Intelligent Data Analysis 00 (2012) 1–23 1
DOI 10.3233/IDA-2012-0552
IOS Press

# Tracking recurrent concepts using context

João Bártolo Gomes[a,*], Pedro A.C. Sousa[b] and Ernestina Menasalvas[a,1]
[a]*Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, Madrid, Spain*
[b]*Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Caparica, Portugal*

**Abstract.** The problem of recurring concepts in data stream classification is a special case of concept drift where concepts may reappear. Although several existing methods are able to learn in the presence of concept drift, few consider contextual information when tracking recurring concepts. Nevertheless, in many real-world scenarios context information is available and can be exploited to improve existing approaches in the detection or even anticipation of recurring concepts.

In this work, we propose the extension of existing approaches to deal with the problem of recurring concepts by reusing previously learned decision models in situations where concepts reappear. The different underlying concepts are identified using an existing drift detection method, based on the error-rate of the learning process. A method to associate context information and learned decision models is proposed to improve the adaptation to recurring concepts. The method also addresses the challenge of retrieving the most appropriate concept for a particular context. Finally, to deal with situations of memory scarcity, an intelligent strategy to discard models is proposed. The experiments conducted so far, using synthetic and real datasets, show promising results and make it possible to analyze the trade-off between the accuracy gains and the learned models storage cost.

Keywords: Data stream mining, concept drift, recurring concepts, context-awareness, ubiquitous knowledge discovery

## 1. Introduction and motivation

In real-world applications where data is continuously being generated, it is common to observe significant changes in the underlying data distribution over time. This has a strong impact when applying data mining to data streams, as predictive models become invalid when the underlying concept changes [13,25]. The problem of learning from time-changing data streams is generally known in the literature as concept drift [9,13,24,25,29]. An effective data stream mining system must recognize and adapt to changes by continuously learning the different time-changing concepts [9,25]. Therefore, learning systems should be able to detect and adapt to concept changes without explicitly being informed about such changes. For example, using the available contextual features [11,28] or the performance of the base learner [9] as a technique to recognize changes in the underlying concept.

A particular case of concept drift is when previously seen concepts reappear [8,15,29,31]. Although concept recurrence is observed in many real-world problems [29], few approaches explore this possibility. When concepts are associated to context, this means that they may reappear when such context reoccurs. For example, a weather prediction model usually changes according to the seasons. The same happens with product recommendation or text filtering models where user interest and behaviour might

*Corresponding author: João Bártolo Gomes, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, s/n 28660 Boadilla del Monte, Madrid, Spain. E-mail: joao.bartolo.gomes@alumnos.upm.es.
[1]The research is partially Lnanced by project TIN2008-05924 of Spanish Ministry of Science and Innovation.

change over time due to fashion, economy, spatio-temporal situation or any other context [11,25,29]. This has motivated some works [11,28] to analyze how to use context to track concept changes.

The usual approach to deal with concept drift is to use some forgetting mechanism and train a new model [9,24]. In scenarios with recurrence, this implies to relearn a previously observed concept. However, a more sophisticated technique to exploit recurrence is to store learned models (that represent past concepts) [8,15,31], reusing these when appropriate. This avoids the effort and need to relearn a known concept when a similar one reappears.

Another interesting idea, in combination with storing past concepts, is to explore the history of concept changes over time. Such has been used in [8,11,32], where the concept history representation is built from the changes observed between concepts in the data stream. This means that if the history repeats itself when concepts recur, it is possible to anticipate the adaptation to change by using a stored model that represents the recurring concept. Therefore, associating context and concepts can improve existing methods, allowing to estimate which concept is likely to recur given the occurring context.

In this paper, we present a data stream learning algorithm to deal with recurring concepts. The approach combines on the one hand the performance of stored models representing previously learned concepts and on the other hand exploits learned relations between context and stored models. The different underlying concepts are identified using an existing drift detection method, based on the error-rate of the learning process [9]. The paper presents solutions to the main challenges that arise from the integration of context in the learning process. Such challenges are:

i) How to represent context information, the context history and its integration with learned concepts.
ii) Preserve information from learned concepts in a compact representation (i.e., models).
iii) Compare context and learned models to recognize recurrent concepts.
iv) Anticipate recurring concepts and adapt to concept drift.

Furthermore, the approach adapts itself to the underlying resource constraints in terms of memory consumption and reduces the number of processed records, in order to optimize the trade-off between accuracy and efficiency. In situations of memory scarcity, an intelligent strategy is executed to discard models that are considered less promising to reuse in a recurrence situation (according to several proposed criteria).

We show experiments in which the proposed approach is able to improve adaptation to concept drift, and thus the overall accuracy and efficiency of the learning process in the presence of recurring concepts. Moreover, the memory costs associated with the approach are analyzed and the proposed memory-aware strategy is tested, showing that despite the memory consumption cost the learning process accuracy increases, even while adapting to the memory constraints.

This paper is organized as follows: In Section 2 the related work is reviewed and compared with the approach presented in this paper, followed by Section 3 where the preliminaries and problem definition are presented. The main challenges of the proposed approach are discussed in Section 4, where the method to store and retrieve models, the drift adaptation strategy and the learning process are explained in detail. An intelligent strategy to deal with situations of memory scarcity is proposed in Section 4.6. In Section 5 the experimental setup and the results are presented and discussed. Finally, we provide some concluding remarks and outline future research work in Section 6.

## 2. Related work

The proposed approach deals with learning recurring concepts from a data stream using context. Consequently, we review firstly methods that address the problem of concept drift and recurring concepts,

then we focus our review on context and context-aware approaches.

### 2.1. Concept drift and recurring concepts

A general review of the literature related to the problem of concept drift can be found in [25]. The different existing approaches can be structured by the technique used to address the problem:

1. Using a window of records or a decay function, where old records are forgotten and the decision model is updated with the most recent data as it appears in the window [16,17,21,29].
2. Building an ensemble using classification models learned from fixed-size sequential chunks of the data stream training records [18,24,27] and adapt the weights of the classifiers according to the underlying concept.
3. Using a drift detection method [1,9,28,32], which is able to signal when drift occurs.

We will now review each of the above approaches in the following subsections.

#### 2.1.1. Time window

This is a simple and efficient approach, however, its main drawback lies in determining the size of the window. Schlimmer and Granger proposed the STAGGER system [21] that was amongst the first to explicitly address the problem of concept drift. STAGGER keeps a set of concept descriptions based on a set of weighted elements, where each element is a Boolean function of attribute-value pairs that is represented by a disjunction of conjunctions. To adapt to concept drift it incrementally updates the concept description weights and functions over a fixed time window. The FLORA learning system proposed by Widmer and Kubat [29] adjusts its window size dynamically using a heuristic based on the prediction accuracy and concept descriptions. It also handles recurrence by storing concept descriptions. Klinkenberg and Joachims [17] monitor the value of several performance indicators, accuracy, recall and precision over time. The key idea is to automatically determine and adjust the window size so that the estimated generalization error on new examples is minimized. Klinkenberg [16] proposed an automatically adaptive approach to the time window, instance selection and weighting of training records, which also aims to minimize the estimated generalization error.

#### 2.1.2. Ensemble of classifiers

In this approach, the main challenges are on how to determine which classifiers to use, their weights and the chunks size.

Street and Kim [24] build separate classifiers on sequential chunks of training records and combine these into a fixed-size ensemble. Wang et al. [27] propose a similar approach but the weights are calculated based on the classifiers accuracy on recent data. The DWM algorithm, proposed by Kolter and Maloof [18], dynamically builds and deletes weighted classifiers in response to changes in performance. Scholz and Klinkenberg [23] propose a boosting-like method for data streams that adapts to concept drift. For each iteration, the classifiers are induced and re-weighted according to the most recent records.

However, the previous ensemble approaches do not explicitly address the problem of recurring concepts and they may have to relearn concepts as if they have never seen them before. Ramamurthy and Bhatnagar [20] presented an ensemble approach that exploits recurring concepts, using a global set of classifiers learned from sequential data chunks. If no classifier in the ensemble performs better than the error threshold, a new classifier to represent the current concept is learned and stored. The classifiers with better performance on the most recent data are part of the ensemble for labeling new records. Similarly, in [15] an ensemble is used but incremental clustering was exploited to maintain information

about historical concepts. The proposed framework captures batches of examples from the stream into conceptual vectors. Conceptual vectors are clustered incrementally by their distance and for each cluster a new classifier is learned. Classifiers in the ensemble are learned using the clusters.

