ParaFIS:A new online fuzzy inference system based on parallel drift anticipation

Clement Leroy, Eric Anquetil, Nathalie Girard

ParaFIS:A new online fuzzy inference system based on parallel drift anticipation

Clement Leroy
Intuidoc Team
Univ Rennes, CNRS, IRISA
Rennes, France
clement.leroy@irisa.fr

Eric Anquetil
Intuidoc Team
Univ Rennes, CNRS, IRISA
Rennes, France
eric.anquetil@irisa.fr

Nathalie Girard
Intuidoc Team
Univ Rennes, CNRS, IRISA
Rennes, France
nathalie.girard@irisa.fr

***Abstract*—This paper proposes a new architecture of incremental fuzzy inference system (also called Evolving Fuzzy System - EFS). In the context of classifying data stream in non stationary environment, concept drifts problems must be addressed. Several studies have shown that EFS can deal with such environment thanks to their high structural flexibility. These EFS perform well with smooth drift (or incremental drift). The new architecture we propose is focused on improving the processing of brutal changes in the data distribution (often called brutal concept drift). More precisely, a generalized EFS is paired with a module of anticipation to improve the adaptation of new rules after a brutal drift. The proposed architecture is evaluated on three datasets from UCI repository where artificial brutal drifts have been applied. A fit model is also proposed to get a "reactivity time" needed to converge to the steady-state and the score at end. Both characteristics are compared between the same system with and without anticipation and with a similar EFS from state-of-the-art. The experiments demonstrates improvements in both cases.**

***Index Terms*—Evolving Fuzzy System EFS, Non Stationary Environment NSE,brutal drift adaptation, long-life learning.**

I. Introduction

More and more data are generated by stream over a long period leading to the necessity for classification methods to include long-life learning in their algorithm to stay reliable in time. As an example, a command-gesture system [1], where the user draws a gesture to apply a command, produces this kind of data stream. The set of gestures is chosen by the user and can change at any time. For example a novice user often draw slowly a gesture, and once he is used to make it, he can speed up the gesture leading to a shift of the target concept in the feature space. This modification is called *concept drift*. Such environment is said to be non stationary. It means that the probability distribution that generates data (here the user) depends on time. To maintain high performance in gestures recognition over time, the classification system must adapt its parameters and/or structure to the concept drift. Thus, the learning process has to be *incremental*. For each new data, the system extends its knowledge (*i.e.*, it learns and adapts its parameters). Moreover, to get a constant complexity in time and avoid a memory overflow, along a long-life learning, data should not be saved.

Recent researches have shown that Evolving Fuzzy Systems - EFS (also called Incremental Fuzzy Inference System) could deal with such environment thanks to their high structural flexibility [2], [3]. The structure (number of rules, and antecedents/consequent parameters) of these fuzzy rule-based systems evolve with the stream of data. Especially, we distinguish between two scales of adaptation to concept drift. One concerns the adaptation of the rules' parameters and tackles incremental drifts (or smooth drifts). The other relates to the adaptation of the structure, by the addition or deletion of rules, and tackles brutal drifts (brutal shifts of the data distribution). Both adaptions use two main algorithmic approaches. The first one relies on temporal (adaptive) sliding windows where only the most recent data are taking into account. As an example, the fuzzy windows concept drift adaptation method (FW-DA) [4] relies on the method and shows promising results outperforming state-of-the-art. The second one weights the data according to their age and interest. An example is [5] which uses a decremental learning on the premise and conclusion parts with the Differed Directional Forgetting (DDF). However, both approaches have to figure out a "forgetting" parameter which controls the relevance of the old data. If the forgetting parameter is too high, the system will be less stable leading to lower performance. If the parameter is too low, the system will not be reactive to the change in the environment leading to bad performance. This issue is often called *the plasticity-stability dilemma*. This dilemma can result in instability, particularly with EFS where the number of rules, their size and their reactivity greatly depend on parameters given *a priori*. To prevent instability, [2] recently proposed a new rule splitting method. The idea is to split into two, a rule which makes too many mistakes or which is too large. However, it takes time after the splitting for the rules to re-adjust to the local distribution of points, introducing inertia in the learning. On the contrary, [6] proposed a method to accelerate the learning of new rules after a drift based on the generation of data from new drifted concept using a GAN [7]. Similarly, we propose a complementary method to prevent instability of the rules while improving reactivity. To do so, we introduce a new original architecture of EFS, called *ParaFIS*. In ParaFIS, a generalized evolving fuzzy system (principal system) is learned synchronously with *an anticipation module*. For each rule of the principal system, two rules are anticipated in the anticipation module. Thus, the anticipation module enforces the system to locally adapt the distribution of points with two rules rather than just one, in order to anticipate a concept drift before it occurs. The paper is organized as follows. Section II introduces the generalized EFS and its learning model with a discussion on its drawbacks. Section III introduces our contribution, the *ParaFIS* evolving system. The experiments showing that the anticipation improves the plasticity of the system while keeping stability, are detailed in section IV. In this section, we propose to measure performance of the system on artificial brutal drifts with fitted parameters of a handcraft model. This evaluation protocol allows to quantify the time of reactivity of the system and its stability in the steady-state.

