This is a postprint version of the following published document:

Škrjanc, I., Iglesias, J.A., Sanchís, A., Leite, D., Lughofer, E., Gomide, F. (2019). Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A Survey. *Information Sciences*, 490, pp. 344-368

DOI: 10.1016/j.ins.2019.03.060

© Elsevier, 2019

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

Evolving Fuzzy and Neuro-Fuzzy Approaches in Clustering, Regression, Identification, and Classification: A Survey

Igor Škrjanc[a], Jose Iglesias[b], Araceli Sanchis[b], Daniel Leite[c], Edwin Lughofer[d], Fernando Gomide[e]

[a] *Faculty of Electrical Engineering, University of Ljubljana, Slovenia. E-mail: igor.skrjanc@fe.uni-lj.si*
[b] *Computer Science Department, Carlos III University of Madrid, Spain. E-mails:{jiglesia,masm}@inf.uc3m.es*
[c] *Department of Engineering, Federal University of Lavras, Brazil. E-mail: daniel.leite@deg.ufla.br*
[d] *Department of Knowledge-Based Mathematical Systems, Johannes Kepler University Linz, Austria. E-mail: edwin.lughofer@jku.at*
[e] *School of Electrical and Computer Engineering, University of Campinas, Brazil. E-mail: gomide@dca.fee.unicamp.br*

Abstract

Major assumptions in computational intelligence and machine learning consist of the availability of a historical dataset for model development, and that the resulting model will, to some extent, handle similar instances during its online operation. However, in many real-world applications, these assumptions may not hold as the amount of previously available data may be insufficient to represent the underlying system, and the environment and the system may change over time. Also, as the amount of data increases, it is no longer feasible to process them efficiently using multiple passes, iterative algorithms. Evolving modeling from data streams has emerged as a framework to address these issues properly by self-adaptation, single-pass learning steps and evolution as well as contraction of model components on demand and on the fly. This survey focuses on evolving fuzzy rule-based models and neuro-fuzzy networks for clustering, classification and regression and system identification in online, real-time environments where learning and model development should be performed incrementally.

Keywords:

Evolving Systems, Incremental Learning, Adaptive Systems, Data Streams.

1. Introduction

Progress of computer and communication technology has increased the capability to produce large amount of heterogeneous data from distinct autonomous sources endlessly. The amount of data increases continuously and changes rapidly over time. These data sets are called data streams. Data streams are common in online trading, financial analysis, e-commerce and business, smart home, health care, transportation systems, global supply logistic chains, smart grids, industrial control, cyber-security, and many other areas.

Data stream processing brings unique challenges which are not easily handled by many of the current computational intelligence and machine learning methods. Ideally, machine learning methods should readily adapt to changing situations. The data generation processes are emergent and dynamic, meaning that stream data processing methods must be capable to adapt to new situations (such as system drifts or non-stationary environments). One important question is how to transform stream data into knowledge. Machine learning and computational intelligence algorithms may fail when they encounter a situation that is distinct from the history embedded in historical data sets. Models are common in science and engineering, and development of domain meaningful models using data from non-stationary environments must allow models with the scope and granularity necessary to answer fundamental cause and effect relationships from new experiences.

Online learning is a powerful way to deal with stream data. An online learning algorithm observes a stream of examples to assemble a model and make predictions. It receives and uses immediate feedback about each prediction to improve performance. In contrast to machine learning and statistical learning schemes, online learning from data streams do not make assumptions on distributions of the observations because the behavior it tries to predict change over time in unforeseen ways, what causes concept drifts and shifts. Concept drift means the way the data distribution changes gradually in time, and concept shift refers to a sudden, abrupt change of the nature of the data distribution. Because data may evolve over time, data streams endows temporal locality. At the model level, the challenge is to develop global models by combining locally developed models to form a unifying knowledge. This requires carefully designed algorithms to verify local models correlations in the data-time space, and combination of the outputs from multiple local models into the best model.

The impact of concept drift and shift in learning algorithms is enormous. While the effect of concept drift can be attenuated using e.g. model parameter adaptation procedures, concept shift may require search in the underlying, eventually distinct hypothesis space from the current one. The key difference of evolving systems to online incremental machine learning (inc-ML) is their ability to simultaneously manage any significant changes (drift, shifts, non-stationary behaviors, environmental conditions etc.) in the system by using parameter *and* structural adaptation algorithms to process a data item at most once, while in inc-ML typically only parameters are updated, but no intrinsic structural change in the model is conducted.

Many types of stream data algorithms have been developed for clustering, classification, frequent pattern mining, anomaly detection, and numerous applications in distinct domains such as sensor networks, real-time finance, forecasting, control of unmanned vehicles, and diagnosis have been reported [1], [68], [168]. Several algorithms and applications of evolving intelligent systems in clustering, classification, forecasting, control, diagnosis, and regression are also found in the literature [19], [133].

This paper gives a systematic survey on evolving systems, focusing on fuzzy classification and regression models. The purpose is to introduce the major ideas and concepts of fuzzy evolving systems, to overview their main structural components, models, and respective learning algorithms. The paper also attempts to guide the reader to the essential literature, the main methodological frameworks and its foundations, and the design principles needed to develop applications as well as advanced concepts to make evolving (neuro)-fuzzy systems, E(N)FS, more robust and better applicable in real-world scenarios. The remainder of the paper is organized as follows. In the next section, the evolving systems are presented in general. An overview of evolving algorithms for regression and an overview of algorithms for classification are given. In Section III, the different mechanisms of adding clusters together with safety conditions and different ways of initialization of new cluster, merging cluster mechanisms, splitting and removing clusters mechanisms are discussed. Section IV discusses several important advanced concepts which were developed during recent years to improve robustness, generalization performance, usability and applicability of E(N)FS. At the end some future directions and conclusion are given.

2. Evolving systems

Many systems are characterized by complex behaviors that emerge as a result of nonlinear spatio-temporal interactions among their components. Adaptation gives a system flexibility to improve its short-term performance, and increases its chances to survive in the long-term despite of changes in the environment and in its own components. While small changes in system parameters can be handled as a form of uncertainty, and repaired using parameter estimation mechanisms, changes in system structure requires a higher level of adaptation. An adaptive system is a nonlinear system that evaluates its performance, assesses the operating conditions of its components, measures the state of the environment, and adapts its dynamics to continuously meet performance specifications. In addition to parameter estimation, adaptation requires maintenance actions for performance and goal achievement (also termed as model maintenance) whenever large changes in system structure and in the environment occur.

Adaptive and learning systems have been studied in science and engineering, especially in the area of adaptive control and system identification since early fifties [34], [183], [184]. In adaptive control, the term adaptive means a class of design techniques applicable when the system model is partially known. These techniques either subsume some form of parameter adjustment algorithm [73], employ a set of finite local models and controllers with higher level supervisory switching [105], or use iterative learning techniques [3]. Adaptive control design techniques are mostly model-based, equipped with data-driven parameter estimation and self-tuning algorithms.

The field of *evolving systems* can be traced back to the year 1991 with the publication of the paper [149], where the method resource allocating network (RAN) was introduced. It deals with a neural-network adapted based on *gradient descent* learning and the *chain* rule to propagate errors backwards. Later [65] suggested the growing cell structure (GCS), a class of self-organizing neural networks that control structural changes using supervised or unsupervised learning. These papers did not attract much attention, perhaps because neural networks were not sufficiently established as a scientific discipline. From that time forth, the field of evolving systems faced a tremendous development. Fig. 1 overviews the different types of evolving intelligent systems.

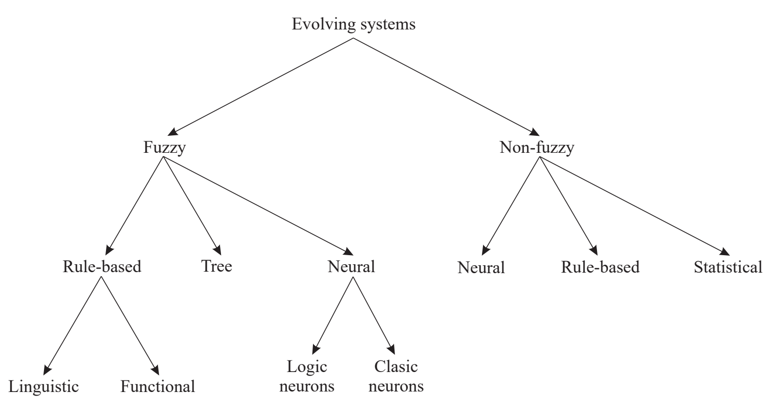

Figure 1: Types of evolving systems

Evolving systems are adaptive intelligent systems that, differently from adaptive and machine learning systems of the last decade, learn their structure and parameters simultaneously using a stream of data. The structural components of evolving systems can be artificial neurons, production rules, fuzzy rules, data clusters, or sub-trees [122]. The structure of rule-based systems is identified by the nature and the number of rules. For instance, evolving fuzzy rule-based systems may use linguistic fuzzy rules, functional fuzzy rules, or their combination. The structure of neuro-fuzzy systems is in turn recognized by the nature of the neurons, the network topology, and the number of neurons in hidden layers.

Evolving intelligent systems as a framework to embody recursive data processing, one-pass incremental learning, and methods to develop systems with enduring learning and self-organization capabilities were first conceptualized in [13] when the term was coined. The authors use the term *evolving in the sense of gradual development of the system structure (rule-base or the architecture of the neural network that represents the system) and their parameters* as Fig. 2 shows. The authors also contrast the name *evolving* with *evolutionary* as used in genetic algorithms and genetic programming: while evolutionary processes proceed with populations of individuals using recombination and variation mechanisms during generations (typically in a temporally static, off-line optimization context), evolving processes advance *over time* during the life span of the system.

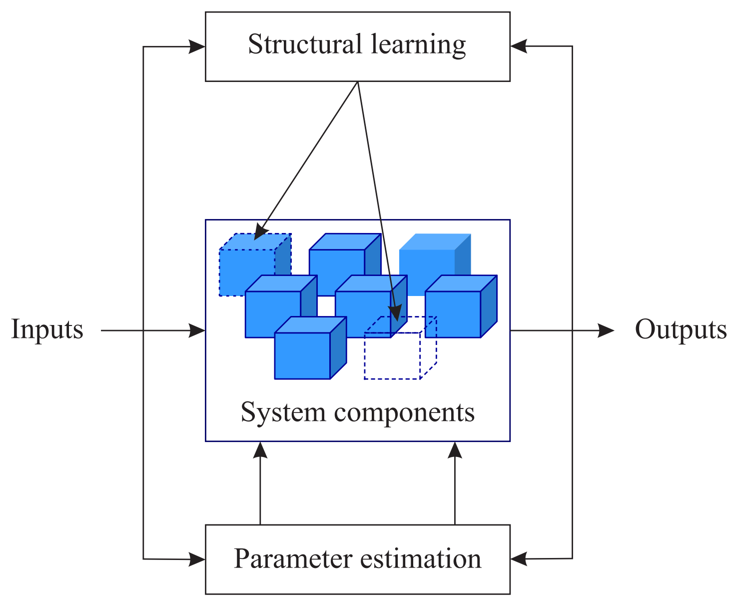

Figure 2: Framework of evolving systems

Summing up, while adaptive systems in control and system theory deal predominantly with parameter estimation, and evolutionary algorithms with populations of models to produce new models, evolving systems benefit from learning from experience, inheritance, gradual change and knowledge generation from (temporal) streams of data [72].

Important milestones in the history of evolving systems can be mentioned such as the publication of the monographs: *Evolving Connectionist Systems* [98], *Evolving Intelligent Systems* [19], and *Evolving Fuzzy Systems Methodologies, Advanced Concepts and Applications* [133] [141]; and the beginning of the international journal entitled *Evolving Systems* [21] by Springer in 2010.

2.1. Evolving systems in clustering, regression, and identification

This section overviews evolving algorithms for regression and identification. Emphasis is on systems that we face in real life, namely, systems that are nonlinear in nature and dependent on the influence of the environment, which vary over time. This also means that the behavior of the systems changes over time. To deal with nonlinear and time-varying processes, the change of the behavior should be identified online, in real time. However, since the data are continuously generated from different sources, their amount is usually very large and samples can be highly heterogeneous and of very high dimension. Therefore, existing intelligent technologies should be adapted through the use of online learning algorithms so that big data streams can be processed in real time, [21] [68]. The use of *off-line* methods in this kind of problems is not possible, [10], neither it is in the case of significant dynamic system changes and non-stationary environments (often appearing in complex real-world scenarios) [168]. This is especially important when the model of such systems is used in control, pattern recognition, monitoring or supervision.

In recent years, a number of successful evolving methods has been developed. The structure of the resulting models is usually based on fuzzy rules, neural networks or hybrid neuro-fuzzy concepts. Some important methods based on fuzzy models can be mentioned: eTS [10], xTS [28, 15], simple_TS [14], +eTS [20], FLEXFIS [130], FLEXFIS+ [131], GS-EFS [140], IBeM [115],[108], FBeM [117, 114], and eFuMo [57].

Similarly, some of the most important neuro-fuzzy-based methods are: EFuNN [94, 93], DENFIS [95], eGNN [116, 118], GANFIS [30], SOFNN [123], SAFIS [162], SCFNN [128], NFCN [127], D-FNN [191], GD-FNN [192], SONFIN [89], NeuroFAST [186], RAN [149], ESOM [51], Neural Gas [66], ENFM [174], GAP-RBF [84], eFuMo [57], SOFMLS [164], PANFIS [152] and RIVMcSFNN [158]. The majority of the evolving methods used in regression is based on neuro-fuzzy local RBF models (*radial basis function models*) or on their generalized form, GRBF (GANFIS). The basic RBF models have equal width of Gauss membership functions as proposed in [192] and the others suggest the use of ellipsoidal basis functions (EBF), which have different widths of membership functions. This kind of approach is given in GD-FNN [192], and in SOFNN [123]. In eGNN hyper-rectangles and trapezoidal membership functions with different widths are used. In [100] a new approach to evolving principal component clustering algorithm with a low run-time complexity for LRF data mapping is presented. In [179] a general evolving fuzzy-model based on supervised hierarchical clustering is shown in use for design of experiment (see also Section 4.5). The general evolving fuzzy model in control is shown in [197]. It is also remarkable, that in SOFMLS an upper bound for the average of the identification error could be found.

Evolving systems, similarly as adaptive neuro-fuzzy systems, learn using learning algorithms to adapt their parameters in an online manner [189]. The parameters in this case are subdivided into linear and nonlinear. The nonlinear parameters, such as centers of clusters, width of radial basis functions or information granules, to mention a few, are related to the partition of the input-output space, whereas the linear parameters refer to the parameters of locally valid affine models. The partition of the input-output space is usually done by using different modifications of clustering and fuzzy clustering methods, which are adapted for online use from their off-line counterparts. This means that the methods are unsupervised and aim at granulating the input-output space. The eTS method, for example, uses recursive clustering with subtraction [11] (*subtractive clustering* [45]). The ENFM method – a recursive version of the Gath-Geva clustering method – and eFuMo use recursive c-means and a recursive Gustafson-Kessel clustering algorithm [55]. To adapt local linear parameters, generally a recursive version of the least squares method, eventually with regularization, forgetting or weighting factor is employed. For example, FBeM [117] uses a specificity-weighted recursive least squares method.

