Trend-Based Dynamic Classification for on-line Diagnosis of Time-Varying Dynamic Systems

Nathalie Andrea Barbosa Roa, Louise Travé-Massuyès, Victor Hugo Grisales Palacio

Preprints, 9th IFAC Symposium on Fault Detection, Supervision and
Safety of Technical Processes
September 2-4, 2015. Arts et Métiers ParisTech, Paris, France

Trend-Based Dynamic Classification for on-line Diagnosis of Time-Varying Dynamic Systems

Nathalie A. Barbosa * **Louise Travé-Massuyès** ** **Victor H. Grisales** ***

* *Universidad Nacional de Colombia, Bogotá, Colombia*
Université Paul Sabatier, LAAS, F-31400 Toulouse, France
(e-mail: nabarbosa@unal.edu.co - nabarbos@laas.fr)
** *CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
Université de Toulouse, LAAS, F-31400 Toulouse, France
(e-mail: louise@laas.fr)
*** *Universidad Nacional de Colombia, Department of Mechanical and Mechatronics Engineering, Bogotá, Colombia*
(e-mail: vhgrisalesp@unal.edu.co)

Abstract:
Nowadays systems must adapt to rapidly changing environments and must show behaviour that evolve with time. On the other hand they are intensively monitored so that huge amounts of data are generally collected and available for further analysis. Data-driven diagnosis systems must hence face new challenges, in particular be supported by efficient classification algorithms that adapt the targeted model âĂŞ or classifier âĂŞ to the evolution of the system and be able to scale to big data. In this paper, a proposal which couples a dynamic clustering method with an on-line trend extraction algorithm that works incrementally on the incoming data is presented. The dynamic clustering method is based on micro-clusters that may drift, merge and split, hence following the evolution of the system. The trend extraction method applies to individual signals and generates a compact abstraction in the form of episodes. The episodes of all signals are then put together to feed the dynamic clustering method. This approach allows data reduction and makes it suitable for on-line data analysis and diagnosis in real-time and low-memory requirements. The proposed algorithm is tested successfully on a continuous stirred tank heater benchmark suffering faults with varying magnitude.

Keywords: Classification, Dynamic systems, Pattern recognition, Data processing, Trajectories

1. INTRODUCTION

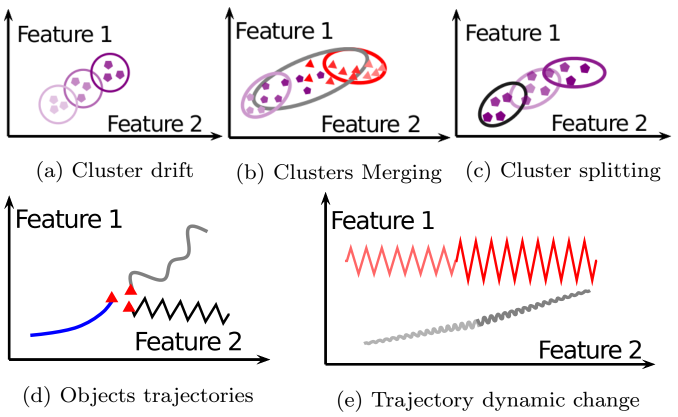
Technological advances of past decades have resulted in production and service infrastructures highly adaptive to a constantly changing environment. In addition, they have changed the way enterprises get information about the state of their systems. Huge amounts of data, arising from various sources, are generally collected and they are available for further analysis. These two facts have promoted the success of machine learning approaches for diagnosis tasks, although they must face new challenges, in particular building efficient classification algorithms that adapt the targeted model – or classifier – to the evolution of the system and that scale to big data.

In classical approaches the classifier is designed using a training set of examples and remains unchanged over time (its structure and sometimes even its parameters). If the performance of the classifier decreases below a given threshold the classifier may be re-learned, but it does not adapt dynamically. In this context, the new objects submitted to the classifier during the recognition stage do not imply a change of the classifier. This type of classification is called *static classification* ([Joentgen et al., 1999]). The interested reader is referred to [Venkatasubramanian et al., 2003] for a detailed review of classic static classification and clustering approaches. If the classifier changes in time (dynamically), the classification task becomes a *dynamic classification problem.*

Dynamism in the classifier is achieved when not only the parameters but the classifier structure changes according to input data in an automatic way. Abrupt changes in the data can be captured by cluster creation or elimination. Smooth changes are usually reflected as cluster drifts and less frequently as cluster merging and splitting, see figures 1a, 1b and 1c. Among the techniques that have been used for dynamic classification, we can mention: evolving clustering ([Angelov and Zhou, 2008, Angelov, 2011]), Self-Adaptive feed-forward neural network (SAFN) ([Li et al., 2011]), LAMDA (Learning Algorithm for Multivariable Data Analysis) ([Kempowsky et al., 2006]) and Growing Gaussian Mixture Models (2G2M) ([Bouchachia and Vanaret, 2011]). All these works consider adaptation to abrupt changes through cluster creation but LAMDA and 2G2M do not consider elimination. Only SAFN considers cluster drift, merge and split. Evolving clustering considers cluster replacement to handle cluster drift. Cluster replacement involves cluster creation around the new focal point followed by the old cluster elimination. Some of these alternatives are really complex and hence not suited to handle online large amounts of data, such as data arriving in a stream. Their requirements in terms of memory and processor power are too high. Two-stages clustering (online/offline) has emerged as an alternative to deal with large amounts of data arriving at fast rates. Some authors have used the two-stage clustering in order to achieve on-line classification. For instance, [Aggarwal et al., 2003a] proposed the CluStream algorithm, [Kranen et al., 2011] introduced ClusTree and [Cao et al., 2006] included density as a pruning factor creating the DenStream technique. In these two-stage algorithms, data are collected, pre-processed, and compressed in the first stage forming what is called *micro-clusters* (μ-clusters). In the second stage, the μ-clusters are grouped into macro-clusters or actual clusters. CluStream uses a pyramidal time framework (as called by the authors) to analyse cluster time evolution by means of stored pictures of the μ-clusters, called "snapshots". These snapshots provide an effective trade-off between storage requirements and the ability to recall summary statistics from different time horizons. ClusTree [Kranen et al., 2011] proposes a hierarchical tree shaped index structure for the μ-clusters which provides efficient μ-cluster location for object insertion at different levels of granularity achieving data-cluster assignation even at fast rates. ClusTree improves the limitation of CluStream about the fix number of micro-clusters. DenStream uses density based final clustering handling outliers and cluster evolution with the use of low, medium and high density μ-clusters.

