Just-In-Time Classifiers for Recurrent Concepts

Cesare Alippi, *Fellow, IEEE*, Giacomo Boracchi, Manuel Roveri

***Abstract*—Just-In-Time (JIT) classifiers operate in evolving environments by classifying instances and reacting to concept drift. In stationary conditions, a JIT classifier improves its accuracy over time by exploiting additional supervised information coming from the field. Differently, in nonstationary conditions, the classifier reacts as soon as concept drift is detected: the current classification setup is discarded and a suitable one activated to keep the accuracy high.**

We present a novel generation of JIT classifiers able to deal with recurrent concept drift by means of a practical formalization of the concept representation and the definition of a set of operators working on such representations. The concept-drift detection activity, which is crucial to promptly react to changes exactly when needed, has been advanced by considering change-detection tests monitoring both inputs and classes distributions.

***Index Terms*—Concept Drift, Adaptive Classifiers, Just-in-time Classifiers, Recurrent Concepts.**

I. Introduction

MOST machine learning techniques assume, either explicitly or implicitly, that the process generating the data is stationary. This assumption guarantees that the model learnt during the initial training phase remains valid over time and that its performance is in line with our expectations. Unfortunately, this assumption does not truly hold in the real world representing, in many cases, a simplistic approximation of the reality. For instance, it is clear that transient or permanent faults affecting the electronics of an embedded system as well as software bugs, influence the performance of algorithms introducing, de facto, nonstationary phenomena. Similarly, thermal drift or ageing effects may affect the transduction mechanism of sensors, the conditioning electronics, or the ADC, hence degrading (slowly) the quality of acquired measurements: again the resulting effect is a concept drift affecting the data-generating process. At the same time, we might consider changes influencing features extracted from data instead of inspecting changes at the data level. The parameters of a linear dynamic model approximating the process are an example of features; here, time-variance changes the probability density function (pdf) in the parameter space. Whatever the cause is, in time-varying situations the statistical distribution of data changes over time, hence impairing the validity of the stationary hypothesis.

Classifiers designed to address concept drift cope with situations where the pdf may change over time. When concept drift occurs, the accuracy of the classifier drops since the current classification setup becomes obsolete and must be substituted with an up-to-date one. The problem of classification in environments affected by concept drift can be found in many application domains. Relevant applications are fraud detection in electronic transactions, health care systems, sensor networks, intelligent vehicles and recommender systems to name the few [1]. In addition, dealing with concept drift may improve classification systems in applications involving information filtering (such as email and spam filtering), meteorology statistics, Internet event logs analysis, stock market transaction forecasting, and context-aware and ubiquitous computing.

The JIT classifier, which has been originally presented in [2], [3], natively implements the detection/adaptation paradigm. As long as the world is stationary, a JIT classifier exploits any supervised information coming from the field to improve the classification accuracy. As a distinctive characteristic, any consistent classifier employed in a JIT framework tends towards the Bayes one, even though the process undergoes an abrupt concept drift.

JIT classifiers monitor the occurrence of concept drift by means of Change-Detection Tests (CDTs) that, sequentially, assess the stationarity of the data-generating process. So far, the proposed JIT classifiers employ CDTs for monitoring the distribution of input data [2], [3], [4], disregarding the information associated with supervised labels. However, monitoring the input data does not allow us for detecting concept drift modifying the way labels are assigned. Furthermore, existing JIT classifiers do not take advantage of recurrent concepts.

This paper advances the works presented in [4] and [5] by proposing a general formulation of the JIT approach that introduces an explicit adaptive management of concepts. As soon as a CDT detects a change (*concept drift detection*), a new concept representation is created (*concept isolation*) and compared with the stored representations of concepts (*recurrent concept identification*). When the representation of a previously encountered concept is considered equivalent to the current one, its supervised samples are used to reconfigure the classifier. As such, the JIT classifier takes advantage of recurrent concepts, e.g., that may arise when the process operates in a sequence of working states or undergoes a cyclic drift.

The original contributions of the paper are:

- a general formulation for JIT classifiers that embodies an explicit management of concepts;
- a technique for isolating concepts;
- a procedure for identifying recurrent concepts by assessing the equivalence between two concept representations;
- a specific JIT classifier able to detect concept drift affecting both the input data distribution and the classification

The authors are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy e-mail: firstname.lastname@polimi.it

(c) 2013 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works

error.

The paper is organized as follows. The analysis of related works is presented in Section II. Section III formalizes the classification problem in environments affected by concept drift. Section IV describes the general methodology for designing classifiers following the JIT approach; Section V details the proposed JIT classifier. Finally, Section VI presents and discusses the experimental results.

II. Related Works

Several classifiers able to deal with evolving and possibly recurrent environments have been proposed in the literature [6]–[8]. In the following, we critically describe the most relevant works from two different points of view: the concept-drift detection and the ability to identify recurrent concepts.

In the former case, classifiers dealing with evolving environments can be grouped into two families according to the way they react to concept drift. Active classifiers [2]–[5], [9]–[21] rely on triggering mechanisms (e.g., a CDT) to detect when the classifier is no more representative of the current concept: an adaptation phase is then activated once a change is detected to train the classifier on the current concept. On the contrary, in passive solutions (refer to [6], [7] and references therein), the classifier undergoes a continuous update every time new supervised samples are available. Passive classifiers generally rely on an ensemble of classifiers with adaptation confined in the update of the weights used in the fusion/aggregation rule and creation/removal of classifiers composing the ensemble.

[20] suggests an active classifier that monitors nonstationarity by inspecting variations in the mean value of a sliding window opened over input data. Differently, [19] takes decisions by inspecting the normalized Kolmogorov-Smirnov distance between the cumulative density functions estimated on the training samples and a window of the most recent ones.

[15]–[18], [21] present triggering mechanisms based on the classification error. In more detail, [15], [17], [21] detect a change when the classification error exceeds a fixed threshold (which is tuned according to the sample standard deviation of the associated Bernoulli distribution). [18] suggests an ad-hoc statistical test on the proportion of incorrectly classified samples to compare two different partitions of supervised couples. [16] introduces an active classifier for concept drift that relies on a sequential CDT (the Bernoulli exponential moving average chart) to assess the stationarity of the classification error over time.

Both active and passive classifiers can be endowed with mechanisms to identify and reactivate recurrent concepts. The pioneering work of [22], which introduced the FLORA 3 and FLORA 4 algorithms, relies on windowing mechanisms for adapting to concept drift and identifying potential recurrent concepts. A concept history is kept and built in [23], which allows the classifier for creating and reactivating past concepts when they reoccur. There, recurrent concepts are identified by measuring the *conceptual equivalence* obtained by contrasting the outputs of two classifiers (the historical and the updated ones) over the most recent observations. Concept drift in [24] is handled by an instance-selection mechanism that identifies which observations fit the current concept descriptor. The classifier also predicts the rate of change and possibly recalls previously encountered concepts. In [25] a procedure for building an ensemble of classifiers is presented: classifiers are selected from the history of all the trained classifiers depending on their classification accuracy evaluated over the most recent supervised samples. [26] introduced an effective algorithm that relies on conceptual vectors, i.e., descriptors extracted from batches of past supervised data, to build an ensemble of classifiers. A clustering algorithm, applied to conceptual vectors, selects the best classifier from the ensemble.

The identification of recurrent concepts has been also addressed in the literature of context-sensitive learning, where it is assumed that additional information about the current context is available (which can be exploited for identifying concept drift as well as recurrent concepts). The issue of identifying contextual features within the observations is addressed in [27], while [1], [28]–[30] focus on how to integrate external contextual information within an ensemble of classifiers with the specific purpose of exploiting recurrent concepts. Contextual features are typically exploited at meta-learning level and are not processed as attributes of observations.

In all above solutions the equivalence between two concepts is assessed by means of supervised samples, without taking advantage of unlabeled observations. In contrast, [31], [32] introduce a two-layered learning scheme that relies on a single classifier predicting labels at the first layer and a meta classifier (which is trained to learn the regions of the input domain where the first layer correctly operates) at the second layer. After each concept-drift detection, the stored meta-classifiers are compared pairwise by evaluating their performance on the most recent unsupervised samples to determine whether the new concept has been already encountered in the past or not. Recently, [33] introduced a classifier able to identify recurrent concepts by exploiting unsupervised samples. However, this solution is classifier-dependent, as it is tailored to an incremental decision tree. Still, these classifiers neither analyze the distribution of unlabeled observations to detect concept drift nor identify recurrent concepts.

Monitoring sequentially the stationarity of the input pdf is one of the peculiarities of the first JIT classifiers [3], [14]. More specifically, JIT classifiers applied the CI-CUSUM CDT [2], whereas the most recent versions have enforced the ICI-based one [4], [11] as the test provides improved detection ability in rapidly evolving environments. Unfortunately, a CDT on the input cannot detect any concept drift that leaves the input distribution unaltered, even when this determines a dramatic fall in the classifiers accuracy (e.g., the swap of the classes). Furthermore, it is difficult to apply a CDT to inputs containing qualitative components. To solve these problems, [5] introduced a JIT classifier able to detect concept drift affecting the accuracy of classifiers.

Here, we propose a JIT classifier that detects concept drift by means of two CDTs monitoring the distribution of input data and the classification error. For the first time in the literature we use both the input data and the supervised couples to detect concept drift and identify and compare concepts. This guarantees a very high adaptiveness to concept drift thanks to an improved detection ability. At the same time the method reduces false positive detections and simplifies the configuration phase in the JIT classifier. Differently from the approaches present in the literature that tailor the concept-drift detection ability to a specific classifier, the proposed solution is very flexible and any classification scheme can be employed within the JIT framework. As an example, in the experimental section, we customize the JIT approach to integrate k-nearest neighbor, naïve Bayes and support vector machine as classifiers.

III. Problem Formulation

Let us consider a classification problem where sequential i.i.d. couples (x_t, y_t) are generated according to an unknown pdf. In particular, let $x_t \in \mathbb{R}^d$ be the input at time t, generated by the unknown process X and y_t be its class label, belonging to a finite set Λ. The pdf of inputs at time t can be expressed as

$$p(x|t) = \sum_{y\in\Lambda} p(y|t)p(x|y,t), \quad \text{such that} \quad \sum_{y\in\Lambda} p(y|t) = 1, \tag{1}$$

where $p(y|t) > 0$, is the probability of receiving a sample of class $y \in \Lambda$, while $p(x|y,t)$ is the class conditional distribution at time t. Both $p(y|t)$ and $p(x|y,t)$ are unknown and may change following a concept drift. The training sequence is composed of the first T_0 observations (either input data or supervised couples) assumed to be generated in stationary conditions, i.e., $p(y|t)$ and $p(x|y,t)$ do not change for $t \in [0, T_0]$ and $\forall y \in \Lambda$. Beside stationarity, we assume that the training sequence contains also supervised pairs, i.e., the true label y_t is made available for some x_t, $t \in [0, T_0]$. No assumptions are made on how often supervised pairs (x_t, y_t) are provided during the operational life ($t > T_0$), as these could be received following a regular time-pattern scheme (e.g., one supervised sample out of m) or asynchronously.

IV. JIT Classifiers for Recurrent Concepts: the General Formulation

The key elements composing a JIT classifier are the set of concept representations $\mathcal{C} = \{C_1, \dots, C_N\}$ and the set of *operators* $\{\mathcal{U}, \Upsilon, \mathcal{D}, \mathcal{E}, K\}$ designed to handle such representations.