Evolving fuzzy and neuro-fuzzy methods can also be divided according to the type of the model. Basically, the most frequent are the models that implement first-order Takagi-Sugeno fuzzy inference systems (SONFIN, D-FNN, GD-FNN, DENFIS, eTS, xTS, FLEXFIS(+), IBeM, FBeM, eGNN, NeuroFAST, SOFNN) or zero-order Takagi-Sugeno models (SCFNN, SAFIS, GAP-RBF, EFuNN). The essential difference between them is the use of a locally valid afine function or a constant in the consequent terms of the rules. Some evolving methods are based on generalized forms of fuzzy models, which consist of a combination of Mamdani, and first-order Takagi-Sugeno models (GANFIS, FBeM, eGNN, eMTSFIS [83]) and thus can achieve linguistic interpretation (due to Mamdani part) with solid or high precision (due to Takagi-Sugeno part).

Evolving methods can also be distinguished regarding the ability of adaptation. Notice that some fuzzy and neuro-fuzzy methods need the initial structure of the model (for example: GANFIS, ANFIS), which is obtained by *off-line* clustering. In this case, the number of fuzzy rules is constant during online operation and therefore the methods are not considered *evolving* methods, but *adaptive* methods since only parameter adaptation is performed online. The first methods to change the structure of the model were called incremental methods. These methods are equipped with mechanisms to add new local models or rules on demand, however they do not have mechanisms to delete old, useless or inactive rules. These methods include RAN, SONFIN, SCFNN, NeuroFAST, DENFIS, eTS, FLEXFIS. Some methods are also supplied with mechanisms to merge or combine clusters that are similar in some sense (ENFM, SOFNN). The incremental methods that are provided with procedures to delete and merge clusters are seen as real *evolving* methods. Some important fuzzy and neuro-fuzzy evolving methods are ESOM, SAFIS, SOFNN, GAP-RBF, Growing Neural Gas

(GNG), EFuNN, IBeM, FBeM, eGNN, D-FNN, GD-FNN, ENFM, simpl_eTS, xTS, +eTS, FLEXFIS+, eFuMo, to mention a few.

At this point it is worth to mention alternative regression algorithms, in particular the incremental fuzzy linear regression tree algorithm of [121]. The algorithm starts with a single leaf with an affine model, and proceeds replacing leaves by sub-trees. The algorithm process data as a stream, and uses a recursive statistical model selection test to update the tree.

2.2. Evolving systems in classification

This section overviews evolving algorithms in classification. Classification is the problem of identifying in which category a new observation belongs. In [49], the classification task is described formally as follows:

Given a set of training examples composed of pairs $\{x_i, y_i\}$, find a function f(x) that maps each attribute vector x_i to its associated class $y_i, i = 1, 2, \ldots, n$, where n is the total number of training examples.

An algorithm that performs classification is called a classifier. To train these classifiers, they receive as input a set of labeled data samples [82]. The training process can be carried out in off-line mode by considering all the data at once before the online operation of the classifier. In that case, it is assumed that a data set containing samples that represent all possible situations is available *a priori*. It is also assumed that changes of the trained classifier over time will not be required when new data arrive. This kind of classification approach is useful in some specific applications [33].

However, it is important to remark that since the beginning of the 21st century, it has been needed to face not only the problem of processing large data sets, but also to handle data streams immediately after the examples arrive [54]. As mentioned before, since the data are continuously generated from different sources, they are usually very large in size and of very high-dimension. In addition, the data usually need to be processed in real time. Often, the training dataset becomes available in small batches over time because the acquisition of these data continuously is expensive and time-consuming. For this reason, the development of classifiers able to manage continuous and high-volume data streams as they arrive has taken place. Big, diverse and rapidly-produced data has also presented novel challenges in classification that are required to be tackled. These new data also provided opportunities to explore new scientific domains recently emerged [71].

This new type of data and emerging needs are related to a kind of classifiers called incremental, which update their parameters with each new data sample. The development of incremental learning systems that can be trained over time from a data stream is a major open problem in the data mining area. An incremental classifier receives and integrates new examples without the need to perform a full learning phase from scratch. As discussed in a survey on supervised classification from data streams [119], a learning algorithm is incremental if for any example $x_1, \cdots, x_n$, it is able to generate hypotheses $f_1, \cdots, f_n$, such that f_{i+1} depends only on f_i and x_i, the current example. The notion of *current example* can be considered as the latest processed example. Incremental classifiers must learn from data much faster then the off-line mode classifiers. Thus, most of the incremental classifiers read the examples just once so that they can efficiently process large amounts of data. In fact, the main properties of an incremental classifier are that it reads examples just once and it generates a similar model to the one obtained by a batch algorithm.

Incremental classifiers have been implemented in many different frameworks:

- In relation to decision trees, the first incremental versions emerged in the 1980s. ID4 [119] and ID5R [187] concern incremental classifiers based on ID3 (*Iterative Dichotomizer 3*) [161] – a well-known algorithm proposed by Quinlan in 1986. Later, in 2006, [59] proposes a classification system based on decision rules that may store updated border examples to avoid unnecessary revisions when virtual drifts are presented in data. Consistent rules classify new test examples by covering, and inconsistent rules classify them by distance – as a nearest neighbor algorithm. The main characteristic of this approach is that the model is incrementally updated according to the new environment conditions.

- Incremental classifiers have been implemented using neural networks [198]. An example of neural classifier is ARTMAP (*Adaptive Resonance Theory*) [39], a class of neural network architectures that performs incremental supervised learning in response to input vectors presented in arbitrary order. Later, a more general ARTMAP system [40] that learns to classify input data by a fuzzy set of features was introduced.

- In relation to a probabilistic framework, the Bayesian classifier is an effective methodology for solving classification problems when all features are considered simultaneously. However, sometimes, all the features do not contribute significantly to the classification. In addition, a huge computation is needed when the features are added one by one in a Bayesian classifier in batch mode using the forward selection method. For this reason, in [2] it was proposed an incremental Bayesian classifier for multivariate normal distribution data sets. In [44], several incremental versions of Bayesian classifiers are addressed.

- An SVM (*Support Vector Machine*) performs classification by constructing an n-dimensional hyperplane that optimally separates the data into two categories [188]. Support Vector Machine is one of the classical machine learning techniques that can help multi-domain applications in a big data environment [177]. However, the support vector machine is mathematically complex and computationally expensive. A training process on new data, discarding previous data, gives not optimal, but approximate results only. Considering this aspect, [42] proposes an incremental procedure (an online recursive algorithm) for training SVM using one vector at a time. In [193], an incremental algorithm that utilizes the properties of support vector set and accumulates the distribution knowledge of the sample space through the adjustable parameters is proposed. The algorithm LASVM [38] is an online approach that incrementally selects a set of examples for

SVM learning. A selection of different incremental SVM algorithms is proposed in [53].

- In relation to lazy learning approaches, such as k-nearest neighbor (KNN), in [167], an incremental KNN algorithm is proposed, which is extended to a fuzzy version (respecting to provide fuzzy weights in the neighbors) in [77]. These kinds of algorithms are useful when a variable number of neighbors are required for each point in the data set. However, lazy learning techniques are usually too slow to cope with (fast) online demands, as a new model is built from scratch locally around each new query point (in dependency of the new query, in fact).

It is fundamental to remark that in these incremental methods, the structure of the resulting classifier (a set of neurons, rules, clusters, support vectors, leaves, etc.) is fixed, as previously chosen. However, new data samples may not follow the same distribution of the training data, and it is necessary to face issues such as overfitting, low generalization and drift and shift of the density in the data stream [132].

Taking these considerations into account, the field of evolving intelligent classifiers started with the evolving fuzzy-rule based classifier eClass (*evolving Classifier*) [16], [17]. An important aspect of eClass is that it can cope with large amounts of data and process streaming data in real time and in online mode. In addition, the different algorithms of the eClass family are one-pass, recursive, and therefore, computationally light since they have low memory requirements. It is important to remark that evolving is not the same as incremental, adaptive or evolutionary.

eClass can evolve/develop from the new data; it has the following properties: eClass can start learning from scratch; and the number of fuzzy rules and the number of classes do not need to be prespecified. These numbers vary by reading and analyzing the input data in the learning process. Thus, its structure is self-developed (evolved).

In addition, eClass classifiers were categorized considering the consequent part of the fuzzy rules that casts the classifiers. In this sense, eClass includes different architectures and online learning methods. The family of alternative architectures includes: *eClass0*, with the classifier consequents representing class label (zero-order) and *eClass1*, which uses a first-order classifier. It is remarkable that recently, the zero-order classifier (*eClass0*) was demonstrated to be fully unsupervised [47].

eClass0 [17] is an FRB classifier and its structure follows the typical construct of an FRB classifier,

$$\begin{aligned} Rule^i :\ &\textbf{if}\ (x_1\ is\ around\ x_i^1)\ \textbf{and}\ \ldots \\ \ldots\ \textbf{and}\ &(x_n\ is\ around\ x_n^i)\ \textbf{then}\ L\ =\ (L_i) \end{aligned} \tag{1}$$

where $Rule^i$ represents the i^{th} fuzzy rule of the FRB structure, $x = [x_1, x_2, \ldots, x_n]^T$ is the vector of features, x^i denotes the prototype (existing sample) of the i^{th} rule antecedent, and L_i is the label of the class of the i^{th} prototype.

About the learning process of *eClass*, it is important to emphasize that FRB antecedent terms are formed from the data stream around highly descriptive prototypes in the input-output space per class. In the case of *eClass0*, its main difference to a conventional FRB classifier is that *eClass0* has an open structure and uses an online learning mechanism that considers such flexible rule-base structure.

eClass1 [17] is an FRB classifier whose architecture regresses over the feature vector using first-order multiple-input-multiple-output evolving Takagi-Sugeno (MIMO-eTS) fuzzy systems. The structure of an *eClass1* rule is

$$\begin{aligned} Rule^i :\ &\textbf{if}\ (x_1\ is\ around\ x_i^1)\ \textbf{and}\ \ldots \\ \ldots\ \textbf{and}\ &(x_n\ is\ around\ x_n^i)\ \ \textbf{then}\ \ (y^i\ = x^T\Theta), \end{aligned} \tag{2}$$

where $Rule^i$ represents the i^{th} fuzzy rule of the FRB structure, $x = [x_0, x_1, \ldots, x_n]^T$

denotes the $(n+1)$-dimensional vector of features, and y_i is the output.

A main aspect in the learning process of *eClass1* is the online identification of the parameters of the FRB structure. These parameters are updated with the arrival of new data sample carrying new information.

In [31], a new family of evolving classifiers is presented, namely simpl_eCLass0, which is an improvement of *eClass*. This family consists of two members: simpl_eCLass0 and simpl_eCLass1 (zero and first order classifiers). These classifier structures have all the advantages of the *eClass* family but their structure adjustment phase is simplified significantly, reducing computational overhead. In the same way as *eClass*, simpl_eCLass works in online mode updating the classifier/rules. In this case, the main differences of these two versions are the consequent part of the fuzzy rules, and their classification strategy, which is simplified based on the simpl_eTS+ approach [18].

A method for training single-model and multi-model fuzzy classifiers incrementally and adaptively was proposed in [129]. This method is called FLEXFIS-Class, as its core learning engine was based on several functionalities (including rule evolution concept) as contained in the original FLEXFIS approach [130]. In [129], two variants for evolving fuzzy classification schemes were presented:

- FLEXFIS-Class SM is an evolving scheme for the single-model case. It exploits a conventional zero-order fuzzy classification model architecture with Gaussian fuzzy sets in the antecedent terms, crisp class labels in the rule consequents and (fuzzy) confidence values for each class in each rule.
- FLEXFIS-Class MM is based on a multi-model architecture that exploits the idea of nonlinear regression by an indicator matrix to evolve a Takagi-Sugeno fuzzy model for each separate class (receiving a label of 1 while all other classes receive a label of 0). To give a final classification statement, the maximal output value from all fuzzy models is elicited: the final class output corresponds to the argument maximum, i.e. it is the class represented by that model which produced the maximal output value.

In [137], the authors extended FLEXFIS-Class to another multi-model variant in the case of multi-class classification problems by using the all-pairs technique, then termed as EFC-AP (Evolving fuzzy classifiers with All-Pairs). For each class pair either a binary FLEXFIS-Class SM model (EFC-AP SC) or a Takagi-Sugeno fuzzy model (by regression on $\{0, 1\}$, EFC-AP TS) is established. For a new query point, a preference value for each class-pair is elicited (how much one class is preferred over the other according to the output confidence), which can be stored in a preference relation matrix. This matrix can be analyzed to produce a final classification statement. Due to the all-pairs technique, the problem of class imbalance in stream learning (leading to deterioration in performance on under-represented classes) could be reduced. This could be successfully evaluated when introducing new classes on the fly in a streaming context for on-line visual inspection systems in [138]: significant increase in classification accuracy trends on new classes (under-represented after their birth) could be observed when using EFC-AP, compared to FLEXFIS-Class SM/MM.

In [22], a new method for defining the antecedent part of a fuzzy rule-based classification system, called AnYa, is proposed. The method removes the need to define the membership functions per variable using often artificial parametric functions such as triangular, Gaussian etc. Instead, it strictly follows the real data distribution by using the concept of data clouds, which can be applied to classification tasks. In addition, as it is based on vector forms, logical connectives are useless. Finally, it uses *relative data density* expressed in a form of a parameter-free (Cauchy type) kernel to derive the activation level of each rule, which are then fuzzily weighted to produce the overall output. In this case, AnYa-Class uses a single rule for each class since all the data of a class form a single data cloud. The number of rules is fixed so this classifier is incremental, but not (fully) evolving. AnYa-Class, as the *eClass* family, is divided in two types: zero order if the consequent of each rule is a single class label, and first order if the consequents of the rules are linear. The concept was used in control to construct the Robust evolving cloud-based controller (Recco) [7] for heat-exchanger plant, and in [8] for real two-tank plant control. This kind of structure was also used in model identification of production control [6] and for evolving model identification for process monitoring and prediction of nonlinear systems in general [9]. Monitoring of large-scale cyber attacks monitoring using evolving Cauchy possibilistic clustering is shown in [182]. Very successful implementation is also reported for evolving cloud-based system for the recognition of drivers' actions in [181]. The comparison of approaches for identification of all-data cloud-based evolving systems is presented in [36], the problems of identification of cloud-based fuzzy evolving systems are studied and elaborated in [37] and a robust fuzzy adaptive law for evolving control systems is presented in [35].

A different version of the *eClass* family, called AutoClassify, is proposed in [23]. As *eClass*, the *AutoClass* family works on a per-sample basis, and requires only the features of that sample plus a small amount of recursively updated information related to the density. In addition, depending on the form of the consequent part of the rules, *AutoClassify* includes:

- AutoClassify0, which is a fully unsupervised method. The learning phase of *AutoClassify0* is unsupervised and based on focal points by clustering or partitioning in data clouds. The term data clouds is proposed in AnYa [22] and refers to structures with no defined boundaries and shapes.

- AutoClassify1 generally provides a better performance compared to its counterpart, but it is semi-supervised and takes advantage of more parameters. AutoClassify1 can work as a MIMO type of model for multiclass classification problems. The learning phase of this classifier is based on the decomposition of the identification problem into: overall system structure design, and parameter identification. However, these tasks are performed in online mode, sample per sample.

A systematic framework for data analytic is proposed in [91]. The underlying classifier is based on the typicality and eccentricity of the data, and it is called TEDAClass (*Typically and Eccentricity based Data Analytics Classifier*). This classifier is evolving, fully recursive, spatially-aware, non-frequentist and non-parametric. TEDAClass is based on the TEDA method [25][92]. It uses local typicality and eccentricity to calculate the closeness to a fuzzy rule.