The algorithms referred above handle the evolution of the system in the sense of behaviour changing in time. The system is then qualified as *time-varying*. In this case, the intrinsic dynamism of the system must also be taken into account. A system is *dynamic* if its output behaviour in response to a request to the input, involves a memory of past states. This suggest that dynamic behaviour could be characterized by trajectories and their temporal patterns. In this work, we are concerned with dynamic classification for solving on-line diagnosis problems for time-varying dynamic systems or processes. Whereas the time-varying property can be handled with dynamic classification methods, dynamic behaviour requires to build specific features that characterize variable trajectories rather than instant *snapshot* behaviour [Zimmermann, 2000]. For example in figure 1d, if classification is performed using only the current value of the variable (triangle points) instead of the associated trajectories, the three completely different behaviours could be diagnosed as the same. Figure 1e shows two temporal patterns representing dynamical behaviour.

In this paper, a proposal which couples a dynamic clustering method with an on-line trend extraction algorithm that works incrementally on the incoming data is presented. It is used to achieve on-line diagnosis and has been tested on a continuous stirred tank heater (CSTH) model developed by [Thornhill et al., 2008]. Section 2 explains the dynamic classifier structure and section 3 presents the application to process data represented by time series for which a



(a) Cluster drift (b) Clusters Merging (c) Cluster splitting

(d) Objects trajectories (e) Trajectory dynamic change

Fig. 1. Time-varying dynamic system's data representation

specific mathematical characterization is performed. Section 4 is concerned with the diagnosis case study on the CSTH benchmark and section 5 provides conclusions and discusses future work.

2. DYNAMIC CLUSTERING ALGORITHM

This paper uses an offline-online clustering approach popularized by [Ester et al., 1996, Aggarwal et al., 2003a, Cao et al., 2006, Kranen et al., 2011] among others. In this approach the online stage creates μ-clusters that are summarized representations of a data set made by using some statistical and temporal information. The principle is that some data groups are close enough to make it possible to consider that they belong to the same final cluster. The offline stage analyses the distribution of those μ-clusters whose density is considered as medium or high and creates the final clusters by a density based approach, that is, dense μ-clusters that are close enough (connected) are said to belong to the same cluster. Moreover, similarly to [Chen and Tu, 2007], a cluster is defined as the group of connected μ-clusters where every inside μ-cluster presents high density and every outside μ-cluster exhibits either medium or low density. Interestingly, this dense μ-cluster structure has proved outlier detection capabilities in evolving environments [Cao et al., 2006]. The online and offline stages are further explained in 2.1 and 2.2. The contributions of the presented technique over those reviewed in section 1 are: a. The technique presented in this paper achieves full on-line dynamic clustering of time-series. b. The joint use of a distance-based and density-based clustering stages(sec. 2.1, sec. 2.2). c. The automatic characterization of signals temporal behaviour (sec. 3.1). d. Large amounts of data can be processed on-line without information loss due to the episode representation (sec. 3)

2.1 *On-line clustering*

Considering a d-dimensional object e, a μ-cluster is the representation of a group of data points **close in all dimensions** and whose information is summarized in a characteristic feature vector (CF). This CF has the following form:

$$CF_k = (n_k, LS_k, SS_k, t_{lk}, t_{sk}, D_k, Class_k) \qquad (1)$$

where $n_k \in \Re$ is the number of objects in the μ-cluster k, $LS_k \in \Re^d$ is the vector containing the linear sum of each feature over the n_k objects, $SS_k \in \Re^d$ is the square sum of feature over the n_k objects, $t_{lk} \in \Re$ is the time when the last object was assigned to that μ-cluster, $t_{sk} \in \Re$ is the time when the μ-cluster was created, D_k is the μ-cluster density and $Class_k$ is the μ-cluster label if known. Using LS_k, SS_k and n_k the variance of the group of objects assigned to μ-cluster k can be calculated. In order to maintain an up-to-date structure, μ-clusters are weighted with an exponential decay function dependent on the current time t_i and the last assignation time t_{lk}. This function $\beta^{-\lambda(t_i - t_{lk})}$ emulates an ageing process over a damped window. If β is chosen as 2^{ψ}, then the half life of data in the window is $\frac{1}{\psi\lambda}$, $\lambda > 0$.

