Detecting and Reacting to Changes in Sensing Units: the Active Classifier Case

Cesare Alippi, *Fellow, IEEE*, Derong Liu, *Fellow, IEEE,* Dongbin Zhao, *Senior Member, IEEE*, Li Bu

***Abstract*—The ability to detect concept drift, i.e., a structural change in the acquired datastream, and react accordingly is a major achievement for intelligent sensing units. This ability allows the unit for actively tuning the application to maintain high performance, changing online the operational strategy, detecting and isolating possible occurring faults to name a few tasks. In the paper we consider a just-in-time strategy for adaptation: the sensing unit reacts exactly when needed, i.e., when concept drift is detected. Change detection tests (CDTs), designed to inspect structural changes in industrial and environmental data, are here coupled with adaptive k-NN and SVM classifiers, and suitably retrained when the change is detected. Computational complexity and memory requirements of the CDT and the classifier, precious limited resources in embedded sensing, are taken into account in the application design. We show that a hierarchical CDT coupled with an adaptive resource-aware classifier is a suitable tool for processing and classifying sequential streams of data.**

***Index Terms*—Intelligent sensing, Active classifiers, Change detection tests, k-NN and SVM classifiers**

I. Introduction

THE adjective *intelligent* associated with a sensing unit can be inflected differently, depending on the reference community. As such, it is somehow intended as the ability to make decisions, the capability to learn from external stimuli, and the potentiality to execute computational intelligence algorithms.

The above definitions, explicitly or implicitly, rely on a computational framework receiving and processing incoming acquisitions to accomplish the requested task. We generally assume the stationarity hypothesis for data coming from an industrial or environmental process and, in turn, stationarity for the intelligent solution to be executed by the unit. In extreme cases, the assumption is so implicitly integrated in our algorithms that we even forget of its existence.

However, the real world is time variant and the stationarity assumption generally holds for short periods of time representing, da facto, a first-order hypothesis.

In fact, ageing effects affecting the readout electronics of the transducer, soft and hard faults influencing the sensor unit, changes in the phenomenon under observation (e.g., a plant) introduce changes in the process generating the data. Such concept drift represents a violation of the stationarity hypothesis assumed during the design phase of the application solution with the consequence that the application performance decreases unless adaptive strategies are taken into account.

Detecting concept drift and reacting to it is then of paramount importance in any industrial process and, from our perspective, one of the main features that an intelligent sensing unit should possess. Interestingly, our position is aligned with Piaget's psychological theory of human cognition [1], where learning is described as a constant effort to maintain or achieve balance between prior and new knowledge. As pointed out in [2], when new knowledge cannot be accommodated under existing schema because of severe conflict (i.e., non-stationarity), the need is to restructure the application to create new schemata that supplement or replace the prior knowledge base.

In the following, we consider an operational framework where a model describing the incoming data is unavailable and the application must be learned from the data themselves. Moreover, we opt for an active approach, i.e., we envisage the presence of a Change detection Test (CDT) inspecting incoming data (or derived features) to discover if new data underwent concept drift, react when the change is detected and update the application afterwards.

[35] follows this framework by proposing a mechanism where a sensor detects and responds to changes. Each sensor builds a linear model for the incoming data and adapts the sampling frequency (reaction aspect) if the new data cannot be locally explained by the model anymore (change detection aspect). There, detection can be easily cast within a CDT setup by inspecting the residual between model prediction and incoming data. A different approach investigating changes in the Nyquist frequency with a CDT has been proposed in [36] for energy-eager sensors: again the sampling frequency is adapted in real time to reduce the energy consumption of the sensing unit.

Clearly, by considering a CDT-based solution we construct a general approach whose validity is well behind that of a specific application, chosen in the sequel of classification type for its relevance in industrial applications. Assessing the quality

Manuscript received August 10, 2012; accepted January 11, 2013. This work was supported partly by the FP7 EU Project-i-SENSE Making Sense of Nonsense, Contract No:, INSFO-ICT-270428, the National Natural Science Foundation of China (Nos. 61273136, 61034002), and visiting professorship of Chinese Academy of Sciences.

Cesare Alippi is with Politecnico, di Milano, Italy and the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation Chinese Academy of Sciences, China, Derong Liu, Dongbin Zhao and Li Bu are with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation Chinese Academy of Sciences, Beijing 100190, China e-mail: cesare.alippi@polimi.it, derong.liu@ia.ac.cn, dongbin.zhao@ia.ac.cn, bulipolly@gmail.com.

of the outcome product [3], validating a sensor [4], and designing a fault diagnosis system with the fault identification ability [5] are some immediate examples of industrial applications requiring classification systems. The focus is then more on applications requiring the inspection of changes affecting the sensing datastream and its impact on the classifier output than those affecting the relationship between sensing data and the output of the classifier only. The reference setup is that of figure 1 where the datastream provided by process P is fed into the classifier and inspected by the CDT. If a change in the datastream is detected the classifier is retrained/adapted provided that supervised data are made available. Also the CDT is reconfigured to detect further possible concept drift. If no change is detected both CDT and application do not require reconfiguration. The approach is hence sensor-centered since we inspect data coming from sensors to detect a possible change.

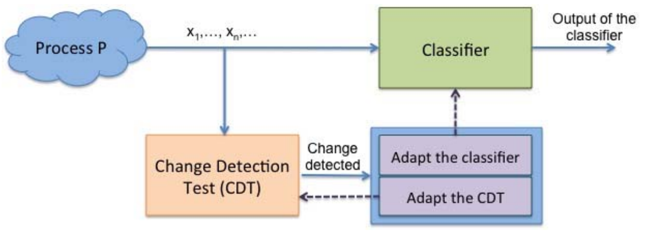


Fig. 1. The envisaged methodology. The “classifier” word can be substituted with the more general “application” one.

A different scenario, say more classifier-centered, would also inspect changes affecting the relationship assigning a class to the inputs. If this is the case, the above approach can be applied to the classification error datastream to be intended now as a virtual sensor. Interested readers can refer to [37] for a comprehensive analysis.

The paper extends conference paper [19] in several ways. First in reviewing the existing literature so as to provide a more complete and comprehensive analysis, second, in also considering the ICI (Intersection of Confidence Intervals)-CDT family recently made available in the literature, third, in providing a complete experimental setup where computational complexity and memory usage of resources-aware classifiers are contrasted.

The structure of the paper is as follows. Section II presents the related work. Section III introduces briefly the CUSUM and ICI-based family of CDTs by providing advantages and disadvantages as well as their validity range. Aspects related to the design of an adaptive active JIT are given in section IV where attention is devoted to computational and memory usage issues. Finally, the experimental section V shows under which conditions a solution should be preferred than the other.

