GT2FC: An online Growing interval Type-2 self-learning Fuzzy Classifier

Abdelhamid Bouchachia, Charlie Vanaret

GT2FC: An Online Growing Interval Type-2 Self-Learning Fuzzy Classifier

Abdelhamid Bouchachia, *Senior Member, IEEE,* and Charlie Vanaret

***Abstract*—In the present paper we propose a Growing Type-2 Fuzzy Classifier (GT2FC) for online rule learning from real-time data streams. While in batch rule learning the training data are assumed to be drawn from a stationary distribution, in online rule learning data can dynamically change over time becoming potentially non-stationary. To accommodate dynamic change, GT2FC relies on a new semi-supervised online learning algorithm called 2G2M (Growing Gaussian Mixture Model). In particular, 2G2M is used to generate the type-2 fuzzy membership functions to build the type-2 fuzzy rules. GT2FC is designed to accommodate data online and to reconcile labeled and unlabeled data using self-learning. Moreover. GT2FC maintains low complexity of the rule base using online optimization and feature selection mechanisms.**

GT2FC is tested on data obtained from an ambient intelligence (AmI) application where the goal is to exploit sensed data to monitor the living space on behalf of the inhabitants. Because sensors are prone to faults and noise, type-2 fuzzy modeling is very suitable for dealing with such an application. Thus, GT2FC offers the advantage of dealing with uncertainty in addition to self-adaptation in an online manner. For illustration purposes, GT2FC is also validated on synthetic and classic UCI data sets. The detailed empirical study shows that GT2FC performs very well under various experimental settings.

***Index Terms*—Online learning, Type-2 fuzzy rule systems, semi-supervised learning, Growing Gaussian mixture models, online optimization.**

I. Introduction

OFFLINE development of fuzzy rule-based systems (FRSs) assumes that the process of rule induction is done at once in a one-shot experiment. The learning and the deployment of FRSs are two sequential and independent stages. If, for instance, the FRS performance deteriorates due to a change of the data distribution or a change of the operating conditions, the FRS needs to be re-designed from scratch. Online learning (OL), on the other hand, enables both learning and deployment to happen concurrently. In this context, learning takes place over long periods of time, and is inherently open-ended [8]. The aim is to ensure that the system remains amenable to refinement as long as data continue to arrive. Moreover, online FRSs can also deal with applications starving of data (e.g., experiments that are expensive and slow to produce data as in some chemical and biological applications) as well as with applications which are data intensive (e.g., monitoring, information filtering, etc.) [3], [5].

Abdelhamid Bouchachia is with Bournemouth University, School of Design, Engineering and Computing, UK (email: abouchachia@bournemouth.ac.uk).

Charlie Vanaret is with École Nationale de l'Aviation Civile, Laboratoire MAIAA, Toulouse, France (email:charlie.vanaret@alumni.enseeiht.fr).

Manuscript received January 21st, 2013; revised July 1st, 2013; accepted July 31st, 2013.

OL faces the challenge of accurately estimating the statistical characteristics of data in the future. In non-stationary environments, the challenge becomes even more important, since the FRS's behavior may need to change drastically over time due to concept drift [46]. The aim of OL is to ensure continuous adaptation. The learning algorithm should store only the learning model (e.g., only rules of the FRS) and uses that model as basis in the future learning steps. As new data arrive, new rules may be created and existing ones may be modified allowing the system to evolve over time.

On the other hand, FRSs are preferred for learning from data due to their transparency to human beings and their power to cope with uncertainty [33], [48]. Of interest in this study are incremental FRSs that aim at combining the technology of fuzzy systems and online learning to deal with online induction of comprehensible rules. In particular, incremental type-1 FRS have been recently introduced in a number of studies involving control [1], diagnostic [29], and pattern classification [2], [8]. Such systems are currently quite established, since they do not only operate online, but also consider related advanced concepts such as concept drift and online feature selection.

Motivated by an ambience intelligence application proposed by the authors [16], [20], [47] using type-2 FRS, this paper presents a novel online type-2 fuzzy rule-based system dedicated to classification to monitor a smart space on behalf of the occupants. *To the best of our knowledge, no online type-2 FRS for classification exists yet*. However, there have been studies related to: (1) online type-1 FRS [2], [8] for various modeling tasks, (2) self-evolving type-2 fuzzy neural networks for control [23] and (3) type-2 FRS for classification [30].

We are interested in an online interval-based type-2 FRS whose rules' premises are Gaussians and which are incrementally generated by a Growing Gaussian Mixture Model (2G2M). We also consider operational mechanisms in 2G2M (i.e., merge of partitions, split of partitions, etc.) to dynamically control the complexity and the quality of the rules. Moreover, in an online setting we cannot expect to receive feedback (i.e., data labels) from the environment all the time; thus, it is necessary to equip the fuzzy classifier with mechanisms that enable learning from both labeled and unlabeled data.

This paper addresses the following questions:

- How can OL be adopted for designing type-2 fuzzy classification systems?
- How can the complexity of the rule base be controlled over time?
- How can both labeled and unlabeled data be accommodated during rule learning?

The remainder of the paper is organized as follows. An overview of the literature related to incremental fuzzy rule systems is presented in Sec. II. Then incremental mixture models are introduced in Sec. III. The description of type-2 fuzzy systems follows in Sec. IV. In Sec. V, we introduce our algorithm 2G2M which generates the partitions of space. In particular the model refinement mechanisms are discussed therein. Section VI explains the process of fuzzy rule generation. Finally Sec. VII shows the empirical results obtained on various datasets before the paper is then concluded in Sec. X.

II. Online Fuzzy Rule-based Systems

Traditional FRSs are designed in batch mode, that is, by using the complete training data at once. For stationary processes this is sufficient, but for time-dependent and complex non-stationary processes, efficient techniques for updating the induced models are needed. To avoid starting from scratch every time, these techniques must be able to adapt the current model using only the new data. They have to be equipped with mechanisms to react to gradual changes or abrupt ones. Moreover, the adaptation of knowledge (model/rule-base) should accommodate any information brought in by the new data and reconcile this with the existing knowledge.

Many approaches do simply perform “adaptive tuning”, that is, they permanently re-estimate the parameters of the computed model. However it is quite often necessary to adapt the structure of the rule-base. In contrast to these approaches, some advances in the direction of fully incremental FRS have been made over the recent years (e.g., [1], [8]). While type-1 FRSs are well established, online type-2 FRSs have not yet received much attention. In this section we highlight some of the work on online fuzzy rule systems.

In [1] an approach for adaptation of an FRS of Takagi-Sugeno type was proposed. It consists of two steps: (a) generating clusters that represent the rule's antecedents using an on-line version of the subtractive clustering algorithm and (b) estimating the consequents' parameters using the least squares algorithm. New rules are added when new clusters are generated. Similar approaches relying on subtractive clustering and least squares have been proposed later in [13]. In [20] a type-2 fuzzy controller was introduced. The FRS is capable of incrementally updating its rules which model the relationship between actuators (output variables) and sensors (input variables) in an ambient environment. Whenever the actuators are changed, the state of the environment is recorded before it is mapped to the rules' premises. The conclusions of the firing rules are then replaced by the actual output emanating from the actuators. If no rule is fired, a new one is added. In [8], an integrated approach called FRCS was proposed. To accommodate incremental rule learning, appropriate mechanisms are applied at all steps: (1) *Incremental supervised clustering* to generate the rule antecedents in a progressive manner, (2) *online and systematic* update of fuzzy partitions, (3) *Incremental feature selection* using an *incremental version* of the Fisher's interclass separability criterion to dynamically select features in an online manner.

III. Online Clustering based on GMMs

To automatically generate fuzzy rules, one needs to define the fuzzy membership functions which specify a partitioning of the input and possibly the output space. To do this, there are two types of fuzzy sets: uni-dimensional and multidimensional [9], [26]. Using uni-dimensional fuzzy sets, the rule premises are expressed as conjunction of fuzzy sets associated with the individual dimensions. The uni-dimensional fuzzy sets are obtained by projecting the partitions on the individual space features. On the other hand, using multi-dimensional fuzzy sets, the premises are expressed by the multi-dimensional partitions as a whole. In other terms, a multidimensional (multivariate) space can be divided into multidimensional fuzzy partitions represented in the form of multidimensional fuzzy membership functions. The use of multidimensional functions in fuzzy systems has the advantage of reducing the complexity of the rule base as well as retaining the correlation between the features [28]. Nevertheless uni-dimensional functions provide more transparency as rules are explicitly expressed in terms of the input features. Note that ”uni-dimensional” is not about ”type-1”, it is rather about how fuzzy granules are interpreted and how the form of fuzzy rules is desired to look like. Uni-dimensional fuzzy sets involve the individual features (see Eq. 9, Sec. IV), whereas multi-dimensional fuzzy sets involve granules (see Eq. 10, Sec. IV).

There are 3 types of fuzzy partitioning [22]: grid partitioning, tree partition, and scatter partitioning via clustering. In this paper, we consider the latter which consists of finding the regions that are susceptible to cover the data. A straightforward way to perform scatter partitioning is clustering. Scatter partitioning offers many advantages because: (a) clustering can be done online, (b) compact rule base can be obtained, and (c) interpretability and transparency can be ensured by transforming multi-dimensional fuzzy sets into uni-dimensional fuzzy sets, that is by projecting the clusters onto the feature axes [8].

To facilitate a straightforward mapping between clusters and rules, we will propose a clustering algorithm that relies on an incrementally growing version of the Gaussian mixture model (GMM) trained by the Expectation-Maximization algorithm (EM) [15]. GMM is appealing because it can model complex and nonlinear pattern variations, it is efficient in terms of memory, it is able of selecting the optimal model, and it is theoretically guaranteed to converge [3], [19].

The approach proposed is characterized by the following:

- Online operation: Since the aim of this study is to investigate growing type-2 FRSs, we will rely on an adaptive growing Gaussian mixture model. The idea is to produce clusters that evolve over time to generate the rules' antecedents. Because the mapping from clusters to type-2 fuzzy sets of the rules must be less burdensome, online Gaussian mixture models are applied.
- Semi-supervision: Because of the nature of online setting, not all data that arrive over time are labeled. Therefore, the online GMM-based clustering algorithm to be used in this paper must have semi-supervision mechanisms to accommodate unlabeled data. Note that for FRSs, semi-supervised learning is not well established. Only very few studies have investigated this subject [6], [25].
- Uncertainty handling: In real-time applications, data may be noisy, unlabeled and may drift, while the application of GMM trained online by Expectation-Maximization (EM)

needs to estimate the influence of the new data. That is, the parameters of the model are not exactly computed due to the very nature of online learning. GMM may not be able to reflect the true data distribution (at time t, EM does not have the data already seen in the past, the new data may be noisy and may drift over time). Thus, it is necessary to consider the model's uncertainty. Here we use type-2 fuzzy GMM which allows to describe each parameter of the model with uncertainty.

GMM's are considered as a model-based clustering method which perceives the data as a population with K different components, where each component is generated by an underlying probability distribution. The density of a $D-$dimensional data point x_i from the j^{th} component is $f_j(x_i;\theta_j)$, where θ_j represents some unknown parameters associated with the component j. The density function associated is expressed as:

$$f(x_i|\theta) = \sum_{j=1}^{K} \tau_j f_j(x_i;\theta_j) \quad (1)$$

where τ_j is the weight of the j^{th} component such that $\sum_{j=1}^{K} \tau_j = 1$. A typical case is when $f_j(x_i;\theta_j)$ is a multivariate normal density ϕ_j with θ_j representing the characteristics of each component j. These characteristics are the mean μ_j and the covariance matrix Σ_j. The density ϕ_j reads:

$$\phi_j(x_i|\mu_j,\Sigma_j) = \frac{exp\{-1/2(x_i-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j)\}}{\sqrt{det(2\pi\Sigma_j)}} \quad (2)$$

Gaussian mixtures are usually trained using the iterative EM algorithm. In the E-step, the posterior probability of the data is computed, while in the M-step, the mixture parameters $(\hat{\tau}_j, \hat{\mu}_j, \hat{\Sigma}_j)$ are computed. This iterative algorithm is not adequate for online learning, since the aim is to avoid storing the entire data (i.e. the continuous stream of data) and each data point is discarded once it has been processed. Therefore, only online variants of GMM can be applied.

Incremental approaches of GMM have been proposed by many authors [3], [27], [36], [41] where various versions of EM are applied. In particular, the approach described by Stauffer and Grimson [44] in a tracking context is one of the mostly established approaches. The parameters of the Gaussian mixture are updated as new data arrive. The existing approaches can be split into two categories, we name them here as: refinement-based methods and learning methods.