When a new object e_i is assigned to a cluster, $t_{lk} = t_i$. If the object belong to a known class (semi-supervised approach), $Class_k = Class_i$. The other features are actualized as follows:

$$n_k^{(t)} = n^{(t-1)}\beta^{-\lambda(t-t_{lk})} + 1 \quad (2)$$

$$LS_k^{(t)} = LS^{(t-1)}\beta^{-\lambda(t-t_{lk})} + e_i \quad (3)$$

$$SS_k^{(t)} = SS_k^{(t-1)}\beta^{-\lambda(t-t_{lk})} + e_i^2 \quad (4)$$

In general, if no object is added to a μ-cluster during the time interval $(t, t + \Delta t)$, its CF at $t + \Delta t$ can be calculated from $CF^{(t)}$ using the decay function for the weighted parameters as follows:

$$CF^{(t+\Delta t)} = \beta^{(-\lambda \Delta t)} CF^{(t)} \quad (5)$$

In order to find clusters with arbitrary shape, density based clustering is executed over the μ-clusters. A μ-cluster may be of three different types: dense μ-cluster ($D\mu$-cluster), semi-dense μ-cluster ($S\mu$-cluster) and low density or outlier μ-cluster ($O\mu$-cluster). The difference between each type is established based on a threshold. $S\mu$-clusters could be the product of an increment in the number of outliers or part of a cluster creation, so they have to be updated more frequently than other μ-clusters. To speed up the analysis three lists were used. The first one includes the active μ-clusters, i.e. $D\mu$-clusters and $S\mu$-clusters. The second one contains the current $O\mu$-clusters. The third list includes those μ-clusters that were once dense but, because of a lack of new elements, have lost density, and became non active μ-clusters.

The μ-cluster is shaped as a d-dimensional box since the absolute value of the difference between the feature and the cluster descriptor is used as distance measure. The size of the boxes are set as a fraction of the feature range. This fraction can be established according to the data context; if no context is available in advance, it may be established on-line. The box size per feature is found according to (6), where ϕ is a constant establishing the fraction. The algorithm can work over a known context, given by the user, or can build the context along the way. The context is based on the minimum and maximum values of each feature and is used for normalization purposes.

$$S_k^d = \phi\,(max_d - min_d) \qquad \forall d \quad (6)$$

μ-cluster density is calculated using the current number of objects n_k and the current hypervolume of the bounding box, as shown in (7).

$$D_k = \frac{n_k}{V} \quad (7)$$

A μ-cluster μC_z is said to be dense at time t if it satisfies the inequality (8), where V_k is the volume of the cluster box k, α is a proportion parameter used to set the threshold and K is the total number of μ-clusters. For the case where all μ-clusters have the same size, the condition can be simplified as seen in (9).

$$D_z \geq \alpha \frac{\sum_{k=1}^{K} n_k}{\sum_{k=1}^{K} V_k} \quad (8)$$

$$D_z \geq \alpha \frac{\sum_{k=1}^{K} n_k}{KV} \quad (9)$$

At time t, a μ-cluster μC_z is said to be semi-dense if its density fulfils the following inequalities:

$$\alpha \frac{\sum_{k=1}^{K} n_k}{KV} \geq \frac{d_z}{\sum_{k=1}^{K} d_k} \geq \frac{\alpha}{2} \frac{\sum_{k=1}^{K} n_k}{KV} \quad (10)$$

A μ-cluster μC_z is considered $O\mu$-cluster if its density is lower than the lower limit in (10).

2.2 Off-line density clustering

To find the final clusters, the dense character of a μ-cluster and its neighbours is analysed with a certain periodicity. Let μC_x and μC_y be μ-clusters, then μC_x and μC_y are said to be **directly connected** if their hyperboxes overlap in all but φ dimensions. The parameter φ establishes the feature selectivity of the classifier.

An μ-cluster μC_1 is said to be connected to μC_n if there is a chain of μ-clusters $\{\mu C_1, \mu C_2, \cdots, \mu C_n\}$ such that μC_i is directly connected to μC_{i+1} for $i = 1, 2, \cdots, n-1$. A set of μ-clusters connected is said to be a *group*. Finally, a μ-cluster group is said to be a cluster if every inside μ-cluster of the group is a $D\mu$-cluster and every border μ-cluster is either a $D\mu$-cluster, a $S\mu$-cluster or a $O\mu$-cluster.

$O\mu$-clusters do not contribute to the final clusters. We assume that μ-clusters with low density are either outliers or possible new clusters in an emerging state. The later case reveals itself with an increment in the cluster density and consequently, this μ-cluster grows into a $S\mu$-cluster in the following time windows. Once a classification is found off-line the results are stored as a snapshot that might be examined in the future in order to extract more information about the system evolution. Snapshots are stored following a pyramidal time scheme as the one proposed in ([Aggarwal et al., 2003b]). Section 2.3 explains the operation of the algorithm in detail.

