{7}$$

In this sense, the linear hyper-planes l_i are connected with multi-variate Gaussians to form an overall smooth function. Then, the output form in Equation (6) becomes some synergies with Gaussian mixture models (GMMs) (Day, 1969; Sun and Wang, 2011), often used for clustering and pattern recognition tasks (Bishop, 2007; Duda *et al.*, 2000). The difference is that l_i's are hyper-planes instead of singleton weights and do not reflect the degree of density of the corresponding rules (as *mixing proportion*), but the linear trend of the approximation/regression surface in the corresponding local parts.

3.2.2.2. *Takagi–Sugeno generalized*

Recently, the generalized form of Takagi–Sugeno fuzzy systems has been offered to the evolving fuzzy systems community, launching its origin in Lemos *et al.* (2011a) and Leite *et al.* (2012a); latter explored and further developed in Pratama *et al.* (2014a) and Pratama *et al.* (2014b). The basic principle is that it employs multidimensional normal (Gaussian) distributions in arbitrary position for representing single rules. Thus, it overcomes the deficiency not being able to model local correlations between input and output variables appropriately, as is the case with the

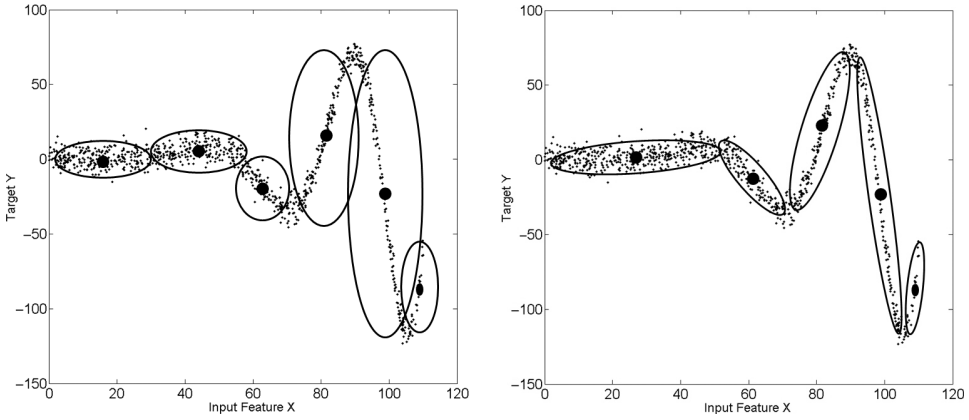


Figure 3.2: Left: Conventional axis parallel rules (represented by ellipsoids) achieve an inaccurate representation of the local trends (correlations) of a non-linear approximation problem (defined by noisy data samples). Right: Generalized rules (by rotation) achieve a much more accurate representation.

t-norm operator used in standard rules (Klement *et al.*, 2000) — these may represent inexact approximations of the real local trends and finally causing information loss in rules (Abonyi *et al.*, 2002).

An example for visualizing this problematic nature is provided in Figure 3.2: in the left image, axis-parallel rules (represented by ellipsoids) are used for modeling the partial tendencies of the regression curves which are not following the input axis direction, but are rotated to some degree; obviously, the volume of the rules are artificially blown-up and the rules do not represent the real characteristics of the local tendencies well → *information loss*. In the right image, non axis-parallel rules using general multivariate Gaussians are applied for a more accurate representation (rotated ellipsoids).

To avoid such information loss, the generalized fuzzy rules have been defined in Lemos *et al.*, 2011a (there used for evolving stream mining), as

$$\text{IF } \vec{x} \text{ IS (about)} \Psi_i \text{ THEN } l_i(\vec{x}) = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + \cdots + w_{ip}x_p, \quad (8)$$

where Ψ denotes a high-dimensional kernel function, which in accordance to the basis function networks spirit are given by the generalized multivariate Gaussian distribution:

$$\Psi_i(\vec{x}) = exp\left(-\frac{1}{2}(\vec{x} - \vec{c}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{c}_i)\right), \quad (9)$$

with $\vec{c}_i$ the center and Σ_i^{-1} the inverse covariance matrix of the ith rule, allowing any possible rotation and spread of the rule. It is also known in the neural network literature that Gaussian radial basis functions are a nice option to characterize local

properties (Lemos *et al.*, 2011a; Lippmann, 1991); especially, someone may inspect the inner core part, i.e., all samples fulfilling $(\vec{x} - \vec{c}_i)^T \Sigma_i^{-1} (\vec{x} - \vec{c}_i) \leq 1$, as the characteristic contour/spread of the rule.

The fuzzy inference then becomes a linear combination of multivariate Gaussian distributions in the form:

$$\hat{y} = \frac{\sum_{i=1}^{C} l_i(\vec{x}) * exp\left(-\frac{1}{2}(\vec{x} - \vec{c}_i)^T \Sigma_i^{-1} (\vec{x} - \vec{c}_i)\right)}{\sum_{i=1}^{C} exp\left(-\frac{1}{2}(\vec{x} - \vec{c}_i)^T \Sigma_i^{-1} (\vec{x} - \vec{c}_i)\right)} = \sum_{i=1}^{C} l_i(\vec{x}) \Phi_i(\vec{x}), \tag{10}$$

with C the number of rules, $l_i(\vec{x})$ the consequent hyper-plane of the ith rule and Φ_i the normalized membership degrees, summing up to 1 for each query sample.

In order to maintain (input/output) interpretability of the evolved TS fuzzy models for users/operators (see also Section 3.5), the authors in Lughofer *et al.* (2013) foresee a projection concept to form fuzzy sets and classical rule antecedents. It relies on the angle between the principal components directions and the feature axes, which has the effect that long spread rules are more effectively projected than when using the inner contour spreads (through axis parallel cutting points). The spread σ_i of the projected fuzzy set is set according to:

$$\sigma_i = max_{j=1,\ldots p} \left(\frac{r}{\sqrt{\lambda_j}} cos(\Phi(e_i, a_j)) \right), \tag{11}$$

with r the range of influence of one rule, usually set to 1, representing the (inner) characteristic contour/spread of the rule (as mentioned above). The center of the fuzzy set in the ith dimension is set equal to the ith coordinate of the rule center. $\Phi(e_i, a_j)$ denotes the angle between principal component direction (eigenvector a_j) and the ith axis e_i, λ_j the eigenvalue of the jth principal component.

3.2.2.3. *Takagi–Sugeno extended*

An extended version of Takagi–Sugeno fuzzy systems in the context of evolving systems has been applied in Komijani *et al.* (2012). There, instead of a hyper-plane $l_i = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + \cdots + w_{ip}x_p$ the consequent function for the ith rule is defined as LS_SVM model according to Smola and Schölkopf (2004):

$$l_i(\vec{x}) = \sum_{k=1}^{N} \alpha_{ik} K(\vec{x}, \vec{x}_k) + \beta_i, \tag{12}$$

with $K(.,.)$ a kernel function fulfilling the Mercer criterion (Mercer, 1909) for characterizing a symmetric positive semi-definite kernel (Zaanen, 1960), N the number of training samples and α and β the consequent parameters (support vectors and intercept) to learn. The l_i's can be in principle combined within in any inference scheme, either with the standard one in Equation (6) or with the generalized one in

Equation (10) [in Komijani *et al.* (2012), they are combined with Equation (6)]. The advantage of these consequents are that they are supposed to provide more accuracy, as a support vector regression modeling (Smola and Schölkopf, 2004) is applied to each local region. Hence, non-linearities within local regions may be better resolved. On the other hand, the consequents are more difficult to interpret.

3.2.3. *Type-2*

Type-2 fuzzy systems were invented by Lotfi Zadeh in 1975 (Zadeh, 1975) for the purpose of modelling the uncertainty in the membership functions of usual (type-1) fuzzy sets. The distinguishing feature of a type-2 fuzzy set $\tilde{\mu}_{ij}$ versus its type-1 counterpart μ_{ij} is that the membership function values of $\tilde{\mu}_{ij}$ are blurred, i.e., they are no longer a single number in [0, 1], but are instead a continuous range of values between 0 and 1, say $[a, b] \subseteq [0, 1]$. One can either assign the same weighting or a variable weighting to membership function values in $[a, b]$. When the former is done, the resulting type-2 fuzzy set is called an interval type-2 fuzzy set. When the latter is done, the resulting type-2 fuzzy set is called a general type-2 fuzzy set (Mendel and John, 2002).

The ith rule of an interval-based type-2 fuzzy system is defined in the following way (Liang and Mendel, 2000; Mendel, 2001):

$$\text{Rule}_i : \text{IF } x_1 \text{ IS } \tilde{\mu}_{i1} \text{ AND} \ldots \text{AND } x_p \text{ IS } \tilde{\mu}_{ip} \text{ THEN } l_i(\vec{x}) = \tilde{f}_i,$$

with $\tilde{f}_i$ a general type-2 uncertainty function.

In case of a Takagi–Sugeno based consequent scheme (as e.g., used in Juang and Tsao (2008), the first approach of an evolving Type-2 fuzzy system), the consequent function becomes:

$$l_i(\vec{x}) = \tilde{w}_{i0} + \tilde{w}_{i1}x_1 + \tilde{w}_{i2}x_2 + \cdots + \tilde{w}_{ip}x_p, \tag{13}$$

with $\tilde{w}_{ij}$ an interval set (instead of a crisp continuous value), i.e.

$$\tilde{w}_{ij} = [c_{ij} - s_{ij}, c_{ij} + s_{ij}]. \tag{14}$$

In case of a Mamdani based consequent scheme (as e.g., used in Tung *et al.* (2013), a recent evolving approach), the consequent function becomes: $l_i = \tilde{\Phi}_i$, with $\tilde{\Phi}_i$ a type two fuzzy set.

An enhanced approach for eliciting the final output is applied, the so-called Karnik–Mendel iterative procedure (Karnik and Mendel, 2001), where a type reduction is performed before the defuzzification process. In this procedure, the consequent values $\bar{l}_i = c_{i0} - s_{i0} + (c_{i1} - s_{i1})x_1 + (c_{i2} - s_{i2})x_2 + \cdots + (c_{ip} - s_{ip})x_p$ and $\underline{l}_i = c_{i0} + s_{i0} + (c_{i1} + s_{i1})x_1 + (c_{i2} + s_{i2})x_2 + \cdots + (c_{ip} + s_{ip})x_p$ are sorted in ascending order denoted as $\bar{y}_i$ and $\underline{y}_i$ for all $i = 1, \ldots, C$. Accordingly, the

membership values $\bar{\Psi}_i(\vec{x})$ and $\underline{\Psi}_i(\vec{x})$ are sorted in ascending order denoted as $\bar{\psi}_i(\vec{x})$ and $\underline{\psi}_i(\vec{x})$. Then, the outputs $\bar{y}$ and $\underline{y}$ are computed by:

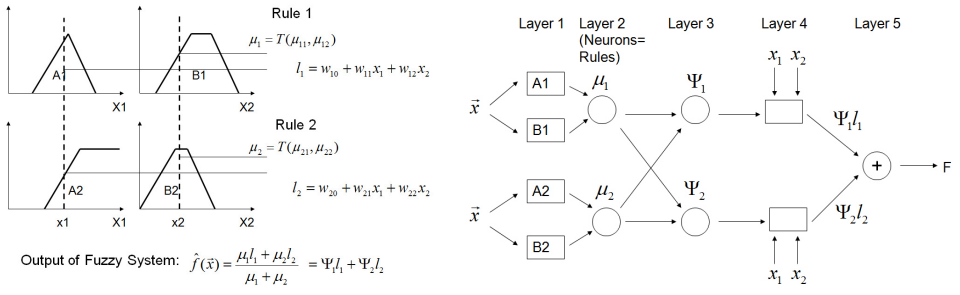
$$\bar{y} = \frac{\sum_{i=1}^{L} \bar{\psi}_i(\vec{x})\bar{y}_i + \sum_{i=L+1}^{C} \underline{\psi}_i(\vec{x})\bar{y}_i}{\sum_{i=1}^{L} \bar{\psi}_i(\vec{x}) + \sum_{i=L+1}^{C} \underline{\psi}_i(\vec{x})} \quad \underline{y} = \frac{\sum_{i=1}^{R} \underline{\psi}_i(\vec{x})\underline{y}_i + \sum_{i=R+1}^{C} \bar{\psi}_i(\vec{x})\underline{y}_i}{\sum_{i=1}^{R} \underline{\psi}_i(\vec{x}) + \sum_{i=R+1}^{C} \bar{\psi}_i(\vec{x})} \tag{15}$$

with L and R positive numbers, often $L = \frac{C}{2}$ and $R = \frac{C}{2}$. Taking the average of these two yields the final output value y.

3.2.4. *Neuro-Fuzzy*

Most of the neuro-fuzzy systems (Fuller, 1999) available in literature can be interpreted as a layered structural form of Takagi–Sugeno–Kang fuzzy systems. Typically, the fuzzy model is transformed into a neural network structure (by introducing layers, connections and weights between the layers) and learning methods already established in the neural network context are applied to the neuro-fuzzy system. A well-known example for this is the ANFIS approach (Jang, 1993), where the back-propagation algorithm (Werbos, 1974) is applied and the components of the fuzzy model (fuzzification, calculation of rule fulfillment degrees, normalization, defuzzification), represent different layers in the neural network structure. However, the inference scheme finally leads to the same model outputs as for conventional TS fuzzy systems. A visualization example is presented in Figure 3.3. This layered structure will be used by several EFS approaches as can be seen from Tables 3.1 and 3.2.

Recently, a new type of neuro-fuzzy architecture has been proposed by Silva *et al.* (2014), termed as neo-fuzzy neuron network, and applied in evolving context. It relies on the idea to use a set of TS fuzzy rules for each input dimension



(a) (b)

Figure 3.3: (a) Standard Takagi–Sugeno type fuzzy system, (b) Equivalent neural network structure.

independently and then connect these with a conventional sum for obtaining the final model output. The domain of each input i is granulated into m complementary membership functions.

3.2.5. *Hierarchical Structures*

Hierarchical architectures for evolving fuzzy modeling have been recently introduced in Shaker *et al.* (2013) and Lemos *et al.* (2011b). Both architectures have been designed for the purposes to provide a more slim, thus more transparent rule base by inducing rules with flexible lengths. This is opposed to all the flat model architectures which have been presented above, always using all input features in all rules' antecedent parts.

The first approach is an incremental extension of top-down induction of fuzzy pattern trees (Senge and Huellermeier, 2011) and thus uses a hierarchical tree-like concept to evolve nodes and leaves on demand. Thereby, a collection of fuzzy sets and aggregation operators can be specified by the user as allowed patterns and conjunction operators in the leaf nodes. In particular, a pattern tree has the outlook as shown in the example of Figure 3.4. Thereby, one basic difference to classical fuzzy systems is that the conjunction operator do not necessarily have to be t-norms [can be a more general aggregation operator (Saminger-Platz *et al.*, 2007)] and the type of the fuzzy sets can be different in different tree levels [as indicated in the rectangles in Figure 3.4 (left)], allowing a composition of a mixture of patterns in hierarchical form. Another difference is the possibility to obtain a single compact rule for describing a certain characteristics of the output (a good house price quality with 0.9 in the example in Figure 3.4).

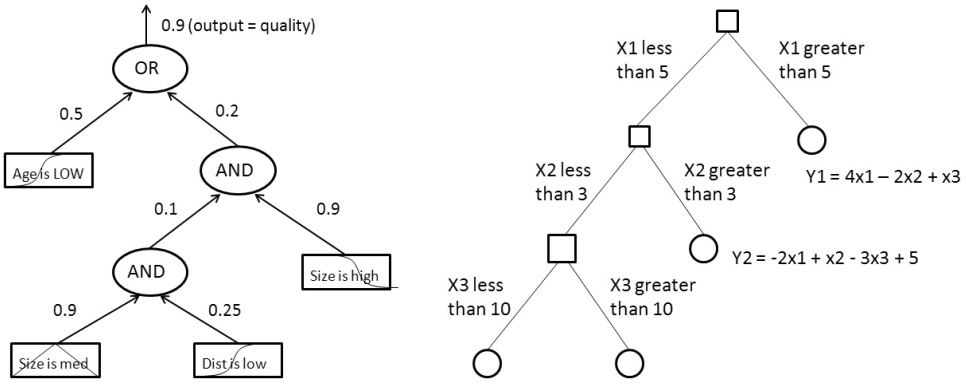


Figure 3.4: Left: Example of a fuzzy pattern tree which can be read as "IF ((Size is med AND Dist is high) AND Size is HIGH) OR Age is LOW THEN Output (Quality) is 0.9". Right: Example of a fuzzy decision tree with four rules, a rule example is "IF x1 is LESS THAN 5 AND x2 is GREATER THAN 3 THEN y2 $= -2$x1 $+$ x2 $-$ 3x3 $+$ 5".

The second one (Lemos *et al.*, 2011b) has some synergies to classical decision trees for classification tasks [CART (Breiman *et al.*, 1993) and C4.5 (Quinlan, 1993)], where, however, the leafs are not class labels, but linear hyper-planes as used in classical Takagi–Sugeno fuzzy systems. Thus, as the partitioning may be arbitrarily fine-granuled as is the case for classical TS fuzzy systems, they still enjoy the favorable properties of being universal approximators. A visual example of such a tree is shown in the right image in Figure 3.4. It is notable that the nodes do not contain crisp decision rules, but fuzzy terms "Smaller Than" and "Greater Than", which are represented by sigmoidal fuzzy sets: e.g., "Less Than 5" is a fuzzy set which cuts the fuzzy set "Greater than 5" at $x = 5$ with a membership degree 0.5. One path from the root node to a terminal node represents a rule, which is then similar to a classical TS fuzzy rule, but allowing an arbitrary length.

3.2.6. *Classifiers*

Fuzzy classifiers have been enjoyed a wide attraction in various applications since almost two decades (Eitzinger *et al.*, 2010; Kuncheva, 2000; Nakashima *et al.*, 2006). Their particular strength is the ability to model decision boundaries with arbitrary non-linearity degree while maintaining interpretability in the sense "which rules on the feature set imply which output class labels (classifier decisions)". In a winner-takes-it-all concept, the decision boundary proceeds between rule pairs having different majority class labels. As rules are usually non-linear contours in the high-dimensional space, the non-linearity of the decision boundary is induced — enjoying arbitrary complexity due to a possible arbitrary number of rules. If rules have linear contours, then overall non-linearity is induced in form of piecewise linear boundaries between rule pairs.

3.2.6.1. *Classical and extended single-model*

The rule in a classical fuzzy classification model architecture with singleton consequent labels is a widely studied architecture in the fuzzy systems community (Ishibuchi and Nakashima, 2001; Kruse *et al.*, 1994; Kuncheva, 2000; Nauck and Kruse, 1998) and is defined by:

$$\text{Rule}_i : \text{IF } x_1 \text{ IS } \mu_{i1} \text{ AND} \ldots \text{AND } x_p \text{ IS } \mu_{ip} \text{ THEN } l_i = L_i \tag{16}$$

where L_i is the crisp output class label from the set $\{1, \ldots, K\}$ with K the number of classes for the ith rule. This architecture precludes use of confidence labels in the single classes per rule. In case of clean classification rules, when each single rule contains/covers training samples from a single class, this architecture provides adequate resolution of the class distributions. However, in real-world

problems, classes usually overlap significantly and therefore often rules are extracted containing samples from more than one class.

Thus, an extended fuzzy classification model that includes the confidence levels $conf_{i1,\ldots,K}$ of the ith rule in the single classes has been applied in evolving, adaptive learning context, see e.g., Bouchachia, 2009; Bouchachia and Mittermeir, 2006:

$$\begin{aligned} \text{Rule}_i: \quad & \text{IF } x_1 \text{ IS } \mu_{i1} \text{ AND} \ldots \text{AND } x_p \text{ IS } \mu_{ip} \\ & \text{THEN } l_i = [conf_{i1}, conf_{i2}, \ldots, conf_{iK}] \end{aligned} \tag{17}$$

Thus, a local region represented by a rule in the form of Equation (17) can better model class overlaps in the corresponding part of the feature space: for instance, three classes overlap with a support of 200, 100 and 50 samples in one single fuzzy rule; then, the confidence in Class #1 would be intuitively 0.57 according to its relative frequency (200/350), in Class #2 it would be 0.29 (100/350) and in Class #3 it would be 0.14 (50/350). A more enhanced treatment of class confidence levels will be provided in Section 3.4.5 when describing options for representing reliability in class responses.