1) **Refinement-based methods**: In this category, the methods rely generally on two abstract stages: (i) Processing and integration of the new data into the current model, (ii) Refinement of the resulting model which aims at reducing the complexity of the model that results from the former stage. Usually the refinement is realized using some merge and split operations. Many merge criteria have been used, e.g., the w-statistic for equality of covariance and Hotellings T^2-statistic for equality of mean [43], fidelity [14], weighting [21], Chernoff bound [21], Kullback-Leibler measure [38], mutual information [49], and minimum description length [3].
2) **Learning methods**: In this category, the idea is to apply an online variant of the EM algorithm [37] to train the Gaussian mixtures in an incremental way as new data become available. Based on this idea, there have been several studies dedicated in particular to surveillance systems and video analysis. The work by Stauffer and Grimson [45] has become the standard formulation for the online learning of Gaussian mixtures which adopts an online EM-approximation based on recursive filter. It relies on a learning rule with a standard scheme:

$$\theta(t) = (1-\eta(t))\theta(t-1) + \eta(t) \bigtriangledown (x(t);\theta(t-1)) \quad (3)$$

where t, θ, η and $\bigtriangledown(.)$ are time, a model parameter (mean, variance), learning rate, and update amount. An analysis of the learning rate has been conducted in [27] showed in particular that when $\eta(t) = 1/t$, the algorithm will run into local optimum on a stationary data distribution and if $\eta(t) = \alpha$ ($\alpha << 1$), the most recent observations will have more impact and force the algorithm to take longer before it converges. The author in [27] proposed a solution that ensures temporal adaptability and fast convergence by letting η converge to α instead of 0. More interestingly each Gaussian has its own η computed based on the current likelihood estimates. This allows it to handle Gaussians independently. The algorithm proposed by Lee in [27] differs from Stauffer and Grimson's [44] with respect to the update rules of the learning rate η_j and the weight of the Gaussian components τ_j.

IV. OVERVIEW OF TYPE-2 FUZZY SYSTEMS

Type-2 fuzzy sets are a generalization of type-1 sets; T2 FS have membership grades that are themselves fuzzy. That is, the membership at any value is no more a single point but a function. There are two classes of type-2 MFs which can be applied, interval type-2 fuzzy sets and general type-2 fuzzy sets as shown in Fig. 1. Most of the published work has focused on interval type-2 fuzzy sets showing successful applications in robotics, control, image processing and time series [10], [20], [32]. The reason for using interval type-2 fuzzy sets lies in their simplicity. Generalized type-2 fuzzy sets have been also applied in [50], [11].

Often the issue is how to define the footprint of uncertainty (FOU) that specifies the notion of type-2 membership functions. FOU is obtained by means of a blurring procedure which allows us to assign an interval of values to each of element of the fuzzy set as shown in Fig. 1. The (possibilistic) weight of such values differ and we could use a three dimensional membership function to show this situation. By doing this, we are characterizing a type-2 fuzzy set. One of the most used methods for defining FOU consists of creating a margin, m, around the original type-1 fuzzy membership function. For instance, for a Gaussian membership function, the FOU allows the mean to lie within an interval $[m_1, m_2]$ [34]. A type-2 fuzzy set $\tilde{A}$ is defined as:

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) | \forall x \in X, \forall u \in J_x \subseteq [0,1]\} \quad (4)$$

where $\mu_{\tilde{A}}(x,u) \in [0,1]$. It follows that for a fixed x, we have a set of values, possibly with different weights, that we call

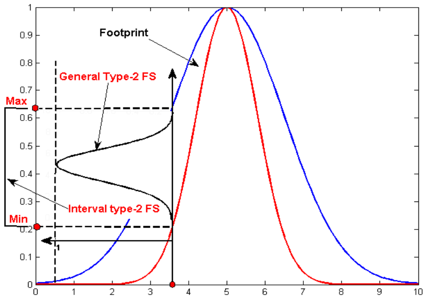


Fig. 1: Type-2 fuzzy sets

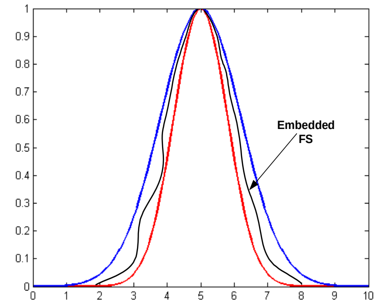


Fig. 2: Embedded MF

secondary membership function, $\mu_{\tilde{A}}(x', u)$), while the domain of this function, J_x, is called *primary membership function* of x. In this sense, FOU includes the set of all possible primary MFs embedded in the type-2 fuzzy set as shown in Fig. 2.

Therefore, the type-2 fuzzy set can be represented as the union of all its type-2 embedded fuzzy sets:

$$\tilde{A} = \sum_{j=1}^{n} \tilde{A}_e^j \tag{5}$$

where $\tilde{A}_e^j$ is the jth type-2 embedded fuzzy set given by:

$$\tilde{A}_e^j = \{(x_i, (u_i^j, f_{x_i}(u_i^j))), u_i^j \in \{u_{ik}, k = 1, \cdots, M_i\}\} \tag{6}$$

where $i = 1, \cdots, S$, S is the number of elements of $\tilde{A}_e^j$, $f_{x_i}(u_i^j)$ is the secondary grade at u_i^j, n is the number of embedded type-2 fuzzy sets and M_i is the number of the possibilistic weights (range of the secondary MFs) associated with each primary grade. That is: $n = \prod_{i=1}^{S} M_i$.

A type-2 fuzzy set can be thought of as a large collection of embedded type-1 sets each having a weight associated with it. At each value x, $x = x'$, the 2-D plane axes are u and $\mu_{\tilde{A}}(x', u)$ is called a vertical slice of $\mu_{\tilde{A}}(x, u)$. The secondary MF is a vertical slice of $\mu_{\tilde{A}}(x, u)$:

$$\mu_{\tilde{A}}(x', u) = \mu_{\tilde{A}}(x') = \int_{u \in J_{x'}} f_{x'}(u)/(u), \;\; J_{x'} \subseteq [0, 1] \tag{7}$$

where $0 \leq f_{x'}(u) \leq 1$. Because x' can be any $x \in X$, $\mu_{\tilde{A}}(x') = \mu_{\tilde{A}}(x)$ which is referred to as the secondary MF [50]. The secondary MF is a type-1 fuzzy set, hence the name secondary set. It can take different forms [34]. As mentioned earlier, the most widely used MF is the rectangularly shaped one, the so-called Interval Type-2 (IT2 MF) due to its simplicity. However, other MFs such as Gaussian, triangular, trapezoidal can also be used.

Referring to Eq. 7, if $f_x(u) = 1, \forall u \in J_x \subseteq [0, 1]$, then the secondary MFs are interval sets. If this holds $\forall x \in X$, then the secondary MF is referred to as interval type-2 MF which characterizes the interval type-2 fuzzy sets. A uniform uncertainty at the primary memberships of x means that we have interval secondary MFs. This is illustrated in Fig. 1. An interval type-2 set is in this sense represented by its domain interval, that is the lower and upper points $[l, u]$ as illustrated in Fig. 3 and which represent the bounds of FOU($\tilde{A}$). The lower and upper bounds are referred to $[\underline{\mu}_{\tilde{A}}(x), \overline{\mu}_{\tilde{A}}(x)]$.

Being interested in fuzzy rule-based classification systems, we need to draw the differences between T1 FRS and T2 FRS. At the operational level, T2 FRS differ from T1 FRS in the type of fuzzy sets and the operations applied on these sets. T2 fuzzy sets are equipped mainly with two newly introduced operators called the *meet*, $\sqcap$ and join, $\sqcup$ which correspond to the fuzzy intersection and fuzzy union. As shown in Fig. 4, T2 FRS at the structural level is similar to T1 FRS but contains an additional module, the *type-reducer*. Schematically, the structure of the T2 FRS consists of:

1) **Fuzzifier**:
It aims at converting the crisp inputs into input type-2 fuzzy sets. Very often and due to simplicity, only Singleton fuzzification is applied [34]. Formally given a crisp vector $\mathbf{x} = [x_1, \cdots, x_p]^t \in X_1 \times X_2 \times \cdots \times X_p = X$, the Singleton fuzzification transforms $\mathbf{x}$ into an input fuzzy set that has only a single point, $\mathbf{x}'$, of nonzero membership, that is:

$$\mu_{\tilde{A}_{\mathbf{x}}}(\mathbf{x}, u) = \begin{cases} 1/1 & \text{for } \mathbf{x} = \mathbf{x}' \\ 1/0 & forall \;\; \mathbf{x} \neq \mathbf{x}' \end{cases} \tag{8}$$

meaning that at $\mathbf{x} = \mathbf{x}'$ the secondary grade is 1 when the primary variable $u = 1$ and 0 at all other values of the primary variable ($\mathbf{x} \neq \mathbf{x}'$).

2) **Fuzzy rule base**:
It consists of fuzzy "IF-THEN" rules which are extracted from raw data by means of an inductive learning process. In a classification type-2 fuzzy rule system, the fuzzy rules for a C-class pattern classification problem with n input variable can be formulated as:

$$R_j \;\; \equiv \;\; \text{if } x_1 \text{ is } \tilde{A}_{j,1} \wedge ... \wedge x_n \text{ is } \tilde{A}_{j,n} \text{ then } C_i \tag{9}$$

where $\mathbf{x} = [x_1, \cdots, x_n]^t$ such that each x_i is a an input variable and $\tilde{A}_{r,i}$ the corresponding fuzzy terms in the form of type-2 fuzzy sets. We may associate these fuzzy sets with linguistic labels to enhance interpretability. C_i is a consequent class (i.e., one of the given C classes), and $j = 1, \cdots, N$ is the number of fuzzy rules.
Multidimensional fuzzy sets can be used also to build FRS's. The rules are expressed in the form:

$$R_j \;\; \equiv \;\; \text{if } x \text{ is } K_p \text{ then } C_i \tag{10}$$

where K_p is a cluster. The rule means that if the sample x is *close* to K_p then x should belong to class C_i.

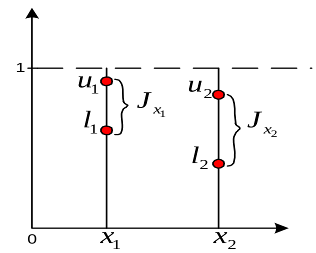


Fig. 3: Interval type-2 fuzzy sets

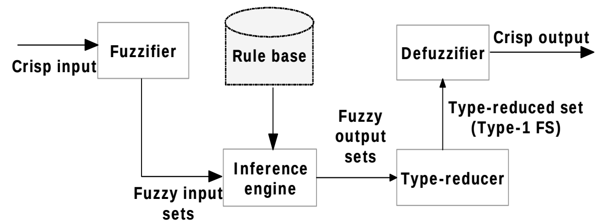


Fig. 4: Type-2 fuzzy logic system

Motivated by interpretability issues, in this paper, we consider the former form (Eq. 9). Type-2 fuzzy rules are generated. In particular, it will describe mechanisms to produce and update the type-2 antecedents of the rules in an online manner to accommodate data arriving sequentially and continuously over time. Such rules should not assume that old data is available.

3) **Fuzzy inference engine**:
The inference engine computes the output of type-2 fuzzy sets by combining the rules. Specifically, the meet operator is used to connect the type-2 fuzzy propositions in the antecedent. The degree of activation of the jth rule using the n input variables is computed as:

$$\tilde{\beta}_j(x) = \sqcap_{q=1}^{n} \mu_{\tilde{A}_{j,q}}(x_q), j = 1, \cdots, N \quad (11)$$

The meet operation that replaces the fuzzy 'and' in T1 FRS is given as follows:

$$\tilde{A} \sqcap \tilde{B} = \sum_{u \in J_x} \sum_{w \in J_x} f_x(u) * g_x(w)/(u \wedge w), \ x \in X \quad (12)$$

If we use the interval Singleton T2 FRS, the meet is given for input $x = x'$ by the firing set, i.e.,

$$\begin{aligned} \tilde{\beta}_j(x) &= [\underline{\beta}_j(x), \overline{\beta}_j(x)] \\ &= [\underline{\mu}_{A_{j,1}}(x_1) * \cdots * \underline{\mu}_{A_{j,n}}(x_n), \\ &\quad \overline{\mu}_{A_{j,1}}(x_1) * \cdots * \overline{\mu}_{A_{j,n}}(x_n)] \end{aligned} \quad (13)$$

In this paper we adopt the Gaussian membership function given as follows:

$$\mu_A(x) = exp\left\{-\frac{(\mathbf{x}-m)^2}{2\sigma^2}\right\} = \mathcal{N}(m, \sigma; x) \quad (14)$$

where m and σ are the mean and the standard deviation of the function. To generate the lower and upper membership functions, we rely on two possible methods:

- The interval set for the Gaussian function with **uncertain mean** is given by:

$$\overline{\mu}_{\tilde{A}}(x) = \begin{cases} \mathcal{N}(m_1, \sigma; x) & x < m_1 \\ 1 & m_1 \leq x \leq m_2 \\ \mathcal{N}(m_2, \sigma; x) & x > m_2 \end{cases} \quad (15)$$

$$\underline{\mu}_{\tilde{A}}(x) = \begin{cases} \mathcal{N}(m_2, \sigma; x) & x \leq \frac{m_1+m_2}{2} \\ \mathcal{N}(m_1, \sigma; x) & x > \frac{m_1+m_2}{2} \end{cases} \quad (16)$$