2.3 Algorithm operation

When a new identified object arrives, the algorithm verifies the existence of μ-clusters (if an offline learning stage was previously used). If no μ-cluster exists, it creates a μ-cluster using the data of the current object. If there is already one or more μ-clusters created, the algorithm finds out which $D\mu$-cluster or $S\mu$-cluster is the closest. The distance between an object x and a μ-cluster μC_k is calculated as the sum of the distances between the μ-cluster representation c_k^i and the value of each object feature x^i. The distance is calculated using the absolute value as shown in (11).

$$dis^i = \left|x^i - c_k^i\right| \tag{11}$$

Once the closest μ-cluster is found, the maximum distance condition is verified, that is, the object must fit inside the maximum possible bounding box MAX_{BB} centred in the μ-cluster. If there is a fit, the point is absorbed by the μ-cluster; otherwise, the algorithm tries to merge the object with the closest $O\mu$-cluster. If the distance condition and the qualitative features condition are fulfilled, the object is merged and after that, the density of the $O\mu$-cluster is verified to decide if the μcluster has grown into a $S\mu$-cluster. When an $O\mu$-cluster becomes denser it is removed from the outlier list and with its statistics a new $S\mu$-cluster is created and inserted into the active μ-cluster list. In the case where no μ-cluster is at size match of the object, a new $O\mu$-cluster is created with the object and the current time is used as t_s for the new $O\mu$-cluster. Each window period $S\mu$-cluster has to be updated and its density evaluated in order to establish if it has fallen below the semi-dense threshold and must be turned into a new $O\mu$-cluster that inherits its statistics. $O\mu$-clusters are also evaluated in each window period to find out if their density is below the low-density threshold. If that is the case, the $O\mu$-cluster is eliminated and no longer considered.

The offline classification of the μ-clusters takes place each t_{off} seconds, been t_{off} an user defined time period. In this classification all dense or semi-dense μ-clusters are used as possible geometric centers of the final clusters. Starting from any dense μ-cluster with unknown class, all its neighbours are found and tag as belonging to the same class. In a second stage the neighbours are used as seeds and all their neighbours are also found and added to the seeds list until all connected μ-clusters are found. Finally all those connected μ-clusters, directly or not, are referred as to a single cluster. If more dense clusters remain unclassified the process continues taking one of these as first seed of a new cluster and finding all the μ-clusters who are connected to it. The process stops were no unclassified $D\mu$-clusters prevail. The results of the classification process are then store as a snapshot of the state of the system in that exact moment. Following a pyramidal time framework snapshots from different times are stored given several time scales that can be used for cluster evolution analysis. An example of this framework for $t = 55$ seconds, assuming a snapshot is taken each second, can be seen in table 1. In this work instead of using time directly, the number of the classification is used as base for the pyramidal framework, like this: Once a classification snapshot is ready, it is stored on the highest pyramidal level determined as $log_\rho(class_n)$. To maintain a reasonable amount of snapshots the amount of elements in each level is limited to ρ^ζ. Therefore, if a level contains more than ρ^ζ snapshots, the oldest one is removed. Parameters ρ and ζ are set according to the amount of memory available for storage.

3. APPLICATION TO PROCESS DATA

Industrial processes are likely to have hundreds to thousands of signals coming from sensors located all over the plant. These signals usually report the behaviour of the process in a normal established operating region, which is characterized by zero valued first and second derivatives (in practice, due to the presence of noise, $|\dot{x}| < \epsilon_1$ and $|\ddot{x}| < \epsilon_2$). It is expected that in the proximity of a process situation change (due to faulty behaviour or to change of the operating point), sensors show a deviation from their normal value range ($|\dot{x}| > \epsilon_1$ and\or $|\ddot{x}| > \epsilon_2$) following transitional dynamics that can be characterized by means of trends. A trend is a representation of the variation of a process variable within a time window. Process situation assessment can be achieved by extracting trends from the measured signals and evaluating their similarity with known behaviour.

Order of snapshots	Clock Time (last snapshots)
0	55 ~~54~~ 53 ~~52~~ 51
1	54 ~~52~~ 50 ~~48~~ 46
2	52 ~~48~~ 44 ~~40~~ 36
3	~~48~~ 40 ~~32~~ 24 ~~16~~
4	48 ~~32~~ 16
5	32

Table 1. Pyramidal framework for $t = 55$. Based on [Aggarwal et al., 2003b]

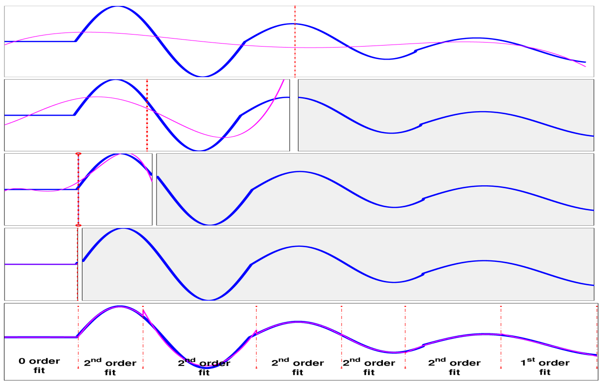


Fig. 2. Interval halving polynomial approximation

Process data takes the form of time series corresponding to each sampled time signal. In order to extract trend information, this work proposes to process those time series into *episodes* ([Dash et al., 2004]). Episodes are representations defined by three elements: a trend context TC, a set of auxiliary variables AV and a time interval T_i leading to (12).

$$e(x^i) = \{TC, AV, T_i\} \tag{12}$$

In this work, trend context is given by polynomial fitting coefficients, therefore, for a polynomial fit of order n, TC is the vector $TC = [c_0, c_1, \ldots, c_n]$. Auxiliary variables can be information related to measurements like: average value, standard deviation, etc. The abstraction of a time series into a sequence of episodes is presented in details in section 3.1. When the episodes corresponding to all signals are considered together, they abstract the whole behaviour of the system.