In a winner-takes-it-all context [the most common choice in fuzzy classifiers (Kuncheva, 2000)], the final classifier output L will be obtained by

$$L = l_{i^*} \text{ with } l_{i^*} = \text{argmax}_{1\leq k\leq K}\ conf_{i^*k} \quad i^* = \text{argmax}_{1\leq i\leq C}\ \mu_i(\vec{x}) \quad \text{with} \tag{18}$$

In a more enhanced (weighted) classification scheme as recently used for evolving fuzzy classifiers (EFC) in Lughofer (2012a), the *degree of purity* is respected as well and integrated into the calculation of the final classification response L:

$$L = \text{argmax}_{k=m,m*} \left(conf_k = \frac{\mu_1(\vec{x}) h *_{1,k} + \mu_2(\vec{x}) h *_{2,k}}{\mu_1(\vec{x}) + \mu_2(\vec{x})} \right) \tag{19}$$

with

$$h *_{1,k} = \frac{h_{1,k}}{h_{1,m} + h_{1,m*}} \quad h *_{2,k} = \frac{h_{2,k}}{h_{2,m} + h_{2,m*}} \tag{20}$$

and $h_{i,k}$ the class frequency of class k in rule i, and $\mu_1(\vec{x})$ the membership degree of the nearest rule (with majority class m), and $\mu_2(\vec{x})$ the membership degree of the second nearest rule with a different majority class $m* \neq m$. This difference is important as two nearby lying rules with the same majority class label do not induce a decision boundary in-between them. The nearest rule and second nearest rule are obtained by sorting the membership degrees of the current query point to all rules.

Figure 3.5 shows an example for decision boundaries induced by Equation (18) (left image) and by Equation (19) (right image). Obviously, a more purified rule (i.e., having less overlap in classes, right side) is favored among that with significant

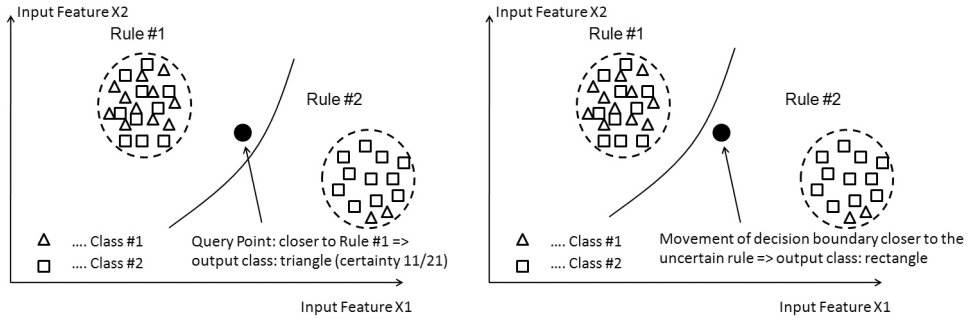


Figure 3.5: (a) Classification according to the winner takes it all concepts using Equation (18); (b) The decision boundary moves towards the more unpurified rule due to the *gravitation concept* applied in Equation (19).

overlap (left side), as the decision boundary moves away → more samples are classified to the majority class in the purified rule, which is intended to obtain a clearer, less uncertain decision boundary. A generalization of Equation (19) would be that k varies over all classes: then, an overwhelmed but significant class in two nearby lying rules may become also the final output class label L, although it has no majority in both rules. On the other hand, this variant would then be able to output a certainty level for each class, an advantage which could be used when calculating a kind of reliability degree overall classes (see Section 3.6.4) respectively when intending to normalize and study class certainty distributions. This variant has not been studied under the scope of EFC so far.

3.2.6.2. *Multi-model one-versus-rest*

The first variant of multi-model architecture is leaned on the well-known *one-versus-rest* classification scheme from the field of machine learning (Bishop, 2007) and has been introduced in the fuzzy community and especially evolving fuzzy systems community in Angelov *et al.* (2008). It diminishes the problematic of having complex non-linear multi-decision boundaries in case of multi-class classification problems, which is the case for single model architecture as all classes are coded into one model. This is achieved by representing K binary classifiers for the K different classes, each one for the purpose to discriminate one single class from the others (→ one-versus-rest). Thus, during the training cycle (batch or incremental), for the kth classifier all feature vectors resp. samples belonging to the kth class are assigned a label of 1, and all other samples belonging to other classes are assigned a label of 0.

The nice thing is that a (single model) classification model $D(f) = C$ respectively any regression model $D(f) = R$ (such as Takagi–Sugeno variants

discussed above) can be applied for one sub-model in the ensemble. Interestingly, in Angelov *et al.* (2008), it has been studied that, when using Takagi–Sugeno architecture for the binary classifiers by regressing on {0, 1}, the masking problem as occurring in *linear regression by indicator matrix* approach can be avoided (Hastie *et al.*, 2009). This is due to the increased flexibility of TS fuzzy systems, being able to resolve non-linearities in the class regression surfaces.

At the classification stage for a new query point $\vec{x}$ that model which is producing the maximal model response is used as basis for the final classification label output L, i.e.,

$$\begin{aligned} L =& \text{argmax}_{m=1,\ldots,K} \hat{f}_m(\vec{x}) \text{ in case when } D(f) = R, \\ L =& \text{argmax}_{m=1,\ldots,K} conf_m(\vec{x}) \text{ in case when } D(f) = C. \end{aligned} \tag{21}$$

Recently, a rule-based one-versus-rest classification scheme was proposed within the context of a MIMO (Multiple Input Multiple Output) fuzzy system and applied in an evolving classification context (Pratama *et al.*, 2014c). There, a rule is defined by:

$$\text{IF} \quad \vec{x} \quad \text{IS (about)} \quad \Psi_i \quad \text{THEN} \quad l_i = \vec{x}\Omega_i \tag{22}$$

where

$$\Omega_i = \begin{bmatrix} w_{i0}^1 & w_{i0}^2 & \ldots & w_{i0}^K \\ w_{i1}^1 & w_{i1}^2 & \ldots & w_{i1}^K \\ \vdots & \vdots & \vdots & \vdots \\ w_{ip}^1 & w_{ip}^2 & \ldots & w_{ip}^K \end{bmatrix}.$$

Thus, a complete hyper-plane for each class per rule is defined. This offers the flexibility to regress on different classes within single rules, thus to resolve class overlaps in a single region by multiple regression surfaces (Pratama *et al.*, 2014c).

3.2.6.3. *Multi-model all-pairs*

The multi-model *all-pairs (aka all-versus-all)* classifier architecture, originally introduced in the machine learning community (Allwein *et al.*, 2001; Fürnkranz, 2002) and firstly introduced for (evolving) fuzzy classifiers design in Lughofer and Buchtala (2013), overcomes the often occurring imbalanced learning problems induced by one-versus-rest classification scheme in case of *multi-class (polychotomous)* problems. On the other hand, it is well-known that imbalanced problems cause severe down-trends in classification accuracy (He and Garcia, 2009). Thus, it is beneficial to avoid imbalanced problems while still trying to enforce the decision boundaries as easy as possible to learn. This is achieved by the all-pairs architecture,

as for each class pair (k, l) an own classifier is trained, decomposing the whole learning problem into binary less complex sub-problems.

Formally, this can be expressed by a classifier $\mathbb{C}_{k,l}$ which is induced by a training procedure $\mathbb{T}_{k,l}$ when using (only) the class samples belonging to classes k and l:

$$\mathbb{C}_{k,l} \longleftarrow \mathbb{T}_{k,l}(X_{k,l}) \quad X_{k,l} = \{\vec{x} | L(\vec{x}) = k \vee L(\vec{x}) = l\}, \tag{23}$$

with $L(\vec{x})$ the class label associated with feature vector $\vec{x}$. This means that $\mathbb{C}_{k,l}$ is a classifier for separating samples belonging to class k from those belonging to class l. It is notable that any classification architecture as discussed above in Section 3.2.6.1 or any regression-based model as defined in Section 3.2.2 can be used for $\mathbb{C}_{k,l}$.

When classifying a new sample $\vec{x}$, each classifier outputs a confidence level $conf_{k,l}$ which denotes the degree of preference of class k over class l for this sample. This degree lies in [0, 1] where 0 means no preference, i.e., a crisp vote for class l, and 1 means a full preference, i.e., a crisp vote for class k. This is conducted for each pair of classes and stored into a *preference relation matrix* R:

$$R = \begin{bmatrix} 0 & conf_{1,2} & conf_{1,3} & \dots & conf_{1,K} \\ conf_{2,1} & 0 & conf_{2,3} & \dots & conf_{2,K} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ conf_{K,1} & conf_{K,2} & conf_{K,3} & \dots & 0 \end{bmatrix}. \tag{24}$$

If we assume reciprocal preferences, i.e., $conf_{k,l} = 1 - conf_{l,k}$, then the training of half of the classifiers can be omitted, hence finally $\frac{K(K-1)}{2}$ binary classifiers are obtained. The preference relation matrix in Equation (24) opens another interpretation dimension on output level: considerations may go into partial uncertainty reasoning or preference relational structure in a fuzzy sense (Hüllermeier and Brinker, 2008). In the most convenient way, the final class response is often obtained by:

$$L = argmax_{k=1,\dots,K}(score_k) = argmax_{k=1,\dots,K}\left(\sum_{K \geq i \geq 1} conf_{k,i}\right). \tag{25}$$

i.e., the class with the highest score = highest preference degree summed up over all classes is returned by the classifier.

In Fürnkranz (2002, 2001) it was shown that pairwise classification is not only more accurate than one-versus-rest technique, but also more efficient regarding computation times [see also Lughofer and Buchtala (2013)], which is an important characteristics for fast stream learning problems. The reason for this is basically that binary classification problems contain significantly lower number of samples, as each sub-problem uses only a small subset of samples.

3.3. Fundamentals

Data streams are one of the fundamental reasons for the necessity of applying evolving, adaptive models in general and evolving fuzzy systems in particular. This is simply because streams are theoretically an infinite sequence of samples, which cannot be processed at once within a batch process, even not in modern computers with high virtual memory capacities. Data streams may not necessarily be on-line based on permanent recordings, measurements or sample gatherings, but can also arise due to a block- or sample-wise loading of batch data sites, e.g., in case of *very large data bases (VLDB)*[6] or in case of *big data* problems (White, 2012); in this context, they are also often referred as *pseudo-streams*. In particular, a data stream (or pseudo-stream) is characterized by the following properties (Gama, 2010):

- The data samples or data blocks are continuously arriving on-line over time. The frequency depends on the frequency of the measurement recording process.
- The data samples are arriving in a specific order, over which the system has no control.
- Data streams are usually not bounded in a size; i.e., a data stream is alive as long as some interfaces, devices or components at the system are switched on and are collecting data.
- Once a data sample/block is processed, it is usually discarded immediately, afterwards.

Changes in the process such as new operation modes, system states, varying environmental influences etc. are usually implicitly also effecting the data stream in way that for instance *drifts* or *shifts* may arise (see Section 3.4.1), or new regions in the feature/system variable space are explored (knowledge expansion).

Formally, a stream can be defined as an infinite sequence of samples $(\vec{x}_1, \vec{y}_1)$, $(\vec{x}_2, \vec{y}_2)$, $(\vec{x}_3, \vec{y}_3)$, ..., where $\vec{x}$ denotes the vector containing all input features (variables) and $\vec{y}$ the output variables which should be predicted. In case of unsupervised learning problems, $\vec{y}$ disappears — note that, however, in the context of fuzzy systems, only supervised regression and classification problems are studied. Often $y = \vec{y}$, i.e., single output systems are encouraged, especially as it is often possible to decast a MIMO (multiple input multiple output problem) system into single independent MISO (multiple input single output problem) systems (e.g., when the outputs are independent).

Handling streams for modeling tasks in an appropriate way requires the usage of *incremental learning* algorithms, which are deduced from the concept of incremental heuristic search (Koenig *et al.*, 2004). These algorithms possess the property to build and learn models in step-wise manner rather than with a whole data set at

[6]http://en.wikipedia.org/wiki/Very_large_database.

once. From formal mathematical point of view, an incremental model update I of the former model f_N (estimated from the N initial samples) is defined by

$$f_{N+m} = I(f_N, (\vec{x}_{N+1,\ldots,N+m}, \vec{y}_{N+1,\ldots,N+m})) \tag{26}$$

So, the incremental model update is done by just taking the new m samples and the old model, but not using any prior data. Hereby, the whole model may also include some additional statistical help measures, which needs to be updated synchronously to the 'real' model. If $m = 1$, we speak about *incremental learning in sample mode* or *sample-wise incremental learning*, otherwise about *incremental learning in block mode* or *block-wise incremental learning*. If the output vector starts to be missing in the data stream samples, but a supervised model has been trained already before which is then updated with the stream samples either in unsupervised manner or by using its own predictions, then someone speaks about *semi-supervised* (on-line) learning (Chapelle *et al.*, 2006).

Two update modes in the incremental learning process are distinguished:

(1) Update of the model parameters: in this case, a fixed number of parameters $\Phi_N = \{\phi_1, \ldots, \phi_l\}_N$ of the original model f_N is updated by the incremental learning process and the outcome is a new parameter setting Φ_{N+m} with the same number of parameters, i.e., $|\Phi_{N+m}| = |\Phi_N|$. Here, we also speak about a *model adaptation* resp. a *model refinement* with new data.
(2) Update of the whole model structure: this case leads to the evolving learning concept, as the number of the parameters may change and also the number of structural components may change automatically (e.g., rules are added or pruned in case of fuzzy systems) according to the characteristics of the new data samples $\vec{x}_{N+1,\ldots,N+m}$. This means that usually (but not necessarily) $|\Phi_{N+m}| \neq |\Phi_N|$ and $C_{N+m} \neq C_N$ with C the number of structural components. The update of the whole model structure also may include an update of the input structure, i.e., input variables/features may be exchanged during the incremental learning process — see also Section 3.4.2.

An important aspect in incremental learning algorithms is the so-called *plasticity-stability dilemma* (Abraham and Robins, 2005), which describes the problem of finding an appropriate tradeoff between flexible model updates and structural convergence. This strongly depends on the nature of the stream: in some cases, a more intense update is required than in others (drifting versus life-long concepts in the stream). If an algorithm converges to an optimal solution or at least to the same solution as the hypothetical batch solution (obtained by using all data up to a certain sample at once), it is called a *recursive algorithm*. Such an algorithm is usually beneficial as long as no drifts arise, which make the older learned relations obsolete (see Section 3.4.1).

An *initial batch mode training step* with the first amount of training samples is, whenever possible, usually preferable to *incremental learning from scratch*, i.e., a building up of the model sample per sample from the beginning. This is because within a batch mode phase, it is possible to carry out validation procedures [such as cross-validation (Stone, 1974) or bootstrapping (Efron and Tibshirani, 1993)] in connection with search techniques for finding an optimal set of parameters for the learning algorithm in order to achieve a good generalization quality. The obtained parameters are then usually reliable start parameters for the incremental learning algorithm to evolve the model further. When performing incremental learning from scratch, the parameters have to be set to some blind default values, which may be not necessarily appropriate for the given data stream mining problem.

In a pure on-line learning setting, however, incremental learning from scratch is indispensable. Then, often start default parameters of the learning engines need to be parameterized. Thus, it is beneficial that the algorithms require as less as possible parameters (see Section 3.3.5). Overcoming unlucky settings of parameters, can be sometimes achieved with dynamic structural changes such as component-based split-and-merge techniques (as described in Section 3.6.2).

3.3.1. *Recursive Learning of Linear Parameters*

A lot of the EFS approaches available in literature (see Section 3.3.5) use the TS-Type fuzzy systems architecture with linear parameters in the consequents. The reason lies in the highly accurate and precise models which can be achieved with these systems (Lughofer, 2011b), and therefore enjoy a wide attraction in several application fields, see Sections 3.2.2 and 3.6.5. Also within several classification variants, TS-fuzzy systems may be used as regression-based binary classifiers, e.g., in all-pairs technique (Lughofer and Buchtala, 2013) as well as in one-versus-rest classification schemes (Angelov *et al.*, 2008). Sometimes, singleton numerical values in the consequents (native Sugeno systems) or higher order polynomials (Takagi–Sugeno–Kang) are used in the consequents. These just change the number of parameters to learn but not the way how to learn them.

The currently available EFS techniques rely on the optimization of the least-squares error criterion, which is defined as the squared deviation between observed outputs $y_1, \ldots, y_N$ and predicted outputs $\hat{y}_1, \ldots, \hat{y}_N$; thus:

$$J = \|y - \hat{y}\|_{L_2} = \sum_{k=1}^{N} (y(k) - \sum_{i=1}^{C} l_{i;\vec{w}}(\vec{x}(k))\Psi_i(\vec{x}(k)))^2 \rightarrow \min_{\vec{w}}!. \qquad (27)$$

This problem can be written as a classical linear least squares problem with a weighted regression matrix containing the global regressors

$$\vec{r}_i(k) = [\Psi_i(\vec{x}(k)) \quad x_1(k)\Psi_i(\vec{x}(k)) \cdots x_p(k)\Psi_i(\vec{x}(k))], \qquad (28)$$

for $i = 1, \ldots, C$, with C the current number of rules and k the kth data sample denoting the kth row. For this problem, it is well-known that a recursive solution exists which converges to the optimal one within each incremental learning step, see Ljung (1999) and Lughofer (2011b). Also Chapter 2 for its detailed derivation in the context of evolving TS fuzzy systems.

However, the problem with this *global learning* approach is that it does not offer any flexibility regarding rule evolution and pruning, as these cause a change in the size of the regressors and thus a dynamic change in the dimensionality of the recursive learning problem, which leads to a disturbance of the parameters in the other rules and to the loss of optimality. Therefore, the authors in Angelov *et al.* (2008) emphasize the usage of *local learning* approach which learns and updates the consequent parameters for each rule separately. Adding or deleting a new rule therefore does not affect the convergence of the parameters of all other rules; thus, optimality in least squares sense is preserved. The local learning approach leads to a weighted least squares formulation for each rule given by (without loss of generality the ith):

$$J_i = \sum_{k=1}^{N} \Psi_i(\vec{x}(k))(y(k) - \hat{y}_i(k))^2 \to \min_{\vec{w}_i} \quad i = 1, \ldots, C, \tag{29}$$

with $\hat{y}_i(k) = l_{i;\vec{w}_i}(\vec{x}(k))$.

This problem can be written as a classical weighted least squares problem, where the weighting matrix is a diagonal matrix containing the basis function values Ψ_i for each input sample. Again an exact recursive formulation can be derived [see Lughofer (2011b) and Chapter 2], which is termed as *recursive fuzzily weighted least squares* (RFWLS). As RFWLS is so fundamental and used in many EFS approaches, we explicitly deploy the update formulas (from the kth to the $k+1$st cycle):

$$\vec{w}_i(k+1) = \vec{w}_i(k) + \gamma(k)(y(k+1) - \vec{r}^T(k+1)\vec{w}_i(k)), \tag{30}$$

$$\gamma(k) = P_i(k+1)\vec{r}(k+1) = \frac{P_i(k)\vec{r}(k+1)}{\frac{\lambda}{\Psi_i(\vec{x}(k+1))} + \vec{r}^T(k+1)P_i(k)\vec{r}(k+1)}, \tag{31}$$

$$P_i(k+1) = \frac{1}{\lambda}(I - \gamma(k)\vec{r}^T(k+1))P_i(k), \tag{32}$$

with $P_i(k) = (R_i(k)^T Q_i(k) R_i(k))^{-1}$ the inverse weighted Hessian matrix and $\vec{r}(k+1) = [1\ x_1(k+1)\ x_2(k+1)\ \ldots\ x_p(k+1)]^T$ the regressor values of the $k+1$st data sample, which is the same for all i rules, and λ a forgetting factor, with default value equal to 1 (no forgetting) — see Section 3.4.1 for a description and meaning of its usage. Whenever $\lambda < 1$ the following function is minimized: $J_i = \sum_{k=1}^{N} \lambda^{N-k} \Psi_i(\vec{x}(k))(y(k) - \hat{y}_i(k))^2$, instead of Equation (29), thus samples which

appeared a long time ago are almost completely out-weighted. Obviously, the actual weight of the sample is Ψ_i (the membership degree to Rule i), thus a sample receives a low weight when it does not fall into rule i: then, the Kalman filter gain $\gamma(k)$ in Equation (31) becomes a value close to 0 and the update of P_i and $\vec{w}_i$ is marginal. Again, Equation (30) converges to the optimal solution within one incremental learning step.

The assurance of convergence to optimality is guaranteed as long as there is not structural change in the rules' antecedents. However, due to rule center movements or resettings in the incremental learning phase (see Section 3.3.4), this is usually not the case. Therefore, a kind of sub-optimality is caused whose deviation degree to the real optimality could have been bounded for some EFS approaches such as FLEXFIS (Lughofer, 2008) and PANFIS (Pratama *et al.*, 2014a).