II. Generalized Evolving Fuzzy Systems

In this paper, we focus on the generalized evolving fuzzy system that uses a generalized version of Takagy-Sugeno fuzzy systems, already used [2], [8], [9].

A. Model Architecture

A Takagy-Sugeno (TS) fuzzy system is a set of fuzzy inference rules $R = \{r_i, 0 \leq i \leq N\}$ with an antecedent part (also called premise), and a consequent part. Each rule's antecedent is defined with a prototype that is set by a cluster with a center μ_i. The structure of a rule r_i, is as follows:

$$\textbf{IF} \;\; \mathbf{x} \text{ is close to } \mu_i \;\; \textbf{THEN} \;\; y_i^1 = l_i^1(\mathbf{x}) \;..\; y_i^c = l_i^c(\mathbf{x}) \quad (1)$$

With l_i^j a polynomial function for r_i of class j; c the number of class and N the number of rules. The degree of the polynomial function is set to 1 with π_{ik}^j the polynomial coefficients (see Eq. (2)).

$$y_i^j = l_i^j(\mathbf{x}) = \pi_{i0}^j + \pi_{i1}^j x_1 + \pi_{i2}^j x_2 + .. + \pi_{in}^j x_n = \Pi_i^j \mathbf{x} \quad (2)$$

The membership of $\mathbf{x}$ to a rule r_i, denoted $\beta_i(\mathbf{x})$, is given by a normalized Radial Basis Function K (RBF), of the distance from x to μ_i (see Eq.(3)). The RBF is often a multivariate Gaussian or Cauchy function [8], [9].

$$\beta_i(\mathbf{x}) = K(||\mathbf{x} - \mu_\mathbf{1}||^2) \quad (3)$$

In the generalized version of TS, the Mahalanobis distance is used to get rotated hyper-ellipsoid clusters as follows:

$$||\mathbf{x} - \mu_\mathbf{i}||^2 = (\mathbf{x} - \mu_i)A^{-1}(\mathbf{x} - \mu_i)^T \quad (4)$$

With A the covariance matrix. Finally, the predicted class for $\mathbf{x}$ is given by Eq. (5),(6).

$$class(\mathbf{x}) = y = argmax_j \; y^j(\mathbf{x}) \quad (5)$$

$$\text{Where } y^j(\mathbf{x}) = \sum_{i=1}^{N} \beta_i(\mathbf{x}) y_i^j \quad (6)$$

B. Rule's adaptation

Each new incoming data $\mathbf{x}_t$ is used to adapt the model parameters. In the premise part, only the most activated rule adapts its center and covariance matrix according to Eq. (7),(8) where t is the number of samples of the rule.

$$\mu_t = \frac{t-1}{t}\mu_{t-1} + \frac{1}{t}\mathbf{x}_t \quad (7)$$

$$\mathbf{A}_t = \frac{t-1}{t}\mathbf{A}_{t-1} + \frac{1}{t}(\mathbf{x}_t - \mu_t)(\mathbf{x}_t - \mu_t)^T \quad (8)$$