II. Related Work

The section introduces the related work on CDTs and adaptive classifiers which can fit within a “detect and react” framework to be executed in embedded system.

A. *Change detection tests*

While there exists a large literature mostly based on statistical tests proposing parametric solutions for concept drift assessment, fewer results are available for non-parametric tests. In the parametric class we find classic textbook tests, e.g., the Student t-test and the Fisher f-test [6], addressing changes affecting the mean and the variance of the features, respectively. A conjunct test on mean and variance requires a more elaborated analysis, e.g., see [7], where a regression technique-based test is proposed.

It must be immediately pointed out that parametric tests require knowledge of the probability density functions and/or priors on concept drift [6].

Nonparametric tests are more flexible tools, which do not require unreasonable hypotheses [8]. For instance, the Mann–Whitney U-test [9] and the Wilcoxon test [10] are nonparametric tests designed to detect a single change point and cannot provide a sequential use, as sensing datastreams require. Differently, Mann– Kendall [11] and CUSUM [12] are widely used tests adequate for a sequential analysis as the recently introduced ICI test [13] and the recurrent Lepage one [34] are. Section II focuses on the ICI-CDT and CUSUM families for their effectiveness and flexibility, like the CI (Computational Intelligence)-CUSUM CDT is able to host features provided by other CDTs, e.g., Mann-Kendall. That said, it is worth point out that any CDT can be used within the suggested adaptive framework.

B. *Resource-aware Adaptive Classifiers*

As previously pointed out, here we couple the CDT test with a reference classification application. The computational intelligence literature suggests many models: Radial Basis Functions (RBF), Feed-forward Neural Networks (FF-NN), k-NN, Support Vector Machines (SVM) and a plethora of hybrid solutions; reference [14] covers them all.

However, at a more careful analysis, the computational complexity of many models does not scale well with the number of training samples. As a consequence, an online training phase required by the “reacting to change” mechanisms becomes prohibitive for sensing units characterized by limited resources, in terms of computational ability and memory (both RAM and flash).

Without the pretention to be exhaustive, Table I presents some figures for embedded architectures used in wireless sensor networks whose processors are also popular within the industry. Energy consumption is another hot topic, particularly delicate when the sensing unit is battery powered (despite the fact that energy harvesting mechanisms might be available). To keep under control power consumption, duty cycle mechanisms, adaptive sampling and CPU clock management solutions must be envisaged [15]. It is clear that embedded Hw resources do not allow the designer for considering CPU and memory eager solutions and resources-aware algorithms should be designed instead.

Passive classifiers, i.e., classification systems continuously updating the classifier, such as those based on ensembles [16] and possibly operating incrementally, e.g., [2], do not consider the computational complexity or the memory requirement as a constraint. As such, they cannot be considered, in general, as a viable solution unless very simple strategies are envisaged.

TABLE I: COMPARISON AMONG SOME OFF-THE-SHELF EMBEDDED SOLUTIONS (ENABLING THE WSN TECHNOLOGY).

Unit	Clock frequency MHz	Power consumtion (operational)	RAM memory Int/ext (B)	Flash memory Int/ext (B)
Mica Z	8 MHz	60 mW	4k/-	128/-
Telos B	8 MHz	4 mW	10k/-	48k/ 1M
Imote 2	13-416 MHz	31mW@13M 992mw@416M	256k/32M	32M/-
Start gate	400 MHz	1455 mW	64k/64M	32M/CF card

Int stands for internal memory on the chip, external refers to the memory available at the board level. – stands for not applicable.

Differently, active classification systems, e.g., those requesting a detection of a change to activate the reconfiguration mechanism, are intrinsically more sensitive to the computation issue since training is carried out only when really needed (we see the affinity with duty-cycling in CPUs).

Among active classifiers we select the Just-In-Time (JIT) framework for its versatility and flexibility [17]. A JIT classifier always exploits, during its operational life, incoming labeled data. If the process is stationary, i.e., no concept drift has been detected by the CDT, new data are used to improve the accuracy performance of the classification system so that, asymptotically with the number of available samples, it tends to the optimal Bayes's one in consistent classifiers. Conversely, when a change is detected the classifier needs to be updated by –conceptually- discarding obsolete data and be re-trained on new ones. A main feature of JIT classifiers is that abrupt concept drift always generate a classifier which asymptotically tends to the optimal one, provided that supervised samples are made available online (as in some industrial quality control analysis where an operator supervisions the process and inspects product samples for quality control). Of course, a temporary decrease in accuracy might be expected when the change is detected but, afterwards, the application reacts promptly so as to recover it. In the case of gradual concept drift (a drift type of change affecting the datastream), the CDT detects the change as a sequence of small abrupt changes. Here, the final performance depends on the ready availability of sampled couples as well as the drift rate (the higher the drift rate the lower the performance associated with the tracking ability). A solution to mitigate the gradual concept drift case has been suggested in [18].

However, since detection of concept drift requires retrain/update the classifier so as to track non-stationarity, only classifiers characterized by a complexity-aware training and recall phases, e.g., k-NN and SVM machines, can be considered as building blocks for designing resource-aware classifiers. The classifier design must be carefully chiseled also to keep under control the consumption of available resources. This aspect is tackled in section IV.

III. Change Detection Tests

Denote by $x(t)$ a sampled instance or a feature vector extracted from the sensor signal at time *t*, e.g., the mean, the variance, the coefficients of a polynomial fit, the coefficients of a linear time invariant dynamic system [31], the parameters of a reservoir network [33] or an Extreme learning machine [32] modeling the datastream and computed over a non-overlapping finite window open on the signal. In the following, the default domain of $x(t)$ is $\Re^1$ unless otherwise specified. By inspecting $x(t)$ over time we wish to identify whether a change occurred in the process generating $x(t)$ or not and provide an estimate $\bar{t}$ of the time instant t^0 the concept drift occurred.

A. CUSUM-based CDT

The CUSUM test [12] has been designed within the control community to detect changes in the probability density function (pdf) of $x(t)$. The test assumes that $x(t)$ is an independent and identically distributed (i.i.d) random variable drawn from a known pdf p_{Θ^0} parameterized in the parameter vector $\Theta = \{\theta_1, \theta_2, \ldots, \theta_\lambda\}$, $\lambda = |\Theta|$. For instance, if the pdf is Gaussian the parameter vector contains the mean and the variance of the distribution. CUSUM assumes that, following concept drift, the parameter vector Θ^0 changes to a known Θ^1 with new $x(t)$ associated with a known pdf p_{Θ^1}. The method requires then to compute

$$R(t) = \sum_{\tau}^{t} \ln \frac{p_{\Theta^1}\left(x(\tau)\right)}{p_{\Theta^0}\left(x(\tau)\right)} \tag{1}$$

over the data set and evaluate the minimum value $m(t) = min_{1 \le \tau \le t}(R(\tau))$. CUSUM detects a change at time $\bar{t}$ when $R(\bar{t}) - m(\bar{t}) > T$, T being a threshold set by the user. The strong assumptions related to the availability of the pdfs as well as the parameter configurations Θ^0, Θ^1 and T limit the applicability of the CUSUM test which, however, is particularly appreciable for its effectiveness and simplicity.