The i-th concept representation is defined as the triplet $C_i = \big(Z_i, F_i, D_i\big)$ where Z_i is a sequence of supervised couples, F_i is a sequence of features characterizing the i-th concept and used to assess the equivalence between two concept representations, and D_i is a sequence of features used to inspect changes in the i-th concept. Not rarely, D_i also contains features F_i. Examples of D_i are the cumulative statistics used by CUSUM-based CDTs [34]; examples of F_i are the sample statistics derived from non-overlapping subsequences of observations.

The operators are defined as follows:

- the *update* operator $\mathcal{U}(C_i, R) \rightarrow C_i$. Operator $\mathcal{U}$ receives concept C_i and R, a sequence of recent supervised couples (or inputs). $\mathcal{U}$ modifies the concept representation C_i by appending recent supervised samples from R to Z_i and features extracted from R to F_i.
- the *split* operator $\Upsilon(C_i) \rightarrow (C_j, C_k)$. The operator Υ divides a concept representation C_i into two disjoint concept representations C_j and C_k. Elements that cannot be safely associated either to C_j or C_k are discarded.
- the *concept-drift detection* operator $\mathcal{D}(C_i) \rightarrow \{0, 1\}$. $\mathcal{D}$ sequentially assesses the stationarity of concept C_i by monitoring features in D_i. When $\mathcal{D}(C_i) = 0$ all observations yielding C_i are considered from the same concept, i.e., “no concept drift” has occurred. When $\mathcal{D}(C_i) = 1$, “concept drift has been detected”.
- the *equivalence* operator $\mathcal{E}(C_i, C_j) \rightarrow \{0, 1\}$. $\mathcal{E}$ checks if C_i and C_j are equivalent: $\mathcal{E}(C_i, C_j) = 1$ means that C_i and C_j are two representations coming from the same concept. $\mathcal{E}(C_i, C_j) = 0$ means that C_i and C_j are representations of different concepts.
- the *classifier* $K(Z, x_t) \rightarrow \Lambda$. K is configured on Z and assigns label $\widehat{y}_t \in \Lambda$ to input sample x_t.

```
1-  Build concept $C_0 = \big(Z_0, F_0, D_0\big)$ from the initial
    training sequence;
2-  $Z_{\text{rec}} = \emptyset$ and $i = 0$;
3-  while ($x_t$ is available) do
4-      $\mathcal{U}(C_i, \{x_t\}) \rightarrow C_i$;
5-      if ($y_t$ is available) then
6-          $\mathcal{U}(C_i, \{(x_t, y_t)\}) \rightarrow C_i$;
        end
7-      if ($\mathcal{D}(C_i) = 1$) then
8-          $i = i + 1$;
9-          $\Upsilon(C_{i-1}) \rightarrow (C_k, C_l)$;
10-         $C_i = C_l$;
11-         $C_{i-1} = C_k$;
12-         $Z_{\text{rec}} = \bigcup_{\substack{\mathcal{E}(C_i, C_j)=1 \\ 0\le j<i}} Z_j$;
        end
13-     if ($y_t$ is not available) then
14-         $\widehat{y}_t = K(Z_i \cup Z_{\text{rec}}, x_t)$.
        end
    end
```
Algorithm 1: General formulation of the JIT classifier for recurrent concepts.

A general description of the JIT classifier is presented in Algorithm 1. The JIT classifier requires an initial training sequence to build $C_0 = \big(Z_0, F_0, D_0\big)$, a representation of the *initial stationary* concept (line 1). The initially-empty set Z_{rec} stores supervised couples coming from previous occurrences of the current concept.

During the operational life, the JIT classifier receives input x_t or supervised couples (x_t, y_t) (line 3) that are used to update C_i –the representation of the current concept– by means of the *update* operator $\mathcal{U}$. This operator extracts features from inputs and appends them to F_i (line 4) and inserts supervised couples into Z_i (line 6).

The concept-drift detection operator $\mathcal{D}$ (line 7) monitors the possible occurrence of concept drift by computing and inspecting features stored in D_i. As soon as a concept drift is detected, a new representation of the concept is created from the most recent inputs and supervised couples (lines 8 - 11). The split operator Υ then separates supervised samples and features belonging to the new concept C_i from those belonging to a previous concept C_{i-1}. Supervised couples and features that cannot be safely associated either to C_i or C_{i-1} are discarded.

After each concept-drift detection, the JIT classifier checks if the actual concept is recurrent, i.e., if the concept drift has moved the process into an already encountered state. To carry out this operation, the concept representation C_i is pairwise compared with each C_j in $\mathcal{C} = \{C_j,\ 0 \leq j < i\}$ by using the equivalence operator $\mathcal{E}$. When C_i and C_j are considered equivalent (i.e., $\mathcal{E}(C_i, C_j) = 1$), the supervised samples Z_j of C_j are inserted into Z_{rec}, the set collecting the supervised couples of concept representations coherent with C_i (line 12). After each concept-drift detection, the operator $\mathcal{D}$ is reconfigured to detect further concept drift that might affect the new state of X.

Finally, the *classifier* K, which exploits both fresh supervised couples in Z_i and supervised couples from recurrent concepts stored in Z_{rec}, assigns label $\widehat{y}_t$ to input x_t (line 14).

V. JIT Classifiers for Recurrent Concepts: a Realization

The section instances a JIT classifier for recurrent concept satisfying the general framework of Algorithm 1. In the following the i-th concept representation C_i is defined as

- Z_i, the set of available supervised couples.
- F_i, the features characterizing the i-th concept are extracted from non-overlapping subsequences Q_X containing ν_X observations (no supervised labels are here requested). These features represent condensed information about the data-generating process and, together with supervised samples, are essential to isolate and identify recurrent concepts. The first feature in F_i is the sample mean M, computed from the s-th subsequence
$$M(s) = \frac{1}{\nu_X} \sum_{t=(s-1)\nu_X+1}^{\nu_X s} x_t; \tag{2}$$
the second feature is a power-law transformation of the sample variance, i.e.,
$$\begin{aligned} V(s) &= \left(\frac{S(s)}{\nu_X - 1}\right)^{h_0}, \text{being} \\ S(s) &= \sum_{t=(s-1)\nu_X+1}^{\nu_X s} (x_t - M(s))^2. \end{aligned} \tag{3}$$
This power-law transform yields values of V approximatively Gaussian distributed; its exponent h_0 is computed as
$$h_0 = 1 - (\kappa_1 \kappa_3)/3\kappa_2^2,$$
where κ_i is the i-th cumulant of the distribution of the sample variance. In practice, these cumulants are computed from the values of S on the training sequence (refer to [35] and [4] for additional details). The features pair (M, V) provides a second-order approximation of the input distribution: for this reason the proposed concept representation encompasses them and the split Υ and equivalence $\mathcal{E}$ operators take them into account to partition and compare concept representations.
- D_i, the features used for detecting concept drift by means of CDTs. Features $D_i = \big(F_i, P_i\big)$ include F_i since we adopt the ICI-based CDT [4] that detects concept drift by monitoring nonstationarities in $M(s)$ and $V(s)$. The sequence P_i contains the estimates of the classification error computed over non-overlapping subsequences Q_ϵs, each containing ν_ϵ supervised samples in Z_i, i.e.,
$$P_i = \Big\{\frac{1}{\nu_\epsilon} \sum_{t \in Q_\epsilon} \epsilon_t,\ Q_\epsilon \subset Z_i\Big\}, \tag{4}$$
where the element-wise error ϵ_t is defined as
$$\epsilon_t = \begin{cases} 0, & \text{if } y_t = K_0(x_t) \\ 1, & \text{otherwise} \end{cases}; \tag{5}$$
K_0 is a classifier used specifically for change-detection purposes and $K_0(x_t)$ is the label assigned to x_t. Nonstationarities in P_i are detected by the CDT proposed in [5]. Thus, the proposed JIT classifier detects concept drift as a violation in stationarity of either F_i or P_i.

A. Updating a Concept: the Update Operator $\mathcal{U}$

$\mathcal{U}$ updates the current representation C_i by inserting a supervised couple or extracting features from a sequence of input data. In particular, the update operator applied to a supervised pair $\{(x_t, y_t)\}$ is defined as
$$C_i = \mathcal{U}(C_i, \{(x_t, y_t)\}) = \begin{cases} Z_i = Z_i \cup (x_t, y_t) \\ F_i = F_i \end{cases}, \tag{6}$$
while the update operator applied to a sequence of ν_X input data $\{x_t\}_{(s-1)\nu_X+1}^{s\nu_X}$ is
$$C_i = \mathcal{U}(C_i, \{x_t\}_{(s-1)\nu_X+1}^{s\nu_X}) = \begin{cases} Z_i = Z_i \\ F_i = F_i \cup [M(s); V(s)] \end{cases}, \tag{7}$$
where $[M(s); V(s)]$ is the column vector obtained by stacking the respective $M(s)$ and $V(s)$ instances.

B. Detecting Concept Drift: the Concept Drift Operator $\mathcal{D}$

The concept-drift detection operator $\mathcal{D}$ consists in two CDTs exploiting the intersection of confidence intervals (ICI) rule [36], [37] as a core technique for detecting changes in stationarity. ICI-based CDTs [4] are sequential CDTs meant for scalar data streams. At first, inputs are partitioned into non-overlapping subsequences where features are extracted, then, the ICI-rule assesses, on-line and sequentially, the stationarity of the expected value of each feature. Remarkably, these CDTs come with a *refinement procedure* (Algorithm 3 in [4]) that isolates, after each detection, observations (both input data and supervised couples) that have been generated by the process in the new state; this information can be used to reconfigure the CDT after each concept-drift detection. This characteristic, coupled with the reduced computational complexity, makes the ICI-based CDT the ideal candidate to be employed within a JIT framework.

The concept drift operator $\mathcal{D}$ is defined as

$$\mathcal{D}(C_i) = (\text{CDT}_X(F_i) || \text{CDT}_\epsilon(P_i)) \in \{0,1\}, \tag{8}$$

where $||$ is the OR-logical operator; $\text{CDT}_X(F_i) = 1$ and $\text{CDT}_\epsilon(P_i) = 1$ imply that a concept drift has been detected in the stream of features F_i and in the estimates of the classification error P_i, respectively. Any new feature value, as well as any new estimate of the classification error of K_0 is included in D_i, as these will be used for reconfiguring the CDTs after a detection. In particular, CDT_X monitors the distribution of input data disregarding possible existing labels, while CDT_ϵ verifies whether the average classification error in P_i is constant or not. In what follows we detail the CDT_X and CDT_ϵ tests.

***CDT*$_{\mathbf{X}}$:** CDT_X assesses the stationarity of the process by inspecting each of the features in F_i for changes. A concept drift is detected as soon as the ICI-rule identifies a nonstationarity either in M or V. However, the CDT structurally introduces false positives, i.e., detections that do not correspond to concept drift. A hierarchical formulation of the ICI-based CDT has been designed in [10] to keep under control the occurrence of false positives. The two-level architecture consists in CDT_X at the lower level and a change-validation procedure at the upper level which, in addition to the assessment of the change, provides a confidence level. The hierarchical CDT exploits $T_{\text{ref},i}$ – a refined estimate of the change time-instant T_i^* – provided by the refinement procedure. Let $\widehat{T_i}$ be the time instant in which CDT_X determines a drift in features F_i. The change validation is formulated as a multivariate hypothesis test based on two sequences of features: those extracted from the time interval $[T_{\text{ref},i}, \widehat{T_i}]$, characterizing the new candidate concept and those extracted in interval $[T_{\text{ref},i-1}, \widehat{T}_{i-1}]$, referring to the previous concept (i.e., the one in which CDT_X was configured). The procedure is as follows: stack in column vectors the features extracted in these time intervals[1] (each feature representing a row) and compute their means $\overline{F}_i$ and $\overline{F}_{i-1}$ and the pooled sample covariance matrix. The null hypothesis H_0 is formulated as

$$H_0 : \text{“}\overline{F}_i - \overline{F}_{i-1} = \underline{0}\text{”}, \tag{9}$$

where $\underline{0}$ represents the two-dimensional vector of null components. Then, an Hotelling T^2 test [38] can be executed to reject the null hypothesis at a predefined confidence level α. When the null hypothesis is not rejected, the CDT_X parameters are reconfigured from the original state (i.e., features in the $[T_{\text{ref},i-1}, \widehat{T}_{i-1}]$ interval). Otherwise, the change is confirmed. We consider the change-validation step as a part of the CDT (Algorithm 2, line 7) and, for this reason, it is not expressively reported in Algorithm 2. It also has to be mentioned that, although the presented solution is meant for features coming from CDT_X, an ad-hoc validation procedure can be in principle implemented for different features.