The forgetting capacity is put in the equation, by setting $t = min(k, tmax)$ (see [5]) with $tmax$ a threshold that defined the forgetting capacity, and k the number of samples that activated the most the rules. $tmax$ is often written with a forgetting factor $\alpha = \frac{tmax-1}{tmax} \in [0, 1]$ where $\alpha = 1$ when there is no forgetting capacity.
The consequent part is learned using a Weighted Recursive Least Square method (WRLS). The membership functions β is assumed to be almost constant to converge to the optimal solution. To reduce computation time, the local learning of the consequent part is often preferred. And, the rules are assumed to be independent to apply RLS on each one. The conclusion matrix $\Pi_{i(t)} = [\Pi_{i(t)}^1, .., \Pi_{i(t)}^c]$ of the rule r_i at time t (*i.e.* after t data points) is recursively computed according to:

$$\Pi_{i(t)} = \Pi_{i(t-1)} + C_{i(t)}\beta_i(\mathbf{x})C_i\mathbf{x}(Y_t - \mathbf{x}\Pi_{i(t-1)}) \quad (9)$$

$$\text{Where } C_{i(t)} = C_{i(t-1)} - \frac{\beta_i(\mathbf{x})C_{i(t-1)}\mathbf{x}\mathbf{x}^T C_i}{1 + \beta_i(\mathbf{x})\mathbf{x}^T C_i \mathbf{x}} \quad (10)$$

With C_i a correlation matrix initialized by $C_{i(t=0)} = \Omega Id$ where Id is the identity matrix and Ω a constant often fixed to 100 (see [8], [10]).

C. Rule creation condition

The adaptation of the parameters is not relevant to handle brutal drifts, as when a class must be represented by several clusters or when the target concept shift in the feature space. Dealing with brutal drifts requires the adaptation of the fuzzy system structure, like rule addition. In a context of online learning from scratch, all classes start with one prototype (*i.e.* one rule). The structure adaptation relies on specific conditions observed on data, via the existing rules. Several criteria are defined to detect brutal drifts. Most EFS uses a distance-based criteria [2], [8] that compares a certain threshold $T(P)$ (depending on parameters P) with the distance between the prototype of the closest rule (its center μ) and the new incoming points $\mathbf{x}_t$. If $T(P) < dist(\mu - \mathbf{x_t})$, then the rule creation criteria is met, a new rule is created over the last incoming point according to Eq. (11).

$$\begin{aligned} \mu_{new} &= \mathbf{x_t} & \Pi_{new} &= \mathbf{0} \\ cov_{kl} &= \frac{1}{100}\delta_{kl} & \forall k, l &\in [1, n] \end{aligned} \quad (11)$$

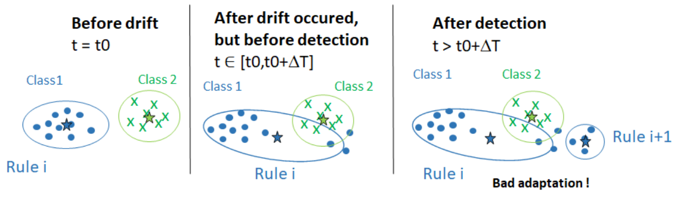

Fig. 1: Problems met in generalized EFS when brutal drift occurs. The antecedent structure of each rule is represented by the ellipse (the covariance matrix) and the star (the center)

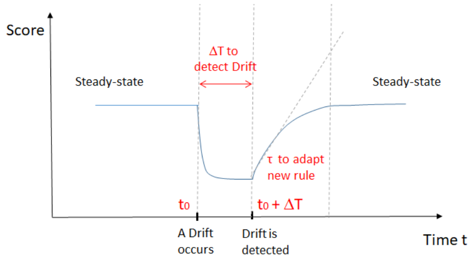

Fig. 2: Impact of a drift on the score in time

D. Discussion on problems in the generalized EFS

Two scales of adaptation co-exist in EFS, the adaptation of rule parameters and the structure adaptation. Smooth drifts are tackled by introducing forgetting capacity in the parameter adaptation whereas brutal drifts are tackled by the creation of new rules. However, the system is degraded by the parameter adaptation when a brutal drift occurs. Indeed, all rule creation conditions lead to a trade-off between speed of detection and sensitivity to noise. But, for all, it exists a time $\Delta T > 0$ between the true occurrence of the drift, and the detection time. As shown in Figure 1 and 2, during this ΔT time, one (ore more) rule tries to adapt to the drift and changes its parameters making it less fit to its previous concept. But, when the rule creation is triggered, this previous adaptation is not cancelled making the old rule perhaps unstable. Moreover, the new rule is created over one single point (the one that triggers the rule creation) although all points during ΔT could have been used to initialize the new rule. This results in a longer time τ for the new rule adaptation to the new concept.