To overcome the limits of CUSUM [20] suggested the CI-CUSUM CDT. The extended test, which assumes the i.i.d hypothesis now for a vector of features $x(t) \in \Re^l$, does not require availability of p_{Θ^0} which is derived by invoking the central limit theorem. The parameter vector Θ^0 of the Gaussian pdf contains the mean and covariance matrix, while a PCA technique is considered to keep under control the size –and hence the computational complexity-, of the input features. The post-change parameter configuration Θ^1 can be generated so as to model the "we are not more in Θ^0" case and a straight CUSUM test can be applied. Threshold T is set as the maximum value $R(t) - m(t)$ estimated on the training set (or the test set if many data are available). The estimate $\bar{t}$ of the concept drift occurrence time is always an upper bound of the real one since the latency introduced by the detection method is induced by the windowing mechanisms needed to generate independent samples.

For its effectiveness and simplicity the CI-CUSUM CDT is a good candidate for designing active adaptive classification solutions following the just-in-time framework.

B. ICI-based CDT

The ICI rule [21], [22] is a method for optimally regularizing data by means of a polynomial regression computed on adaptive supports. The ICI rule operates on not overlapping sequences of observations/features ruled by $z(t)=G\left(\mu(t),\sigma^2\right)$. [13] suggests to use features such as the mean and the pooled variance estimated from non overlapping sequences of $x(t)$. Such features, thanks to the central limit theorem and an ad-hoc power transform for the estimated variance satisfy the gaussian hypothesis requested by the method.

In its current version the ICI-CDT requires streams of data that are scalar and then proceeds sequentially with the analysis. However, when the envisaged problem is multivariate the method would request the estimate of the covariance matrix, a computationally intensive operation that requires availability of a large number of samples. A viable solution is to consider a diagonal covariance matrix: in this way signals are considered independent and the basic scalar ICI-CDT can be used. Interestingly, [23] introduces a variant of the CDT that provides a refinement procedure leading to an improved estimate for $\bar{t}$.

The ICI-CDT is particular sensitive to concept drift but also introduces structural false positives as time passes. It is clear that such a problem must be removed if we are designing an adaptive application for an industrial process even if false positives basically require the classifier and the CDT to be unnecessary retrained. A solution to the problem was obtained with a hierarchical version of the ICI-CDT as presented in the following section.

C. Hierarchical CDT

False positive reduction in concept drift detection can be addressed by considering a hierarchical CDT (H-CDT). The H-CDT test is composed of two levels. The first one runs a CDT test providing an alarm (either a real concept drift or a false positive detection) and $\bar{t}$ to the second level CDT. Based on $\bar{t}$ the second level CDT partitions the datastream in two intervals characterizing the states of the process before and after the supposed change. Then, a multivariate hypothesis test based on the Hotelling's T square statistic [6] is executed.

The Hotelling test confutes or accepts the concept drift detection proposed by the first level CDT and its outcome is the final one. Basically, Hotelling verifies if the feature means (of sample mean and variance) before and after the change are the same (i.e., their statistical difference is null).

Since the ICI-CDT is particularly effective in detecting changes and is characterized by a low-computational complexity, it is a perfect tool to detect concept drift when coupled with the Hotelling test to configure a H-CDT.

Algorithm I: the Hierarchical CDT (H-CDT)

```
Configure the first-level CDT on the training set
while (1)
  acquire a new observation
  if (ICI-CDT detects concept drift)
    Estimate the time of the change $\bar{t}$
    if (Hotelling validates the change) concept drift detected
      else re-learn the parameters of ICI-CDT
  endif
endwhile
```

The high level algorithm for the H-CDT is given in Algorithm I. Interestingly, the H-CDT allows the ICI-CDT to be reconfigured online after a false positive is detected so as to improve performance over time.

IV. Resource-aware adaptive Just in Time Classifiers

Let KB be the knowledge base of the classifier, namely the set of N couples $\{x(t), y(t)\}$ available at a given instant of time to characterize it. Consider inputs $x(t) \in \Re^l$ and, without loss of generality, outputs $y(t) \in \{-1,1\}$.

A. Adaptive k-NN

The k-nearest neighbor algorithm (k-NN) is a statistic classification method where the label to be assigned to the input sample is that of the majority of its k nearest neighbors.

Algorithm II: JIT adaptive k-NN classifier

```
Estimate $k$ through LOO applied to training samples KB
Configure the k-NN classifier and the CDT
while (1)
  if (new knowledge IKB is available)
    KB=IKB $\bigcup$KB;
  endif
  if  (CDT= nonstationary)
    Remove obsolete knowledge from KB;
    Estimate $k$ by means of LOO applied to KB
    Configure the CDT
  endif
  Classification=k-NN($x$, $k$, KB);
endwhile
```

The k-NN classifier, e.g., see [24], is the simplest among the consistent classifiers (i.e., it asymptotically tends to the optimal Bayes classifier under mild hypotheses on k and N) since a proper training phase is not required. An euclidean distance is commonly used to compute the distance between two instance vectors.

Despite the fact that small values of k are generally used, in the practice one must be aware that when N increases similarly k has to (but less than N) to grant consistency. As such, for increasing values of N one needs to re-estimate k, e.g., as suggested in [17], [24].

The adaptive JIT algorithm based on k-NN is given in Algorithm II, which is easy to follow (LOO stands for Leave One Out). The key points of the algorithm can be summarized as: a) if we are in a stationary condition and new information is coming then it must be exploited to asymptotically tend towards the optimal classifier; b) when a change is detected with a CDT obsolete information must be discarded; c) the CDT must be reconfigured on the new state. Again, the approach is conceptually aligned with Piaget's theory.