CDT_X is initially configured on a training sequence (generated in stationary conditions) to estimate h_0 in (3), the expected value, and the confidence interval for each feature. In order to reliably reconfigure CDT_X we require at least M_X subsequences (i.e., $\nu_X M_X$ observations) and, whenever the input data received within the $[T_{\text{ref},i}, \widehat{T_i}]$ interval are not enough, the reconfiguration of the CDT is postponed (and the JIT classifier temporarily operates without concept-drift detection abilities). Further details regarding CDT_X and its hierarchical formulation can be found in [4] and [10], respectively.

***CDT*$_\epsilon$:** There are two main reasons justifying the importance of monitoring the evolution of the classification error over time. At first, the analysis of the classification error allow us to detect concept drift affecting $p(y|x)$. Second, monitoring the classification error allows the JIT classifier to react to concept drift when it directly influences its accuracy. As such, it is particularly useful to consider also CDT_ϵ to monitor the classification error, since CDT_X cannot perceive changes – even dramatic– that leave $p(x)$ unchanged, e.g., a classes' swap. It has to be mentioned that detection performance of CDT_ϵ strongly depends on the availability of supervised information.

The proposed CDT_ϵ consists in a customized ICI-based CDT for monitoring the stationarity of P_i, defined as in (4). Then, we need to configure an additional classifier K_0, which is used to compute the average classification error over non-overlapping subsequences of supervised samples, and it is never updated.

The element-wise classification error in (5) can be modeled as a Bernoulli random variable having expectation p_0. In stationary conditions, p_0 is constant since K_0 is not updated and, since x_t are independent, ϵ_ts are also independent. The sum of ν_ϵ i.i.d. Bernoulli random variables follows a Binomial distribution $\mathcal{B}(p_0, \nu_\epsilon)$, which can be approximated with a Gaussian distribution whenever ν_ϵ is sufficiently large, i.e.,

$$\mathcal{B}(p_0, \nu_\epsilon) \approx \mathcal{N}\big(p_0\nu_\epsilon, p_0(1-p_0)\nu_\epsilon\big). \tag{10}$$

Thanks to (10) we can apply the ICI-based CDT to P_i, to check whether the sequence (4) containing the average errors of K_0 over non-overlapping sequences Q_ϵ of ν_ϵ supervised samples

$$\widehat{p}_0 = \frac{1}{\nu_\epsilon} \sum_{t \in Q_\epsilon} \epsilon_t, \tag{11}$$

is stationary over time or not.

Similar to CDT_X, CDT_ϵ is paired with its own change-validation procedure to reduce the occurrence of false positives. The validation of any CDT_ϵ detection is here formulated as an inference problem on the proportions of misclassified observations by K_0 on two disjoint validation sets. An univariate t-test can be defined with the following null hypothesis

$$H_0 : \text{“}\overline{\epsilon}_{i-1} - \overline{\epsilon}_i = 0\text{”}$$

being $\overline{\epsilon}_{i-1}$ and $\overline{\epsilon}_i$ the average classification error of K_0 computed on supervised observations provided within $[T_{\text{ref},i-1}, \widehat{T}_{i-1}]$ and $[T_{\text{ref},i}, \widehat{T_i}]$, respectively. The test statistics

[1]For this comparison the values of V in $[T_{\text{ref},i}, \widehat{T_i}]$ and $[T_{\text{ref},i-1}, \widehat{T}_{i-1}]$ are computed from an unique value h_0 of (3), e.g. that one estimated from the concept spanning the longest time interval.

follow a Gaussian distribution, which can be rejected according to a predefined confidence level α (that in principle could differ from that used for CDT$_X$).

The configuration of CDT$_\epsilon$ requires the partitioning of Z_i –the set of supervised samples– into two subsets TS_i and VS_i such that $TS_i \cap VS_i = \emptyset$ and $TS_i \cup VS_i = Z_i$. TS_i is used to train K_0, while VS_i is used to compute the sequence of classification errors P_i as in (4). The initial configuration consists in computing confidence intervals using the Gaussian approximation (10), where p_0 is replaced by the average over P_i. As for CDT$_X$, the configuration of CDT$_\epsilon$ after each concept-drift detection relies on T_{ref} estimated by the refinement procedure; similarly, CDT$_\epsilon$ requires a minimum number of $\nu_\epsilon M_\epsilon$ supervised couples for training K_0 and computing P_i. Other considerations related to the configuration of CDT$_X$ hold also for CDT$_\epsilon$. Further details concerning CDT$_\epsilon$ can be found in [5].

It is important to remark that monitoring the classification error of K (i.e., the classification error measured on the available supervised couples, as in [15]) would result in a high false-positive rate for CDT$_\epsilon$ since K is expected to improve its accuracy, when supervised samples are made available in stationary conditions.

C. Concept Isolation: the Split Operator $\Upsilon(C_i)$

Every time a concept drift is detected, the JIT classifier builds a representation of the concept C_i by relying on the split operator Υ. In particular, C_i can be obtained by considering all supervised samples and features coming from time interval $[T_{\text{ref},i}, \widehat{T}_i]$. Then, also C_{i-1} – now referring to the previous concept – has to be updated by removing features and supervised couples related to the new concept. Different solutions for Υ are possible.

For instance, the simplest solution would include in C_{i-1} all available supervised samples and features acquired in time interval $[T_{\text{ref},i-1}, T_{\text{ref},i}]$ (being $T_{\text{ref},i-1}$ provided by the refinement procedure at the detection $\widehat{T}_{i-1}$). The corresponding split operator Υ is defined as

$$\Upsilon(C_i) = (C_i, C_{i-1}) \text{ where}$$
$$\begin{cases} C_i = \left(Z_{i|[T_{\text{ref},i},\widehat{T}_i]}, F_{i|[T_{\text{ref},i},\widehat{T}_i]}\right) \\ C_{i-1} = \left(Z_{i|[T_{\text{ref},i-1},T_{\text{ref},i})}, F_{i|[T_{\text{ref},i-1},T_{\text{ref},i})}\right) \end{cases}.$$

where $Z_{i|[t_1,t_2]}$ and $F_{i|[t_1,t_2]}$ collect the supervised couples and the feature values acquired in the time interval $[t_1, t_2]$ from Z_i and F_i, respectively. Since $T_{\text{ref},i}$ typically overestimates T_i^*, C_{i-1} might still contain supervised couples and features generated after the concept drift (and, as such, belonging to C_i and incoherent with C_{i-1}).

A different, more conservative, solution would consider supervised couples and features acquired within the interval $[T_{\text{ref},i-1}, \widehat{T}_{i-1}]$. In this case, the split operator Υ is defined as

$$\Upsilon(C_i) = (C_i, C_{i-1}) \text{ where}$$
$$\begin{cases} C_i = \left(Z_{i|[T_{\text{ref},i},\widehat{T}_i]}, F_{i|[T_{\text{ref},i},\widehat{T}_i]}\right) \\ C_{i-1} = \left(Z_{i|[T_{\text{ref},i-1},\widehat{T}_{i-1}]}, F_{i|[T_{\text{ref},i-1},\widehat{T}_{i-1}]}\right) \end{cases}.$$

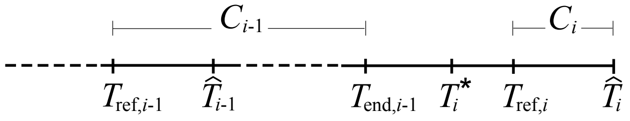

$C_{i\text{-}1}$
C_i
$T_{\text{ref},i\text{-}1}$
$\widehat{T}_{i\text{-}1}$
$T_{\text{end},i\text{-}1}$
T_i^*
$T_{\text{ref},i}$
$\widehat{T}_i$

Fig. 1. Time instants characterizing the isolation of the new concept C_i and the previous concept C_{i-1}.

This option guarantees to consider only features and supervised couples coming from the previous concept C_{i-1}, but discards those features and supervised couples belonging to interval $[\widehat{T}_{i-1}, T_{\text{ref},i}]$ (over-conservative approach).

A preferable solution would require computing $T_{\text{end},i-1}$, which is an estimate of the concept-drift time instant T_i^* satisfying the conditions $T_{\text{end},i-1} \leq T_i^* \leq T_{\text{ref},i}$, and use $T_{\text{end},i-1}$ to bound C_{i-1}. Thus, the split operator Υ would become

$$\Upsilon(C_i) = (C_i, C_{i-1}) \text{ where}$$
$$\begin{cases} C_i = \left(Z_{i|[T_{\text{ref},i},\widehat{T}_i]}, F_{i|[T_{\text{ref},i},\widehat{T}_i]}\right) \\ C_{i-1} = \left(Z_{i|[T_{\text{ref},i-1},T_{\text{end},i-1}]}, F_{i|[T_{\text{ref},i-1},T_{\text{end},i-1}]}\right) \end{cases} \tag{12}$$

This split operator discards features and supervised couples in $[T_{\text{end},i-1}, T_{\text{ref},i}]$ as they cannot be safely associated to any concept. The first CDT detecting a concept drift (either CDT$_X$ or CDT$_\epsilon$) is used to compute $T_{\text{end},i-1}$. The core idea is to use the CDT *backward*, by configuring it on the new concept C_i and then processing the (corresponding) features in D_i in a reverse time order to detect a deviation from the new encountered concept. In practice, the CDT is trained on the features (or the average classification error) in $[T_{\text{ref},i}, \widehat{T}_i]$ and processes those between $T_{\text{ref},i}$ and $T_{\text{ref},i-1}$: the change-point detected by such CDT is denoted by $T_{\text{end},i-1}$ and typically underestimates T_i^* (considering the forward time). Fig. 1 illustrates how the estimates $T_{\text{end},i-1}$, $\widehat{T}_i$ and $T_{\text{ref},i}$, provided by the online CDTs, the backward CDT and the refinement procedure, respectively, are typically set along the time-horizon w.r.t. the true concept-drift time-instant T^*, and how concepts C_i and C_{i-1} are isolated by the split operator Υ. In our implementation we use the split operator defined in (12).