#### *2.1.3. Drift detection*

In this type of approach it is assumed that periods of stable concepts are followed by change into another stable concept period. Gama et al. [9] and Yang et al. [32] approaches monitor the error-rate of the learning algorithm to find drift events. In [9], when the learning process error-rate increases above certain pre-defined levels, the method signals that the underlying concept has changed. Alternatively Baena et al. [1], uses the distribution of the distances between classification errors to signal drift. If the distance, which results from more consecutive errors, is above a pre-defined threshold, the underlying concept must be changing and a change event is triggered. The basic adaptation strategy after drift is detected is to discard the old model and learn a new one to represent the new underlying concept [1,9].

Most of these approaches discard old information and the possibility to exploit concept recurrence is not even considered. However, more sophisticated approaches that exploit this possibility, such as [8, 31], store learned models and reuse them when a similar concept reappears in the stream, thus avoiding the effort to relearn a previously observed concept. The method proposed by Yang et al. [31] uses a proactive approach to recurring concepts, which means to reuse a concept from the history of concepts. This history of concepts is represented as a Markov chain and allows selecting the most likely concept according to a given transition matrix. The approach proposed by Gama and Kosina [8] uses the drift detection method presented in [9] to identify stable concepts and memorizes learned classifiers that represent these concepts. After change is detected in situations of recurrence, referees are used to choose the most appropriate classifier to reuse (i.e., the referee prediction about the applicability of the classifier is greater than a given threshold).

From the reviewed approaches the ones more similar to our proposal are [8,31], as both use drift detection and store past models as a means to adapt to concept drift and recurring concepts. However, none explores the usage of context information, and use different techniques to select stored models in situations of recurrence, while the approach we propose in this paper uses learned relations between concepts and context to improve the adaptation to drift in such situations.

### *2.2. Context and context-aware approaches*

Context representation in information systems is a problem studied by many researchers as they attempt to formally define the notion of context. Schmidt et al. [22] defined context as the knowledge about the users and device state. Moreover, Dey [3] defines context as 'Context is any information that can be used to characterize the situation of an entity'. In contrast, Brezillon and Pomerol [2] argue that there is no particular knowledge that can be objectively called context, as context is in the eye of the beholder. They state that 'knowledge that can be qualified as 'contextual' depends on the context!'. In addition, Padovitz et al. [19] proposed a general approach that models context using geometrical spaces called Context Spaces, which allows the inference of situations in context-aware environments. The context spaces representation is used as the basis for the context representation in the approach proposed in this paper.

Context dependence has been recognized as a problem in several real world domains [11,26,28]. Turney [26] was among the first ones to introduce the problem of context in machine learning, where he presented a formal definition in which the notions of primary, contextual and context-sensitive features were introduced. Such notions are based on a probability distribution for the observed classes given the

features. However, when the probability distribution is unknown it is often possible to use background knowledge to distinguish between features. In the proposed learning system the same approach is followed, as the system processes what experts define as contextual features in a meta-learning level and primary features in the base learning level.

Widmer [28] exploits what is referred as contextual clues (based on the Turney [26] definition of primary/contextual features) and proposes a meta-learning method to identify such clues. Contextual clues are context-defining attributes or combinations of attributes whose values are characteristic of the underlying concept. When more or less systematic changes in their values are observed this might indicate a change in the target concept. The method automatically detects contextual clues on-line, and when a potential context change is signalled, the knowledge about the recognized context clues is used to adapt the learning process in some appropriate way. However, if the *hidden context* is not represented in the contextual clues it is not possible to detect and adapt to change.

The approach of conceptual clustering proposed by Harries [11], identifies stable *hidden contexts* from a training set by clustering the instances assuming that similarity of context is reflected by the degree to which instances are well classified by the same concept. A set of models is constructed based on the identified clusters. This idea proved to work very well with recurring concepts and real world problems. However, its main drawback is the off-line training required to obtain the conceptual clusters, as these could lead to inaccuracy with concepts or patterns that were not seen during training.

In the approach proposed in this paper context integration shares the motivation with the approach presented in [11] where the method infers periods when the context is stable (from available context features), that are described as contextual clusters. However, we propose an on-line method that learns context-concept relations from the concept history. Moreover, the proposed method does not require the partition of the dataset into small batches as the concept representations are learned from an arbitrary number of records, as determined by the drift detection method.

## 3. Preliminaries

Online supervised learning [4,5], aims to learn a classification model from a stream of training records (possibly infinite) and apply this model to predict the class of unlabeled records with high accuracy. Data stream mining algorithms cannot keep a large number of records in memory and should process each record only once to deal with the massive volumes of data that are typically processed in data stream learning systems [4,5]. Consequently, the classification model is learned incrementally and after the first training records it is possible to use it to predict the class of incoming unlabeled records [4]. An anytime classification scenario [4,33] is assumed and the classification model accuracy is expected to increase as the number of training records grows. However, this will not happen if the underlying data distribution changes, as the classification performance will decrease. An explicit adaptation strategy to drift must be performed, for instance by simply forgetting old patterns in favour of new ones [25]. Therefore, it is important that data stream mining algorithms not only learn incrementally but also detect and adapt to changes in the underlying data distribution.

A particular type of concept change is that of reappearing concepts [8,15,29,31]. In such cases recognizing an already learned concept might improve the adaptation by avoiding to relearn it.

### *3.1. Definitions*

Let $X$ be the space of attributes and its possible values and $Y$ the set of possible (discrete) class labels. Let the data stream training records $X_i = (\vec{x_i}, y_i)$ with $x_i \in X$ and $y_i \in Y$, that arrive sequentially,

where $\vec{x_i}$ is a vector of attribute values and $y_i$ is the class label for the $i^{th}$ record in the stream. These records are processed by a base learner to incrementally train a model $m$ that returns the class label of a record $\vec{x} \in X$, such that $m(\vec{x}) = y \in Y$. A stable concept can be learned when the records of a given period (or set) $k$ (with an arbitrary number of records) are independently identically distributed according to a distribution $\Pr_k(x, y)$. In situations of concept change, $\Pr_k(x, y) \neq \Pr_{k+1}(x, y)$. The case of a recurring concept is that when the records from a period $k$ are generated from the same distribution as a previously observed period $\Pr_k(x, y) = \Pr_{k-j}(x, y)$. The goal is that the trained model $m$ minimizes the number of prediction errors. Furthermore, a model $m_k$ learned from a certain period $k$ can be saved and then reused in situations where the underlying concept represented by $m_k$ reappears in the stream. This would improve the on-line learning process because it does not require to learn from scratch a previously known concept when the underlying concept changes. Furthermore, it reduces the number of training records that need to be processed compared with approaches that do not take recurrence into account (i.e., forget old models).

### *3.2. Context integration*

In situations where contextual information is related to the underlying concepts, such knowledge could be exploited to detect and adapt to recurring concepts. Nevertheless, these relations are not known apriori, and it is even possible that given the available context information it is not possible to find such relations. Still, in many real-world problems we find examples where the available context information does not explain all global concept changes, but partially explains some of these. For example, user preferences often change with time or location. Imagine a user that has different interests during the weekends, weekdays, when at home or at work. In general, different concepts can recur due to periodic context (e.g., seasons, locations, week days) or non-periodic context (e.g., rare events, fashion, economic trends).

The available context information is represented as a vector of contextual attributes. Finding associations between context and concepts is not trivial, as their relations are not known in advance. Furthermore, when exploiting such relations to improve adaptation to recurring concepts, one could argue that context information should be simply added as additional attributes in the base learner. However, that would increase the problem dimensional complexity and introduce noise to the learning process, since concepts may change due to factors that may not be expressed as context attributes. Therefore, we believe that such context information should be integrated carefully in a meta-learning level (as discussed in [11,26,28]).

## 4. Context-aware learning system

In this section an overview of the proposed learning system is introduced, which is followed by a complete description of its components in the corresponding sections.

Data stream learning algorithms have to deal with detection and adaptation to change. In situations of concept recurrence, anticipating to the reappearing concept can improve the learning process efficiency. Consequently, we propose to continuously store learned models from the data stream learning process and associate context information with these models. This context information will be used later when a similar concept reappears, in the selection of an appropriate model, which is able to represent the reappearing concept.