III. Contribution: ParaFIS system

In order to attend stability and plasticity, we present in this section the ParaFIS system.

A. Model's architecture

As shown in Figure 3, the proposed model is based on the generalized EFS similar to this described section II. This part is dedicated to smooth drifts to keep stability (no structural change). And, for each rule r_i, a module of anticipation is added to deal with brutal drifts. This anticipation module is composed of two sub-rules r_{i_1} and r_{i_2} that have an antecedent part (a center, a covariance matrix, a Cauchy membership function) and a consequent part with c hyperplanes (c the number of classes). The system classifies the data at any time

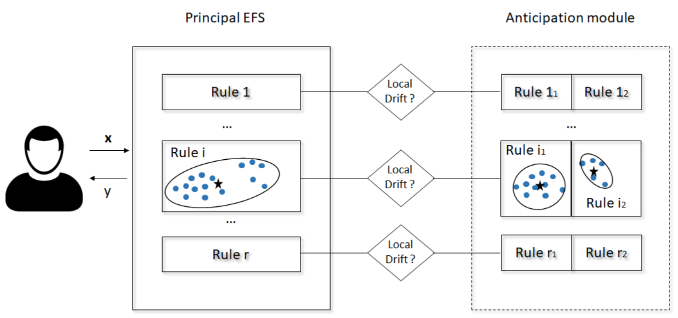

Fig. 3: Architecture of our proposition ParaFIS

independently of the anticipation module as it is done in the generalized EFS. The sub-rules are just used in the learning phase where different forgetting factors are applied to adapt differently the distribution of points in time. Thus, the system gets information at different scales of time.

B. Model's learning

Each new point coming into the system will be used to learn both the principal system and the anticipation module. As in the generalized EFS, the principal system will adapt the antecedent part of the most activated rule r_m by updating its center and its covariance matrix using Eq. (7),(8) with a factor $\alpha = 1$ (no forgetting capacity). Then, the sub-rules r_{m_1} and r_{m_2} are updated using the same equations with $\alpha 1, \alpha 2 \leq 1$. The two sub-rules have two different forgetting factors leading to two temporal scales of learning. r_{m_1} has a low forgetting factor and is learned on the most recent data whereas r_{m_2} has a high forgetting factor and is learned on a long history. In this way, r_{m_1} quickly reacts to a change in the distribution of points whereas r_{m_2} preserves the old concept with a slow adaptation.
The consequent part in the principal system is learned as usual (Eq. (9),(10)). In the anticipation module, r_{i_1}, r_{i_2} have the same consequent part as r_i (the c hyperplanes y_i^j with $j \in [1, c]$).

C. Detection of brutal concept drifts

Contrary to current rule creation criterion which use no information of neighborhood [2], [8], we propose here to integrate a brutal drift detector based on a clustering separability criteria. The idea is to assume that if the two sub-rules in the anticipation module are enough separated, then a brutal drift occurred. Then, r_{m_2} learned over the large history matches the old concept and r_{m_1} learned over the few last points matches the new drifted concept. Eq. (12) presents the proposed separability criteria that is based on the covariance of both clusters.

Condition 1 $$||\mu_i - \mu_j|| > \sigma_i + \sigma_j \qquad (12)$$

Where, as depicetd in Figure 4, σ_i (resp. σ_j) is the distance between μ_i (resp. μ_j) and the hyper-ellipsoid's envelop of cluster i (resp. j), along (μ_i, μ_j) axis. Besides, to force the rules to take into account a certain number of points n_{min} (20

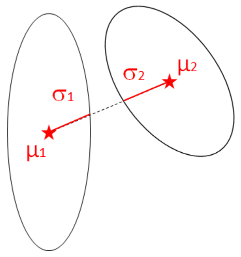

Fig. 4: Separability criteria using covariance of both clusters

by default) before deciding to create a new rule, the following inertia criteria is added:

Condition 2 $$k_i > n_{min} \quad (13)$$

D. Initialization of new rules based on the anticipation module

If the rule creation conditions are met for the most activated rule r_m from the principal system, then r_m is replaced by r_{m_1} and r_{m_2} in the principal system. Then for each new r_{m_i} of the principal system, two new sub-rules $r_{m_{i_1}}$, $r_{m_{i_2}}$ are initialized in the anticipation module, as follows:

$$\begin{array}{lll} \mu_{r_{m_{i_1}}} = \mu_{r_{m_{i_2}}} = \mu_{r_{m_i}} & & A_{r_{m_{i_1}}} = A_{r_{m_{i_2}}} = A_{r_{m_i}} \\ \Pi_{r_{m_{i_1}}} = \Pi_{r_{m_i}} & \Pi_{r_{m_{i_2}}} = 0 & k_{r_{m_{i_1}}} = k_{r_{m_{i_2}}} = 0 \end{array} \quad (14)$$

The idea is to keep the learned information of r_{m_i} in the sub-rule $r_{m_{i_1}}$, and to initialize the second $r_{m_{i_2}}$ from scratch. All steps of the learning are summarized in Algorithm 1.

Algorithm 1 Learn ParaFIS model

```
Require: $x_{new}$, current fuzzy rule system containing set of rules R
  and set of classes C
  if $class(x_{new}) \in C$ then
    Find max activated rule $r_m \in$ R according to (3)
    Check the condition 1 and condition 2 Eq. (12-13) with $r_m$
    if Condition 1 & Condition 2 == FALSE then
      According to Eq.(7)-(8), update the antecedent part of $r_m$
      with $\alpha = 1$, and $r_{m_1}$, $r_{m_2}$ with $\alpha_1 \neq \alpha_2$
      According to Eq. (9)-(10), update the consequent part of all
      rules $r_i$, $r_{i_1}$ and $r_{i_2}$
    else
      Replace $r_m$ by $r_{m_1}$,$r_{m_2}$ in the principal system
      Initialize 4 new sub-rules in the anticipation module Eq. (14)
    end if
  else
    Create a new rule Eq. (11)
    Initialized two sub-rules in the anticipation module with (11)
  end if
```

IV. Experimental Validation

A. Evaluation protocol

1) Prequential test with artificial brutal drift: To evaluate the performance of an online classifier, it is current to use the prequential test [11]. In this test, data are given one by one to the system. The system first tests the new data to get a score (1 if the class is well classified, 0 otherwise) and then learns

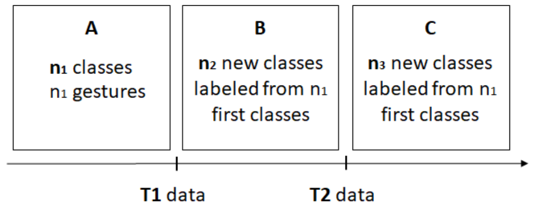

Fig. 5: Protocol P - Generation of data stream with brutal drifts

Dataset	Classes	Features	Samples	Scriptwriters
Letters	26	16	20000	20
LaViola	48	50	16891	34
PenDigits	10	16	10992	44

TABLE I: Information on the datasets used in the experiments

on it. In this way, all the data are used to test the system and then, to train it while maintaining independence between each phase. Scores are then averaged over a certain window (of size n) to get a smooth curve of the performance over time. This test simulates a real usecase as for many online application where the system must adapt to the behavior of a user along time. In the following experiments, $n = 5$ (except to plot the figures $n = 100$ to smooth the score).
There is no existing dataset with annotation of the nature of the drift or with the occurrence time of the drift making the evaluation and comparison of online classifiers complex. To make the task easier, the paper proposes to generate artificial brutal drifts in real data at a chosen time. To do so, each dataset is split into three sub-datasets A, B, C to create a specific data stream. As illustrated in Figure 5, A is composed of the first $T1$ data belonging to n_1 classes. B (resp. C) is composed of the data in the stream between $T1$ and $T2$ (resp. between $T2$ and $T3$), this data belongs to $n_2 \leq n_1$ (resp. $n_3 \leq n_1$) different classes and are relabeled with the n_1 labels to produce the brutal drift. In this ways, n_2 brutal drifts are done at the time $T1$ and n_3 others at time $T2$. Thereafter, we call this approach the **protocol** P. In the context of a command-gesture system, the protocol is equivalent to a user changing the gesture of a command while keeping the possibility to make the old gesture.

