A just-in-time adaptive classification system based on the intersection of confidence intervals rule

Cesare Alippi, Giacomo Boracchi, Manuel Roveri

Dipartimento di Elettronica e Informazione
Politecnico di Milano, Milano, Italy
{cesare.alippi, giacomo.boracchi, manuel.roveri}@polimi.it

Abstract. Classification systems meant to operate in nonstationary environments are requested to adapt when the process generating the observed data changes. A straightforward form of adaptation implementing the instance selection approach suggests releasing the obsolete data onto which the classifier is configured by replacing it with novel samples before retraining. In this direction, we propose an adaptive classifier based on the intersection of confidence intervals rule for detecting a possible change in the process generating the data as well as identifying the new data to be used to configure the classifier. A key point of the research is that no assumptions are made about the distribution of the process generating the data. Experimental results show that the proposed adaptive classification system is particularly effective in situations where the process is subject to abrupt changes.

Keywords: Adaptive Classifiers, Change Detection Tests.

1 Introduction

In the real world, data coming from industrial or environmental processes change their statistical behavior over time due to thermal drifts, aging effects, transient and permanent faults. This evolutionary nature is particularly evident in sensors subject to stress such as in X-ray detectors (due to the invasive nature of the radiation), electronic noses (due to thermal and humidity effects, as well as degradation of the active film) and, more generally, in monitoring systems working in harsh environments (e.g., presence of water, dust, solar radiation, aging phenomena, etc). Data acquired after a change violate the stationary hypothesis generally assumed when designing the application solution, here assumed to contain a classification system. As a consequence, the classifier accuracy degrades, by possibly impairing the quality of service of the application.

In the related literature, the need to deal with nonstationary conditions or *concept drift* (Helmbold & Long, 1994) (Kuh & Petsche, 1990) has led to the development of classification systems able to adapt their knowledge base (i.e., training set) and, in turn, the parameters/model family to track the process evolution.

In the following, we consider a framework in which a change-detection test is applied first to identify a possible change in stationarity, and the adaptation of the classifier follows. More specifically, the classifier undergoes an adaptation phase when a change in the stationarity of the process is detected; otherwise, it integrates the new information in the knowledge-based to improve the classification accuracy over time.

The Intersection of Confidence Intervals (ICI) rule (Goldenshluger & Nemirovski, 1997) has been considered as a core technique both to design the change-detection test and manage the knowledge base of the associated classifier. A high level description of the proposed Just-in-Time (JIT) adaptive classification framework is presented in Algorithm 1. The joint use of a change-detection test and an adaptive classifier allows us for improving the classification accuracy in stationary conditions, and promptly reacting to changes in nonstationary ones. Experimental results confirm the goodness of the proposed adaptive classifier over other methods.

This work extends our previous paper (Alippi, Boracchi, & Roveri, 2010), by presenting a detailed description of the ICI-based change-detection test and by introducing a novel, modified version of the ICI-based change-detection test that combines the ICI rule with a local polynomial approximation (LPA) estimator (Section 3.4). The experimental section assesses the improvements provided by this modified test, and presents a detailed analysis of the performance of the corresponding JIT adaptive classifiers.

The structure of the paper is as follows. Section 2 critically reviews the literature of adaptive classifiers working in nonstationary environments. Section 3 introduces the ICI-based change-detection test. The knowledge-base management procedure that is activated after detecting a change is described in Section 4, while the ICI-based JIT adaptive classifier is presented in Section 5. Finally, experimental results are given in Section 6.

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

Algorithm 1: the ICI-based JIT adaptive classifier

```
1.  Configure the classifier and the change detection test;
2.  while (1){
3.      input receive new data;
4.      if (ICI-based change-detection test detects a nonstationary behavior) {
5.          Characterize the new process state;
6.          Configure the classifier and the test on the new process state; }
7.        else integrate the available information in the knowledge-base;
8.      Classify the new input samples; }
```

2 Classifiers Working in Nonstationary Environments: a Review

Classifiers able to operate in a nonstationary environment can be partitioned into three classes (Tsymbal, 2004): instance selection, instance weighting, and multiple classification systems.

The *instance selection* approach aims at identifying the samples of the knowledge base that are relevant to the current state of the process. An adaptive classification system following this approach generally uses a subset of the most recent samples (i.e., a data window) to classify the upcoming data. In addition to the straight approach, where the length of the data window is a priori fixed, several authors have suggested some heuristics to adapt the window size to the nature of the concept drift. For example, FLORA and FLORA2 (Widmer & Kubat, 1996) suggest removing a fixed 20% of the oldest training samples from the knowledge base when a change (defined as a loss in the classifier's accuracy) is suspected, while (Klinkenberg, 2004) suggests adapting the window size so as to maximize the accuracy of the classifier. In contrast, the JIT adaptive approach (Alippi & Roveri, 2008) integrates the temporal detection of changes in the data-generating process with an adaptive knowledge-base management, which involves the removal of obsolete training samples and the insertion of fresh ones into the training set.

In the *instance weighting* approach training samples are not removed from the knowledge base but, suitably weighted, take part in the classification phase. Samples can be weighted according to their age or their relevancy w.r.t. the classification accuracy of the last batch of supervised data (Klinkenberg, 2004). Experimentally, it appears that instance selection is more effective in reacting to concept drifts than instance weighting, due to the difficulty of assigning weights to past samples. (Alippi, Boracchi, & Roveri, 2009) extends (Alippi & Roveri, 2008) by proposing a joint instance selection/weighting approach: an adaptive weighted *k*-NN classifier provides a fine-grain adaptation to smooth drifts, while an instance selection phase is activated whenever a change is detected.

Finally, *Multiple Classification Systems* are based on an ensemble of classifiers whose decisions are combined by means of voting or weighting mechanisms to form the final decision (Elwell & Polikar, 2009) (Nishida & Yamauchi, 2009), (Street & Kim, 2001), (Wang, Wei, Yu, & Han, 2003), (Wang, Wang, Wu, Wang, & Shi, 2007), (Kolter & Maloof, 2003), (Kolter & Maloof, 2007). All these systems provide techniques for dynamically including new classifiers, reactivating previously created classifiers or deleting obsolete ones (i.e., pruning techniques aiming at removing the oldest classifier or the one with the lowest accuracy) (Tsymbal, 2004). In particular, in (Elwell & Polikar, 2009) a batch-learning system is proposed in which a new classifier is designed for each batch of received data and pruning methods remove the old classifiers and those with low accuracy. (Nishida & Yamauchi, 2009) suggests the use of a Bernoulli EWMA chart to assess the stationarity of the classification accuracy: when a change is detected, a new classifier is added to the multiple classification system.

The instance selection approach seems to provide the best trade-off between accuracy and computational complexity, hence making such a solution particularly interesting for embedded devices (characterized by memory, computation and energy constraints). As stated in (Klinkenberg, 2004), experimental results show that a short data window or an adaptive window provides higher classification accuracy compared to a long window solution in guaranteeing a prompt reaction to changes in the data-generating process. Unfortunately, the optimal length of the window needs to be identified a-priori by exploiting information related to the nature of the change (an impractical situation in real-world applications) whereas adaptive window solutions rely on heuristics that require a preliminary configuration phase.

Most of aforementioned classifiers exploit the classification accuracy, computed from supervised samples, to monitor the status of the data-generating process, as well as for adapting to concept drifts. Thus, these require an adequate amount of supervised samples coming from the field during the operational life to allow the on-line estimation of the classification accuracy. Differently, JIT classification systems (Alippi & Roveri, 2008), (Alippi, Boracchi, & Roveri, 2009), (Alippi, Boracchi, & Roveri, 2010), monitor process stationarity by means of a change-detection tests directly executed on the received data, without inspecting the classifier performance. Therefore, JIT classifiers can operate when limited supervised samples are provided. However, JIT classifiers are not able to perceive concept drifts that do not alter the distribution of the data-generating process, while changing the joint distribution of observations and classes: for example, JIT classifiers cannot detect a swap between the distributions of equiprobable classes, as this concept drift does not affect the distribution of the unlabeled observations. Adaptive classifiers exploiting the classification accuracy can in principle deal with similar concept drifts, although their change-detection ability always