Whenever a new rule is evolved by a rule evolution criterion, the parameters and inverse weighted Hessian matrix (required for an exact update) have to be initialized. In Ljung (1999), it is emphasized to set $\vec{w}$ to 0 and P_i to αI with α big enough. However, this is for the purpose of a global modeling approach starting with a faded out regression surface over the whole input space. In local learning, the other rules defining other parts of the feature space remain untouched. Thus, setting the hyperplane of the new rule which may appear somewhere inbetween the other rules to 0 would lead to an undesired muting of one local region and to discontinuities in the on-line predictions (Cernuda *et al.*, 2012). Thus, it is more beneficial to inherit the parameter vector and the inverse weighted Hessian matrix from the most nearby lying rule (Cernuda *et al.*, 2012).

Recent extensions of RFWLS are as follows:

- In PANFIS (Pratama *et al.*, 2014a), an additional constant α is inserted, conferring a noteworthy effect to foster the asymptotic convergence of the system error and weight vector being adapted, which acts like a binary function. In other words, the constant α is in charge to regulate the current belief of the weight vectors $\vec{w}_i$ and depends on the approximation and the estimation errors. It is 1 whenever the approximation error is bigger than the system error, and 0 otherwise. Thus, adaptation takes fully place in the first case and completely not place in the second case (which may have advantages in terms of flexibility and computation time). A similar concept is used in the improved version of SOFNN, see Leng *et al.* (2012).
- Generalized version of RFWLS (termed as *FWGRLS*) as used in GENEFIS (Pratama *et al.*, 2014b): this exploits the generalized RLS as derived in Xu *et al.* (2006) and adopts it to the local learning context in order to favor from its benefits as discussed above. The basic difference to RFWLS is that it adds a weight decay regularization term in the least squares problem formulation in

order to punish more complex models. In a final simplification step, it ends up with similar formulas as in Equations (30) to (32), but with the difference to subtract the term $\alpha P_i(k+1)\nabla\phi(\vec{w}_i(k))$ in Equation (30), with α a regularization parameter and ϕ the weight decay function: one of the most popular ones in literature is the quadratic function defined as $\phi(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2$, thus $\nabla\phi = \vec{w}$).

- In some approaches [e.g., eMG (Lemos *et al.*, 2011a) or rGK (Dovzan and Skrjanc, 2011)], weights Ψ_i of the data samples are integrated in the second term of Equation (30) as well, which however do not exactly follow the original derivation of recursive weighted least-squares (Aström and Wittenmark, 1994; Ljung, 1999), but leads to a similar effect.

Alternatively, Cara *et al.* (2013) proposes a different learning scheme for singleton consequent parameters (in a Sugeno fuzzy system) within an evolving fuzzy controller design, which relies on the prediction error of the current model. The update of the ith rule singleton consequent parameter w_{i0} becomes:

$$w_{i0}(k+1) = w_{i0}(k) + C\mu_i(\vec{x}(k+1))(y(k+1) - \hat{y}(k+1)), \tag{33}$$

with μ_i the activation degree of the ith rule as present in the previous time step k (before being updated with the new sample $\vec{x}(k+1)$), and C a normalization constant. Hence, instead of γ, μ_i is used as update gain which is multiplied with the normalization constant.

3.3.2. *Recursive Learning of Non-Linear Parameters*

Non-linear parameters occur in every model architecture as defined throughout Section 3.2.2, mainly only in the fuzzy sets included in the rules' antecedent parts — except for the extended version of TS fuzzy systems (Section 3.2.2.3), where they also appear in the consequents. Often, the parameters in the fuzzy sets define their centers c and characteristic spreads σ, but often the parameters may appear in a different form — for instance, in case of sigmoid functions they define the slope and the point of gradient change. Thus, we generally refer to a non-linear parameter as ϕ and a whole set of non-linear parameters as Φ. The incremental update of non-linear parameters is necessary in order to adjust and move the fuzzy sets and rules composed by the sets to the actual distribution of the data in order to achieve always the correct, well-placed positions. An example is provided in Figure 3.6 where the initial data cloud (circles) in the left upper part changes slightly the position due to new data samples (rectangles). Leaving the original rule (marked with an ellipsoid) untouched, would cause a misplacement of the rule. Thus, it is beneficial to adjust the rule center and its spread accordingly to the new samples. This figure also shows the case of a rule evolution in the lower right part (new samples significantly away from the old rule contour) — as will be discussed in the subsequent section.

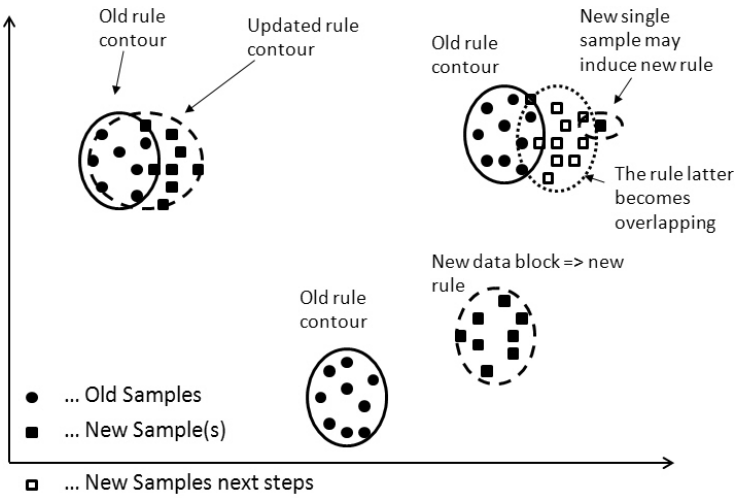


Figure 3.6: Three cases affecting rule contours (antecedents): The left upper part shows a case where a rule movement is demanded to appropriately cover the joint partition (old and new samples), the lower right part shows a case where a new rule should be evolved and the upper right part shows a case where sample-wise incremental learning may trigger a new rule which may turn out to be superfluous later (as future samples are filling up the gap forming one cloud) → (back-)merge requested as discussed in Section 3.3.4.

A possibility to update the non-linear parameters in EFS is again, similar to the consequent parameters, by applying a numerical incremental optimization procedure. Relying on the least squares optimization problem as in case of recursive linear parameter updates, its formulation in dependency of the non-linear parameters Φ becomes:

$$J = J(\Phi) = \sum_{k=1}^{N} \|(y_k - \hat{y}(\Phi))\|_{L_2} \rightarrow min_{\Phi;[\vec{w}]}!. \tag{34}$$

In case of TS fuzzy systems, for instance, $\hat{y}(\Phi) = \sum_{i=1}^{C} l_i(\vec{x})\Psi_i(\Phi)(\vec{x})$. Then, the linear consequent parameters $\vec{w}$ needs to be synchronously optimized to the non-linear parameters (thus, in optional braces), in order to guarantee optimal solution. This can be done either in an alternating nested procedure, i.e., perform an optimization step for non-linear parameters first, see below, and then optimizing the linear ones, e.g., by Equation (30), or within one joint update formula, e.g., when using one Jacobian matrix on all parameters, see below).

Equation (34) is still a free optimization problem, thus any numerical, gradient-based or Hessian-based technique for which a stable incremental algorithm can be developed is a good choice: this is the case for steepest descent, Gauss–Newton method and Levenberg–Marquardt. Interestingly, a common parameter

update formula can be deduced for all three variants (Ngia and Sjöberg, 2000):

$$\Phi(k+1) = \Phi(k) + \mu(k) P(k)^{-1} \psi(\vec{x}(k), \Phi) e(\vec{x}(k), \Phi), \tag{35}$$

where $\psi(\vec{x}(k), \Phi) = \frac{\partial y}{\partial \Phi}(\vec{x}(k))$ the partial derivative of the current model y after each non-linear parameter evaluated at the current input sample $\vec{x}(k)$, $e(\vec{x}(k), \Phi)$ is the residual in the kth sample: $e(\vec{x}(k), \Phi) = y_k - \hat{y}_k$. and $\mu(k) P(k)^{-1}$ the learning gain with $P(k)$ an approximation of the Hessian matrix, which is substituted in different ways:

- For the steepest descent algorithm, $P(k) = I$, thus the update depends only on first order derivative vectors; furthermore, $\mu(k) = \frac{\mu}{\|\vec{r}(k)\|^2}$ with $\vec{r}(k)$ the current regression vector.
- For Gauss–Newton, $\mu(k) = 1-\lambda$ and $P(k) = (1-\lambda) H(k)$ with $H(k)$ the Hessian matrix which can be approximated by $Jac^T(k) Jac(k)$ with Jac the Jacobian matrix (including the derivatives w.r.t. all parameters in all rules for all samples up to the kth) resp. by $Jac^T(k) diag(\Psi_i(\vec{x}(k))) Jac(k)$ in case of the weighted version for local learning (see also Section 3.3.1) — note that the Jacobian matrix reduces to the regression matrix R in case of linear parameters, as the derivatives are the original input variables (thus, $H = R^T R$ in case of recursive linear parameter learning resulting in the native (slow) recursive least squares without inverse matrix update). Additionally to updating the parameters according to Equation (35), the update of the matrix P is required, which is given by:

$$P(k) = \lambda P(k-1) + (1-\lambda) \psi(\vec{x}(k), \Phi) \psi(\vec{x}(k), \Phi)^T. \tag{36}$$

- For Levenberg–Marquardt, $P(k) = (1-\lambda) H(k) + \alpha I$ with $H(k)$ as in case of Gauss–Newton and again $\mu(k) = 1 - \lambda$. The update of the matrix P is done by:

$$P(k) = \lambda P(k-1) + (1-\lambda) \left(\psi(\vec{x}(k), \Phi) \psi(\vec{x}(k), \Phi)^T + \alpha I \right). \tag{37}$$

Using matrix inversion lemma (Sherman and Morrison, 1949) and some reformulation operations to avoid matrix inversion in each step (P^{-1} is required in Equation (35)) leads to the well-known recursive Gauss–Newton approach, which is e.g., used in Komijani *et al.* (2012) for recursively updating the kernel widths in the consequents and also for fine-tuning the regularization parameter. It also results in the recursive least squares approach in case of linear parameters (formulas for the local learning variant in Equation (30)). In case of recursive Levenberg Marquardt (RLM) algorithm, a more complex reformulation option is requested to approximate the update formulas for $P(k)^{-1}$ directly (without intermediate inversion step). This leads to the recursive equations as successfully used in the EFP method by Wang and Vrbanek (2008) for updating centers and spreads in Gaussian fuzzy sets (multivariate Gaussian rules), see also Lughofer (2011b) and Chapter 2.

3.3.3. *Learning of Consequents in EFC*

The most common choice in EFC design for consequent learning is simply to use the class majority voting for each rule separately. This can be achieved by incrementally counting the number of samples falling into each class k and rule i, h_{ik}, (the rule which is the nearest one in the current data stream process). The class with majority count $k* = argmax_{k=1}^{K} h_{ik}$ is the consequent class of the corresponding (ith) rule in case of the classical architecture in Equation (16). The confidences in each class per rule can be obtained by the relative frequencies among all classes $conf_{ik} = \frac{h_{ik}}{\sum_{k=1}^{K} h_{ik}}$ in case of extended architecture in Equation (17). For multi-model classifiers, the same strategy can be applied within each single binary classifier. An enhanced confidence calculation scheme will be handled under the scope of *reliability* in Section 3.4.5.

3.3.4. *Incremental Partitioning of the Feature Space* (*Rule Learning*)

A fundamental issue in *evolving systems*, which differs them from *adaptive systems* is that they possess the capability to change their structure during on-line, incremental learning cycles — adaptive systems are only able to update their parameters as described in the preliminary two sub-sections. The evolving technology addresses the dynamic expansion and contraction of the rule base. Therefore, almost all of the EFS approaches foresee two fundamental concepts for incremental partitioning of the feature space (only some foresee only the first option):

- *Rule evolution*: It addresses the problem when and how to evolve new rules on-the-fly and on demand → knowledge expansion.
- *Rule pruning*: It addresses the problem when and how to prune rules in the rule base on-the-fly and on demand → knowledge contraction, rule base simplification.

The first issue guarantees to include new systems states, operation modes, process characteristics in the models to enrich their knowledge and expand them to so far unexplored regions in the feature space. The second issue guarantees that a rule base cannot grow forever and become extremely large, hence is responsible for smart computation times and compactness of the rule base which may be beneficial for interpretability reasons, see Section 3.5. Also, it is a helpful engine for preventing model over-fitting, especially in case when rules are evolved close to each other or are moving together over time, thus turning out to be superfluous at a later stage. Whenever new rules are evolved, the incremental update of their parameters (as described in the preliminary sub-sections) can begin and continue in the subsequent cycles.

The current state-of-the-art in EFS is that both concepts are handled in different ways in different approaches, see Lughofer (2011b) and "Evolving Systems" Journal by Springer[7] for recently published approaches. Due to space limitations of this book chapter in the whole encyclopedia, it is not possible to describe the various options for rule evolution and pruning anchored in the various EFS approaches. Therefore, we outline the most important directions, which enjoy some common understanding and usage in various approaches.

One concept which is widely used is the *incremental clustering* technique [see Bouchachia (2011)] for a survey of methods], which searches for an optimal grouping of the data into several clusters, ideally following the natural distribution of data clouds. In particular, the aim is that similar samples contribute to the (formation of the) same cluster while different samples should fall into different clusters (Gan *et al.*, 2007). In case when using clustering techniques emphasizing clusters with *convex shapes* (e.g., ellipsoids), these can then be directly associated with rules. Due to their projection onto the axes, the fuzzy sets appearing in the rule antecedents can be obtained. The similarity concept applied in the various approaches differ: some are using distance-oriented criteria [e.g., DENFIS (Kasabov and Song, 2002), FLEXFIS (Lughofer, 2008) or eFuMo (Zdsar *et al.*, 2014)], some are using density-based criteria [e.g., eTS (Angelov and Filev, 2004) and its extension eTS+ Angelov (2010), or Almaksour and Anquetil (2011)] and some others are using statistical-oriented criteria [e.g., ENFM (Soleimani *et al.*, 2010)]; this also effects the rule evolution criterion, often being a threshold (e.g., a maximal allowed distance) which decides whether a new rule is evolved or not. Distance-based criteria may be more prone to outliers than density-based and statistical-based ones; on the other hand, the latter ones can be quite lazy until new rules are evolved (e.g., a significant new dense area is required such that a new rule is evolved there). A summary of EFS approaches and which one applies which criterion will be given in Section 3.3.5.

Fundamental and quite common to many incremental clustering approaches is the update of the centers $\vec{c}$ defining the cluster prototype given by

$$\vec{c}(N+1) = \frac{N\vec{c}(N) + \vec{x}(N+1)}{N+1}, \tag{38}$$

and the update of the inverse covariance matrix Σ^{-1} defining the shape of clusters given by

$$\Sigma^{-1}(N+1) = \frac{\Sigma^{-1}(N)}{1-\alpha} - \frac{\alpha}{1-\alpha}\frac{(\Sigma^{-1}(N)(\vec{x}-\vec{c}))(\Sigma^{-1}(N)(\vec{x}-\vec{c}))^T}{1+\alpha((\vec{x}-\vec{c})^T\Sigma^{-1}(N)(\vec{x}-\vec{c}))},$$

[7] http://www.springer.com/physics/complexity/journal/12530.

with $\alpha = \frac{1}{N+1}$ and N the number of samples seen so far. Usually, various clusters/rules are updated, each representing an own covariance matrix Σ_i^{-1}, thus the symbol $N = k_i$ then represents the number of samples "seen by the corresponding cluster so far", i.e., falling into the corresponding cluster so far (also denoted as the *support of the cluster*).

Other concepts rely

- On the degree of the coverage of the current input sample, i.e. when the coverage is low, a new rule is evolved [e.g., SOFNN (Leng *et al.*, 2005), the approach in Leng *et al.* (2012)].
- On the system error criteria such as the one-step-ahead prediction, i.e., when this is high, a new rule is evolved (e.g., as in SOFNN (Leng *et al.*, 2005) or the approach in Leng *et al.* (2012)) or even a split is performed as in AHLTNM (Kalhor *et al.*, 2010).
- On the rule significance in terms of (expected) statistical contribution and influence, i.e., when a new rule is expected to significantly influence the output of the system it is actually added to the system [e.g. SAFIS (Rong *et al.*, 2006) and its extensions (Rong *et al.*, 2011; Rong, 2012) PANFIS (Pratama *et al.*, 2014a), GENEFIS (Pratama *et al.*, 2014b)).
- On Yager's participatory learning concept (Yager, 1990), comparing the arousal index and the compatibility measure with thresholds [as in ePL (Lima *et al.*, 2010; Lemos *et al.*, 2013), eMG (Lemos *et al.*, 2011a)].
- On the goodness of fit tests based on statistical criteria (e.g., F-statistics) for candidate splits. The leafs are replaced with a new subtree, inducing an expansion of the hierarchical fuzzy model [e.g., in Lemos *et al.* (2011b) or incremental LOLIMOT (Local Linear Model Tree) (Hametner and Jakubek, 2013)].

Furthermore, most of the approaches which are applying an adaptation of the rule contours, e.g., by recursive adaptation of the non-linear antecedent parameters, are equipped with a merging strategy for rules. Whenever rules are forced to move together due to the nature of data stream samples falling in-between these, they may become inevitably overlapping, see the upper right case in Figure 3.6 for an example. The rule evolution concepts cannot omit such occasions in advance, as streaming samples are loaded in the same timely order as they are appearing/recorded in the system — sometimes, originally it may seem that two clusters are contained in the data, which may turn out later as erroneous assumption. Various criteria have been suggested to identify such occurrences and to eliminate them, see Lughofer (2013) and Section 3.2 for a recent overview. In Lughofer and Hüllermeier (2011); Lughofer *et al.* (2011a), a generic concept has been defined for recognizing such overlapping situations based on fuzzy set and rule level. It is applicable for most of the conventional EFS techniques as relying on a geometric-based criterion employing a

rule inclusion metric. This has been expanded in Lughofer *et al.* (2013) to the case of adjacent rules in the feature space showing the same trend in the antecedents and consequents, thus guaranteeing a kind of joint homogeneity. Generic rule merging formulas have been established in Lughofer *et al.* (2011a) and go hand in hand with consistency assurance, especially when equipped with specific merging operations for rule consequent parts, see also Section 3.5.

3.3.5. *EFS Approaches*

In this section, we provide an overview of most important EFS approaches developed since the invention of evolving fuzzy systems approximately 10 years ago. Due to space limitations and the wide variety and manifold of the approaches, we are not able to give a compact summary about the basic methodological concepts in each of these. Thus, we restrict ourselves to report a rough comparison, which is based on the main characteristics and properties of the EFS approaches. This comparison is provided in Table 3.1. Additionally, we demonstrate a pseudo-code in Algorithm 3.1, which shows more or less a common denominator which steps are performed within the learning engines of the EFS approaches.

Algorithm 3.1. Key Steps in an Evolving Fuzzy Systems Learning Engine

(1) Load new data sample $\vec{x}$.
(2) Pre-process data sample (e.g., normalization).
(3) **If** rule-base is empty, initialize the first rule with its center to the data sample $\vec{c} = \vec{x}$ and its spread (range of influence) to some small value; go to Step (1).
(4) **Else**, perform the following steps (5–10):
(5) Check if rule evolution criteria are fulfilled
    (a) If yes, evolve a new rule (Section 3.3.4) and perform body of Step (3) (without if-condition).
    (b) If no, proceed with next step.
(6) Update antecedents parts of (some or all) rules (Sections 3.3.4 and 3.3.2).
(7) Update consequent parameters (of some or all) rules (Sections 3.3.3 and 3.3.1).
(8) Check if the rule pruning/merging criteria are fulfilled
    (a) If yes, prune or merge rules (Section 3.3.4); go to Step (1).
    (b) If no, proceed with next step.
(9) Optional: Perform corrections of parameters towards more optimality.
(10) Go to Step (1).