2) Benchmark dataset: Regarding the context of command-gestures system, three datasets from the handwritten pattern recognition community have been chosen to assess the performance of the system: PenDigits [12], Letters [13] and LaViola [14]. They are all available in the UCI machine learning repository [15]. All three datasets contain different features extracted from the handwritten patterns (digits, letters or symbols handwritten by different writers). Description of each dataset is given in Table I. These datasets are built in a static context meaning that there is no order between data. Thus, the prequential test with artificial brutal drifts can be done several times (m) by shuffling the dataset. All results given thereafter are averaged over m prequential tests with $m = 100$. The experimental parameters are given in Table

	T1	T2-T1	T3-T2	n1	n2	n3
Letters	2000	4000	4000	10	10	6
PenDigits	2000	3000	3000	4	3	3
Laviola	2000	3000	3000	10	10	10

TABLE II: Experimental parameters

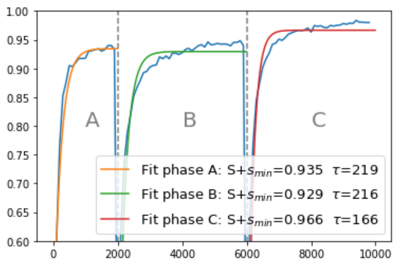

Fig. 6: Fit of the three phases score using Eq. (15)

II. The $n1, n2, n3$ classes of each dataset follow the order of occurrence in the data file from UCI.

3) Characterization of the plasticity and stability: In order to measure the plasticity and stability of the system, we propose to fit the prequential score with an handcraft model given by Eq. (15).

$$y(t) = S(1 - e^{-\frac{t}{\tau}}) + s_{min} \quad (15)$$

Where we define a characteristic time τ, which represents the reactivity time (or the plasticity). In particular after a time $t = \tau$, 63% of the score have been reached until the steady state. At the steady state, the score is given by $S + s_{min}$. A least square method is used to determine the parameters which best fit the curve with the model. The example of fits on the Letters dataset (experimented with the protocol P) given in Figure 6 shows that this model fits well the prequential score.

B. Evaluation of the anticipation

1) Importance of covariance matrix initialization: No study has yet tackled the issue of optimizing the initialization parameters of the antecedent part. Yet, the initialization of the prototype (the antecedent part) greatly influence the score. To show it, we propose to compare three methods currently used in incremental fuzzy system to initialize the covariance of a new rule: Method I1 (Eq. (16), [5]), Method I2 (Eq. (17), [2], [9]), Method I3 (Eq. (16)). The center of prototypes is initialized just on the last point for all methods.

$\forall k, l \in [1, n]$, $\forall j \in [1, N]$

$$\textbf{I1:} \quad cov_{kl} = min_j(cov^j_{kl})\delta_{kl} \quad (16)$$

$$\textbf{I2:} \quad cov_{kl} = \frac{1}{100} * \delta_{kl} \quad (17)$$

$$\textbf{I3:} \quad cov_{kl} = \frac{mean_j(cov^j_{kl})}{10} \quad (18)$$

To compare them, we use the generalized evolving fuzzy system as described section II with **Condition 1** and **Condition 2** to create rules (section III). Results are obtained from the three datasets and a fit using Eq. (15) is done. Results are displayed in Figure 7 column A, and the fitted parameters are shown in Table III. Results show that the best choice of initialization depends on the dataset. However it is clear that the initialization of the rules widely impacts the results. For instance, there is 5.6% of difference for the $S + s_{min}$ parameters between methods $I3$ and $I2$ for letters dataset at the phase B. This highlights the importance of well initializing the covariance matrix of the new rule. This can be explained by the fact the covariance matrix plays a great role in the learning. Indeed, only the antecedent part of the most activated rule is learned on the new incoming point and the activation, computed with a Mahalanobis distance, crucially depends on the shape of the covariance matrix. If it is too small, few data from the new concept will activate the new rule. If it is too big, data from other concepts may also activate the new rule.