The proposed learning system is based on a two-level framework:

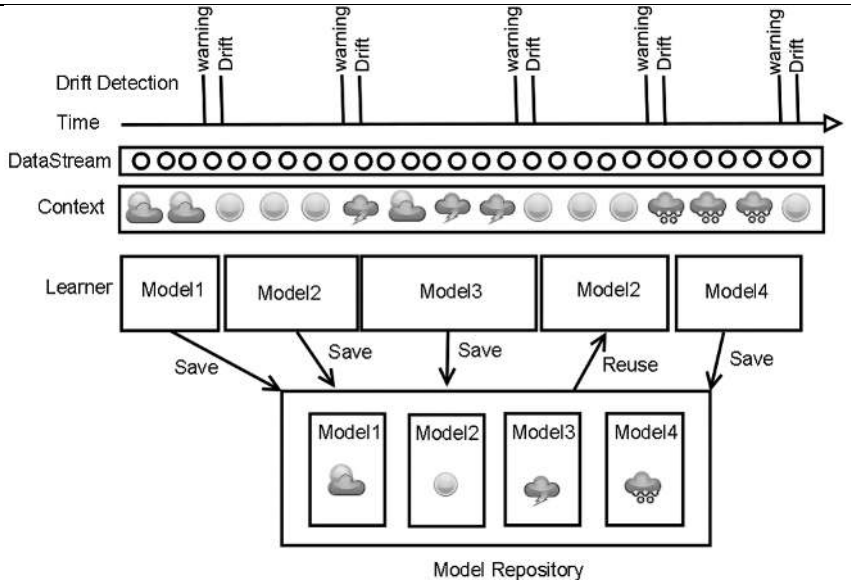



Fig. 1. Context-aware data stream learning system. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-2012-0552)

a) base learner level where an incremental algorithm learns the underlying concept, building a classification model.
b) meta-learning level where: i) detection and adaptation to concept drift is performed; ii) the context-concept relations are learned and used to deal with the recurring concepts.

One of the main assumptions behind the proposed approach is that concepts reappear. We take this as the basis of our approach to improve the learning process. The main advantage is avoiding to relearn previously observed concepts, which translates into faster adaptation to concept changes, accuracy gains and efficiency of the learning process (i.e., less processing time, due to less training records processed). The drawback associated with it, is the additional memory consumption, as models have to be stored so they can be reused when the concept they represent reappears. Therefore, there is an accuracy-efficiency trade-off in storing more decision models. Consequently, as part of the approach we propose a memory-aware strategy to discard models to deal with situations of memory scarcity (see Section 4.6). Moreover, there is the possibility of conflicts with recurring concepts in the presence of different context. The approach is sensitive to these situations because the context-concept relation history is constantly being updated, and context is not the only factor when selecting a model to reuse.

Figure 1 illustrates the learning process and its components. This continuous learning process consists of the following steps:

1. Process the incoming records from the data stream using an incremental learning algorithm (base learner) to obtain a decision model $m$ capable of representing the underlying concept, and classify unlabeled records.
2. Context records are associated with the current model $m$. The history of context-concepts relations will be referred to as context-concepts relations history.
3. A drift detection method that monitors the error-rate of the learning algorithm. When error-rate goes above pre-defined levels the drift detection method signals a *warning* (possible drift) or *drift*.

4. When change is detected two situations are possible: i) the underlying concept is new (i.e., no equivalent concept is represented in the classifiers repository) and the base learner will learn the new underlying concept by processing the current incoming labeled records. The incremental classifier that is being learned will also classify the incoming unlabeled records as anytime classification is assumed; ii) the underlying concept is recurrent (i.e., has been learned previously). In this situation a classifier from the repository that represents the underlying concept is used to classify incoming unlabeled records. Further description of this process and the utility function used to select the retrieved model is described in Section 4.3.4.

The major issues to deal with when integrating contextual information to improve the adaptation to recurring concepts in the on-line learning process are:

- Representing the data stream underlying concepts in a compact form that is adequate for storage. Compare the similarity between learned concept representations.
- Representing and comparing dynamic context information.
- Detecting and adapting to recurring concept changes.
- Deciding which information to store about past concepts and related context.
- Deciding how to retrieve past concepts in situations of recurrence.
- Dealing with the accuracy-efficiency trade-off of the learning system under situations with memory constraints.

Context representation and similarity are presented in Section 4.1. The base learner algorithm is described in Section 4.2. Concept representation, context-concepts relations history and the measure of similarity between models, used to check if different models represent the same concept, are discussed in Section 4.3. Also in the same section, the model storage and retrieval procedures are presented. To detect when drift occurs the Drift detection method, proposed by Gama et al. [9] is used. This method is briefly summarized in Section 4.4, where the adaptation strategy to drift is proposed and discussed. Finally, in Section 4.5 the pseudo-code of the learning process is presented. In addition, to address the problem of memory scarcity, Section 4.6 discusses model utility criteria and presents an intelligent strategy to forget models based on such criteria.

### *4.1. Context representation*

The context representation and similarity we propose to integrate in our approach is inspired on the Context Spaces model [19], where a context state is represented as an object in a multidimensional Euclidean space. A context state $c_i$ is defined as a tuple of N context attribute-values,

$$c_i = (a_1^i, \ldots, a_n^i)$$

where $a_n^i$ represents the value of context attribute $a_n$ for the $i^{th}$ context state $c_i$.

The available context information depends on the learning environment and data mining problem. Context information can represent simple sensors (e.g., temperature, humidity) or a more complex context (e.g., season, location, gait) defined by domain experts or inferred by other means beyond the scope of the problem discussed in this work.

#### *4.1.1. Context similarity*

Context similarity is not a trivial problem [19], because while it could be more immediate to measure the (dis)similarity between two values in a continuous attribute, the same is not that easy when we

consider categorical ones and to a greater extent when integrating the heterogeneous attributes similarity into a (dis)similarity measure between context records/states. For the purposes of this work the degree of similarity between context states $c_i$ and $c_j$, using the Euclidean distance is defined as:

$$|c_i - c_j| = \sqrt{\sum_{K=1}^{N} dist(a_k^i - a_k^j)}$$

where $a_k^i$ represents the $k^{th}$ attribute-value in context state $c_i$. For numerical attributes distance is defined as:

$$dist(a_k^i, a_k^j) = \frac{(a_k^i - a_k^j)^2}{s^2}$$

where $s$ is the estimated standard deviation for $a_k$. For nominal attributes distance is defined as:

$$dist(a_k^i, a_k^j) = \begin{cases} 0 \text{ if } & a_k^i = a_k^j \\ 1 \text{ otherwise} & \end{cases}$$

We considered two context states $c_i$, $c_j$ to be similar if the distance between them is below a predefined threshold $\epsilon$:

$$similar(c_i, c_j) = \begin{cases} \text{true} & \text{if } |c_i - c_j| \leqslant \epsilon \\ \text{false} & \text{if } |c_i - c_j| > \epsilon \end{cases}$$

The definition of $\epsilon$ depends on the context space being represented and must be specified according to the problem domain knowledge.

### *4.2. Base learner*

The base learner is used to learn a model that represents the data stream underlying concept. Any classification algorithm able to learn incrementally can be used for this task, and such decision can be made according to the nature of data, choosing the algorithm that best suits it (e.g., high accuracy, handles noise, memory consumed, fast execution). For this task we propose to use the NaiveBayes algorithm, because it represents the concepts in a compact form (i.e., results in memory efficiency), is incremental, handles nominal as well as continuous attributes and has shown good results as a base learner [8,15,28].

In this section the NaiveBayes algorithm is briefly described. For further information the reader should refer to [30].

The Naive Bayes classifier provides a simple, incremental and efficient approach to learn probabilistic knowledge. From the Bayes theorem, the probability that an record $\vec{x} \in X$ belongs in class $y \in Y$

$$\Pr(Y = y | X = \vec{x}) = \frac{\Pr(Y = y) \Pr(X = \vec{x} | Y = y)}{\Pr(X = \vec{x})}$$

It is straight forward to estimate $\Pr(Y = y)$ by counting the number of records that belong to class $y$, which will be referred as $P_y$, and the number $N$ of records processed. Still, estimating $\Pr(X = \vec{x} | Y = y)$

in not feasible. Using the NaiveBayes assumption (i.e., that all attributes in $X$ are independent given the class in $Y$) it is possible to obtain,

$$\Pr(X = \vec{x}|Y = y) = \prod_{x_i \in \vec{x}}^{|X|} \Pr(X = x_i|Y = y)$$

that can be estimated by counting the number of records belonging to class $y$ with $x_i$, which will be referred as $P_{x_i,y}$. Finally, it is not required to estimate $\Pr(X = \vec{x})$ because it is a constant. The method presented by John and Langley [14], is used to handle numeric type attributes. The method assumes that the value of each attribute is normally distributed within each class $y$. Then it is possible to use,

$$\Pr(X = x_i|Y = y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right)$$

for each numeric attribute $x_i$ in $X$. It is possible to estimate $\mu$ and $\sigma^2$ by storing the sum and squared sum of the values of $x_i$.