3.1 Episode characterization

The simplest approach to extract a trend is to fit a polynomial to the data. According to the Weierstrass approximation theorem [Stone, 1948], a continuous function,

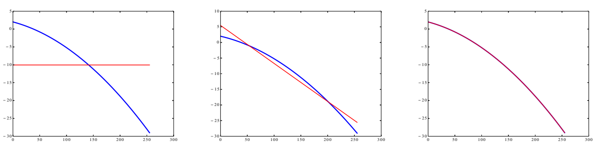
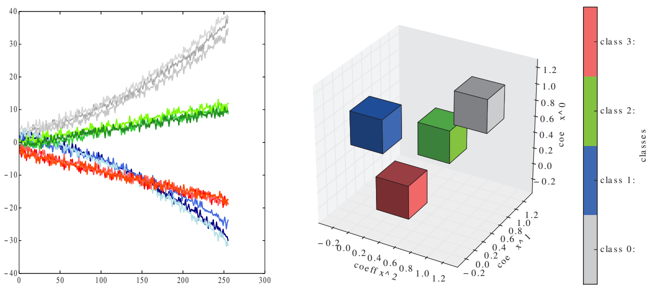


Fig. 3. Fit with polynomials of order 0 to 2. a. Order $n = 0$ b. Order $n = 1$ c. Order $n = 2$.

defined in a closed interval, can be approximated as closely as desired using polynomials of sufficiently high order. There are two ways to approximate a time function using polynomials: increase the polynomial order until there is a fit or reduce the interval size into one in which the function can be approximated by a low order polynomial. Between the two ways, the later is generally preferred since polynomial-fitting complexity increases with polynomial order. One example can be seen in [Dash et al., 2004], where the authors propose an interval-halving algorithm for trend extraction which automatically identifies qualitative trends using polynomial fitting. The algorithm, known as an *interval halving algorithm* tries to fit a polynomial of order $O < n_{max}$ to the data within a time window. If the fitting error is bigger than some threshold established before, the window is split at the midpoint and polynomial fitting is performed on the new half size window. The split process finishes when the fitting error is below the threshold. The leftover data is analysed following the same method until all the data within the initial window is fitted. A graphical representation of the interval halving algorithm is shown in figure 2. An online implementation of the interval halving algorithm has been developed by [Maurya et al., 2010], where the online trend extraction is achieved using a time window that moves as more data become available.

Based on the same interval halving principle, we develop an algorithm that fits polynomials of maximal order 2 to the data. The time window is built with the incoming data and as soon as the window length is completed, data is analysed and the current trend is extracted. Once a window is established, the fitting process is performed and the coefficients of the best fit (minimum fitting error) are saved if the fitting is acceptable. An example of polynomial fit is shown in figure 3, where the coefficients found for polynomial approximation of order $n = 0$ are $c = [0, 0, -10.0496]$, of order $n = 1$ are $c = [0, 31.008, 5.4544]$ and of order $n = 2$ are $c = [-20.808, -10.2, 2]$. The fitting is said to be acceptable if the variance of the fitting residues is less or equal to the variance of noise in the signal. An estimation of the noise standard deviation in the measured signal can be achieved using wavelet decomposition. As described by [Bakshi, 1999], in the wavelet multiscale decomposition, coefficients corresponding to the true signal are larger in magnitude than those corresponding to noise; nevertheless, if the noise is known to be white or uncorrelated, the majority of coefficients will relate to noise and show a constant power spectrum at all frequencies. The above principles guided us to the use of the robust median absolute deviation (MAD) as a method for finding an estimation of the noise variance from wavelet decomposition coefficients. Noise standard deviation is calculated by (13), where d_m denotes the wavelet coefficients at the selected scale:



(a) Signals to be classified (b) Episode based classification

Fig. 4. Illustrative classification task of noisy time series

Variable	OP 1	OP 2
$Level(mA)$	12.00	12.00
$Level(cm)$	20.48	20.48
$CWflow(mA)$	11.89	7.330
$CWflow(m^3s^{-1})$	9.038×10^{-5}	3.823×10^{-5}
$CWvalve(mA)$	12.96	7.704
$Temperature(mA)$	10.50	10.50
$Temperature(°\mathrm{C})$	42.52	42.52
$Steamvalve(mA)$	12.57	6.053
$HWvalve(mA)$	0	5.500
$HWflow(m^3s^{-1})$	0	5.215×10^{-5}

Table 2. Suggested operational points for the CSTH

$$\sigma_m = \frac{1}{0.6745} median\left(|d_m|\right) \tag{13}$$

For illustrative purposes, a set of twelve synthetic signals distributed in four groups between linear and exponential are tested. Signals have been contaminated with random white noise as seen in figure 4a. These signals where processed with our algorithm and the resulting classification is shown in figure 4b in which all the found hyperboxes can be seen. Figure 4b axes correspond to the normalized values of TC (c_2, c_1, c_0). In this figure, one of the advantages of the presented method is clearly identified: In this particular case if only current value of the signals is considered, signals of groups two and three, plotted in blue and red colours in fig 4a, appears to be the closest ones. Nevertheless, using episode representation these two groups of signals can be easily separated in this feature space (red and blue boxes in fig. 4b).

4. DIAGNOSIS CASE STUDY

4.1 The CSTH

The CSTH is a benchmark developed by [Thornhill et al., 2008] of a stirred tank in which hot (50°C) and cold (24°C) water are mixed and further heated using steam; the final mix is then drained using a long pipe. The CSTH configuration is shown in figure 5. It is assumed that the tank is well mixed so the temperature of the outflow is the same as that in the tank. System inputs are setpoints for the cold water, hot water and steam valves. System outputs are hot and cold water flow, tank level and temperature. System inputs and outputs represent electronic signals in the range 4âĂŞ-20 mA.