One comment refers to the update of antecedents and consequents: some approaches may only update those of *some* rules (e.g., the rule corresponding to the

Table 3.1: Comparison of properties and characteristics of important EFS approaches

Method	Architecture	Ant. Learning	Cons. Learning	Rule Pr.	Forget.	Dim. Red.	# of P.	Note
AHLTNM (Kalhor *et al.*, 2010)	TS fuzzy system	Incremental Split-and-Merge (own)	RFWLS	Yes	Yes, cons. only	No	1–2	
DENFIS (Kasabov and Song, 2002)	Neuro-fuzzy system	Evolving Clustering (ECM)	RFWLS with distance weights	No	No	No	1–2	One of the first approaches
EFC-AP (Lughofer and Buchtala, 2013)	All-pairs classifiers	Evolving Clustering (eVQ)	RFWLS or Consequent labeling	Yes	No	No	1–2	Very high accuracy, first all-pairs EFC
EFP (Wang and Vrbanek, 2008)	Mamdani + TS fuzzy system	Evolving Clustering (own) + RLM	RLM + RLS (global)	Yes	Yes, cons. only	No	1–2	First attempt of recursive learning of non-linear params
eFPT (Shaker *et al.*, 2013)	Tree-based structure	Dynamic substitution with neighbor tree	None	Yes	No	Yes	3	Good interpretability, exploits architecture from Huang *et al.* (2008)
eFT (Lemos *et al.*, 2011b)	Tree-based structure	Replacement of leaves with subtrees	RFWLS double-weight	No	No	Yes	4	First approach of evolving fuzzy decision tree

(*Continued*)

July 20, 2015 16:57 Handbook on Computational Intelligence — Volumes I & II - 9.75in x 6.5in b2017-v1-ch03 2nd Reading page 96

Table 3.1: (*Continued*)

Method	Architecture	Ant. Learning	Cons. Learning	Rule Pr.	Forget.	Dim. Red.	# of P.	Note
eFuMo (Zdsar *et al.*, 2014)	Dynamic TS fuzzy system (lags)	Evolving Clustering (Gustafson-Kessel)	RFWLS double-weight.	Yes	Yes, cons. + ante.	No	3–5	Capable of splitting of rules
eMG (Lemos *et al.*, 2011a)	Generalized TS fuzzy system	Participatory Learning	RFWLS double-weight	Yes	No	No	4	First approach using generalized rules
ENFM (Soleimani *et al.*, 2010)	Generalized TS fuzzy system	Recursive Clustering (Gath-Geva)	RFWLS	Yes	Yes, cons. only	No	3	
eClass (Angelov *et al.*, 2008)	single model, one-versus-rest classifiers and MIMO	Evolving clustering (eClustering)	RFWLS + or Cons. labeling	No	Yes, cons. only	No	1–2	First one-vs-rest in EFC
ePL (Lima *et al.*, 2010)	TS fuzzy system	Participatory Learning	RLS (global)	Yes	No	No	5	First usage of participatory (Yager, 1990)
eT2FIS (Tung *et al.*, 2013)	Type-2 Mamdani	Native activation levels, parameter updates	same as antecedent learning	Yes	No	No	3	First evolving Type-2 Mamdani system

(*Continued*)

Table 3.1: (*Continued*)

Method	Architecture	Ant. Learning	Cons. Learning	Rule Pr.	Forget.	Dim. Red.	# of P.	Note
eTS(+) (Angelov, 2010; Angelov and Filev, 2004)	TS fuzzy system	Evolving clustering (eClustering)	RFWLS double-weight	Yes	Yes, later in (Lughofer and Angelov, 2011)	Yes	1–2	One of the pioneering methods (2004)
eTS-LS-SVM (Komijani *et al.*, 2012)	Extended TS fuzzy system	Recursive clustering, suitability	Recursive Gauss–Newton	Yes	No	No	6 (2 + 4)	First approach for evolving extended TS fuzzy system

July 20, 2015 16:57 Handbook on Computational Intelligence — Volumes I & II - 9.75in x 6.5in b2017-v1-ch03 2nd Reading page 98

winning cluster), others may always update those of *all* rules. The former may have some advantages regarding preventing the *unlearning effect* in parts where actual samples do not fall (Lughofer, 2010a), the latter achieves significance and thus reliability in the rule parameters faster (more samples are used for updating).

Although many of these have different facets with a large variety of pros and cons, which cannot be strictly arranged in an ordered manner to say one method is for sure better than the other, the number of parameters (last but one column) gives somehow a bit clarification about the useability resp. the effort to tune the method and finally, to let it run. Tendentially, the more parameters a method has, the more sensitive it is to a particular result and the higher the effort to get it successfully run in an industrial installation. In case when mentioning "$X - Y$" number of parameters it means that Y parameters are the case when forgetting is parameterized (fixed forgetting factor), which is often an optional choice in many applications.

For further details about the concrete algorithms and concepts of the approaches listed in Tables 3.1 and 3.2, please refer to Lughofer (2011b), describing approaches in a compact detailed form from the origin of EFS up to June 2010 and to recently published ones (since July 2010) in "Evolving Systems" Journal by Springer[8] as well as papers in the recent special issues "Evolving Soft Computing Techniques and Applications" (Bouchachia *et al.*, 2014) in *Applied Soft Computing* Journal (Elsevier) and "On-line Fuzzy Machine Learning and Data Mining" (Bouchachia *et al.*, 2013) in *Information Sciences* Journal (Elsevier), and also some recent regular contributions in "IEEE Transactions on Fuzzy Systems"[9] and "IEEE Transactions on Neural Networks and Learning Systems"[10] (neuro-fuzzy approaches).

3.4. Stability and Reliability

Two important issues when learning from data streams are the assurance of stability during the incremental learning process and the investigation of reliability of model outputs in terms of predictions, forecasts, quantifications, classification statements etc. These usually leads to an enhanced *robustness* of the evolved models. Stability is usually guaranteed by all the aforementioned approaches listed in Tables 3.1 and 3.2 as long as the data streams from which the models are learnt appear in a quite "smooth, expected" fashion. However, specific occasions such as *drifts* and *shifts* (Klinkenberg, 2004) or high noise levels may appear in these, which require a specific handling within the learning engine of the EFS approaches. Another problem

[8]http://www.springer.com/physics/complexity/journal/12530.
[9]http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=91.
[10]http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=5962385.

Table 3.2: Comparison of properties and characteristics of important EFS approaches

Method	Architecture	Ant. Learning	Cons. Learning	Rule Pr.	Forget.	Dim. Red.	# of P.	Note
FLEXFIS (++) (Lughofer, 2008, 2012b)	TS fuzzy system	Evolving clustering (eVQ)	RFWLS	Yes	Yes, cons. + ante.	No	1–2	First approach forgetting in ante, enhanced robustness
FLEXFIS-Class (Angelov *et al.*, 2008; Lughofer, 2011c)	Single model, one-vs-rest classifiers	Evolving clustering (eVQ)	RFWLS	No	Yes, cons. only	Yes (smooth)	1–2	First one-vs-rest in EFC
FPA (Wang *et al.*, 2013)	Single model classifiers	Incremental con.-based optimization	Weight vector update	No	No	No	2	
Gen-Smart-EFS (GS-EFS) (Lughofer *et al.*, 2013)	Generalized TS fuzzy system	Evolving clustering (extended eVQ)	RFWLS	Yes	Yes, cons. only	Yes (smooth)	1–2	First joint concept dim. red. for gen. rules + pruning
GENEFIS (Pratama *et al.*, 2014b)	Generalized TS fuzzy system	Gen. adaptive resonance theory + stat. influence	FWGRLS	Yes	No	Yes (crisp)	3–4	Gen. rules + on-line dim. red.

(*Continued*)

July 20, 2015 16:57 Handbook on Computational Intelligence — Volumes I & II - 9.75in x 6.5in b2017-v1-ch03 2nd Reading page 100

Table 3.2: (*Continued*)

Method	Architecture	Ant. Learning	Cons. Learning	Rule Pr.	Forget.	Dim. Red.	# of P.	Note
HAW-NFS (Bodyanskiy and Vynokurova, 2013)	Neuro-fuzzy system	inc opt. [mod. of (Kaczmarz, 1993)]	RLS (global)	No	No	No	3	First wavelet-based neuro-fuzzy
LOLIMOT inc. (Hametner and Jakubek, 2013)	Tree-based structure	Node replacement (recursive split)	Recursive non-linear least squares	No	No	No	1–2	First approach for an incremental LOLIMOT (Nelles, 2001)
PANFIS (Pratama *et al.*, 2014a)	Generalized TS fuzzy system	Extended Self-Organizing Map, statistical influence	FWGRLS	Yes	No	No	2	First proj. concept of gen. rules, stability proofs
rGK (Dovzan and Skrjanc, 2011)	TS fuzzy system	Recursive clustering (GK)	RLS (global), RFWLS (local)	No	Yes, cons. only	No	3–6	No rule evolution
(E)SAFIS (Rong *et al.*, 2014, 2006)	TS fuzzy system (MIMO)	Statistical influence, distance criterion	RLS (global), inc. stability with Lyapunov	Yes	No	No	3	One of the pioneering methods (2005–2006)

(*Continued*)

Table 3.2: (*Continued*)

Method	Architecture	Ant. Learning	Cons. Learning	Rule Pr.	Forget.	Dim. Red.	# of P.	Note
SAFIS-Class (Rong *et al.*, 2011)	Single model using TS fuzzy system	Statistical influence, distance criterion	RLS (global)	Yes	No	No	6	
SEIT2FNN (Juang and Tsao, 2008)	Type-2 TS fuzzy system	Incremental steepest descent	RLS (global)	No	Yes, cons. only	No	3	First evolving Type-2 fuzzy system
SOFMLS (Rubio, 2009)	Mamdani	Evolving clustering (nearest neighborhood)	Modified RLS with increased stability	Yes	No	No	4	First approach of an evolving Mamdani fuzzy system
SOFNN (imp) (Leng *et al.*, 2005, 2012)	Neuro-fuzzy system	Coverage and system error criteria, rule enlargement	Modified RLS (global) with weights	Yes	No	No	2	One of the pioneering methods (2005)

July 20, 2015 16:57 Handbook on Computational Intelligence — Volumes I & II - 9.75in x 6.5in b2017-v1-ch03 2nd Reading page 102

is dedicated to high-dimensional streams, usually stemming from large-scale time-series (Morreale *et al.*, 2013) embedded in the data, and mostly causing a *curse of dimensionality* effect, which leads to over-fitting and downtrends in model accuracy. As can be realized from Column #7 in Tables 3.1 and 3.2, only a few approaches embed any on-line dimensionality reduction procedure so far in order to diminish this effect. Although high noise levels can be automatically handled by RFWLS, its spin-offs and modifications as well as by the antecedent learning engines discussed in Sections 3.3.2 and 3.3.4, reliability aspects in terms of increasing the certainty in model outputs respecting the noise are weakly discussed. Drift handling is included in some of the methods by forgetting strategies, but these are basically only applied for consequent learning in Takagi–Sugeno type fuzzy models (see Column 6 in Tables 3.1 and 3.2).

This section is dedicated to a summary of recent developments in stability, reliability and robustness of EFS which can be generically used in combination with most of the approaches listed in Tables 3.1 and 3.2.

3.4.1. *Drift Handling in Streams*

Drifts in data streams refer to a gradual evolution of the underlying data distribution and therefore of the learning concept over time (Tsymbal, 2004; Widmer and Kubat, 1996) and are frequent occurrences in nowadays non-stationary environments (Sayed-Mouchaweh and Lughofer, 2012). Drifts can happen because the system behavior, environmental conditions or target concepts dynamically change during the on-line learning process, which makes the relations, concepts contained in the old data samples obsolete. Such situations are in contrary to new operation modes or system states which should be integrated into the models with the same weight as the older ones in order to extend the models, but keeping the relations in states seen before untouched (still valid for future predictions). Drifts, however, usually mean that the older learned relations (as parts of a model) are not valid any longer and thus should be incrementally out-weighted, ideally in a smooth manner to avoid catastrophic forgetting (French, 1999; Moe-Helgesen and Stranden, 2005).

A smooth forgetting concept for consequent learning employing the idea of exponential forgetting (Aström and Wittenmark, 1994), is used in approximately half of the EFS approaches listed in Tables 3.1 and 3.2 (refer to Column #6). The strategy in all of these is to integrate a forgetting factor $\lambda \in [0, 1[$ for strengthening the influence of newer samples in the Kalman gain γ — see Equation (31). Figure 3.7 shows the weight impact of samples obtained for different forgetting factor values. This is also compared to a standard sliding window technique, which weights all samples up to a certain point of time in the past equally, but forget all others before completely $\rightarrow$ non-smooth forgetting. This variant is also termed as decremental

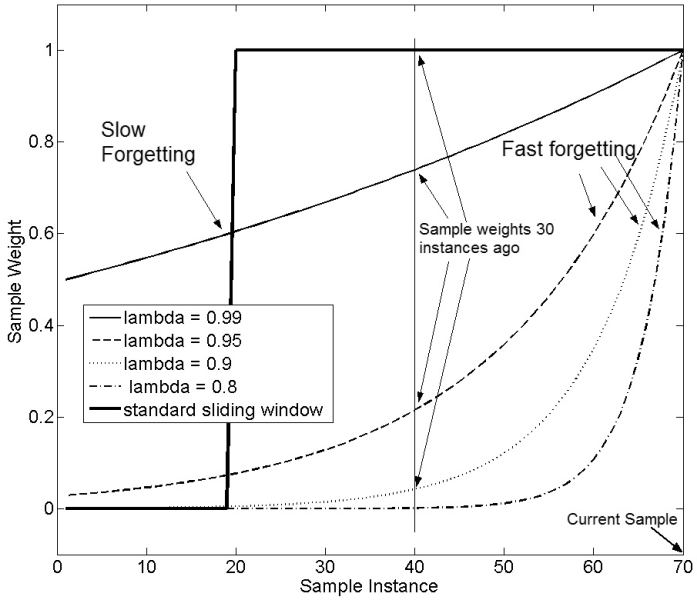


Figure 3.7: Smooth forgetting strategies achieving different weights for past samples; compared to a sliding window with fixed width → complete forgetting of older samples (decremental learning).

learning (Bouillon *et al.*, 2013; Cernuda *et al.*, 2014), as the information contained in older samples falling out of the sliding window is fully unlearned (=decremented) from the model. The effect of this forgetting factor integration is that changes of the target concept in regression problems can be tracked appropriately, i.e., a movement and shaping of the current regression surface towards the new output behavior is enforced, see Lughofer and Angelov (2011). Decremental learning may allow more flexibility (only the latest N samples are really used for model shaping), but increases the likelihood of catastrophic forgetting.

Regarding a drift handling in the antecedent part, several techniques may be used such as a reactivation of rule centers and spreads from a converged situation by an increase of the learning gain: this is conducted in Lughofer and Angelov (2011) for the two EFS approaches eTS and FLEXFIS and has the effect that rules are helped out from their stucked, converged positions to cover the new data cloud appearance of the drifted situation. In eFuMo, the forgetting in antecedent learning is integrated as the degree of the weighted movement of the rule centers $\vec{c}_.$ towards a new data sample $\vec{x}_{N+1}$:

$$\begin{aligned}\vec{c}_i(N+1) &= \vec{c}_i(N) + \Delta\vec{c}_i(N+1) \quad \text{with } \Delta\vec{c}_i(N+1), \\ &= \frac{\mu_i(\vec{x}_{N+1})^\eta(\vec{x}_{N+1} - \vec{c}_i(N))}{s_i(N+1)},\end{aligned} \tag{39}$$

with $s_i(N+1) = \lambda s_i(N) + \mu_i(\vec{x}_{N+1})^\eta$ the sum of the past memberships $\mu_i(\vec{x}_j)$, $j = 1, \ldots, N$, η the fuzziness degree also used as parameters in fuzzy c-means, and λ the forgetting factor. Forgetting is also integrated in the inverse covariance matrix and determinant update defining the shape of the clusters. All other EFS techniques in Tables 3.1 and 3.2 do not embed an antecedent forgetting.

An important investigation is the question when to trigger forgetting and when to apply the conventional life-long learning concept (all samples equally weighted) (Hamker, 2001). In Shaker and Lughofer (2014), it could be analyzed that when using a permanent (fixed) forgetting factor respectively an increased model flexibility in case when no drift happens, the accuracy of the evolved models may decrease over time. Thus, it is necessary to install a drift indication concept, which is able to detect drifts and in ideal cases also to quantify their intensity levels; based on these, it is possible to increase or decrease the degree of forgetting, also termed as *adaptive forgetting*. A recent approach for EFS which performs this by using an extended version of Page–Hinkley test (Mouss *et al.*, 2004), (a widely known and respected test in the field of drift handling in streams (Gama, 2010; Sebastiao *et al.*, 2013) is demonstrated in Shaker and Lughofer (2014). It is also the first attempt to localize the drift intensity by quantifying drift effects in different local parts of the features space with different intensities and smoothly: EFS are a perfect model architecture to support such a local smooth handling (fuzzy rules with certain overlaps).

3.4.2. *On-Line Curse of Dimensionality Reduction*

For models including localization components as is the case of evolving fuzzy systems (in terms of rules), it is well-known that *curse of dimensionality* is very severe in case when a high number of variables are used as model inputs (Pedrycz and Gomide, 2007), e.g., in large-scale time-series, recorded in multi-sensor networks (Chong and Kumar, 2003). This is basically because in high-dimensional spaces, someone cannot speak about locality any longer (on which these types of models rely), as all samples are moving to the edges of the joint feature space — see Hastie *et al.* (2009) and Chapter 1 for a detailed analysis of this problem.

Therefore, the reduction of the dimensionality of the learning problem is highly desired. In data stream sense, to ensure an appropriate reaction onto the system dynamics, the feature reduction should be conducted on-line and be open for *anytime changes*. A possibility is to track the importance levels of features over time and to cut out that ones which become unnecessary — as has been used in connection with EFS for regression problems in Angelov (2010); Pratama *et al.* (2014b) and for classification problems in a first attempt in Bouchachia and Mittermeir (2006). However, features which are unimportant at an earlier point of time may become important at a later stage (*feature reactivation*). This means that crisply cutting out

some features with the usage of on-line feature selection and ranking approaches such as Li (2004); Ye *et al.* (2005) can fail to represent the recent feature dynamics appropriately. Without a re-training cycle, which, however slows down the process and causes additional sliding window parameters, this would lead to discontinuities in the learning process (Lughofer, 2011b), as parameters and rule structures have been learnt on different feature spaces before.

An approach which is addressing input structure changes incrementally on-the-fly is presented in Lughofer (2011c) for classification problems using classical single model and one-versus-rest based multi-model architectures (in connection with FLEXFIS-Class learning engine). It operates on a global basis, hence features are either seen as important or unimportant for the whole model. The basic idea is that feature weights $\lambda_1, \ldots, \lambda_p \in [0, 1]$ for the p features included in the learning problem are calculated based on a stable separability criterion (Dy and Brodley, 2004):

$$J = trace(S_w^{-1} S_b), \tag{40}$$

with S_w the within scatter matrix modeled by the sum of the covariance matrices for each class, and S_b the between scatter matrix, modeled by the sum of the degree of mean shift between classes. The criterion in Equation (40) is applied 1). Dimension-wise to see the impact of each feature separately — note that in this case it reduces to a ratio of two variances — and 2). For the remaining $p - 1$ feature subspace in order to gain the quality of separation when excluding each feature. In both cases, p criteria $J_1, \ldots, J_p$ according to Equation (40) are obtained. For normalization purposes to [0, 1], finally the feature weights are defined by:

$$\lambda_j = 1 - \frac{J_j - min_{j=1,\ldots,p}(J_j)}{max_{j=1,\ldots,p}(J_j) - min_{j=1,\ldots,p}(J_j)}. \tag{41}$$

To be applicable in on-line learning environments, updating the weights in incremental mode is achieved by updating the within-scatter and between-scatter matrices using the recursive covariance matrix formula (Hisada *et al.*, 2010). This achieves a smooth change of feature weights = feature importance levels over time with new incoming samples. Features may become out-weighted (close to 0) and reactivated (weights significantly larger than 0) at a later stage without "disturbing" the parameter and rule structure learning process. Hence, this approach is also denoted as *smooth and soft on-line dimension reduction* — the term *softness* comes from the decreased weights instead of a crisp deletion. Down-weighted features then play a marginal role during the learning process, e.g., the rule evolution criterion relies more on the highly weighted features.

The feature weights concept has been recently employed in Lughofer *et al.* (2014), in the context of data stream regression problems, there with the usage

of generalized rules as defined in Section 3.2.2.2, instead of axis-parallel ones as used in Lughofer (2011c). The features weights are calculated by a combination of future-based expected statistical contributions of the features in all rules and their influences in the rules' hyper-planes (measured in terms of gradients), see Lughofer *et al.* (2014).