B. Adaptive SVM

The SVM kernel classifier [25] is

$$f(x) = sgn(\omega \cdot x + b) = sgn\left(\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b \right) \qquad (2)$$

where parameters ω (vector normal to the separating hyperplane) and b (hyperplane offset) characterize the classifier. ω and b are determined by solving the optimization problem

$$min(\frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{N}\xi_i) \qquad (3)$$

subject to conditions $y_i((\omega \cdot x_i) + b) \geq 1 - \xi_i$, $\xi_i \geq 0$, *i=1,...,N.* $C > 0$ is a cost positive parameter, ξ_i the *i*-th slack variable and

$$K(x_i, x_j) = exp\left(-\frac{\|x_i, -x_j\|^2}{\sigma^2} \right)$$

is a kernel function, here chosen to be a radial basis function parameterized in σ. Parameters C and σ significantly influence the performance of the classifier and can be selected through K-fold cross-validation [26].

(3) can be cast in a dual form [25] leading to coefficients α_i which, when different from zero, identify the Support Vectors (SVs). SVs fully describe the classification problem given the available data and can be intended as pivot points to the classification method. Since the number d of SVs is smaller or equal (rare event) than N, we recommend to consider the SVs instead of the N couples in the KB to classify new instances. The use of SVs saves memory. To apply something similar to k-NN we should use condensing techniques [27] which, however, are very expensive in terms of requested computation.

Important for our algorithm, each supervised sample $\mathrm{x_i}, \mathrm{y_i}$ satisfies the Karush-Kuhn-Tucker (KKT) condition [25]

$$\alpha_i = 0 \Rightarrow y_i f(x_i) \geq 1;$$
$$0 < \alpha_i < C \Rightarrow y_i f(x_i) = 1;$$
$$\alpha_i = C \Rightarrow y_i f(x_i) \leq 1$$

We comment that samples satisfying the KKT conditions do not change the SVs and, as such, the structure of the classifier. It follows that, within an incremental learning strategy, only those samples violating the KKT condition should be kept for improvement and stored, the others removed. Of course, if outliers can be present, they should be identified in advance and discarded.

A JIT classifier based on SVs must then require that only those new samples violating the KKT must be added to the incremental knowledge base (IKB). Since insertion of a new sample in the SV set represents a perturbation to the method in stationary conditions, we need to merge KB with IKB and re-train the SVM when the size of IKB is above a threshold *Th* (sophisticated methods might think of evaluating the novel information content associated in IKB before retraining the classifier).

Differently, when concept drift is detected, data contained in KB become obsolete, hence negatively impacting on performance. As such they must be removed and only the most recent ones kept. The final algorithm of a SVM-based active classifier is given in Algorithm III.

Algorithm III: JIT adaptive SVM classifier

```
Configure the test CDT and train classifier SVM on training set $KB_0$;
$KB = support\ vectors\ of\ KB_0; n = |KB_0|; IKB = 0;$
While (1) {
   Acquire sample $x$;
   if(new supervised knowledge is available)
     Insert samples $(x, y)$ violating the KKT condition in $IKB$;
   endif
   if(CDT $(x)$ = nonstationary)
     $KB_0 = last\ N\ samples\ of\ new\ supervised\ samples$;
     Train SVM and CDT on $KB_0$;
     $KB$= support vectors of $KB_0$; $n = |KB_0|$;
   else
     $n_{IKB} = |IKB|$;
     if( $n_{IKB}/n > Th$)
        $KB_0 = KB_0 \cup IKB$;
        Train SVM and CDT on $KB_0$;
        $KB$= support vectors of $KB_0$;
        $n = |KB_0|; IKB = 0;$
     endif
   endif
   Classification=SVM ( $x$ , $KB$ , $C$ ,$\sigma$);
endwhile
```

C. Complexity Aspects

As shown in table II the k-NN and the SVM classifiers are particularly appealing candidates as resource-aware classifiers for their contained complexities.

Again, *N* represents the number of samples in the KB, *k* is the number of samples to be considered in the neighborhood, *d* is the number of support vectors in KB. In general, we have that $d < N$ (and in some applications $d << N$) which favors the SVM classifier. However, as pointed out in [28], the computational cost of solving the quadratic problem required by the training phase in SVM grows at least as N^2 for small *C* values and up to N^3 for large *C*, as also experimentally pointed out in [29].

TABLE II: COMPLEXITY OF K-NN AND SVM CLASSIFIERS

Complexity	k-NN	SVM
Computational (Training)	$O(1)$	$O(N^2)-O(N^3)$
Computational (Recall)	$O(kN\log N)$	$O(d)$
Memory	$O(N)$	$O(d)$

However, in many applications one hardly needs to estimate the optimal solution for the quadratic problem. Moreover, the training time for a linear SVM to reach an expected level of error actually decreases as the training set size increases [30]; as such, for practical problems one should expect a N^2 complexity.

From table II it is clear that a k-NN classifier is to be preferred for training reasons while the SVM for memory and recall. That said, the final decision depends on the particular application and on the specific balance between the train and recall activities, a critical issue in incremental active solutions since the classifier needs to be reconfigured following concept drift. At the same time, in stationarity conditions, the k-NN scales badly in memory and computation time during the recall phase. Here condensing techniques [27] should be considered to keep under control the increase in N at the cost of a not negligible computational time, which makes the approach not viable.

V. EXPERIMENTAL SECTION

The experimental section aims at comparing accuracy performance and computational and memory complexity of the CI-CUSUM and the H-CDT tests coupled with the active k-NN and the SVM classifiers. What presented is a new set of experiments, a synthetic one and an industrial one, which also extends that given in [19].

Here, deliberately, we consider at first a synthetic experiment characterized by classification complexity well above that of most real applications. The classification framework is that suggested in [14], here reported for completeness. More specifically, the experiment constructs a random pdf composed of a mixture of ten gaussians computed as follows. Generate 10 mean values μ_i^1, $i \in \{1,10\}$ from a Gaussian two-dimensional distribution $N([1,2]^T, I)$ representative of class 1 and 10 mean values μ_i^2, $i \in \{1,...,10\}$ from a multivariate Gaussian distribution $N([2,1]^T, I)$. Now, generate for each class, 200 observations by picking a random mean μ_i^1 and produce a feature sample $N(\mu_i^1, I/5)$. The same procedure is repeated for the second class. The final classification problem, composed of 400 training data, is very complex, with a strong overlap between the two classes as can be seen in Figure 2. This situation makes change detection a very complex task. During operational time new samples are provided according to the above distributions and need to be classified by the active classifiers. Perturbations are then generated in the test dataset following both the abrupt and drift change model affecting all the means of the classes.

More specifically, the abrupt change affects the $N(\mu_i^j, I/5)$, $j \in \{1,2\}, i \in \{1,...,10\}$ distributions according to a multiplicative perturbation model $\mu_{p,i}^j = \mu_i^j(1+\lambda)$ where $\lambda = \{0.1, 0.2, 0.5\}$ is the intensity of the perturbation. In the drift case the used model is $\mu_{p,i}^j = \mu_i^j(1+\lambda t/T)$ so that, at time nT incremental with n the perturbation assumes value $\mu_{p,i}^j = \mu_i^j(1+n\lambda)$. In any case the perturbations influence the magnitude of the affected value of λ.