In [163], an Extended Sequential Adaptive Fuzzy Inference System for Classification, called ESAFIS, is presented. It is based on the original SAFIS approach [162], which itself is based on the functional equivalence between a radial basis function network and a fuzzy inference system. The SAFIS algorithm consists of two aspects: determination of the fuzzy rules and adjustment of the premise and consequent parameters in fuzzy rules. ESAFIS extends SAFIS to classification problems and proposes some modifications in calculating the influence of a fuzzy rule, adding fuzzy rules and especially a faster RLSE based estimation of consequent parameter to speed up the learning process. In [166], a new algorithm is proposed as the combination of SAFIS, and the stable gradient descent algorithm (SGD) [165]. The modified sequential adaptive fuzzy inference system (MSAFIS) is the SAFIS with the difference that the SGD is used instead of the Kalman filter for the updating of parameters.

Evolving semi-supervised classification is discussed in [113],[107]. The granulation method used to construct the antecedent part of evolving granular predictors, often referred to as eGM (*evolving Granulation Method*), is applicable to the partition of unbalanced numerical and granular-valued partially-supervised streaming data subject to gradual and abrupt changes. If an unlabeled sample causes the creation of a granule, then the class of the granule remains undefined until a new labeled sample falls within its bounds. The class label of the new sample tags the granule. Contrarily, if an unlabeled sample rests within the bounds of an existing granule whose label is known, it borrows the granule label. Core and support parameters of trapezoidal fuzzy sets are adapted to represent the essence of the data. More abstract, high-level granules are easier to manage and interpret.

Ensemble learning has also been used in evolving frameworks. Ensemble learning is a machine learning paradigm in which multiple learners are trained to solve the same problem [151] and where the diversity of so-called weak learners (e.g., simple fuzzy classifiers with low number of rules) can improve the prediction accuracy when being combined [150] — Learn++ was one of the first method to address ensemble learning in an incremental context, but it is not evolving. In this sense, [86] presents a method for constructing ensembles based on individual evolving classifiers. In [87], a scheme for constructing ensembles which are created considering the idea behind the stacking technique [190] is addressed. In addition, an evolving ensemble classifier, termed parsimonious ensemble (pENsemble) is proposed in [159], where local experts (base classifiers) are weighted according to their classification accuracy: models with low weights are discarded to make the ensemble more compact. Base classifiers are added on the fly whenever a drift is confirmed by a drift detector based on Hoeffding's inequality. The base classifiers themselves are internally updated and evolved with the usage of pClass method [154]. It has been recently successfully applied in an extended variant for on-line tool condition monitoring in [160]. TEDA, eTS and xTS are combined as an ensemble in [173], where diversity among their outputs is exploited in order to improve classification accuracy.

Since clustering can be defined as an unsupervised classification of observations into groups (clusters) according to their similarity, it can be considered as a type of classification. This well-known unsupervised classification problem has been solved by a variety of off-line approaches such as k-Nearest Neighbor, fuzzy c-means, where the recursive version of this algorithm is first reported in [56] and in [55] in Gustafson-Kessel modification. Other well-known approaches are incremental/on-line, namely, Self-Organizing Maps, SOM [102], extended in [52] to an evolving approach or Adaptive Resonance Theory, ART [41]. However, these approaches are not fully unsupervised and autonomous since some problem-specific thresholds and guesses on the number of clusters in the data set are required. In this respect, evolving methods are different since they can start learning from scratch with no need of initial information. Moreover, the number of clusters depends on the data.

Considering these aspects, the notion of autonomous clustering was pioneered with *eClustering* [12], an evolving clustering approach based on the potential/density of the data samples which is recursively calculated by using RDE

[17]. In such clustering method, the first data sample represents the first cluster center. The density of the other data samples is calculated using RDE when they arrive. A new data sample represents a new cluster center if it has higher descriptive power than any of the other centers. In addition, the algorithm checks if the existing clusters should be removed or cluster parameters should be adapted. Similar to *eClass*, *eClustering* is one pass, non-iterative, recursive and can be used interactively. In [18], an improvement of *eClustering*, called *eClustering+*, which does not rely on user- or problem-specific thresholds is proposed. It estimates the density at a data point using a Cauchi function.

In [96], an evolving clustering method (ECM) that employs a type of fuzzy inference, denoted as *dynamic evolving neural-fuzzy inference system* (DENFIS) is proposed. ECM does not ask for the number of clusters, and cluster centers are represented by evolved nodes. In this case, a threshold value to define the maximum distance between a data sample and cluster centers is required.

An evolving version of the Gustafson-Kessel (GK) algorithm [74], called eGKL (evolving Gustafson-Kessel-like), is proposed in [61]. eGKL provides a methodology for adaptive, step-by-step identification of clusters that are similar to the GK cluster. In this sense, eGKL estimates the number of clusters and recursively updates its parameters based on the data stream. The algorithm is applicable to a wide range of practical time-varying issues such as real-time classification. In [180], the idea of evolving Gustafson-Kessel possibilistic c-means clustering (eGKPCM), as an extension of the PCM clustering algorithm, is introduced. PCM is given in [104].

In [32], an on-line evolving clustering approach from streaming data that extends the mean-shift clustering algorithm is proposed. The algorithm is called Evolving Local Mean (ELM), because it uses the concept of non-parametric gradient estimate of a density function using local mean. An ELM prototype consists of a cluster center and a distance parameter. The approach is defined as evolving since the local mean is updated from the data stream and new clusters are added to its structure when the density pattern changes.

Finally, autonomous split-and-merge techniques for assuring homogeneous and compact prototype-based clusters in an incremental, single-pass learning context are proposed in [136] [139]. These are based on conventional and extended evolving vector quantization (EVQ) concepts, the latter leading to arbitrarily rotated and shaped clusters with the usage of a recursive estimation of local inverse covariance matrices.

The next section discusses the main differences of evolving algorithms according to the mechanisms of adding, deleting, merging, and splitting local models.

3. Different evolving mechanisms

Evolving systems should change the structure of the model that describes the behavior of the data stream and should be able to adapt parameters associated to local models. The latter is generally dealt with by using some version of recursive or weighted recursive least-mean squares. The most challenging task, and also the basic feature of the evolving systems, is therefore related to adding, deleting, splitting and merging of clusters, neurons, granules or clouds, which delimit the bounds of local models.

Basic constituting elements of evolving intelligent systems can be defined. Fundamentally, these systems consist of three basic blocks, as shown in Fig. 3. The main block is the *a central decision logic* block. This block calls the remaining, *adaptation* and *evolving*, blocks whenever necessary. In the adaptation block, the local model, rule or cluster parameters are adapted according to the novelties in the incoming data samples that belong to the region of the data space covered by the local model. By contrary, in the evolution block, the structure of the global model is changed. In other words, parameter adaptation is useful to model gradual or slight changes of behavior (*concept drift*), while structural evolution is useful to fit new patterns or completely different behaviors or events into the model (*concept shift*).

The basic ideas behind evolution mechanisms are very different and suitable for different tasks. Next, these mechanisms and corresponding algorithms are

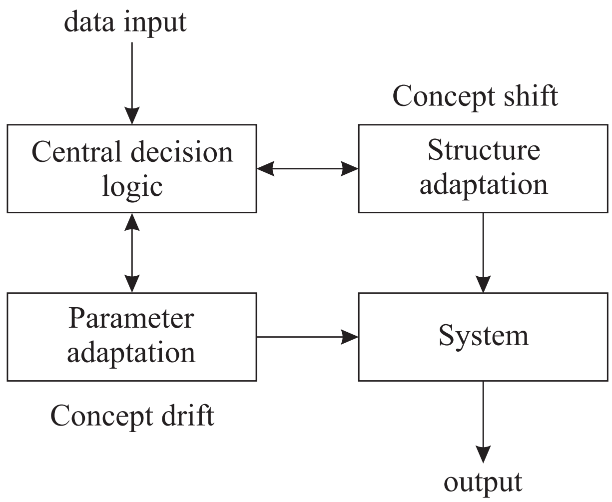

Figure 3: Basic evolving method

discussed in more details.

3.1. Adding clusters

Cluster adding is the most essential mechanism of evolving systems. Usually, learning starts with no local models or clusters; they are added to the global model on the fly in order to expand its knowledge to new regions of interest in the feature space (reducing extrapolation likelihood for new query points). After adding a cluster, a very important task concerns the *initialization* of the parameters of the new local model. Another key decision is related to when and in which place of the data space to consider the cluster. Such decision usually depends on *thresholds*. These thresholds can be given according to (i) the output error – the error between the current measured output and the estimated model output; (ii) some distance, similarity or density metric regarding the current measured input data and cluster prototypes (centers generally); or (iii) the condition of ϵ-completeness, which is connected to the membership degree of the current sample in the current clusters.

The criteria to add a new neuron in the case of evolving systems which are based on neural networks are quite different. In the case of GNG, [67], the new neuron is added at each n-new samples where n stands for a user-defined constant. In many cases, a criterion is defined according to the Euclidean distance between the current sample and cluster centers. This criterion is used in the case of ESOM, [51], DENFIS and FLEXFIS. This means that such a criterion is used in an unsupervised manner. For supervised learning, adding criteria are generally based on the error between the measured and estimated outputs together with some logic and conditions regarding the distance to the cluster centers. This is taken into account in the following methods: EFuNN, D-FNN, GD-FNN, SAFIS [162], SCFNN. The condition for cluster addition can also be given in the form of ϵ-*completeness*, which is used in RAN, SCFNN, SONFIN, eTS. This condition defines the minimal allowed membership value for triggering of closest rule.

In [78], DFKNN considers an adding mechanism based on the Euclidean distance from a sample to the cluster centers and on the change of the local variance caused by the sample. To add a new cluster, the distance and the variance should be greater than a given threshold. As additional condition, the number of samples that belongs to a cluster is monitored. If this number is greater than a threshold, defined by the user, then a new cluster is created.

In [48], a dynamic data clustering algorithm is presented. Cluster addition takes into account the distance between the current sample and the cluster centers, which should be larger than half of the minimal distance between two cluster centers. Moreover, the membership degree of the sample in the clusters should be greater than a pre-defined threshold.

In the case of DENFIS [95], cluster addition is based on a generalized Euclidian distance. If the current sample is within the radius of at least one of the clusters, then the model is not changed. Contrarily, the sum of the cluster radius and the distance between the current sample and the center of the chosen cluster is calculated. This is done for all clusters. If the minimal sum is larger than the double of a threshold value, then a new cluster is added, otherwise the parameters of the cluster are adapted. The threshold is equal to the maximum allowed cluster radius.

The algorithms D-FNN [191] and GD-FNN [192] are similar. Cluster addition is realized according to the output error and the distance of the current sample to the current centers of the clusters. If only the output error is greater than a threshold, the parameters of the local models are adapted. If only the distance if larger that the threshold, the parameters of membership functions are adapted. Otherwise, if both are above the threshold, than a new cluster is added. The thresholds are adaptive. At the beginning, they assume higher values, which are reduced over the iterations. This means that initially a general model is obtained, which becomes more detailed with the time. This is accomplished by decreasing the thresholds. The difference between D-FNN and GD-FNN is the way the thresholds are adapted. The method RAN [149] is also similar, but it uses constant thresholds.

The NeuralGas algorithm [66] monitors the accumulated error between the measured output and the output of the system model in the prescribed time interval. If such error exceeds a predefined threshold, then a new cluster is added. A very similar approach is performed by the NeuroFAST algorithm [186]. The algorithms GAP-RBF [84] and SAFIS [162] add new clusters according to the output error and the distance to the active cluster. In the meantime, the improvement in case a new cluster is added to the position of the current measured sample is calculated. If these three criteria are fulfilled, namely the output error is larger than a threshold, the minimum distance to the cluster centers is larger than a threshold, and sufficient model improvement in relation to the reduction of the output error is observed, then a new cluster is added.

The criterion for cluster addition in the case of EFuNN [94, 93] is based on sensitivity, which is a function of normalized distances. The NFCN [127], ENFM [174], SONFIN [89], SCFNN [128] and SOFNN [123] algorithms are based on the principle of ϵ-completeness, which means that the maximum membership degree considering the current sample and the clusters should not be smaller than a predefined threshold. The SOFNN and SCFNN algorithms take into account not only the ϵ-completeness criterion, but also an additional criterion based on the variation of the output error.

In the case of eTS [10], cluster addition is based on the potential of a current sample. The sample is accepted as the center of a new cluster if the distance to the closest center exceeds a predefined threshold and the potential of the sample is larger than the potential of the current clusters. If the distance condition is not fulfilled, the closest cluster centers move toward the sample. Otherwise, if the distance condition is fulfilled, but the potential of the candidate is lower than the potential of the centers, then only the parameters of the local models are adapted.

Granular evolving methods, IBeM [115, 108], FBeM [117, 114] and eGNN [116, 118], consider a maximum expansion region (a hyper-rectangle) around information granules. Granules and expansion regions are time-varying. They may contract and enlarge independently for different attributes based on the data stream, the frequency of activation of granules and rules, and on the size of the rule base (IBeM, FBeM) or neuro-fuzzy network (eGNN). If a sample does not belong to the expansion region of the current granules, a new granule is created. In eGNN, the use of nullnorm and uninorm-based fuzzy aggregation neurons may provide granules with different geometries [116].

FLEXFIS [130] and its classification versions FLEXFIS-Class SM and MM [129] add a new cluster according to the distance between a sample and the cluster centers. The cluster is added if the smallest distance exceeds a *vigilance parameter* which is normalized subject to the current input dimension in order to avoid too intense cluster growing due to curse of dimensionality. GS-EFS [140] adds a new cluster (in arbitrary position) according to the Mahalanobis distance between a sample and its nearest cluster. The statistical estimation of the so-called *prediction interval* by using an approximated, fast version of the Ξ^2-quantile serves as tolerance region around the ellipsoidal cluster contour in order to decide whether a new (generalized) rule should be evolved or not.

In the eFuMo algorithm [57], the decision about adding clusters can be based on the Euclidean or Mahalanobis distance regarding the current sample and the cluster centers. Calculations can be based on all or on just certain particular elements of the data and cluster vectors.

IN PANFIS [152], a new cluster is added whenever the model error on the new sample is high and also its significance to the PANFIS overall output is given (both factors are multiplied). The latter is measured by the integration of the winning rule (nearest one to the current samples) over the complete feature space, normalized by its range: in order to avoid the revisit of past samples, this can be approximated with determinant operations on covariance matrices representing the shapes and orientations of generalized rules.

It is recommended to consider multiple criteria and different conditions for cluster addition. This is performed by NEUROFast and eFuMo. In eFuMo, the concept of *delay of evolving mechanisms* is introduced. The delay is an interval in which evolving mechanisms are not enabled. Only adaptation of centers and model parameters is conducted during the time interval. The delay of evolving mechanisms takes place after a change in the structure of the system performed by any evolving mechanism. The model should have a certain period to adapt on the new structure. The duration of the delay should be defined by the user and depends on the data and on the amount of data samples. Additional safety conditions are discussed next.

3.1.1. Safety conditions

When evolving algorithms are based on Euclidean distance, there may be regions inside a hypershpere with no representative data. This is not true if the Mahalanobis distance is used because the distances in this case are normalized by the variance of the attributes. This allows multiple ellipsoids to develop close to each other but oriented to different angles.

In real-world data streams, some issues may arise when evolving models deal with outliers. Ideally, outliers should not cause the creation of a new cluster. Therefore, an additional safety condition is generally given, and should be tested before adding clusters to a model. In eFuMo, this condition is based on the number of output samples that do not belong to the current clusters. A delay is introduced into the adding mechanism, but the addition of unnecessary clusters is prevented. The safety condition is given as: *a new cluster is added if*

N consecutive output samples belong to a same cluster and fulfill the necessary criteria for cluster addition. The probability of adding a cluster as a result of outliers is decreased to $P(x)^N$, where $P(x)$ stands for the probability of forming a new cluster from an outlier. The number of samples N is usually chosen from 5 to 10. Similarly, FLEXFIS(-Class) and GS-EFS embed a rule-base procrastination option, where, after adding of new clusters, several samples are waited before the cluster becomes significant and thus alive as rule in the rule base (thus, also being using when predicting new samples).