- For the Gaussian with **uncertain deviation**, concentration and dilation hedges are used to generate the footprint. These are given as follows:

$$\overline{\mu}_{\tilde{A}}(x) = \mu_{DIL(A)}(x) = [\mu_A(x)]^{1/2} \quad (17)$$

$$\underline{\mu}_{\tilde{A}}(x) = \mu_{CON(A)}(x) = [\mu_A(x)]^2 \quad (18)$$

The graphical representations of these two types of Gaussian are portrayed in Fig. 5 and Fig. 6. Moreover, according to Eq. 14, we will have the following expressions:

$$\overline{\mu}_{\tilde{A}}(x) = \mathcal{N}(m, \sqrt{2}\sigma; x) \quad (19)$$

$$\underline{\mu}_{\tilde{A}}(x) = \mathcal{N}(m, \frac{\sigma}{\sqrt{2}}; x) \quad (20)$$

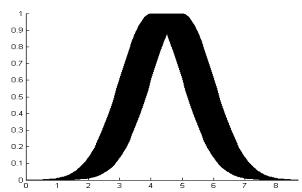


Fig. 5: Uncertain mean

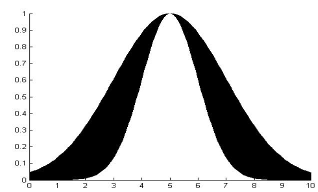


Fig. 6: Uncertain deviation

4) **Type-reducer**:
It converts T2 FSs obtained as output of the inference engine into T1 FSs. Type-reduction for FRS (Mamdani and Sugeno models) was proposed by Karnik and Mendel [24], [35]. This will not be adopted in our case, since the rules' consequent represents the label of a class.
Traditionally in Type-1 fuzzy classification systems, the output of the classifier is determined by the rule that has the highest degree of activation:

$$\text{Winner} = C_{j^*}, \quad j^* = \underset{1 \leq j \leq N}{\text{argmax}} \ \beta_j \quad (21)$$

where β_j is the firing degree of the j rule. In type-2 fuzzy classification systems, we have an interval $\tilde{\beta}_j = [\underline{\beta}_j, \overline{\beta}_j]$ as defined in Eq. 13. Therefore, we compute the winning class by considering the center of the interval $[\underline{\beta}_j(x), \overline{\beta}_j(x)]$, that is:

$$\begin{aligned} \text{Winner} &= C_{j^*}, \text{where} \quad (22) \\ j^* &= \underset{1 \leq j \leq N}{\text{argmax}} \ \frac{1}{2}(\underline{\beta}_j(x) + \overline{\beta}_j(x)) \quad (23) \end{aligned}$$

V. 2G2M: A Growing Gaussian Mixture Model

The algorithm 2G2M shown in Algorithm 1 is an incremental algorithm like the ones mentioned in Sec. III. It extends the algorithm presented in [27] in many aspects. The novelties concern the processing of the labeling information, the explicit handling of multi-dimensional data, handling the complexity of the model generated as will be explained later, and finally the re-structuration to accommodate the named changes. The similarities concern mainly the online update procedure which is based on an online approximation of the standard expectation maximization method provided in [45].

The symbols appearing in Alg. 1 are defined as follows:

- K : the number of Gaussians $\phi_j,\ j = 1 \cdots K$
- $\boldsymbol{\mu}_j$: the center of the jth Gaussian
- Σ_0 : the initial covariance matrix of the Gaussians (= $k_0 I_D$ proportional to the identity matrix of size D -the number of features-)
- α : the learning rate
- T_Σ : the closeness threshold (Eq. 24)
- τ_j : the weights of the jth Gaussian
- c_j : the sum of the expected posterior (a sufficient statistics in the EM method) of the jth Gaussian
- L_j : the label of the jth Gaussian (cluster)

Moreover, Alg. 1 is characterized by the following aspects:

A. Probability of Match

Equation 24 has to be adapted to enable semi-supervised learning. We compute the probability of matching the input x_i with the jth Gaussian by considering the following cases:

1) The input x_i is unlabeled: all Gaussians need to be considered during the match operation (Eq. 24).
2) The input x_i is labeled and jth Gaussian is unlabeled: such Gaussian is considered during the match operation.
3) The input x_i and jth Gaussian have the same label: the Gaussian is of course considered.
4) The input x_i and jth Gaussian do not have the same label: the Gaussian is discarded.

Moreover the distance $d_M(x_i, \phi_j)$ between an input x_i and a Gaussian $\phi_j(\mu_j, \Sigma_j)$ in Eq. 24 is the Mahalanobis distance expressed as follows:

$$d_M(x_i, \phi_j) = \sqrt{(x_i - \mu_j)^T \Sigma_j^{-1} (x_i - \mu_j)} \tag{35}$$

B. Processing of Unlabeled Data

To achieve a semi-supervised learning setting, the 2G2M algorithm is equipped with a mechanism for accommodating labeled and unlabeled data. The Gaussians are updated and eventually created according to the label of both the Gaussian and the incoming input. Similar to the cases outlined in Sec. V-A, the accommodation of new data is done as follows:

1) If the incoming input is not labeled, simply find the highest matching Gaussian, add the contribution of the input and update all Gaussians.
2) If the incoming input is labeled, then
   a) If the highest matching Gaussian is unlabeled, label it with the label of the input.
   b) If the highest matching Gaussian and the new input have different labels, then create a new Gaussian.
   c) If the highest matching Gaussian and the incoming input have the same label, add the contribution of the new input and update all Gaussians.

Algorithm 1 : Steps of 2G2M

Set parameters: K, Σ_0, α, T_Σ

Initialization:
$\forall j = 1 \cdots K, (\tau,\ c,\ \boldsymbol{\mu},\ \Sigma,\ L)_j = (\alpha,\ 1,\ \infty,\ \Sigma_0,\ 0)$

Given a new input x_i, compute the probability of match of the input with the Gaussians: $\forall j = 1 \cdots K$

$$p_j = \begin{cases} \tau_j \phi_j(x_i; \boldsymbol{\mu}_j, \Sigma_j) & \text{if } d_M(x_i, \phi_j) < T_\Sigma \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

Let $R = \{j | p_j > 0\}$ be the index set of Gaussians matching the input

createGaussian $\leftarrow (R == \emptyset)$ (i.e., no match is found: $\nexists j, p_j > 0$)

if not createGaussian **then**

  Compute the index of highly matching Gaussian:

$$w = \underset{j=1\cdots K}{\operatorname{argmax}}\ \{p_j\} \tag{25}$$

  if x_i is labeled **then**

    if the wth Gaussian is not labeled **then**

      Assign the wth Gaussian the label of x_i

    else if the wth Gaussian and x_i have different labels **then**

      Set createGaussian $\leftarrow$ $true$

    end if

  end if

  if not createGaussian **then**

    for $j = 1 \cdots K$ **do**

      Compute the expected posterior:

$$q_j = \frac{p_j}{\sum_{k=1\cdots K} p_k} \tag{26}$$

      Update the parameters of the Gaussian:

$$c_j = c_j + q_j \tag{27}$$

$$\tau_j(t) = (1-\alpha)\tau_j(t-1) + \alpha q_j \tag{28}$$

$$\eta_j = q_j \left(\frac{1-\alpha}{c_j} + \alpha \right) \tag{29}$$

$$\boldsymbol{\mu}_j(t) = (1-\eta_j)\boldsymbol{\mu}_j(t-1) + \eta_j x_i \tag{30}$$

$$\Sigma_j(t) = (1-\eta_j)\Sigma_j(t-1) + \eta_j (x_i - \boldsymbol{\mu}_j(t-1))^2 \tag{31}$$

    end for

  end if

end if

if createGaussian **then**

  Decay the weight of all Gaussians

$$\forall j = 1 \cdots K, \quad \tau_j(t) = (1-\alpha)\tau_j(t-1) \tag{32}$$

  Remove the less contributing Gaussian and create a new one initialized with the new input:

$$m = \underset{j}{\operatorname{argmin}}\ \{\tau_j\} \tag{33}$$

$$(\tau, c, \boldsymbol{\mu}, \Sigma, L)_m = (\alpha, 1, x_i, \Sigma_0, 0) \tag{34}$$

end if

Split the largest Gaussian if volume $> T_{split}$

Merge the closest Gaussians if KL distance $< T_{merge}$

Normalize the τ_j's

C. Model Refinement

To lower the complexity and enhance the model accuracy, we rely on refinement-based methods that use the two operations: *split* and *merge* as explained in Sec. III. Due to the presence of unlabeled data, Gaussians may overlap but also may cover several classes. To avoid these situations, the algorithm has been equipped with mechanisms that perform split and merge operations on the Gaussians.

1) Split Operation: In order to keep the algorithm accurate and precise, large Gaussians (considered to be as "too big") are split into two smaller Gaussians. The chosen criterion for deciding to split a Gaussian $\phi(\mu, \Sigma)$ refers to its volume:

$$V(\phi(\mu, \Sigma)) = det(\Sigma) \tag{36}$$

The Gaussian $\phi(\tau, \mu, \Sigma)$ with the maximum volume V above a given threshold T_{split} is split into two Gaussians ($\phi_1(\tau_1, \mu_1, \Sigma_1)$, $\phi_2(\tau_2, \mu_2, \Sigma_2)$) along its dominant principal component (Fig. 7a). The τ's are the weights of the corresponding Gaussians. The resulting Gaussians after split are assigned a weight $\tau/2$. Let v the (normalized) eigenvector corresponding to the largest eigenvalue λ, and a (which is set to 0.8) a split factor that determines how far the centers of the two new Gaussians are apart from each other. Clearly the expression of Δv below shrinks the original ellipse along the dominant principal component by a certain proportion which serves to create the new centers and the covariance matrices.

$$\begin{cases} \Delta v = \sqrt{a\lambda} v \\ \tau_1 = \tau_2 = \frac{\tau}{2} \\ \mu_1 = \mu + \Delta v \\ \mu_2 = \mu - \Delta v \\ \Sigma_1 = \Sigma_2 = \Sigma - \Delta v \Delta v^T \end{cases} \tag{37}$$

2) Merge Operation: In order to reduce the complexity, Gaussians of the same label can be merged. The merge operation combines the two closest Gaussians into one single Gaussian. It relies on the Kullback-Leibler divergence (*kld*) between two D-dimensional Gaussians ($\phi_1(\mu_1, \Sigma_1)$, $\phi_2(\mu_2, \Sigma_2)$) as a merge criterion which reads as follows [42]:

$$kld(\phi_1, \phi_2) = \; log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) + tr(\Sigma_2^{-1}\Sigma_1) + \vartheta^T \Sigma_1^{-1} \vartheta - D \tag{38}$$

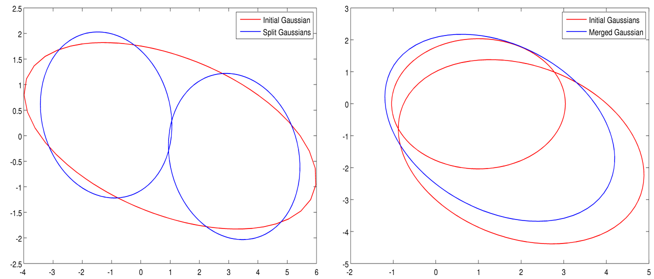
where $\vartheta = (\mu_2 - \mu_1)$, tr indicates the trace. Since *kld* is asymmetric, a symmetrized variant (*skld*) of the divergence is used for similarity estimation:

$$skld(\phi_1, \phi_2) = \frac{1}{2}\left(kld(\phi_1, \phi_2) + kld(\phi_2, \phi_1)\right) \tag{39}$$

The pair of Gaussians ($\phi_1(\tau_1, \mu_1, \Sigma_1)$, $\phi_2(\tau_2, \mu_2, \Sigma_2)$) whose $skld$ is minimum and falling below a threshold T_{merge} are combined into a new Gaussian $\phi(\tau, \mu, \Sigma)$ (Fig. 7b) using:

$$\begin{cases} f_1 = \frac{\tau_1}{\tau_1 + \tau_2} \\ f_2 = \frac{\tau_2}{\tau_1 + \tau_2} \\ \tau = \tau_1 + \tau_2 \\ \mu = f_1\mu_1 + f_2\mu_2 \\ \Sigma = f_1\Sigma_1 + f_2\Sigma_2 + f_1 f_2 (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \end{cases} \tag{40}$$

3) Cannot-Merge Link: There is a risk that a Gaussian be split, then re-merged straight away. To avoid this phenomenon, we endow each split or merged Gaussian with a **cannot-merge link** (CML) similar to "must-link" and "cannot-link" in constraint-based clustering. CML is a positive integer that decreases over time and that disables temporarily subsequent split or merge of the same Gaussians as long as it is greater than 0. In the experiments, CML is empirically set to 50.