A set of 14600 data was generated to test the methods. When a change is detected the classifier automatically reacts to track the change.

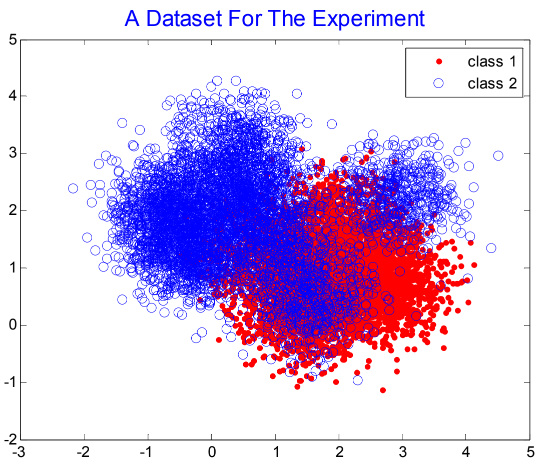


Fig. 2. A dataset for the two-dimensional synthetic experiment. In circled blue data from the first class, in crossed red those of the second class. It can be seen the complex mixture of Gaussian structure for the pdf.

The industrial dataset is composed of 28 datastreams taken from couples of photodiodes receiving, over time, X-ray radiations at four different energies. Each datastream is composed of 12000 16-bit measurements (6000 per sensor); the goal is to classify the sensor providing the measurement. The training set is composed of the first 400 instances.

The figures of merit considered to assess the performance of the CDTs are,

- False positive (FP). FP measures the percentage of changes detected by the CDT when the change is not there;
- False negative (FN). FN detects the percentage of changes missed by the CDT;
- Latency (L). Latency measures the time (in samples) required to detect a change;

and, for the classifier,

- Accuracy. Accuracy measures the accuracy of the JIT classification system over time after training (test phase);
- Computational time. The computation time measures, given a reference Hw platform (Asus Intel i5 core running @ 2.4GHz, 4G RAM), the averaged time per sample requested to run the JIT classifier;

- Memory usage. It represents the maximum number of samples over test data stored in the KB (samples data for the k-NN classifier, SVs for the SVM).

Where it does apply experiments have been averaged over 50 runs for the synthetic dataset and 28 runs for the real one.

A. Detecting Changes

The experiments on the synthetic dataset were executed as discussed above and led to results given in table III where we present the expected value of latency and, between parenthesis, the standard deviation.

As expected we see that FN and latency decrease with the increase of the intensity of the perturbation affecting mean values in both CDTs. However, the H-CDT is always better than the CI-CUSUM both in terms of FN and latency. This is associated with the fact that the Hotelling's test introduces a further control on the proliferation of false positives (we set the gamma parameter of H-CDT at 0.8 to induce a very sensitive CDT. The presence of some FPs introduce a positive conservative approach as we will see in the sequel).

TABLE III: THE COMPARED PERFORMANCE OF THE CDTS

		CI-CUSUM CDT			H-CDT		
	λ	FP	FN	L	FP	FN	L
Abrupt	0.1	0%	98%	2221(0)	26%	2%	1297 (1850)
	0.2	0%	72%	1508.1 (1139.3)	20%	0%	254.4 (444.2)
	0.5	0%	2%	446.7 (497.5)	10%	0%	186.8 (426.5)
Drift	0.1	2%	100 %	-(-)	32%	24%	3432 (3535)
	0.2	2%	98%	1081(0)	18%	2%	1124.1 (1556.7)
	0.5	0%	36%	1127.3 (761.3)	10%	0%	218.4 (413.6)

B. Reacting to Changes

The second set of experiments refers to 1) the case where inputs are affected by a series of concept drift; 2) data coming from the industrial datastreams.

A synthetic dataset

An abrupt change affecting the mean value, $\lambda = 0.5$ is injected at sample $t_1 = 3000$ followed by a drift within time interval [$t_2 = 6000$, $t_3 = 10000$] $\lambda = 1$; At time t_3 the means go back to the nominal ones and a change in the variance $\sigma_\lambda^2 = (1+\lambda)\sigma^2$, $\lambda = 0.5$ takes place.

The signature profile of the change is given in figure 3 where the abscissa shows the values assumed by the first feature over time. Detection performance of the two CDTs is shown in Figure 3 (lower plot); compared results are given in table IV. From the figure we see that H-CDT always detects the change, with an obvious delay (tests need to acquire data to provide enough confidence to support the "change" statement), while CI-CUSUM introduces false negatives. It is worth recalling that the proposed systems react to each detected change by activating the active modality for the envisaged classifier. However, at the same time, the envisaged classifiers might adapt also when the change is not detected provided that new supervised samples are provided.

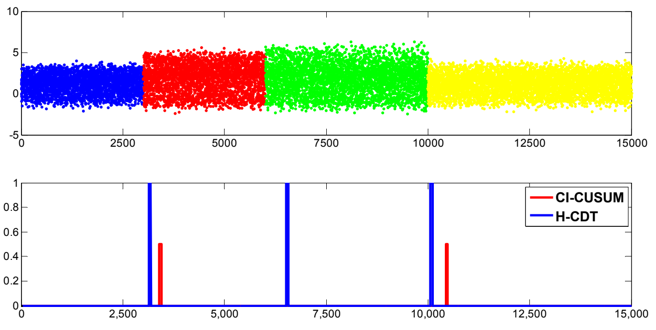


Fig. 3. The time profile of the change (upper plot), and the instants of time a change is detected by the CI-CUSUM and the H-CDTs (lower plot). In the abscissa we have the test samples.

TABLE IV: COMPARED PERFORMANCE ON THE SYNTHETIC DATASET

Method	Classification Accuracy	Computation Time (ms)	Memory Usage (couples)
SVM + CI-CUSUM	82.28%(0.008)	1.062(0.080)	695(148.3)
SVM+H-CDT	81.94%(0.010)	0.440(0.079)	371(69.7)
k-NN + CI-CUSUM	83.25%(0.005)	1.096(0.088)	2214(353.4)
k-NN+H-CDT	82.02%(0.013)	0.721(0.139)	1298(235.1)
Static k-NN	81.26%(0.013)	0.095(0.005)	400(0)
Static SVM	79.46%(0.012)	0.032(0.003)	154(11.4)

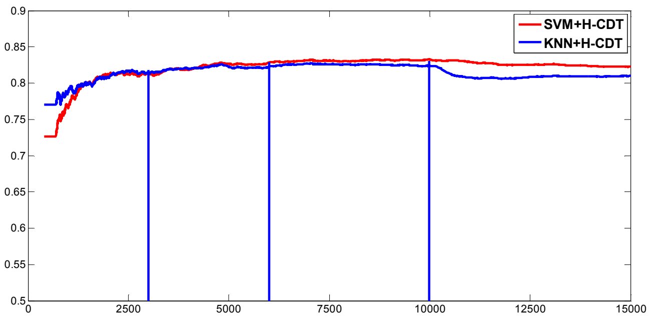


Fig. 4. The average accuracy of the classifiers over time; the reference CDT is the H-CDT.