4.2 Dynamic classification on the CSTH

Thornhill et al. [2008] suggests two operation points described in table 2. Simulink models for $OP1$ and $OP2$ are

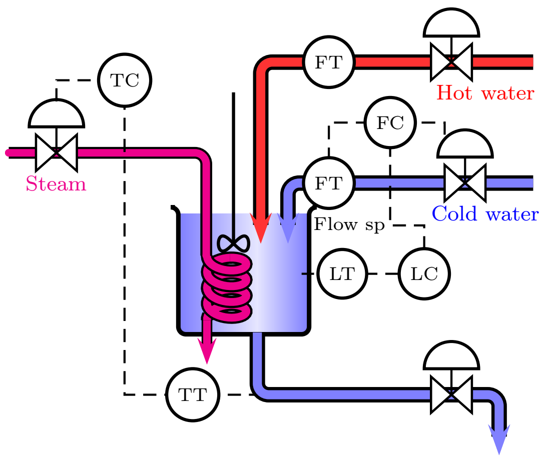


Fig. 5. The continuous stirred tank heater

Parameter	Value	Parameter	Value
β	2.0	λ	0.2
ϕ	0.05	α	0.2
win_{length}	128	min win_{length}	64
φ	1	t_off	300 sec
ρ	2.0	ζ	1.6

Table 3. Parameters used in the experiment

available at [Thornhill] website, with and without disturbances. The provided disturbances are real data sequences experimentally measured from the pilot plant at the University of Alberta. Several states were simulated using the suggested operational points $OP1$ and $OP2$. Undisturbed operation, disturbed operation and tank leakage scenarios were simulated in Simulink and the resulting data were fed to the dynamic classification algorithm. The following experiments will use (unless otherwise stated) the user defined parameters and values shown in table 3. Window length and minimal length were selected according to process dynamics. Parameters β and λ (from ageing function) were chosen with respect to the process dynamics, so that the data that weight more on the clustering are in a time window consistent with the process evolution. Parameter ϕ (micro-cluster size) was tuned to obtain the clustering that best tracks process dynamics (Note that bigger micro-cluster means less precision in the representation of dynamics). Parameter α (threshold for a micro-cluster to be considered dense) was tuned to discard at best outliers resulting from noise. As mentioned in section 2.3 parameters ρ, ζ and also t_off are set according to the amount of memory available for snapshot storage and display.

Since the features dimension space is high $(12 - d$ space) and therefore not directly plottable, the three features with larger variance were selected as axes for a $3D$ plot. Features variance were calculated for each of k μ-clusters in an incremented fashion using equation 14. $3D$ plot axes show the normalized features.

$$Var_k^{(t)} = \frac{1}{n_k^{(t)}} SS_k^{(t)} - \left(\frac{LS_k^{(t)}}{n_k^{(t)}} \right)^2 \qquad (14)$$

In pursuance of testing robustness against disturbances, the algorithm was fed with non disturbed data of $OP1$ for 2000 seconds. In $t = 2001$ the provided disturbances

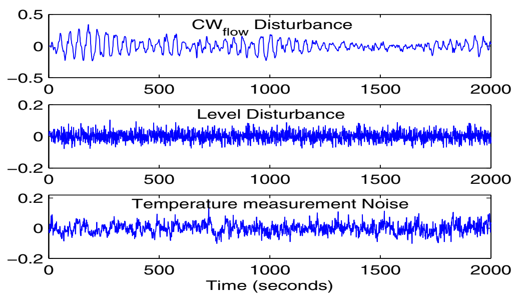


Fig. 6. Deviation in output measures (provided by [Thornhill]): CW_{flow} disturbance, level disturbance due bubbles and temperature measurement noise.

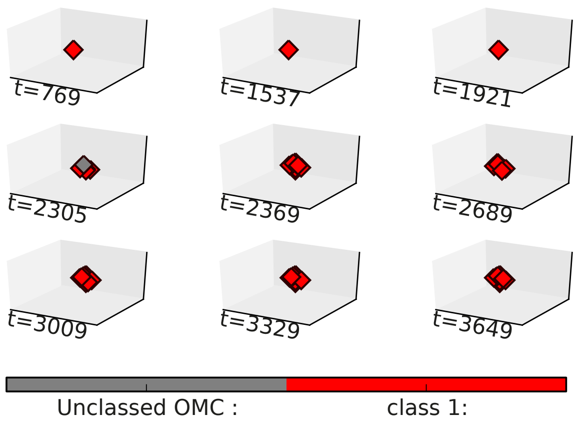


Fig. 7. Dynamic Classification of the CSTH in op1. Change from a non disturbed environment to a disturbed one.

in the tank level and CW flow were added as well as the temperature measurement noise. These disturbances can be seen in figure 6. The resulting classification is shown in figure 7.

Snapshots in figure 7 with $t < 2000$ show one class with one box only. This behaviour is in conformity with uniform data. After disturbances are introduced, an $O\mu$-cluster is detected in snapshot $t = 2305$ showing the detection of a different behaviour. As soon as the amount of disturbed data increases, the algorithm is able to detect that this apparently new behaviour in no more than an evolution of the known normal $OP1$ behaviour, as can be shown in snapshots with $t > 2305$ where all the clusters are labelled as belonging to class 1.