Some evolving methods accept the creation of clusters in a passive way. In this case, the cluster added to the model based on an outlier will probably not be activated for a number of iterations. Deleting procedures play a key role in these methods to keep the rule base concise and updated.

3.1.2. Initialization of a new cluster

When a sample fulfills all condition for cluster addition, usually it defines the new cluster center. A second parameter to be defined is the size of the cluster. In ellipsoid-based models, the size depends on the covariance matrix. In the literature, a number of different initialization approaches is given: the size of the new cluster depends on the distance to the closest cluster (DENFIS [95], D-FNN [191]); the initial covariance matrix is fixed and given as a user defined parameter (SCFNN [128], SONFIN [89]); it can also be given as the average of the covariance matrices of the existing clusters (xTS [28, 15]). In ENFM [174], the covariance matrix is equal to the covariance matrix of the closest cluster, and in FLEXFIS [130] it is set to a small value of ϵ to guarantee numerical stability of the rules and fuzzy sets. In GS-EFS [140], the inverse covariance matrix is initialized by a fraction of the range of the input feature space or by a weighted average of neighboring rules (where the weights are the support of the rules, i.e. number of data samples which formed them). In PANFIS [152], the covariance matrix is initialized as diagonal matrix in a way that ϵ-completeness is guaranteed (similarly as in SONFIN), i.e. achieving a minimal overlap degree of ϵ with any of the adjacent clusters. Initialization based on the distance to the closest cluster is successful because it covers the gap between clusters. Gap covering was discussed, for example, in [106].

Parameters of local linear models should also be initialized. In [10], the parameters are initialized as 'zero' in the case of using the local fuzzy least squares algorithm, and as the weighted average of the other local linear models for the case of using the global least-square algorithm. The weights are the membership values. In [95], the parameters of the new local model are equal to those of the closest local model.

Initialization of local linear model parameters by weighted average [10] is common. Together with the initialization of local model parameters, covariance matrices can also be taken into account. Weights used to initialize the new local model parameters may consider the variance of a certain parameter. When a new local model is added, covariance matrices in recursive algorithms can be multiplied by a factor $\rho = \frac{c^2+1}{c^2}$, where c is the number of current clusters. This makes further adaptations more sensitive.

3.2. Merging clusters

Cluster merging is necessary when cluster are moving together over time, thus becoming overlapping. This effect is called cluster fusion and is usually caused by samples successively filling up the gaps in-between two or more clusters, which seem to be disjoint at a former point of time in the data stream — but latter turn out that they are not, thus should be merged to eliminate overlapping, redundant information. Merging of clusters not only provides a more accurate representation of the local data distributions, but also keeps E(N)FS more compact and thus better interpretable and faster adaptable.

Different mechanisms for cluster merging are given in this section. In DKFNN, the algorithm monitors the positions of the clusters centers. If two of them approach one another, the underlying clusters should be merged. A measure of cluster similarity, useful for merging, is given in [64]. The measure is based on the membership degree of samples in clusters and is similar to the correlation between the past activations. Merging based on correlation among previous activations is also given in [94] (EFuNN). In this algorithm, merging is based on the maximum cluster radius. Neighbor clusters that present the sum of radii less than a maximum threshold are merged. In ENFM, two clusters are merged when the membership of the first cluster center into the second cluster is greater than the predefined threshold, and vice-versa. In SOFNN [123], clusters are merged when they exhibit the same centers, which is almost impossible in practice.

The algorithm FLEXFIS+ [131] calculates the intersection of the membership functions in each dimension. This is the basis to define the index of overlapping, which is then used to judge whether whole clusters (rules) should be merged or not. If the index is greater than a predefined threshold, then clusters are merged. Merging itself is conducted in the antecedents by an extended variant of recursive variance formula and in the consequents by exploiting Yager's participatory learning concept [194] in order to resolve possibly conflicting rules properly. GS-EFS adds a homogeneity condition among both, antecedent and consequent spaces, to decide whether two clusters should be merged: the angles between their hyper-planes should not be too small and their joint volume should not explode too much. This assures that clusters are not merged inappropriately when they are actually needed to resolve the nonlinearity degree in the local regions where they are defined.

The eFuMo algorithm merges clusters based on the normalized distance between their centers. The distance is calculated based on the Mahalanobis measure. The parameters of the merged cluster are initialized by weighted average [174] or using normal average, such as in [94], while the merged covariance matrix can be defined as proposed in [125]. The algorithm uses also the parameters of the local model similar to FLEXFIS+, but eFuMo also takes into account the prediction of the local models. Three conditions for merging are: angle condition, correlation condition, and distance ratio condition. Two clusters are merged if they fulfill one of these conditions. Additionally, a condition for the local model outputs is taken into account. The difference between two outputs should be less than a predefined threshold, and they should have support set higher than a predefined value.

An instantaneous similarity measure is introduced in FBeM [114, 113] for multidimensional trapezoidal fuzzy sets as

$$
\begin{aligned}
S(A^{i_1}, A^{i_2}) \;=\; & 1 - \frac{1}{4n} \sum_{j=1}^{n} (|l_j^{i_1} - l_j^{i_2}| + |\lambda_j^{i_1} - \lambda_j^{i_2}| + ... \\
& + |\Lambda_j^{i_1} - \Lambda_j^{i_2}| + |L_j^{i_1} - L_j^{i_2}|), \qquad (3)
\end{aligned}
$$

where $A^i = (l^i, \lambda^i, \Lambda^i, L^i)$ is an n-dimensional trapezoid. Such measure is more discriminative than, for example, distance between centers of neighbor clusters, and its calculation is fast. If $S(A^{i_1}, A^{i_2})$ is less than a maximum width allowed for clusters, the underlying clusters are merged. The cluster that results from the merging operation takes into account the bounds of the combined clusters to provide the highest level of data coverage.

3.3. Splitting clusters

The splitting of clusters is defined for a finer structuring of the data space and the model structure. Basically, an evolving algorithm should, in the case of regression and identification problems, accept a larger number of clusters in the region where the model output error (approximation or prediction error) is greater than the expected one or grows extraordinary. This can be because clusters may grow over time due to gradual drifts or due to inappropriately (too pessimistically) set cluster/rule evolution thresholds (parameters). Especially the latter can be the case when using evolving methods in a kind of plug-and-play manner for new applications with tuned (optimized) parameters on previous ones.

The concept of splitting is proposed in [76] and in [50]. In the first, the Chernoff measure is used while the latter assumes a *fidelity measure*. The author in [136] proposes a penalized BIC (*Bayesian information criterion*) to decide whether the current cluster structure should be kept or whether the latest updated cluster should be split into two (the partition receiving a lower penalized BIC value should be preferred). The penalization of the log-likelihood is extended with a product term which vanishes in case of close over-lapping clusters, thus punishing them more than clearly disjoint clusters. As also the punished BIC could not fully represent real cluster homogeneity versus cluster heterogeneity, the split approach in [139] extends this approach by applying a Gaussian mixture model estimation along each principal component direction of an updated cluster with two Gaussians and then checking whether any of the two Gaussians (in each direction) are significantly different (according to the Welch test): if so, a heterogeneous cluster is found (i.e. a cluster which internally represents two disjoint data clouds), and thus it should be split. The split point is estimated through the cutting point of the adjacent (but statistically different) Gaussians.

In NeuroFAST [186], clusters are split according to their mean square error (MSE). The algorithm calculates the error in each P steps and split the cluster and the local model with the greatest error. The mechanism of splitting in eFuMo is based on the relative estimation error, which is accumulated in a certain time interval. The error is calculated for each sample that falls in one of the existing clusters. The initialization of the resulting clusters is based on the eigenvectors of the cluster covariance matrix, as in [79].

An innovative and efficient (fast) incremental rule splitting in the context of generalized evolving fuzzy systems (extending GS-EFS approach [140]) is presented in [144] for the purpose to split blown-up rules with high local errors over past samples into smaller ones to increase model precision. In this sense, it can autonomously compensate drifts which can not be automatically detected, see also Section 4.2.

3.4. Removing clusters

Mechanisms of removing clusters are convenient to delete old or inactive clusters, which are no longer valid. These mechanisms are of utmost importance in classification and pattern recognition. In general, it happens that a cluster is created in a part of the input-output space where there are just a few representative samples. This is justified by errors in measurements or due to a change of the system behavior so that a cluster is not useful after a number of iterations. These clusters can be removed from the model, because they do not help in the description of the data. Nonetheless, careful should be taken with seasonal behaviors since a cluster may be reactivated latter. Moreover, in anomaly detection problems, unusual and idle clusters may be more important than those highly operative, and therefore should not be removed.

The mechanisms to remove clusters are mainly based on the following principles: the age of the rule (xTS, GNG, ESOM), the size of the support set of the cluster (+eTS), the contribution of the rule to the output error (SAFIS [162], GAP-RBF, D-FNN, GD-FNN), the combination of the age and the total number of activations (EFuNN, IBeM, FBeM, eGNN), or the minimal allowed distance between the cluster centers (ENFM). In [48], a cluster is removed from the model if no sample in a certain time interval rests within its bounds. The time interval is defined by the user. A drawback of this approach concerns long steady-state regimes. In this case, important clusters can be removed.

In algorithms D-FNN [191], GD-FNN [192], GAP-RBF [84], SAFIS [162] and SOFNN [123] the removing of a cluster depends on the model output error. In D-FNN, an *error reduction ratio* is introduced to define the contribution of a certain local model to the overall output error. If the local model does not contribute significantly to the error reduction, the cluster is removed. A similar approach is addressed in GD-FNN [192]. Beside an *error reduction ratio*, a *sensitivity index* is introduced. The clusters are removed according to these two values. In SAFIS [162], an estimation of the change in the output error is given when the cluster is removed from the model structure. If this value is higher than a threshold, the cluster is removed. SOFNN [123] introduces a procedure to remove clusters according to the concept of *optimal brain surgeon approach* [81, 124]. This approach is based on the sensitivity of the model output error according to the change of local model parameters. If the sensitivity is greater than a user defined threshold, the cluster is removed. Very similar mechanisms were introduced in [191] and [192].

In [66] (NeuralGas), clusters are removed if they were generated $k - a_{max}$ iterations before, where k stands for the current iteration, and a_{max} is a user-defined threshold.

In [28, 15] (xTS), clusters are removed based on their support set and age. The support set is defined as the number of samples that belongs to a cluster. A sample always belongs to the closest cluster. The age of a cluster is defined as the ratio between the accumulated time of samples and the current time. Clusters are removed according to the ratio between the support set and the overall number of samples and age of clusters. The same condition is also used in +eTS [20], where the condition of *utility* is also used. The utility is defined as the ratio between the number of cluster activations and the time the cluster was added to the model. The cluster is removed when these values differ from the average value, where the confidence band is defined by the standard deviation.

DFKNN [78] removes clusters if their support sets are not larger than a minimum value defined. The minimal support set is a user-defined parameter. A second condition is based on a time interval in which it is required that at least one new sample is within the cluster, otherwise the cluster is removed.

In EFuNN [94, 93], a cluster is removed regarding the age and the sum of cluster activations. The age is defined as the number of samples from the creation of the cluster to the current iteration. If the age of the cluster is higher than a predefined threshold and its number of activations is less than the age of the cluster multiplied by a user defined constant in $[0, 1]$, the cluster is removed. The eGNN approach in [116, 118] is closely related.

In PANFIS [152], clusters are removed when they are inconsequential in terms of contributing very little to outputs on past samples and on possible future samples when observations grow to infinity. This can be reduced to a compact closed analytical form through u-fold numerical integration for any arbitrary probability density functions $p(x)$ of the input data manifold.

In eFuMo [57], the removing mechanism is a modification of that used in +eTS. It is based on the ration between the support set N_{p_i}, and the age of the cluster, a_i. The age of a cluster is defined as $a_i = k - k_i$, where k stands for the current time instant, and k_i is the number of samples from the time instant when the $i-th$ cluster was created. The minimal condition for the existence of the cluster is

$$\textbf{if} \;\; N_{trh} \geq N_{p_i}(a_{trh}), \;\; \textbf{then} \text{ remove cluster,} \tag{4}$$

where N_{trh} stands for the minimal number of samples in the cluster (the support set), a_{trh} is the threshold for the age of the cluster, and $N_{p_i}(a_{trh})$ is the value of the support set when it reaches the age threshold a_{trh}. All thresholds are defined by the user, but they have some commonly used default values. The condition to remove a cluster is given in the form of the ratio between the support set and the age of the cluster and is equal to

$$\frac{N_{p_i}}{a_i} > \epsilon_{Na} \frac{1}{c} \sum_{i=1}^{c} \frac{N_{p_i}}{a_i}, \tag{5}$$

where ϵ_{Na} stands for a user-defined constant, which is less than one, and c is the number of clusters. All clusters that fulfill this condition remain in the model structure, whereas the others are removed.

In [113], the concept of *half-life* of a cluster or granule is introduced. Let

$$\Theta^i = 2^{(-\psi(h-h_a^i))} \tag{6}$$

be the activity factor associated to a cluster. The constant ψ is a decay rate, h the current time step, and h_a^i the last time step that the cluster was activated. Factor Θ^i decreases exponentially when h increases. The half-life of a cluster is the time spent to reduce the factor Θ^i by half, that is, $1/\psi$. Half-life $1/\psi$ is a value useful to remove inactive clusters. Large values of ψ express lower tolerance to inactivity and higher privilege of more compact structures. Small values of ψ add robustness and prevent catastrophic forgetting. ψ should be set in $]0, 1[$ to keep model evolution active.

4. Advanced Aspects and Methodologies

4.1. Advanced Architectures for Increased Performance and Representation

Almost all of the aforementioned E(N)FS approaches employ the classical fuzzy model architecture regarding the antecedent space, which is the AND-connection of fuzzy sets in the single rules with the usage of a t-norm [101]. Formally, a rule is thereby defined in the following way:

$$\begin{aligned}\text{Rule}_i\text{: IF } x_1 \text{ is } \mu_{i1} \;\text{ AND ... AND }\; x_p \text{ is } \mu_{ip} \\ \text{THEN } y_i = f_i(\vec{x}).\end{aligned} \qquad (7)$$

with p the dimensionality of the feature space and $\mu_{i1}, ..., \mu_{ip}$ linguistic terms (such as, e.g., high, intense, weak), formally represented by fuzzy sets [195], and $f_i(\vec{x})$ the consequent part, which can be a real value, a function or a class label. Through the AND-connections, rule activation levels can be achieved, which are typically normalized in the inference process and aggregated over all rules (through a t-conorm) to achieve a final model output — which is either a fuzzy set or already a crisp value, depending on whether a Mamdani-type or a Takagi-Sugeno type fuzzy system is applied.

The AND-connections in (7), when established through t-norms in order to achieve a rule activation level, induce axis-parallel rules. This prevents the possibility to model local correlations between input dimensions accurately and compactly, as t-norms do not allow arbitrarily rotated rules in the multi-dimensional input space. Either more rules are needed for an accurate representation of local data or inaccurate representations are obtained. Figure 4 gives a two dimensional example of this issue.