The NaiveBayes classifier $m$ outputs the class label $y$ for a record $\vec{x}$,

$$m(\vec{x}) = y = \arg\max_y \left(\Pr(Y = y) \prod_{x_i \in \vec{x}}^{|X|} \Pr(X = x_i|Y = y)\right)$$

### *4.3. Concept history*

For the purpose of our approach it is important to have a memory efficient representation of concepts. NaiveBayes as a base learner achieves this, because given a model $m$ it only requires to store the estimation of the class $P_y$ and the estimation of each attribute given the class $P_{x_i,y}$, we will refer to these as conceptual vectors,

$$cv = \{P_y, P_{x_i,y}\}$$

#### *4.3.1. Model storage*

Learned models are kept stored so they can be reused in situations of recurrence. In such situations the repository is searched for an adequate model, which means, finding a model that represents the current underlying concept. If a match is found, it is expected a reduction in the computational cost that comes associated with learning a new model and also improved adaptation to concept drift (i.e., better learning curve in terms of accuracy). For each classification model $m$ in the model repository we store:

- The concept representation: as the Naive Bayes algorithm is used, this means to store the conceptual vectors of model $m$, that is represented as $cv$.
- $Acc(m)$ is an estimate of the accuracy of $m$ obtained during the period $m$ was used. Let $numCRecords_m$ be the number of correctly classified records by $m$ and $numRecords_m$ be the total number of records classified by $m$. The accuracy $Acc(m)$ is defined as:

$$Acc(m) = \frac{numCRecords_m}{numRecords_m}$$

- The timestamp $t$ that records the period when a model $m$ was used.

Consequently each decision model $m$ stored in the model repository is defined as the tuple:

$$m = \{cv(P_y, P_{x_i,y}), numCRecords(m), numRecords(m), t\}$$

Most of the information that is kept, together with the context-concepts history is used in the selection of past models to represent the latest underlying concept and also in the selection of models to delete in situations of memory scarcity. The memory consumption of the proposed approach is a function of the size used by the conceptual vectors $cv$, and as described previously, this depends on the number of attributes and classes that are considered, as more estimators need to be kept.

*4.3.2. Context-concepts relation history*

The main assumption under our approach is that when a concept reappears normally the context previously associated with it also reappears. We take advantage of this fact to anticipate the adaptation to recurring concepts. Here a description of how to create and represent the context-concepts relations history is presented.

Let $m_j$ be the model learned or used in a certain period $j$ (i.e., that represents the underlying concept during that period $j$) and $C_j = \{c_1, c_2, \ldots, c_n\}$ a sequence of $n$ context records observed during this period $j$. The context-concepts history representation uses the Naive Bayes algorithm to associate context with concepts. It is incrementally learned from the sequence of context records $C_j$, where the model $m_j$ identifier is used as the class label. This allow us to estimate the probability that a certain model $m_k$ represents the current underlying concept given a certain context state $c_i$, we denote this estimation of probability as $h(m_k|c_i)$, similarly as has been previously explained for the base learner prediction of the class label given $\vec{x}$. As a consequence, we can keep an approximate and compact representation of the context-concepts relation history, without keeping the context records, which would be impossible due to the memory required. The maximum number of models (i.e., the number of classes in the context-concepts history) that we can store is determined by the $m_{limit}$.

It is possible to aggregate the different context records of $C_j$ into one context record that we call frequent context $freqC$, where each attribute value of $freqC$ is the most frequent value that attribute takes in these records. This can be used to have more control over the creation of the context-concepts history. For example balance possible bias that can occur when the period length (in terms of records) of different concepts is not similar.

The Context-Concepts relation history also allows knowing the most frequent context for a given model. This is used when we need to compare the distance between the most frequent contexts that are associated with certain stored models. The proposed strategy to deal with memory scarcity uses this context distance as a criterion.

*4.3.3. Concept similarity*

To determine whether a certain model represents a new concept or a reappearing one a similarity measure is required. The *Conceptual equivalence* measure, that is based on the one proposed by Yang et al. [32] is used for this purpose. The measure is independent of the concept representation. Given two classification models $m_1, m_2$ and a sample dataset $D_n$ of $n$ records, it calculates for each instance $X_i = (\vec{x_i}, y_i)$ a score,

$$score(X_i) = \begin{cases} +1 \text{ if } m_1(\vec{x}) = m_2(\vec{x_i}) \\ -1 \text{ if } m_1(\vec{x}) \neq m_2(\vec{x_i}) \end{cases}$$

that is used to represent the degree of equivalence between $m_1$ and $m_2$, that is an average continous value score with range $[-1, 1]$, defined as,

$$ce = \frac{\sum_{X_i \in D_n} score(X_i)}{N}$$

The larger the output value, the higher the degree of conceptual equivalence. For the records in $D_n$ it compares how $m_1$ and $m_2$ classify the records. The authors [32] argue that the accuracy and the conceptual equivalence degree are not necessarily positively correlated. The reasoning is that, despite $m_1$ and $m_2$ might classify $D_n$ with low accuracy, their equivalence degree can be very high if their classifications match, even when both misclassify. Moreover, accuracy does not represent conceptual equivalence as models can still achieve the same accuracy and misclassify different parts of the attribute space.

We consider that if the obtained $ce$ value is above a pre-defined threshold, the models are similar and thus represent the same underlying concept.

*4.3.4. Model retrieval*

The main objective of model retrieval procedure is to find from the stored models the one which better represents the current underlying concept.

The proposed model retrieval procedure, combines an accuracy approach similar to the proposed in [27] and the information learned from the context-concepts history described in Section 4.3.2. The model $m_i$ accuracy is estimated using the Mean Square Error. This measure estimates the error of the classifier for a window of records $W_n$. The Mean Square Error $MSI_i$ for model $m_i$, using the window $W_n$ of $n$ records in the form of $(\vec{x}, y)$, where $y$ is the true class label for that record, can be expressed as:

$$MSE_i = \frac{1}{|W_n|} \sum_{(\vec{x},y) \in W_n} (1 - m_i^y(\vec{x}))^2$$

where the error of $m_i$ on record $(\vec{x}, y)$ is $1 - m_i^y(\vec{x})$, and $m_i^y(\vec{x})$ is the probability given by $m_i$ that $\vec{x}$ is an instance of class $y$.

Let $w_m$ and $w_c$ be the weights assigned to the $MSE_i$ calculated using dataset $W_n$ and to the context-concepts history $h(m_i|c_o)$, representing the probability estimation that the current underlying concept is represented by $m_i$ given the occurring context $c_o$ (i.e., the most frequent context observed for the records in $W_n$):

$$u(MSE_i, h(m_i|c_o)) = w_m * \frac{1}{1 + MSE_i} + w_c * h(m_i|c_o)$$

The utility function is calculated for all the models in the repository or if processing time is limited just the models associated to a certain context $c_o$. The model that has the highest utility value is selected. If its utility value is bellow a given threshold, it is not reused and we proceed as in a situation where the underlying concept is new.

Note that here instead of the accuracy we could have used the degree of conceptual equivalence (discussed in Section 4.3.3). However, the conceptual equivalence degree and the model accuracy are not necessarily correlated, and the accuracy based approach is less restrictive as it is more flexible to changes.

### 4.4. Drift detection and adaptation

The proposed learning system requires to identify when drift occurs, and for this purpose it uses the method proposed by Gama et al. [9], that is based on the error-rate of the learning process. We present a summary of the drift detection method but for further details we refer the reader to [9].

The mentioned drift detection method (DDM) [9] assumes that periods of stable (i.e., the data distribution is stationary) concepts are observed followed by changes leading to a new period of stability with a different underlying concept. It considers the error-rate (i.e., false predictions) of the learning algorithm to be a random variable from a sequence of Bernoulli trials. The binomial distribution gives the general form of the probability of observing an error. For each record $i$ in the sequence being sampled and $error_i$ the number of misclassifications at $i$, the error-rate is the probability of misclassifying $p_i = (error_i/i)$, with standard deviation given by $s_i = \sqrt{p_i(1-p_i)/i}$. It is assumed that $p_i$ will decrease while $i$ increases if the distribution of the examples is stationary. A significant increase in $p_i$, indicates that the class distribution is changing. The values of $p_i$ and $s_i$ are calculated incrementally and their minimum values $(p_{\min}, s_{\min})$ are recorded when $p_i + s_i$ reaches its minimum value. A warning level and a drift level, which represent confidence levels, are defined using $p_i, s_i, p_{\min}, s_{\min}$. The levels and the adaptation strategies for each one are defined as follows:

- $p_i + s_i \geqslant p_{\min} + 2 * s_{\min}$ for the warning level (95% confidence). Beyond this level, the incoming records are stored in anticipation for a possible change in concept.
- $p_i + s_i \geqslant p_{\min} + 3 * s_{\min}$ for the drift level (99% confidence). Beyond this level the concept drift is considered to be true, the adaptation strategy consists in resetting the model induced by the learning method and use the records stored during the warning period to learn a new model that reflects the current target concept. The values for $p_{\min}$ and $s_{\min}$ are also reset.