(a) Split of a Gaussian (b) Merge of Gaussians

Fig. 7: Split and merge operations

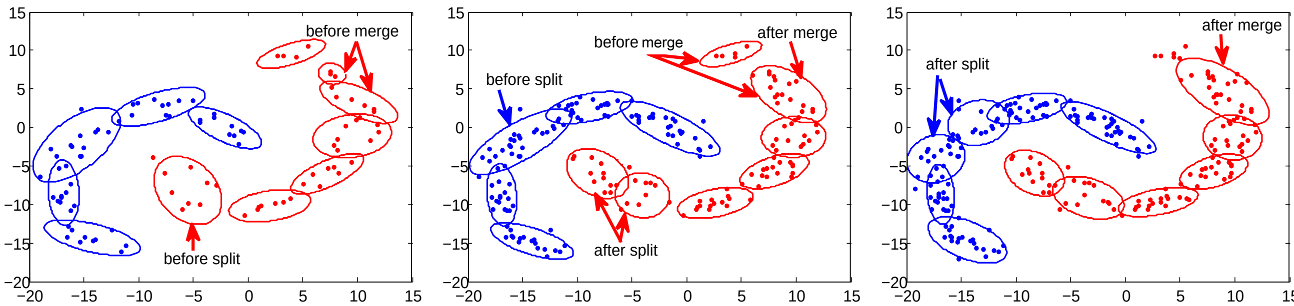
Figure 8 shows the results of merge and split operations occurring at different time points during clustering of the banana dataset. The accommodation of new data can lead to either modifying the shapes and orientations of the clusters, merging and splitting of existing clusters, or creating new ones. The *virtual* ellipses show approximately the region of influence of each Gaussian around its center, since 2G2M outputs only the means, the covariance matrices and the mixing parameters.

D. Parameter optimization

Due to their number (maximum number of clusters K, thresholds T_Σ - T_{merge} - T_{split}, learning rate α, initial spread Σ_0), setting the parameters is in general a difficult problem. Usually they have to be set for each dataset. However, in this paper we rely on genetic algorithms (GA), widely used for solving optimization problems, to provide a set of parameters corresponding to a local optimum of the fitness function. We choose the fitness function to be the number of misclassified examples by the 2G2M algorithm. While the parameters, α and K, are considered as user-defined parameters, the remaining ones, T_Σ, T_{merge}, T_{split} and k_0 are determined by GA.

Given the online setting, we do not have access to all data. Hence, to ensure online optimisation of these parameters, we propose an original way of optimizing the parameters online and continuously. We did the following: We generated two versions of GF2FC, one running *online* which we use in real time and an *offline* version involving only 2G2M that runs only periodically on small batches of labeled data. For this offline version, data are accumulated and whenever a batch of n labeled examples is collected, it is used in a GA loop to find the optimal parameter values. These are then transferred into the online version. To ensure the optimization continuity, the top m individuals of a given cycle are used to initialize the GA population for the next cycle.

Note that the parameters themselves are encoded together as a chromosome with the classification accuracy measure as a fitness function. The GA parameters used in all experiments are: |Population|=100, number of iteration=100, recombination probability =0.8, mutation probability=0.3, selection method: linear ranking, and crossover method: one-point crossover. The size of the batch is $n = 30$ and the number of chromosomes retained from the next generation is $m = 10$.



(a) After 100 data points (b) After 200 data points (1 merge + 1 split) (c) Final clustering (1 merge + 1 split)

Fig. 8: Clustering of the Banana shaped dataset.

Algorithm 2 : Steps of the rule generation process

---

for a new input **do**
  Run Alg. 1 to cluster the new input.
  if the new input is labeled **then**
    Re-select the meaningful features using the incremental feature selection algorithm in [5] if desired;
  end if
  Generate the appropriate fuzzy sets by projecting the clusters on the selected features and form the rules.
end for

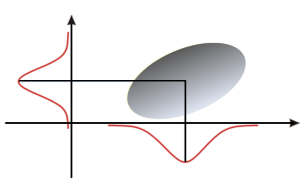


Fig. 9: Projection of a bidimensional Gaussian

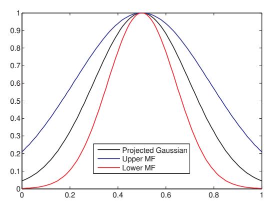


Fig. 10: Contraction and dilatation of type-1 MF

VI. Fuzzy Rule Generation

The main steps for rule generation are displayed in Alg. 2. These steps are kept general to ease understanding. In the following, the issues related to rule generation are highlighted.

A. *Projection of Gaussians*

Once the new input has either generated a new Gaussian or updated an existing one, we use projections of the n-dimensional Gaussians on the n axes to determine the MFs of the T2 FRS (Fig. 9). For a Gaussian $\mathcal{N}(\mu, \Sigma)$, the projected MF on axis d is a unidimensional Gaussian $\mathcal{N}(m, \sigma; x)$ (Eq. 14), whose parameters are determined as follows:

$$\begin{cases} m = \mu_d \\ \sigma = \sqrt{\Sigma_{dd}} \end{cases} \tag{41}$$

where μ_d is the d-th coordinate of the vector μ and Σ_{dd} is the d-th diagonal term of the covariance matrix Σ. This Gaussian defines a type-1 membership function.

It is worthwhile to recall that with multi-dimensional fuzzy sets, the number of rules will be the same as that of fuzzy partitions and hence no projection is needed. However, unidimensional fuzzy sets offer a better degree of interpretability, despite the lengthy premises of rules that can be generated (as discussed in Sec. III). Unfortunately the complexity remains high if the data is highly dimensional. This paper focuses on generating rules whose form is shown by Eq. 9. Thus, the features are explicitly used. Thus, to enhance the transparency and the compactness of the rules, it is important that the if-part of the rules does not involve many features [18]. Very often, low classification performance results from non-informative features. To get rid of those features, usually feature selection methods are used which are of two classes [12]: filters and wrappers. Many methods do not lend themselves the possibility to find the most relevant features from incrementally arriving data. They assume that the whole data is available or at least there is sufficient amount of data.

In order to cope with the complexity of the rules, enhancing the transparency by reducing the number of features in the rules, the incremental feature selection algorithm described in [5][8] is used. The importance of incremental feature selection comes from the fact that the set of selected features change over time. The algorithm uses an incremental version of Fishers interclass separability criterion. As new data instances arrive, some features may be neglected in the rules and some others may be included. The premises of the rules are dynamically and systematically updated. The main goal is that at any time, the rule base reflects the semantical contents of the data already used. In our experiments, when the data dimensionality exceeds a certain number n (one can fix this number at wish), the incremental feature selection algorithm is applied and only the first n important features are retained. It is, however, up to the user to decide, either to keep all features or to select a particular number of them over time.

B. *Rule base Optimization*

In order to enhance the quality of the T2 fuzzy rules [18], we use two methods:

1) *Cluster refinement*: by merging the highly overlapping Gaussians using the Mahalanobis distance, a lower number of type 2 FS's can be obtained which will be used to generate the fuzzy rules.
2) *Rule refinement*: once the Gaussians are obtained, rules are then generated. These rules undergo an optimization

process which consists of merging the membership functions (fuzzy sets) using the Jaccard's similarity measure for type-2 FS's which is based on the analysis provided in [31]. Those exceeding a threshold are merged using an adapted formula for fusing univariate Gaussians like: $\mu_{new} = (s_1 * \mu_1 + s_2 * \mu_2)/(s_1 + s_2)$, $s_{new} = s_1 + s_2$ where $s_{\{new,1,2\}} = 1/\sigma^2_{\{new,1,2\}}$. One could also merge the underlying clusters and then consider only the targeted dimension. Another merge method would be to use the Join operation [34], but rules premises will not be expressed as Gaussians in a straightforward way.

Both methods yield quite compact rules. The first one is easy to perform given that the clustering algorithm, 2G2M, is already equipped with a merge operation. It is however difficult in general to re-define the closeness threshold used in the clustering algorithm, because it is originally and automatically determined by means of a GA-based optimization process on batches of data (See Sec. V-D). The second method on the other hand requires also the definition of the similarity threshold, but it is easier to do so; the higher it is, the less frequent the merge is. For instance with the second method, we can generate rules of the form:

$$R_j \equiv \text{ if } x_i \text{ is } (\tilde{A}_{j,i} \vee ... \vee \tilde{A}_{j,i'}) \wedge ((x_k \text{ is } \tilde{A}_{j,k} \vee ... \vee \tilde{A}_{j,k'}) \vee (x_l \text{ is } (\tilde{A}_{j,l} \vee ... \vee \tilde{A}_{j,l'})) \text{ then } C_n \quad (42)$$

For the Banana-shaped data, the fuzzy partitions obtained without any optimization are shown in Fig. 11a. Figures 11b and 11c show the optimization results after applying method (1) and method (2). For the purpose of legibility, type-1 fuzzy membership functions are used. Type-2 fuzzy rules are shown in Tab. I. In particular, we notice that rules generated after method 2 are fewer than those generated after method 1. The number of original rules is usually much higher. Note that in the experiments (Sec. VII), we used the original rule set.

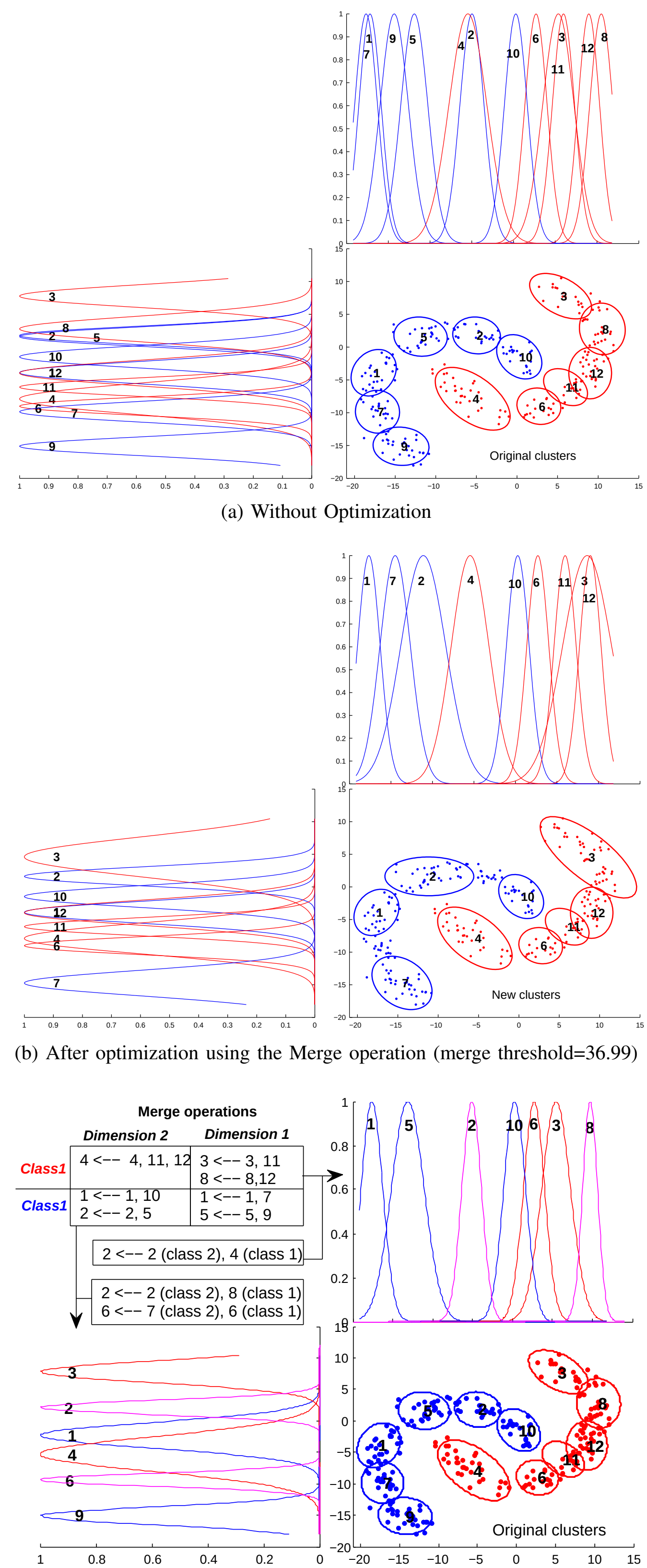
To generate linguistic terms in the rules, the Gaussians will be ordered from low/small to high/large with respect to each dimension. Then those linguistic terms can replace the Gaussians in the rules. However, the meaning of these terms changes dynamically over time as new data arrive. For example, at time t, "large" on the x-axis corresponds to values between 20 and 30. In the future, "large" may indicate values between 50 and 60. A term can, however, be assigned to only one function, but it may represent many clusters.

C. Unlabeled Clusters

Because the 2G2M algorithm (Alg. 1) accommodates both labeled and unlabeled data, the resulting clusters can be either labeled or unlabeled. The unlabeled clusters do not produce any rule, although it is possible to generate rules by computing the class of the neighboring clusters and then use majority voting to assign a label to the unlabeled cluster and therefore, this uncertain aspect is implicitly taken care of by the type-2 fuzzy rule system.

VII. Empirical Evaluation

We evaluate the proposed classifier GT2FC using two sets of experiments. For the sake of illustration, the first set of



(a) Without Optimization

(b) After optimization using the Merge operation (merge threshold=36.99)

(c) After optimization using the Jaccard's similarity (Threshold=0.4): the legend shows which functions are merged. The new function is assigned the smallest number (id) of the merged functions. The clusters (bottom right) are the original ones. Functions with magenta color result from merge of functions with different classes.