3.4.3. *Incremental Smoothing*

Recently, a concept for incremental smoothing of consequent parameters has been introduced in Rosemann *et al.* (2009), where a kind of *incremental regularization* is conducted after each learning step. This has the effect to decrease the level of over-fitting whenever the noise level in the data is high. Indeed, when applying the local recursive least squares estimator as in Equations (30) to (32) and some of its modifications, the likelihood of over-fitting is small due to the enforcement of the local approximation and interpretation property [as analyzed in Angelov *et al.* (2008)], but still may be present. The approach in Rosemann *et al.* (2009) accomplishes the smoothing by correcting the consequent functions of the updated rule(s) by a template T measuring the violation degree subject to a meta-level property on the neighboring rules.

Finally, it should be highlighted that this strategy assures smoother consequent hyper-planes over nearby lying or even touching rules, thus increases the likelihood of further rule reduction through extended simplicity assurance concepts (adjacency, homogeneuity, trend-continuation criteria) as discussed in Lughofer (2013) and successfully applied to obtain compact rule bases in data stream regression problems in Lughofer *et al.* (2013), employing generalized fuzzy rules, defined in Section 3.2.2.2.

3.4.4. *Convergence Analysis/Ensurance*

Another important criterion when applying EFS is some sort of convergence of the parameters included in the fuzzy systems over time (in case of a regular stream without a drift etc.) — this accounts for the stability aspect in the stability-plasticity dilemma (Hamker, 2001), which is important in the life-long learning context. This is for instance accomplished in the FLEXFIS approach, which, however only guarantees a sub-optimality subject to a constant (thus guaranteeing finite solutions) due to a quasi-monotonic decreasing intensity of parameter updates, but is not able to provide a concrete level of this sub-optimality, see Lughofer (2008) and Lughofer (2011b) and Chapter 3. In the approach by Rubio (2010), a concrete upper bound on the identification error is achieved by the modified least squares algorithm to train both, parameters and structures, is achieved with the support of Lyapunov function. The upper bound depends on the actual certainty of the model

output. Another approach which is handling the problem of constraining the model error while assuring parameter convergence simultaneously is applied within the PANFIS learning engine (Pratama *et al.*, 2014a): This is achieved with the usage of an extended recursive least squares approach.

3.4.5. *Reliability*

Reliability deals with the *interpretation of the model predictions* in terms of classification statements, quantification outputs or forecasted values in time series. Reliability points to the trustworthiness of the predictions of current query points which may be basically influenced by two factors:

- The quality of the training data resp. data stream samples seen so far.
- The nature and localization of the current query point for which a prediction should be obtained.

The trustworthiness/certainty of the predictions may support/influence the users/operators during a final decision finding — for instance, a query assigned to a class may cause a different user's reaction when the trustworthiness about this decision is low, compared to when it is high. In this sense, it is an essential add-on to the actual model predictions which may also influence its further processing.

The first factor basically concerns the noise level in measurement data. This also may cover aspects in the direction of uncertainties in users' annotations in classification problems: a user with lower experience level may cause more inconsistencies in the class labels, causing overlapping classes, finally increasing conflict cases (see below); similar occasions may happen in case when several users annotate samples on the same systems, but have different opinions in borderline cases.

The second factor concerns the position of the current query point with respect to the definition space of the model. A model can show a perfect trend with little uncertainty, but a new query point appears far away from all training samples seen so far, yielding a severe degree of extrapolation when conducting the model inference process. In a classification context, a query point may also fall close in a highly overlapped region or close to the decision boundary between two classes. The first problem is denoted as *ignorance*, the second as *conflict* (Hüllermeier and Brinker, 2008; Lughofer, 2012a). A visualization of these two occasions is shown in Figure 3.8 (a) (for conflict) and 3.8 (b) (for ignorance). The conflict case is due to a sample falling in-between two classes and the ignorance case due to a query point falling outside the range of training samples, which is indeed linearly separable, but by several possible decision boundaries [also termed as the *variability of the version space* (Hüllermeier and Brinker, 2008)].

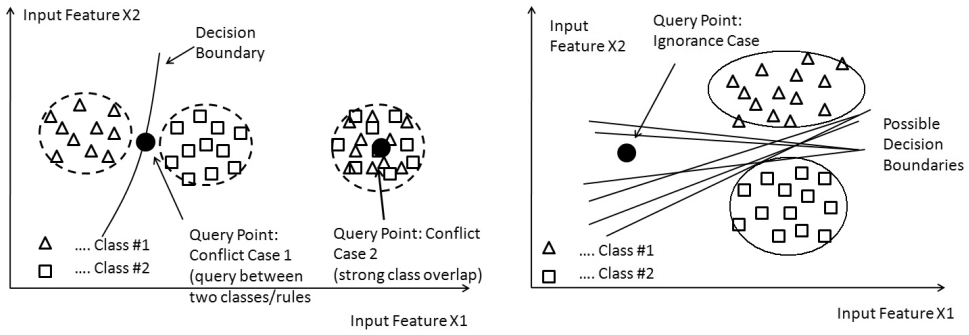


Figure 3.8: (a) Two conflict cases: query point falls inbetween two distinct classes and within the overlap region of two classes; (b) Ignorance case as query point lies significantly away from the training samples, thus increasing the *variability of the version space* (Hüllermeier and Brinker, 2008); in both figures, rules modeling the two separate classes are shown by an ellipsoid, the decision boundaries indicated by solid lines.

In the regression context, the estimation of parameters through RWFLS and modifications usually can deal with high noise levels in order to find a non-overfitting trend of the approximation surface. However, it is not possible to represent uncertainty levels of the model predictions later on. These can be modeled by so-called error bars or confidence intervals (Smithson, 2003). In EFS they have been developed based on parameter uncertainty in Lughofer and Guardiola (2008a) [applied to on-line fault detection in Serdio *et al.* (2014a)] and in an extended scheme in Skrjanc (2009) (for chemometric calibration purposes). The latter is based on a funded deduction from statistical noise and quantile estimation theory (Tschumitschew and Klawonn, 2012). Both are applicable in connection with local (LS) learning of TS consequent parameters.

In the classification context, conflict and ignorance can be reflected and represented by means of fuzzy classifiers in a quite natural way (Hühn and Hüllermeier, 2009). These concepts have been only recently tackled once in the field of evolving fuzzy classifiers (EFC), see Lughofer and Buchtala (2013), where multiple binary classifiers are trained in an all-pairs context for obtaining simper decision boundaries in multi-class classification problems (see Section 3.2.6.3). On a single rule level, the confidence in a prediction can be obtained by the confidence in the different classes coded into the consequents of the rules having the form of Equation (17). This provides a perfect conflict level (close to 0.5 $\rightarrow$ high conflict, close to 1 $\rightarrow$ low conflict) in case of overlapping classes within a single rule. If a query point $\vec{x}$ falls in-between rules with different majority classes (different maximal confidence levels), then the extended weighted classification scheme in Equation (19) is requested to represent a conflict level. If the confidence in the final

output class L, $conf_L$, is close to 0.5, conflict is high, when it is close to 1, conflict is low. In case of all-pairs architecture, Equation (19) can be used to represent conflict levels in the binary classifiers. Furthermore, an overall conflict level on the final classifier output is obtained by Lughofer and Buchtala (2013):

$$conflict_{deg} = \frac{score_k}{score_k + score_l}, \tag{42}$$

with k and l the classes with the highest scores.

The ignorance criterion can be resolved in a quite natural way, represented by a degree of extrapolation, thus:

$$Ign_{deg} = 1 - \max_{i=1}^{C} \mu_i(\vec{x}), \tag{43}$$

with C the number of rules currently contained in the evolved fuzzy classifier. In fact, the degree of extrapolation is a good indicator of the degree of ignorance, but not necessarily sufficient, see Lughofer (2012a), for an extended analysis and further concepts. However, integrating the ignorance levels into the preference relation scheme of all-pairs evolving fuzzy classifiers according to Equation (24) for obtaining the final classification statement, helped to boost the accuracies of the classifier significantly, as then classifiers which show a strongly extrapolated situation in the current query are down-weighted towards 0, thus masked-out, in the scoring process. This lead to an out-performance of incremental machine learning classifiers from MOA framework[11] (Bifet *et al.*, 2010) on several large-scale problems, see Lughofer and Buchtala (2013). The overall ignorance level of an all-pairs classifier is the minimal ignorance degree calculated by Equation (43) over all binary classifiers.

3.5. Interpretability

Improved transparency and interpretability of the evolved models may be useful in several real-world applications where the operators and experts intend to gain a deeper understanding of the interrelations and dependencies in the system. This may enrich their knowledge and enable them to interpret the characteristics of the system on a deeper level. Concrete examples are decision support systems or classification systems, which sometimes require the knowledge why certain decisions have been made, e.g., see Wetter (2000): Insights into these models may provide answers to important questions (e.g., providing the health state of a patient) and support the user in taking appropriate actions. Another example is the substitution of expensive hardware with *soft sensors*, referred to as *eSensors* in an evolving context (Angelov

[11] http://moa.cms.waikato.ac.nz/.

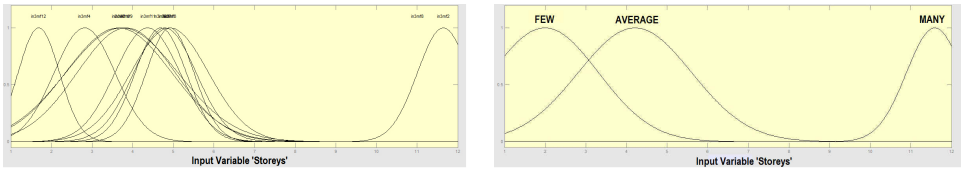
and Kordon, 2010a; Macias-Hernandez and Angelov, 2010): The model has to be linguistically or physically understandable, reliable, and plausible to an expert, before it can be substituted for the hardware. In often cases, it is beneficial to provide further insights into the control behavior (Nelles, 2001).

Interpretability, apart from pure complexity reduction, has been addressed very little in the evolving systems community so far (under the scope of data stream mining). A recent position paper published in *Information Sciences* journal Lughofer (2013) summarizes the achievements in EFS, provides avenues for new concepts as well as concrete new formulas and algorithms and points out open issues as important future challenges. The basic common understanding is that complexity reduction as a key pre-requisite for compact and therefore transparent models is handled in most of the EFS approaches, which can be found nowadays in literature (please also refer to Column "Rule pruning" in Tables 3.1 and 3.2), whereas other important criteria [known to be important from conventional batch off-line design of fuzzy systems (Casillas *et al.*, 2003; Gacto *et al.*, 2011)], are more or less loosely handled in EFS. These criteria include:

- Distinguishability and Simplicity
- Consistency
- Coverage and Completeness
- Local Property and Addition-to-One-Unity
- Feature Importance Levels
- Rule Importance Levels
- Interpretation of Consequents
- Interpretability of Input–Output Behavior
- Knowledge Expansion

Distinguishability and *simplicity* are handled under the scope of complexity reduction, where the difference between these two lies in the occurrence of the degree of overlap of rules and fuzzy sets: Redundant rules and fuzzy sets are highly overlapping and therefore indistinguishable, thus should be merged, whereas obsolete rules or close rules showing similar approximation/classification trends belong to an unnecessary complexity which can be simplified (due to pruning). Figure 3.9 visualizes an example of a fuzzy partition extracted in the context of house price estimation (Lughofer *et al.*, 2011b), when conducting native precise modeling (left) and when conducting some simplification steps according to merging, pruning and constrained-based learning (right). Only in the right case, it is possible to assign linguistic labels to the fuzzy sets and hence to achieve interpretation quality.

Consistency addresses the problem of assuring that no contradictory rules, i.e., rules which possess similar antecedents but dissimilar consequents, are present in the system. This can be achieved by merging redundant rules, i.e., those one which



(a) (b)

Figure 3.9: (a) Weird un-interpretable fuzzy partition for an input variable in house pricing; (b) The same partition achieved when conducting merging, pruning options of rules and sets during the incremental learning phase → assignments of linguistic labels possible.

are similar in their antecedents, with the usage of the participatory learning concept introduced by Yager (1990). An appropriate merging of the linear parameter vectors $\vec{w}$ is given by Lughofer and Hüllermeier (2011):

$$\vec{w}_{new} = \vec{w}_{R_1} + \alpha \cdot Cons(R_1, R_2) \cdot (\vec{w}_{R_2} - \vec{w}_{R_1}), \tag{44}$$

where $\alpha = k_{R_2}/(k_{R_1} + k_{R_2})$ represents the *basic learning rate* with k_{R_1} the support of the more supported rule R_1 and $Cons(R_1, R_2)$ the *compatibility measure* between the two rules within the participatory learning context. The later is measured by a consistency degree between antecedent and consequents of the two rules. It relies on the exponential proportion between rule antecedent similarity degree (overlap) and rule consequent similarity degree.

Coverage and *completeness* refers to the problem of a well-defined coverage of the input space by the rule-base. Formally, ϵ-completeness requires that for each new incoming data sample there exists at least one rule to which this sample has a membership degree of at least ϵ. A specific re-adjustment concept of fuzzy sets and thus rules is presented in Lughofer (2013), which restricts the re-adjustment level in order to keep the accuracy high. An alternative, more profound option for data stream regression problems is offered as well in Lughofer (2013), which integrates a punishment term for ϵ-completeness into the least squares optimization problem. Incremental optimization techniques based on gradients of the extended optimization function may be applied in order to approach but not necessarily fully assure ϵ-completeness. On the other hand, the joint optimization guarantees a reasonable tradeoff between model accuracy and model coverage of the feature space.

Local property and *addition-to-one-unity* have been not considered so far, but will go a step further by ensuring fuzzy systems where only maximal 2^p rules fire at the same time (Bikdash, 1999), with p the input dimensionality. From our point of experience, this cannot be enforced by significantly loosing some model accuracy, as it requires significant shifts of rules away from the real position of the data clouds/density swarms.

Feature importance levels are an outcome of the on-line dimensionality reduction concept discussed in Section 3.4.2. With their usage, it is possible to obtain an interpretation on input structure level which features are more important and which ones are less important. Furthermore, they may also lead to a reduction of the rule lengths thus increasing rule transparency and readability, as features with weights smaller than ϵ have a very low impact on the final model output and therefore can be eliminated from the rule antecedent and consequent parts when showing the rule base to experts/operators.

Rule importance levels could serve as essential interpretation component as they tend to show the importance of each rule in the rule base. Furthermore, rule weights may be used for a smooth rule reduction during learning procedures, as rules with low weights can be seen as unimportant and may be pruned or even re-activated at a later stage in an on-line learning process (*soft rule pruning mechanism*). This strategy may be beneficial when starting with an expert-based system, where originally all rules are fully interpretable (as designed by experts/users), however some may turn out to be superfluous over time for the modeling problem at hand (Lughofer, 2013). Furthermore, rule weights can be used to handle inconsistent rules in a rule base, see e.g., Pal and Pal (1999); Cho and Park (2000), thus serving for another possibility to tackle the problem of consistency (see above). The usage of rule weights and their updates during incremental learning phases, was, to our best knowledge, not studied so far in the evolving fuzzy community. In Lughofer (2013), a first concept was suggested to adapt the rule weights, integrated as non-linear parameters in the fuzzy systems architecture, based on incremental optimization procedures, see also Section 3.3.2.

Interpretation of consequents is assured by nature in case of classifiers with consequent class labels plus confidence levels; in case of TS fuzzy systems it is assured as soon as local learning of rule consequents is used (Angelov *et al.*, 2008; Lughofer, 2011b) Chapter 2 and Lughofer (2013). Please also refer to Section 3.3.1: Then, a snuggling of the partial linear trends along the real approximation surface is guaranteed, thus giving rise in which parts of the feature space the model will react in which way and intensity (gradients of features).

Interpretability of Input–Output Behavior refers to the understanding which output(s) will be produced when showing the system concrete input queries. For instance, a model with constant output has a maximal input–output interpretability (as being very predictable what outcome will be produced for different input values), however usually suffers from predictive accuracy (as long as the behavior of the system to be modeled is non-constant). Most actively firing rules can be used as basis for this analysis.

Knowledge expansion refers to the automatic integration of new knowledge arising during the on-line process on demand and on-the-fly, also in order to expand

the interpretation range of the models, and is handled by all conventional EFS approaches through rule evolution and/or splitting, see Tables 3.1 and 3.2.

3.5.1. *Visual Interpretability*

Visual interpretability refers to an interesting alternative to *linguistic interpretability* (as discussed above), namely to the representation of a model in a graphical form. In our context, this approach could be especially useful if models evolve quickly, since monitoring a visual representation might then be easier than following a frequently changing linguistic description. Under this scope, alternative "interpretability criteria" may then become interesting which are more dedicated to the timely development of the evolving fuzzy model — for instance, a trajectory of rule centers showing their movement over time, or trace paths showing birth, growth, pruning and merging of rules; first attempts in this direction have been conducted in Henzgen *et al.* (2013), employing the concept of *rule chains*. These have been significantly extended in Hentzgen *et al.* (2014) by setting up a visualization framework with a grown-up user front-end (GUI), integrating various similarity, coverage and overlap measures as well as specific techniques for an appropriate catchy representation of high-dimensional rule antecedents and consequents. Internally, it uses the FLEXFIS++ approach (Lughofer, 2012b) as incremental learning engine.

3.6. Useability and Applications

In order to increase the useability of evolving fuzzy systems, several issues are discussed in this section, ranging from the reduction of annotation efforts in classification settings through a higher plug-and-play capability (more automatization, less tuning) to the decrease of computational resources and as well as to on-line evaluation measures for supervising modeling performance. At the end of this section, a list of real-world applications making use of evolving fuzzy systems will be discussed.

Finally, the increase of the useability together with the assurance of interpretability serves as basis for a successful future development of the *human-inspired evolving machines (HIEM)* concept as discussed in Lughofer (2011a), which is expected to be the *next generation of evolving intelligent systems*[12] — the aim is to enrich the pure machine learning systems with human knowledge and feelings, and to form a joint concept of *active learning and teaching* in terms of a higher-educated computational intelligence useable in artificial intelligence.

[12]http://en.wikipedia.org/wiki/Evolving_intelligent_system.

3.6.1. *Single-Pass Active Learning*

3.6.1.1. *For classification problems*

In on-line classification tasks, all evolving and incremental classifier variants require provision of the *ground truth* in form of *true class labels* for incoming samples to guarantee smooth and well-defined updates for increased classifiers' performance. Otherwise, classifiers' false predictions self-reinforce and are back-propagated into their structure and parameters, leading to a deteriorization of their performance over time (Gama, 2010; Sayed-Mouchaweh and Lughofer, 2012). The problem, however, is that the true class labels of new incoming samples are usually not included in the stream neither provided automatically by the system. Mostly, operators or experts have to provide the ground truth which requires considerable manpower and fast manual responses in case of on-line processes. Therefore, in order to attract the operators and users to work and communicate with the system, thus to assure classifier useability in on-line mode, decreasing the number of required samples for evolving and improving a classifier over time is essential.

This task can be addressed by *active learning* (Settles, 2010), a technique where the learner itself has control over which samples are used to update the classifiers (Cohn *et al.*, 1994). Conventional active learning approaches operate fully in batch mode: 1. New samples are selected iteratively and sequentially from a pool of training data; 2. The true class labels of the selected samples are queried from an expert; and 3. Finally, the classifier is re-trained based on the new samples together with those previously selected. In an on-line setting such iteration phases over a pool of data samples are not practicable. Thus, a requirement is that the sample selection process operates autonomously in a single-pass manner, omitting time-intensive re-training phases.

Several first attempts have been made in connection with linear classifiers (Chu *et al.*, 2011; Sculley, 2007). A non-linear approach which employs both, single fuzzy model architecture with extended confidence levels in the rule consequents [as defined in Equation (17)] and the all-pairs concept as defined in Section 3.2.6.3, is demonstrated in Lughofer (2012a). There, the actual evolved fuzzy classifier itself decides for each sample whether it helps to improve the performance and, when indicated, requests the true class label for updating its parameters and structure. In order to obtain the certainty level for each new incoming data sample (query point), two reliability concepts are explored: *conflict* and *ignorance*, both motivated and explained in detail in Section 3.4.5: when one of the two cases arises for a new data stream sample, a class label is requested from operators. A common understanding based on several results on high-dimensional classification streams (binary and multi-class problems) was that a very similar tendency of accuracy trend lines over time can be achieved when using only 20–25% of the data samples in the stream for

classifier updates, which are selected based on the single-pass active learning policy. Upon random selection, the performance deteriorates significantly. Furthermore, a batch active learning scheme based on re-training cycles using SVMs classifiers (Schölkopf and Smola, 2002) (lib-SVM implementation[13]) could be significantly out-performed in terms of accumulated one-step-ahead accuracy (see Section 3.6.4 for its definition).

