

A Survey on Concept Drift Adaptation

JOÃO GAMA, University of Porto, Portugal

INDRÉ ŽLIOBAITÉ, Aalto University, Finland

ALBERT BIFET, Yahoo! Research Barcelona, Spain

MYKOLA PECHENIZKIY, Eindhoven University of Technology, the Netherlands

ABDELHAMID BOUCHACHIA, Bournemouth University, UK

Concept drift primarily refers to an online supervised learning scenario when the relation between the input data and the target variable changes over time. Assuming a general knowledge of supervised learning in this paper we characterize adaptive learning process, categorize existing strategies for handling concept drift, discuss the most representative, distinct and popular techniques and algorithms, discuss evaluation methodology of adaptive algorithms, and present a set of illustrative applications. This introduction to the concept drift adaptation presents the state of the art techniques and a collection of benchmarks for researchers, industry analysts and practitioners. The survey aims at covering the different facets of concept drift in an integrated way to reflect on the existing scattered state-of-the-art.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: concept drift, change detection, adaptive learning

ACM Reference Format:

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. 2013. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.* 1, 1, Article 1 (January 2013), 35 pages.

DOI = 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

1. INTRODUCTION

Our digital universe is rapidly growing. The volume of data generated in 2012 has been estimated to surpass 2.8 zetabytes (2.8 trillion gigabytes) as reported in the IDC survey [Gantz and Reinsel 2012]. Efficient and effective tools and analysis methods for dealing with the ever-growing amount of data in different applications and fields are of paramount need. Very often data comes in the form of streams rendering its analysis and processing even more resource demanding.

Traditionally in data mining data is first collected and then processed in an *offline* mode. For instance, predictive models are trained using historical data given as a set of pairs (input, output). Models trained in such a way can be afterwards applied for predicting the output for new unseen input data. However, streaming data can not be processed similarly because data comes continuously over time and possibly is never-ending. Accommodating such data in the machine's main memory is impractical and often infeasible. Hence, only an *online* processing is suitable. In this case, predictive models can be trained either incrementally by continuous update or by retraining using recent batches of data.

In dynamically changing and non-stationary environments, the data distribution can change over time yielding the phenomenon of *concept drift* [Schlimmer and Granger 1986; Widmer and Kubat 1996]. The *real* concept drift¹ refers to changes in the conditional distribution of the output (i.e., *target variable*) given the input (*input features*), while the distribution of the input may stay unchanged. A typical example of the real concept drift is a change in user's interests when following an online news stream. Whilst the distribution of the incoming news documents often remains the same, the conditional distribution of the interesting (and thus not interesting) news

¹The term *real* refers to one particular type of concept drift. It doesn't mean that other types of drift are not concept drifts.

documents for that user changes. *Adaptive learning* refers to updating predictive models online during their operation to react to concept drifts.

Over the last decade research related to learning with concept drift has been increasingly growing and many drift-aware adaptive learning algorithms have been developed. In spite of the popularity of this research topic, no comprehensive survey on concept drift handling techniques is available to the community. One of the reasons for that is that the problem is of a wide scope and spans across different research fields. Moreover, terminology is not well established, thus similar adaptive learning strategies have been developed independently under different names in different contexts.

Taking account of the current picture of research on concept drift, being very popular but also scattered among various communities, there is a strong need for a comprehensive summary of the research done so far to unify the concepts and terminology among the researchers and to survey the state-of-the-art methodologies and techniques investigated over the past.

Several reviews related to drift-aware learning are available. However, they either do not focus exclusively on concept drift or relate to specific topics of adaptive learning. Thus these reviews are fragmented and/or are outdated. Currently the most cited survey on concept drift was published back in 2004 in [Tsymbol 2004]. The following overviews which are related to the topic of concept drift focused on ensemble techniques [Kuncheva 2004; 2008], inductive rule learning algorithms [Maloof 2010], or mainly on non-incremental learning techniques [Zliobaite 2009] that can use computational resources unrestrictedly, thus were limited in scope. Reviews on data streams [Gaber et al. 2005; Gama 2010; Bifet et al. 2011a] only partially deal with data drift. Data streams research covers adaptive learning only to some extent, while the main focus remains on making learning algorithms incremental and optimizing the balance of computational resources and the predictive accuracy.