Fig. 11: Optimization process

TABLE I: Fuzzy rules for the Banana-shaped dataset

Rule	Antecedent	C	Antecedent	C
1	x_1 IN N(-17.66, [0.96,1.93]) $\wedge$ x_2 IN N(-3.89, [1.17,2.34])	2	x_1 IN N(-17.65, [0.96, 1.92]) $\wedge$ x_2 IN N(-3.89, [1.17, 2.34])	2
2	x_1 IN N(-4.97, [1.02,2.04]) $\wedge$ x_2 IN N(1.78, [0.97,1.94])	2	x_1 IN N(-11.09, [1.91, 3.82]) $\wedge$ x_2 IN N(1.61, [1.01, 2.02])	2
3	x_1 IN N(5.37, [1.32,2.65]) $\wedge$ x_2 IN N(7.78, [1.20,2.39])	1	x_1 IN N(8.631, [2.076, 4.153]) $\wedge$ x_2 IN N(4.609, [2.137, 4.275])	1
4	x_1 IN N(-5.46, [1.60,3.20]) $\wedge$ x_2 IN N(-7.90, [1.65,3.30])	1	x_1 IN N(-5.46, [1.60, 3.20]) $\wedge$ x_2 IN N(-7.89, [1.65, 3.30])	1
5	x_1 IN N(-11.88, [1.14,2.28]) $\wedge$ x_2 IN N(1.60, [1.01,2.03])	2	x_1 IN N(2.69, [0.93, 1.87]) $\wedge$ x_2 IN N(-9.00, [0.96, 1.92])	1
6	x_1 IN N(2.70, [0.94,1.88]) $\wedge$ x_2 IN N(-9.01, [0.96,1.92])	1	x_1 IN N(-14.48, [1.29, 2.58]) $\wedge$ x_2 IN N(-14.74, [1.37, 2.74])	2
7	x_1 IN N(-17.18, [0.94,1.89]) $\wedge$ x_2 IN N(-9.85, [1.08,2.16])	2	x_1 IN N(0.28, [0.96, 1.93]) $\wedge$ x_2 IN N(-1.477, [1.171, 2.343])	2
8	x_1 IN N(10.51, [0.98,1.96]) $\wedge$ x_2 IN N(2.78, [1.34,2.68])	1	x_1 IN N(5.98, [0.93, 1.87]) $\wedge$ x_2 IN N(-6.09, [0.97, 1.94])	1
9	x_1 IN N(-14.28, [1.20,2.39]) $\wedge$ x_2 IN N(-15.12, [0.98,1.96])	2	x_1 IN N(9.01, [0.91, 1.82]) $\wedge$ x_2 IN N(-3.96, [1.34, 2.68])	1
10	x_1 IN N(0.28, [0.97,1.93]) $\wedge$ x_2 IN N(-1.48, [1.17,2.34])	2		
11	x_1 IN N(5.98, [0.94,1.88]) $\wedge$ x_2 IN N(-6.10, [0.97,1.94])	1		
12	x_1 IN N(9.02, [0.91,1.83]) $\wedge$ x_2 IN N(-3.96, [1.34,2.69])	1		
Rule	**Antecedent**	**C**		
1	x_1 IN N(-17.60, [0.968,1.93]) $\wedge$ (x_2 IN N(-2.31, [1.42,2.85]) $\vee$ (x_2 IN N(-9.45, [2.04, 4.08]))	2		
2	(x_1 IN N(-5.27, [2.62, 5.24]) $\vee$ x_1 IN N(-13.10, [1.44,2.89])) $\wedge$ x_2 IN N(1.61, [1.01,2.02])	2		
3	x_1 IN N(-13.10, [1.44,2.89]) $\wedge$ (x_2 IN N(1.61, [1.01,2.02]) $\vee$ x_2 IN N(-15.12, [0.98,1.96]))	2		
4	x_1 IN N(5.47, [1.28,2.56]) $\wedge$ (x_2 IN N(7.78, [1.19,2.39]) $\vee$ x_2 IN N(-6.10, [0.97,1.94]))	1		
5	(x_1 IN N(-5.27, [2.62, 5.24]) $\vee$ x_1 IN N(5.470, [1.28,2.56]) $\vee$ x_1 IN N(9.79, [1.89, 3.78]) $\wedge$ x_2 IN N(-5.30, [1.91,3.83])	1		
6	x_1 IN N(2.69, [0.93,1.87]) $\wedge$ x_2 IN N(-9.45, [2.04, 4.08])	1		
7	(x_1 IN N(0.28, [0.97,1.93]) $\vee$ x_1 IN N(-17.60, [0.96,1.93])) $\wedge$ x_2 IN N(-2.31, [1.42,2.85])	2		

experiments is carried out on a synthetic data set, a Banana shaped dataset. In the second set of experiments, we use real-world datasets: classic datasets (from UCI [17]) and an ambient intelligence dataset [20]. We will investigate many aspects to give insight into the approach. In particular, we study the behavior of online adaptation which is measured by Eq. 43 that expresses the *current error rate* as:

$$e(t) = \frac{misses(t)}{seen\_so\_far(t)} \tag{43}$$

where $misses(t)$ represents the number of misclassified data until time t and $seen\_so\_far(t)$ represents the number of presentations (inputs) seen so far.

The datasets are split into labeled and unlabeled sets in two ways:

1) *random decreasing*: the desired proportion of labeled data is randomly selected and the position in the sequence of the whole data is also random but the frequency of occurrence of labeled examples decreases over time. That is, labeled examples are more frequent at the beginning of the sequence, but become rare and then non-existent towards the end of the sequence. The idea reflects the fact that the system may need more labeled data in earlier stages of the training. Given that the datasets used in this paper are labeled, samples satisfying the following probability are retained, while the rest is considered unlabeled:

$$\forall i \in \{1, \cdots, n\}, p_{label}(x_i) = 1 - \frac{i}{n} \tag{44}$$

where i is the order of x_i in the input sequence and n is the size of the data(i.e, sequence length).

2) *random selection*: the desired proportion of labeled data is randomly selected. The position in the sequence of the whole data is random.

TABLE II: Datasets used to test the algorithm

Dataset	Features	Classes	Instances
Iris	4	3	150 [50 50 50]
Wine	13	3	178 [59 71 48]
Banana shaped	2	2	400 [200 200]

A. Classic Datasets

The first class of datasets consists of: *Iris*, *Wine* which are obtained from the UCI repository [17] and a *banana-shaped* dataset which is artificially generated (already used in the previous sections). The description of these datasets is shown in Tab. II. The banana-shaped dataset consists of a synthetic bidimensional set of 400 points. It has been chosen due to:

- its difficulty for semi-supervised learning algorithms because of its non-linear separability,
- its shape which is not Gaussian and
- the possibility to visualize it easily, since it is 2-dimensional.

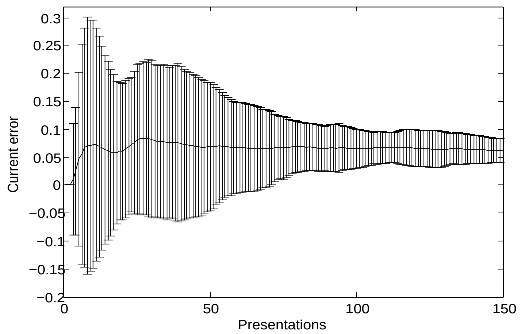
Using the online optimization approach presented in Sec. V-D, the value of the parameters T_Σ, T_{merge}, T_{split} and k_0 will be automatically and over time computed. For the sake of illustration, Tab. III shows the evolution of the parameter setting (early stage, middle stage and final stage).

Considering the two modes of labeled data selection, *random decreasing* and *random*, we obtain the results shown in Figs. 12, 13, and 14 which are averaged over 30 runs. Three main conclusions can be drawn:

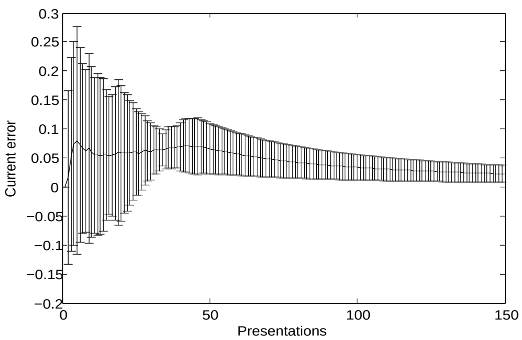
- the adaptation capability of the algorithm 2G2M: the learning effect is clear and the accuracy of the classification increases as more data are presented.
- the *random decreasing* mode of labeled data selection looks slightly better than the *random* mode

TABLE III: Excerpt of the parameter settings for 2G2M resulting from GA optimization for the classic and the banana datasets.

Parameter	Iris			Wine			Banana		
	Early	Middle	Final	Early	Middle	Final	Early	Middle	Final
K (maxnumber of clusters)	10	10	10	10	10	10	20	20	20
α (learning rate)	0.25	0.25	0.25	0.25	0.25	0.25	0.01	0.01	0.01
T_{Σ} (closeness threshold)	3.16	4.83	5.01	14.30	10.26	10.05	3.73	5.40	5.08
T_{merge} (merge threshold)	14.74	12.12	10.13	132.33	159.33	150.67	19.5	18.14	15.05
T_{split} (split threshold)	9.33	2.42	5.25	6.00	15.52	10.97	6.34	9.80	10.19
k_0 (initial Gaussians' spread)	0.14	1.31	0.009	1.72	1.99	4.05	3.18	2.81	1.15

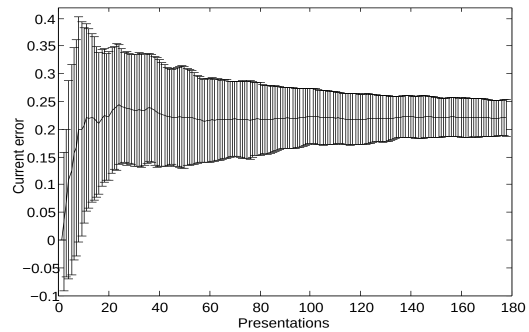


(a) Random selection of labeled examples

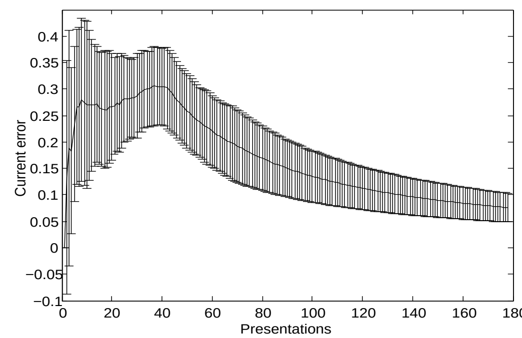


(b) Random decreasing selection of labeled examples

Fig. 12: Effect of the selection method (Iris data)



(a) Random selection of labeled examples



(b) Random decreasing selection of labeled examples

Fig. 13: Effect of the selection method (Wine data)

- the variability of the results (standard deviation) at the beginning of the learning is higher than that towards the end of the data presentation.

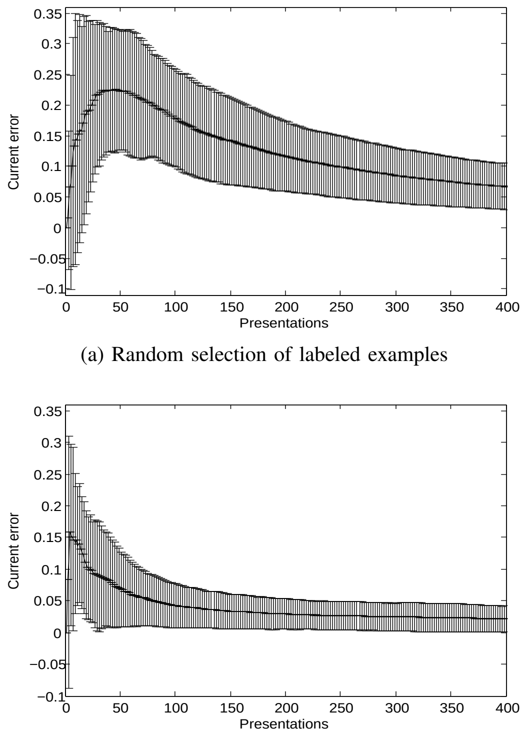
Moreover, the algorithm has been tested on these data sets by splitting them into $60\% - 40\%$ (60% for training, 40% for testing) to test the performance. The accuracy measure (i.e., the ratio of the correctly classified data) has been used to evaluate the performance of the type-2 fuzzy classifier. Different configurations (proportion of the labeled data, mode of labeled data selection, the maximum number of clusters) have been tested yielding the final classification results shown in Tab. IV after 30 runs. The results look highly promising.

Table V shows the 2G2M behavior over the banana data. It shows in particular new labeled/unlabeled data (0 for unlabeled, 1 for labeled; the true label is between brackets), the matching Gaussian - Winner (the label of the winner is between brackets), the matching probability, the actual number of Gaussians, and the boolean operations creation, merge and split (they are set to 1 if they are executed, otherwise 0).

B. Real-World Application: Ambient Intelligence

Ambient intelligence (AmI) aims at exploiting sensed information to seamlessly assist the inhabitants of a smart space to carry on their daily activities. Intelligent monitoring solutions of the living space intend to deal efficiently with various aspects like comfort, security and safety, productivity, and energy efficiency for the benefit of the occupants. The major challenges of intelligent environments research include the modeling of the occupants activities and actions, the dynamic evolution of the occupant behavior, the interaction, and the smart monitoring of the physical infrastructure.