Thereby, the authors in [120] proposed the use of generalized versions of fuzzy rules in evolving context. These are defined as:

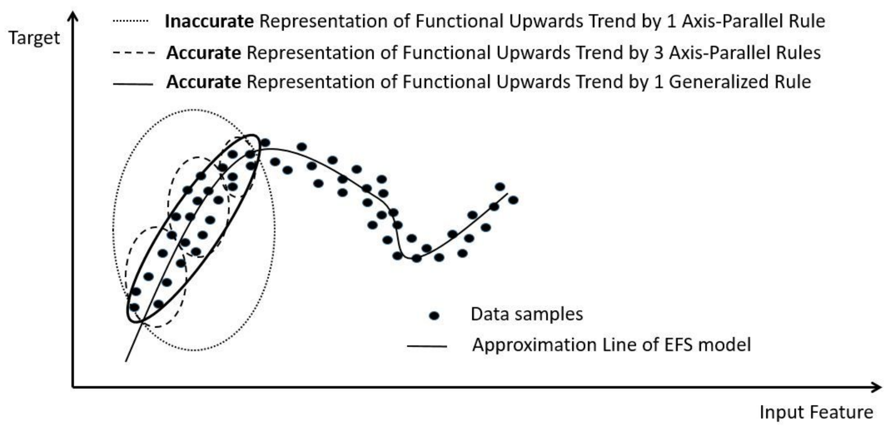

Figure 4: Different representations of a one-dimensional approximation problem by axis-parallel (conventional) and generalized (arbitrarily rotated) rules. Notice the more compact (while still accurate) representation of the left-most upwards trend by the generalized rule (solid ellipsoid)

$$\text{Rule}_i\text{: IF } \vec{x} \text{ IS (about) } \mu_i \text{ THEN } y_i = f_i(\vec{x}). \tag{8}$$

μ_i denotes a high-dimensional kernel function, which, in accordance to the basis function networks spirit, is given by the multivariate Gaussian distribution:

$$\mu_i(\vec{x}) = \exp(-\frac{1}{2}(\vec{x} - \vec{c}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{c}_i)) \tag{9}$$

with $\vec{c}_i$ the center and Σ_i^{-1} the inverse covariance matrix of the ith rule, allowing rotation and spread of the rule. This generalized form of fuzzy rules has been also successfully used in GS-EFS [140] and PANFIS [152], where specific projection concepts have been developed in order to gain an equivalent axis-parallel rule base with conventional fuzzy sets, to maintain linguistic interpretability [195]. In [138], generalized rules have been successfully integrated in the all-pairs technique (EFC-AP, see Section 2.2) for better representing rules in multi-class classification problems. In [5], generalized rules have been used in evolving TS neuro-fuzzy classifiers employing classical single model architecture.

An extension of classical EFS architecture has been proposed in [103], which defines the consequent of rules as a weighted combination of mercer kernels. Therefore, LS-SVM can be applied in order to estimate the weights as support vectors in each local region, which may provide more accuracy especially when there is intrinsic local nonlinearity.

In order to address uncertainty contained in data streams (or even in expert knowledge) on a second level appearance, e.g., fuzzy data which is influenced by noise, [90] proposed an *evolving type-2 fuzzy systems* approach, termed as SEIT2-FNN. Type-2 fuzzy systems were invented by Lotfi Zadeh in 1975 [196] for the purpose of modeling the uncertainty in the membership functions of usual (type-1) fuzzy sets. Through this so-called footprint of uncertainty (FOU) [126], they are thus able to model such occurrences of second level uncertain fuzzy data. SEIT2-FNN uses classical interval-valued fuzzy sets, where the firing strength of type-2 fuzzy rules serves as motivation for rule and fuzzy set evolution. Thereby, this approach assures ϵ-completeness with ϵ being the threshold used for the maximal firing strength. It also embeds a fuzzy set reduction method for strongly overlapping sets. It applies a rule-ordered Kalman filter for consequent learning and an incremental gradient descent algorithm for antecedent learning.

Latter, other techniques for evolving type-2 fuzzy systems have been suggested in [185] (eT2FIS), in [176] (McIT2FIS), in [157] (eT2RFNN) and in [156] (for classification), which significantly expand the original approach in [90] by several concepts such as active learning for sample selection policies, curse of dimensionality reduction by feature weighting and handling of cyclic drifts.

A new variant of neuro-fuzzy architecture has been proposed in [172], which has been termed evolving neo-fuzzy neuronal network (ENFN). ENFN splits the multi-dimensional input space to single uni-variate rules, which therefore reduces error-proneness of the model due to curse of dimensionality effects on structural basis in a natural way (see Section 4.3 below). Even more important, the inference process and the learning is completely independent from the number of inputs; the former just applies the sum of functional activations of each single rule (thus, over all inputs) to a combined output. The functional activation of a single rule is given by a weighted average of activations of two fuzzy sets more adjacent to the current query sample, where the weights are incrementally learnt from data. New membership functions are created whenever the local error exceeds the mean over the global error plus its standard deviation. ENFN also removes unnecessary membership functions due to inactiveness [132].

Multi-model classifiers as discussed in Section 2.2 can also be seen as advanced architectures, contributing to less class imbalance due to class-decomposition and the use of advanced techniques (from preference relation theory) for combining the outputs and evolving models as weak classifiers.

Furthermore, recently evolving deep (fuzzy) rule-based classifiers have been proposed [26]. They are based on the autonomous multi-model systems architecture (ALMMo) [27] and avoid the limitations of current deep learning neural networks structures, which: i) are usually completely un-interpretable (apart from some hierarchical feature representations with different zooms in the case of context-based image data); and ii) require very high computational efforts in batch off-line training cycles.

4.2. Drift Handling for Increased Flexibility

In predictive analytics and machine learning, the *concept drift* means that the statistical properties of either the input or the target variable(s), change over time in unforeseen ways. In particular, drifts either denote changes in the underlying data distribution (input space drift), in the underlying relationship between model inputs and targets (joint drift) or in the prior probabilities of the target class resp. in the distribution of the target vector (target concept drift) — see [99] for a recent comprehensive survey discussing several variants of drifts. Drifts can happen because the system behavior, environmental conditions or process states dynamically change during the online learning process, which makes the (input/output) relations and dependencies contained in and modeled from the old data samples ‘more obsolete’ as time passes.

As already pointed out at the beginning of Section 3, evolving modeling techniques are an adequate methodology to handle drifts in a natural way — especially, when the drift is intense enough (abrupt drifts, shifts), new model components (rules) are typically evolved automatically; such an automatic handling within the learning procedure is also referred as *passive drift handling* [99], which abandons the necessity of detecting drifts explicitly. On the other hand, drifts may also be of lower intensity or of gradual nature [69], which typically deteriorates the local rules and hence overall performance [79].

The pioneering study to handle such drift cases is [132]. The idea is increasing the flexibility of the parameter updates through forgetting concepts. Forgetting is achieved through exponentially outweighing older samples over time with the use of a factor, whose value can be adapted according to the intensity of a drift, measured with the usage of the *concept of rule ages* proposed in [20]. Forgetting of both, antecedent and consequent parameters in EFS was performed in [132] for achieving increased flexibility (of eTS+ and FLEXFIS) and thus significantly increased performance on several real-world (drifting) data sets. Many other EF(N)S methods also include the idea of forgetting older samples, but typically solely in the consequent parameters when being updated through recursive (fuzzily) weighted least squares (RFWLS) [10] (an exception is the eFuMo approach [57] [197], which also performs forgetting in the antecedent space). The RFWLS technique proposed in [10] is fundamental in many E(N)FS methods that rely on the update of linear consequent parameters (see [141]).

Handling of *local drifts*, which are drifts that may appear with different intensities in different parts of the feature space (thus affecting different rules with varying intensity) has been considered in [171] — the idea of this approach is that different forgetting factors are used for different rules instead of a global factor. This steers the local level of flexibility of the model. Local forgetting factors are adapted according to the local drift intensity (elicited by a modified variant of the Page-Hinkley test [146]) and the contribution of the rules in previous model errors.

Another form of drift is the *cyclic drift*, where changes in the (input/target) data distribution may happen at a certain point of time, but latter older distributions are re-visited. ENFS approaches to deal with such drift cases were addressed in [157] [156] using type-2 recurrent (neuro-)fuzzy systems, termed as eT2RFNN. The idea is to prevent re-learning of older local distributions from scratch and thus increase the early significance of the rules.

Whenever a drift cannot be explicitly detected nor it implicitly triggers the evolution of a new rule/neuron, a posteriori *drift compensation* is a promising option in order to (back-)improve the accuracy of the rules. This can be achieved through incremental rule splitting [144]. 'Blown-up' rules with high local errors and high volume are split into two smaller ones along the main principal component axis (with the highest eigenvalue).

4.3. Curse of Dimensionality and Over-fitting Avoidance

High dimensionality of the data stream mining and modeling problem becomes apparent whenever a larger variety of features and/or system variables are recorded, e.g., in multi-sensor networks, which characterize the dependencies and interrelations contained in the system/problem to be modeled. Depending on the ratio between the number of samples (seen so far) and the number of input dimensions, the curse of dimensionality may become apparent, which usually cause significant over-fitting effects [169] and thus affects the whole performance of the model. This is especially the case for models including localization components (granules) as is the case of E(N)FS (in terms of rules/neuron) [147] [148], 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 the analysis in Chapter 1 in [82].

Therefore, the reduction of the dimensionality is highly desired. In a data-stream modeling context, the goal is ambitious and much more sophisticated than in batch learning, because, as in case of changing/drifting data distributions, also the importance of features for explaining the target may change over time. This may be reflected in the ranks or weights of the features. A first work for performing online dimension reduction in a data stream context has been proposed in [20], where the contribution of the features in the consequents of the rules is measured in terms of their gradients in the hyper-planes: those features whose contribution over all rules is negligible can be discarded. Thus, this approach performs a crisp feature selection, but does not respect the possibility that some features may become important again at a later stage, thus should be also reactivated in the model. The same consideration goes to the approach in [153] which extends the approach in [20] by also integrating the contribution of the features in the antecedent space (regarding their significance in the rules premise parts). In [4], online crisp feature selection was extended to a local variant, where for each rule a separate feature (importance) list was incrementally updated. This achieves more flexibility due to a *local feature selection* characteristics, thus features may become differently important in different parts of the feature space, and requires a new design of the fuzzy inference process when predicting new samples.

To overcome a crisp selection and to offer feature reactivation, the approach in [134] proposes the incremental learning of *feature weights* $\in [0, 1]$, where a weight close to 1 denotes that the feature is important and a weight close to 0 that it is unimportant. By updating the features weights with single samples (achieved through an incremental version of Dy and Brodley's separability criterion [58]), slight changes in the weights are achieved over time. They prevent an abrupt inclusion or exclusion of features. Therefore, a feature is able to become reactivated automatically through weight updating, because the model is always learnt on the same whole feature space (thus no input structure changes in the model are needed which requires time-intensive re-training phases). Curse of dimensionality reduction is then achieved i) by integrating the weights in the incremental learning procedure to down-weigh the contributions of unimportant features in rule evolution criteria and parameter update, ii) by integrating the weights in the inference process when producing predictions on new samples to down-weigh the contributions of unimportant features to the final model output and iii) when showing the learnt model to the experts/operators (by simply discarding features with low weights in the antecedents and consequent parts of the rules). The approach in [134] handles classification problems and designs the incremental feature weighting method for evolving fuzzy classifiers using single-model and all-pairs architectures (see Section 2.2). In [140], the feature weighting concepts have been adopted to the regression case where a re-scaled Mahalanobis distance measure had to be developed to integrate weights in distance calculations consistently for generalized EFS.

Another possibility for a smooth input structure change has been proposed in [145] for regression problems with the use of partial least squares (PLS). PLS performs a transformation of the input space into latent variables (LVs) by maximizing the covariance structure between inputs and the target [75]. The coordinate axes are turned into a position (achieving latent variables as weighted linear combination of the original ones) that allows to better explain the complexity of the modeling problem. Typically, a lower number of LVs is needed to achieve accurate regression. Scores on the lower number of LVs (projected samples) are used as input in the evolving models. LVs are updated incrementally with new incoming samples. Previous works in [97] [43] and [60] also perform incremental update of the LV space for evolving models, but using unsupervised principal component analysis (PCA) [88].

4.4. Uncertainty and Reliability

Uncertainty arises during modeling whenever i) either data is affected significantly by noise or is not dense enough (statistically insignificant), especially at the start of the learning process; and ii) the input by humans (in the form of fuzzy rules) is vague due to limited expertise level or forms of cognitive impairments, e.g., distraction, fatigue, boredom, tiredness. Concern with uncertainty is an important aspect especially during the inference process when predicting and/or classifying new samples in order to indicate how reliable model and predictions are. For instance, in a classification system, the certainty of the predictions may support/influence the users/operators in a final decision.

The pioneering approach for achieving *uncertainty in evolving fuzzy modeling* for regression problems was proposed in [178]. The approach is deduced from statistical noise and quantile estimation theory. The idea is to find a lower and an upper fuzzy function for representing a confidence interval, i.e.,

$$\underline{f}(\vec{x}_k^*) \leq f(\vec{x}_k^*) \leq \overline{f}(\vec{x}_k^*) \quad \forall k \in \{1, ..., N\} \tag{10}$$

with N the number of data samples seen so far. The main requirement is to define the band to be as narrow as possible and to contain a certain percentage of the data. This is based on the calculation of the expected covariance of the residual between the model output and new data in local regions as modeled by a linear hyper-plane. The following formulas for the local error (jth rule) were obtained in [178] after deductions and reformulations:

$$\overline{\underline{f}}_j(\vec{x}_k^*) = \Psi_j(\vec{x}_k^*) l_j(\vec{x}_k^*) \pm t_{\alpha,\Sigma(N)-deg} \hat{\sigma} \sqrt{(\vec{x}_k^* \Psi_j(\vec{x}_k^*))^T P_j (\Psi_j(\vec{x}_k^*) \vec{x}_k^*)} \tag{11}$$

where $t_{\alpha,\Sigma(N)-deg}$ stands for the percentile of the t-distribution for $100(1-2\alpha)$ percentage confidence interval (default $\alpha = 0.025$) with $\Sigma(N) - deg$ degrees of freedom and P_j the inverse Hessian matrix. deg denotes the degrees of freedom in a local model. The symbol $\hat{\sigma}$ is the variance of model errors and the first term denotes the prediction of the jth local rule. The sum over all f_j's before the $\pm$ symbol refers to the conventional TS fuzzy model output, with $\Psi_j(.)$ the normalized membership degree and l_j the consequent function of the jth rule. The term after $\pm$ provides the output uncertainty for $\vec{x}$.

Another approach to address uncertainty in model outputs has been proposed in [110] [111], where fuzzy rule consequents are represented by two terms, a linguistic – containing a fuzzy set (typically of trapezoidal nature) – and a functional – as in the case of TS fuzzy systems. The linguistic term offers a direct fuzzy output which according to the widths of the learned fuzzy sets may reflect more or less uncertainty in the active rules (i.e., those rules which have non-zero or at least ϵ membership degree). A granular prediction is given by the convex hull of those sets which belong to active rules. The width of the convex hull can be interpreted as confidence intervals and given as final model output uncertainty. Evolving granular methods were successfully applied to financial time-series forecasting [109], Parkinson's telemonitoring [116], control of chaotic systems [117], rainfall prediction [115] and autonomous robot navigation [112].