3.6.1.2. *For regression problems*

On-line active learning may be also important in case of regression problems whenever for instance the measurements of a supervised target are quite costly to obtain. An example are the gathering of titration values within a spin-bath analytics at a viscose production process (courtesy to Lenzing GmbH), which are for the purpose to supervise the regulation of the substances H_2SO_4, Na_2SO_4 and Z_nSO_4 (Cernuda *et al.*, 2014). There, active learning is conducted in *incremental and decremental stages* with the help of a sliding window-based approach (sample selection for incoming as well as outgoing points) and using TS fuzzy models connected with PLS. It indeed exceeds the performance of conventional equidistant and costly model updates, but is not fully operating in a single-pass manner (a window of samples is required for re-estimation of statistics etc.). Single-pass strategies have been, to our best knowledge, not handled in data stream regression problems, neither in connection with evolving fuzzy systems.

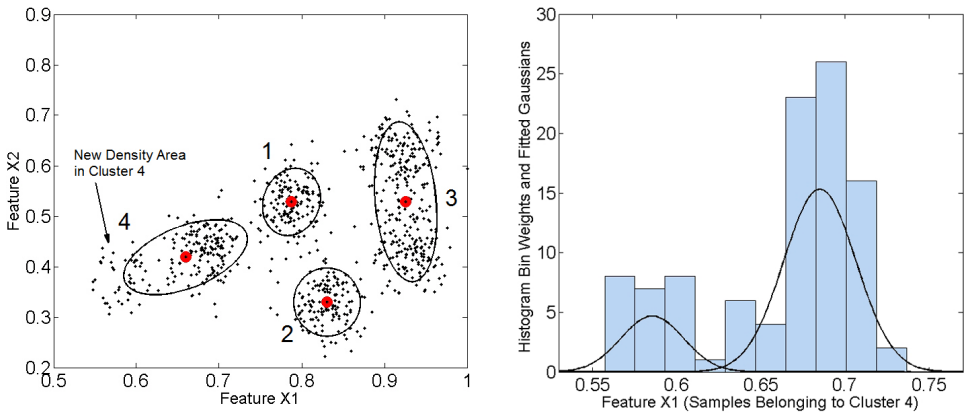
3.6.2. *Toward a Full Plug-and-Play Functionality*

The *plug-and-play functionality* of on-line incremental learning methods is one of the most important properties in order to prevent time-intensive pre-training cycles in batch mode and to support an easy useability for experts and operators. The situation in the EFS community is that all EFS approaches as listed in Tables 3.1 and 3.2 allow the possibility to incrementally learn the models from scratch. However, all of these require at least one or a few learning parameters guiding the engines to correct, stable models — see Column #8 in these tables. Sometimes, a default parametrization exists, but is sub-optimal for upcoming new future learning tasks, as having been optimized based on streams from past processes and application scenarios. Cross-validation (Stone, 1974) or boot-strapping (Efron and Tibshirani, 1993) may help to guide the parameters to good values during the start of the learning phase (carried out on some initial collected samples), but, apart that these iterative batch methods are eventually too slow (especially when more than two parameters need to be adjusted), a stream may turn out to change its characteristics later on (e.g., due

[13] http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

to a drift, see Section 3.4.1). This usually would require a dynamic update of the learning parameters, which is not supported by any EFS approach so far and has to be specifically developed in connection with the concrete learning engine.

An attempt to overcome such an unlucky or undesired parameter setting is presented in Lughofer and Sayed-Mouchaweh (2015) for evolving cluster models, which, however, may be easily adopted to EFS approaches, especially to those ones performing an incremental clustering process for rule extraction. Furthermore, the prototype-based clusters in Lughofer and Sayed-Mouchaweh (2015) are axis-parallel defining local multivariate Gaussian distributions in the feature space. Thus, when using Gaussian fuzzy sets in connection with product t-norm, the same rule shapes are induced. The idea in Lughofer and Sayed-Mouchaweh (2015) is based on dynamic split-and-merge concepts for clusters (rules) which are either moving together forming a homogenous joint region ($\rightarrow$ merge requested) or are internally containing two or more distinct data clouds, thus already housing some internal heterogeneity ($\rightarrow$ split requested), see Figure 3.10 (Cluster #4) for an example, also showing the internal structure of Cluster #4 in the right image. Both occurrences may be arising either due to the nature of the stream or often due to a wrong parametrization of the learning engine (e.g., a too low threshold such that new rules are evolved too early). The main difficulty lies on the identification of when to merge and when to split: parameter-free options are discussed in Lughofer and Sayed-Mouchaweh (2015). Opposed to other *joint* merging and splitting concepts in some EFS approaches, one strength of the approach in Lughofer and Sayed-Mouchaweh (2015) is that it can be used independently from the concrete learning engine. The application of the unsupervised automatic splitting and merging concepts



(a) (b)

Figure 3.10: (a) The cluster structure after 800 samples, Cluster #4 containing already a more distinct density area, (b) Its corresponding histogram along Feature X1, showing the clear implicit heterogenous nature.

to supervised streaming problems under the scope of EFS/EFC may thus be an interesting and fruitful future challenge.

3.6.3. *On Improving Computational Demands*

When dealing with on-line data stream processing problems, usually the computational demands required for updating the fuzzy systems are an essential criteria whether to install and use the component or not. It can be a kick-off criterion, especially in real-time systems, where the update is expected to terminate in real-time, i.e., before the next sample is loaded, the update cycle should be finished. Also, the model predictions should be in-line the real-time demands, but these are known to be very fast in case of fuzzy inference scheme (significantly below milliseconds) (Kruse *et al.*, 1994; Pedrycz and Gomide, 2007; Piegat, 2001). An extensive evaluation and especially a comparison of computational demands for a large variety of EFS approaches over a large variety of learning problems with different number of classes, different dimensionality etc. is unrealistic, also because most of the EFS approaches are hardly downloadable or to obtain from the authors. A loose attempt in this direction has been made by Komijani *et al.* (2012), where they classify various EFS approaches in terms of computation speed into three categories: *low, medium, high*.

Interestingly, the consequent update is more or less following the complexity of $O(Cp^2)$ with p the dimensionality of the feature space and C the number of rules, when local learning is used (as for most EFS approaches, compare in Tables 3.1 and 3.2), and following the complexity of $O((Cp)^2)$ when global learning is applied. The quadratic terms p^2 resp. $(Cp)^2$ are due to the multiplication of the inverse Hessian with the actual regressor vector in Equation (31), and because their sizes are $(p+1)\times(p+1)$ and $p+1$ in case of local learning (storing the consequent parameters of one rule) resp. $(C(p+1)) \times (C(p+1))$ and $C(p+1)$ in case of global learning (storing the consequent parameters of all rules). Regarding antecedent learning, rule evolution and pruning, most of the EFS approaches try to be restricted to have at most cubic complexity in terms of the number of rules plus the number of inputs. This may guarantee some sort of smooth termination in an on-line process, but it is not a necessary pre-requisite and has to be inspected for the particular learning problem at hand.

However, some general remarks on the improvement of computational demands can be given: first of all, the reduction of *unnecessary complexity* such as merging of redundant overlapping rules and pruning of obsolete rules (as discussed in Section 3.3.4) is always beneficial for speeding up the learning process. This also ensures that fuzzy systems are not growing forever, thus restricted in their expansion and virtual memory requests. Second, some fast version of incremental optimization

techniques could be adopted to fuzzy systems estimation, for instance, there exists a fast RLS algorithm for recursively estimating linear parameters in near linear time ($O(nlog(n))$), but with the costs of some stability and convergence, see Douglas (1996); Gay (1996) or Merched and Sayed (1999). Another possibility for decreasing the computation time for learning is the application of active learning for exquisitely selecting only a subset of samples, based on which the model will be updated, please also refer to Section 3.6.1.

3.6.4. *Evaluation Measures*

Evaluation measures may serve as indicators about the actual state of the evolved fuzzy systems, pointing to its accuracy and trustworthiness in predictions. In a data streaming context, the temporal behavior of such measures plays an essential role in order to track the model development over time resp. to realize down-trends in accuracy at an early stage (e.g., caused by drifts), and to react appropriately (e.g., conducting a re-modeling phase, changes at the system setup etc.). Furthermore, the evaluation measures are indispensable during the development phase of EFS approaches. In literature dealing with incremental and data-stream problems (Bifet and Kirkby, 2011), basically three variants of measuring the (progress of) model performance are suggested:

- Interleaved-test-and-then-train
- Periodic hold out test
- Fully-train-and-then-test

Interleaved-test-and-then-train, also termed as *accumulated one-step ahead error/accuracy*, is based on the idea to measure model performance in one-step ahead cycles, i.e., based on one newly loaded sample only. In particular, the following steps are carried out:

(1) Load a new sample (the Nth).
(2) Predict its target $\hat{y}$ using the current evolved fuzzy systems.
(3) Compare the prediction $\hat{y}$ with the true target value y and update the performance measure pm:

$$pm(y, \hat{y})(N) \leftarrow upd(pm(y, \hat{y})(N-1)). \tag{45}$$

(4) Update the evolved fuzzy system (arbitrary approach).
(5) Erase sample and go to Step 1.

This is a rather optimistic approach, assuming that the target response is immediately available for each new sample after its prediction. Often, it may be delayed (Marrs *et al.*, 2012; Subramanian *et al.*, 2013), postponing the update of the model performance to a later stage. Furthermore, in case of single sample updates its

prediction horizon is minimal that makes it difficult to really provide an clear distinction between training and test data, hence weakening their independence. In this sense, this variant is sometimes too optimistic, under-estimating the true model error. On the other hand, all training samples are also used for testing, thus it is quite practicable for small streams.

The periodic holdout procedure can "look ahead" to collect a batch of examples from the stream for use as test examples. In particular, it uses each odd data block for learning and updating the model and each even data block for eliciting the model performance on this latest block; thereby, the data block sizes may be different for training and testing and may vary in dependence of the actual stream characteristics. In this sense, a lower number of samples is used for model updating/tuning than in case of *interleaved-test-and-then-train* procedure. In experimental test designs, where the streams are finite, it is thus more practicable for longer streams. On the other hand, this method would be preferable in scenarios with concept drift, as it would measure a model's ability to adapt to the *latest trends* in the data — whereas in *interleaved-test-and-then-train* procedure all the data seen so far is reflected in the current accuracy/error, becoming less flexible over time. Forgetting may be integrated into Equation (45), but this requires an additional tuning parameter (e.g., a window size in case of sliding windows). The following steps are carried out in a periodic holdout process:

(1) Load a new data block $X_N = \vec{x}_{N*m+1}, \ldots, \vec{x}_{N*m+m}$ containing m samples.
(2) If N is odd:
    (a) Predict the target values $\hat{y}_1, \ldots, \hat{y}_m$ using the current evolved fuzzy systems.
    (b) Compare the predictions $\hat{y}_{N*m+1}, \ldots, \hat{y}_{N*m+m}$ with the true target values $y_{N*m+m}, \ldots, y_{N*m+m}$ and calculate the performance measure (one or more of Equation (46) to Equation (51)) using $\vec{y}$ and $\vec{\hat{y}}$.
    (c) Erase block and go to Step 1.
(3) Else (N is even):
    (a) Update the evolved fuzzy system (arbitrary approach) with all samples in the buffer using the real target values.
    (b) Erase block and go to Step 1.

Last but not least, an alternative to the on-line evaluation procedure is to evolve the model an a certain training set and then evaluate it on an independent test set, termed as *fully-train-and-then-test*. This procedure is most heavily used in many applications of EFS, especially in non-linear system identification and forecasting problems, as summarized in Table 3.3. It extends the prediction horizon of *interleaved-test-and-then-train* procedure with the size of the test set, but only does this in one occasion (at the end of learning). Therefore, it is not useable in drift

cases or severe changes during on-line incremental learning processes and should be only used during development and experimental phases.

Regarding appropriate performance measures, the most convenient choice in classification problems is the number of correctly classified samples (accuracy). In the time instance N (processing the Nth sample), the update of the performance measure as in Equation (45) is then conducted by

$$Acc(N) = \frac{Acc(N-1) * (N-1) + I(\hat{y}, y)}{N}, \tag{46}$$

with $Acc(0) = 0$ and I the indicator function, i.e., $I(a, b) = 1$ whenever $a == b$, otherwise $I(a, b) = 0$, $\hat{y}$ the predicted class label and y the real one. It can be used in the same manner for eliciting the accuracy on whole data blocks as in the periodic hold out case. Another important measure is the so-called *confusion matrix* (Stehman, 1997), which is defined as:

$$C = \begin{bmatrix} N_{11} & N_{12} & \dots & N_{1K} \\ N_{21} & N_{22} & \dots & N_{2K} \\ \vdots & \vdots & \vdots & \vdots \\ N_{K1} & N_{K2} & \dots & N_{KK} \end{bmatrix}, \tag{47}$$

with K the current number of classes, where the diagonal elements denote the number of class j samples which have been correctly classified as class j samples and the element N_{ij} denotes the number of class i samples which are wrongly classified to class j. These can be simply updated by counting.

Furthermore, often someone may be not only interested how strong the samples are miss-classified, but also how certain they are classified (either correctly or wrongly). For instance, a high classification accuracy with a lot of certain classifier outputs may have a higher value than with a lot of uncertain outputs. Furthermore, a high uncertainty degree in the statements point to a lot of conflict cases (compare with Equation (19) and Figure 3.5), i.e., a lot of samples falling into class overlap regions. Therefore, a measure telling the uncertainty degree over samples seen so far is of great interest — a widely-used measure is provided in (Amor *et al.*, 2004):

$$Rel_N = 1 - \frac{1}{K} \sum_{k=1}^{K} |conf_k(N) - y_k(N)|, \tag{48}$$

where $y_k(i) = 1$ if k is the class the current sample N belongs to, and $y_k(i) = 0$ otherwise, and $conf_k$ the certainty level in class k, which can be calculated by Equation (19), for instance. It can be accumulated in the same manner as the accuracy above.

In case of regression problems, the most common choices are the root mean squared error (RMSE), the mean absolute error (MAE) or the average percentual deviation (APE). Their updates are achieved by:

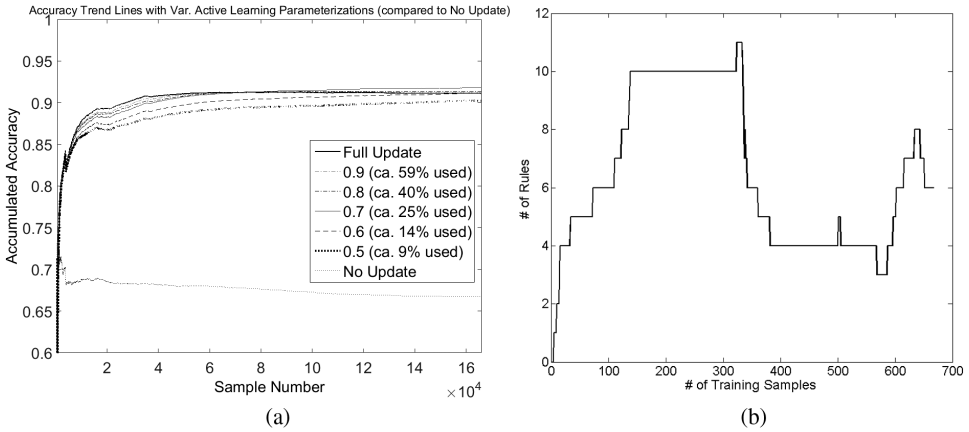
$$\text{RMSE}(N) = \sqrt{\frac{1}{N} * \big((N-1) * (\text{RMSE}(N-1)^2) + (y - \hat{y})^2\big)}. \tag{49}$$

$$\text{MAE}(N) = \frac{1}{N} * \big((N-1) * \text{MAE}(N-1) + |y - \hat{y}|\big). \tag{50}$$

$$\text{APE}(N) = \frac{1}{N} * \left((N-1) * \text{APE}(N-1) + \frac{|y - \hat{y}|}{\text{range}(y)}\right). \tag{51}$$

Their batch calculation for even blocks in periodic hold out test is following the standard procedure and thus can be realized from Wikipedia. Instead of calculating the concrete error values, often the observed versus predicted curves over time and their correlation plots are shown. This gives an even better impression under which circumstances and at which points of time the model behaves in which way. A systematic error shift can be also realized.

Apart from accuracy criteria, other measures rely on the complexity of the models. In most EFS application cases (refer to Table 3.3), the development of the number of rules over time is plotted as a two-dimensional function. Sometimes, also the number of fuzzy sets are reported as additionally criteria which depend on the rule lengths. These mostly depend on the dimensionality of the learning problem. In case of embedded feature selection, there may be a big drop in the number of fuzzy sets once some features are discarded resp. out-weighted (compare with Section 3.4.2). Figure 3.11 shows a typical accumulated accuracy curve over time in the left image



(a) (b)

Figure 3.11: (a) Typical accumulated accuracy increasing over time in case of full update and active learning variants (reducing the number of samples used for updating while still maintaining high accuracy); (b) Typical evolution of the number of rules over time.

(using an all-pairs EFC including active learning option with different amount of samples for updating) and a typical development of the number of rules in the right one: At the start, rules are evolved and accumulated, later some rules turned out to be superfluous and hence are back-pruned and merged. This guarantees an anytime flexibility.

3.6.5. *Real-World Applications of EFS — Overview*

Due to space restrictions, a complete description of application scenarios in which EFS have been successfully implemented and used so far is simply impossible. Thus, we restrict ourselves to a compact summary within Table 3.3 showing application types and classes and indicating which EFS approaches have been used in the

Table 3.3: Application classes in which EFS approaches have been successfully applied so far

Application Type/Class (alphabetically)	EFS approaches (+ refs)	Comment
Active learning / human-machine interaction	FLEXFIS-Class (Lughofer *et al.*, 2009; Lughofer, 2012d), EFC-AP (Lughofer, 2012a), FLEXFIS-PLS (Cernuda *et al.*, 2014)	Reducing the annotation effort and measurement costs in industrial processes
Adaptive on-line control	evolving PID and MRC controllers in (Angelov and Skrjanc, 2013), eFuMo (Zdsar *et al.*, 2014), rGK (Dovzan *et al.*, 2012), self-evolving NFC (Cara *et al.*, 2013), adaptive controller in (Rong *et al.*, 2014).	Design of fuzzy controllers which can be updated and evolved on-the-fly
Bioinformatics	EFuNN (Kasabov, 2002)	Specific applications such as ribosome binding site (RBS) identification, gene profiling
Chemometric Modeling and Process Control	FLEXFIS++ (Cernuda *et al.*, 2013, 2012); the approach in Bodyanskiy and Vynokurova (2013)	The application of EFS onto processes in chemical industry (high-dim. NIR spectra)
EEG signals classification and processing	eTS (Xydeas *et al.*, 2006), epSNNr (Nuntalid *et al.*, 2011)	Time-series modeling with the inclusion of time delays
Evolving Smart Sensors (eSensors)	eTS+ (Macias-Hernandez and Angelov, 2010) (gas industry), (Angelov and Kordon, 2010a, 2010b) (chemical process industry), FLEXFIS (Lughofer *et al.*, 2011c) and PANFIS (Pratama *et al.*, 2014a) (NOx emissions)	Evolving predictive and forecasting models in order to substitute cost-intensive hardware sensors

(*Continued*)

Table 3.3: (*Continued*)

Application Type/Class (alphabetically)	EFS approaches (+ refs)	Comment
Forecasting and prediction (general)	AHLTNM (Kalhor *et al.*, 2010) (daily temp.), eT2FIS (Tung *et al.*, 2013) (traffic flow), eFPT (Shaker *et al.*, 2013) (Statlog from UCI), eFT (Lemos *et al.*, 2011b) and eMG (Lemos *et al.*, 2011a) (short-term electricity load), FLEXFIS+ (Lughofer *et al.*, 2011b) and GENEFIS (Pratama *et al.*, 2014b) (house prices), LOLIMOT inc. (Hametner and Jakubek, 2013) (maximum cylinder pressure), rGK (Dovzan *et al.*, 2012) (sales prediction) and others	Various successful implementations of EFS
Financial domains	eT2FIS (Tung *et al.*, 2013), evolving granular systems (Leite *et al.*, 2012b), ePL (Maciel *et al.*, 2012), PANFIS (Pratama *et al.*, 2014a), SOFNN (Prasad *et al.*, 2010)	Time-series modeling with the inclusion of time delays
Identification of dynamic benchmark problems	DENFIS (Kasabov and Song, 2002), eT2FIS (Tung *et al.*, 2013), eTS+ (Angelov, 2010), FLEXFIS (Lughofer, 2008), SAFIS (Rong, 2012), SEIT2FNN (Juang and Tsao, 2008), SOFNN (Prasad *et al.*, 2010)	Mackey-Glass, Box-Jenkins, etc.
On-line fault detection and condition monitoring	eMG for classification (Lemos *et al.*, 2013), FLEXFIS++ (Lughofer and Guardiola, 2008b; Serdio *et al.*, 2014a), rGK (Dovzan *et al.*, 2012)	EFS applied as SysID models for extracting residuals
On-line monitoring	eTS+ (Macias-Hernandez and Angelov, 2010) (gas industry)	Supervision of system behaviors
Robotics	eTS+ (Zhou and Angelov, 2007)	In the area of self-localization
Time-series modeling	DENFIS (Widiputra *et al.*, 2012), ENFM (Soleimani *et al.*, 2010) and eTS-LS-SVM (Komijani *et al.*, 2012) (sun spot)	Local modeling of multiple time-series versus instance-based learning
User behavior identification	eClass and eTS (Angelov *et al.*, 2012; Iglesias *et al.*, 2010), eTS+ (Andreu and Angelov, 2013), FPA (Wang *et al.*, 2013)	Analysis of the user's behaviors in multi-agent systems, on computers, indoor environments etc.
Video processing	eTS, eTS+ (Angelov *et al.*, 2011; Zhou and Angelov, 2006)	Including real-time object id., obstacles tracking and novelty detection
Visual quality control	EFC-AP (Lughofer and Buchtala, 2013), FLEXFIS-Class (Eitzinger *et al.*, 2010; Lughofer, 2010b), pClass (Pratama *et al.*, 2014c)	Image classification tasks based on feature vectors

circumstance of which application type. In all of these cases, EFS(C) helped to increase automatization capability, improving performance of the models and finally increasing the useability of the whole systems; in some cases, no modeling has been (could be) applied before at all.