In order to simulate fault dynamical behaviour, an increasing size leak was simulated. Starting from the plant $OP1$ a leak was introduced in $t = 2200$ seconds. The starting radius of the hole is $r = 1mm$. The experiment emulate the case were the size of the hole leaking the mix increases as time passes. The final radius of the hole at $t = 3400$ seconds is $r = 3.75mm$. The measured output signals with disturbances can be seen in figure 8. These signals were take as input for the proposed dynamic classification algorithm and some of the clusters snapshots can be seen in figure 9. Class distribution shows how class one, associated with normal behaviour, is detected and its evolution is followed until $t = 2881$ where some new behaviour is found. This behaviour is not yet classified as a new class since the amount of data following that pattern is low. Appearing in the snapshot $t = 3649$ the μ-clusters have

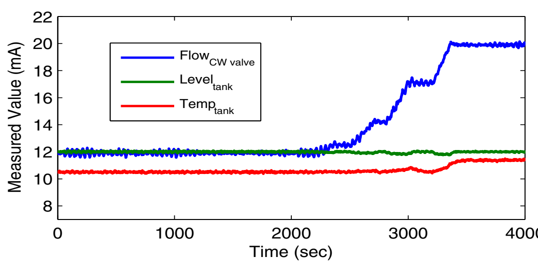


Fig. 8. Leak evolution simulation with the CSTH in op1

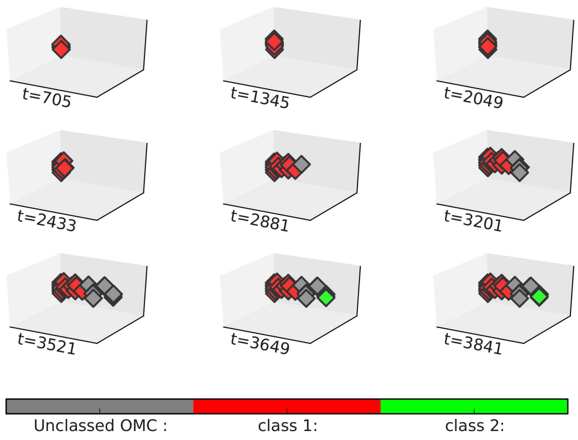


Fig. 9. Dynamic Classification of the CSTH in op1

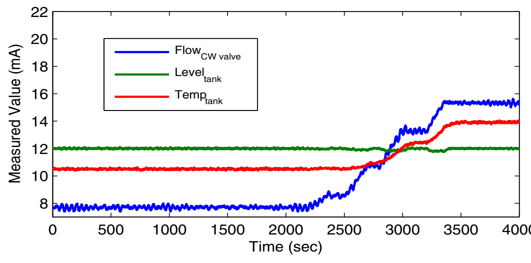


Fig. 10. Leak evolution simulation with the CSTH in op2

evolved into dense μ-clusters and the new class can be formally established.

The same experiment were performed for the $OP2$ and the measured signals are shown in figure 10. The classification snapshots can be seen in figure 11. An interesting behaviour can be appreciated by looking in detail the change between snapshots $t = 705$ and $t = 1537$. The former show μ-clusters belonging to classes one and two as well as some low density $O\mu$-clusters. The later shows how class 1 and class 2 μ-clusters were merged into class 1, when some new μ-clusters emerged between them. Here the data evolution is reflected in the clusters structure with the merge of classes. Snapshots with $t < 2241$ exhibit the expected behaviour when all data is in $OP - 2$. In snapshot $t = 2625$ some $O\mu$-clusters are present showing the detection of new, evolving behaviour. The same trend was observed in snapshots $t = 3009$ and $t = 3329$ proving a continuing, and rapid, evolution. Once this new behaviour is established, it is identified as such and a class 2 label is assign to the elements exhibiting the same trend.

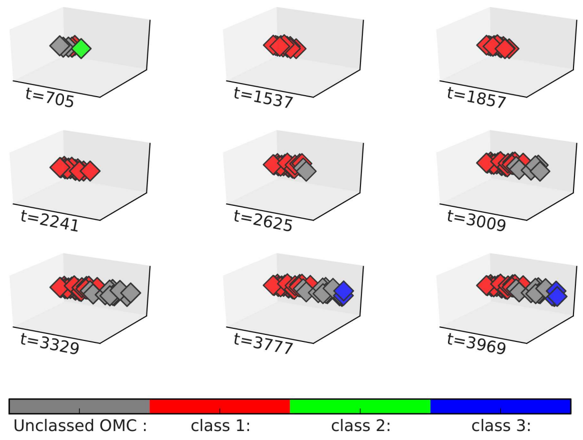


Fig. 11. Dynamic Classification of the CSTH in op2

Functionality		TBDC	TBDC-A	TBDC-AD	TBDC-T
Computation Time	average	1.00	1.16	0.99	1.26
	max	1.00	1.11	0.96	1.20
	min	1.00	1.16	1.00	1.30
No. of classes		2	2	1	2
No. μ-clusters		1.00	1.20	1.20	1.34
outlier detection		yes	yes	no	yes

Table 4. Algorithm performance with and without some functionalities

4.3 Performance analysis

An empirical study was performed to find the improvements achieved by this algorithm with the described functionalities. 300 evolving leak scenarios were run with and without some of our Trend Based Dynamic Clusterer (TBDC) functionalities. Computation time, clustering precision and outlier detection were compared removing ageing functionality (eq. 2-5) (TBDC-A), removing density analysis (clusters are formed only upon micro-clusters proximity) at the second stage of the algorithm (TBDC-D), removing both ageing and density and finally, removing episode characterization (classifying directly time series data, no trending TBDC-T). Normalized performances measures can be seen in table 4 (normalized w.r.t. the performance of TBDC).