Uncertainty in classification problems using evolving fuzzy classifiers has been addressed in [135], see subsequent section. The confidence in predicted class labels is given by a combination of: i) the closeness of the sample to the decision boundary (the closer, the more ambiguous the final classification statement); ii) class overlap degrees (the more overlap, the more ambiguous the final class) and iii) the novelty content calculated through the concept of ignorance (the higher is the novelty, the higher is the unreliability in the final class). A confidence vector is delivered additionally to the class label, representing the confidences in all classes. These concepts are also applied for all-pairs classification: i) by integrating confidence levels of pair-wise classifiers in a preference relation matrix, see Section 2.2); and ii) where the final uncertainty is additionally achieved through calculating the difference between the most and second most supported class. It is interesting to notice that novelty content is also implicitly handled in the error bars in [178] (see Eq. (11)) as for samples lying in extrapolation regions, the statistically motivated error bars are wider.

Apart from model uncertainty, *parameter uncertainty* can be an important aspect when deciding whether the model is stable and robust. Especially in the cases of insufficient or poorly distributed data, parameter uncertainty typically increases. Parameter uncertainty in EFS has been represented in [179] and [143] in terms of the use of the Fisher information matrix [63], with the help of some key measures extracted from it. In [179], parameter uncertainty is used for guiding the design of experiments process in an online incremental manner. In [143], it is used for guiding online active sample selection.

4.5. Online Active Learning and Design of Experiments

Most of the aforementioned ENFS methods require supervision in order to guide the incremental and evolving learning mechanisms into the right direction, to maintain a predictive performance. This is especially true for the recursive update of consequent parameters and input/output product-space clustering. Alternatively, predictions may be used by the update mechanisms to reinforce the model. However, erroneous and imprecise predictions may spread, sum up over time and deteriorate model performance [168].

The problem in today's industrial systems with increasing complexity is that target values may be costly or even impossible to obtain and measure. For instance, in decision support and classification systems, ground truth labels of samples (from a training set, historic data base) have to be gathered by experts or operators to establish reliable and accurate classifiers — which typically require time-intensive annotation and labeling cycles [29] [142]. Within a data stream mining process, this problem becomes even more apparent as experts have to provide a ground truth feedback quickly to meet real-time demands.

Therefore, it is important to decrease the number of samples for model update using sample selection techniques: annotation feedbacks or measurements for only those samples are required, which are expected to maintain or increase accuracy. This task can be addressed by *active learning* [170], a technique where the learner itself has control over which samples are used to update the models [46]. However, conventional active learning approaches operate fully in batch mode by iterating multiple times over a data base.

To select the most appropriate samples from data streams, *single-pass active learning (SP-AL)* for evolving fuzzy classifiers has been proposed in [135]. It relies on the concepts of *conflict* and *ignorance* [85]. The former addresses the degree of uncertainty in the classifier decision in terms of the class overlaping degree considering the local region where a sample falls within and in terms of the closeness of the sample to the decision boundary. The latter addresses the degree of novelty in the sample. A variant of SP-AL is given in [176] [175], where a *meta-cognitive* evolving scheme that relies on the concepts of *what-to-learn*, *when-to-learn* and *how-to-learn* is proposed. The what-to-learn aspect is handled by a sample deletion strategy, i.e., a sample is not used for model updates when the knowledge in the sample is similar to that of the model. Metacognitive learning has been further extended in [158] to regression problems (with the use of a fuzzy neural network architecture) and in [155] with the integration of a budget-based selection strategy. Such a budget-based learning was demonstrated to be of great practical usability.

In case of regression problems, permanent measurements of the targets can also be costly, e.g. in chemical or manufacturing systems that require manual checking of product quality. Therefore, online active learning for regression has been proposed in [143] for evolving generalized fuzzy systems (see Section 4.1) using GS-EFS, which relies on: i) the novelty of a sample (ignorance); ii) the predicted output uncertainty measured in terms of local errors (see Section 4.4); and iii) the reduction of parameter uncertainty measured by the change in the E-optimality of the Fisher information matrix [63].

In summary, it is not only a matter of deciding if targets should be measured/labeled for available samples, but which samples in the input space should be gathered. The model should expand its knowledge or increase significance of its parameters? Techniques from the field of design of experiments (DoE) [62] [70] has been proposed. The pioneering online method for E(N)FS has been proposed in [179]. It relies on a combination of pseudo-Monte Carlo sampling algorithm (PM-CSA) [80] and max-min optimization criterion based on uniformly generated samples which are satisfying a membership degree criterion for the worst local model.

5. Future Directions

A variety of methods have been proposed over the last 15 years to guide the development and incremental adaptation of rule-based and neuro-fuzzy models from data streams. Interesting and persuasive practical solutions have been achieved. Nonetheless, propositions, lemmas, theorems and assurance that certain conditions will be fulfilled are still lacking in the field of evolving clustering and evolving neuro-fuzzy and rule-based modeling from data streams. For instance, necessary and sufficient conditions to guarantee short term adaptation and long term survivability still are to be found. This is a major challenge because it will require the formalization of concept shift and concept drift, and to show how they affect search in a hypothesis space from the point of view of simultaneous parameter estimation and structural adaptation. Systematic approaches to deal with the stability-plasticity trade-off to ensure short-term adaptation and long term survivability still are lacking.

Missing data are common in real-world applications. They arise due to incomplete observations, transfer problems, malfunction of sensors, incomplete information obtained from experts or on public surveys. The missing data issue in spite of having been extensively investigated in off-line settings, in nonstationary data stream environments it is still an open topic.

Further issues that remain unsatisfactorily addressed in the literature concerns characterization, design of experimental setups, and construction of workflows to guide development, performance evaluation, testing, validation, and comparison of algorithms in nonstationary environments. The evolution of rough-set models, Dempster-Shafer models and also aggregation functions are also important topics to expand the current scope of the area. Moreover, a variety of particularities of different applications and evolution aspects in hardware are still to be addressed.

6. Conclusion

We presented a survey on evolving intelligent systems for regression and classification with emphasis on fuzzy and neuro-fuzzy methods. In-depth analyses of research contributions, especially over the last 15 years, which are fundamental to the current state-of-the-art of the field were discussed. The objective is guiding the readers to a clear understanding of the past and current challenges and relevant issues in the area. The survey discussed various evolution mechanisms such as adding, removing, merging and splitting clusters and local models in real-time. We highlighted open or partially addressed research directions, which we believe will help future investigations and developments in the area.

Acknowledgement

Igor Škrjanc, Jose Iglesias and Araceli Sanchis would like to thank to the Chair of Excellence of Universidad Carlos III de Madrid, and the Bank of Santander Program for their support. Igor Škrjanc is grateful to Slovenian Research Agency with the research program P2-0219, Modeling, simulation and control. Daniel Leite acknowledges the Minas Gerais Foundation for Research and Development (FAPEMIG), process APQ-03384-18. Igor Škrjanc and Edwin Lughofer acknowledges the support by the "LCM — K2 Center for Symbiotic Mechatronics" within the framework of the Austrian COMET-K2 program. Fernando Gomide is grateful to the Brazilian National Council for Scientific and Technological Development (CNPq) for grant 305906/2014-3.

References

[1] Aggarwal C., Data Streams: Models and Applications. Springer, New York, NY, USA, 2007.

[2] Agrawal R. and R. Bala, "Incremental bayesian classification for multivariate normal distribution data," *Pattern Recognition Letters*, vol. 29, no. 13, pp. 1873–1876, 2008.

[3] Ahn H., Y. Chen, and K. Moore, Iterative learning control: Brief survey and categorization. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 37(6):1099-1120, November 2007.

[4] Alizadeh S., A. Kalhor, H. Jamalabadi, B. Araabi, and M. Ahmadabadi, "Online local input selection through evolving heterogeneous fuzzy inference system," *IEEE Trans on Fuzzy Syst*, vol. 24, 6, pp. 1364–1377, 2016.

[5] Almaksour A. and E. Anquetil, "Improving premise structure in evolving takagi-sugeno neurofuzzy classifiers," *Evolving Systems*, vol. 2, pp. 25–33, 2011.

[6] Andonovski G., G. Mušič, S. Blažič, and I. Škrjanc, "Online Evolving Cloud-based Model Identification for Production Control," *IFAC-PapersOnLine*, 49(5), pp. 79–84, 2016.

[7] Andonovski G., P. Angelov, S. Blažič, and I. Škrjanc, "A practical implementation of Robust Evolving Cloud-based Controller with normalized data space for heat-exchanger plant," *Appl Soft Comput*, 48, pp. 29–38, 2016.

[8] Andonovski G., B. S. J. Costa, S. Blažič, and I. Škrjanc, "Robust evolving controller for simulated surge tank and for real two-tank plant". *at - Automatisierungstechnik*, 66(9), pp. 725–734, 2018.

[9] Andonovski G., G. Mušič, S. Blažič, and I. Škrjanc, "Evolving model identification for process monitoring and prediction of nonlinear systems," *Engineering Applications of Artificial Intelligence*, 68, pp. 214–221, 2018.

[10] Angelov P. and D. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Transactions on Systems Man and Cybernetics - Part B*, vol. 34, no. 1, pp. 484–498, 2004.

[11] Angelov P., "An approach for fuzzy rule-base adaptation using online clustering," *Integration of Methods and Hybrid Systems*, vol. 35, no. 3, pp. 275–289, 2004.

[12] Angelov P., "An approach for fuzzy rule-base adaptation using online clustering," *Int J Approx Reason*, vol. 35, no. 3, pp. 275–289, 2004.

[13] Angelov P. and N. Kasabov, Evolving computational intelligence systems. Proc. 1st Int. WS on Genetic Fuzzy Systems, 76-82, Granada, Spain, 2005.

[14] Angelov P. and D. Filev, "*Simpl_eTS*: A simplified method for learning evolving Takagi-Sugeno fuzzy models," in *Proc IEEE Int Conf on Fuzzy Systems*, May 2005, pp. 1068 – 1073.

[15] Angelov P. and Z. Xiaowei, "Evolving fuzzy systems from data streams in real-time," in *I. Symp. on Evolving Fuzzy Systems*, Sep. 2006, pp. 29–35.

[16] Angelov P., X. Zhou, and F. Klawonn, "Evolving fuzzy rule-based classifiers," in *Computational Intelligence in Image and Signal Processing, 2007. CIISP 2007. IEEE Symposium on.* IEEE, 2007, pp. 220–225.

[17] Angelov P. and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *IEEE Trans on Fuzzy Syst*, vol. 16, no. 6, pp. 1462–1475, 2008.

[18] Angelov P., "Evolving takagi-sugeno fuzzy systems from streaming data (ets+)," *Evolving intelligent systems: methodology and applications*, pp. 21–50, 2010.

[19] Angelov P., D. P. Filev, and N. Kasabov, *Evolving Intelligent Systems: Methodology and Applications*, P. Angelov, D. P. Filev, and N. Kasabov, Eds. New Jersey: Wiley-IEEE Press, 2010.

[20] Angelov P. , *Evolving Intelligent Systems: Methodology and Applications*. New Jersey: Wiley-IEEE Press, 2010. Evolving Takagi-Sugeno Fuzzy Systems From Streaming Data (eTS+), pp. 21 – 50.

[21] Angelov P., D.Filev, and N. Kasabov, "Editorial," *Evolving Systems*, vol. 1, no. 1, pp. 1–2, 2010.

[22] Angelov P. and R. Yager, "Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density," in *IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS)*, 2011, pp. 62–69.

[23] Angelov P., *Autonomous learning systems: from data streams to knowledge in real-time*. Wiley, 2012.

[24] Angelov P., *Evolving rule-based models: a tool for design of flexible adaptive systems*. Physica, 2013, vol. 92.

[25] Angelov P., "Anomaly detection based on eccentricity analysis," in *IEEE Symp on Evolving and Autonomous Learning Syst (EALS)*, 2014, pp. 1–8.

[26] Angelov P. and X. Gu, "A cascade of deep learning fuzzy rule-based image classifier and svm," in *Proc of the IEEE Int Conf on Systems, Man, and Cybernetics (SMC2017)*, 2017, pp. 746–751.

[27] Angelov P., X. Gu, and J. Principe, "Autonomous learning multi-model systems from data streams," *IEEE Transactions on Fuzzy Systems*, vol. DOI:10.1109/TFUZZ.2017.2769039, 2018.

[28] Asif M., P. Angelov, and H. Ahmed, "An approach to real-time color-based object tracking," in *Proceedings of International Symposium on Evolving Fuzzy Systems 2006*, Sep. 2006, pp. 86 –91.

[29] Azcarraga A., M.-H. Hsieh, S.-L. Pan, and R. Setiono, "Improved SOM labeling methodology for data mining applications," in *Soft Computing for Knowledge Discovery and Data Mining*, NY: Springer, 2008, pp. 45–75.

[30] Azeem M. F., H. Hanmandlu, and N. Ahmad, "Structure identification of generalized adaptive neuro-fuzzy inference systems," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 5, pp. 666–681, 2003.

[31] Baruah R. D., P. Angelov, and J. Andreu, "Simpl_eclass: simplified potential-free evolving fuzzy rule-based classifiers," in *IEEE Int Conf on Systems, Man, and Cybernetics (SMC)*, 2011, pp. 2249–2254.

[32] Baruah D. and P. Angelov, "Evolving local means method for clustering of streaming data," in *IEEE Int Conf on Fuzzy Systems*, 2012, pp. 1–8.

[33] Bishop C., *Pattern Recognition and Machine Learning*. New York: Springer, 2007.

[34] Black W., P. Haghi, K. Arigur, Adaptive systems: History, techniques, problems and perspectives, Systems, 2, 606-660, 2014.

[35] Blažič S., I. Škrjanc, and D. Matko, "A robust fuzzy adaptive law for evolving control systems," *Evolving Systems*, 5(1), pp. 3–10, 2014.

[36] Blažič S., P. Angelov, and I. Škrjanc, "Comparison of Approaches for Identification of All-data Cloud-based Evolving Systems," In *IFAC Conf on Embedded Systems, Computer Intelligence and Telematics CESCIT*, 2015, pp. 129–134, Maribor, Slovenia.

[37] Blažič S. and I. Škrjanc, "Problems of Identification of Cloud-Based Fuzzy Evolving Systems," *Artificial Intelligence and Soft Computing. ICAISC 2016. Lecture Notes in Computer Science*, 9692, pp. 173–182, 2016.

[38] Bordes A. and L. Bottou, "The huller: a simple and efficient online svm," in *European Conf on Machine Learning*. Springer, 2005, pp. 505–512.

[39] Carpenter G. A., S. Grossberg, and J. H. Reynolds, "Artmap: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network," *Neural Netw*, vol. 4, no. 5, pp. 565–588, 1991.

[40] Carpenter G., S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy artmap: A neural network architecture for incremental

supervised learning of analog multidimensional maps," *IEEE Transactions on neural networks*, vol. 3, no. 5, pp. 698–713, 1992.

[41] Carpenter G. and S. Grossberg, *Adaptive resonance theory.* Springer, 2016.

[42] Cauwenberghs G. and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in neural information processing systems*, 2001, pp. 409–415.

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

[44] Chai K. M. A., H. L. Chieu, and H. T. Ng, "Bayesian online classifiers for text classification and filtering," in *Proc ACM SIGIR Conf on Research and development in information retrieval.* ACM, 2002, pp. 97–104.

[45] Chiu S. L., "Fuzzy model identification based on cluster estimation," *Journal of Intellegent and Fuzzy Systems*, vol. 2, pp. 267–278, 1994.

[46] Cohn D., L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, no. 2, pp. 201–221, 1994.

[47] Costa B., P. Angelov, and L. A. Guedes, "Fully unsupervised fault detection and identification based on recursive density estimation and self-evolving cloud-based classifier," *Neurocomputing*, 150, pp. 289–303, 2015.