A suitable monitoring system for this application must be evolving, in the sense that as new data become available, it has to self-adapt in an online manner. Because, sensors are prone to faults and noise, type-2 fuzzy modeling is very appropriate for dealing with such application. Designed for applications confronting uncertainty, GT2FC is tested on datasets stemming from an AmI environment which is an intelligent student



(a) Random selection of labeled examples

(b) Random decreasing selection of labeled examples

Fig. 14: Effect of the selection method (Banana data)

TABLE IV: Average accuracy results using the classic datasets

Dataset	Ratio of labeled data	Clusters	Accuracy (%)
Iris	10% (random)	10	89.17±0.10
	30% (random)	15	93.25±0.03
	10% (random decr.)	10	93.58±0.04
	30% (random decr.)	15	94.25±0.05
Wine	10% (random)	10	80.28±0.33
	30% (random)	10	82.41±0.09
	10% (random decr.)	10	81.25±0.02
	30% (random decr.)	10	85.02±0.10
Banana	10% (random)	20	97.00±0.05
	30% (random)	20	99.15±0.02
	10% (random decr.)	20	98.13±0.03
	30% (random decr.)	20	98.98±0.02

dormitory (called iDorm) [4], [16], [20]. The iDorm test bed serves to predict the different classes of user actions.

The dataset is generated by recording student activities in the iDorm which is equipped with embedded sensors, actuators, processors and heterogeneous networks that are concealed in a way letting the users behave naturally [20]. The data (7 input features) are provided by the sensors: internal light level, external light level, internal temperature, external temperature, chair pressure, bed pressure, occupancy and time. The data used in the following experiments concern two inhabitants: D_1

TABLE V: Execution of running 2G2M on the banana dataset

data	labeled[label]	Matching prob.	Winner	# Gaussians	Creation
1	0[2]	0.00000	-	0	1
2	0[2]	0.00000	-	2	1
3	1[1]	0.00000	-	3	1
...	...	...	...	...	...
100	0[2]	0.00492	1(2)	8	0
101	0[1]	0.00000	-	8	1
102	1[2]	0.14111	1(2)	9	0
...	...	...	...	...	...
278	0[1]	0.05659	7(1)	10	0
279	1[2]	0.00021	9(2)	10	0
280	0[2]	0.11371	10(2)	10	0

which consists of 5 classes and D_2 which consists of 2 classes. D_1 has been collected over *June* and *September*, while D_2 has been collected over *March* and *June*. It is important to note that data of each month differ from that of the other month; hence notions like data drift can be studied. Tab. VI summarizes both datasets in terms of number of data points and classes. The classes have been generated by combining the output variables corresponding to 6 actuators: intensity spot light, desk and bed side lamps, window blinds, heater, and PC-based applications comprising word and media playing program. The different combinations of values are considered as classes. The classes reflect some distinct settings.

The goal of the study is to use the data obtained over long periods of time in order to build a classifier capable of learning the user's behavior and actions on the actuators given some environmental conditions that are captured by the sensors.

The parameters T_Σ, T_{merge}, T_{split} and k_0 are again determined online using GA. Table VIII shows an excerpt of the parameter evolution (early stage, middle stage, final stage). The actual values change over time as batches of labeled data are collected. It is worth mentioning that the values shown in Tab. VIII change depending mainly on size of labeled data.

1) Effect of Increasing Labeled Data: In real-world applications, it is important to know how often a labeling feedback might be necessary to help the learning system to self-adjust. Each of the datasets D_1 and D_2 has been split into 2 sets: training and testing. The training set consists of the first 3/4 of month 1, while the testing set consists of the last 1/4 of month 1 and the whole data of month 2. Note that the classifier is trained online. Details of the data split are shown in Tab. VII.

For both datasets, we obtained the accuracy results illustrated in Fig. 16 and the evolution trend of the current error during the training phase shown in Fig. 15. Clearly larger ratios of labeled data provide better prediction results in both cases of D_1 and D_2 after running the algorithm 100 times and averaging the results. On the other hand the evolution of the current error during the training phase seems to be higher when the ratio of labeled data is 10%, but for the other ratios, the evolution is almost stable as the error is small and sometimes is almost 0. It is important to recall that the classifier has not been trained on the second month.

2) Effect of Labeled Data Presentation: Similar to Sec. VII-A, the effect of using two modes of labeled data selection, random and random decreasing, is studied. The data

TABLE VI: Characteristics of the datasets

Data set	Month 1	Month 2	Number of classes
Inhabitant 1	904	561	5
Inhabitant 2	326	995	2

TABLE VII: Split of the iDorm datasets

Data set	Online Training (Month 1)	Testing (Month 1)	Testing (Month 2)
D_1	684	220	561
D_2	246	80	995

TABLE VIII: Parameter evolution for the iDorm datasets

Parameter	Stages		
	Early	Middle	Final
K	20	20	20
α	0.05	0.05	0.05
T_Σ	0.75	0.39	0.85
T_{merge}	27.99	83.15	64.25
T_{split}	3.01	0.41	0.53
k_0	4.44	0.12	0.61

TABLE IX: Effect of the mode of data presentation on the accuracy (30% of data is labeled)

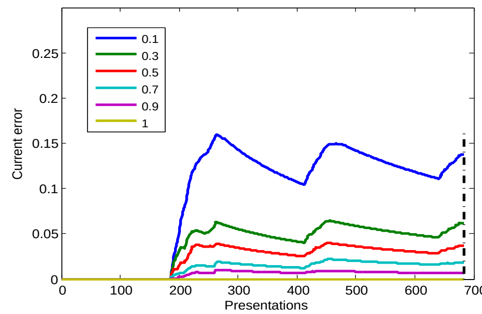
Presentation mode	D_1	D_2
Random	77.59	88.06
Random decreasing	66.21	72.21

TABLE X: Effect of the mode of data presentation on the accuracy (50% of data is labeled)

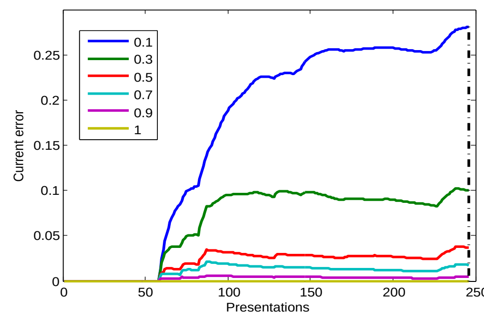
Presentation mode	D_1	D_2
Random	80.98	94.91
Random decreasing	67.76	78.80

set is split according to the scheme described in Sec. VII-B1. Two ratios of labeled data, 30% and 50%, are considered. Figure 17 shows the evolution of the current error for D_1 and D_2 and for both ratios of labeled data. Table IX and Tab. X present the classification results. Clearly the GT2FC algorithm performs better in the case of random mode of labeled data presentation independently of the data set used. Note that the results are averaged over 100 runs.

3) Effect of the Maximum Number of Gaussians: To observe the effect of changing the maximum number of clusters



(a) Case of D_1 using different labeled ratios



(b) Case of D_2 using different labeled ratios

Fig. 15: Evolution of the current error during the training phase for different ratios of labeled data

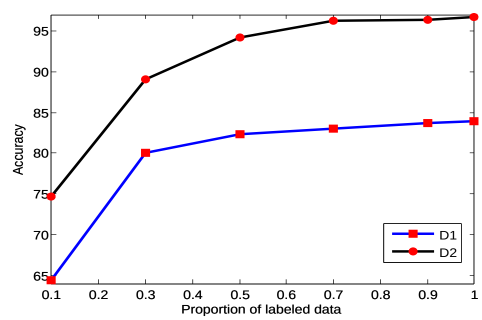


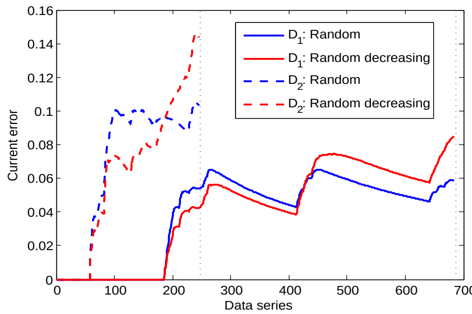
Fig. 16: Effect of increasing the size of labeled data

TABLE XI: Effect of the maximum number of clusters (30% of data is labeled, random selection mode)

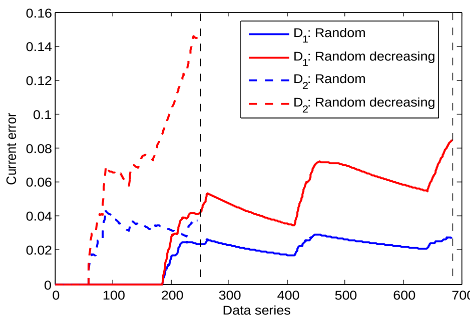
Max. Clusters	D_1	D_2
5	74.31	70.82
10	77.25	88.06
20	78.18	92.98
30	78.91	93.67
40	82.24	95.48
50	81.42	96.04

to be created, we vary the number between 5 and 50 for random selection scenario (Fig. 18). The results are obtained by averaging over 100 runs.

In general, a higher number of Gaussians than the actual number of classes tends to provide better accuracy. Also, the

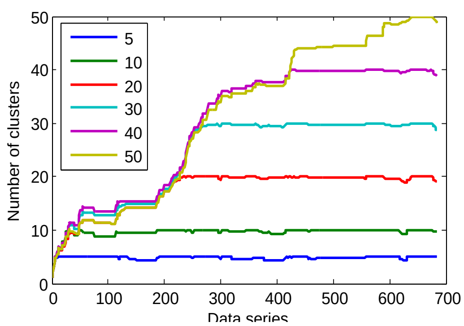


(a) Ratio of labeled data is 30%

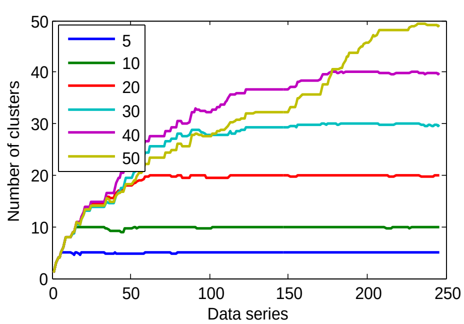


(b) Ratio of labeled data is 50%

Fig. 17: Evolution of the current error during the training phase



(a) Case of D_1



(b) Case of D_2

Fig. 18: Evolution of the number of clusters during training

maximum number of clusters is quickly reached if it set small. Running the algorithm with as many clusters as classes is not efficient enough. Table XI illustrates that better results are obtained with a higher number of clusters. The problem is then the transparency of the rule base which is generated from the clusters. In other terms, increasing the number of clusters

TABLE XII: Feature ordered by importance

	Features							
Batch 1	3	1	8	2	7	5	4	6
Batch 2	1	8	2	3	4	5	6	7
Batch 3	1	2	3	8	7	4	5	6
Batch 4	8	1	2	7	3	4	5	6

TABLE XIII: Features of the iDorm datasets

Feature	Id	Feature	Id
InternalLightLevel	1	ExternalLightLevel	2
InternalTemperature	3	ExternalTemperature	4
ChairPressure	5	BedPressure	6
Occupancy	7	Hour	8

leads to a quite large rule base. The tradeoff is then between the accuracy and the transparency. An optimization stage may be required to remove redundant clusters/rules as explained in Sec. VI-B. Note, however, that the number of clusters generated does not depend only on the maximum (threshold) number of clusters allowed, but also on the thresholds T_Σ, T_{merge}, T_{split}, the learning rate α and the initial spread Σ_0.

4) Rule Generation and Feature Selection: To illustrate the effect of adding new data on the selected features, Tab. XII shows the process of feature selection when using D2 data with 70% labeled data (batch size=60). The features are shown in Tab. XIII. The incremental feature selection uses the new batches to update the relevance order of features. An excerpt of rules generated after the presentation of the last labeled batch is shown in Tab. XIV. After merging the similar fuzzy sets according to the 2nd optimization method (see Sec. VI-B), the new version of the rules are shown in Tab. XV.

C. Data Drift Handling

In dynamically evolving environments, the data distribution may drift over time, leading to performance deterioration of the system, that is, the model built using old data becomes inconsistent with the new data. To tackle this problem, the system needs to be equipped with appropriate mechanisms to handle concept drift.

The current state-of-the-art techniques of concept drift are rather data driven, meaning that drift is handled only from the perspective of data. There are many techniques [7], [46] such as instance selection, instance weighting, and ensemble learning. Although there are several research avenues using data-driven techniques, in this paper we use *model-driven drift* handling techniques [39]. Model-driven drift means using appropriate models that are incremental and able to handle drift. We are interested in models that are capable of handling drift in a systematic way without relying on additional techniques, such as time windowing.

