Pattern Recognition Letters 34 (2013) 1018–1025

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

# RCD: A recurring concept drift framework

Paulo Mauricio Gonçalves Jr [a,b,*], Roberto Souto Maior de Barros [a]

[a] Centro de Informática, Universidade Federal de Pernambuco, Cidade Universitária, 50.740-560 Recife, Brazil
[b] Instituto Federal de Pernambuco, Cidade Universitária, 50.740-540 Recife, Brazil

ARTICLE INFO

*Article history:*
Received 2 January 2012
Available online 26 February 2013

Communicated by F. Tortorella

*Keywords:*
Data streams
Concept drift
Recurring contexts
On-line learning
Multivariate non-parametric statistical test

ABSTRACT

This paper presents recurring concept drifts (RCD), a framework that offers an alternative approach to handle data streams that suffer from recurring concept drifts (on-line learning). It creates a new classifier to each context found and stores a sample of data used to build it. When a new concept drift occurs, the algorithm compares the new context to previous ones using a non-parametric multivariate statistical test to verify if both contexts come from the same distribution. If so, the corresponding classifier is reused. The RCD framework is compared with several algorithms (among single and ensemble approaches), in both artificial and real data sets, chosen from frequently used algorithms and data sets in the concept drift research area. We claim the proposed framework had better average ranks in data sets with abrupt and gradual concept drifts compared to both the single classifiers and the ensemble approaches that use the same base learner.



## 1. Introduction

Recent years have witnessed an increase in the amount of applications that have to deal with information that occur in the form of a flow of data arriving continuously, in large quantities, and quickly, making it impossible to store data for later analysis, except in small amounts. Thus, data must be processed on-line, i.e., on arrival. This processing model is named data streams.

Concept drift is an area of data stream management that has been receiving much attention. Wang et al. (2011) describe that, in machine learning, "the term concept refers to the quantity that a learning model is trying to predict, i.e., the variable. Concept drift is the situation in which the statistical properties of the target concept change over time". Any application that tries to model human behavior is subject to concept drift, a common situation in data stream environments (Gaber et al., 2005). Examples include intrusion detection (Lane and Brodley, 1998), spam filtering (Delany et al., 2005), and credit card fraud detection (Wang et al., 2003).

Concept drifts can be categorized in different ways. One is by the speed of change. According to Minku et al. (2010), "drifts can be categorized as either abrupt, when the complete change occurs in only one time step, or gradual, otherwise". For example, "someone graduating from college might suddenly have completely different monetary concerns, whereas a slowly wearing piece of factory equipment might cause a gradual change in the quality of output parts" (Stanley, 2003). Stanley (2003) even divides gradual concept drifts in moderate and slow concept drifts, depending on the speed of change.

Another form to categorize concept drifts reflects the reason of change. A *real* concept drift "occurs when a set of examples has legitimate class labels at one time and different legitimate labels at another time" (Kolter et al., 2007), whereas a *virtual* concept drift occurs when "the target concepts remain the same but the data distribution changes" (Delany et al., 2005). In practice, they "often occur together" (Tsymbal et al., 2008).

Context recurrence is a common situation concerning concept drifts. According to Harries et al. (1998), domains where it can happen include "financial prediction, dynamic control and other commercial data mining applications". It is also claimed that "recurring contexts may be due to cyclic phenomena, such as seasons of the year, or may be associated with irregular phenomena, such as inflation rates or market mood". With the occurrence of many concept drifts, algorithms tend to better represent the last observed concepts, forgetting previously learned ones.

This paper presents the RCD framework, specifically developed to handle recurring concept drifts. It works by storing classifiers and samples of data used to build them. At predefined intervals, RCD compares the data distribution of actual data to stored data samples and, if they are similar, the stored classifier associated to that specific data sample is reused. To compare two data distributions and check their similarity, RCD uses a multivariate non-parametric statistical test.

To analyze the performance of the proposed framework in terms of accuracy and concept drift handling, six algorithms were

* Corresponding author at: Centro de Informática, Universidade Federal de Pernambuco, Cidade Universitária, 50.740-560 Recife, Brazil. Tel.: +55 81 2126 8430.
*E-mail addresses:* pmgj@cin.ufpe.br (P.M. Gonçalves Jr), roberto@cin.ufpe.br (Roberto Souto Maior de Barros).

0167-8655/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2013.02.005

used in the tests, including single and ensemble classifiers. The tests were performed with commonly used data sets in the concept drift research area, including abrupt and gradual concept drifts.

The rest of this paper is organized as follows: Section 2 presents approaches to deal with concept drifts, algorithms, and how they work. Section 3 presents RCD, the proposed framework. Section 4 presents the analysis of the comparison between RCD and the other algorithms in the selected data sets, showing accuracy, statistics, confidence intervals, and how the algorithms deal with concept drifts in the testing phase. Finally, Section 5 presents our conclusions and proposes future work.

## 2. Background

Different approaches to handle concept drifts exist. Here, we discuss four of them: adapting batch classifiers, detecting concept drifts and creating a new classifier to represent the new context, ensemble classifiers, and storing data related to built classifiers to deal with recurring concept drifts.

Some classifiers can be used directly in data stream environments when there is no concept drift, e.g., naive Bayes. However, adapting batch classifiers to deal with data streams and concept drifts is also common. One example that received much attention and research was adapting decision trees.