[48] Crespo F. and R. Weber, "A methodology for dynamic data mining based on fuzzy clustering," *Fuzzy Sets and Syst*, 150, no. 2, pp. 267–284, 2005.

[49] de Carvalho A. C. and A. A. Freitas, "A tutorial on multi-label classification techniques," in *Foundations of Computational Intelligence Volume 5.* Springer, 2009, pp. 177–195.

[50] Declercq A. and J. Piater, "Online learning of gaussian mixture models–a two-level approach," in *Proc 3rd Int Conf on Computer Vision Theory and Applications (VISAPP)*, Funchal, Portugal, 2008, p. 605–611.

[51] Deng D. and N. Kasabov, "ESOM: an algorithm to evolve self-organizing maps from online data streams," in *Proc IEEE-INNS-ENNS: Int Joint Conf on Neural Networks*, vol. 6, 2000, pp. 3–8.

[52] Deng D. and N. Kasabov, "Online pattern analysis by evolving self-organizing maps," *Neurocomputing*, vol. 51, pp. 87–103, 2003.

[53] Domeniconi C. and D. Gunopulos, "Incremental support vector machine construction," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on.* IEEE, 2001, pp. 589–592.

[54] Domingos P. M. and G. Hulten, "Catching up with the data: Research issues in mining data streams." in *DMKD*, 2001.

[55] Dovžan D. and I. Škrjanc, "Recursive clustering based on a gustafson-kessel algorithm," *Evolving Systems*, vol. 2, no. 1, pp. 15–24, 2011.

[56] Dovžan D. and I. Škrjanc, Recursive fuzzy c-means clustering for recursive fuzzy identification of time-varying processes. *ISA Transactions*, vol. 50, no. 2, pp. 159–169, 2011.

[57] Dovžan D., V. Logar, and I. Škrjanc, "Implementation of an evolving fuzzy model (efumo) in a monitoring system for a waste-water treatment process," *IEEE Trans on Fuzzy Syst*, vol. 23, no. 5, pp. 1761–1776, 2015.

[58] Dy J. and C. Brodley, "Feature selection for unsupervised learning," *Journal of Machine Learning Research*, vol. 5, pp. 845–889, 2004.

[59] Ferrer-Troyano F., J. S. Aguilar-Ruiz, and J. C. Riquelme, "Incremental rule learning based on example nearness from numerical data streams," in *Proc ACM Symp on Applied Computing*. ACM, 2005, pp. 568–572.

[60] Filev D. P. and F. Tseng, "Novelty detection based machine health prognostics," in *Proc. of the 2006 International Symposium on Evolving Fuzzy Systems*, Lake District, UK, 2006, pp. 193–199.

[61] Filev D. and O. Georgieva, "An extended version of the gustafson–kessel algorithm for evolving data stream clustering," *Evolving intelligent systems: Methodology and applications*, pp. 273–300, 2010.

[62] Franceschini G. and S. Macchietto, "Model-based design of experiments for parameter precision: State of the art," *Chemical Engineering Science*, vol. 63, no. 19, pp. 4846–4872, 2008.

[63] Frieden B. and R. Gatenby, *Exploratory Data Analysis Using Fisher Information*. New York: Springer Verlag, 2007.

[64] Frigui H. and R. Krishnapuram, "A robust algorithm for automatic extraction of an unknown number of clusters from noisy data," *Pattern Recognition Letters*, vol. 17, no. 12, pp. 1223–1232, 1996.

[65] Fritzke B., Growing cell structures: A self-organizing network for unsupervised and supervised learning. Neural Networks, 7(9):1441-1460, 1994.

[66] Fritzke B., *Advances in Neural Information Processing Systems 7.* Cambridge: MIT Press, 1995, ch. A growing neural gas network learns topologies, p. 625–632.

[67] Fritzke B., "A Growing Neural Gas Network Learns Topologies". Advances in Neural Information Proc Syst. vol. 7, pp. 625–632, 1995.

[68] Gama J., Knowledge Discovery from Data Streams. Chapman and Hall/CRC, Boca Raton, FL, USA, 2010.

[69] Gama J., I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, p. article: 44, 2014.

[70] García S., A. Fernandez, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, pp. 2044–2064, 2010.

[71] Géczy P., "Big data characteristics," *The Macrotheme Review*, vol. 3, no. 6, pp. 94–104, 2014.

[72] Gomide F., Evolving granular neural networks from data streams. Wiley Online Library, https://doi.org/10.1002/047134608X.W8358, Nov 2017.

[73] Goodwin G. and K. Sin, Adaptive Filtering, Prediction, and Control. Prentice-Hall, Englewood Cliffs, N.J., USA, 1984.

[74] Gustafson D. E. and W. C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," in *IEEE Conf on Adaptive Processes*, 1979, pp. 761–766.

[75] Haenlein M. and A. Kaplan, "A beginner's guide to partial least squares (PLS) analysis," *Understanding Statistics*, vol. 3, no. 4, pp. 283–297, 2004.

[76] Hall P. M. and Y. Hicks, "A method to add gaussian mixture models," University of Bath, Technical report CSBU-2004-03, 2004.

[77] Hartert L., M. Sayed-Mouchaweh, and P. Billaudel, "A semi-supervised dynamic version of fuzzy k-nearest neighbours to monitor evolving systems," *Evolving Systems*, vol. 1, pp. 3–15, 2010.

[78] Hartert L., M. Sayed-Mouchaweh, and P. Billaudel, "A semi-supervised dynamic version of fuzzy k-nearest neighbours to monitor evolving systems," *Evolving Systems*, vol. 1, pp. 3–15, 2010.

[79] Hartmann B., O. Banfer, O. Nelles, A. Sodja, L. Teslič, and I. Škrjanc, "Supervised hierarchical clustering in fuzzy model identification," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 6, pp. 1163–1176, 2011.

[80] Hartmann B., J. Moll, O. Nelles, and C.-P. Fritzen, "Hierarchical local model trees for design of experiments in the framework of ultrasonic structural health monitoring," in *Proceedings of the IEEE International Conference on Control Applications*, 2011, pp. 1163–1170.

[81] Hassibi B. and D. G. Stork, "Second-order derivatives for network pruning: Optimal brain surgeon," *Advances in Neural Information Processing*, vol. 4, pp. 164–171, 1993.

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

[83] Ho W., W. Tung, and C. Quek, "An evolving mamdani-takagi-sugeno based neural-fuzzy inference system with improved interpretability–accuracy," in *Proceedings of the WCCI 2010 IEEE World Congress of Computational Intelligence*, Barcelona, 2010, pp. 682–689.

[84] Huang G.-B., P. Saratchandran, and N. Sundararajan, "A recursive growing and pruning RBF (GAP-RBF) algorithm for function approximations," in *Proc of The Fourth International Conf on Control and Automation (ICCA'03)*. Montreal: IEEE, Jun. 2003, pp. 491 – 495.

[85] Hüllermeier E. and K. Brinker, "Learning valued preference structures for solving classification problems," *Fuzzy Sets and Systems*, vol. 159, no. 18, pp. 2337–2352, 2008.

[86] Iglesias J. A., A. Ledezma, and A. Sanchis, "Ensemble method based on individual evolving classifiers," in *Evolving and Adaptive Intelligent Systems (EAIS), 2013 IEEE Symposium on*, April 2013, pp. 56–61.

[87] Iglesias J. A., A. Ledezma, and A. Sanchis, "An ensemble method based on evolving classifiers: estacking," in *Evolving and Autonomous Learning Systems (EALS), 2014 IEEE Symposium on*, Dec 2014, pp. 124–131.

[88] Jolliffe I., *Principal Component Analysis.* New York: Springer, 2002.

[89] Juang C. F. and C. T. Lin, "An online self-constructing neural fuzzy inference network and its applications," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 1, pp. 12–32, 1998.

[90] Juang C. F. and Y. Tsao, "A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1411–1424, 2008.

[91] Kangin D., P. Angelov, J. A. Iglesias, and A. Sanchis, "Evolving classifier tedaclass for big data," *Procedia Computer Science*, 53, pp. 9–18, 2015.

[92] Kangin D. and P. Angelov, "Evolving clustering, classification and regression with teda," in *Int Joint Conf on Neural Networks*, 2015, pp. 1–8.

[93] Kasabov N., T. Yamakawa, and G. Matsumoto, "Evolving fuzzy neural networks-Algorithms, applications and biological motivation," in *Methodologies for the Conceptation, Design and Application of Soft Computing*, vol. 1. Japan: World Scientific, 1998, pp. 271–274.

[94] Kasabov N., "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning," *IEEE Transactions on Systems Man and Cybernetics - Part B*, vol. 31, no. 6, pp. 902–918, 2001.

[95] Kasabov N. and Q. Song, "DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and its application for time-series prediction," *IEEE Transactions on Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, 2002.

[96] Kasabov N. and Q. Song, "Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 144–154, 2002.

[97] Kasabov N., D. Zhang, and P. Pang, "Incremental learning in autonomous systems: evolving connectionist systems for online image and speech recognition," in *Proc of IEEE Workshop on Advanced Robotics and its Social Impacts*, Hsinchu, Taiwan, 2005, pp. 120–125.

[98] Kasabov N., *Evolving Connectionist Systems*. London: Springer, 2007.

[99] Khamassi I., M. Sayed-Mouchaweh, M. Hammami, and K. Ghedira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving Systems*, vol. 9, no. 1, pp. 1–23, 2017.

[100] Klančar G. and I. Škrjanc, Evolving principal component clustering with a low run-time complexity for LRF data mapping. *Applied Soft Computing Journal*, 35:349–358, 2015.

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

[102] Kohonen T., "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[103] Komijani M., C. Lucas, B. Araabi, and A. Kalhor, "Introducing evolving Takagi-Sugeno method based on local least squares support vector machine models," *Evolving Systems*, vol. 3, no. 2, pp. 81–93, 2012.

[104] Krishnapuram R. and J. M. Keller, "A possibilistic approach to clustering," *IEEE Trans on Fuzzy Systems*, vol. 1, no. 2, pp. 98–110, 1993.

[105] Kuipers M. and P. Ioannou, Multiple model adaptive control with mixing. IEEE Transactions on Automatic Control, 55(8):1822-1836, August 2010.

[106] Leite D., P. Costa and F. Gomide, "Interval-based evolving modeling," in: *IEEE WS on Evolving and Self-Developing Intel Syst*, 2009, pp. 1-8.

[107] Leite D., P. Costa and F. Gomide, "Evolving granular neural network for semi-supervised data stream classification," in: *Int Joint Conf on Neural Networks (IJCNN)*, 2010, pp. 1-8.

[108] Leite D., P. Costa and F. Gomide, "Granular approach for evolving system modeling," *Lecture Notes in Artificial Intelligence: Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Springer, Berlin, Heidelberg, pp. 340–349, 2010.

[109] Leite D., F. Gomide, R. Ballini, and P. Costa, "Fuzzy granular evolving modeling for time series prediction," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, 2011, pp. 2794–2801.

[110] Leite D., R. Ballini, P. Costa, and F. Gomide, "Evolving fuzzy granular modeling from nonstationary fuzzy data streams," *Evolving Systems*, vol. 3, no. 2, pp. 65–79, 2012.

[111] Leite D., P. Costa, and F. Gomide, "Interval approach for evolving granular system modeling," in *Learning in Non-Stationary Environments: Methods and Applications*, New York: Springer, 2012, pp. 271–300.

[112] Leite D., and F. Gomide, "Evolving linguistic fuzzy models from data streams," Combining Experimentation and Theory, Berlin, Heidelberg: Springer pp. 209-223, 2012.

[113] Leite D., Evolving Granular Systems, *PhD Thesis - School of Electrical and Computer Engineering*, University of Campinas, 2012.

[114] Leite D., R. Ballini, P. Costa and F. Gomide, "Evolving fuzzy granular modeling from nonstationary fuzzy data streams," *Evolving Systems*, vol. 3, no. 2, pp. 65–79, 2012.

[115] Leite D., P. Costa and F. Gomide, "Interval approach for evolving granular system modeling," *Learning in Non-Stationary Environments*, Springer, New York, NY, pp. 271–300, 2012.

[116] Leite D., P. Costa and F. Gomide, "Evolving granular neural networks from fuzzy data streams," *Neural Networks*, vol. 38, pp. 1–16, 2013.

[117] Leite D., R. Palhares, V. Campos and F. Gomide, "Evolving granular fuzzy model-based control of nonlinear dynamic systems," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 4, pp. 923–938, 2015.

[118] Leite D., M. Santana, A. Borges and F. Gomide, "Fuzzy granular neural network for incremental modeling of nonlinear chaotic systems," in: *IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE)*, 2016, pp. 64–71.

[119] Lemaire V., C. Salperwyck, and A. Bondu, "A survey on supervised classification on data streams," in *European Business Intelligence Summer School*. Springer, 2014, pp. 88–125.

[120] Lemos A., W. Caminhas, and F. Gomide, "Multivariable gaussian evolving fuzzy modeling system," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 1, pp. 91–104, 2011.

[121] Lemos A., F. Gomide, W. Caminhas, Fuzzy evolving linear regression trees. Evolving Systems, vol. 2(1), pp.117–159, Springer, Heidelberg, 2011.

[122] Lemos A., W. Caminhas, F. Gomide, Evolving intelligent systems: methods, algorithms and applications, in S. Ramanna, L. Jain, R. Howlett (eds.), Emerging Paradigms in Machine Learning, 117-159, Springer, Berlin Heidelberg, Germany, 2013.

[123] Leng G., G. Prasad, and T. M. McGinnty, "An online algorithm for creating self-organizing fuzzy neural networks," *Neural Networks*, vol. 17, no. 10, pp. 1477–1493, 2004.

[124] Leung C. S., K. W. Wong, P. F. Sum, and L. W. Chan, "A pruning method for the recursive least squared algorithm," *Neural Networks*, vol. 14, no. 2, p. 147–174, 2001.

[125] Li W., H. H. Yue, S. Valle-Cervantes, and S. J. Qin, "Recursive PCA for adaptive process monitoring," *Journal of Process Control*, vol. 10, no. 5, pp. 471–486, 2000.

[126] Liang Q. and J. Mendel, "Interval type-2 fuzzy logic systems: Theory and design," *IEEE Trans on Fuzzy Systems*, vol. 8, no. 5, pp. 535–550, 2000.

[127] Lin C.-T., "A neural fuzzy control system with structure and parameter learning," *Fuzzy Sets and Systems*, vol. 70, no. 2-3, pp. 183–212, 1995.

[128] Lin F.-J., C.-H. Lin, and P.-H. Shen, "Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 5, pp. 751–759, 2001.

[129] Lughofer E., P. Angelov, and X. Zhou, "Evolving single-and multi-model fuzzy classifiers with flexfis-class," in *Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*. IEEE, 2007, pp. 1–6.

[130] Lughofer E., "FLEXFIS: A robust incremental learning approach for evolving TS fuzzy models," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1393–1410, 2008.

[131] Lughofer E., J.-L. Bouchot, and A. Shaker, "Online elimination of local redundancies in evolving fuzzy systems," *Evolving Systems*, no. 2, pp. 165–187, 2011.

[132] Lughofer E. and P. Angelov, "Handling drifts and shifts in online data streams with evolving fuzzy systems," *Applied Soft Computing*, vol. 11, no. 2, pp. 2057–2068, 2011.

[133] Lughofer E., *Evolving Fuzzy Systems-Methodologies, Advanced Concepts and Applications*, ser. Studies in Fuzziness and Soft Computing Series, J. Kacprzyk, Ed. Berlin Heidelberg: Springer-Verlag, 2011, vol. 266.