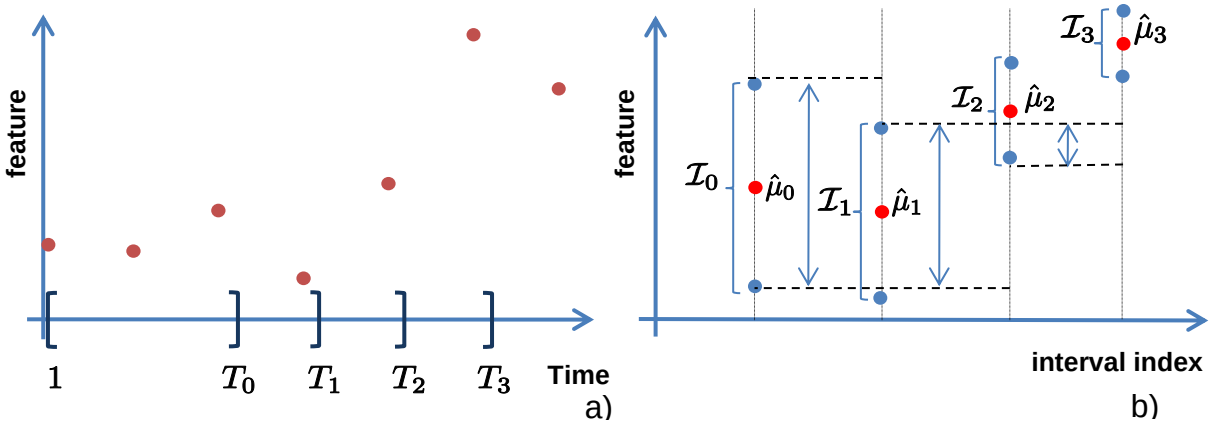


Fig. 1. An illustrative example of the ICI rule in the setting used for change detection: **a)** feature values and the set of intervals $\{[1,T_0],[1,T_1],[1,T_2],[1,T_3]\}$, **b)** the corresponding polynomial zeroth-order estimates and their confidence intervals. The ICI rule selects $i^+ = 2$ (since $\mathrm{I}_1 \cap \ldots \cap \mathrm{I}_2 \neq \varnothing$ and $\mathrm{I}_1 \cap \ldots \cap \mathrm{I}_3 = \varnothing$). The brackets in **b)** represent the confidence intervals; arrows their intersections.

depends on the amount of supervised samples available during the operational life. It follows that JIT classifiers and systems exploiting classification accuracy assume rather different operating conditions and, therefore, it is not straightforward to fairly compare these two approaches.

Here, we present a novel instance-selection based, JIT adaptive classifier able to deal with abrupt changes, such as those induced by permanent or transient faults (the drift case has been addressed by the authors in (Alippi, Boracchi, & Roveri, 2009)). The novelty resides in the use of an ICI-based change-detection test both to detect the occurrence of changes and, after each change, to adaptively dimension the size of the data window to be used as classifier knowledge-base. The interplay between the adaptive classifier and the change-detection test allows the system to improve accuracy in stationary conditions (by means of the introduction of supervised samples during the operational life) and to promptly react to changes in nonstationary ones.

3 A Change Detection Test Using the Intersection of Confidence Intervals Rule

3.1 The ICI Rule

The Intersection of Confidence Intervals (ICI) rule (Goldenshluger & Nemirovski, 1997), (Katkovnik, 1999) is a technique that regularizes data through polynomial regression computed on adaptive supports. The ICI rule operates on sequences of noisy data (or observations) $z(t) \in \mathbb{R}$ extracted from a Gaussian distribution having expectation $\mu(t)$ (the time-varying signal) and a constant standard deviation σ:

$$z(t) \sim N\left(\mu(t), \sigma^2\right) \quad t \in W, \qquad (1)$$

where W is an uniformly spaced sampling grid. Here, $U \subset \mathbb{N}$ represents the sequence of observations and, thus, t denotes the observation arrival index.

Given a time instant t, the technique generates a sequence of point-wise estimates $\{\hat{\mu}_i(t), i = 1,\ldots,L\}$ of $\mu(t)$ corresponding to a set of L nested neighborhoods $\{U_i(t),\ i = 1,\ldots,L\}$ of t, $U_i(t) \subset W$, $i = 1,\ldots,L$; each $\hat{\mu}_i(t)$ is obtained by means of an operator that fits an order m polynomial function $\mathrm{P}^{m}_{|U_i}$ to observations belonging to the neighborhood $U_i(t)$, according to the least-square method. In particular, each estimate $\hat{\mu}_i(t)$ corresponds to the value that the fitting polynomial $\mathcal{P}^m_{|U_i}$ assumes in t, i.e., $\hat{\mu}_i(t) = \mathcal{P}^m_{|U_i}(t)$. The ICI rule selects within $\{\hat{\mu}_i(t),\ i = 1,\ldots,L\}$ the value $\hat{\mu}_{i^+}(t)$ regularizing $z(t)$; such a value, computed from the ICI-selected neighborhood $U_{i^+}(t)$, is the closest among all $\{\hat{\mu}_i(t), i = 1,\ldots,L\}$ to the one providing the *ideal risk* (i.e. the minimum error achievable by fitting a polynomial of order m to the observations). The estimate $\hat{\mu}_{i^+}(t)$ satisfies the optimality criteria expressed in (Goldenshluger & Nemirovski, 1997) and is selected by trading-off bias and variance terms in the error expression.

Denote with σ_i the standard deviation of the polynomial estimator of the neighborhood $U_i(t)$ and with I_i the confidence interval of $\hat{\mu}_i(t)$

$$\mathrm{I}_i = \left[\hat{\mu}_i(t) - \Gamma\sigma_i(t)\,;\, \hat{\mu}_i(t) + \Gamma\sigma_i(t)\right], \qquad (2)$$

where $\Gamma > 0$ is a confidence parameter.

Then, the ICI rule selects that neighborhood $U_{i^+}(t) \in \{U_i(t) \;\; i=1,\ldots,L\}$ characterized by the largest index j for which the intersection $\bigcap_{i=1}^{j} \mathrm{I}_i$ is not empty.

3.2 Detecting Changes Using the ICI Rule: the General Approach

Let $X: \mathbb{N} \rightarrow \mathbb{R}$ be a stochastic process generating the data according to an unknown probability density function (pdf). Denote by $O_T = \{X(t),\, t=1,\ldots,T\}$ the sequence of observations measured up to time T, and assume that these data are independent realizations of X. Assume also that the sequence of initial T_0 observations, O_{T_0}, has been generated in a stationary condition, i.e., observations in O_{T_0} are i.i.d.. Then, the process X either remains in the initial stationary state or moves, following an abrupt change, into a novel stationary state. Denote by $T^* > T_0$ the possible change time, which is unknown.

In ICI-based change detection tests (Alippi, Boracchi, & Roveri, 2010), process stationarity is monitored by means of N features $\{F^n, n=1,\ldots,N\}$, scalar functions of subsets of observations. Each feature is designed to provide values distributed as z in (1): a change in X is detected by inspecting each feature separately using the ICI rule. As such, without loss of generality, we focus on the ICI-based change-detection test applied to a single feature.

The change-detection test uses the ICI rule to identify the adaptive neighborhood at $t=1$ and, more precisely, $U_{i^+}(1)$ is selected within the set of nested neighborhoods $\{[1,T],\, T > T_0\}$. As such, each neighborhood contains the training set O_{T_0} and the set grows by integrating new incoming samples. Refer to Fig. 1a for an illustration of the neighborhood creation mechanism.

In stationary conditions feature values are stationary, since the distribution of X does not change, and in particular the expected value of each feature is constant. In turn, this suggests the use of the ICI rule in the feature space associated with a 0^{th} order polynomial fit, which corresponds to approximating the value of each feature with a constant. The ICI rule is used to assess variations in the expected value of each feature separately, and the change-detection test notifies a change in X as soon as any feature shows such a variation.

The change-detection test needs to be configured on the training set before becoming operative. In particular, the initialization phase consists in evaluating the confidence intervals for the polynomial fit of each feature in the training set $O_{T_0} = [1,T_0]$. Let $\{F^n(t), t=1,\ldots,T_0\}$ be the sequence of values assumed by a generic feature F^n within O_{T_0}, $\hat{\mu}^n_{T_0}$ the constant that provides the best fit of these values, and $\hat{\sigma}^n_{T_0}$ the standard deviation of the corresponding estimator. The confidence interval $\mathrm{I}^{\,n}_{T_0}$ for the mean of the n-th feature in the initial stationary state is evaluated according to (2).

During the operational phase the test computes, for each time instant $T > T_0$, the zeroth-order polynomial fit for the n-th feature based on observations in $O_T = [1,T]$, and the ICI rule is used to assess its stationarity. If the intersection of all confidence intervals $\bigcap_{t=T_0}^{T} \mathrm{I}^{\,n}_t$, $T > T_0$ differs from the empty set, then the associated observations can be considered as generated by the same distribution characterizing the initial stationary state, without any change. In contrast, when $\bigcap_{t=T_0}^{T} \mathrm{I}^{\,n}_t$, $T > T_0$ becomes the empty set, the ICI rule reveals a non-stationarity in X at time T. Fig. 1b illustrates how the ICI rule selects the adaptive neighborhood thus identifying feature changes.

3.3 Features

As stated in the previous section, the proposed test monitors the distribution of X by means of N features, assuming that these are generated from a stationary Gaussian distributions when X is stationary. Thus, the ICI-based change-detection test is able to perceive only changes of X that modify the expected value of (at least) one of these features.