D. Identifying Recurrent Concepts: the Equivalence Operator $\mathcal{E}$

The identification of recurrent concepts consists in an exhaustive, pair-wise, comparison of the new concept representation C_i with each concept representation in $\mathcal{C} = \{C_j, 0 \leq j < i\}$ by means of the equivalence operator $\mathcal{E}$. The comparison between C_i and C_j determines if the two representations belong to the same concept. Such comparison cannot be carried out by estimating the underlying pdfs, as in general this requires a large number of supervised couples. Differently, we analyze sample moments of the input pdf (by comparing F_i and F_j) and information derived from the $p(y|x)$ (by comparing Z_i and Z_j). The comparison between F_i and F_j is formulated as an equivalence-test problem, while comparison of supervised samples consists in analyzing the similarity of two classifiers trained on Z_i and Z_j on a common validation set. The recurrent concept operator $\mathcal{E}$ is defined as:

$$\mathcal{E}(C_i, C_j) = \begin{cases} 1 & \text{if } F_i \text{ is } \textit{equivalent} \text{ to } F_j \text{ and} \\ & \quad Z_i \text{ is } \textit{equivalent} \text{ to } Z_j \\ 0 & \text{otherwise} \end{cases} .$$

Comparing F_i and F_j: Equivalence tests are statistical techniques originally developed to assess if two different formulations of the same drug are bioequivalent, i.e., they provide equal therapeutic effects. Bioequivalence is typically assessed by analyzing univariate biological responses using the Two One-Sided Tests (TOST) procedure [39], [40]. In TOST, the null hypothesis corresponds to the non-equivalence of the analyzed quantities, while the equivalence corresponds to the alternative hypothesis. It follows that, when the null hypothesis is rejected, the TOST concludes the equivalence at a given confidence level α.

When comparing any pair of concept representations C_i and C_j, we apply two independent TOSTs to determine if the mean of the κ-th component $\overline{F}_i^{\kappa}$ of F_i is equivalent to the mean of the κ-th component $\overline{F}_j^{\kappa}$ of F_j, where $\kappa = 1, 2$. The null (H_0) and alternative (H_1) hypothesis are formulated as

$$H_0 : \text{“}\left(\overline{F}_j^{\kappa} - \overline{F}_i^{\kappa}\right) < \theta_L \text{ or } \left(\overline{F}_j^{\kappa} - \overline{F}_i^{\kappa}\right) > \theta_U\text{,”} \qquad (13)$$
$$H_1 : \text{“}\theta_L \leq \overline{F}_j^{\kappa} - \overline{F}_i^{\kappa} \leq \theta_U\text{,”}$$

where θ_L and θ_U are the upper and lower bounds for the equivalence region. Since $\overline{F}_i^{\kappa}$ and $\overline{F}_j^{\kappa}$ follow a Gaussian distribution, the above test can be performed by using two one-sided t-tests.

Comparing Z_i and Z_j: The comparison of features F_i and F_j disregards the supervised information (stored in Z_i and Z_j) and, as such, it may erroneously state the equivalence between two concepts characterized by the same input distribution, although having labels ruled by very different data-generating processes.

To check whether Z_i and Z_j are generated from the same pdf we train two classifiers K_i and K_j on two subsets $TS_i \subset Z_i$ and $TS_j \subset Z_j$, such that the number of couples in TS_i equals the one in TS_j, i.e., $\#TS_i = \#TS_j$, where $\#$ denotes the set cardinality. Classifiers K_i and K_j are estimates of the distributions determining the labels in Z_i and Z_j, respectively: our approach consists in measuring how different these distributions are by comparing the outputs of K_i and K_j on a common validation set. To this purpose, let VS_i and VS_j be the supervised couples of Z_i and Z_j that are not in TS_i and TS_j, respectively, and define the common validation set VS as the largest between VS_i and VS_j. We then compute the percentage of samples q for which K_i and K_j provide the same output on VS, i.e.,

$$q = 1 - \frac{\sum\limits_{VS} \delta_{i,j}(x_t)}{\#VS}, \qquad (14)$$

where $\delta_{i,j}(x_t)$ is 1 if $K_i(x_t) = K_j(x_t)$, 0 otherwise. We say that Z_i and Z_j are not equivalent when q exceeds threshold τ. The rationale behind this procedure is similar to that leading to the measure of conceptual equivalence in [23].

E. Classify Input Samples: Classifier K

Classifier K assigns the label $\widehat{y}_t = K(x_t)$ to each input x_t, by relying on knowledge base that includes Z_i (the supervised couples provided since the last concept-drift detection) and Z_{rec} (the set of supervised couples belonging to concept representations that are considered equivalent). The JIT framework is general and different solutions can be obtained by using linear discriminant analysis, k-nearest neighbor, naïve Bayes classifier, feedforward neural networks, support vector machines and any other model both for K and K_0.

F. Algorithm

The JIT adaptive classifier for recurrent concepts of Algorithm 2 is detailed in the sequel. Lines 1-8 refer to the training phase. In stationary conditions (lines 9-22, 41-42), the classifier improves its accuracy over time by exploiting additional supervised information coming from the field and updates the current concept representation. Differently, in nonstationary conditions (lines 23-40), the current classifier is discarded and a new one configured with supervised pairs coming from the current concept (lines 24-29) or from coherent previously encountered concepts (lines 30-40).

In precisely, the training phase of the proposed classifier (lines 1-8) consists in configuring K, K_0 (trained on TS_0), CDT$_X$, and CDT$_\epsilon$ (configured on P_0 computed on VS_0, where $Z_0 = TS_0 \cup VS_0$). Each of these elements requires a different amount of samples to be configured: for sake of simplicity, in the following, we assume that the initial training sequence contains enough observations (both supervised and not) to configure all of them. However, in circumstances where a suitable training sequence is not available, it is possible to postpone the activation of the CDTs until enough samples for the configuration become available. It is important to remark that K_0 is never updated with additional supervised information coming from the field and is retrained only when a concept drift is detected (lines 32-34).

During the operational life (line 9), the classifier receives inputs x_t from the data-generating process, and C_i is updated according to the $\mathcal{U}$ operator described in Section V-A (lines 12, 20). The classifier K is promptly retrained as soon as a new couple (x_t, y_t) is made available (line 13). When ν_ϵ supervised couples have been acquired (line 15), CDT$_\epsilon$ is executed. Similarly, when a sequence of ν_X observations (disregarding their labels) has been acquired (line 19), the features M and V are computed as in (2) and (3), appended to F_i (line 20), and CDT$_X$ is then executed (line 21). The combination of CDT$_X$ and CDT$_\epsilon$ according to (8) yields the $\mathcal{D}$ operator (line 23) described in Section V-B. The buffer sequence Q_X is used to temporarily store input data for feature extraction, while Q_ϵ temporarily stores the supervised couples to compute the estimates of classification error of K_0.

When either CDT$_X$ or CDT$_\epsilon$ detects a change, the corresponding refinement procedure is executed yielding T_{ref} (line 25), and the new concept is isolated (lines 26, 27) by means of the split operator Υ (12). After the change, all obsolete information is removed from the knowledge base of K (line 28), which is then trained by using only those supervised

```
1-  Acquire the initial training sequence $O_{T_0}$;
2-  Extract $F_0$ from $O_{T_0}$ and configure CDT$_X$;
3-  Build $Z_0$ as the set of supervised couples in $O_{T_0}$;
4-  Partition $Z_0$ into $TS_0$ and $VS_0$;
5-  Train $K$ on $Z_0$ and $K_0$ on $TS_0$;
6-  Compute $P_0$ as in (4) and configure CDT$_\epsilon$;
7-  $D_0 = \big(F_0, P_0\big)$ and $C_0 = \big(Z_0, F_0, D_0\big)$;
8-  $i = 0$, $Q_X = \emptyset$, $Q_\epsilon = \emptyset$;
9-  while ($x_t$ is available) do
10-     $Q_X = Q_X \cup \{x_t\}$;
11-     if ($y_t$ is available) then
12-         $Z_i = Z_i \cup \{(x_t, y_t)\}$;
13-         Update $K$ with $(x_t, y_t)$;
14-         $Q_\epsilon = Q_\epsilon \cup \{(x_t, y_t)\}$;
15-         if ($\#Q_\epsilon = \nu_\epsilon$) then
16-             $\widehat{p}_0 = \frac{1}{\nu_\epsilon} \sum_{Q_\epsilon} (|K_0(x_t) - y_t|)$ as in (11) and
                append it to $P_i$;
17-             $r_\epsilon$= CDT$_\epsilon(P_i)$;
18-             $Q_\epsilon = \emptyset$;
            end
        end
19-     if ($\#Q_X = \nu_X$) then
20-         Compute new feature value and insert in $F_i$;
21-         $r_X$= CDT$_X(F_i)$;
22-         $Q_X = \emptyset$;
        end
23-     if (($r_X = 1$)||($r_\epsilon = 1$)) then
24-         $i = i + 1$;
25-         Estimate $T_{\text{ref},i}$ with the refinement proc. [4];
26-         Estimate $T_{\text{end},i-1}$ with CDT$_X$ (or CDT$_\epsilon$)
            configured on $C_i$ operating backward on $D_i$;
27-         Isolate $C_i$ and $C_{i-1}$ as described in (12);
28-         Configure $K$ from $Z_i$;
29-         reconfigure $= 1$;
        end
30-     if (reconfigure $= 1$ and $\#F_i \geq M_X$ and $\#P_i \geq M_\epsilon$)
        then
31-         Configure CDT$_X$ on $F_i$;
32-         Partition $Z_i$ into $TS_i$ and $VS_i$;
33-         Train $K_0$ on $TS_i$;
34-         Compute $P_i$ as in (4) and configure CDT$_\epsilon$;
35-         $Z_{\text{rec}} = \emptyset$;
36-         for (j = 0 ; j < i ; j++) do
37-             if ($\mathcal{E}(C_i, C_j) = 1$) then
38-                 $Z_{\text{rec}} = Z_{\text{rec}} \cup Z_j$;
                end
            end
39-         train $K$ on $Z_i \cup Z_{\text{rec}}$;
40-         reconfigure $= 0$;
        end
41-     if ($y_t$ is not provided) then
42-         Assign label $\widehat{y}_t = K(x_t)$ to $x_t$.
        end
    end
```

Algorithm 2: The proposed JIT classifier for recurrent concepts.

couples belonging to Z_i. In contrast, the CDTs configuration and the procedure for identifying recurrent concepts typically require a minimum number of samples (i.e., $\#F_i > M_X$ and $\#P_i > M_\epsilon$). For this reason, the reconfiguration is scheduled (line 29) and postponed until enough samples (either supervised or not) are made available (line 30). This induces a latency in the JIT classifier that is temporarily forced to operate without any CDT to monitor concept drift (thus operating without $\mathcal{D}$). Therefore, no concept drift can be detected during such a latency period, although the classifier K (configured on the most recent samples) assigns labels to inputs (line 42) and is regularly updated at each supervised couple (line 13); C_i is also regularly updated (lines 12 and 20). As soon as enough features and supervised samples are available, the CDTs are reconfigured to detect a further concept drift; in particular, CDT$_X$ is configured on F_i (line 31), K_0 is trained on TS_i (line 33) and the average classification error P_i is computed over VS_i. CDT$_\epsilon$ is then configured from P_i (line 34).

Afterwards (lines 36 - 38), C_i is compared with all the concept representations stored by the JIT classifier in $\mathcal{C} = \{C_j, 0 \leq j < i\}$ by means of the equivalence operator $\mathcal{E}$ described in Section V-D. This procedure allows the JIT classifier to take advantage of recurrent concepts, since K is trained again by using supervised couples belonging to C_i and all concept representations equivalent to C_i (lines 39 - line 40). Finally, when x_t is not supervised (line 41), it is classified by K (line 42).

We emphasize that the need to store feature values F_i and supervised samples Z_i in concepts C_i could make the use of the proposed JIT classifier critical on potentially-infinite datastreams. To address this issue we could consider different solutions. First, we could impose a bound on the number of supervised samples stored in Z_i. Second, pruning techniques on the supervised samples can be considered (e.g., condensing and editing techniques of the knowledge base). Third, an oblivion coefficient could be considered to implicitly remove oldest occurrences of concepts.