		A		B		C	
		$S{+}\,s_{min}$	τ	$S{+}\,s_{min}$	τ	$S{+}\,s_{min}$	τ
Letters	I1	**93.4**	218	87.9	443	94.0	377
	I2	**93.4**	215	83.3	556	89.8	697
	I3	92.6	**157**	**89.9**	**243**	**94.1**	**276**
PenDigits	I1	**98.9**	80	**99.3**	**29**	**98.1**	**233**
	I2	98.9	80	98.7	59	96.9	415
	I3	98.9	77	98.7	36	97.4	235
Laviola	I1	**98.7**	**66**	96.7	88	96.5	131
	I2	98.7	68	96.7	90	96.3	135
	I3	98.2	67	**97.1**	**66**	**97.2**	**92**

TABLE III: Fitted parameters of the handcraft model (15)

2) Comparison with our proposition: Now, the impact of the anticipation will be evaluated.The antecedent part initialization is compared when using the best method among {I1,I2,I3} for each dataset, and using the anticipation module Eq.(14). However the drift detector can depend on the parameters of the system as it is in ours, Eq. (12) (this depends on the covariance matrices). Thus different initialization of prototype change the times of detection. To avoid this bias, the drift detector is first used on the system with anticipation and the times t, at which the drifts are detected, are saved. Then, rather than using the detector, we use the saved files with all times t to create rules. In such ways, the system can be compared fairly without bias of the detection.
Two sets of parameters are used for the anticipation, $Para_1 = (\alpha_1 = 1, \alpha_2 = 0.9)$ and $Para_2 = (\alpha_1 = 1, \alpha_2 = 0.95)$ with α_1 (resp. α_2) the forgetting factor of the sub-rules $i1$ (resp. $i2$) of the secondary system for $i \in [1, r]$.
Moreover, a comparison is done with our own implementation of an EFS similar to Gen-Smart EFS [2] called GEFS*. In this last one, the rule creation criteria (given by Eq. (19)-(20)) is the same than Gen-smart EFS [2] with the same rule initialization method (I2). However, there are no rule merging or rule splitting methods as there are in Gen-Smart EFS.

$$(\mathbf{x} - \mu_i)A^{-1}(\mathbf{x} - \mu_i)^T > r_i^2 \quad (19)$$

$$r_i = \kappa p^{\frac{1}{\sqrt{2}}} \frac{1.0}{(1 - \frac{1}{k_i+1})^m} \quad (20)$$

		$< S + s_{min} >$	$< \tau >$	$< Acc >$
Letters	$Para_1$	**94.3**	200	**90.3**
	$Para_2$	93.8	**188**	90.1
	I3	92.2	214	88.1
	GEFS*	89.9	471	80.7
	$\kappa = 1.6$, $m = 4$			
PenDigits	$Para_1$	98.8	**56**	97.9
	$Para_2$	**98.9**	**56**	**98.0**
	I1	98.8	103	97.2
	GEFS*	98.7	156	96.3
	$\kappa = 2.6$, $m = 4$			
Laviola	$Para_1$	**98.2**	**74**	**96.2**
	$Para_2$	**98.2**	84	95.9
	I3	97.7	76	95.7
	GEFS*	97.1	158	94.4
	$\kappa = 2.5$, $m = 4$			