It turns out that by increasing the number of features employed we widen the range of process changes that can be detected. In this direction, customized features can be designed by relying on data transformations that provide values having approximately a Gaussian distribution: operatively, at first a feature of interest is extracted from the observed data, and then an ad-hoc Gaussian-approximating transformation is leveraged to define the final feature. The statistical literature presents several parametric transformations providing Gaussian distributed values out of i.i.d. samples with

transformation parameters that are directly identified from the training sequence. For instance, features can be defined using the Box and Cox (Box & Cox, 1964) transformations that guarantee approximately Gaussian distributed values when used on i.i.d. bounded values. An example of feature defined according to this two-step approach is given in (13).

Since each feature is monitored separately and a change is detected in X as soon as the ICI rule reveals a non-stationary behavior in at least one feature, increasing the number of features means also increasing the probability of having a false positive (i.e., changes detected when X is stationary). It follows that features have to be carefully selected when designing an ICI-based change-detection test by considering the trade-off between detection capabilities and false-positive rates and exploiting, whenever available, any information concerning the data-generating process.

3.4 The Polynomial Fitting Operator

The ICI rule requires the associated polynomial fitting operator to be linear and accurate on polynomial functions. Furthermore, among all polynomial fitting operators of order m, the ICI rule requires the one having minimum variance. If we select a zeroth-order polynomial fitting operator for the values of the n-th feature, we have $\mathcal{P}^0_{|U_T} = c$, $c \in \mathbb{R}$ and this constant corresponds to the one minimizing the classical squared error expression:

$$\hat{\mu}^n_T = \underset{c\in\mathbf{R}}{\operatorname{argmin}} \sum_{t\in U_T} \left(F^n(t) - c\right)^2 \tag{3}$$

The estimate $\hat{\mu}^n_T$, associated with neighborhood $U_T = [1,T]$, and the standard deviation $\hat{\sigma}^n_T$ of the corresponding estimator have the following expressions:

$$\hat{\mu}^n_T = \sum_{t=1}^{T} \frac{F^n(t)}{T} \quad \text{and} \quad \hat{\sigma}^n_T = \frac{\sigma^n}{\sqrt{T}}, \tag{4}$$

where σ^n is the standard deviation of the n-th feature. This estimate can be efficiently and iteratively computed on nested neighborhoods $O_T = [1,T]$ as follows:

$$\hat{\mu}^n_T = \frac{(T-1)\,\mu^n{}_{T-1} + F^n(T)}{T} \ . \tag{5}$$

Equation (3) refers to the minimum-variance estimator for zeroth-order polynomials.

When the ICI rule is used for change-detection purposes, it is reasonable to assign different weights to error terms in (3) e.g., by inheriting the weighting mechanism suggested by the Local Polynomial Approximation (LPA) technique. LPA (Katkovnik, Egiazarian, & Astola, 2006), (Katkovnik, Egiazarian, & Astola, 2002), is a linear and accurate operator with estimates efficiently computable by means of convolutional kernels (for this reason, it has been combined with the ICI rule in several signal and image-processing applications, (Katkovnik, Egiazarian, & Astola, 2006) (Katkovnik, Egiazarian, & Astola, 2005), (Foi, Katkovnik, & Egiazarian, 2007), (Katkovnik, Egiazarian, & Astola, 2002) , (Foi, Katkovnik, Astola, & Egiazarian, 2004)) . For instance, a weighted LPA zeroth-order estimate is defined as

$$\hat{\mu}^n_T = \underset{c\in\mathbf{R}}{\operatorname{argmin}} \sum_{t\in U_T} w_T(t)(F^n(t) - c)^2 , \tag{6}$$

with weights satisfying

$$w_T(t) \geq 0,\ \forall t \text{ and } \sum_{t\in U_T} w_T(t) = 1,\ \forall\ U_T \subset W . \tag{7}$$

The function w_T is determined by scaling a window function w, i.e. $w_T(\bullet) = w\left(\frac{\bullet}{T}\right)$. The window function w has limited support, which we assume included in $[0,1]$, and plays a central role in the LPA-ICI procedure: first of all it allows to differently weight the error terms in (6) and, second, it defines the set of nested neighborhoods $\{U_i(1),\ i = 1,\ldots,T\}$. These neighborhoods are indeed determined by the support of w suitably scaled i.e., $U_i(1) = \operatorname{supp}(w_i)$, $\forall i \geq 1$, where $\operatorname{supp}(f)$ denotes the support of the function f . The zeroth-order LPA estimate on $U_T = [1,T]$ is

$$\hat{\mu}^n_T = \frac{1}{T} \sum_{t\in U_T} w_T(t) F^n(t) , \tag{8}$$

and the standard deviation of the corresponding estimator is

$$\sigma_T^n = \sqrt{\sum_{t \in U_T} w_T(t)^2} \ . \qquad (9)$$

Experiments in Section 6 compare the change-detection performance of the ICI rule combined with the minimum-variance estimator and with a particular LPA estimator.

3.5 Detecting Changes Using the ICI rule: a Feature-Specific Algorithm

In Section 3.2 we presented the general approach for designing a change-detection test based on the ICI rule philosophy. In this section we provide a test exploiting two specific features.

This particular test acts on disjoint subsequences A_s, obtained by partitioning O_T in subsequences having length ν:

$$A_s = \{X(t), (s-1)\nu + 1 \le t < \nu s\}, \qquad (10)$$

where $s \in \mathbb{N}$ is the subsequence index. The two features operate at the subsequence level, and provide a scalar value for each A_s; the ICI rule is then combined with the minimum-variance estimator (4) or LPA (8) to detect the presence of a possible change in stationarity in the expected value of each feature.

The first feature is the sample mean M evaluated over each subsequence A_s:

$$M(s) = \frac{1}{\nu} \sum_{t=(s-1)\nu+1}^{\nu s} X(t). \qquad (11)$$

The Central Limit Theorem guarantees that, in stationary conditions, $M(s)$ approaches a Gaussian distribution. The main consequence is that $M(s)$ satisfies the hypothesis requested by the ICI rule.

To generate a second feature, we consider the quadratic deviation evaluated on each subsequence A_s:

$$S(s) = \sum_{t=(s-1)\nu+1}^{\nu s} \left(X(t) - M(s)\right)^2. \qquad (12)$$

Unfortunately $S(s)/(\nu-1)$, which is the sample variance of Y_s, is not Gaussian distributed hence preventing its use within the ICI framework. However, by applying the power-law transform

$$V(s) = \mathrm{T}\left(S(s)\right) = \left(\frac{S(s)}{\nu - 1}\right)^{h_0}, \qquad (13)$$

we obtain a feature $V(s) = \mathrm{T}\left(S(s)\right)$, which is approximately Gaussian distributed. This power-law transform has been analytically devised in (Mudholkar & Trivedi, 1981), and guarantees approximately Gaussian distribution for the sample variance computed out of i.i.d samples. The parameter

$$h_0 = 1 - \left(\kappa_1 \kappa_3\right) / 3\kappa_2^2, \qquad (14)$$

being κ_i the i^{th} cumulant of the distribution of the sample variance (refer to (Mudholkar & Trivedi, 1981) and (Alippi, Boracchi, & Roveri, 2010) for further details), which can be computed from the first six cumulants of X according to Eq. (2.6) in (Mudholkar & Trivedi, 1981).

To summarize, the change-detection test exploiting the features (11) and (13) is detailed in Algorithm 2. There, lines 1-11 represent the test configuration phase, where the values of $\hat{\mu}_{S_0}^M$, $\hat{\sigma}^M$, $\hat{\sigma}_{S_0}^M$, $\hat{\mu}_{S_0}^V$, $\hat{\sigma}^V$, $\hat{\sigma}_{S_0}^V$, and h_0 are estimated from the training set O_{T_0}. Note that at line 3 (10), $\hat{\sigma}^M$ ($\hat{\sigma}^V$) represents the estimate of the standard deviation of the feature M (V) in stationary conditions, while $\hat{\sigma}_{S_0}^M$ ($\hat{\sigma}_{S_0}^V$) represents the standard deviation of the corresponding polynomial estimator on the training set O_{T_0}.

The operational phase of the test is represented by the loop within lines 13-20, which iterates every time a subsequence A_s of ν new observations arrives. The feature values are first computed on A_s (line 16), then fitted with the respective zeroth-order polynomials, and finally the standard deviations of the corresponding estimators are computed (line 17, 18). Afterwards, the ICI rule is applied to each feature separately, by intersecting the confidence interval of the current estimates with the intersection of all the previous ones (line 19, 20). These operations are repeated until one of these intersections become empty and in this case a non-stationary behavior of X is detected within the last observed subsequence (line 22).