VI. Experiments

To test the performance of the JIT classifier provided in Algorithm 2 we considered synthetic datasets (both scalar and multivariate) and a real world application characterized by high-dimensional data. We consider the following base classifiers:

- a k-Nearest Neighborhood (k-NN) classifier where k is set as in [3].
- a Naïve Bayes (NB) classifier based on Gaussian distributions.
- a Support Vector Machine (SVM) classifier.

The above classification cores were then inserted into different adaptive classification frameworks for comparison:

- the proposed JIT classifier of Algorithm 2 (green dashed line with square marker).
- the JIT classifier of Algorithm 2 without the recurrent concept management ability (blue solid line with triangle down marker).

- A classifier U, trained on all available data every time a new supervised couple is provided (dotted black line). This classifier guarantees the best performance in stationary conditions.
- A short memory classifier W, trained on a sliding window open over the latest 40 supervised samples (solid red line with circle marker). This passive classifier naturally adapts to concept drift.
- An ensemble E that combines the proposed JIT classifier with W by using the following selection rule: the label is assigned by the classifier that guarantees the highest accuracy over the last 40 supervised couples (dashed-dotted purple line with triangle up marker).

All the implementations of JIT classifiers use the same family of classifiers for both K and K_0.

Figures of Merit: The classification accuracy on unsupervised samples is the main figure of merit used to assess the performance of the considered adaptive classifiers (in a controlled environment we know which label should be associated to each input). Plots in Fig. 2 and 3 show, in each time instant, the percentage of misclassified samples in 2000 runs, averaged over a sliding window of 40 samples. The average classification error over the entire dataset is reported in Table I that also shows the performance in terms of *precision* and *recall* of the equivalence operator, computed as

$$precision = \frac{tp}{tp + fp} \quad \text{and} \quad recall = \frac{tp}{tp + fn}, \qquad (15)$$

being:

tp	(true positives): number of pairs (C_i, C_j) referring to the same concept and yielding $\mathcal{E}(C_i, C_j) = 1$;
fp	(false positives): number of pairs (C_i, C_j) referring to different concepts but yielding $\mathcal{E}(C_i, C_j) = 1$;
fn	(true negative): number of pairs (C_i, C_j) referring to the same concept and yielding $\mathcal{E}(C_i, C_j) = 0$.

In the above figures of merit we consider only pairs (C_i, C_j) such that both C_i and C_j are entirely included within their respective concept (i.e., no concept drift within each concept representation). Note that *precision* and *recall* can be computed also when the dataset does not contain recurrent concepts, since descriptors within the same concept are equivalent.

A. Testbeds Description

In the following classification problems a two-classes' output is considered ($\Lambda = \{\omega_1, \omega_2\}$), with supervised couples provided at regular time instants. We generate synthetic datasets with classes having same probability and we provide a supervised sample out of $m = 5$ observations. The initial training sequence is composed of T_0 = 400 observations (80 supervised samples). In the real world dataset we kept the settings of [26]: the supervised label is provided once the corresponding observation has been classified; the classification error is computed at each sample. Here, the initial training sequence is composed of T_0 = 50 (supervised) observations.

Synthetic Dataset, the SCALAR Testbed: Each dataset is composed of $N = 10000$ scalar observations while concepts are ruled by Gaussian pdfs whose parameters are reported in Table II. In the following, we consider situations where both concept drift affect $p(y|x)$ and the input distribution or the $p(y|x)$ only:

- *SCALAR_1*: an abrupt concept drift affects both classes; at time $t = 5000$ the concept drifts from C_1 to C_2.
- *SCALAR_2*: a sequence of recurrent abrupt concept drift that affects both the input distribution and $p(y|x)$. The concept alternates between C_1 and C_2 every 2000 samples (recurrent concepts).
- *SCALAR_2A*: same as *SCALAR_2* but the concept drift is less evident since the concept alternates between C_1 and C_3.
- *SCALAR_3*: a sequence of recurrent abrupt concept drift that affects only $p(y|x)$. Every 2000 samples the concept alternates between C_1 and C_4 (classes' swap).
- *SCALAR_3A*: a sequence of classes' swap similar to *SCALAR_3* alternating between C_5 and C_6. Classification and detection of classes' swap are here more challenging problems compared with those in *SCALAR_3*.
- *SCALAR_4*: a sequence of abrupt concept drift affecting both classes. An offset of $+2$ is added to each class mean every 2000 samples (the initial concept is C_1). Each concept drift affects both the input distribution and $p(y|x)$ (no recurrent concepts).
- *SCALAR_4A*: similar to *SCALAR_4*. Here the concept drift is less evident since the offset added to each class mean is $+1$.

TABLE II
CONSIDERED CONCEPTS

Concept	$p(x\|\omega_1)$	$p(x\|\omega_2)$
C_1	$\mathcal{N}(0,4)$	$\mathcal{N}(2.5,4)$
C_2	$\mathcal{N}(2,4)$	$\mathcal{N}(4.5,4)$
C_3	$\mathcal{N}(1,4)$	$\mathcal{N}(3.5,4)$
C_4	$\mathcal{N}(2.5,4)$	$\mathcal{N}(0,4)$
C_5	$\mathcal{N}(0,4)$	$\mathcal{N}(2,4)$
C_6	$\mathcal{N}(2,4)$	$\mathcal{N}(0,4)$

Synthetic Datasets, MULTIVARIATE Testbed: These datasets model classification problems where the concept drift affects only $p(y|x)$ and leaves unaltered the input distribution. In particular, we consider the following datasets:

- *CHECKERBOARD_1*: each data sequence is composed of 10000 observations uniformly distributed in the $[0,1] \times [0,1]$ domain; the classification function is ruled by a checkerboard in $[0,1] \times [0,1]$ containing 4 squares of side 0.5. The concept drift affects the classification function by rotating the checkerboard of $\{0, \pi/6, \pi/3, \pi/2, 4/3\pi\}$ every 2000 samples (no recurrent concepts).
- *CHECKERBOARD_2*: similar to *CHECKERBOARD_1* with the five concepts alternating between 0 and $\pi/6$ (recurrent concepts).
- *CHECKERBOARD_3*: similar to *CHECKERBOARD_1*, but each sequence is composed of 20000 observations and the rotations of $\{0, \pi/3, 2/3\pi, \pi, 4/3\pi, 5/3\pi, 2\pi\}$ occur at regular time intervals (recurrent concepts).
- *MULTIVARIATE_GAUSSIAN*: each class is drawn from a two-variate Gaussian distribution. Class ω_1 is characterized by mean $[0,0]$ and covariance matrix $\mathbb{I}_2$ (the

TABLE I
AVERAGE CLASSIFICATION ERROR (%) AND RECURRENT CONCEPTS IDENTIFICATION PERFORMANCE.

Experiment	**Base classifier**	**JIT**	**Ensemble**	**JIT w/o recurrent**	**Short Memory (W)**	**Continuous Update (U)**	**Precision Recurrent**	**Recall Recurrent**
SCALAR_1	k-NN	**27.75**	28.44	27.76	30.87	29.64	1	0.530
	NB	**27.02**	27.32	27.03	28.00	29.05	1	0.840
SCALAR_2	k-NN	**28.84**	29.32	28.93	31.07	29.62	1	0.537
	NB	**27.62**	27.75	27.67	28.21	29.06	1	0.870
SCALAR_2A	k-NN	28.47	29.04	28.54	30.90	**27.82**	1	0.543
	NB	**27.29**	27.52	27.33	28.02	27.33	1	0.864
SCALAR_3	k-NN	38.30	**31.66**	38.24	32.46	47.08	1	0.510
	NB	35.48	**29.69**	35.44	29.79	46.46	1	0.833
SCALAR_3A	k-NN	43.63	**36.53**	43.50	36.99	47.76	1	0.255
	NB	40.73	**34.03**	40.67	34.19	47.20	1	0.592
SCALAR_4	k-NN	28.95	29.40	**28.91**	31.07	37.93	0.259	0.504
	NB	27.71	27.81	**27.66**	28.20	37.75	0.407	0.899
SCALAR_4A	k-NN	28.54	29.10	**28.50**	30.89	31.52	0.062	0.452
	NB	27.38	27.59	**27.32**	28.01	31.07	0.072	0.818
CHECKERBOARD_1	k-NN	21.45	**17.06**	21.41	21.77	44.58	0.422	0.724
CHECKERBOARD_2	k-NN	19.92	**14.32**	20.37	18.93	24.48	1	0.799
CHECKERBOARD_3	k-NN	18.60	**15.60**	18.83	20.48	25.67	0.977	0.833
MULTIVARIATE	k-NN	23.60	**21.74**	23.61	25.00	47.85	1	0.947
GAUSSIAN	NB	21.52	**19.97**	21.52	21.08	49.03	1	1
SINE_2	k-NN	14.33	**11.09**	15.50	15.59	44.07	1	0.987
SINE_2A	k-NN	19.49	**12.80**	20.55	18.10	44.43	1	0.932
SINE_IRREL_2	k-NN	23.76	**18.37**	24.79	24.19	45.49	1	0.793
SINE_IRREL_2A	k-NN	31.23	**22.05**	31.64	27.33	45.83	1	0.415
EMAIL_LIST	k-NN	42.00	**36.65**	42.00	36.55	37.03	-	0
	SVM	22.34	**17.31**	22.90	22.62	42.83	1	0.250

2×2 identity matrix), class ω_2 is characterized by mean $[2, 0]$ and covariance matrix $4\mathbb{I}_2$. The concept drift occurs at $t = 5000$ and induces a classes' swap. Each data sequence is composed of 10000 observations.

- *SINE_2*: inputs (x_1, x_2) are uniformly distributed over the $[0, 1] \times [0, 1]$ domain and the classification function is $y_t = \mathcal{I}(x_2 > 0.5 + 0.3 \sin(3\pi x_1))$, being $\mathcal{I}$ the indicator function. Concept drift occurs every 2000 samples, producing a classes' swap. Each data sequence is composed of 10000 observations.
- *SINE_2A*: the same as *SINE_2* but 10% of supervised samples are provided with the wrong label (class noise).
- *SINE_IRREL_2*: the same as *SINE_2* but observations are uniformly distributed in $[0, 1]^4$, with two attributes that do not influence the classification function (irrelevant attributes).
- *SINE_IRREL_2A*: the same as *SINE_IRREL_2* with 10% of class noise added to supervised samples.

Datasets *MULTIVARIATE_GAUSSIAN* was taken from [15], as well as the *SINE_2*, *SINE_IRREL_2*, which, however, have been periodically repeated to yield recurrent concepts. Datasets *SINE_2A* and *SINE_IRREL_2A* present the same class noise as the moving hyperplanes dataset of [41]. Datasets *CHECKERBOARD_1*, 2 and 3 take inspiration from [7].

Real World Dataset, the Emailing list: We considered the *EMAIL_LIST* real-world dataset presented in [26], addressing a spam email filter application. The dataset contains a sequence of email messages, each related to a specific topic. The task consists in classifying emails (as a virtual user would do according to their topic) either as *interesting* or *junk*.

Concept drift is simulated as a sudden change of the user behavior who labels as interesting/junk messages from topics that previously were considered as junk/interesting. Concept drift is introduced every 300 samples; the dataset contains a sequence of recurrent classes' swap. Email messages are provided in the Boolean bag-of-words representation, each containing 913 Boolean attributes; the whole dataset consists on 1500 supervised messages.