It should be noted that other methods for change detection can be used instead, without need to change the proposed learning process.

#### 4.4.1. Adaptation strategy to concept drift using context

One of the main contributions of the proposed approach is how contextual information is exploited when a concept change is detected. Consequently, the drift detection method adaptation strategy is extended to integrate the context information. The method puts the learning process in one of the levels, $stable$, $warning$, or $drift$. The main contribution is made in the *warning* and *drift* levels. The adaptation strategy to any of these levels is executed in the meta-learning level of the proposed approach through the following actions:

- $stable$, means that the error-rate is less than the pre-defined $warning$ or $drift$ levels. In this situation no adaptation is needed independently of the changes in context, because the performance is stable or increasing.
- $warning$, could represent a potential false alarm (i.e., the change level is not reached and the error-rate decreases to normal level). In this situation we:
  - prepare a new instantiation of the base learner $m_{new}$ to represent the new underlying concept in case the error-rate continues to increase and drift is detected in a near future.
  - if the statistics collected from context-concepts history are sufficient (i.e., after some pre-defined period considered for initial training) and $h(m_k|c_o))$ for a certain kept $m_k$ is above a given threshold, we anticipate to the recurring concept using $m_k$ to classify unlabeled records. Two situations can happen:

  - i) the model $m_k$ that we use to anticipate represents the current underlying concept and the error-rate decreases (i.e., the underlying concept is recurrent)
  - ii) the warning level continues, in this situation the process waits for $n$ records to execute the same adaptation strategy as in drift level.
- $drift$, in this case the current decision model is replaced with $m_{new}$ from the new base learner and waits until it processes $n$ records. If $m_{new}$ represents a new concept (i.e., no similar model is stored in the model repository), it continues to be incrementally updated and is added to the repository. Otherwise the model retrieval procedure is used to obtain from the repository the model that best represents the recurring concept, which is used to classify unlabeled records.

### *4.5. Learning process*

The on-line learning process of the proposed learning system is detailed in Algorithm 1. The process proceeds as follows:

- It continuously processes the records $X_i = \{\vec{x}, y\}$ as they appear in the Data Stream.
- In line 3, *currentClassifier* represents the classifier that is currently being used to classify unlabeled records. Its performance (i.e., right or wrong) on the prediction of $X_i$ is passed to the drift detection algorithm that identifies the current state of the learning process.
- If the process is in the normal level, the record that represents the occurring context is associated with the current model in the context-concepts relation history, and if the *currentClassifier* is new (i.e., not recurrent) it gets updated with the new training record.
- In the case of warning level (line 11), if the repository does not have the *currentClassifier*, it is stored. In addition (line 15), if the context-concepts relation history suggests a certain model with high probability, this model is reused and becomes the *currentClassifier*. This is a way to anticipate the adaptation that takes place in the drift level, but without requiring the collection of training records, it simply tries to predict what model would best represent the underlying concept given the current context. Still in this level (in lines 18 and 19), a *newLearner* is updated with the training record and it is added to a $warningWindow$. This window contains the latest records (that should belong to the most recent concept), and will be used to calculate the conceptual equivalence and estimate the accuracy of stored models with the current concept.
- When drift is signaled (line 20), the $currentClassifier$ is replaced by one from the model repository according to the model retrieval procedure described in Section 4.3.4, if none represents the current concept the $newLearner$ is used. During a pre-defined stability period (line 25), the $newLearner$ is updated and when it finishes (line 28) it is compared with repository models in terms of conceptual equivalence. If the current underlying concept is recurrent a stored model is reused to replace the $currentClassifier$, otherwise the $newLearner$ is used.
- A false alarm (line 22) is when a warning is signaled and then returns back to normal without reaching drift, in this case the $warningWindow$ and the $newLearner$ are cleared.

### *4.6. Resource-awareness*

The proposed approach may lead to higher accuracy and a reduction in the number of processed records when comparing it with approaches that relearn a recurrent concept from scratch. However, there is a memory cost associated with model storage that must be taken into account. Consequently, the proposed learning system is aware of the space consumed by the models in the repository and ensures

**Algorithm 1** Data Stream Learning Process

**Require:** Data stream $DS$, ModelRepository $MR$

**repeat**
  Get next record $DS_i$ from $DS$;
  prediction = $currentClassifier.classify(DS_i)$;
  $DriftDetection.update(prediction)$;
  **switch** DriftDetection.level
  **case** Normal
    $history.train(c_o, currentClassifier.ID)$;
  **if** $\neg recurrent$ **then**
    $currentClassifier.train(DS_i)$;
  **end if**
  **case** Warning
  **if** $\neg MR.contains(currentClassifier)$ **then**
    $MR.store(currentClassifier)$;
  **end if**
  **if** $history(c_o) > \rho$ **then**
    $currentClassifier = MR.getModelID(history(c_o))$
  **end if**
    $WarningWindow.add(DS_i)$;
    $newLearner.train(DS_i)$;
  **case** Drift
    $currentClassifier = MR.getModel(c_o)$;
  **case** FalseAlarm
    $WarningWindow.clear()$;
    $newLearner.delete()$;
  **case** Stability Period
    $WarningWindow.add(DS_i)$;
    $newLearner.train(DS_i)$;
  **if** $WarningWindow.size > \tau$ **then**
    **if** $\neg MR.containsEquivalent(newLearner)$ **then**
      $currentClassifier = newLearner$;
    **else**
      $currentClassifier = MR.getModel(c_o)$;
    **end if**
  **end if**
  **end switch**
**until** END OF STREAM

that this value is kept within a the predefined memory limit $m_{limit}$. Using the proposed concept representation (see Section 4.2), the maximum number of models that can be kept is a function of the number of attributes in the dataset, the number of classes and of the available memory. In situations where its not possible to add a new model to the repository without exceeding the memory limits, an intelligent strategy is used to discard the classifier that is considered to have the lowest utility. This strategy is based on the following criteria:

1. Equivalent classifiers (i.e., have the highest $ce$ value)
2. Classifiers that are associated with similar contexts (i.e., associated context has lowest distance)
3. Accuracy, $Acc(m)$ that is stored with the model.
4. Timestamp, in the rare case a tie results from using the other criteria.

The criteria we propose tries to promote heterogeneity in the repository, and thus the overall accuracy-efficiency of the learning system.

In situations of memory scarcity (i.e., when the system tries to store a model and $m_{limit}$ is reached), the system executes a function that deletes the model with the lowest utility (using the proposed criteria)

from the model repository. This function searches the model repository for the lowest accuracy models that share high concept equivalence, with the lower context distance, in order to maximize the concept and context heterogeneity in the repository. From this subset the one with lowest accuracy is deleted. If more than one model is the candidate after this step, the model with lowest timestamp is the one to be deleted.

There is a trade-off between the number of models that can be stored simultaneously and the possibility that this model can be reused in the future. In a worst case scenario where only one model fits the memory available for the model repository, the learning process will behave like the standard drift detection method, that exploits only a single classifier representing the latest concept, and thus it will not take advantage of concept recurrence. In the experimental results different memory configurations are tested, which allow us to analyze how the learning process efficiency changes with the possibility to keep a reduced number of models.

## 5. Experiments

This section, presents the experiments to test the proposed approach in terms of accuracy and efficiency to situations with strong memory limitations. The implementation of the proposed learning system was developed in Java, using the MOA [12] environment as a test-bed. MOA [12] stands for Massive Online Analysis and is an open-source framework for data stream mining written in Java. Related to the WEKA project [30], it includes a collection of machine learning algorithms and evaluation tools particular to data stream learning problems. The MOA evaluation features (i.e., prequential-error [6]), the incremental NaiveBayes class as base learner and the *SingleClassifierDrift* class were used, and provided a starting point to implement the specific components of our approach. The *SingleClassifierDrift* class implements the drift detection method proposed in [9] and adapts to it by learning a new classifier (i.e., discards previous concept representations).

The approach was tested with synthetic and real world datasets, in all but in one dataset (*Elec2*), the concept changes and contextual attributes are generated artificially in order to control how the system is able to learn existing relations recurring concepts and context.