We comment that, in order to use the LPA estimator instead of the minimum-variance one, we simply need to replace the formulas of $\hat{\mu}_s^M, \hat{\mu}_s^V$ (at lines 2, 9, 17, 18) with the corresponding LPA estimates (8), and formulas providing $\hat{\sigma}_s^M, \hat{\sigma}_s^V$ (at lines 3, 10, 17, 18) with the analogous expressions given by (9).

Algorithm 2 : ICI-based change-detection test

1. Compute $\{M(s), s=1,\ldots,S_0\}$, being $S_0 = T_0 / \nu$.
2. $\hat{\mu}^M_{S_0} = \sum_{s=1}^{S_0} \frac{M(s)}{S_0}$.
3. $\hat{\sigma}^M_{S_0} = \frac{\hat{\sigma}^M}{\sqrt{S_0}}$, $\hat{\sigma}^M = \sqrt{\sum_{s=1}^{S_0} \frac{\left(M(s) - \hat{\mu}^M_{S_0}\right)^2}{S_0 - 1}}$
4. Define $\mathrm{I}^M_{S_0} = \left[\hat{\mu}^M_{S_0} - \Gamma\hat{\sigma}^M_{S_0} \; ; \; \hat{\mu}^M_{S_0} + \Gamma\hat{\sigma}^M_{S_0}\right]$.
5. Compute the first six cumulants of X from O_{T_0} and κ_i, $i = 1, \ldots, 3$ from Eq. (2.6) in (Mudholkar & Trivedi, 1981)
6. Compute h_0 as in (14), and define T .
7. Compute $\{S(s), s=1,\ldots,S_0\}$.
8. Compute $\{V(s) = \mathrm{T}(S(s)), s=1,\ldots,S_0\}$.
9. $\hat{\mu}^V_{S_0} = \sum_{s=1}^{S_0} \frac{V(s)}{S_0}$.
10. $\hat{\sigma}^V_{S_0} = \frac{\hat{\sigma}^V}{\sqrt{S_0}}$, $\hat{\sigma}^V = \sqrt{\sum_{s=1}^{S_0} \frac{\left(V(s) - \hat{\mu}^V_{S_0}\right)^2}{S_0 - 1}}$.
11. Define $\mathrm{I}^V_{S_0} = \left[\hat{\mu}^V_{S_0} - \Gamma\hat{\sigma}^V_{S_0} \; ; \; \hat{\mu}^V_{S_0} + \Gamma\hat{\sigma}^V_{S_0}\right]$.
12. Set $s = S_0$.
13. **while** $\left(\mathrm{I}^M_s \neq \varnothing \;\&\&\; \mathrm{I}^V_s \neq \varnothing\right)\{$
14. Set $s = S_0 + 1$.
15. Wait for ν observations, until A_s is populated.
16. Compute $M(s)$ and $V(s)$ from observations in A_s.
17. Compute $\hat{\mu}^M_s = \frac{(s-1)\cdot\hat{\mu}^M_{s-1} + M(s)}{s}$ and $\hat{\sigma}^M_s = \frac{\hat{\sigma}^M}{\sqrt{s}}$.
18. Compute $\hat{\mu}^V_s = \frac{(s-1)\cdot\hat{\mu}^V_{s-1} + V(s)}{s}$ and $\hat{\sigma}^V_s = \frac{\hat{\sigma}^V}{\sqrt{s}}$.
19. $\mathrm{I}^M_s = \left[\hat{\mu}^M_s - \Gamma\hat{\sigma}^M_s \; ; \; \hat{\mu}^M_s + \Gamma\hat{\sigma}^M_s\right] \cap \mathrm{I}^M_{s-1}$.
20. $\mathrm{I}^V_s = \left[\hat{\mu}^V_s - \Gamma\hat{\sigma}^V_s \; ; \; \hat{\mu}^V_s + \Gamma\hat{\sigma}^V_s\right] \cap \mathrm{I}^V_{s-1}$.$\}$
21. Detect a change in $\left[(s-1)\nu + 1 \, , \, s\nu\right]$.

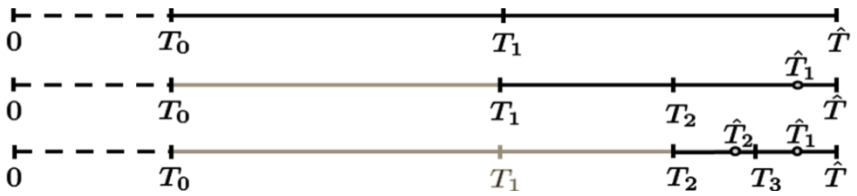


Fig. 2. ICI-based knowledge-base management procedure: an example with $\lambda = 2$. Initially, (first line) a change is detected in $\hat{T}$, and the procedure starts by computing $T_1 = (T_0 + \hat{T})/2$. The test is thus executed on $[0, T_0] \cup [T_1, \hat{T}]$, resulting in a detection at $\hat{T}_1$ (second line). This procedure iterates by computing $T_2 = (T_1 + \hat{T})/2$ and running the ICI change-detection test on the $[0, T_0] \cup [T_2, \hat{T}]$ interval. The procedure terminates when $T_3 > \hat{T}_2 (= \min\{\hat{T}_j\})$. The output is $T_{\text{ref}} = \hat{T}_2$, and $[T_2, \hat{T}]$ is assumed to be generated in the novel stationary state of X.

3.6 Accuracy of the ICI-based Change Detection Test

Simulations presented in (Alippi, Boracchi, & Roveri, 2010) applied to both synthetic and real datasets show that the ICI-based change-detection test provides a high detection accuracy, promptness in detecting changes, and a reduced computational complexity w.r.t. to other nonparametric change detection tests. Moreover, the test preserves acceptable detection abilities even when a reduced training set is available and, thus, it represents a particularly promising solution for being included in JIT adaptive classifiers.

Unfortunately, as presented later (in particular in Fig. 4 of Section 6.1), the ICI-based change-detection test shows a mean delay (i.e., the number of observations required to detect an occurred change) which increases with the change time T^*. Of course, this is an undesirable behavior that can be however attenuated by combining the ICI rule with a proper LPA estimator, instead of considering the minimum-variance estimator, as discussed in Section 6.1.

Another element that influences the way the mean delay increases w.r.t. T^* is Γ. In fact, Γ allows for balancing the mean delay with false positives (i.e., changes detected when there is no change) and, hence, influences the performance of the change-detection test. More specifically, high values of Γ reduce the mean delay at the expenses of an increment in false positives; conversely, low values of Γ reduce the false positives but increase the mean delay.

4 Reconfiguring the Classifier and the Test: the ICI-based Management Procedure

As stated in Section 1, the goal of the change-detection phase is to detect variations in the data generating process to track the change dynamics with the adaptive classifier. In turn, tracking requires reconfiguration of both the classifier and the change-detection test and it is carried out by exploiting the availability of "fresh" knowledge (i.e., supervised samples associated with the new state of the process).

A fixed sliding window opened over the last supervised training samples (e.g., see (Alippi & Roveri, 2008)) is an immediate but not optimal solution, as this may include data belonging to the previous (old) state. Differently, an adaptive window would solve the problem, by preserving only data coming after the change-time instant. Of course, an accurate identification of the time instant associated with the change in the data-generating process is of paramount importance.

This section presents an ICI-inspired procedure for obtaining, after detecting a change, an improved estimate of the change time T^*, hence mitigating the systematic delays delineated in Section 3.6, and illustrated in Fig. 4. Such refined estimate of T^*, denoted by T_{ref} in the following, allows identifying the new knowledge-base representing the current state of the process. The procedure is illustrated in Algorithm 3.

Whenever the ICI change-detection test reveals a change at time instant $\hat{T}$ we split the interval $[T_0, \hat{T}]$ in two intervals $[T_0, T_1]$ and $[T_1, \hat{T}]$, with $T_1 = T_0 + (\hat{T} - T_0)/\lambda$ defined according to the user-set parameter $\lambda > 1$ (line 2). The change-detection test is then applied to $[0, T_0] \cup [T_1, \hat{T}]$ (line 5), providing a detection at time $\hat{T}_1$. Note that $\hat{T}_1$ is considered a more accurate estimate of T^*, since the test has operated on a shorter sequence w.r.t. the initial detection $\hat{T}$ (thus reducing the systematic delay phenomenon). The interval $[T_1, \hat{T}]$ is further split into two intervals $[T_1, T_2]$ and $[T_2, \hat{T}]$ where $T_2 = T_1 + (\hat{T} - T_1)/\lambda$ (line 6). If $T_2 > \hat{T}_1$, the procedure stops, and $T_{\text{ref}} = \hat{T}_1$.