Results are given in table IV; computation time must be intended as the expected computation burden per sample. Notation X(Y) refers to expected value X with standard deviation Y. Static k-NN and static SVM refer to the case where the classifiers are static and not adaptive.

From the table we see that the static versions of the classifiers are less performing than the adaptive ones and that the best performing classifier is the k-NN+CI-CUSUM. We comment that the datastream contains three stationary plateau where the process is stationary and recall that, in stationary conditions, the JIT classifier with k-NN tends towards the optimal Bayes one. Even if CI-CUSUM is less effective in detection, the supervised samples are inserted in the KB and mitigate occurrences of false negatives. However, from the complexity point of view the solution SVM+H-CDT is clearly preferable than the others with a 63% improvement in computation time w.r.t the second best one (also based on H-CDT) and a 87% in memory savings (compared to the SVM+CI-CUSUM solution). The light computation required by H-CDT (about 0.06ms to run the two levels hierarchical test) vs. the CI-CUSUM (0.5ms) makes the difference in the computation cost. This is a direct consequence of CI-CUSUM, which is a computationally intensive CDT. In fact, in order to keep generality w.r.t. the pdf generating the datastream (or the derived features) it needs to compute several numerical features. In particular, in addition to the mean and the variance values, the CDT computes features related to the pdf and its cumulative density function as well as others inspired by the Mann–Kendall and the CUSUM CDTs. A pdf-free test, as CI-CUSUM is, provides a very high flexibility and generality in the pdf at the cost of an increased computational load and latency as we have seen in table III. Differently, ICI inspects solely the mean and the transformed variance of *x(t)*, which are features characterized by a Gaussian distribution. Such implicit priors impact both on results and computational complexity.

When the best active SVM-based and k-NN solutions are compared, the former introduces a gain on the latter of about 250% in memory savings. We refer to table III to further investigate results. As far as memory is concerned we should always expect SVM to be better than k-NN since *d*, the number of support vectors, is smaller-equal than *N*, the number of available samples. However, when computational complexity is envisaged, we cannot decide a priori which classifier should be considered. In fact, computational complexity is function of the number of times a change is detected (and training requested). We shall expect that SVM, characterized by a costly training and a cheap recall phases should be preferred when concept drift are infrequent, otherwise k-NN might be better. This issue is application specific and should be decided application by application.

In order to scale the computation time performance of table IV (Matlab) to an embedded system, we executed a numerical intensive application composed of multiplications, sums and divisions both in Matlab and two embedded processors. The first embedded microprocessor is the 8 bits Arduino ATMEGA328P working at 16MHz, the second one the 16 bits STM32F103ZET6 one running at 72MHz. The ratio between the two execution times is taken as a rough estimate of the gain that the particular pc-based platform provides over the embedded system. It comes out that the Arduino processor is 78.2 times slower than corresponding execution in Matlab whereas the STM32 one is only 10.9. With these figures, the expected execution time per sample would be, in the SVM+H-CDT case of table IV, 34ms for the Arduino and 4.5ms for the STM32, respectively.

As a last note we comment about figure 4. We see how averaged performances are kept over time despite the changes. In fact, a reaction to the detection the active mechanism is enabled in addition to the adaptive mechanism intrinsic with the classifiers and accuracy are high. The fact that *k*-NN is slightly worse in the *average* after sample 10000 is associated with the fact that the change, even if detected, keeps in the KB of the k-NN samples related to the drift (only old samples are discarded and the drift operated for long time). Afterwards, new data cause the classifier to steadily improve its accuracy: new incoming data are inserted in the KB of the *k*-NN, made available and immediately used. Differently, the SVM either is retrained following a change detection or stores new samples not satisfying the KKT conditions in the incremental KB (IKB) and, only when they represent a significant percentage of the SVs present in KB a new training procedure is activated to redesign the classifier (the new SVs are identified). This is what happens with the SVM of figure 4.

As a final note, it is worth point out that false positives introduce an unnecessary retrain of the classifier which is positive to track drifts and changes hardly detectable for their magnitude at the expenses of a higher cost in computation.

A real-world dataset

An example of a real datastream is shown in Figure 5 where also its detection performance of the methods is provided. Here, the H-CDT detects all changes, the CI-CUSUM only the first one, characterized by a larger change in the mean.

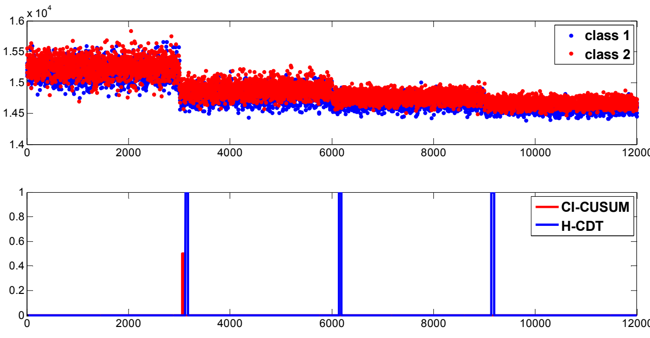


Fig. 5. Photodiodes sensing. The input radiation changes and introduces four abrupt changes in the datastream (upper plot); the instants of time a change is detected by the CI-CUSUM and the H-CDTs are also given (lower plot).

The tracking-to-change experiments were run with results given in table V. At first the advantages introduced by adaptation are evident when we compare accuracies provided by the JIT classifiers with those of static solutions. Again, SVM is advantageous over k-NN in terms of memory consumption and the use of H-CDT provides a gain in computation complexity over the CI-CUSUM.

Here, the best accuracy performances are associated with SVM classifiers even if k-NN performs well thanks to the effective detection ability provided by H-CDT. However, as mentioned in the previous experiment, the choice of the optimal classifier is application dependent.