Acknowledgments

This work was funded by the research programme at the LCM GmbH as part of a K2 project. K2 projects are financed using funding from the Austrian COMET-K2 programme. The COMET K2 projects at LCM are supported by the Austrian federal government, the federal state of Upper Austria, the Johannes Kepler University and all of the scientific partners which form part of the K2-COMET Consortium. This publication reflects only the authors' views.

References

Abonyi, J., Babuska, R. and Szeifert, F. (2002). Modified Gath–Geva fuzzy clustering for identification of Takagi-Sugeno fuzzy models. *IEEE Trans. Syst., Man Cybern. Part B*, 32(5), pp. 612–621.

Abonyi, J. (2003). *Fuzzy Model Identification for Control*. Boston, U.S.A.: Birkhäuser.

Abraham, W. and Robins, A. (2005). Memory retention: The synaptic stability versus plasticity dilemma. *Trends Neurosci.*, 28(2), pp. 73–78.

Affenzeller, M., Winkler, S., Wagner, S. and Beham, A. (2009). *Genetic Algorithms and Genetic Programming*: *Modern Concepts and Practical Applications*. Boca Raton, Florida: Chapman & Hall.

Allwein, E., Schapire, R. and Singer, Y. (2001). Reducing multiclass to binary: a unifying approach for margin classifiers. *J. Mach. Learn. Res.*, 1, pp. 113–141.

Almaksour, A. and Anquetil, E. (2011). Improving premise structure in evolving Takagi–Sugeno neuro-fuzzy classifiers. *Evolving Syst.*, 2, pp. 25–33.

Amor, N., Benferhat, S. and Elouedi, Z. (2004). Qualitative classification and evaluation in possibilistic decision trees. In *Proc. FUZZ-IEEE Conf.*, Budapest, Hungary, pp. 653–657.

Andreu, J. and Angelov, P. (2013). Towards generic human activity recognition for ubiquitous applications. *J. Ambient Intell. Human Comput.*, 4, pp. 155–156.

Angelov, P. (2010). Evolving Takagi–Sugeno fuzzy systems from streaming data, eTS+. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems*: *Methodology and Applications*. New York: John Wiley & Sons, pp. 21–50.

Angelov, P. and Filev, D. (2004). An approach to online identification of Takagi–Sugeno fuzzy models. *IEEE Trans. Syst. Man Cybern., part B*: *Cybern.*, 34(1), pp. 484–498.

Angelov, P. and Kasabov, N. (2005). Evolving computational intelligence systems. In *Proc. of the 1st Int. Workshop on Genet. Fuzzy Syst.*, Granada, Spain, pp. 76–82.

Angelov, P. and Kordon, A. (2010a). Evolving inferential sensors in the chemical process industry. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems*: *Methodology and Applications*. New York: John Wiley & Sons, pp. 313–336.

Angelov, P. and Kordon, A. (2010b). Adaptive inferential sensors based on evolving fuzzy models: An industrial case study. *IEEE Trans. Syst., Man Cybern., part B*: *Cybern.*, 40(2), pp. 529–539.

Angelov, P. and Skrjanc, I. (2013). Robust evolving cloud-based controller for a hydraulic plant. In *Proc. 2013 IEEE Conf. Evolving Adapt. Intell. Syst. (EAIS)*. Singapore, pp. 1–8.

Angelov, P., Filev, D. and Kasabov, N. (2010). *Evolving Intelligent Systems — Methodology and Applications*. New York: John Wiley & Sons.

Angelov, P., Ledezma, A. and Sanchis, A. (2012). Creating evolving user behavior profiles automatically. *IEEE Trans. Knowl. Data Eng.*, 24(5), pp. 854–867.

Angelov, P., Lughofer, E. and Zhou, X. (2008). Evolving fuzzy classifiers using different model architectures. *Fuzzy Sets Syst.*, 159(23), pp. 3160–3182.

Angelov, P., Sadeghi-Tehran, P. and Ramezani, R. (2011). An approach to automatic real-time novelty detection, object identification, and tracking in video streams based on recursive density estimation and evolving Takagi–Sugeno fuzzy systems. *Int. J. Intell. Syst.*, 26(3), pp. 189–205.

Aström, K. and Wittenmark, B. (1994). *Adaptive Control Second Edition*. Boston, MA, USA: Addison-Wesley Longman Publishing Co. Inc.

Babuska, R. (1998). *Fuzzy Modeling for Control*. Norwell, Massachusetts: Kluwer Academic Publishers.

Balas, V., Fodor, J. and Varkonyi-Koczy, A. (2009). *Soft Computing based Modeling in Intelligent Systems*. Berlin, Heidelberg: Springer.

Bifet, A., Holmes, G., Kirkby, R. and Pfahringer, B. (2010). MOA: Massive online analysis, *J. Mach. Learn. Res.*, 11, pp. 1601–1604.

Bifet, A. and Kirkby, R. (2011). Data stream mining — a practical approach. Technical report, University of Waikato, Japan, Department of Computer Sciences.

Bikdash, M. (1999). A highly interpretable form of sugeno inference systems. *IEEE Trans. Fuzzy Syst.*, 7(6), pp. 686–696.

Bishop, C. (2007). *Pattern Recognition and Machine Learning*. New York: Springer.

Bodyanskiy, Y. and Vynokurova, O. (2013). Hybrid adaptive wavelet-neuro-fuzzy system for chaotic time series identification, *Inf. Sci.*, 220, pp. 170–179.

Bouchachia, A. (2009). Incremental induction of classification fuzzy rules. In *IEEE Workshop Evolving Self-Dev. Intell. Syst. (ESDIS) 2009*. Nashville, U.S.A., pp. 32–39.

Bouchachia, A. (2011). Evolving clustering: An asset for evolving systems. *IEEE SMC Newsl.*, 36.

Bouchachia, A. and Mittermeir, R. (2006). Towards incremental fuzzy classifiers. *Soft Comput.*, 11(2), pp. 193–207.

Bouchachia, A., Lughofer, E. and Mouchaweh, M. (2014). Editorial to the special issue: Evolving soft computing techniques and applications. *Appl. Soft Comput.*, 14, pp. 141–143.

Bouchachia, A., Lughofer, E. and Sanchez, D. (2013). Editorial to the special issue: Online fuzzy machine learning and data mining. *Inf. Sci.*, 220, pp. 1–4.

Bouillon, M., Anquetil, E. and Almaksour, A. (2013). Decremental learning of evolving fuzzy inference systems: Application to handwritten gesture recognition. In Perner, P. (ed.), *Machine Learning and Data Mining in Pattern Recognition*, 7988, *Lecture Notes in Computer Science*. New York: Springer, pp. 115–129.

Breiman, L., Friedman, J., Stone, C. and Olshen, R. (1993). *Classification and Regression Trees*. Boca Raton: Chapman and Hall.

Cara, A., Herrera, L., Pomares, H. and Rojas, I. (2013). New online self-evolving neuro fuzzy controller based on the tase-nf model. *Inf. Sci*., 220, pp. 226–243.

Carr, V. and Tah, J. (2001). A fuzzy approach to construction project risk assessment and analysis: construction project risk management system. *Adv. Eng. Softw*., 32(10–11).

Casillas, J., Cordon, O., Herrera, F. and Magdalena, L. (2003). *Interpretability Issues in Fuzzy Modeling*. Berlin Heidelberg: Springer Verlag.

Castro, J. and Delgado, M. (1996). Fuzzy systems with defuzzification are universal approximators. *IEEE Trans. Syst. Man Cybern. part B*: *Cybern*., 26(1), pp. 149–152.

Cernuda, C., Lughofer, E., Hintenaus, P., Märzinger, W., Reischer, T., Pawlicek, M. and Kasberger, J. (2013). Hybrid adaptive calibration methods and ensemble strategy for prediction of cloud point in melamine resin production. *Chemometr. Intell. Lab. Syst*., 126, pp. 60–75.

Cernuda, C., Lughofer, E., Mayr, G., Röder, T., Hintenaus, P., Märzinger, W. and Kasberger, J. (2014). Incremental and decremental active learning for optimized self-adaptive calibration in viscose production. *Chemometr. Intell. Lab. Syst*., 138, pp. 14–29.

Cernuda, C., Lughofer, E., Suppan, L., Röder, T., Schmuck, R., Hintenaus, P., Märzinger, W. and Kasberger, J. (2012). Evolving chemometric models for predicting dynamic process parameters in viscose production. *Anal. Chim. Acta*, 725, pp. 22–38.

Chapelle, O., Schoelkopf, B. and Zien, A. (2006). *Semi-Supervised Learning*. Cambridge, MA: MIT Press.

Cho, J. and Park, D. (2000). Novel fuzzy logic control based on weighting of partially inconsistent rules using neural network. *J. Intell. Fuzzy Syst*., 8(2), pp. 99–110.

Chong, C.-Y. and Kumar, S. (2003). Sensor networks: Evolution, opportunities, and challenges. *Proc. of the IEEE*, 91(8), pp. 1247–1256.

Chu, W., Zinkevich, M., Li, L., Thomas, A. and Zheng, B. (2011). Unbiased online active learning in data streams. In *Proc. KDD 2011*. San Diego, California.

Cleveland, W. and Devlin, S. (1988). Locally weighted regression: An approach to regression analysis by local fitting. *J. Am. Stat. Assoc*., 84(403), pp. 596–610.

Cohn, D., Atlas, L. and Ladner, R. (1994). Improving generalization with active learning. *Mach. Learn*., 15(2), pp. 201–221.

Day, N. E. (1969). Estimating the components of a mixture of normal distributions. *Biometrika*, 56(463–474).

Diehl, C. and Cauwenberghs, G. (2003). SVM incremental learning, adaptation and optimization. In *Proc. Int. Joint Conf. Neural Netw.*, Boston, 4, pp. 2685–2690.

Douglas, S. (1996). Efficient approximate implementations of the fast affine projection algorithm using orthogonal transforms. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Atlanta, Georgia, pp. 1656–1659.

Dovzan, D. and Skrjanc, I. (2011). Recursive clustering based on a Gustafson–Kessel algorithm. *Evolving Syst*., 2(1), pp. 15–24.

Dovzan, D., Logar, V. and Skrjanc, I. (2012). Solving the sales prediction problem with fuzzy evolving methods. In *WCCI 2012 IEEE World Congr. Comput. Intell.*, Brisbane, Australia.

Duda, R., Hart, P. and Stork, D. (2000). *Pattern Classification, Second Edition*. Southern Gate, Chichester, West Sussex, England: Wiley-Interscience (John Wiley & Sons).

Dy, J. and Brodley, C. (2004). Feature selection for unsupervised learning. *J. Mach. Learn. Res*., 5, pp. 845–889.

Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap*. Boca Raton, Florida: Chapman & Hall/CRC.

Eitzinger, C., Heidl, W., Lughofer, E., Raiser, S., Smith, J., Tahir, M., Sannen, D. and van Brussel, H. (2010). Assessment of the influence of adaptive components in trainable surface inspection systems. *Mach. Vis. Appl*., 21(5), pp. 613–626.

Fürnkranz, J. (2001). Round robin rule learning. In *Proc. Int. Conf. Mach. Learn. (ICML 2011)*, Williamstown, MA, pp. 146–153.

Fürnkranz, J. (2002). Round robin classification. *J. Mach. Learn. Res*., 2, pp. 721–747.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends Cogn. Sci*., 3(4), pp. 128–135.

Fuller, R. (1999). *Introduction to Neuro-Fuzzy Systems*. Heidelberg, Germany: Physica-Verlag.

Gacto, M., Alcala, R. and Herrera, F. (2011). Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Inf. Sci*., 181(20), pp. 4340–4360.

Gama, J. (2010). *Knowledge Discovery from Data Streams*. Boca Raton, Florida: Chapman & Hall/CRC.

Gan, G., Ma, C. and Wu, J. (2007). *Data Clustering*: *Theory, Algorithms, and Applications* (*Asa-Siam Series on Statistics and Applied Probability*). U.S.A.: Society for Industrial & Applied Mathematics.

Gay, S. L. (1996). Dynamically regularized fast recursive least squares with application to echo cancellation. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Atalanta, Georgia, pp. 957–960.

Hametner, C. and Jakubek, S. (2013). Local model network identification for online engine modelling. *Inf. Sci*., 220, pp. 210–225.

Hamker, F. (2001). RBF learning in a non-stationary environment: the stability-plasticity dilemma. In Howlett, R. and Jain, L. (eds.), *Radial Basis Function Networks* 1*: Recent Developments in Theory and Applications*, Heidelberg, New York: Physica Verlag, pp. 219–251.

Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning*: *Data Mining, Inference and Prediction, Second Edition*. New York Berlin Heidelberg: Springer.

Haykin, S. (1999). *Neural Networks*: *A Comprehensive Foundation, 2nd Edition*. Upper Saddle River, New Jersey: Prentice Hall Inc.

He, H. and Garcia, E. (2009). Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.*, 21(9), pp. 1263–1284.

Hentzgen, S., Strickert, M. and Huellermeier, E. (2014). Visualization of evolving fuzzy rule-based systems. *Evolving Syst*., DOI: 10.1007/s12530-014-9110-4, on–line and in press.

Henzgen, S., Strickert, M. and Hüllermeier, E. (2013). Rule chains for visualizing evolving fuzzy rule-based systems. In *Advances in Intelligent Systems and Computing*, 226, *Proc. Eighth Int. Conf. Comput. Recognit. Syst. CORES 2013*. Cambridge, MA: Springer, pp. 279–288.

Herrera, L., Pomares, H., Rojas, I., Valenzuela, O. and Prieto, A. (2005). TaSe, a taylor series-based fuzzy system model that combines interpretability and accuracy. *Fuzzy Sets Syst*., 153(3), pp. 403–427.

Hisada, M., Ozawa, S., Zhang, K. and Kasabov, N. (2010). Incremental linear discriminant analysis for evolving feature spaces in multitask pattern recognition problems, *Evolving Syst*., 1(1), pp. 17–27.

Ho, W., Tung, W. and Quek, C. (2010). An evolving Mamdani–Takagi–Sugeno based neural-fuzzy inference system with improved interpretability–accuracy. In *Proc. WCCI 2010 IEEE World Congr. Comput. Intell.*, Barcelona, pp. 682–689.

Holmblad, L. and Ostergaard, J. (1982). Control of a cement kiln by fuzzy logic. *Fuzzy Inf. Decis. Process*., pp. 398–409.

Huang, Z., Gedeon, T. D. and Nikravesh, M. (2008). Pattern trees induction: A new machine learning method. *IEEE Trans. Fuzzy Syst*., 16(4), pp. 958–970.

Hüllermeier, E. and Brinker, K. (2008). Learning valued preference structures for solving classification problems. *Fuzzy Sets Syst*., 159(18), pp. 2337–2352.

Hühn, J. and Hüllermeier, E. (2009). FR3: A fuzzy rule learner for inducing reliable classifiers, *IEEE Trans. Fuzzy Syst*., 17(1), pp. 138–149.

Iglesias, J., Angelov, P., Ledezma, A. and Sanchis, A. (2010). Evolving classification of agent's behaviors: a general approach. *Evolving Syst*., 1(3), pp. 161–172.

Ishibuchi, H. and Nakashima, T. (2001). Effect of rule weights in fuzzy rule-based classification systems. *IEEE Trans. Fuzzy Syst*., 9(4), pp. 506–515.

Jang, J.-S. (1993). ANFIS: Adaptive-network-based fuzzy inference systems. *IEEE Trans. Syst., Man Cybern.*, 23(3), pp. 665–685.

Juang, C. and Tsao, Y. (2008). A self-evolving interval type-2 fuzzy neural network with on-line structure and parameter learning. *IEEE Trans. Fuzzy Syst*., 16(6), pp. 1411–1424.

Kaczmarz, S. (1993). Approximate solution of systems of linear equations. *Int. J. Control*, 53, pp. 1269–1271.

Kalhor, A., Araabi, B. and Lucas, C. (2010). An online predictor model as adaptive habitually linear and transiently nonlinear model. *Evolving Syst*., 1(1), pp. 29–41.

Karer, G. and Skrjanc, I. (2013). *Predictive Approaches to Control of Complex Systems*. Berlin, Heidelberg: Springer Verlag.

Karnik, N. and Mendel, J. (2001). Centroid of a type-2 fuzzy set. *Inf. Sci*., 132(1–4), pp. 195–220.

Kasabov, N. (2002). *Evolving Connectionist Systems — Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines*. London: Springer Verlag.

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

Klement, E., Mesiar, R. and Pap, E. (2000). *Triangular Norms*. Dordrecht Norwell New York London: Kluwer Academic Publishers.

Klinkenberg, R. (2004). Learning drifting concepts: example selection vs. example weighting, *Intell. Data Anal*., 8(3), pp. 281–300.

Koenig, S., Likhachev, M., Liu, Y. and Furcy, D. (2004). Incremental heuristic search in artificial intelligence. *Artif. Intell. Mag*., 25(2), pp. 99–112.

Komijani, M., Lucas, C., Araabi, B. and Kalhor, A. (2012). Introducing evolving Takagi–Sugeno method based on local least squares support vector machine models. *Evolving Syst*. 3(2), pp. 81–93.

Kruse, R., Gebhardt, J. and Palm, R. (1994). *Fuzzy Systems in Computer Science*. Wiesbaden: Verlag Vieweg.

Kuncheva, L. (2000). *Fuzzy Classifier Design*. Heidelberg: Physica-Verlag.

Leekwijck, W. and Kerre, E. (1999). Defuzzification: criteria and classification. *Fuzzy Sets Syst*. 108(2), pp. 159–178.

Leite, D., Ballini, R., Costa, P. and Gomide, F. (2012a). Evolving fuzzy granular modeling from nonstationary fuzzy data streams. *Evolving Syst*. 3(2), pp. 65–79.

Leite, D., Costa, P. and Gomide, F. (2012b). Interval approach for evolving granular system modeling. In Sayed-Mouchaweh, M. and Lughofer, E. (eds.), *Learning in Non-Stationary Environments*: *Methods and Applications*. New York: Springer, pp. 271–300.

Lemos, A., Caminhas, W. and Gomide, F. (2011a). Multivariable gaussian evolving fuzzy modeling system. *IEEE Trans. Fuzzy Syst*., 19(1), pp. 91–104.

Lemos, A., Caminhas, W. and Gomide, F. (2011b). Fuzzy evolving linear regression trees. *Evolving Syst*., 2(1), pp. 1–14.

Lemos, A., Caminhas, W. and Gomide, F. (2013). Adaptive fault detection and diagnosis using an evolving fuzzy classifier. *Inf. Sci*., 220, pp. 64–85.

Leng, G., McGinnity, T. and Prasad, G. (2005). An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network. *Fuzzy Sets Syst*., 150(2), pp. 211–243.