Algorithm 3: ICI-based knowledge-base management procedure

1. Let $\hat{T}$ be the ICI-based change-detection test output;
2. Compute $T_1 = T_0 + (\hat{T} - T_0)/\lambda$;
3. $i = 1$; `continue = true`;
4. **`while`** (`continue == true`){
5. Apply the ICI- based change-detection test to $[0, T_0] \cup [T_i, \hat{T}]$, detecting $\hat{T}_i$;
6. Compute $T_{i+1} = T_i + (\hat{T} - T_i)/\lambda$;
7. Define $T_{\min} = \min(\{\hat{T}_j\}_{j=1,\ldots,i})$;
8. **`If`** ($T_{\min} < T_{i+1}$)
9. `continue = false`;
10. i++;
11. }
12. Define $T_{\text{ref}} = T_{\min}$ and $T_0 = \hat{T}$.

Otherwise, the procedure iterates: at the i-th iteration the test is executed on $[0, T_0] \cup [T_i, \hat{T}]$, providing $\hat{T}_i$ (line 5). The interval $[T_i, \hat{T}]$ is then split by $T_{i+1} = T_i + (\hat{T} - T_i)/\lambda$ (line 6). The procedure ends when T_{i+1} is larger than $T_{\min}$, the earliest detection identified during the iteration of this procedure (line 7). Finally, $T_{\min}$ is the refined estimate of T^* (i.e. $T_{\text{ref}} = T_{\min}$) and is more accurate than $\hat{T}$, since it has been obtained by analyzing fewer observations. It turns out that observations within the interval $[T_{\text{ref}}, \hat{T}]$ can be considered as generated in the new stationary state of X. An illustrative example of the change-detection refinement procedure is shown in Fig. 2.

The choice of λ determines the partition of the interval $[T_0, \hat{T}]$: the lower the λ, the smaller the neighborhood of $\hat{T}$ where the search of T_{ref} is constrained. However, when λ is too small (i.e., close to one), the interval $[T_1, \hat{T}]$ may not contain T^*, and the management procedure would then provide a not accurate value of T_{ref}. Conversely, when λ is large, the ICI change-detection test is executed several times, and the probability of having a false positive in the management procedure (i.e., $T_{\text{ref}} < T^*$) increases. Experimental results presented in Section 6 show how the presence of false positives in the knowledge-base management procedure may influence the classification accuracy. The value of λ has been experimentally fixed to $\lambda = 2$.

5 The ICI-based JIT Adaptive Classifier

The joint use of the ICI change-detection test and the knowledge-base management procedure presented in Section 4 allows us for devising a novel adaptive classification system following the philosophy of the JIT soft adaptive classifier delineated in (Alippi, Boracchi, & Roveri, 2009). Similarly to the JIT soft classifier, classification is performed with a *k*-NN classifier, while the assessment of stationarity and the identification of the current knowledge-base are performed through the ICI-based change-detection test.

The proposed ICI-based JIT adaptive classifier is presented in Algorithm 4. The sequence $Z_T = \{(X(t), Y(t)), t \in I_T\}$ represents the knowledge-base of the classifier at time *T*, which contains the supervised couples $(X(t), Y(t))$: $Y(t)$ is the classification label associated with the observation $X(t)$, and I_T is the set containing the arrival times of supervised samples.

Initially, the knowledge-base of the *k*-NN classifier is $Z_0 = \{(X(t), Y(t)), t \in I_0\}$ (line 1), being $I_0 = \{1, \ldots, T_0\}$, while the value of *k* is set to k_{LOO}, estimated by means of the Leave-One-Out (LOO) technique on Z_0 (line 2). The ICI-based change-detection test is configured on the initial training set O_{T_0} (line 3).

After this configuration phase, the algorithm works on-line by classifying upcoming samples and by introducing, whenever available (line 6), new supervised data $(X(t), Y(t))$ into the knowledge base of the classifier. In this case, the algorithm stores in I_T the time instant *t* when the sample has been received (line 7), includes the pair $(X(t), Y(t))$ in Z_T (line 8), and updates the parameter *k* as in (Alippi & Roveri, 2008), i.e., according to:

$$k = k_{\text{LOO}} \left(\frac{|Z_T| + 1)}{|Z_T|} \right)^{\frac{4}{d+4}}, \qquad (15)$$

Algorithm 4: ICI-based JIT adaptive classifier (x)

1. $I_0 = \{1,..,T_0\}$, $Z_0 = \{(X(t), Y(t)), t \in I_0\}$;
2. estimate k_{LOO} by means of LOO on Z_0, and set $k = k_{\text{LOO}}$.
3. configure the ICI-based change-detection test using O_{T_0};
4. $t = T_0 + 1$;
5. **while** (1) {
6. **if** (new knowledge on $X(t)$ is available) {
7. $I_t = I_{t-1} \cup \{t\}$;
8. $Z_t = Z_{t-1} \cup \{(X(t), Y(t))\}$;
9. update k using (15). }
10. **else** {
11. $I_t = I_{t-1}$;
12. $Z_t = Z_{t-1}$; }
13. **if** (ICI test $(\text{sequence contatinig } X(t))$ == "X is NOT stationary") {
14. run the knowledge-base management procedure (Algorithm 3);
15. configure the test using $[T_{\text{ref}}, T_0]$ as training set;
16. set $I_t = \{t \in I_t, t > T_{\text{ref}}\}$;
17. set $Z_t = \{(X(t), Y(t)), t \in I_t\}$;
18. estimate k_{LOO} by means of LOO on Z_t, and set $k = k_{\text{LOO}}$. }
19. classify $X(t)$ using $k - NN(X(t), k, Z_t)$;
20. $t = t + 1$; }

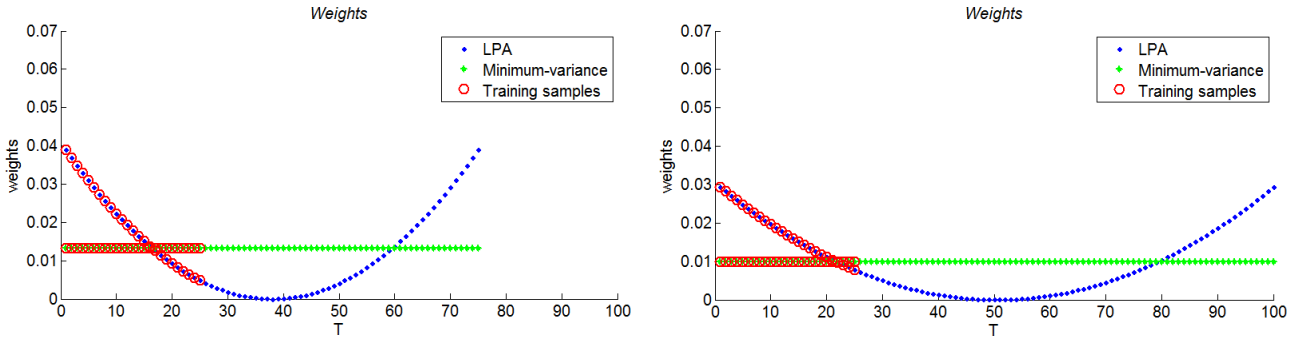
being $|Z_T|$ the cardinality of the knowledge-base, and d the dimension of the observations (here d = 1). In stationary conditions, the classification accuracy can be always increased by introducing additional supervised samples during the operational life (Fukunaga, 1972). In contrast, when $X(t)$ is not supervised, I_t and Z_t are not updated (lines 11-12).

When the ICI-based change-detection test notifies a variation in the subsequence containing $X(t)$ (line 13), the ICI-based knowledge-base management procedure is activated, providing T_{ref} (line 14). The ICI-based change-detection test is then reconfigured on the observations received within $[T_{\text{ref}}, t]$ (line 15), which are seen as generated by X in the novel status. This information is then used to remove those training samples acquired before T_{ref} both from I_t and from Z_t (lines 16, 17). This is the main difference w.r.t. the JIT adaptive classifiers presented in (Alippi & Roveri, 2008) where the window size was fixed and a-priori defined by the user. The new value of k_{LOO} is then estimated by means of the LOO procedure on the new knowledge-base (line 18), and the value of k is set to k_{LOO}. Finally, $X(t)$ is classified by relying on the updated knowledge-base Z_t, and the current value of k (line 19).

6 Experiments

To assess the performance of the proposed ICI-based JIT adaptive classifier, we performed an experimental campaign to investigate the change-detection effectiveness, the performance of the knowledge-base refinement procedure, and the classification accuracy of the proposed classifier.

First, we evaluated the impact of using LPA instead of the minimum-variance estimator to fit a polynomial to the feature values in the ICI-based change-detection test (Section 6.1); second, in Section 6.2, we show the effectiveness of the knowledge-based management procedure, which was presented in Section 4. Finally, in Section 6.3, we assess the improvements in the classification accuracy provided by the classifier suggested in Section 5.



a) Weights used in U_{75}. **b)** Weights used in U_{100}.