Configuration of the JIT Classifier: Both CDT$_X$ and CDT$_\epsilon$ are configured as described in Section V, with ν_X and ν_ϵ set to 20. To reconfigure CDT$_X$ and CDT$_\epsilon$ we require a minimum of $M_X = 4$ feature values (corresponding to $\nu_X M_X = 80$ input data) and $M_\epsilon = 4$ estimates of $\widehat{p}_0$ (corresponding to $\nu_\epsilon M_\epsilon = 80$ supervised couples). During the initial configuration and after each concept-drift detection, the classifier K_0 is trained on 40 supervised samples, while the classification error computed on the remaining 40 samples is used to configure CDT$_\epsilon$. In case of scalar datasets the JIT classifier relies on both CDT$_X$ and CDT$_\epsilon$, while on datasets characterized by multivariate observations only CDT$_\epsilon$ is used. Similarly, in the case of multivariate datasets the procedure for identifying recurrent concepts consists only in the analysis of supervised couples.

Other relevant parameters of the CDTs and of the JIT classifier are described in Table III. The Γ parameter tunes the ICI-based CDT sensitivity, while Γ_{ref} and λ refer to the refinement procedure; detailed descriptions and guidelines on how to set these parameters can be found in [4]. To speed up the concept-drift detection (at the expenses of a larger number of false positives introduced by the CDTs) we do not enforce here the change-validation procedure, though in different contexts it can be conveniently used. The parameters θ_L, θ_U, α and τ of $\mathcal{E}$ are set in a very conservative way so as to prevent merging supervised couples of different concepts (at

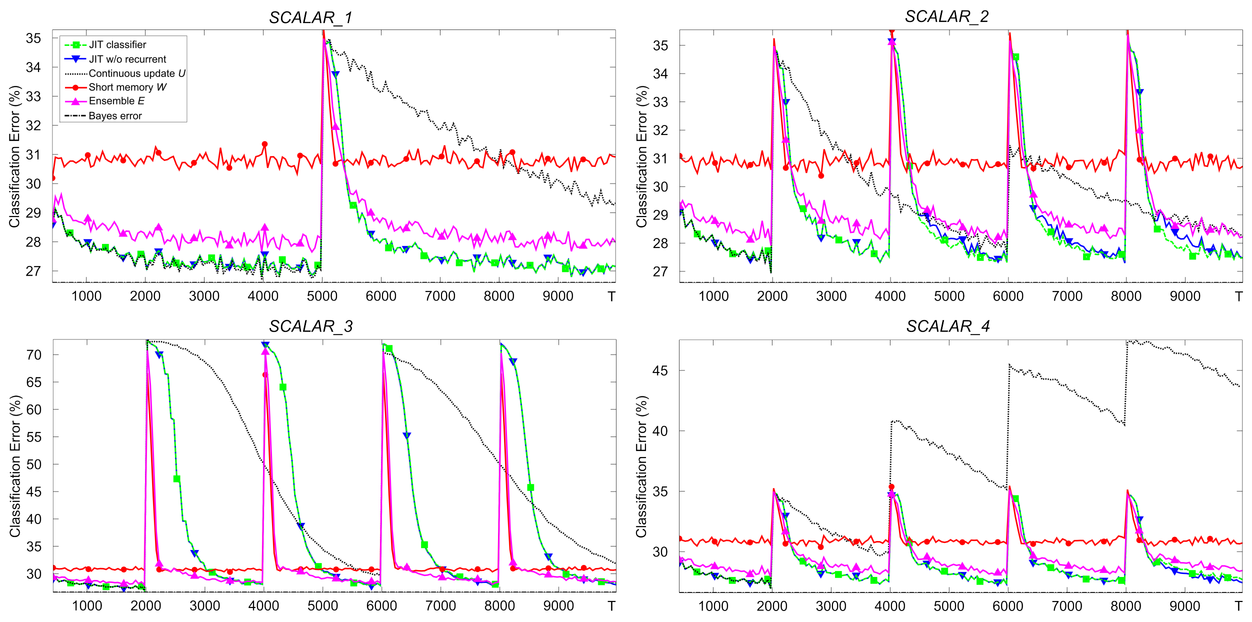


Fig. 2. Scalar Datasets: The classification errors over time for the proposed JIT classifier and the other methods. The reference classifier is a k-NN

the cost of missing some correct matches between concepts).

Each dataset has been processed by using k-NN as a base classifier. In some datasets, experiments have been run with the NB and the SVM classifiers; results are reported in Table I.

TABLE III
JIT CLASSIFIER PARAMETERS

Parameter	Value	Description
Γ	2	ICI-based CDT parameter
Γ_{ref}	2	ICI-based CDT parameter
λ	1.5	ICI-based CDT parameter
θ_L and θ_U	$0.3\overline{F^{\kappa}}_j$	parameters of TOST
α	0.1	parameters of TOST
τ	0.8	Threshold of (14)

B. Discussion

The *SCALAR_1* dataset shows that both JIT classifiers tend to the Bayes error even when an abrupt concept drift occurs: in fact, before the concept drift their performance are very close to that of U, the continuously reconfigured classifier (black line). Then, after $t = 5000$, the JIT classifiers quickly adapt to the new concept by removing the obsolete training samples. The short memory classifier W provides the best performance right after the change, however, it is not able to improve its accuracy over time: it follows that in stationary conditions both JIT classifiers outperform W and the ensemble E as the selection rule of E may choose W. Although the dataset does not contain recurrent concepts, Table I shows that the proposed JIT classifier provides a marginal improvement over the others. This is due to the fact that the identification of recurrent concepts mitigates the impact of CDT false positives, as the operator $\mathcal{E}$ eventually merges sequences of supervised samples that have been erroneously split by the operator $\mathcal{D}$. Since here the *precision* equals 1, representations from different concepts are never considered equivalent. Results are similar for both the k-NN and the NB classifier.

The effectiveness in identifying recurrent concepts clearly emerges in *SCALAR_2* and *SCALAR_2A* datasets. The error curves in Fig. 2 show that recovering recurrent concepts is definitively beneficial: as soon as the recurrent concept procedure is activated (i.e., after $\nu_X M_X$ observations since T_{ref} or $m\nu_\epsilon M_\epsilon$ observations when the detection comes from CDT$_\epsilon$) the JIT outperforms the JIT without the recurrent concept management capability. Such an improvement is more substantial in the last concept drift, where it is possible to gather supervised couples from the two previous occurrences of the same concept. According to Table I, the U classifier has a lower classification error than the JIT on the *SCALAR_2A* when k-NN is used as a base classifier; this is due to the fact that U takes always advantage of recurrent concepts, as shown in Fig. 2, since it stores all the supervised couples in its knowledge base. However, U achieves better performance only when the concept drift results in reduced losses in the classifier accuracy. The *precision* and *recall* values in Table I show that the equivalence operator provides no false positives and the false negatives increase as the concept drift detection becomes more challenging (*SCALAR_2A*).

In the *SCALAR_3* and *SCALAR_3A* datasets the JIT is slower in adapting to concept drift than in other scalar datasets: CDT$_X$ cannot detect this type of concept drift since the input data distribution remains unaltered. A possible large detection latency might arise due to the fact that only one sample out of $m = 5$ is supervised and CDT$_\epsilon$ is executed every $m\nu_\epsilon = 100$ observations. Here, E outperforms other classifiers, since after each concept drift it selects the classifier W that regularly refreshes its knowledge base. The identification of recurrent concepts is successful in both datasets as the equivalence operator $\mathcal{E}$ does not introduce false positives (*precision* is 1). In the *SCALAR_3A* dataset, characterized by a larger overlap among classes than in *SCALAR_3*, the *recall* values are lower, indicating that the identification of recurrent concepts becomes

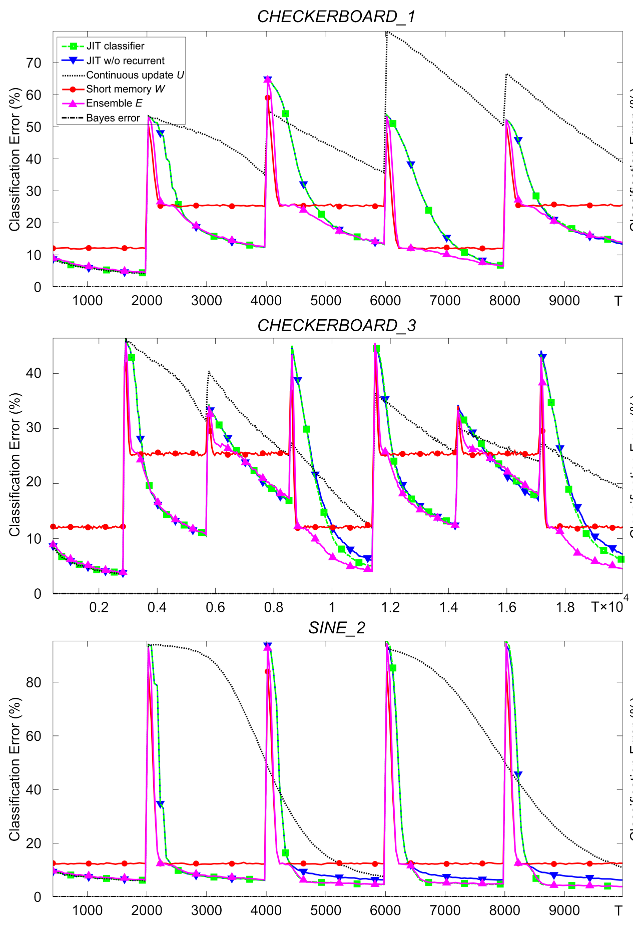
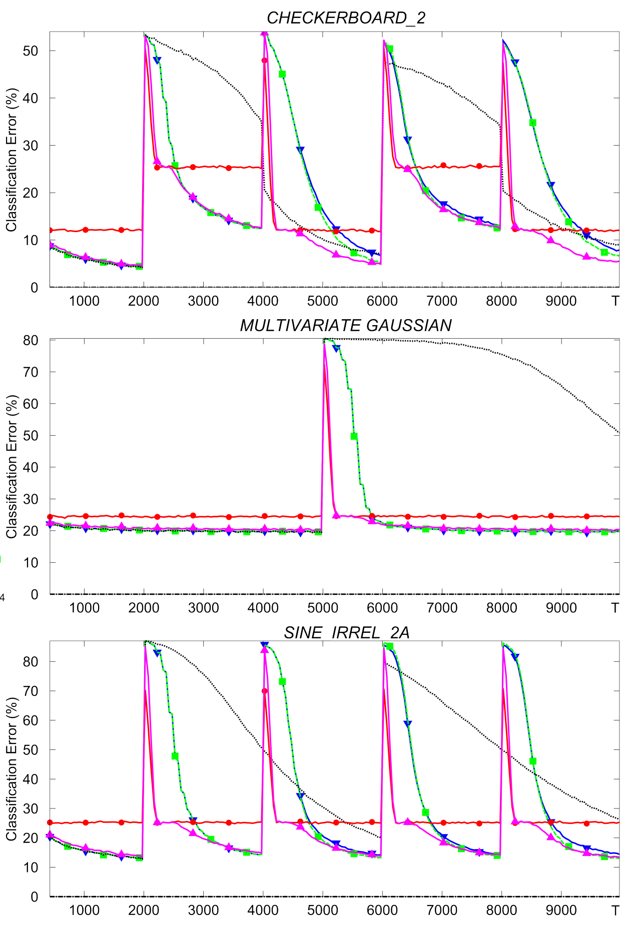



Fig. 3. Multivariate Datasets: the classification error as function of time for the proposed JIT and the other classifiers. These plots have been obtained using a k-NN as base classifier.