Several reviews are limited to specific application fields. A focused position paper [Grisogono 2006] presents a set of requirements for complex adaptive systems to be used for defence. A recent focused review [Kadlec et al. 2011] surveys adaptation mechanisms that have been used for soft sensors. Finally, a recent article [Moreno-Torres et al. 2012] focuses on describing various ways how data distribution can change over time and only briefly covers adaptation techniques from dataset shift community perspective, mostly leaving out works on concept drift. A recent review [Alberg et al. 2012] focuses on decision trees.

The present contribution provides an integrated view on handling concept drift, by surveying adaptive learning methods, presenting evaluation methodologies and discussing illustrative applications. It focuses on online supervised learning when the relation between the input features and the target variable changes over time.

The paper is organized as follows. In Section 2 we introduce the problem of concept drift, characterize adaptive learning algorithms and present motivating application examples. Section 3 presents a comprehensive taxonomy of methods for adaptive learning. Section 4 discusses the experimental settings and evaluation methodologies of adaptive learning algorithms. Section 5 concludes the survey.

2. ADAPTIVE LEARNING ALGORITHMS

Learning algorithms often need to operate in dynamic environments, which is changing unexpectedly. One desirable property of these algorithms is their ability of incorporating new data. If the data generating process is not strictly stationary (as applies to most of the real world applications), the underlying concept, which we are predicting (for example, interests of a user reading news), may be changing over time. The ability to adapt to such *concept drift* can be seen as a natural extension for the incremental learning systems [Giraud-Carrier 2000] that learn predictive models example by

example. *Adaptive learning algorithms* can be seen as advanced incremental learning algorithms that are able to adapt to evolution of the data generating process over time.

This section introduces concept drift and characterizes adaptive learning.

2.1. Setting and definitions

In machine learning the supervised learning problem is formally defined as follows. We aim to predict a target variable $y \in \mathbb{R}^1$ in regression tasks (or y categorical in classification tasks) given a set of input features $X \in \mathbb{R}^p$. An *example* is one pair of (X, y) . For instance, X is a set of sensor readings of a chemical process at 2 p.m. on the 2nd of January and $y = \text{“good”}$ is the true quality of the produced product at that time. In the *training examples*, that are used for model building, both X and y are known. In the new examples, on which the predictive model is applied, X is known, but y is not known at the time of prediction.

According to the Bayesian Decision Theory [Duda et al. 2001], a classification can be described by the prior probabilities of the classes $p(y)$ and the class conditional probability density functions $p(X|y)$ for all classes $y = 1, \dots, c$, where c is the number of classes. The classification decision is made according to the posterior probabilities of the classes, which for class y can be represented as

$$p(y|X) = \frac{p(y)p(X|y)}{p(X)}, \quad (1)$$

where $p(X) = \sum_{y=1}^c p(y)p(X|y)$. Here equal costs of misclassification are assumed.

The type of the target variable space depends on the task. In classification the target variable takes categorical values (class labels), while in regression the target variable takes continuous values.

We can distinguish two learning modes: *offline* learning and *online* learning. In offline learning the whole training data must be available at the time of model training. Only when training is completed the model can be used for predicting. In contrast, online algorithms process data sequentially. They produce a model and put it in operation without having the complete training data set available at the beginning. The model is continuously updated during operation as more training data arrives.

Less restrictive than online algorithms are *incremental* algorithms that process input examples one-by-one (or batch-by-batch) and update the decision model after receiving each example. Incremental algorithms may have random access to previous examples or representative/selected examples. In such a case these algorithms are called *incremental algorithms with partial memory* [Maloof and Michalski 2004]. Typically, in incremental algorithms, for any new presentation of data, the update operation of the model is based on the previous one. *Streaming* algorithms are online algorithms for processing high-speed continuous flows of data. In streaming, examples are processed sequentially as well and can be examined in only a few passes (typically just one). These algorithms use limited memory and limited processing time per item.

In the setting that we are considering data arrives *online*, often in real time, forming a stream which is potentially infinite. The machinery is given input data that has just arrived to predict its target variable(s). That is, a prediction machinery is defined as a mapping function between the input (feature) space and its corresponding output (target) space. For instance, given sensor readings in a chemical production process the task is to predict the quality of the product (output).

Because data is expected to *evolve over time* - especially in dynamically changing environments, where non-stationarity is typical, its underlying distribution can change dynamically over time. The general assumption in the concept drift setting is that the change happens unexpectedly and is unpredictable, although in some particular

real-world situations the change can be known ahead of time in correlation with the occurrence of particular environmental events. But solutions for the general case of drift entail the solutions for the particular cases. Moreover the change may take different forms, i.e. the input data characteristics or the relation between the input data and the target variable may change.