TABLE V: COMPARED PERFORMANCE ON REAL DATASET

Method	Classification Accuracy	Computation Time (ms)	Memory Usage (couples)
SVM + CI-CUSUM	73.05%(0.069)	0.672(0.073)	887(226.9)
SVM+H-CDT	73.39 %(0.073)	0.243(0.032)	551(108.7)
k-NN + CI-CUSUM	70.40%(0.063)	0.577(0.028)	1688(221.1)
k-NN+H-CDT	73.37%(0.059)	0.426(0.048)	937(28.2)
Static k-NN	54.22%(0.014)	0.050(0.001)	400(0)
Static SVM	45.16%(0.072)	0.032(0.004)	269 (49.8)

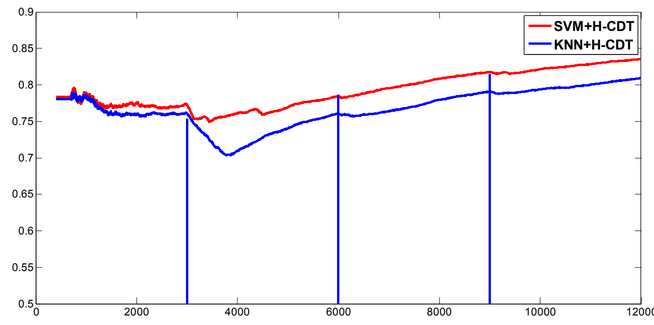


Fig. 6. The average accuracy of the classifiers over time; the reference CDT is the H-CDT.

The evolution over time of the accuracy performance (averaged over runs) is given in figure 6. The "peaks" are associated with the detection phase after which reaction takes place. The fact that SVM is very responsive implies that many samples violate here the KKT conditions inducing the classifier to be retrained in addition to retrains induced by detection of changes in stationarity. De facto, given the abrupt type of perturbations, every time we retrain the SVM we generate a classifier that works in locally stationary conditions. Interestingly, the performance of the classifier improves over time despite the abrupt changes. This implies a structural concept drift in the pdf of the process generating the data which evolves towards problems whose classification complexity reduces.

VI. CONCLUSION

Designing applications able to react just in time to changes affecting the stationarity of datastreams is a main feature for any intelligent embedded system dedicated to sensing applications. Hardware resources are here finite both in terms of computational power and memory usage hence pushing the designer towards resource-aware solutions. The paper, by referring to a classification application, compares two interesting candidates as CDTs, namely the CI-CUSUM and the ICI CDT families. At the same time it envisages active k-NN and SVM classifiers, modified to host a CDT. The outcome classifier is able to detect changes in the incoming samples and react accordingly by updating the classifier to track the change and maximize accuracy. It comes out that the H-CDT is lighter and best performing both in terms of false positives and negatives than the CI-CUSUM rival which, however, is more general and able to detect more subtle concept drift as those not affecting the mean and the variance of the pdf generating the data (or derived features). We surely recommend the use of the H-CDT (or the newly introduced sequential version of the LP-CPM [34]) as CDT in intelligent embedded systems. As such, the H-CDT code has been made available and can be downloaded from the link given in [38]. If we move to the Just-In-Time adaptive classifier issue then we cannot immediately claim whether a SVM or a k-NN core should be considered. While memory consumption is always in favor of a SVM-based solution, computational complexity depends on the frequency a change is detected in the datastream and, as such, is application dependent. Both algorithms should be tested for computational complexity as we did in the experimental section before being ported to the embedded system. The use of CDT introduces, a priori, false positives FP and false negatives FN. FP mainly introduce an unnecessary retrain of the classifier with an increase of the computational complexity and energy consumption. In terms of accuracy they might negatively affect the k-NN since old samples are removed (when they should be not). However, this introduces a mechanism which is beneficial if subtle drift or small magnitude changes affect the datastream since a periodic training is induced. FN, a priori unwished events, are here strongly kept under control by tolerating FP and mitigated by the particular nature of the adaptive classifiers. In fact, both classifiers introduce implicit adaptation mechanisms by taking advantage of new supervised samples either by inserting them in the knowledge base (k-NN) or in a suitable set (SVM) whenever they violate the KKT conditions.

REFERENCES

[1] H. Flavell, "Piaget's legacy," *Psychol. Sci.*, vol. 7, no. 4, pp. 200–203, Jul. 1996.

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

[3] G.Acciani, G. Brunetti, and G. Fornarelli, "Application of neural networks in optical inspection and classification of solder joints in surface mount technology," *IEEE Transactions on Industrial Informatics*, vol. 2, no.3, pp. 200–209, 2006.

[4] A. Rizzo and M. G. Xibilia, "An innovative intelligent system for sensor validation in tokamak machines," *IEEE Transactions on Control Systems Technology*, vol.10, no. 3, pp. 421–431, 2002.

[5] E. Athanasopoulou and C. N. Hadjicostis, "Probabilistic approaches to fault detection in networked discrete event systems," *IEEE Transactions on Neural Networks*, vol.16, no.5, pp.1042–1052, 2005

[6] I. W. Burr, *Statistical Quality Control Methods*. New York: Dekker, 1976.

[7] G. A. F. Seber, *Linear Regression Analysis*. New York: Wiley, 1976.

[8] R. Gnanadesikan, *Methods for Statistical Data Analysis of Multivariate Observations*. New York: Wiley, 1977.

[9] R. J. Herrnstein, D. H. Loveland, and C. Cable, "Natural concepts in pigeons," *J. Exp. Psychol.: Animal Behavior Process*, vol. 2, pp. 285–302, 1976.

[10] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, pp. 80–83, 1945.

[11] M. Kendall, *Rank Correlation Methods*, 4th ed. London, U.K.: Griffin, 1975.

[12] B. F. J. Manly and D. I. MacKenzie, "A cumulative sum type of method for environmental monitoring," *Environmetrics*, vol. 11, pp. 151–166, 2000.

[13] C. Alippi, G. Boracchi, and M. Roveri, "Adaptive classifiers with ICI-based adaptive knowledge base management," in *Artificial Neural*

Networks (ICANN 2010), Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 6353, pp. 458–467.

[14] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, 2009

[15] G. Anastasi, M. Conti, M. D. Francesco, and A. Passarella, “Energy conservation in wireless sensor networks: A survey,” *Ad Hoc Networks*, vol. 7, pp. 537–568, 2009

[16] L. Rokach, “Ensemble-based classifiers,” *ACM Journal Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, Feb. 2010.

[17] C. Alippi, G. Boracchi, and M. Roveri, “Just-in-time adaptive classifiers – part II: Designing the classifier,” *IEEE Transactions on Neural Networks*, vol. 19, no. 12, pp. 2053–2064, Dec. 2008.

[18] C. Alippi, G. Boracchi, and M. Roveri, “An effective just-in-time adaptive classifier for gradual concept drifts,” in *Proceedings of International Joint Conference on Neural Networks*, San Jose, California July 31–Aug. 5, 2011, pp. 1675–1682.