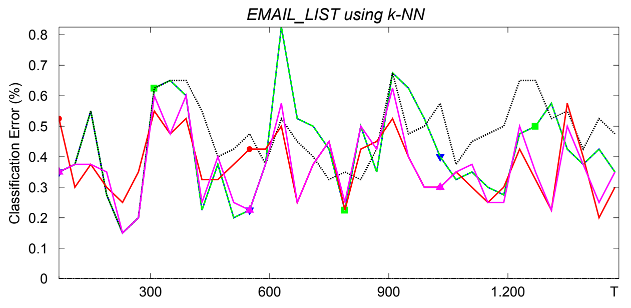
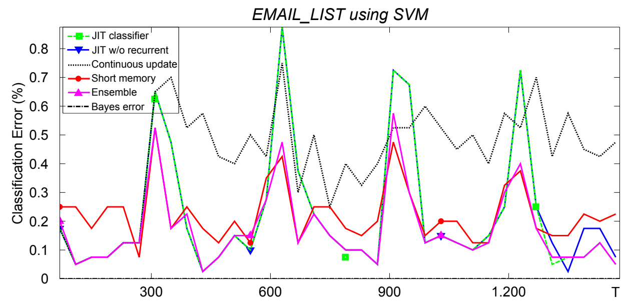



Fig. 4. Real World Datasets: the classification error as function of time for the proposed JIT and the contrasted classifiers. The average classification error is computed on a window containing the last 40 supervised samples.

more difficult when the change is less evident. The classification errors in Table I show that, here, the improvement provided by additional supervised information coming from recurrent concepts is marginal, and is negatively compensated by detection delays introduced by CDT$_\epsilon$ and errors in concept isolation. Immediately after each concept drift, the classifier K becomes obsolete: the larger the knowledge base, the less beneficial contribution of fresh supervised couples. Coherently, when NB is used as base classifier, also *recall* improves since K is more accurate on this dataset.

The *SCALAR_4* and *SCALAR_4A* datasets illustrate situations where the procedure for identifying recurrent concepts fails, and the JIT classifier without the procedure to identify recurrent concepts outperforms the proposed JIT classifier. The low values of *precision* show that the operator $\mathcal{E}$ introduces some false positives considering equivalent descriptors coming from different concepts. Of course, the probability of having a false positive increases with the number of comparisons (the error curves diverge in the last part of the dataset in Fig 2), and when the concept drift is less severe (the *precision* and *recall* values in *SCALAR_4A* are lower in than *SCALAR_4*). In both datasets, the best performing solution is the JIT without the identification of recurrent concepts ability.

Results on other multivariate datasets presented in Fig. 3 are in line with the comments about the *SCALAR* datasets. In particular, the *CHECKERBOARD* datasets show that both the

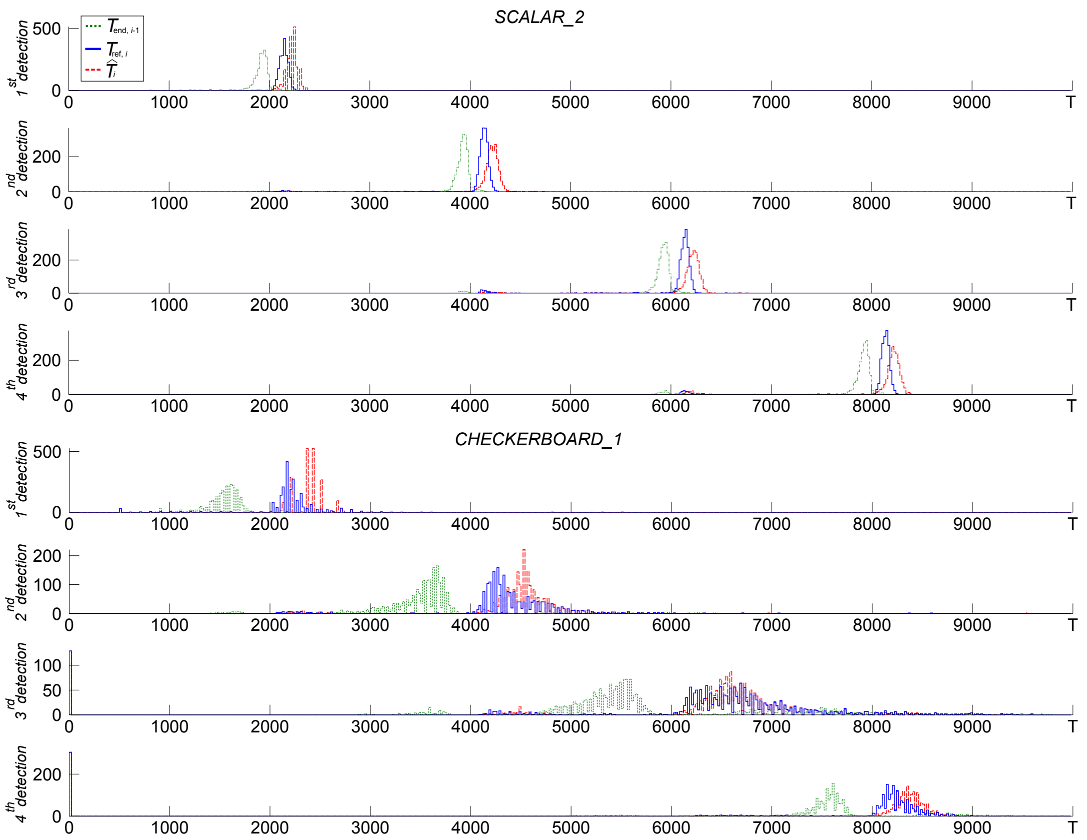


Fig. 5. The histograms of $\hat{T}$, T_{end} and T_{ref}, the outcomes of the CDTs, in two considered datasets. The skewed and specular shape of the histograms of T_{end} and T_{ref} proves that the concept isolation procedure is effective. These histograms show that the JIT classifier is successfully reconfigured, and operates after each detection as in its initial state, without drifting.

JIT classifiers and the procedure for dealing with recurrent concepts successfully operate when the classification task becomes more difficult. Moreover, its performance does not degrade significantly when longer sequences, characterized by several concept drifts, are considered as in *CHECKERBOARD_3* (although the equivalence operator has few false positives). In all multivariate datasets, the lowest classification error is achieved by the ensemble E, whose structural promptness compensates the absence of CDT$_X$, as in the *SCALAR_3* dataset.

On the *CHECKERBOARD* datasets, the concept drift is sometimes more difficult to be detected, resulting in a longer detection delay. To have an idea about how difficult detecting a specific concept drift is, we observe the performance of the short memory classifier W. Since both W and K_0 are trained over the last 40 samples, the error curves of W provide reliable estimates of p_0 in (10), i.e., the expected value of the error monitored by CDT$_\epsilon$. Basically, the lower the variation in the classification error of W, the more difficult the detection. For instance, it is more difficult to detect the third concept drift in *CHECKERBOARD_1* than the others (p_0 varies from $\approx 25\%$ to $\approx 48\%$, while in the first concept drift it varies from $\approx 12\%$ to $\approx 50\%$, and in the second from $\approx 25\%$ to $\approx 60\%$) and this results in a longer detection latency that may impair the concept isolation performance. It follows that in Fig. 5 the histograms of $\hat{T}$, T_{ref} and T_{end} are more spread in correspondence of the third concept drift of *CHECKERBOARD_1* than elsewhere. In contrast, histograms of *SCALAR_2* show that, when both CDT$_X$ and CDT$_\epsilon$ simultaneously operate, the distributions of $\hat{T}$, T_{ref} and T_{end} do not significantly change and the number of missed detections is negligible. The skewed and specular shapes in the histograms of T_{end} and T_{ref} prove the effectiveness of the concept isolation, as the largest area below these histograms is concentrated around T^*. These histograms, together with the trend of the classification errors in all plots, show that the JIT classifier is successfully reconfigured after each detection.

Datasets derived from *SINE_2* show that the class noise as well as the injection of irrelevant attributes increase the classification error of K_0, as it emerges from the error curves of *SINE_2* and *SINE_IRREL_2A* in Fig. 3 (as a consequence the detection latency of CDT$_\epsilon$ is higher). Furthermore, the operator $\mathcal{E}$ becomes less accurate, as shown the by *recall* values in Table I. Nevertheless, in these datasets, the management of recurrent concepts is always successful and provides substantial improvements in the classification errors of Table I. The trend of the classification error on *GAUSSIAN MULTIVARIATE* is in line with the other results, with the operator $\mathcal{E}$ that eventually merges concepts erroneously split by false positives of $\mathcal{D}$, as in *SCALAR_1*.

The performance on the *EMAIL_LIST* dataset is consistent with the above considerations. The classification task is here rather challenging due to the very large dimensionality of the observations. The plots in Fig. 4 show that using SVM as the base classifier guarantees, in stationary conditions, lower classification errors. Thus, the detection of a classes' swap and the identification of recurrent concepts is more successful. In fact, after the last detection, a recurrent concept is identified. In contrast, when a k-NN is used as the base classifier, the operator $\mathcal{E}$ never assesses the equivalence of concept representations and there are no false positives or true positives. In both cases, E provides the best performance and its trend is similar to results given in [26]. The NB classifier has not been considered here, as well as in *CHECKERBOARD* and *SINE_2* datasets, since the Gaussian prior is inadequate to model these classification problems. Differently, when the prior well fits the classification problem, the NB achieves the accuracy bound set by the theory (in stationary conditions) as well as the best recurrent concepts identification performance.

We emphasize that differences in the classification error of Table I between the JIT and the JIT classifier without the recurrent concept management ability are statistically significant according to a paired t-test at significance level 0.01 (except in *SCALAR_1* when using k-NN and *MULTIVARIATE_GAUSSIAN* when using NB). It follows that exploiting recurrent concepts provides substantial improvements when facing challenging classification tasks affected by recurrent concept drift (such as the *CHECKERBOARD* and *SINE_2* datasets).

The experiments show that detecting concept drift by solely monitoring the classification error might require many supervised samples and, in turn, the classifier might keep obsolete knowledge for several observations. It is clear that the JIT classifier often relies on the concept-drift detection ability of CDT_X, monitoring the input distribution. In those situations where CDT_X is not a viable option, for instance when the observations contain quantitative observations, the JIT classifier relies only on the concept-drift detection ability of CDT_ϵ. Then, it is convenient to enforce an ensemble combining the JIT with a short memory classifier such as W, to exploit the prompt reaction to concept drift that this passive classifier intrinsically guarantees (in Fig. 2 - 4 the classifier E is definitively successful in reacting to concept drift). However, in stationary conditions (i.e., between two abrupt concept drifts), the JIT classifier asymptotically tends to the Bayes error (whenever the concept drift is detected and concepts are correctly isolated), while the short memory classifier might impair the performance of the ensemble, as it emerges by analyzing the long stationary sequences of *SCALAR_1* in Fig. 2 and *MULTIVARIATE_GAUSSIAN* in Fig. 3.

VII. Conclusion

The proposed JIT approach is a flexible tool for designing adaptive classifiers able to cope with classification problems affected by concept drift. In this work we enhance the JIT classifier framework by adding, beside the concept-drift detection ability, a general formulation that includes an explicit management of recurrent concepts. When facing challenging classification problems, the proposed JIT classifier successfully exploits supervised information acquired in the past, and shows to be particularly effective if inserted within an ensemble framework.

The JIT classifier is meant for abrupt concept drifts, while different approaches could be pursued for extending this result to the gradual drift case. Here, to a first extent, the detection/adaptation paradigm does not allow to achieve the optimality like in the abrupt case. Gradual concept drift is characterized by a non-stationary condition lasting for a long time, and is perceived by the JIT approach as a sequence of abrupt concept drifts having small magnitude, causing the continuous renewal of the classifier knowledge base. A preferable approach would consist in predicting and compensating such nonstationarity, by learning the drift model. Other works will inspect techniques to enhance the concept-drift detection ability e.g., by monitoring the distribution of the classifier output on both supervised and unsupervised samples, as well as enforcing ensembles of CDTs.