We evaluate the adaptation capability of the algorithm GT2FC to data drift. GT2FC is dedicated to dynamically

TABLE XIV: Fuzzy rules for $D2$

Rule	Antecedent	C
1	x_1 IN N(83.655, [0.964,1.928]) $\wedge$ x_2 IN N(81.115, [0.863,1.726]) $\wedge$ x_3 IN N(22.809, [0.851,1.703]) $\wedge$ x_4 IN N(7.443, [0.853,1.706]) $\wedge$ x_5 IN N(-0.115, [0.863,1.726]) $\wedge$ x_6 IN N(1.115, [0.863,1.726]) $\wedge$ x_7 IN N(1.000, [0.850,1.699]) $\wedge$ x_8 IN N(11.564, [0.544,1.088])	1
2	x_1 IN N(67.000, [0.850,1.699]) $\wedge$ x_2 IN N(76.589, [0.714,1.427]) $\wedge$ x_3 IN N(22.736, [0.850,1.699]) $\wedge$ x_4 IN N(7.500, [0.850,1.699]) $\wedge$ x_5 IN N(0.000, [0.850,1.699]) $\wedge$ x_6 IN N(1.000, [0.850,1.699]) $\wedge$ x_7 IN N(1.000, [0.850,1.699]) $\wedge$ x_8 IN N(11.242, [0.654,1.309])	1
3	x_1 IN N(8.733, [2.425,4.850]) $\wedge$ x_2 IN N(0.683, [0.869,1.738]) $\wedge$ x_3 IN N(23.556, [0.677,1.353]) $\wedge$ x_4 IN N(8.158, [0.706,1.413]) $\wedge$ x_5 IN N(0.000, [0.643,1.286]) $\wedge$ x_6 IN N(1.000, [0.643,1.286]) $\wedge$ x_7 IN N(1.000, [0.643,1.286]) $\wedge$ x_8 IN N(18.645, [1.363,2.726])	2
...	...	...
18	x_1 IN N(13.000, [1.143,2.286]) $\wedge$ x_2 IN N(82.000, [1.143,2.286]) $\wedge$ x_3 IN N(24.400, [1.143,2.286]) $\wedge$ x_4 IN N(8.500, [1.143,2.286]) $\wedge$ x_5 IN N(0.000, [1.143,2.286]) $\wedge$ x_6 IN N(1.000, [1.143,2.286]) $\wedge$ x_7 IN N(1.000, [1.143,2.286]) $\wedge$ x_8 IN N(13.112, [0.511,1.022])	2
19	x_1 IN N(0.000, [1.143,2.286]) $\wedge$ x_2 IN N(1.000, [1.143,2.286]) $\wedge$ x_3 IN N(23.260, [1.143,2.286]) $\wedge$ x_4 IN N(8.500, [1.143,2.286]) $\wedge$ x_5 IN N(0.000, [1.143,2.286]) $\wedge$ x_6 IN N(0.000, [1.143,2.286]) $\wedge$ x_7 IN N(1.000, [1.143,2.286]) $\wedge$ x_8 IN N(19.896, [0.511,1.022])	2
20	x_1 IN N(6.000, [0.303,0.606]) $\wedge$ x_2 IN N(0.745, [0.520,1.039]) $\wedge$ x_3 IN N(23.587, [0.323,0.647]) $\wedge$ x_4 IN N(8.391, [0.443,0.886]) $\wedge$ x_5 IN N(0.000, [0.303,0.606]) $\wedge$ x_6 IN N(1.000, [0.303,0.606]) $\wedge$ x_7 IN N(1.000, [0.303,0.606]) $\wedge$ x_8 IN N(19.194, [0.438,0.876])	2

TABLE XV: Fuzzy rules for $D2$ after merge of similar fuzzy sets

Rule	Antecedent	C
1	x_1 IN N(83.655, [0.964,1.928]) $\wedge$ x_2 IN N(81.115, [0.863,1.726]) $\wedge$ x_3 IN N(22.772, [0.601,1.203] $\wedge$ x_4 IN N(7.472, [0.602,1.204]) $\wedge$ x_8 IN N(11.432, [0.418,0.836])	1
2	x_1 IN N(67.000, [0.850,1.699]) $\wedge$ x_2 IN N(76.589, [0.714,1.427]) $\wedge$ x_3 IN N(22.772, [0.601,1.203] $\wedge$ x_4 IN N(7.472, [0.602,1.204]) $\wedge$ x_8 IN N(11.432, [0.418,0.836])	1
3	x_1 IN N(8.733, [2.425,4.850]) $\wedge$ x_2 IN N(0.729, [0.446,0.892]) $\wedge$ x_3 IN N(23.562, [0.282,0.565]) $\wedge$ x_4 IN N(8.356, [0.340,0.681]) $\wedge$ x_8 IN N(19.444, [0.323,0.646])	2
...	...	...
18	x_1 IN N(13.000, [1.143,2.286]) $\wedge$ x_2 IN N(82.000, [1.143,2.286]) $\wedge$ x_3 IN N(24.400, [1.143,2.286]) $\wedge$ x_4 IN N(8.356, [0.340,0.681]) $\wedge$ x_8 IN N(13.112, [0.511,1.022])	2
19	x_1 IN N(0.000, [1.143,2.286]) $\wedge$ x_2 IN N(1.000, [1.143,2.286]) $\wedge$ x_3 IN N(23.562, [0.282,0.565]) $\wedge$ x_4 IN N(8.356, [0.340,0.681]) $\wedge$ x_8 IN N(19.444, [0.323,0.646])	2
20	x_1 IN N(6.000, [0.303,0.606]) $\wedge$ x_2 IN N(0.729, [0.446,0.892]) $\wedge$ x_3 IN N(23.562, [0.282,0.565]) $\wedge$ x_4 IN N(8.356, [0.340,0.681]) $\wedge$ x_8 IN N(19.444, [0.323,0.646])	2

incremental settings requiring to self-adjust over time so that a continuous performance is ensured. The sensitivity of GT2FC to drift is tested using the iDorm data in two ways:

1) Artificial Drift: To generate drifting data from the real world iDorm dataset, we rely on *feature selection* as a method to determine the feature which is the highly correlated with the classes measured by a partial F-test. Once such a feature has been found, the data set is sorted according to that feature. Then, the indicated feature is discarded from the data. The drift is generated, while its origin is hidden (the drift origin is often unknown in real world situations).

The feature which is most correlated with the output of the iDorm dataset is "internal light level" (first feature). We sort the data according to this feature, then we discard it. Figures 19 and 20 show the evolution of the error using the whole data sets D_1 and D_2. Clearly the algorithm GT2FC is capable of self-adjusting in response to drift as the error rate decreases. Note that in this experiment different numbers of clusters have been tried, the ratio of labeled data has been set to 30% randomly presented, and the results have been averaged over 100 runs.

2) Natural Drift over Two Seasons: As explained in Sec. VII-B, the first iDorm data set D_1 concerns two months *June* and *September*, while D_2 concerns *March* and *June*. The data of each month differ from that of the other month for both datasets. The two pairs of months refer to two pairs of seasons in UK: (Summer vs. Autumn) and (Spring vs. Summer) respectively. Hence, drift is natural in the data, but sofar in the experiments the data has been split into 2 sets: training and testing, where the training set consists of the first 3/4 of month 1 and the testing set consists of the last 1/4 of month 1 and the whole month 2. In the present experiment, the whole data will be used as a stream to observe how the current error evolves over time.

Figures 21 and 22 show the current error over the two seasons for different numbers of clusters. The current error starts to increase in the vicinity of the season change point. While for the first data set D_1 the error increases slightly (but remains small), for the second data set D_2 the error decreases as new data arrive.

VIII. Type-2 vs. Type 1 and Other Incremental Learning Systems

To highlight the suitability of the proposed incremental type-2 FS, GT2FC to the ambient intelligence application at hand, we compare it against the type-1 version of GT2FC, denoted in the following as GT1FC, and against other incremental learning algorithms namely Incremental Fuzzy Rule-based Classification System (IFCS) [6] and the Nearest Generalized Exemplar (NGE) [40].

Tables XVI and XVII show the results for different ratios of labeled data. Note that IFCS and NGE are trained using only the labeled data, while GT1FC and GT2FC are trained using both labeled and unlabeled data. The training and the testing data are obtained according to the split explained in Sec. VII-B. Judged by the standard deviation, clearly GT1FC and GT2FC are more efficient and stable as the labeled data size increases compared to IFCS and NGE. Moreover, GT2FC outperforms GT1FC on average especially in the case of D2 (Tab. XVII) although the difference is not significantly high.

For 10% labeled data, the performance of GT2FC and GF1FC is below that of the other algorithms. The reason is that the other algorithms are trained on only labeled data and therefore their prediction power is higher than that of GT2FC which uses labeled and unlabeled data. We notice that the boundary may be between 10% and 30% included, where the

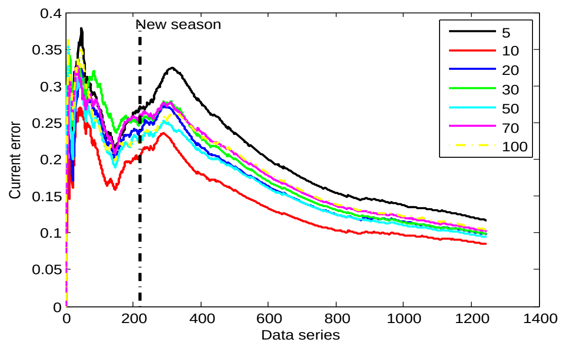


Fig. 19: Artificial drift based on feature selection (D_1)

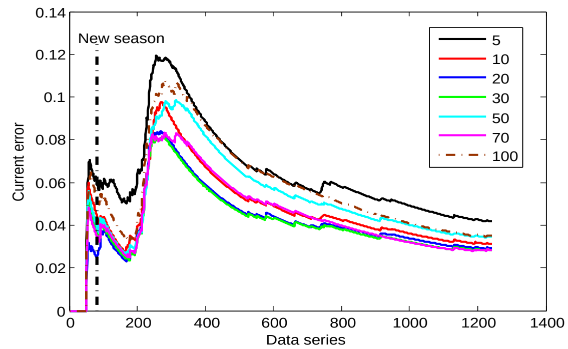


Fig. 20: Artificial drift based on feature selection (D_2)

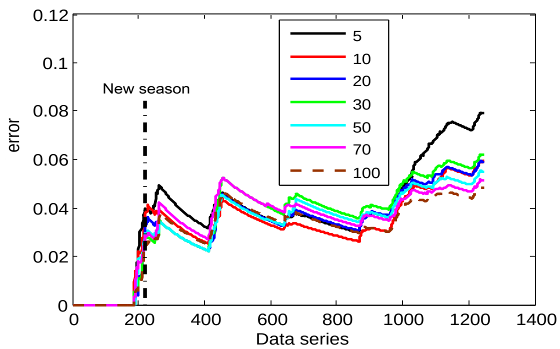


Fig. 21: Natural drift - season change (D_1)

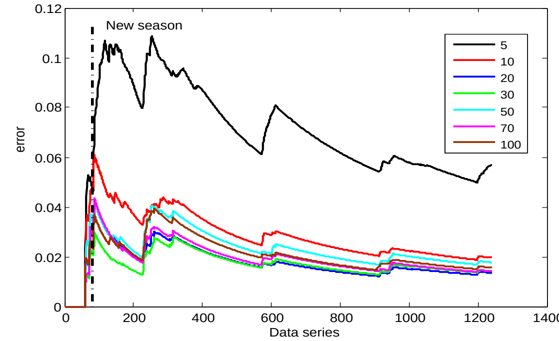


Fig. 22: Natural drift - season change (D_2)

rise occurs, although this observation may not always be true (e.g., in Tab. XVI, even with 30%, GT2FC does not perform as well as NGE). To formally check the observation, we run further experiments for the percentages 15%, 20%, and 25% of labelled data which are in the interval [10%, 30%].

The outcome of the new experiments on D1 is $61.1230 \pm 0.2900, 68.4492 \pm 0.180$ and 73.1105 ± 0.1391 and on D2 it is 62.2714 ± 0.3632, 74.1809 ± 0.3170 and 83.5477 ± 0.2428 respectively. Comparing the values in Tab. XVI and Tab. XVII and these new results, the boundary seems rather to be around 30%. The major conclusion is, however, that the accuracy of GT2FC is higher for large ratios of labeled data and increases as such ratio increases.

From a computational time perspective, Tab. XVIII compares the computational time of the 4 algorithms (IFCS, NGE, GT1FC, GT2FC) for one run on the same data (D2) for different ratios of labeled data (10%, 50%, 100%). The results indicate that GT2FC requires more time compared with the other algorithms, especially when dealing with more labeled data. This, however, is expected due to the steps involved in GT2FC which includes many checks related to labels of the data. In particular, IFCS and NGE use only the labeled data.

IX. Online vs. Batch Learning

An online learner receives a new data point x_t along with the current hypothesis H_{t-1}, checks if the data point is covered by the current hypothesis and updates the hypothesis accordingly. Ideally the online learner performs as well as its corresponding offline version, where it can see the data for many epochs. If this happens we say that the algorithm is lossless. In this experiment, we compare the performance accuracy of the proposed online GT2FC against its offline version. Table XIX shows that when the ratio of labeled data is below 30%, the offline mode performs relatively better, but when the labeled data is above 30%, the online version performs better. The results of the online and the offline versions are close. Therefore GT2FC can potentially perform as well as its offline version.