Very fast decision tree (VFDT) (Domingos and Hulten, 2000) can deal with huge amounts of data using few computational resources, with a performance comparable to a batch decision tree, given enough examples. It reads each example only once and needs a small amount of time to process it, allowing the creation of a classifier based on huge data sets. To identify the best attribute to be tested at a given node, it is sufficient to consider a small subset of the training examples that pass that node. To identify exactly how many examples are needed to each node, it uses a statistical result known as Hoeffding bound (Hoeffding, 1963; Maron and Moore, 1994), which is independent of the probability distribution that generated the observations.

The concept-adapting very fast decision tree (CVFDT) (Hulten et al., 2001) "is an extension to VFDT, which maintains VFDT's speed and accuracy advantages but adds the ability to detect and respond to changes in the example-generating process". It uses a sliding window of examples to try to keep its model up to date. For each new arriving example, it recomputes the statistics, reducing the influence of the oldest examples. If the concept starts to change, alternative attributes will increase their information gain, making the split no longer pass the Hoeffding test. At this moment, an alternative tree begins to grow with the new best attribute at its root. If this subtree becomes more accurate than the old one on new data, it is substituted.

Some approaches try to identify concept drifts and create a new classifier when they occur. An example is the drift detection method (DDM) (Gama et al., 2004), which can be used with any classifier and detects changes in the data distribution using the notion of *context*: a set of contiguous examples where the distribution is stationary. The idea behind the drift detection is controlling the algorithm's error rate. Statistical theory guarantees that, when the distribution is stationary, the error will decrease, otherwise it will increase. The method controls the classifier error dynamically, defining a warning level and a drift level to the actual context. A new context is declared if, in a sequence of examples, the error increases reaching *warning* or *drift* level, at example $k_w$ or $k_d$, respectively. The algorithm then trains a new classifier using only the examples obtained after $k_w$.

The early DDM (EDDM) (Baena-García et al., 2006) works similarly but uses the distance between two errors, instead of controlling solely the amount of error of the classifier, to identify concept drifts. The authors argued that this approach was more adequate to detect gradual concept drifts.

The accuracy updated ensemble (AUE) (Brzeziński and Stefanowski, 2011) improves AWE (Wang et al., 2003). Both use classifier ensembles and are associated with weights that are updated as data arrive. The main differences between them are that AUE uses incremental instead of batch classifiers; it proposes a simpler weighting function to avoid zeroing the weight of all classifiers, a possible situation in AWE; and updates classifiers only if they have high accuracy in recent data.

The weighted majority algorithm (WMA) (Blum, 1997) implements a weighted ensemble classifier to specifically handle concept drifts. In WMA, all the arriving instances are passed to all classifiers in the ensemble. The initial weight of the classifiers is 1 and, if a classifier makes an error, its weight is reduced by a factor of $\beta$, provided it is greater than a specified minimum threshold. Then, the classifier is trained. After all the ensemble classifiers have been trained, their weights are normalized.

Dynamic weighted majority (DWM) (Kolter et al., 2007) is an ensemble classifier that extends WMA to add and remove classifiers according to the algorithms' global and local performance. If the ensemble commits an error, a classifier is added. If one classifier commits an error, its weight is reduced. If after many examples a classifier continues with low accuracy, it is removed from the ensemble.

To deal with recurring concept drifts, different approaches have been proposed, usually storing information about the concepts and, if necessary, reusing them.

FLORA3 (Widmer and Kubat, 1996) deals with categorical attributes and represents data using a simple representation language based on attribute–value logic without negation. It also uses the notion of description items, which is a conjunction of attribute–value pairs. FLORA represents a *concept description* in the form of three description sets: ADES (*Accepted Descriptors*) matches positive examples and is used to classify new incoming examples, NDES (*Negative Descriptors*) summarizes the negative examples and is used to prevent over-generalization of ADES, and PDES (*Potential Descriptors*) acts as a repository of hypotheses that are currently too general but might become relevant in the future.

SPLICE-2 (Harries et al., 1998) also deals with categorical attributes but the classifier training is made in batch mode: it assumes that a concept will be stable over some time interval. Sequences of examples in the data set are combined into intervals if they appear to belong to the same context. SPLICE-2 then attempts to cluster similar intervals by applying the notion that similarity of context is reflected by the degree to which intervals are well classified by the same concept. To perform training, each example must have an attribute that uniquely identifies its position in the sequence of training data. SPLICE-2 begins by guessing an initial partitioning of the data; subsequent stages refine the initial guess.

These two methods have the disadvantage of only being able to deal with categorical attributes while RCD can deal with both *numerical* and *categorical* attributes. An additional disadvantage of SPLICE-2 is training in batch mode.

A more recent approach to deal with recurring concept drifts was proposed by Ramamurthy and Bhatnagar (2007), here named ensemble building (EB). EB is an ensemble classifier, similar to AWE and AUE, where classifiers are created from sequential data chunks and a subset of them is used in the ensemble. Each classifier weight is based on its accuracy on the last data chunk. If none of the stored classifiers performs above a specified threshold in the current chunk, a new classifier is built and stored in the ensemble, indicating that the stored classifiers do not correctly represent actual data. Thus, differently than AWE and AUE, which create a new classifier for each data chunk, this proposal only adds a new classifier to the ensemble if no classifier is well suited to actual data.