TABLE IV: Mean score over the three phases A,B,C

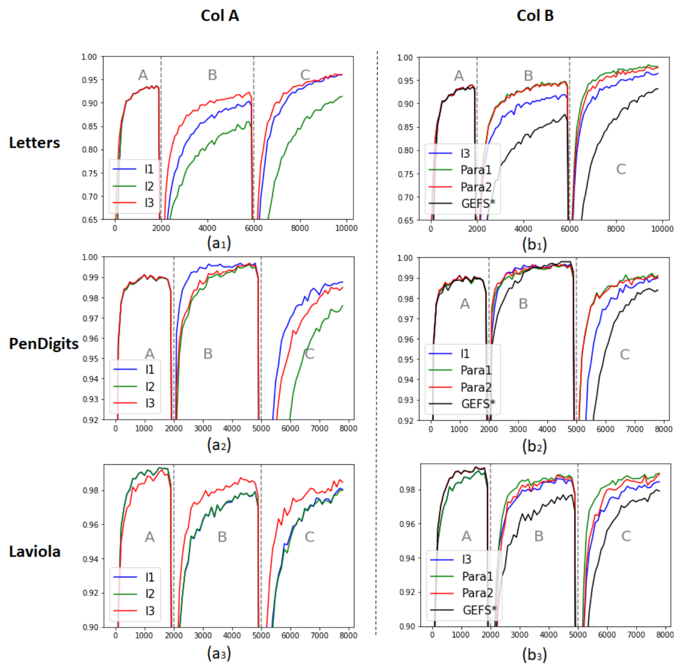

Fig. 7: Prequential score (y-axis) with respect to data (x-axis) for the three datasets. The first (resp. the second) column refers to the first (resp. second) experiment

Results are displayed in Figure 7 column B, and Table IV. The fitted parameters of each configuration are averaged over the three phases. The mean accuracy score is also given to compare global performance of each configuration. The results show that anticipation brings a gain in the mean score, not only on the reactivity with a lower time τ, but also on the score reached in the steady state. This results in a better mean accuracy score. It means that the anticipation effectively accelerates the learning of the new concept when new rule are created, but also stabilizes the covariance matrix to better match the target concept at end. Moreover, our global system ParaFIS with detector+anticipation outperforms, in term of reactivity and stability, an equivalent system GEFS* with detector+initialization from the state of the art [2].

V. Conclusion & Outlooks

This paper has introduced a new design of EFS which integrates an anticipation module to deal with brutal drifts. Anticipation opens up a new interesting way to tackle non stationary environment problems. It has allowed to detect brutal drifts and adapt new rules to the drifted concept with a gain in reactivity and stability. The new design opens up to new outlooks with the two sub-rules that map the rule creation problem into a 2-cluster clustering problem. It may use other online clustering validity criteria to detect brutal drifts or add new anticipation modules to tackle other natures of drifts such as gradual drifts (where data switch from one target concept to another one several times).

References

[1] M. Bouillon and E. Anquetil. Online active supervision of an evolving classifier for customized-gesture-command learning. *Neurocomputing*, 262:77–89, Nov 2017.

[2] E. Lughofer, M. Pratama, and I. Skrjanc. Incremental rule splitting in generalized evolving fuzzy systems for autonomous drift compensation. *IEEE Transactions on Fuzzy Systems*, 26(4):1854–1865, Aug 2018.

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

[4] A. Liu, G. Zhang, and J. Lu. Fuzzy time windowing for gradual concept drift adaptation. *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, Jul 2017.

[5] M. Bouillon, E. Anquetil, and A. A. Almaksour. Decremental learning of evolving fuzzy inference systems using a sliding window. In *Proceedings of the 2012 11th International Conference on Machine Learning and Applications - Volume 01*, ICMLA '12, pages 598–601, Washington, DC, USA, 2012. IEEE Computer Society.

[6] Y. Song, G. Zhang, H. Lu, and J. Lu. A self-adaptive fuzzy network for prediction in non-stationary environments. *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8, 2018.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[8] A. A. Almaksour and E. Anquetil. Improving premise structure in evolving takagi–sugeno neuro-fuzzy classifiers. *Evolving Systems*, 2(1):25–33, Mar 2011.

[9] A. Lemos, W. Caminhas, and F. Gomide. Multivariable gaussian evolving fuzzy modeling system. *IEEE Transactions on Fuzzy Systems*, 19(1):91–104, Feb 2011.

[10] P. P. Angelov and D. P. Filev. An approach to online identification of takagi-sugeno fuzzy models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):484–498, Feb 2004.

[11] J. Gama, R. Sebastião, and P. P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, Mar 2013.

[12] F. Alimoglu. *Combining Multiple Classifiers for Pen-Based Handwritten Digit Recognition*. PhD thesis, Bogazici Univeristy, 1996.

[13] P. W. Frey and D. J. Slate. Letter recognition using Holland-style adaptive classifiers. *Mach. Learn.*, 6(2):161–182, Mar 1991.

[14] J. J. LaViola and R. C. Zeleznik. A practical approach for writer-dependent symbol recognition using a writer-independent symbol recognizers, November 2007.

[15] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017.