Test have shown that no ageing implies more μ-clusters to be processed (20% more) and bad evolution of the clustering. Clusters appears to extend but not to drift. Removing the density-based analysis implies the creation of more clusters, but some of them are almost empty. Potentially this could be caused by the fact that outliers are not discarded. Arguably, the clustering algorithm without both ageing and density functionalities seems to have a slightly better computation time. Nevertheless, it fails to detect the faulty behaviour. Finally, test without trending have proved the computation time reduction achieved with the episode characterization. No trending implies much higher computation time (26.5% average) and worse real-time processing, this seems to be related to the increasing number of μ-clusters to be treated (34% more).

5. CONCLUSIONS

In this paper, we have presented a proposal which couples a dynamic clustering method based on micro-clusters with an on-line trend extraction algorithm that works incrementally on the incoming data. The proposed algorithm is used for on-line data analysis and diagnosis and has been tested on a continuous stirred tank heater benchmark affected by varying magnitude faults. The algorithm show good performance in presence of disturbances and the results follow the evolution of the system. For instance, in the case of a leakage which increases with time, the classifier recognizes the drift and creates a new class representing the fault, and then gradually transfers data into this class as the leak increases. Test have shown performance improvement when distance-based and density-based clustering techniques are used together. Furthermore, by using episode representation and ageing capabilities the amount of data to be processed is reduced, hence improving computation time and reducing memory requirements. However, more tests are required to assess the efficiency of the algorithm in real-time conditions.

From a technical point of view, it would be interesting to compare the proposed algorithm, which is density-based in the second stage, with a distance-based algorithm. Future work also includes to provide a feature selection algorithm targeting the dynamic classifier and able to rank the available features according to their discriminability power.

ACKNOWLEDGEMENTS

This work was partially supported by the Colombian administrative department of science, technology and innovation COLCIENCIAS.

REFERENCES

Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment, 2003a.

Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment, 2003b.

P. Angelov. Fuzzily connected multimodel systems evolving autonomously from data streams. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 41(4):898–910, 2011.

P. Angelov and X. Zhou. Evolving fuzzy-rule-based classifiers from data streams. *Fuzzy Systems, IEEE Transactions on*, 16(6):1462–1475, 2008.

Bhavik R. Bakshi. Multiscale analysis and modeling using wavelets. *Journal of Chemometrics*, 13(3-4): 415–434, 1999. ISSN 1099-128X. doi: 10.1002/(SICI)1099-128X(199905/08)13:3/4<415::AID-CEM544>3.0.CO;2-8.

A. Bouchachia and C. Vanaret. Incremental learning based on growing gaussian mixture models. In *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, volume 2, pages 47–52. IEEE, Elsevier, 2011.

Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, 2006.

Yixin Chen and Li Tu. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and data mining*, pages 133–142, 2007.

Sourabh Dash, Mano Ram Maurya, Venkat Venkatasubramanian, and Raghunathan Rengaswamy. A novel interval-halving framework for automated identification of process trends. *AIChE journal*, 50(1):149–162, 2004.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databdata with noise. In *KDD*, 1996.

A. Joentgen, L. Mikenina, R. Weber, and H.J. Zimmermann. Dynamic fuzzy data analysis based on similarity between functions. *Fuzzy Sets and Systems*, 105(1):81–90, 1999.

T. Kempowsky, A. Subias, and J. Aguilar-Martin. Process situation assessment: From a fuzzy partition to a finite state machine. *Engineering Applications of Artificial Intelligence*, 19(5):461–477, 2006.

Philipp Kranen, Ira Assent, Corinna Baldauf, and Thomas Seidl. The ClusTree: indexing micro-clusters for any stream mining. *Knoledge and information systems*, 29 (2):249–272, 2011.

K. Li, F. Yao, and R. Liu. An online clustering algorithm. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, volume 2, pages 1104–1108. IEEE, 2011.

Mano Ram Maurya, Praveen K Paritosh, Raghunathan Rengaswamy, and Venkat Venkatasubramanian. A framework for on-line trend extraction and fault diagnosis. *Engineering Applications of Artificial Intelligence*, 23(6):950–960, 2010.

Marshall H Stone. The generalized weierstrass approximation theorem. *Mathematics Magazine*, 21(5):237–254, 1948.

Nina Thornhill. The csth web site. URL `http://personal-pages.ps.ic.ac.uk/~nina/CSTHSimulation/index.htm`. Accessed: 2014-07-06.

Nina F Thornhill, Sachin C Patwardhan, and Sirish L Shah. A continuous stirred tank heater simulation model with applications. *Journal of Process Control*, 18(3): 347–360, 2008.

V. Venkatasubramanian, R. Rengaswamy, S.N. Kavuri, and K. Yin. A review of process fault detection and diagnosis. Part III: Process history based methods. *Computers & Chemical Engineering*, 27(3):327–346, 2003.

H.J. Zimmermann. Dynamic fuzzy data analysis and uncertainty modeling in engineering. In *Intelligent Techniques and Soft Computing in Nuclear Science and Engineering*, pages 3–15. World Scientific, 2000. Proceedings of the 4th International FLINS Conference Bruges, Belgium 28–30 August.