[19] C. Alippi, L. Bu, and D. B. Zhao, “SVM-based just-in-time adaptive classifiers,” in *ICONIP 2012, Part II, Lecture Notes in Computer Science.* Springer Berlin/ Heidelberg, 2012, vol. 7664, pp.664–672.

[20] C. Alippi and M. Roveri, “Just-in-time adaptive classifiers – part I: detecting nonstationary changes,” *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1145–1153, July 2008.

[21] A. Goldenshluger, A. Nemirovski, “On spatial adaptive estimation of nonparametric regression,” *Mathematical Methods of Statistics*, vol. 6, pp.135–170, 1997

[22] V. Katkovnik, “A new method for varying adaptive bandwidth selection,” *IEEE Transactions on Signal Processing*, vol. 47, no. 9, pp. 2567–2571, 1999.

[23] 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.

[24] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1972.

[25] N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, 2000

[26] H. Shen, H. Xi, and G. Xie, “The improved grid-search algorithm used in the fault diagnosis by SV,” *Mechanical Engineering & Automation*, pp.108–110, 2012

[27] G. W. Gates, “The reduced nearest neighbor rule,” *IEEE Transactions on Information Theory*, vol. 18, no. 3, pp. 431–433, May 1972.

[28] A.Bordes, S. Ertekin, J. Waston, and L. Bottou, “Fast kernel classifiers with online and active learning,” *Journal of Machine Learning Research*, vol. 6, no. 1, pp. 579–1619, 2005.

[29] R. Collobert and S. Bengio, “SVM torch: support vector machines for large-scale regression problems,” *Journal of Machine Learning Research*, vol. 1, pp. 143–160, 2001.

[30] S. Shalev-Shwartz and N. Srebro, “SVM optimization: inverse dependence on training set size,” in *Proceedings of the 25th International Conference on Machine Learning (ICML)*, Helsinki, Finalnd, July 2008, pp.928–935.

[31] R. Isermann, *Fault-Diagnosis Systems: An Introduction from Fault Detection To Fault Tolerance*. Springer Verlag, 2006.

[32] G. B. Huang, Q. Y. Zhu, and C. K. Siew, “Extreme learning machine: theory and applications,” *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[33] B. Schrauwen, D. Verstraeten, and J. V. Campenhout, “An overview of reservoir computing: theory, applications and implementations,” in *Proceedings of the 15th European Symposium on Artificial Neural Networks*. Bruges, Belgium, April 25-27, 2007, pp. 471–482.

[34] G. J. Ross, D. K. Tasoulis, and N. M. Adams, “Nonparametric monitoring of data streams for changes in location and scale”, *Technometrics*, vol. 53, no. 4, pp. 379–389, 2012

[35] P. Padhy, R. K. Dash, K. Martinez, and N. R. Jennings, "A utility-based sensing and communication model for a glacial sensor network", in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'06)*, Hakodate, Japan, May 8–12, 2006, pp. 1353–1360.

[36] C. Alippi, G. Anastasi, M. Di Francesco, and M. Roveri, “An adaptive sampling algorithm for effective energy management in wireless sensor networks with energy-hungry sensors,” *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 2, pp. 335–344, Feb. 2010.

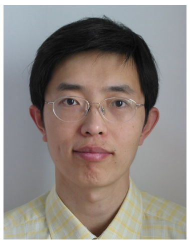
[37] C. Alippi, G. Boracchi, and M. Roveri, “Just in time classifiers for recurrent concepts,” *IEEE Transactions on Neural Networks and Learning Systems*, DOI:10.1109/TNNLS.2013.2239309, to be published.

[38] http://www.i-sense.org/open_library.html

Cesare Alippi (SM’94–F’06) received the degree in electronic engineering cum laude in 1990 and the PhD in 1995 from Politecnico di Milano, Italy.

Dr. Alippi 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). He holds 5 patents and has published about 200 papers in international journals and conference proceedings. His current research addresses adaptation and learning in non-stationary environments and Intelligent embedded systems. Prof. Alippi is Vice-President Education of the IEEE Computational Intelligence Society (CIS), Associate editor (AE) of the *IEEE Computational Intelligence Magazine*, past AE of the *IEEE Transactions on Neural Networks*, the *IEEE Transactions on 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 was awarded Knight of the Order of Merit of the Italian Republic.

Derong Liu (S'91-M'94-SM'96-F'05) received the Ph.D. degree in electrical engineering from the University of Notre Dame in 1994. Dr. Liu was a Staff Fellow with General Motors Research and Development Center, Warren, MI, from 1993 to 1995. He was an Assistant Professor in the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ, from 1995 to 1999. He joined the University of Illinois at Chicago in 1999, and became a Full Professor of electrical and computer engineering and of computer science in 2006. He was selected for the "100 Talents Program" by the Chinese Academy of Sciences in 2008. He has published 10 books. Dr. Liu has been an Associate Editor of several IEEE publications. Currently, he is the Editor-in-Chief of the IEEE Transactions on Neural Networks and Learning Systems, and an Associate Editor of the IEEE Transactions on Control Systems Technology. He was an elected AdCom member of the IEEE Computational Intelligence Society (2006-2008). He received the Faculty Early Career Development (CAREER) award from the National Science Foundation (1999), the University Scholar Award from University of Illinois (2006-2009), and the Overseas Outstanding Young Scholar Award from the National Natural Science Foundation of China (2008).



Dongbin Zhao (M’06-SM’10) received the B.S., M.S., Ph.D. degrees in material processing engineering from Harbin Institute of Technology, Harbin, China, in Aug. 1994, Aug. 1996, and Apr. 2000 respectively.

Dr. Zhao was a postdoctoral fellow with Tsinghua University, Beijing, China, from May 2000 to Jan. 2002. He became an associate professor, and a professor in 2002 and 2012 respectively at the Institute of Automation, Chinese Academy of Sciences, China. He has published one book and over thirty international journal papers. His current research interest covers computational intelligence, adaptive dynamic programming and applications.

Dr. Zhao has been an Associate Editor of the *IEEE Transactions on Neural Networks and Learning Systems*, and *Cognitive Computation*, and the *Newsletter Editor* of IEEE Computational Intelligence Society. He received five scientific awards, including the Third Scientific Award of Beijing (2010).

Li Bu received the B. S. degree in electronic engineering and automation in 2012 from the Chinese University of Mining and Technology. Miss Bu is now a graduate in the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences. She has published one conference paper and submitted two journal papers. His current research interest lies in the area of adaptation and learning in non-stationary environments and computational intelligence.