Formally *concept drift* between time point t_0 and time point t_1 can be defined as

$$\exists X : p_{t_0}(X, y) \neq p_{t_1}(X, y), \quad (2)$$

where p_{t_0} denotes the joint distribution at time t_0 between the set of input variables X and the target variable y . Changes in data can be characterized as changes in the components of this relation [Kelly et al. 1999; Gao et al. 2007]. In other terms,

- the prior probabilities of classes $p(y)$ may change,
- the class conditional probabilities $p(X|y)$ may change, and
- as a result, the posterior probabilities of classes $p(y|X)$ may change affecting the prediction.

We are interested to know two implications of these changes: First, we are interested to know (i) whether the data distribution $p(y|X)$ changes and affects the predictive decision and (ii) whether the changes are visible from the data distribution without knowing the true labels, i.e. $p(X)$ changes. From a predictive perspective only the changes that affect the prediction decision require adaptation.

We can distinguish the following types of drifts:

- (1) *Real concept drift* refers to changes in $p(y|X)$. Such changes can happen either with or without change in $p(X)$. Real concept drift has been referred to as *concept shift* in [Salganicoff 1997] and *conditional change* in [Gao et al. 2007].
- (2) *Population drift* refers to changes in the population from which future samples will be drawn compared the design/training sample was drawn [Kelly et al. 1999].
- (3) *Virtual drift* happens if the distribution of the incoming data changes (i.e., $p(X)$ changes) without affecting $p(y|X)$ [Delany et al. 2005; Tsymbal 2004; Widmer and Kubat 1993]. However virtual drift has had different interpretations in the literature:
 - Originally a virtual drift has been defined [Widmer and Kubat 1993] to occur due to incomplete data representation rather than change in concepts in reality,
 - Virtual drift corresponds to change in data distribution that leads to changes in the decision boundary [Tsymbal 2004],
 - Virtual drift is a drift that does *not* affect the target concept [Delany et al. 2005],
 - Virtual drift has been also referred to as *temporary drift* [Lazarescu et al. 2004], *sampling shift* [Salganicoff 1997] and *feature change* [Gao et al. 2007],
 In this paper *virtual drift* refers to change in the data distribution $p(X)$.

Example: Consider an online news stream of articles on real estate. The task for a given user is to classify the incoming news into *relevant* and *not relevant*. Suppose that the user is searching for a new apartment, then news on dwelling houses are *relevant* whereas holiday homes are *not relevant*. If the editor of the news portal changes, the writing style changes as well, but the dwelling houses remain *relevant* for the user. This scenario corresponds to population drift. If due to a crisis more articles on dwelling houses come out and less articles on holiday homes do, but the editor, the writing style, and the interests of the user remain the same, this situation corresponds to drift in prior probabilities of the classes. If on the other hand the user has bought a house and starts looking for a holiday destination, dwelling houses become *not relevant* and holiday homes become *relevant*. This scenario corresponds to the real concept

drift. In this case that the writing style and the priors remain the same. It may happen that all types of drifts takes place at the same time.

Figure 1 illustrates these concepts. We see that only the real concept drift changes the decision boundary and the previous decision model becomes obsolete. In reality the virtual drift, changing priors or novelties may appear in combination with the real drift, in those cases the decision boundary is also affected.

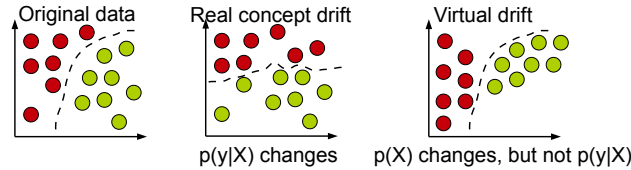


Fig. 1. Types of drifts: circles represent instances, different colors represent different classes.

This survey primarily focuses on handling the real concept drift which is not visible from the input data distribution. In many cases the techniques that handle the real concept drift can also handle drifts that manifest in the input data distributions, but not vice versa. The techniques that handle real concept drift typically rely on the feedback about the predictive performance. In this present paper, drift that can be detected from the incoming data distribution is not covered. This corresponds to tracking drifting priors (an interested reader is referred to [Zhang and Zhou 2010]), and novelty detection (an interested reader is referred to [Markou and Singh 2003; Masud et al. 2011]). Furthermore, semi-supervised drift handling techniques based on clustering (an interested reader is referred to [Aggarwal 2005; Bouchachia et al. 2010]) are not discussed in this paper.