[134] Lughofer E., "Online incremental feature weighting in evolving fuzzy classifiers," *Fuzzy Sets and Systems*, vol. 163, no. 1, pp. 1–23, 2011.

[135] Lughofer E., "Single-pass active learning with conflict and ignorance," *Evolving Systems*, vol. 3, no. 4, pp. 251–271, 2012.

[136] Lughofer E., "A dynamic split-and-merge approach for evolving cluster models," *Evolving Systems*, vol. 3, no. 3, pp. 135–151, 2012.

[137] Lughofer E. and O. Buchtala, "Reliable all-pairs evolving fuzzy classifiers," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 4, pp. 625–641, 2013.

[138] Lughofer E., E. Weigl, W. Heidl, C. Eitzinger, and T. Radauer, "Integrating new classes on the fly in evolving fuzzy classifier designs and its application in visual inspection," *Applied Soft Computing*, vol. 35, pp. 558–582, 2015.

[139] Lughofer E. and M. Sayed-Mouchaweh, "Autonomous data stream clustering implementing incremental split-and-merge techniques — towards a plug-and-play approach," *Info Sci*, vol. 204, pp. 54–79, 2015.

[140] Lughofer E., C. Cernuda, S. Kindermann, and M. Pratama, "Generalized smart evolving fuzzy systems," *Evolving Systems*, vol. 6, no. 4, pp. 269–292, 2015.

[141] Lughofer E., "Evolving fuzzy systems — fundamentals, reliability, interpretability and usability," in *Handbook of Computational Intelligence*, P. Angelov, Ed. New York: World Scientific, 2016, pp. 67–135.

[142] Lughofer E., R. Richter, U. Neissl, W. Heidl, C. Eitzinger, and T. Radauer, "Explaining classifier decisions linguistically for stimulating and improving operators labeling behavior," *Info Sci*, vol. 420, pp. 16–36, 2017.

[143] Lughofer E. and M. Pratama, "Online active learning in data stream regression using uncertainty sampling based on evolving generalized fuzzy models," *IEEE Trans on Fuzzy Syst*, vol. 26, no. 1, pp. 292–309, 2018.

[144] Lughofer E., M. Pratama, and I. Škrjanc, Incremental rule splitting in generalized evolving fuzzy systems for autonomous drift compensation. *IEEE Trans on Fuzzy Systems*, vol. 26, no. 4, pp. 1854–1865, 2018.

[145] Lughofer E., A.-C. Zavoianu, R. Pollak, M. Pratama, P. Meyer-Heye, H. Zörrer, C. Eitzinger, J. Haim, and T. Radauer, "Self-adaptive evolving forecast models with incremental PLS space updating for online prediction of micro-fluidic chip quality," *Engineering Applications of Artificial Intelligence*, vol. 68, pp. 131–151, 2018.

[146] Mouss H., D. Mouss, N. Mouss, and L. Sefouhi, "Test of Page-Hinkley, an approach for fault detection in an agro-alimentary production system," in *Proc of the Asian Control Conf, Vol 2*, 2004, pp. 815–818.

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

[148] Pedrycz W., A. Skowron, and V. Kreinovich, *Handbook of Granular Computing*. Chichester, West Sussex, England: John Wiley & Sons, 2008.

[149] Platt J., "A resource allocating network for function interpolation," *Neural Computat.*, vol. 3, no. 2, pp. 213–225, 1991.

[150] Polikar R., L. Upda, S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans on SMC, Part C: Applications and Reviews*, vol. 31, no. 4, pp. 497–508, 2001.

[151] Polikar R., "Ensemble based systems in decision making," *IEEE Circuits and systems magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[152] Pratama M., S. Anavatti, P. Angelov, and E. Lughofer, "Panfis: A novel incremental learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 55–67, 2014.

[153] Pratama M., S. Anavatti, and E. Lughofer, "GENEFIS: Towards an effective localist network," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 3, pp. 547–562, 2014.

[154] Pratama M., S. Anavatti, M. Er, and E. Lughofer, “pClass: An effective classifier for streaming examples,” *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 2, pp. 369–386, 2015.

[155] Pratama M., S. Anavatti, and J. Lu, “Recurrent classifier based on an incremental meta-cognitive scaffolding algorithm,” *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 6, pp. 2048–2066, 2015.

[156] Pratama M., J. Lu, E. Lughofer, G. Zhang, and S. Anavatti, “Scaffolding type-2 classifier for incremental learning under concept drifts,” *Neurocomputing*, vol. 191, no. 304–329, 2016.

[157] Pratama M., J. Lu, E. Lughofer, G. Zhang, and M. Er, “Incremental learning of concept drift using evolving type-2 recurrent fuzzy neural network,” *IEEE Trans on Fuzzy Syst*, vol. 25, no. 5, pp. 1175–1192, 2017.

[158] Pratama M., E. Lughofer, M. Er, S. Anavatti, and C. Lim, “Data driven modelling based on recurrent interval-valued metacognitive scaffolding fuzzy neural network,” *Neurocomputing*, vol. 262, no. 4–27, 2017.

[159] Pratama M., W. Pedrycz, and E. Lughofer, “Evolving ensemble fuzzy classifier,” *IEEE Trans on Fuzzy Systems*, 2018 (In process).

[160] Pratama M., E. Dimla, T. Tjahjowidodo, E. Lughofer, and W. Pedrycz, “Online tool condition monitoring based on parsimonious ensemble+,” *IEEE Trans on Cybernetics*, doi: 10.1109/TCYB.2018.2871120, 2018.

[161] Quinlan J. R. , “Induction of decision trees,” *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[162] Rong H. J., N. Sundararajan, G.-B. Huang, and P. Saratchandran, “Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction,” *Fuzzy Sets and Systems*, vol. 157, no. 9, pp. 1260–1275, 2006.

[163] Rong H.-J., N. Sundararajan, G.-B. Huang, and G.-S. Zhao, “Extended sequential adaptive fuzzy inference system for classification problems,” *Evolving Systems*, vol. 2, no. 2, pp. 71–82, 2011.

[164] Rubio J., “Sofmls: Online self-organizing fuzzy modified least square network,” *IEEE Trans on Fuzzy Systems*, vol. 17, no. 6, pp. 1296–1309, 2009.

[165] Rubio J., P. Angelov, and J. Pacheco, "Uniformly stable backpropagation algorithm to train a feedforward neural network," *IEEE Transactions on Neural Networks*, vol. 22, no. 3, pp. 356–366, 2011.

[166] Rubio J. and A. Bouchachia, "Msafis: an evolving fuzzy inference system," *Soft Computing*, vol. 21, no. 9, pp. 2357–2366, 2017.

[167] Sankaranarayanan J., H. Samet, and A. Varshney, "A fast all nearest neighbor algorithm for applications involving large point-clouds," *Computers & Graphics*, vol. 31, no. 2, pp. 157–174, 2007.

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

[169] Schaffer C., "Overfitting avoidance as bias," *Machine Learning*, vol. 10, no. 2, pp. 153–178, 1993.

[170] Settles B., *Active Learning*. Morgan & Claypool Publishers, 2012.

[171] Shaker A. and E. Lughofer, "Self-adaptive and local strategies for a smooth treatment of drifts in data streams," *Evolving Systems*, vol. 5, no. 4, pp. 239–257, 2014.

[172] Silva A. M., W. Caminhas, A. Lemos, and F. Gomide, "A fast learning algorithm for evolving neo-fuzzy neuron," *Applied Soft computing*, vol. 14, no. B, pp. 194–209, 2014.

[173] Soares E., P. Costa, B. Costa and D. Leite, "Ensemble of evolving data clouds and fuzzy models for weather time series prediction," *Applied Soft Computing*, vol. 64, pp. 445–453, 2017.

[174] Soleimani-B H., C. Lucas, and B. N. Araabi, "Recursive Gath-Geva clustering as a basis for evolving neuro-fuzzy modeling," *Evolving Systems*, vol. 1, no. 1, pp. 59–71, 2010.

[175] Subramanian K., S. Suresh, and N. Sundararajan, "A metacognitive neuro-fuzzy inference system (mcfis) for sequential classification problems," *IEEE Trans on Fuzzy Syst*, vol. 21, no. 6, pp. 1080–1095, 2013.

[176] Subramanian K., A. K. Das, S. Sundaram, and S. Ramasamy, "A metacognitive interval type-2 fuzzy inference system and its projection based learning algorithm," *Evolving Systems*, vol. 5, no. 4, pp. 219–230, 2014.

[177] Suthaharan S., "Support vector machine," in *Machine learning models and algorithms for big data classification*. Springer, 2016, pp. 207–235.

[178] Škrjanc I., "Confidence interval of fuzzy models: An example using a waste-water treatment plant," *Chemometrics and Intelligent Laboratory Systems*, vol. 96, pp. 182–187, 2009.

[179] Škrjanc I., Evolving Fuzzy-Model-Based Design of Experiments With Supervised Hierarchical Clustering. *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 4, pp. 861–871, 2015.

[180] Škrjanc I. and D. Dovžan, "Evolving Gustafson-Kessel possibilistic c-means clustering," *Procedia Computer Science*, vol. 53, pp. 191–198, 2015.

[181] Škrjanc I., G. Andonovski, A. Ledezma, O. Sipele, J. A. Iglesias, and A. Sanchis, "Evolving cloud-based system for the recognition of drivers' actions," *Expert Systems with Applications*, pp. 1–8, 2017.

[182] Škrjanc I., S. Ozawa, T. Ban, and D. Dovžan, "Large-scale cyber attacks monitoring using Evolving Cauchy Possibilistic Clustering," *Applied Soft Computing Journal*, 62, pp. 592–601, 2018.

[183] Tsypkin Y., Adaptation and Learning in Automatic Systems. Academic Press, New York, NY, USA, 1971

[184] Tsypkin Y., Foundations of the Theory of Learning Systems. Academic Press, New York, NY, USA, 1973

[185] Tung S., C. Quek, and C. Guan, "eT2FIS: An evolving type-2 neural fuzzy inference system," *Information Sciences*, vol. 220, pp. 124–148, 2013.

[186] Tzafestas S. G. and K. C. Zikidis, "NeuroFAST: On-line neuro-fuzzy ART-based structure and parameter learning TSK model," *IEEE Trans on System, Man and Cybernetics - Part B*, vol. 31, no. 5, pp. 797–802, 2001.

[187] Utgoff P. E. , "Incremental induction of decision trees," *Machine learning*, vol. 4, no. 2, pp. 161–186, 1989.

[188] Vapnik V., *Statistical Learning Theory*. New York: Wiley, 1998.

[189] Werbos P., "Beyond regression: New tools for prediction and analysis in the behavioral sciences," PhD Dissertation, Harvard University, 1974.

[190] Wolpert D. H., "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[191] Wu S. and M. J. Er, "Dynamic fuzzy neural networks - A novel approach to function approximation," *IEEE Transactions on Systems Man and Cybernetics - Part B*, vol. 30, no. 2, pp. 358–364, 2000.

[192] Wu S., M. J. Er, and Y. Gao, "A fast approach for automatic generation of fuzzy rules by generalized dynamicc fuzzy neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 578–594, 2001.

[193] Xiao R., J. Wang, and F. Zhang, "An approach to incremental svm learning algorithm," in *Proc IEEE Int Conf on Tools with Artificial Intelligence, ICTAI*, 2000, pp. 268–273.

[194] Yager R. R., "A model of participatory learning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 5, pp. 1229–1234, 1990.

[195] Zadeh L., "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[196] Zadeh L., "The concept of a linguistic variable and its application to approximate reasoning," *Info Sci*, vol. 8, no. 3, pp. 199–249, 1975.

[197] Zdešar A., D. Dovžan, and I. Škrjanc, Self-tuning of 2 DOF control based on evolving fuzzy model. *Applied Soft Computing*, 19, pp. 403–418, 2014.

[198] Zhang G. P., "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, pp. 451–462, 2000.

Table .1: List of abbreviations and meanings.

Abbreviation	Meaning
GNG	Growing Neural Gas
ESOM	Evolve Self-organizing Maps
DENFIS	Dynamic Evolving Neural-Fuzzy Inference System
FLEXFIS	Flexible Fuzzy Inference Systems
GS-EFS	Generalized Smart Evolving Fuzzy Systems
EFuNN	Evolving Fuzzy Neural Network
D-FNN	Dynamic Fuzzy Neural Network
GD-FNN	Genetic Dynamic Fuzzy Neural Network
SAFIS	Sequential Adaptive Fuzzy Inference System
SCFNN	Self-Constructing Fuzzy Neural Network
RAN	Resource Allocating Network
GCS	Growing Cell Structure
SONFIN	Self Constructing Neural Fuzzy Inference Network
eTS	Evolving Takagi-Sugeno
DFKNN	Dynamic Fuzzy K-Nearest Neighbors
NeuroFAST	Neuro Function Activity Structure and Technology
GAP-RBF	Growing and Pruning Radial Basis Function
NFCN	Neural Fuzzy Control Network
ENFM	Evolving Neuro-Fuzzy Model
SOFNN	Self Organizing Fuzzy Neural Network
SOFMLS	Online Self-Organizing Fuzzy Modified Least-Squares Network
IBeM	Interval-Based Evolving Modeling
FBeM	Fuzzy set Based Evolving Modeling
eGNN	Evolving Granular Neural Networks
eFuMO	Evolving Fuzzy Model
RDE	Recursive Density Estimation
GANFIS	Generalized Adaptive Neuro-Fuzzy Inference Systems
NFCN	Neural Fuzzy Control Network
PANFIS	Parsimonious Network based on Fuzzy Inference System
RIVMcSFNN	Recurrent Interval-Valued Metacognitive Scaffolding Fuzzy Neural Network
eT2RFNN	Evolving Type-2 Recurrent Fuzzy Neural Network
SEIT2-FNN	Self-evolving Interval Type-2 Fuzzy Neural Network
ENFN	Evolving Neo-Fuzzy Neural Network
RBF	Radial Basis Function models
ANFIS	Adaptive Network-based Fuzzy Inference System
ESOM	Evolving Self-Organizing Map
ID3	Iterative Dichotomizer 3
ID4	Iterative Dichotomizer 4
ID5R	Incremental Decision Tree
LaSVM	Online Support Vector Machine
AnYa	Angelov and Yager system
TEDAClass	Typically and Eccentricity based Data Analytics Classifier
TEDA	Typically and Eccentricity based Data Analytics

Table .2: List of abbreviations and meanings (continuation)

Abbreviation	Meaning
EFC-AP	Evolving Fuzzy Classifier using All-Pairs Technique
ALMMo	Autonomous Multi-Model Systems Architecture
pClass	Parsimonious Classifier
pEnsemble	Parsimonious Ensemble
McIT2FIS	Meta-cognitive Interval Type-2 Neuro-fuzzy Inference System
MSAFIS	Modified Sequential Adaptive Fuzzy Inference System
eGM	Evolving Granulation Method
SOM	Self-Organizing Maps
ART	Adaptive Resonance Theory
ECM	Evolving Clustering Method
GK	Gustafson-Kessel clustering
eGKL	Evolving Gustafson-Kessel Like
eGKPCM	Evolving Gustafson-Kessel Possibilistic C-Means clustering
ELM	Evolving Local Mean
ARTMAP	Adaptive Resonance Theory
BIC	Bayesian Information Criterion
MSE	Mean Square Error
GRBF	Generalized Radial Basis Function
EBF	Ellipsoidal Basis Function
SVM	Support Vector Machine
KNN	K-Nearest Neighbor
eClass	Evolving Classifier
FRB	Fuzzy Rule Based
MIMO	Multi-Input Multi-Output
SGD	Stable Gradient Descent Algorithm