Acknowledgment

This research has been funded by the European Commission's 7th Framework Program, under grant Agreement INSFO-ICT-270428 (iSense).

References

[1] J. a. B. Gomes, E. Menasalvas, and P. A. C. Sousa, "Learning recurring concepts from data streams with a context-aware ensemble," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11. New York, NY, USA: ACM, 2011, pp. 994–999. [Online]. Available: http://doi.acm.org/10.1145/1982185.1982403

[2] C. Alippi and M. Roveri, "Just-In-Time adaptive classifiers – part I: Detecting nonstationary changes," *Neural Networks, IEEE Transactions on*, vol. 19, no. 7, pp. 1145 –1153, july 2008.

[3] ——, "Just-In-Time adaptive classifiers – part II: Designing the classifier," *Neural Networks, IEEE Transactions on*, vol. 19, no. 12, pp. 2053 –2064, dec. 2008.

[4] C. Alippi, G. Boracchi, and M. Roveri, "A Just-In-Time adaptive classification system based on the Intersection of Confidence Intervals rule," *Neural Networks*, vol. 24, no. 8, pp. 791 – 800, 2011.

[5] ——, "Just-in-time ensemble of classifiers," in *International Joint Conference on Neural Networks (IJCNN 2012)*, 10 - 15 June 2012, pp. 1–8.

[6] I. Zliobaite, "Learning under concept drift: an overview," Technical report, Faculty of Mathematics and Informatics, Vilnius University: Vilnius, Lithuania, Tech. Rep., 2009.

[7] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *Neural Networks, IEEE Transactions on*, vol. 22, no. 10, pp. 1517 –1531, oct. 2011.

[8] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Trans. on Knowl. and Data Eng.*, vol. 22, no. 5, pp. 730–742, May 2010. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2009.156

[9] C. Alippi, G. Boracchi, and M. Roveri, "A distributed self-adaptive nonparametric change-detection test for sensor/actuator networks," in *Artificial Neural Networks and Machine Learning (ICANN 2011)*, ser. Lecture Notes in Computer Science, T. Honkela, W. Duch, M. Girolami, and S. Kaski, Eds. Springer Berlin / Heidelberg, 2011, vol. 6792, pp. 173–180.

[10] ——, "A hierarchical, nonparametric, sequential change-detection test," in *International Joint Conference on Neural Networks (IJCNN 2011)*, 31 2011-aug. 5 2011, pp. 2889 –2896.

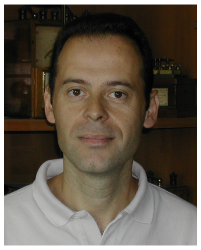
[11] ——, "An effective Just-In-Time adaptive classifier for gradual concept drifts," in *International Joint Conference on Neural Networks (IJCNN 2011)*, 31 2011-aug. 5 2011, pp. 1675 –1682.

[12] ——, "Adaptive classifiers with ICI-based adaptive knowledge base management," in *Artificial Neural Networks (ICANN 2010)*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 6353, pp. 458–467.

[13] ——, "Change detection tests using the ICI rule," in *International Joint Conference on Neural Networks (IJCNN 2010)*, 2010, pp. 1 –7.

[14] ——, "Just in time classifiers: Managing the slow drift case," *International Joint Conference on Neural Networks (IJCNN 2009)*, vol. 0, pp. 114–120, 2009.

[15] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, “Learning with drift detection,” in *Advances in Artificial Intelligence SBIA 2004*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004, vol. 3171, pp. 66–112.
[16] K. Nishida and K. Yamauchi, “Learning, detecting, understanding, and predicting concept changes,” in *International Joint Conference on Neural Networks (IJCNN 2009)*. IEEE, 2009, pp. 2280–2287.
[17] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, “Early drift detection method,” 2006.
[18] K. Nishida and K. Yamauchi, “Detecting concept drift using statistical testing,” in *Discovery Science*. Springer, 2007, pp. 264–269.
[19] J. Patist, “Optimal window change detection,” in *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*. IEEE, 2007, pp. 557–562.
[20] A. Bifet and R. Gavalda, “Learning from time-changing data with adaptive windowing,” in *SIAM International Conference on Data Mining*, 2007, pp. 443–448.
[21] L. Cohen, G. Avrahami-Bakish, M. Last, A. Kandel, and O. Kipersztok, “Real-time data mining of non-stationary data streams from sensor networks,” *Information Fusion*, vol. 9, no. 3, pp. 344–353, 2008.
[22] G. Widmer and M. Kubat, “Learning in the presence of concept drift and hidden contexts,” *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996. [Online]. Available: http://dx.doi.org/10.1023/A:1018046501280
[23] Y. Yang, X. Wu, and X. Zhu, “Mining in anticipation for concept change: Proactive-reactive prediction in data streams,” *Data Min. Knowl. Discov.*, vol. 13, no. 3, pp. 261–289, Nov. 2006. [Online]. Available: http://dx.doi.org/10.1007/s10618-006-0050-x
[24] M. Lazarescu, “A multi-resolution learning approach to tracking concept drift and recurrent concepts,” in *PRIS*, 2005, p. 52.
[25] S. Ramamurthy and R. Bhatnagar, “Tracking recurrent concept drift in streaming data using ensemble classifiers,” in *Machine Learning and Applications, 2007. ICMLA 2007. Sixth International Conference on*, dec. 2007, pp. 404 –409.
[26] I. Katakis, G. Tsoumakas, and I. Vlahavas, “Tracking recurring contexts using ensemble classifiers: an application to email filtering,” *Knowl. Inf. Syst.*, vol. 22, no. 3, pp. 371–391, Mar. 2010. [Online]. Available: http://dx.doi.org/10.1007/s10115-009-0206-2
[27] G. Widmer, “Tracking context changes through meta-learning,” *Mach. Learn.*, vol. 27, no. 3, pp. 259–286, Jun. 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1007365809034
[28] J. a. B. Gomes, E. Menasalvas, and P. A. C. Sousa, “Tracking recurrent concepts using context,” in *Proceedings of the 7th international conference on Rough sets and current trends in computing*, ser. RSCTC’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 168–177. [Online]. Available: http://dl.acm.org/citation.cfm?id=1876210.1876234
[29] J. a. B. Gomes, M. M. Gaber, P. A. C. Sousa, and E. Menasalvas, “Context-aware collaborative data stream mining in ubiquitous devices,” in *Proceedings of the 10th international conference on Advances in intelligent data analysis X*, ser. IDA’11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 22–33. [Online]. Available: http://dl.acm.org/citation.cfm?id=2075337.2075344
[30] J. a. B. Gomes, E. Menasalvas, and P. A. C. Sousa, “Improving the learning of recurring concepts through high-level fuzzy contexts,” in *Proceedings of the 5th international conference on Rough set and knowledge technology*, ser. RSKT’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 234–239. [Online]. Available: http://dl.acm.org/citation.cfm?id=1929344.1929383
[31] J. Gama and P. Kosina, “Tracking recurring concepts with meta-learners,” in *Progress in Artificial Intelligence*, ser. Lecture Notes in Computer Science, L. Lopes, N. Lau, P. Mariano, and L. Rocha, Eds. Springer Berlin / Heidelberg, 2009, vol. 5816, pp. 423–434, 10.1007/978-3-642-04686-535. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04686-535
[32] J. a. Gama and P. Kosina, “Learning about the learning process,” in *Proceedings of the 10th international conference on Advances in intelligent data analysis X*, ser. IDA’11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 162–172. [Online]. Available: http://dl.acm.org/citation.cfm?id=2075337.2075356
[33] P. Li, X. Wu, and X. Hu, “Mining recurring concept drifts with limited labeled streaming data,” *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 2, pp. 29:1–29:32, Feb. 2012. [Online]. Available: http://doi.acm.org/10.1145/2089094.2089105
[34] M. Basseville and I. V. Nikiforov, *Detection of abrupt changes: theory and application*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.
[35] G. S. Mudholkar and M. C. Trivedi, “A Gaussian approximation to the distribution of the sample variance for nonnormal populations,” *Journal of the American Statistical Association*, vol. 76, no. 374, pp. pp. 479–485, 1981.
[36] A. Goldenshluger and A. Nemirovski, “On spatial adaptive estimation of nonparametric regression,” *Math. Meth. Statistics*, vol. 6, pp. 135–170, 1997.
[37] V. Katkovnik, “A new method for varying adaptive bandwidth selection,” *IEEE Trans. on Signal Proc*, vol. 47, pp. 2567–2571, 1999.
[38] R. Johnson and D. Wichern, *Applied multivariate statistical analysis*. Prentice Hall, 2002, no. v. 1.
[39] J. C. Hsu, J. T. G. Hwang, H.-K. Liu, and S. J. Ruberg, “Confidence intervals associated with tests for bioequivalence,” *Biometrika*, vol. 81, no. 3, pp. 103–114, 1994. [Online]. Available: http://www.jstor.org/stable/2337054?origin=crossref
[40] P. Bauer and M. Kieser, “A unifying approach for confidence intervals and testing of equivalence and difference,” *Biometrika*, vol. 83, no. 4, pp. pp. 934–937, 1996. [Online]. Available: http://www.jstor.org/stable/2337298
[41] W. N. Street and Y. Kim, “A streaming ensemble algorithm (sea) for large-scale classification,” in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD ’01. New York, NY, USA: ACM, 2001, pp. 377–382. [Online]. Available: http://doi.acm.org/10.1145/502512.502568



Cesare Alippi received the degree in electronic engineering cum laude in 1990 and the PhD in 1995 from Politecnico di Milano, Italy. Currently, he is a Full Professor of information processing systems with the Politecnico di Milano. He has been a visiting researcher at UCL (UK), MIT (USA), ESPCI (F), CASIA (CN).

Alippi is an IEEE Fellow, Vice-President education of the IEEE Computational Intelligence Society (CIS), Associate editor (AE) of the IEEE Computational Intelligence Magazine, past AE of the IEEE-Tran. Neural Networks, IEEE-Trans Instrumentation and Measurements (2003-09) and member and chair of other IEEE committees including the IEEE Rosenblatt award.

In 2004 he received the IEEE Instrumentation and Measurement Society Young Engineer Award; in 2011 has been awarded Knight of the Order of Merit of the Italian Republic. Current research activity addresses adaptation and learning in non-stationary environments and Intelligent embedded systems. He holds 5 patents and has published about 200 papers in international journals and conference proceedings.

Giacomo Boracchi received the M.S. degree in Mathematics from the Università Statale degli Studi di Milano, Italy, and the Ph.D. degree in Information Technology at Politecnico di Milano, Italy, in 2004 and 2008, respectively. He was researcher at Tampere International Center for Signal Processing, Finland, during 2004-2005.

Currently, he is a postdoctoral researcher at the Dipartimento di Elettronica e Informazione of the Politecnico di Milano. His main research interests focus on mathematical and statistical methods for designing intelligent systems operating in nonstationary environments and image/video processing.

Manuel Roveri received the Dr.Eng. degree in Computer Science Engineering from the Politecnico di Milano (Milano, Italy) in June 2003, the MS in Computer Science from the University of Illinois at Chicago (Chicago, Illinois, U.S.A.) in December 2003 and the Ph.D. degree in Computer Engineering from Politecnico di Milano (Milano, Italy) in May 2007.

Currently, he is an assistant professor at the Departent of Electroncis and Information of the Politecnico di Milano. His research interests include intelligent embedded systems, computational intelligence and adaptive algorithms.