Fig. 3. A comparison of LPA and minimum-variance weights for $U_{75}=[1,75]$ and $U_{100}=[1,100]$. The weights corresponding to the training set values $O_{T_0}=[1,25]$ are marked with a circle. As one can see, larger weights are assigned to feature values that are associated with both the training set and the most recent values.

In our experiments we consider the following figures of merit.

- Mean delay (MD): which is an estimate of the expected detection delay, i.e., $\mathrm{E}[|\hat{T}-T^*|]$.
- False positives rate (FPR): measured as the percentage of detections not corresponding to a change in X (false positives).
- Classification error: the percentage of misclassification at each time instant.

Simulations have been averaged over 200 realizations of X.

6.1 LPA vs. Minimum-Variance Polynomial Estimator

This section presents simulations aiming at assessing the effectiveness of the minimum-variance polynomial estimator and LPA to fit a constant to the feature values in the ICI-based change-detection test. Effectiveness is evaluated in terms of MD and FPR (since there are no false negatives). The LPA weights are given by the quadratic function

$$w(t)=\begin{cases}4t^2-4t+1, & t\in[0,1]\\ 0 & \text{else}\end{cases} \qquad (16),$$

and each interval U_i, $i>1$ is identified by the support of the scaled function w_i, which is also centered to guarantee that the leftmost point of the interval is in $t=1$. Each function w_i is normalized and the weights of the error expression (6) are determined by sampling w_i in the time instants belonging to the interval U_i. Fig. 3 shows an example of these weights for two different intervals: as one can see, this window function assigns larger weights to the error terms in (6) that are associated with both the training set and the most recent values.

In this experiment we focus on the polynomial fitting operator and the ICI rule, and we do not perform any feature extraction: thus, observations (i.e., feature values) are drawn from a Gaussian stationary process (pdf $N(0,1)$) that undergoes an abrupt change affecting its mean (hence switching to pdf $N(1,1)$). The considered change times vary between sample 1000 and 200000, while the training set is always composed of the first 500 samples. The Γ parameter for the minimum-variance estimator and the LPA has been set to Γ=2 and Γ=3 respectively.

Fig. 4 shows that LPA reduces the increase of the MD as a function of T^* w.r.t. the minimum-variance estimator. We stress that such improvement cannot be compensated by setting a smaller value of Γ in case of the minimum-variance estimator. In fact, smaller values of Γ would necessarily result in an increased FPR, while the LPA estimator outperforms the minimum-variance estimator both in terms of MD (as shown in Fig. 4) and FPR (as shown in Fig. 5). The reason for such improvement is the weighing mechanism provided by LPA, which allows the ICI rule to be prompter in detecting changes, by better emphasizing variations between the leftmost samples in the interval (i.e., the training samples) and the rightmost samples in the interval (i.e., the most recent samples), in contrast to the minimum-variance estimator, which assigns the same weight to all the considered samples.

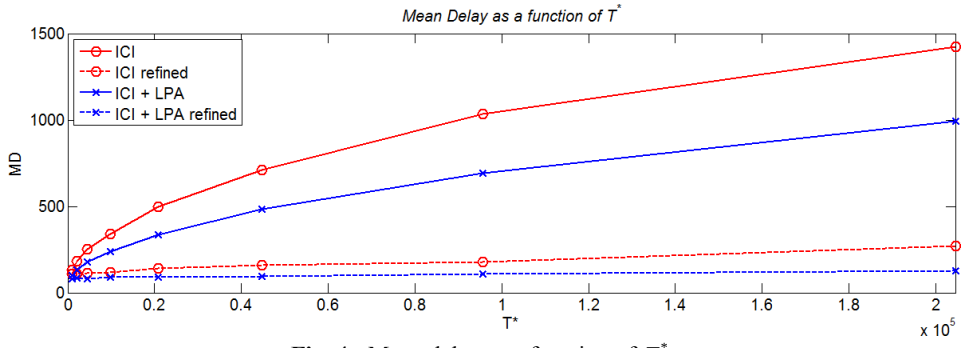


Fig. 4 - Mean delay as a function of T^*.

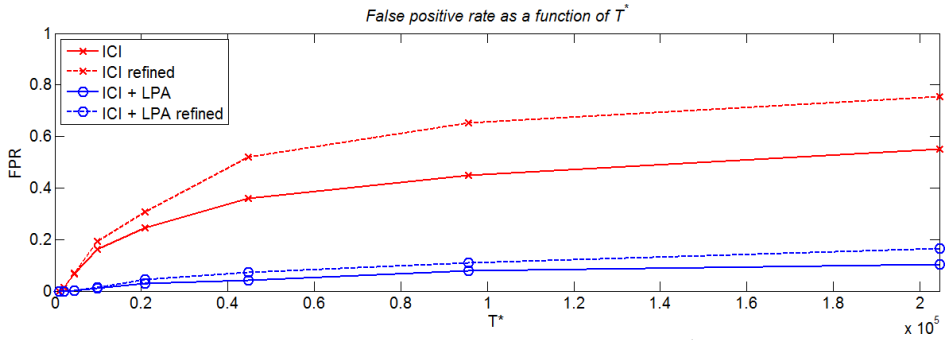


Fig. 5 - False positives rate as a function of T^*.

6.2 ICI-based Knowledge-base Management Procedure

The simulation results presented in the previous section show that the mean delay increases with T^*. Although the considered LPA performs better than the minimum-variance estimation, its MD still increases with T^*. Even if this problem cannot be solved, the knowledge-base management procedure proposed in Section 4 allows the test to provide, once a change is detected at time $\hat{T}$, a refined estimate T_{ref}. This does not result in a prompter detection but allows for defining the knowledge-base subset that is up to date with the current state of the process.

To reduce the probability of having a false positive in the knowledge-base management procedure (as this iterates the change-detection test several times), we set values of Γ larger than the one in the test; in particular, we set Γ=3 and Γ=3.5 for the minimum-variance and the LPA estimator, respectively; we also set $\lambda = 2$. The simulation results presented in Fig. 4 and Fig. 5, obtained from the same experimental framework of Section 6.1, show the great improvement in terms of reduced MD and FPR provided by the knowledge-base management procedure (Algorithm 3). More specifically, Fig. 4 shows that the knowledge-base management procedure both effectively reduces MD when T^* increases, and provides a more accurate estimate T_{ref} of T^* than the sole LPA ICI. In fact, MD is always smaller than the one of the sole test: this allows the suggested procedure for effectively identifying the training samples that are up to date with the current state of the process and that can be used for reconfiguring both the classifier and the test.

Fig. 5 shows that the refinement procedure slightly increases the number of false positives w.r.t. to the sole test, but this drawback is well compensated by the significant reduction in the mean delay. Note, however, that, in contrast to from the test associated with LPA, the increase in the false positives of the test associated with the minimum-variance estimator is far from being negligible.

6.3 The ICI-based JIT Adaptive Classifier

The performance of the proposed ICI-based JIT adaptive classifier (exploiting both the minimum-variance estimator and the LPA estimator in the change-detection test) has been compared with those of JIT (Alippi & Roveri, 2008) and JIT soft (Alippi, Boracchi, & Roveri, 2009) in the case of synthetically generated data (Application D1), and measurements coming from photodiodes (Application D2).

Application D1 contains three classification datasets characterized by different concept drifts: *abrupt, transient, stairs*. Each dataset is composed of 200 sequences of 24000 real-valued observations drawn from two equiprobable Gaussian-distributed classes (ω_0, ω_1) that, in the initial stationary conditions, are distributed as $p(x \mid \omega_0) = N(0,3)$, and $p(x \mid \omega_1) = N(4,3)$. In the *abrupt* case, the non-stationarity occurs in the middle of each sequence and increases of $+\delta$ the mean of both classes: we considered three values, $\delta = 0.3, 1.5, 3$. In the *transient* dataset, the mean of both classes increases by $\delta = 3$ after one third of the dataset and then returns to the original value after two thirds of the dataset. The

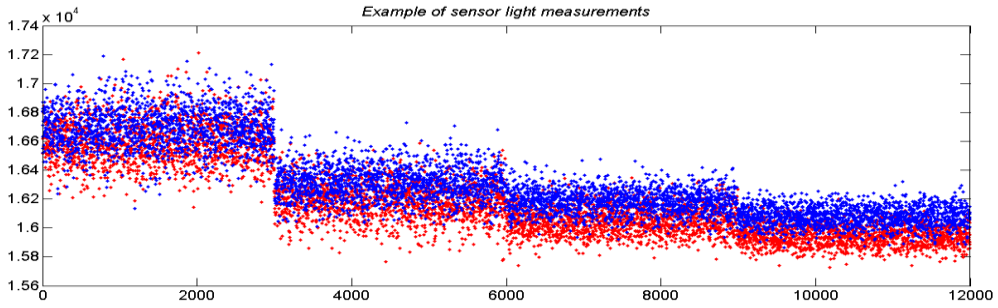


Fig. 6 An example of dataset for Application D2**.**

stairs dataset is characterized by a concatenation of changes at one fourth, two fourths (one half) and three fourths of the sequence; each changes increases both classes' mean of $\delta = 6$.