TABLE XVIII: Computational time

Algorithm	Ratio of labeled data		
	0.1	0.5	1
IFCS	0.91	3.09	8.15
NGE	0.18	0.20	0.23
GT1FC	11.05	14.37	14.54
GT2FC	14.50	18.66	19.52

X. Conclusion

In this paper, we proposed a growing interval type-2 fuzzy classifier, called GT2FC, which is equipped with various mechanisms for online learning, quality control and data drift management. The aim is to deal with data whose distribution may change dynamically over time. To generate the type-2 fuzzy membership functions associated with the features in the rules' premises, we proposed a new semi-supervised online learning algorithm, called 2G2M. This algorithm is designed to operate online and to learn from both labeled

TABLE XVI: GT2FC vs. other online classifiers: Case $D1$

Algorithm	Ratio of labeled data in the training data					
	0.1	0.3	0.5	0.7	0.9	1
IFCS	76.14±2.38	75.33±1.12	75.49±0.68	75.27±0.27	75.24±0.27	75.22±0.03
NGE	82.27±3.62	83.83±2.81	79.63±5.22	75.54±3.43	74.31±0.05	74.33±0.05
GT1FC	55.43±0.23	75.36±0.15	80.62±0.02	81.42±0.01	82.88±0.01	83.13±0.01
GT2FC	60.28±0.24	79.55±0.02	80.73±0.01	81.65±0.01	82.88±0.01	83.76±0.01

TABLE XVII: GT2FC vs. other online classifiers: Case $D2$

Algorithm	Ratio of labeled data in the training data					
	0.1	0.3	0.5	0.7	0.9	1
IFCS	91.17±5.03	88.39±3.28	86.64±1.16	86.25±0.42	86.12±0.27	85.92±0.10
NGE	88.08±2.34	87.37±0.10	87.40±0.04	87.42±0.02	87.43±0.01	87.43±0.03
GT1FC	59.12±0.30	89.71±0.13	93.68±0.05	96.20±0.01	96.68±0.01	97.01±0.01
GT2FC	60.19±0.34	89.44±0.10	95.50±0.04	96.89±0.01	96.78±0.01	97.00±0.01

TABLE XIX: Online vs. batch ($D2$)

Algorithm	Ratio of labeled data in the training data					
	0.1	0.3	0.5	0.7	0.9	1
GT2FC (Online)	60.19±0.34	89.44±0.10	95.50±0.04	96.89±0.01	96.78±0.01	97.00±0.01
GT2FC (Offline)	70.05±0.31	93.23±0.04	94.86±0.03	96.20±0.01	96.46±0.01	96.68±0.01

and unlabeled data. The 2G2M parameters are optimized continuously and regularly using batches of labeled data. To maintain compactness of the rules, the GT2FC classifier uses an online feature selection algorithm, while 2G2M optimizes the clusters online. A broad range of experiments using different datasets (artificial, classic and real-world) were conducted. These experiments provided a picture on the performance of GT2FC and 2G2M algorithms under various settings. The major outcome is that the type-2 classifier performs very well.

Moreover, the approach presented in this paper is in the first place illustrative to show how type-2 fuzzy systems can be developed for open-ended dynamic environments to accommodate continuous learning over time.

In the future, we will extend the present work to (i) growing *generalized* type-2 fuzzy classification systems and (ii) growing type-2 fuzzy classification systems where Gaussian membership functions are characterized not only by uncertain deviation as presented here, but also by uncertain mean.

Acknowledgment

The authors wish to thank very much Hani Hagras for kindly providing the iDorm data.

References

[1] P. Angelov. An approach for fuzzy rule-base adaptation using on-line clustering. *International Journal of Approximate Reasoning*, 35(3):275–289, 2004.

[2] P. Angelov and X. Zhou. Evolving fuzzy-rule-based classifiers from data streams. *IEEE Tran. Fuzzy Syst.*, 16(6):1462 –1475, 2008.

[3] O. Arandjelovic and R. Cipolla. Incremental learning of temporally coherent Gaussian mixture models. In *Proceedings of the The 16th British Machine Vision Conference*, pages 759–768, 2005.

[4] A. Bilgin, J. Dooley, L. Whittington, H. Hagras, M. Henson, C. Wagner, A. Malibari, A. Al-Ghamdi, M.J. Alhaddad, and D. Alghazzawi. Dynamic profile-selection for zslices based type-2 fuzzy agents controlling multi-user ambient intelligent environments. In *IEEE International Conference on Fuzzy Systems*, pages 1–8, 2012.

[5] A. Bouchachia. An evolving classification cascade with self-learning. *Evolving Systems*, 1:143–160, 2010.

[6] A. Bouchachia. Fuzzy classification in dynamic environments. *Soft Comput.*, 15(5):1009–1022, 2011.

[7] A. Bouchachia. Incremental learning with multi-level adaptation. *Neurocomputing*, 74(11):1785–1799, 2011.

[8] A. Bouchachia and R. Mittermeir. Towards fuzzy incremental classifiers. *Soft Computing*, 11(2):193–207, 2006.

[9] G. Castellano, A. Fanelli, and C. Mencar. *Advances in Fuzzy Clustering and its Applications*, chapter Mining Diagnostic Rules Using Fuzzy Clustering, pages 211–226. Wiley, 2007.

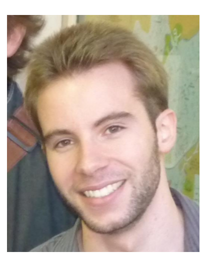
[10] O. Castillo and M. Patricia. *Type-2 Fuzzy Logic: Theory and Applications*. Springer-Verlag, series: studies in fuzziness and soft computing, vol. 223 edition, 2008.

[11] S. Coupland, J. Wheeler, and M. Gongora. A generalised type-2 fuzzy logic system embedded board and integrated development environment. In *Proceedings of FUZZ-IEEE'08*, pages 681–687, 2008.

[12] R. De, N. Pal, and S. Pal. Feature analysis: Neural network and fuzzy set theoretic approaches. *Pattern Recognition*, 30(10):1579–1590, 1997.
[13] J. de Barros and L. Dexter. On-line identification of computationally undemanding evolving fuzzy models. *Fuzzy Sets and Systems*, 158(16):1997–2012, 2007.
[14] A. Declercq and J. Piater. Online learning of Gaussian mixture models - a two-level approach. In *Proceedings of the Third International Conference on Computer Vision Theory and Applications*, pages 605–611, 2008.
[15] A. Dempster, M. Laird, and D. Rubin. Maximum likelihood for incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B 39*, pages 1–38, 1977.
[16] F. Doctor, H. Hagras, and V. Callaghan. A type-2 fuzzy embedded agent to realise ambient intelligence in ubiquitous computing environments. *Information Sciences*, 171(4):309–334, 2005.
[17] A. Frank and A. Asuncion. UCI machine learning repository, 2010.
[18] M. Gacto, Alcalá, R., and F. Herrera. Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Inf. Sci.*, 181(20):4340–4360, 2011.
[19] R. Gross, J. Yang, and A. Waibel. Growing Gaussian mixture models for pose invariant face recognition. In *Proceedings of the International Conference on Pattern Recognition*, pages 1088–1091. IEEE Computer Society, 2000.
[20] H. Hagras, F. Doctor, A. Lopez, and V. Callaghan. An incremental adaptive life long learning approach for type-2 fuzzy embedded agents in ambient intelligent environments. *IEEE Transactions on Fuzzy Systems*, 15(1):41–55, 2007.
[21] P. Hall and Y. Hicks. A method to add Gaussian mixture models. Technical report, University of Bath, 2004.
[22] J. Jang, C. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, 1996.
[23] C-F Juang and Y-W. Tsao. A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning. *IEEE Transactions on Fuzzy Systems*, 16(6):1411–1424, 2008.
[24] N. Karnik and J. Mendel. Operations on type-2 fuzzy sets. *Fuzzy Sets and Systems*, 122(2):327–348, 2001.
[25] A. Klose and J. Schneider. Semi-supervised induction of fuzzy rules applied to image segmentation. In *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, volume 3, pages 1425–1430, 2001.
[26] A. Krone and T. Slawinski. Data-based extraction of unidimensional fuzzy sets for fuzzy rule generation. In *IEEE World Congress on Computational Intelligence - The 1998 IEEE International Conference on Fuzzy Systems Proceedings*, pages 1032–1037, 1998.
[27] D. Lee. Effective Gaussian mixture learning for video background subtraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):827–832, 2005.
[28] C. Leondes. *Fuzzy theory systems: techniques and applications*. Systems Series. Academic, 1999.
[29] E. Lughofer. *Evolving Fuzzy Systems - Methodologies, Advanced Concepts and Applications*. Studies in Fuzziness and Soft Computing. Springer, 2011.
[30] U. Mahmood, A. Al-Jumaily, and M. Al-Jaafreh. Type-2 fuzzy classification of blood pressure parameters. In *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, pages 595–600, 2007.
[31] C. Mencar, G. Castellano, and A. Fanelli. Distinguishability quantification of fuzzy sets. *Inf. Sci.*, 177(1):130–149, 2007.
[32] A. Mencattini, M. Salmeri, and R. Lojacono. Estimation of uncertainty in measurement by means of type-2 fuzzy variables. In *FUZZ-IEEE'07*, pages 1–6, 2007.
[33] J. Mendel. Uncertainty, fuzzy logic, and signal processing. *Signal Process.*, 80:913–933, 2000.
[34] J. Mendel. Type-2 fuzzy sets and systems. *IEEE Computational Intelligence Magazine*, 2(1):20–29, 2007.
[35] J.M. Mendel. On km algorithms for solving type-2 fuzzy set problems. *IEEE Transactions on Fuzzy Systems*, 21(3):426–446, 2013.
[36] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368, 1999.
[37] N. Radford and G. Hinton. *Learning in graphical models*, chapter A view of the EM algorithm that justifies incremental, sparse, and other variants, pages 355–368. MIT Press, 1999.
[38] S. Roberts, C. Holmes, and D. Denison. Minimum-entropy data partitioning using reversible jump Markov chain monte carlo. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(8):909–914, 2001.
[39] Z. Sahel, A. Bouchachia, and B. Gabrys. Adaptive mechanisms for classification problems with drifting data. In *Proc. of the 11th Int. Conf. on Knowledge-Based Intelligent Information and Engineering Systems (KES'07), LNCS 4693*, pages 419–426, 2007.
[40] S. Salzberg. A nearest hyperrectangle learning method. *Machine learning*, 6:277–309, 1991.
[41] M. Sato and S. Ishii. On-line EM algorithm for the normalized Gaussian network. *Neural Comput.*, 12(2):407–432, 2000.
[42] D. Schnitzer, Flexer A., Widmer G., and Gasser M. Islands of Gaussians: The self organizing map and Gaussian music similarity features. *Proceedings of the 11th International Society for Music Information Retrieval Conference*, 2010.
[43] M. Song and H Wang. Highly efficient incremental estimation of Gaussian mixture models for online data stream clustering. In *Intelligent Computing: Theory and Applications*, pages 174–183. SPIE, 2005.
[44] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:246–252, 1999.
[45] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):747–757, 2000.
[46] A. Tsymbal. The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15, Dept. of Computer Science Trinity College, Dublin, 2004.
[47] C. Wagner and H. Hagras. An approach for the generation and adaptation of zslices based general type-2 fuzzy sets from interval type-2 fuzzy sets to model agreement with application to intelligent environments. In *FUZZ-IEEE*, pages 1–8, 2010.
[48] G-D Wu and P-H Huang. A vectorization-optimization-method-based type-2 fuzzy neural network for noisy data classification. *IEEE Transactions on Fuzzy Systems*, 21(1):1–15, 2013.
[49] Z. Yang and M. Zwolinski. Mutual information theory for adaptive mixture models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(4):396–403, 2001.
[50] D. Zhai and J. Mendel. Computing the centroid of a general type-2 fuzzy set by means of the centroid-flow algorithm. *IEEE Transactions on Fuzzy Systems*, 19(3):401–422, 2011.

Abdelhamid Bouchachia (M'06-SM'13) received the Engineering degree and Master in Informatics, respectively from the University of Oran and the University of Sciences and Technology, Oran, Algeria, and the Ph.D. degree in Informatics from the University of Klagenfurt, Austria. In 2001 he spent one year as post-doc at the University of Alberta, Department of Computer and Electrical Engineering, Edmonton, Canada.

He is currently Associate Professor at Bournemouth University, School of Design, Engineering and Computing, UK. His major research interests include Machine Learning and Soft computing with a particular focus on online/incremental learning, semi-supervised learning, prediction systems, and uncertainty modeling. He is the general chair of the International Conference on Adaptive and Intelligent Systems (ICAIS). He serves as program committee member for many conferences. He also serves as Associate Editor of Evolving Systems and acts as member of Evolving Intelligent Systems (EIS) Technical Committee (TC) of the IEEE Systems, Man and Cybernetics Society, the IEEE Task-Force for Adaptive and Evolving Fuzzy Systems and the IEEE Computational Intelligence Society.



Charlie.Vanaret received his engineering degree in Computer Science and Applied Mathematics from the Institut National Polytechnique (INP), Toulouse, France in 2011 after completing an internship at the University of Klagenfurt, Austria. He is currently a PhD candidate at École Nationale de l'Aviation Civile (ENAC), laboratoire MAIAA, Toulouse, France. His research interests cover computational intelligence and global optimisation.