### *5.1. Datasets*

#### *5.1.1. Synthetic dataset*

*SEA Concepts* [24] using MOA [12] as the stream generator. *SEA Concepts* is a *benchmark* data stream that uses different functions to simulate concept drift, allowing control over the target concepts and its recurrence in our experiment. The *SEA Concepts* dataset has two classes {class0, class1} and three features with values between 0 and 10 but only the first two features are relevant. The target concept function classifies a record as class1 if $f_1 + f_2 \leqslant \theta$ and otherwise as class0, $f_1$ and $f_2$ are the two relevant features and $\theta$ is the threshold value between the two classes. Four target concept functions as in proposed in [24] are used, threshold values 8, 9, 7 and 9.5 are set to define these functions.

#### *5.1.2. Real world datasets*

As real world dataset, the Electricity Market Dataset(*Elec2*) [10] is used. The data was collected from the Australian New South Wales Electricity Market, where the electricity prices are not stationary and are affected by the market supply and demand. The market demand is influenced by context such as season, weather, time of the day and central business district population density. In addition, the

supply is influenced primarily by the number of on-line generators, an influencing factor for the price evolution of the electricity market is time. During the time period described in the dataset the electricity market was expanded with the inclusion of adjacent areas (Victoria state), which lead to more elaborated management of the supply as oversupply in one area could be sold interstate. The *Elec2* dataset contains 45.312 records obtained from 7 May 1996 to 5 December 1998, with one record for each half hour (i.e., there are 48 instances for each time period of one day). Each record has 5 attributes, the day of week, the time period, the NSW demand, the Victoria demand, the scheduled electricity transfer between states and the class label. The class label identifies the change of the price related to a moving average of the last 24 hours. The class level only reflects deviations of the price on a one day average and removes the impact of longer term price trends. As shown in [10] the dataset exhibits substantial seasonality and is influenced by changes in context. This motivates its use as a real world dataset in our experiments.

The real world emailing list (*Elist*) dataset [15] is a stream of email messages from different topics that are labelled by a user as interesting or junk according to his preferences. The original dataset is a stream of 1.500 records. We processed the dataset two times in a resulting data stream with 3.000 records.

### 5.2. Context and recurrence settings

As context for the *SEA* dataset we used a numerical context feature space with two features $a_1$ and $a_2$ with values between 1 and 4. It was generated independently as a context stream where the context attribute $a_1$ is equal to the target concept function number, and $a_2$ value is a random value, which introduces noise in the context stream. We generated 250.000 records and changed the underlying concept every 15.000 records. We tested a recurrence situation, where the order of concepts is repeated periodically (i.e. 1,2,3,4). The test was executed with a 10% noise value as in the original paper [24], this means the class value of the training record is wrong in 10% of the records, testing how sensitive the base learner is to noise.

For the Electricity Market (*Elec2*) dataset we have considered the classification problem to predict the changes in prices relative to the next half hour, using as predictive attributes, the time period, the NSW demand, the Victoria demand and the scheduled electricity transfer. As context we used the day of week attribute, as in [10] experiments using it lead to 10 different contextual clusters. We expect that the association of this context with the stored models achieves good accuracy results, when compared with the original paper results, that uses the SPLICE-2 algorithm [11]. However, one drawback of a real world dataset is that we do not know for sure what the actual *hidden context* is and when such changes occur, which makes more difficult to evaluate the obtained results. Anyway, this represents the learning scenario that motivates our approach, which is common in real-world problems. This dataset was also used in [9] to test the drift detection method in real world problems, achieving good performance results.

The Email List (*Elist*) dataset contains 2 recurrent concepts that change every 300 records. The first concept represents messages where user is only interested in medicine and in the second concept the interest changes to space and baseball. A contextual attribute location that is correlated with the target concept was used as context.

### 5.3. Experimental setup

Two different experiments were performed, one to test the accuracy of the learning process over time, and other also tests the accuracy of the learning process but in situations where the memory available is limited.

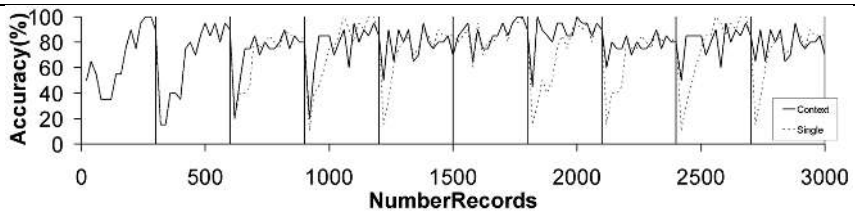



Fig. 2. Accuracy of the Proposed approach(Context) vs SingleClassifierDrift(Single) using the *Elist* dataset where concept recurrence follows periodicity. Black lines show when drift. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-2012-0552)

#### *5.3.1. Accuracy test experiments*

For both datasets the approach proposed in this paper is compared in terms of accuracy with the *SingleClassifierDrift* implemented in MOA [12]. For this purpose we monitored the records where change occurs and observed if the adaptation to change is as expected and the approach is able to learn the relations between concepts and context. In the case with the synthetic dataset we monitored if the mechanism is able to predict the underlying concept after change is detected by recording its accuracy. The *SingleClassifierDrift* approach also uses the incremental Naive Bayes algorithm and detects drift using the drift detection method [9], that does not consider recurrence. In the real world dataset we also compare results with an incremental Naive Bayes algorithm [7] (without any mechanism to adapt to drift), again to be used as reference.

The parameter values presented were set according to the different datasets (size of the windows) so an adequate value could be defined. For the experiments, the number records used in sample $D_n$ to compare the models (i.e. the number of records we consider the learned concept stable) was 20 for *Elist* and 100 for the *SEA* and *Elec2* dataset. The weights assigned to the utility function where 0.5 to each factor after the training period, while in this period the MSE factor was given all the weight. We used a training period for the context-concepts history of 60.000 records for the *SEA* dataset, 10.000 for *Elec2* dataset and 900 for the *Elist* dataset.

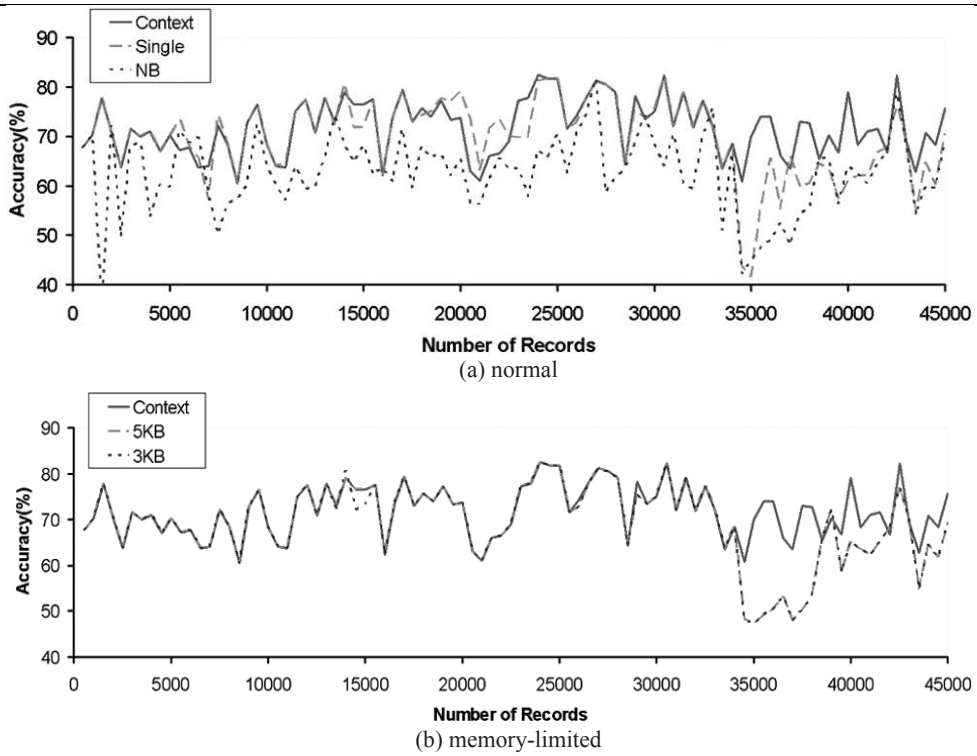
#### *5.3.2. Memory-awareness experiments*

We compared the memory-aware strategy in situations with memory constraints, using 7 KBytes, 5 KBytes and 3 KBbytes of available memory running the *SEA* and *Elec2* dataset. These values were chosen because they represent scenarios. On the one hand, scenarios where it is possible to store enough models and represent the different target functions, on the other hand scenarios where due to the strong memory constraints only a reduced number of models can be stored. Note that in the *SEA* of concepts dataset, this means being forced to store fewer models than existing target functions. Such constraint enables us to observe and measure how the accuracy declines as the memory available is reduced and fewer models can be kept. Also, it is important to understand the impact of discarding models and the utility of the stored models in the adaptation to recurrence in such memory constrained scenarios.