Application D2 refers to a dataset composed of 28 sequences of measurements taken from couples of photodiodes. Each sequence is composed of 12000 16-bit measurements (6000 per sensor). We tested the algorithms by classifying the observations according to the sensor. An example of such a sequence is shown in Fig 6.

The length of the initial training set is $T_0 = 500$ samples; after time T_0 we provide each classifier with one supervised observation out of five samples coming from the data-generating process to update the knowledge base. We imposed a minimum size of 80 observations to the training sets adaptively identified by both the ICI-based JIT adaptive classifiers; the JIT soft has been configured with a minimum training set size of 80 observations for the classifier and 400 for the test (as required in (Alippi & Roveri, 2008)), while the JIT requires 400 observations both for the classifier and the test (as stated in (Alippi & Roveri, 2008)). The values of Γ have been set as in Section 6.1 and 6.2, and we also preserved $\lambda = 2$.

The effectiveness of the each classifier is measured by the classification error at time t: Fig. 7 presents these percentages averaged over a window of the 1000 previous values. These plots show a comparison among the classification errors of the four considered classifiers.

In stationary conditions (i.e., before the change), the classification error typically decreases thanks to the introduction of additional supervised samples, approaching the Bayes error. Thus, any detection (false positive) results in an unnecessary removal of new training samples, which may significantly reduce the classifier accuracy. In particular, the JIT soft shows the highest classification error due to the fact that false positives significantly reduce the knowledge base of the classifier. On the contrary, the LPA ICI-based classifier provides the lowest classification error since, at the considered values of Γ (as shown in Section 6.1), the LPA approach guarantees lower false positives than the minimum-variance approach. In stationary conditions, the JIT also yields good detection accuracy, as its training set after a change is always composed of at least 400 samples, thus attenuating the loss of knowledge-base associated to false positives (the JIT and JIT soft rely on the same the change-detection test).

The *abrupt* datasets (Fig. 7) shows that in nonstationary conditions (i.e., when an abrupt change occurs in the data generating process), the ICI-based classifier exploiting LPA has a lower classification error w.r.t. the minimum-variance estimator thanks to the prompter detections provided by the associated test (see Section 6.1). After the change, the knowledge-base management procedures of both these classifiers (presented in Section 4 and whose effectiveness has been experimentally evaluated in Section 6.2), timely identify an adaptive training set evolving with the process and the occurring changes. It follows that both these classifiers are able to recover the classification error levels characterizing the initial stationary state: this proves that the adaptively identified knowledge-base allows the classifier to perfectly track the process dynamics. The good response of the JIT soft classifier to the change shows that the CI-CUSUM test (Alippi & Roveri, 2008) (associated with JIT soft classifiers) does not suffer of the same increase of the MD as the ICI-based change-detection tests. However, after the change, the JIT soft classifier is not able to recover the original performance: this is mostly due to its less effective knowledge-base management, and to its higher FPR (Alippi, Boracchi, & Roveri, 2010). The JIT classifier, which after the change is reconfigured with a fixed window containing the last 400 observations, shows the highest classification error as it likely preserves samples not coherent with the current state of the process within the knowledge-base, and it takes much longer to recover the original performance. The classifier behavior is consistent when different amplitudes of change are considered, however, at low values of δ, the classification error induced by detection delays is less significant, and the plots are mostly influenced by false positives.

When the nonstationary behavior is characterized by a sequence of abrupt changes (the *transient* and *stairs* datasets in Fig. 7), the improvement provided by the ICI-based JIT adaptive classifier exploiting LPA is even more evident: after the first change, the LPA ICI-based classifier successfully adapts both the classifier and the test to the novel operating conditions and thus the test is ready to detect further changes. The minimum-variance ICI-based JIT shows a similar behavior but the larger detection delays of the corresponding test result in higher classification errors right after the beginning of the change. Conversely, after the first change, the JIT and the JIT soft cannot successfully adapt to the novel operating conditions and this affects the detection abilities on subsequent states. It is interesting to note that, in the *transient* dataset, the JIT classifier outperforms the JIT soft; this is justified by the fact that obsolete knowledge may

Table 1. Average execution times (expressed in sec.) and average number of runs of the ICI-based change-detection test for each detection of the considered JIT adaptive classifiers, computed from datasets of Application D1.

	Execution Times (sec.)				test runs (#iterations)	
	ICI	ICI LPA	JIT	JIT soft	ICI	ICI+LPA
Abrupt 0.3	15.33	16.89	50.59	56.87	9.0	10.3
Abrupt 1.5	14.51	14.95	62.68	50.88	6.1	7.2
Abrupt 3	14.04	14.54	60.56	55.45	6.0	5.8
Transient	13.13	13.72	59.06	52.92	6.0	6.0
Stairs	12.64	13.34	87.69	62.74	6.1	5.9

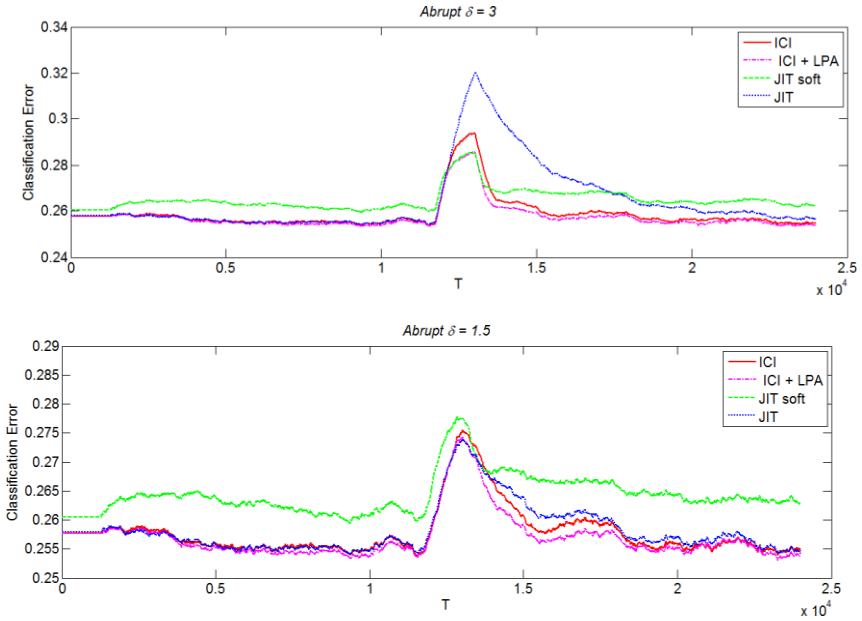
still be present in both the classifier knowledge-base and in the samples used for configuring the change-detection test after the first detection.

Experiments run on *sensor measurements* datasets (Figure 4d) show classification errors in line with the *stairs* synthetically generated datasets. In this case however, the performance of the two ICI-based adaptive classifiers is similar, as the difference in the MDs of the corresponding tests is not significant on short sequences.

6.4 Execution Times

Table 1 shows the execution times averaged on each dataset of application D1. These values refer to our nonparallelized Matlab implementations executed on a PC provided with Intel(R) Core2(TM) 2.40GHz CPU, and 4 GB of RAM. As one can see, both the ICI-based JIT adaptive classifiers are less time demanding than the other JIT classifiers based on CI-CUSUM change-detection tests and this is mainly due to the lower computational complexity of the ICI-based change-detection test, as shown in (Alippi, Boracchi, & Roveri, 2010).

The refinement procedure used in the knowledge-base management of both the proposed classifiers requires iterative runs of the ICI-based change-detection test: as such, this procedure is more resource demanding than the knowledge-base management of the JIT-soft. Then, in data-streaming applications it might be necessary to postpone the classification of few samples arrived after detecting a change, waiting for the refinement procedure to end. Although the number of test iterations depends on T^*, this can be bounded by either discarding the oldest samples in the refinement, or by decreasing the value of λ. Alternatively, it is possible to stop the refinement when a maximum number of iterations is reached. Table 1 reports also the average number of test runs (per detection) in the experiments.