Leng, G., Zeng, X.-J. and Keane, J. (2012). An improved approach of self-organising fuzzy neural network based on similarity measures. *Evolving Syst*., 3(1), pp. 19–30.

Leondes, C. (1998). *Fuzzy Logic and Expert Systems Applications* (*Neural Network Systems Techniques and Applications*). San Diego, California: Academic Press.

Li, Y. (2004). On incremental and robust subspace learning. *Pattern Recognit*., 37(7), pp. 1509–1518.

Liang, Q. and Mendel, J. (2000). Interval type-2 fuzzy logic systems: Theory and design, *IEEE Trans. Fuzzy Syst*., 8(5), pp. 535–550.

Lima, E., Hell, M., Ballini, R. and Gomide, F. (2010). Evolving fuzzy modeling using participatory learning. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems*: *Methodology and Applications*. New York: John Wiley & Sons, pp. 67–86.

Lippmann, R. (1991). A critical overview of neural network pattern classifiers. In *Proc. IEEE Workshop Neural Netw. Signal Process.*, pp. 266–275.

Ljung, L. (1999). *System Identification*: *Theory for the User*. Upper Saddle River, New Jersey: Prentice Hall PTR, Prentic Hall Inc.

Lughofer, E., Smith, J. E., Caleb-Solly, P., Tahir, M., Eitzinger, C., Sannen, D. and Nuttin, M. (2009). Human-machine interaction issues in quality control based on on-line image classification. *IEEE Trans. Syst., Man Cybern., part A*: *Syst., Humans*, 39(5), pp. 960–971.

Lughofer, E. (2008). FLEXFIS: A robust incremental learning approach for evolving TS fuzzy models. *IEEE Trans. Fuzzy Syst*., 16(6), pp. 1393–1410.

Lughofer, E. (2010a). Towards robust evolving fuzzy systems. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems*: *Methodology and Applications*. New York: John Wiley & Sons, pp. 87–126.

Lughofer, E. (2010b). On-line evolving image classifiers and their application to surface inspection, *Image Vis. Comput*., 28(7), 1065–1079.

Lughofer, E. (2011a). Human-inspired evolving machines — the next generation of evolving intelligent systems? *SMC Newsl*., 36.

Lughofer, E. (2011b). *Evolving Fuzzy Systems — Methodologies, Advanced Concepts and Applications*. Berlin Heidelberg: Springer.

Lughofer, E. (2011c). On-line incremental feature weighting in evolving fuzzy classifiers. *Fuzzy Sets Syst*., 163(1), pp. 1–23.

Lughofer, E. (2012a). Single-pass active learning with conflict and ignorance. *Evolving Syst*., 3(4), pp. 251–271.

Lughofer, E. (2012b). Flexible evolving fuzzy inference systems from data streams (FLEXFIS++). In Sayed-Mouchaweh, M. and Lughofer, E. (eds.), *Learning in Non-Stationary Environments*: *Methods and Applications*. New York: Springer, pp. 205–246.

Lughofer, E. and Sayed-Mouchaweh, M. (2015). Autonomous data stream clustering implementing incremental split-and-merge techniques — Towards a plug-and-play approach. *Information Sciences*, 204, pp. 54–79.

Lughofer, E. (2012d). Hybrid active learning (HAL) for reducing the annotation efforts of operators in classification systems, *Pattern Recognit*., 45(2), pp. 884–896.

Lughofer, E. (2013). On-line assurance of interpretability criteria in evolving fuzzy systems — achievements, new concepts and open issues. *Inf. Sci*., 251, pp. 22–46.

Lughofer, E. and Angelov, P. (2011). Handling drifts and shifts in on-line data streams with evolving fuzzy systems. *Appl. Soft Comput*., 11 (2), pp. 2057–2068.

Lughofer, E. and Buchtala, O. (2013). Reliable all-pairs evolving fuzzy classifiers. *IEEE Trans. Fuzzy Syst*., 21(4), pp. 625–641.

Lughofer, E. and Guardiola, C. (2008a). Applying evolving fuzzy models with adaptive local error bars to on-line fault detection. In *Proc. Genet. Evolving Fuzzy Syst., 2008*. Witten-Bommerholz, Germany, pp. 35–40.

Lughofer, E. and Guardiola, C. (2008b). On-line fault detection with data-driven evolving fuzzy models. *J. Control Intell. Syst*., 36(4), pp. 307–317.

Lughofer, E. and Hüllermeier, E. (2011). On-line redundancy elimination in evolving fuzzy regression models using a fuzzy inclusion measure. In *Proc. EUSFLAT 2011 Conf*., Aix-Les-Bains, France: Elsevier, pp. 380–387.

Lughofer, E., Bouchot, J.-L. and Shaker, A. (2011a). On-line elimination of local redundancies in evolving fuzzy systems. *Evolving Syst*., 2(3), pp. 165–187.

Lughofer, E., Trawinski, B., Trawinski, K., Kempa, O. and Lasota, T. (2011b). On employing fuzzy modeling algorithms for the valuation of residential premises. *Inf. Sci*., 181(23), pp. 5123–5142.

Lughofer, E., Macian, V., Guardiola, C. and Klement, E. (2011c). Identifying static and dynamic prediction models for NOx emissions with evolving fuzzy systems, *Appl. Soft Comput*., 11(2), pp. 2487–2500.

Lughofer, E., Cernuda, C. and Pratama, M. (2013). Generalized flexible fuzzy inference systems. In *Proc. ICMLA 2013 Conf.*, Miami, Florida, pp. 1–7.

Lughofer, E., Cernuda, C., Kindermann, S. and Pratama, M. (2014). Generalized smart evolving fuzzy systems. *Evolving Syst*., online and in press, doi: 10.1007/s12530–015–9132–6

Macias-Hernandez, J. and Angelov, P. (2010). Applications of evolving intelligent systems to the oil and gas industry. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems*: *Methodology and Applications*. New York: John Wiley & Sons, pp. 401–421.

Maciel, L., Lemos, A., Gomide, F. and Ballini, R. (2012). Evolving fuzzy systems for pricing fixed income options. *Evolving Syst*., 3(1), pp. 5–18.

Mamdani, E. (1977). Application of fuzzy logic to approximate reasoning using linguistic systems. *Fuzzy Sets Syst*. 26(12), pp. 1182–1191.

Marrs, G., Black, M. and Hickey, R. (2012). The use of time stamps in handling latency and concept drift in online learning. *Evolving Syst*., 3(2), pp. 203–220.

Mendel, J. (2001). *Uncertain Rule-Based Fuzzy Logic Systems*: *Introduction and New Directions*. Upper Saddle River: Prentice Hall.

Mendel, J. and John, R. (2002). Type-2 fuzzy sets made simple. *IEEE Trans. Fuzzy Syst*., 10(2), pp. 117–127.

Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. R. Soc. A*, 209, pp. 441–458.

Merched, R. and Sayed, A. (1999). Fast RLS laguerre adaptive filtering. In *Proc. Allerton Conf. Commun., Control Comput*., Allerton, IL, pp. 338–347.

Moe-Helgesen, O.-M. and Stranden, H. (2005). Catastophic forgetting in neural networks. Technical report. Trondheim, Norway: Norwegian University of Science and Technology.

Morreale, P., Holtz, S. and Goncalves, A. (2013). Data mining and analysis of large scale time series network data. In *Proc. 27th Int. Conf. Adv. Inf. Netw. Appl. Workshops (WAINA)*. Barcelona, Spain, pp. 39–43.

Mouss, H., Mouss, D., Mouss, N. and Sefouhi, L. (2004). Test of Page-Hinkley, an approach for fault detection in an agro-alimentary production system. In *Proc. Asian Control Conf.*, 2, pp. 815–818.

Nakashima, Y. Y. T., Schaefer, G. and Ishibuchi, H. (2006). A weighted fuzzy classifier and its application to image processing tasks. *Fuzzy Sets Syst*., 158(3), pp. 284–294.

Nauck, D. and Kruse, R. (1998). NEFCLASS-X — a soft computing tool to build readable fuzzy classifiers. *BT Technol. J*., 16(3), pp. 180–190.

Nelles, O. (2001). *Nonlinear System Identification*. Berlin: Springer.

Ngia, L. and Sjöberg, J. (2000). Efficient training of neural nets for nonlinear adaptive filtering using a recursive Levenberg–Marquardt algorithm. *IEEE Trans. Signal Process*. 48(7), pp. 1915–1926.

Nguyen, H., Sugeno, M., Tong, R. and Yager, R. (1995). *Theoretical Aspects of Fuzzy Control*. New York: John Wiley & Sons.

Nuntalid, N., Dhoble, K. and Kasabov, N. (2011). EEG classification with BSA spike encoding algorithm and evolving probabilistic spiking neural network. In *Neural Inf. Process., LNCS 7062*. Berlin Heidelberg: Springer Verlag, pp. 451–460.

Pal, N. and Pal, K. (1999). Handling of inconsistent rules with an extended model of fuzzy reasoning. *J. Intell. Fuzzy Syst*., 7, pp. 55–73.

Pedrycz, W. and Gomide, F. (2007). *Fuzzy Systems Engineering*: *Toward Human-Centric Computing*. Hoboken, New Jersey: John Wiley & Sons.

Piegat, A. (2001). *Fuzzy Modeling and Control*. Heidelberg, New York: Physica Verlag, Springer Verlag Company.

Prasad, G., Leng, G., McGuinnity, T. and Coyle, D. (2010). Online identification of self-organizing fuzzy neural networks for modeling time-varying complex systems. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems*: *Methodology and Applications*. New York: John Wiley & Sons, pp. 201–228.

Pratama, M., Anavatti, S., Angelov, P. and Lughofer, E. (2014a). PANFIS: A novel incremental learning machine. *IEEE Trans. Neural Netw. Learn. Syst*., 25(1), pp. 55–68.

Pratama, M., Anavatti, S. and Lughofer, E. (2014b). GENEFIS: Towards an effective localist network. *IEEE Trans. Fuzzy Syst*., 22(3), pp. 547–562.

Pratama, M., Anavatti, S. and Lughofer, E. (2014c). pClass: An effective classifier to streaming examples. *IEEE Transactions on Fuzzy Systems*, 23(2), pp. 369–386.

Quinlan, J. R., *C4.5*: *Programs for Machine Learning*. San Francisco, CA: Morgan Kaufmann Publishers.

Reichmann, O., Jones, M. and Schildhauer, M. (2011). Challenges and opportunities of open data in ecology. *Science*, 331(6018), pp. 703–705.

Reveiz, A. and Len, C. (2010). Operational risk management using a fuzzy logic inference system. *J. Financ. Transf.* 30, pp. 141–153.

Riaz, M. and Ghafoor, A. (2013). Spectral and textural weighting using Takagi–Sugeno fuzzy system for through wall image enhancement. *Prog. Electromagn. Res. B*., 48, pp. 115–130.

Rong, H.-J. (2012). Sequential adaptive fuzzy inference system for function approximation problems. In Sayed-Mouchaweh, M. and Lughofer, E. (eds.), *Learning in Non-Stationary Environments*: *Methods and Applications*. New York: Springer.

Rong, H.-J., Han, S. and Zhao, G.-S. (2014). Adaptive fuzzy control of aircraft wing-rock motion. *Appl. Soft Comput*., 14, pp. 181–193.

Rong, H.-J., Sundararajan, N., Huang, G.-B. and Saratchandran, P. (2006). Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. *Fuzzy Sets Syst*., 157(9), pp. 1260–1275.

Rong, H.-J., Sundararajan, N., Huang, G.-B. and Zhao, G.-S. (2011). Extended sequential adaptive fuzzy inference system for classification problems. *Evolving Syst*., 2(2), pp. 71–82.

Rosemann, N., Brockmann, W. and Neumann, B. (2009). Enforcing local properties in online learning first order TS fuzzy systems by incremental regularization. In *Proc. IFSA-EUSFLAT 2009*. Lisbon, Portugal, pp. 466–471.

Rubio, J. (2009). SOFMLS: Online self-organizing fuzzy modified least square network. *IEEE Trans. Fuzzy Syst*., 17(6), pp. 1296–1309.

Rubio, J. (2010). Stability analysis for an on-line evolving neuro-fuzzy recurrent network. In Angelov, P., Filev, D. and Kasabov, N. (eds.), *Evolving Intelligent Systems*: *Methodology and Applications*. New York: John Wiley & Sons, pp. 173–199.

Saminger-Platz, S., Mesiar, R. and Dubois, D. (2007). Aggregation operators and commuting. *IEEE Trans. Fuzzy Syst*., 15(6), pp. 1032–1045.

Sayed-Mouchaweh, M. and Lughofer, E. (2012). *Learning in Non-Stationary Environments*: *Methods and Applications*. New York: Springer.

Schölkopf, B. and Smola, A. (2002). *Learning with Kernels — Support Vector Machines, Regularization, Optimization and Beyond*. London, England: MIT Press.

Sculley, D. (2007). Online active learning methods for fast label efficient spam filtering. In *Proc. Fourth Conf. Email AntiSpam*. Mountain View, California.

Sebastiao, R., Silva, M., Rabico, R., Gama, J. and Mendonca, T. (2013). Real-time algorithm for changes detection in depth of anesthesia signals. *Evolving Syst*., 4(1), pp. 3–12.

Senge, R. and Huellermeier, E. (2011). Top–down induction of fuzzy pattern trees. *IEEE Trans. Fuzzy Syst*., 19(2), pp. 241–252.

Serdio, F., Lughofer, E., Pichler, K., Buchegger, T. and Efendic, H. (2014a). Residual-based fault detection using soft computing techniques for condition monitoring at rolling mills. *Inf. Sci*., 259, pp. 304–320.

Serdio, F., Lughofer, E., Pichler, K., Pichler, M., Buchegger, T. and Efendic, H. (2014b). Fault detection in multi-sensor networks based on multivariate time-series models and orthogonal transformations. *Inf. Fusion*, 20, pp. 272–291.

Settles, B. (2010). Active learning literature survey. Technical report, Computer Sciences Technical Report 1648. Madison: University of Wisconsin.

Shaker, A. and Hüllermeier, E. (2012). IBLStreams: a system for instance-based classification and regression on data streams. *Evolving Syst*., 3, pp. 239–249.

Shaker, A. and Lughofer, E. (2014). Self-adaptive and local strategies for a smooth treament of drifts in data streams. *Evolving Syst.*, 5(4), pp. 239–257, 2014.

Shaker, A., Senge, R. and Hüllermeier, E. (2013). Evolving fuzzy patterns trees for binary classification on data streams. *Inf. Sci*., 220, pp. 34–45.

Sherman, J. and Morrison, W. (1949). Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *Ann. Math. Stat*., 20, p. 621.

Shilton, A., Palaniswami, M., Ralph, D. and Tsoi, A. (2005). Incremental training of support vector machines. *IEEE Trans. Neural Netw.*, 16(1), pp. 114–131,

Silva, A. M., Caminhas, W., Lemos, A. and Gomide, F. (2014). A fast learning algorithm for evolving neo-fuzzy neuron, *Appl. Soft Comput*., 14(B), pp. 194–209.

Skrjanc, I. (2009). Confidence interval of fuzzy models: An example using a waste-water treatment plant. *Chemometri. Intell. Lab. Syst*., 96, pp. 182–187.

Smithson, M. (2003). *Confidence Intervals*. SAGE University Paper, Series: Quantitative Applications in the Social Sciences. Thousand Oaks, California.

Smola, A. and Schölkopf, B. (2004). A tutorial on support vector regression. *Stat. Comput*., 14, pp. 199–222.

Soleimani, H., Lucas, K. and Araabi, B. (2010). Recursive Gath-Geva clustering as a basis for evolving neuro-fuzzy modeling. *Evolving Syst*., 1(1), pp. 59–71.

Stehman, V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote Sens. Environ*., 62(1), pp. 77–89.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *J. R. Stat. Soc*. 36(1), pp. 111–147.

Subramanian, K., Savita, R. and Suresh, S. (2013). A meta-cognitive interval type-2 fuzzy inference system classifier and its projection based learning algorithm. In *Proc. IEEE EAIS 2013 workshop (SSCI 2013 conf.)*, Singapore, pp. 48–55.

Sun, H. and Wang, S. (2011). Measuring the component overlapping in the gaussian mixture model. *Data Min. Knowl. Discov*., 23, pp. 479–502.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Syst., Man Cybern*., 15(1), pp. 116–132.

Tschumitschew, K. and Klawonn, F. (2012). Incremental statistical measures. In Sayed-Mouchaweh, M. and Lughofer, E. (eds.), *Learning in Non-Stationary Environments*: *Methods and Applications*. New York: Springer, pp. 21–55.

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15. Trinity College Dublin, Ireland, Department of Computer Science.

Tung, S., Quek, C. and Guan, C. (2013). eT2FIS: An evolving type-2 neural fuzzy inference system. *Inf. Sci*., 220, pp. 124–148.

Wang, L. and Mendel, J. (1992). Fuzzy basis functions, universal approximation and orthogonal least-squares learning. *IEEE Trans. Neural Netw*., 3(5), pp. 807–814.

Wang, L., Ji, H.-B. and Jin, Y. (2013). Fuzzy passive-aggressive classification: A robust and efficient algorithm for online classification problems, *Inf. Sci*., 220, pp. 46–63.

Wang, W. and Vrbanek, J. (2008). An evolving fuzzy predictor for industrial applications. *IEEE Trans. Fuzzy Syst*., 16(6), pp. 1439–1449.

Werbos, P. (1974). *Beyond Regression*: *New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis. Harvard University, USA: Appl. Math.

Wetter, T. (2000). Medical decision support systems. In L. N. (eds.), *Computer Science*, *Medical Data Analysis*. Berlin/Heidelberg: Springer, pp. 458–466.

White, T. (2012). *Hadoop*: *The Definitive Guide*. O'Reilly Media.

Widiputra, H., Pears, R. and Kasabov, N. (2012). Dynamic learning of multiple time series in a nonstationary environment. In Sayed-Mouchaweh, M. and Lughofer, E. (eds.), *Learning in Non-Stationary Environments*: *Methods and Applications*. New York: Springer, pp. 303–348.

Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts, *Mach. Learn*., 23(1), pp. 69–101.

Wu, X., Kumar, V., Quinlan, J., Gosh, J., Yang, Q., Motoda, H., MacLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z.-H. Steinbach, M., Hand, D. and Steinberg, D. (2006). Top 10 algorithms in data mining. *Knowl. Inf. Syst*. 14(1), pp. 1–37.

Xu, Y., Wong, K. and Leung, C. (2006). Generalized recursive least square to the training of neural network. *IEEE Trans. Neural Netw*., 17(1), pp. 19–34.

Xydeas, C., Angelov, P., Chiao, S. and Reoulas, M. (2006). Advances in eeg signals classification via dependant hmm models and evolving fuzzy classifiers. *Int. J. Comput. Biol. Med., special issue on Intell. Technol. Bio-inf. Med*., 36(10), pp. 1064–1083.

Yager, R. R. (1990). A model of participatory learning. *IEEE Trans. Syst., Man Cybern*., 20(5), pp. 1229–1234.

Ye, J., Li, Q., Xiong, H., Park, H., Janardan, R. and Kumar, V. (2005). IDR, QR: An incremental dimension reduction algorithms via QR decomposition. *IEEE Trans. Knowl. Data Eng*., 17(9), pp. 1208–1222.

Zaanen, A. (1960). *Linear Analysis*. Amsterdam: North Holland Publishing Co.

Zadeh, L. (1965). Fuzzy sets. *Inf. Control*, 8(3), pp. 338–353.

Zadeh, L. (1975). The concept of a linguistic variable and its application to approximate reasoning. *Inf. Sci*., 8(3), pp. 199–249.

Zavoianu, A. (2010). *Towards Solution Parsimony in an Enhanced Genetic Programming Process*. PhD thesis. Linz, Austria: Johannes Kepler University Linz.

Zdsar, A., Dovzan, D. and Skrjanc, I. (2014). Self-tuning of 2 DOF control based on evolving fuzzy model. *Appl. Soft Comput*., 19, pp. 403–418.

Zhou, X. and Angelov, P. (2006). Real-time joint landmark recognition and classifier generation by an evolving fuzzy system. In *Proc. FUZZ-IEEE 2006*. Vancouver, Canada, pp. 1205–1212.

Zhou, X. and Angelov, P. (2007). Autonomous visual self-localization in completely unknown environment using evolving fuzzy rule-based classifier. In *2007 IEEE Int. Conf. Comput. Intell. Appl. Def. Secur.*, Honolulu, Hawaii, USA, pp. 131–138.