### *5.4. Results*

#### *5.4.1. Email list dataset*

Figure 2 shows the results from the accuracy experiment using the *Elist* dataset. In the figure one can observe that the context-aware approach leads to better overall accuracy than *SingleClassifierDrift*. Moreover, the proposed approach in general adapted faster to drift and the models retrieved using context




(a) normal

(b) memory-limited

Fig. 3. *Elec2* dataset experiment, testing the accuracy of the Proposed approach(Context) vs SingleClassifierDrift(Single) vs NaiveBayes (a) – Memory-awareness (b) comparing Proposed approach in scenarios with different memory limits. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-2012-0552)

integration were able to represent the underlying concept. This is not observed in the *SingleClassifierDrift* approach that always has to relearn the underlying concept from scratch after drift is detected. Furthermore, it is also noticeable that the proposed approach achieves a more stable accuracy over time, as it recovers much faster from drift than the approach without stored models. We can clearly see the improved adaptation after record 1.500 where the accuracy loss after a drift event is much softer than the *SingleClassifierDrift* approach. The proposed approach obtained 75.5% accuracy vs 72.4% single classifier approach and required to process 2.361 less records, as the system was able to recognize and exploit the concept recurrence. The integration of context enabled the system to exploit the associations between recurrent concepts and context as a way to track concept recurrence and in situations where this association occurs, it was possible to achieve better results.

### *5.4.2. Electricity market dataset*

As it can be seen in Fig. 3(a) the proposed approach obtained better overall accuracy results (i.e., 72%) when compared with *SingleClassifierDrift* (i.e., 69%) and incremental NaiveBayes (i.e., 62%) respectively. It is also possible to observe that the proposed approach achieves a more stable accuracy and recovers faster from changes. This can be seen more clearly around record 35.000. Despite the promising results, it is not possible to determine exact borders between concepts in this dataset and its relations with the context represented by the day of week. This limits the extent in which is possible to evaluate the results. However, such drawbacks are a consequence of learning from real world data.

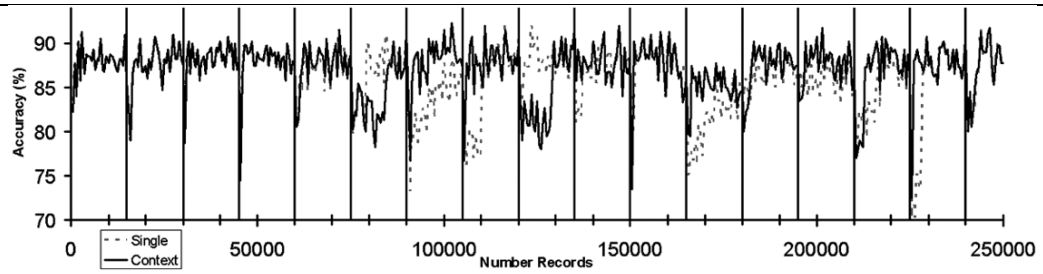



Fig. 4. Accuracy of the proposed approach (context) vs SingleClassifierDrift (Single) using the SEA concepts dataset. Black lines indicate when drift occurs. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-2012-0552)

In relation to the experiments with memory constraints the results in Fig. 3(b) show that the overall accuracy is similar between the experiments, in the order of 72% correctly classifier records. For the different tested scenarios the proposed approach still obtains better or equal overall accuracy than the memoryless approach (i.e., *SingleClassifierDrift*). However, the accuracy over specific periods depends on the model that is reused and which ones that were previously discarded in situations of memory scarcity, which is a direct consequence of the proposed model utility criteria. Such difference can be seen around record 35.000, where the test that kept the adequate models (i.e., 7Kb) are still able to show improved adaptation. In contrast, the accuracy loss in the tests with 5Kb and 3Kb is more evident, but results in less number of processed records, 9032 and 8731 less records respectively than 37345 that were processed in the 7Kb test.

### *5.4.3. SEA concepts dataset*

As can be observed in Fig. 4, in the *SEA concepts* dataset the proposed approach leads to better results than *SingleClassifierDrift* when recovering from concept drift. In general, the proposed approach adapted to drift faster and the models selected by the context-aware mechanism were able to represent the target concepts as can be seen by the accuracy obtained. The *SingleClassifierDrift* approach always has to relearn the underlying concept from scratch after drift is detected. However, in some situations for example at record 17.500 and 100.500, where selected model does not seems to represent the target concept at first and the *SingleClassifierDrift* approach is able to achieve better results. In this case, the fast adaptation of our greedy approach leads to the selection of a worse model. A more conservative approach could be used instead by increasing the number of records in the *warningWindow* or the selection threshold. It is also noticeable that the proposed approach achieves a more stable accuracy over time, because it recovers much faster from drift than the approach without stored models. This is significant where the proposed approach obtained 2.526 more correct predictions and required to process only 98.431 of the 250.000 records. The usage of context as part of our mechanism enables us to exploit the associations between recurrent concepts and context as a way to track concept recurrence and achieve better results in situations where this association exists.

In Fig. 5, the memory-aware approach is compared in scenarios with different available memory values. As expected, when the available memory is reduced, the accuracy decreases. In the test scenario with 7 Kbytes Fig. 5(a) it is possible to store 7 models, which allow us to keep more than one model for each concept. As a result the performance was almost the same (with only 11 more misclassified records) to the scenario without memory constraints where 10 models are stored. In the scenario with 5 Kbytes Fig. 5(b), the reduction in accuracy is more significant, with 4.318 more misclassified records and the accuracy curve starts to resemble the *SingleClassifierDrift* seen in 4 especially around records 120.000

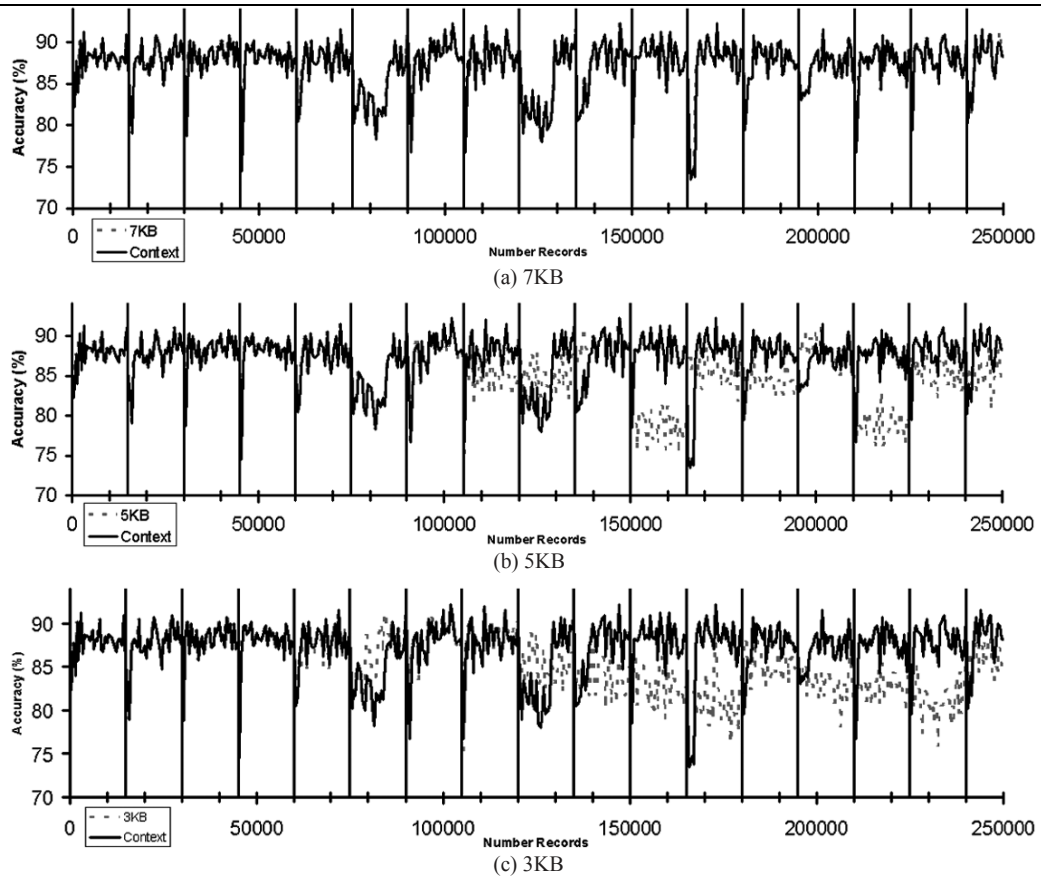



Fig. 5. Accuracy of the Proposed approach (Context) using the *SEA concepts* dataset in memory limited scenarios. Black lines show when drift occurs. (Colours are visible in the online version of the article; http://dx.doi.org/10.3233/IDA-2012-0552)

and 165.000 (where the concept with $\theta = 8$ is the target concept). Finally in the scenario with 3 Kbytes Fig. 5(c) only 3 models can be kept in memory and as a result the performance is further reduced, with more 4.918 misclassified records. In terms of the number of records processed the numbers for the 7 Kb, 5 Kb and 3 Kb where 98.474, 68.098 and 119.352 respectively. The increase in the number of processing records in the approach more constrained is because it has to relearn from scratch some concepts as the models that it can store do not represent the underlying concept, as it was forced to delete it previously due to the severe memory limit.