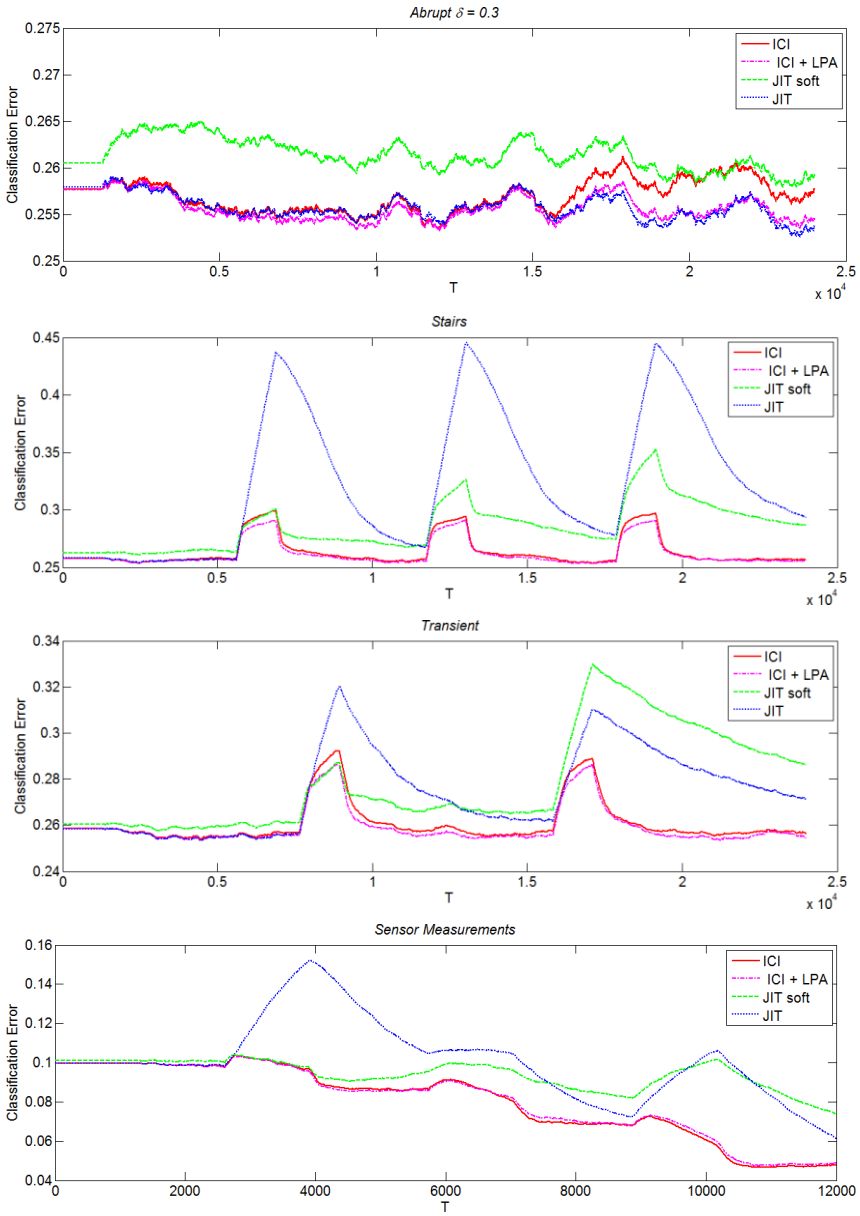


Fig. 7. Experimental results on applications D1 and D2. The classification error has been averaged over a window of 1000 values

7 Conclusions

The paper suggests an ICI-based JIT adaptive classifier able to effectively react to abrupt changes in an unknown data-generating process. The novel content is in the proposal of both the ICI-based change-detection test combined with LPA estimator, and the knowledge-base management procedure that allows the integration of the test within the JIT framework. The outcome is an adaptive classification system that is able to effectively identify, in nonstationary conditions, the training samples coherent with the current state of the process, which are then used to configure the test and to update the knowledge base of the classifier.

Experimental results show that the ICI-based JIT adaptive classifier exploiting LPA provides higher classification accuracy than the minimum-variance one, the traditional JIT and the JIT soft adaptive classifiers on both synthetically generated sequences, and luminance sensor measurements presenting abrupt perturbations. In particular, tests on sequences of abrupt changes show that the proposed ICI-based JIT adaptive classifiers are able to effectively track the process dynamics, providing a prompter adaptation to the new operating conditions than other approaches.

Ongoing works concern the extension of the proposed classifier to track process drifts, by exploiting higher-order polynomial fits within the same ICI-based and JIT framework.

References

Alippi, C., & Roveri, M. (2008). Just-in-time adaptive classifiers—Part I: Detecting nonstationarity changes. *Neural Networks, IEEE Transactions on vol.19, no.7*, 1145-1153.

Alippi, C., & Roveri, M. (2008). Just-in-Time Adaptive Classifiers--Part II: Designing the classifier. *Neural Networks, IEEE Transactions on, vol. 19, no. 11*, 2053-2064.

Alippi, C., Boracchi, G., & Roveri, M. (2009). Just in time classifiers: Managing the slow drift case. *IEEE 2009 International Joint Conference on Neural Networks*, (pp. 114-120).

Alippi, C., Boracchi, G., & Roveri, M. (2010). Adaptive Classifiers with ICI-based Adaptive Knowledge Base Management. *20th International Conference on Artificial Neural Networks* (pp. 458-467). Lecture Notes on Computer Science. Springer Berlin / Heidelberg.

Alippi, C., Boracchi, G., & Roveri, M. (2010). Change Detection Tests Using the ICI rule. *Neural Networks (IJCNN), The 2010 International Joint Conference on*, (pp. 1-7).

Box, G. E., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society, Series B 26 (2)*, 211–252.

Elwell, R., & Polikar, R. (2009). Incremental learning in nonstationary environments with controlled forgetting. *International Joint Conference on Neural Networks*, (pp. 771-778).

Foi, A., Katkovnik, V., & Egiazarian, K. (2007). Pointwise Shape-Adaptive DCT for High-Quality Denoising and Deblocking of Grayscale and Color Images. *Image Processing, IEEE Transactions on, vol. 16, no. 5,*, 1395-1411.

Foi, A., Katkovnik, V., Astola, J., & Egiazarian, K. (2004). Inverse halftoning based on the anisotropic LPA-ICI deconvolution. *TICSP Workshop Spectral Meth. Multirate Signal Process*, (pp. 49-56).

Fukunaga, K. (1972). Introduction to Statistical Pattern Recognition. New York Academic.

Goldenshluger, A., & Nemirovski, A. (1997). On spatial adaptive estimation of nonparametric regression. *Mathematical Methods of Statistics, vol 6*, 135-170.

Helmbold, D. P., & Long, P. M. (1994). Tracking drifting concepts by minimizing disagreements. *Machine Learning, vol. 14, no. 1,*, 27-45.

Katkovnik, V. (1999). A new method for varying adaptive bandwidth selection. *Signal Processing, IEEE Transactions on, vol. 47, no. 9*, 2567-2571.

Katkovnik, V., Egiazarian, K., & Astola, J. (2002). Adaptive window size image de-noising based on intersection of confidence intervals (ICI) rule. *Journal of Mathematical Imaging and Vision, vol. 16, no. 3*, 223-235.

Katkovnik, V., Egiazarian, K., & Astola, J. (2005). A Spatially Adaptive Nonparametric Regression Image Deblurring. *Image Processing, IEEE Transactions on vol. 14, no. 10*, 1469-1478.

Katkovnik, V., Egiazarian, K., & Astola, J. (2006). *Local Approximation Techniques in Signal and Image Processing.* SPIE Press.

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis, vol 8, issue 3*, 281-300.

Kolter, J. Z., & Maloof, A. M. (2003). Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift. *Third IEEE International Conference on Data Mining*, (pp. 123-130).

Kolter, J. Z., & Maloof, A. M. (2007). Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts. *The Journal of Machine Learning Research, vol 8*, 2755-2790.

Kuh, A., & Petsche, T. (1990). Learning Time Varying Concepts with Applications to Pattern Recognition Problems. *In: Signals, Systems and Computers: Twenty-Fourth Asilomar Conference* (p. 971). T. Petsche, Ed. vol. 2,.

Mudholkar, G. S., & Trivedi, M. C. (1981). A Gaussian Approximation to the Distribution of the Sample Variance for Nonnormal Populations. *Journal of the American Statistical Association, Vol. 76, No. 374*, 479-485.

Nishida, K., & Yamauchi, K. (2009). Learning, detecting, understanding, and predicting concept changes. *International Joint Conference on Neural Networks*, (pp. 2280-2287).

Street, W. N., & Kim, Y. S. (2001). A streaming ensemble algorithm (SEA) for large-scale classification. *eventh ACM SIGKDD international conference on Knowledge discovery and data mining*, (pp. 377-382).

Tsymbal, A. (2004). *The problem of concept drift: definitions and related work.* Dublin, Ireland: Trinity College.

Wang, H., Wei, F., Yu, S. P., & Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. *ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, (pp. 226-235).

Wang, P., Wang, H., Wu, X., Wang, W., & Shi, B. (2007). A Low-Granularity Classifier for Data Streams with Concept Drifts and Biased Class Distribution. *IEEE Transactions on Knowledge and Data Engineering*, 1202-1213.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning, vol. 23, no. 1,*, 69-101.