From the results we observe that the proposed approach achieves better overall accuracy, adapts faster to change and in situations of recurrent concepts is able to improve the accuracy between changes. Also, it reduces the number of processed records (i.e., training records processed by the base learner).

## 6. Conclusions and future work

In this work, we have proposed a context-aware data stream learning system, which improves the learning process accuracy when concepts reappear by integrating context information with learned concepts. The system exploits this context information to improve existing approaches to handle concept

drift and recurring concepts. We have also analyzed the main challenges associated with the integration of context, such as concept and context representation, storage, similarity and adaptation to reappearing concepts. The challenges are addressed through the definition of formal concept and context representations, similarity functions between these and an adaptation strategy to recurring concepts.

Experimental results to test the effectiveness and efficiency of the proposed learning system have been performed, using the SEA concepts, Electricity Market and Email List datasets. In what concerns the accuracy over time, the results show that the proposed approach adapts faster when compared to a single classifier approach that detects concept drift but does not exploit recurrence. Besides, in these recurring concept situations we observe an overall improvement on the accuracy and a reduction in the number of processed records. Based on these results the proposed technique could be used in several real world applications where recurring concepts are associated with context, for instance news recommender systems or spam filtering. Furthermore, the ability to process less records is of great interest for ubiquitous computation, as processing fewer records may extend the battery life of the ubiquitous device.

The proposed learning system is also sensitive to memory scarcity, which is another important factor to consider in ubiquitous devices. Experiments to analyze the performance of the learning system in situations of memory scarcity have been conducted. The results show that whenever possible the system keeps the more promising models in memory and forgets the ones that are considered less useful for situations of recurrence, increasing its performance accordingly.

In future work, we plan to use an ensemble to represent recurring concepts instead of a single classifier. Moreover, despite the promising results obtained so far, the context representation and similarity function used are not robust to uncertainty and noise. Therefore, it would be interesting to explore the usage of fuzzy functions, which may lead to a more robust and enhanced learning system. Furthermore, tuning the system specially in what relates to setting the window sizes and some thresholds represents an additional challenge for future work.

## Acknowledgments

The work of J.P. Bártolo Gomes is supported by a PhD Grant of the Portuguese Foundation for Science and Technology (FCT) and a mobility grant from Consejo Social of UPM that made possible his stay at the University of Portsmouth. This research is partially financed by project TIN2008-05924 of Spanish Ministry of Science and Innovation. We would also like to thank João Gama and Petr Kosina for their comments and encouragement. Thanks to the FCT project KDUDS (PTDC/EIA-EIA/98355/2008).

## References

[1] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda and R. Morales-Bueno, Early drift detection method, In *Fourth International Workshop on Knowledge Discovery from Data Streams*, Citeseer (2006), 77–86.

[2] R. Brezillon and J.C. Pomerol, Contextual knowledge sharing and cooperation in intelligent assistant systems, *Travail Humain* **62** (1999), 223–246.

[3] A.K. Dey, G.D. Abowd and D. Salber, A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, *Human-Computer Interaction* **16**(2) (2001), 97–166.

[4] P. Domingos and G. Hulten, Mining high-speed data streams, In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM New York, NY, USA (2000), 71–80.

[5] P. Domingos and G. Hulten, Catching up with the data: Research issues in mining data streams, In *Workshop on Research Issues in Data Mining and Knowledge Discovery*, Citeseer, 2001.

[6] J. Gama, Knowledge discovery from data streams (series: chapman & hall/crc data mining and knowledge discovery series), 2010.

[7] J. Gama and M.M. Gaber, Learning from data streams: Processing techniques in sensor networks, Springer-Verlag New York Inc, 2007.
[8] J. Gama and P. Kosina, Tracking recurring concepts with meta-learners, In *Progress in Artificial Intelligence: 14th Portuguese Conference on Artificial Intelligence, Epia 2009, Aveiro, Portugal, Proceedings*, Springer (October 12–15 2009), 423.
[9] J. Gama, P. Medas, G. Castillo and P. Rodrigues, Learning with drift detection, *Lecture Notes in Computer Science* (2004), 286–295.
[10] M. Harries, Splice-2 comparative evaluation: Electricity pricing, Technical report, The University of South Wales, 1999.
[11] M.B. Harries, C. Sammut and K. Horn. Extracting hidden context, *Machine Learning* **32**(2) (1998), 101–126.
[12] G. Holmes, R. Kirkby and B. Pfahringer, MOA: Massive online analysis, 2007 – http://sourceforge.net/projects/moa-datastream/.
[13] G. Hulten, L. Spencer and P. Domingos, Mining time-changing data streams, In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM New York, NY, USA (2001), 97–106.
[14] G.H. John and P. Langley, Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, **1** (1995), 338–345.
[15] I. Katakis, G. Tsoumakas and I. Vlahavas, Tracking recurring contexts using ensemble classifiers: An application to email filtering, *Knowledge and Information Systems*, 1–21.
[16] R. Klinkenberg, Learning drifting concepts: Example selection vs. example weighting, *Intelligent Data Analysis* **8**(3) (2004), 281–300.
[17] R. Klinkenberg and T. Joachims, Detecting concept drift with support vector machines, In *Proceedings of the Seventeenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc (2000), 494.
[18] J.Z. Kolter and M.A. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts, *The Journal of Machine Learning Research* **8** (2007), 2755–2790.
[19] A. Padovitz, S.W. Loke and A. Zaslavsky, Towards a theory of context spaces, In *Pervasive Computing and Communications Workshops, Proceedings of the Second IEEE Annual Conference on* (2004), 38–42.
[20] S. Ramamurthy and R. Bhatnagar, Tracking recurrent concept drift in streaming data using ensemble classifiers, In *Proc of the Sixth International Conference on Machine Learning and Applications* (2007), 404–409.
[21] J.C. Schlimmer and R. Granger, Beyond incremental processing: Tracking concept drift, In *Proceedings of the Fifth National Conference on Artificial Intelligence* **1** (1986), 502–507.
[22] A. Schmidt, M. Beigl and H.W. Gellersen, There is more to context than location, *Computers & Graphics* **23**(6) (1999), 893–901.
[23] M. Scholz and R. Klinkenberg, Boosting classifiers for drifting concepts, *Intelligent Data Analysis* **11**(1) (2007), 3–28.
[24] W.N. Street and Y.S. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM New York, NY, USA (2001), 377–382.
[25] A. Tsymbal, The problem of concept drift: Definitions and related work, *Computer Science Department, Trinity College Dublin*, 2004.
[26] P.D. Turney, Exploiting context when learning to classify, In *Proceedings of the European Conference on Machine Learning (ECML-93)* (1993), 402–407.
[27] H. Wang, W. Fan, P.S. Yu and J. Han, Mining concept-drifting data streams using ensemble classifiers, In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM New York, NY, USA (2003), 226–235.
[28] G. Widmer, Tracking context changes through meta-learning, *Machine Learning* **27**(3) (1997), 259–286.
[29] G. Widmer and M. Kubat, Learning in the presence of concept drift and hidden contexts, *Machine Learning* **23**(1) (1996), 69–101.
[30] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Pub, 2005.
[31] Y. Yang, X. Wu and X. Zhu, Combining proactive and reactive predictions for data streams, In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ACM (2005), 715.
[32] Y. Yang, X. Wu and X. Zhu, Mining in anticipation for concept change: Proactive-reactive prediction in data streams, *Data Mining and Knowledge Discovery* **13**(3) (2006), 261–289.
[33] M.J. Zaki, Editorial: Online, interactive, and anytime data mining, *SIGKDD Explorations* **3**(2), 2002.