2.2. Changes in data over time

Changes in data distribution over time may manifest in different forms, as illustrated in Figure 2 on a one-dimensional data. There changes happen in the data mean. Drift

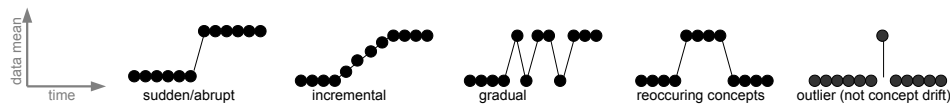


Fig. 2. Patterns of changes over time (outlier is not concept drift).

may happen *suddenly/abruptly* by switching from one concept to another (e.g. replacing a sensor in a chemical plant that has a different calibration), or *incrementally* consisting on many intermediate concepts in between (e.g. a sensor slowly wears off and becomes less accurate). Drift may happen suddenly (e.g. the topics of interest that one is surveying as a credit analyst may suddenly switch from, for instance, meat prices to public transportation) or *gradually* (e.g. relevant news topics change from dwelling to holiday homes, while the user does not switch abruptly, but rather keeps going back to the previous interest for some time). One of the challenges for concept drift handling algorithms is not to mix the true drift with an *outlier* or noise which refers to a once-off random deviation or anomaly (see [Chandola et al. 2009] for outlier detection). No adaptivity is needed in the latter case. Finally, drifts may introduce new concepts that were not seen before, or previously seen concepts may *reoccur* after some time

(e.g. in fashion). Changes can be further characterized by severity, predictability and frequency [Minku et al. 2010; Kosina et al. 2010].

Most of adaptive learning techniques implicitly or explicitly assume and specialize in some subset of concept drifts. Many of them assume sudden non-reoccurring drifts. But in reality often mixtures of many types can be observed.

2.3. Requirements for predictive models in changing environments

Predictive models that operate in these settings need to have mechanisms to detect and adapt to evolving data over time, otherwise their accuracy will degrade. As time passes, the decision model may need update taking into account the new data or gets completely replaced to meet the changed situation. Predictive models are required to:

- (1) detect to concept drift (and adapt if needed) as soon as possible;
- (2) distinguish drifts from noise and be adaptive to changes, but robust to noise;
- (3) operate in less than example arrival time and use not more than a fixed amount of memory for any storage.

2.4. Online adaptive learning procedure

The online adaptive learning is formally defined as follows. A *decision model* is a function \mathcal{L} that maps the input variables to the target: $y = \mathcal{L}(X)$. A *learning algorithm* specifies how to build a model from a set of data instances.

Online adaptive learning procedure is the following.

- (1) *Predict*. When new example X_t arrives, a prediction \hat{y}_t is made using the current model \mathcal{L}_t .
- (2) *Diagnose*. After some time we receive the true label y_t and can estimate the *loss* as $f(\hat{y}_t, y_t)$, and
- (3) *Update*. we can use the example (X_t, y_t) for model update to obtain \mathcal{L}_{t+1} .

Depending on the computational resources, the data may need to be discarded once processed using the latest version of the model $\mathcal{L}_{t+1} = \text{train}((X_t, y_t), \mathcal{L}_t)$. Alternatively, some of the past data may remain accessible $\mathcal{L}_{t+1} = \text{train}((X_i, y_i), \dots, (X_t, y_t), \mathcal{L}_t)$. There are different ways of handling data online (e.g., partial memory: some of the examples are stored and used regularly in the training, window-based: data is presented as chunks, instance-based: an example is processed upon its arrival). More details on various schemes related to data presentation will follow in Section 3.

After updating the model new example X_{t+1} arrives and the loop of receiving-predicting-feedback-model update continues infinitely. At some time steps one may choose to preserve the current model $\mathcal{L}_{t+1} = \mathcal{L}_t$.

Figure 3 depicts a generic schema for an online adaptive learning algorithm. In a nutshell, the memory module defines how and which data is presented to the learning algorithm (learning module). The loss estimation module tracks the performance of the learning algorithm and sends information to the control module to update the model if necessary. Section 3 will discuss the four modules of the system (Data Management, Learning, Change Detection, Loss Estimation) in detail.

This setting has variations where, for instance, the true values for the target variable (feedback) come with a delay or are not available at all. Moreover, new examples for prediction may arrive before we get feedback for the data that has already been processed. In such a case model update would be delayed, but the principles of operation remain the same. Finally, in some settings we may need to process examples in batches rather than one-by-one.

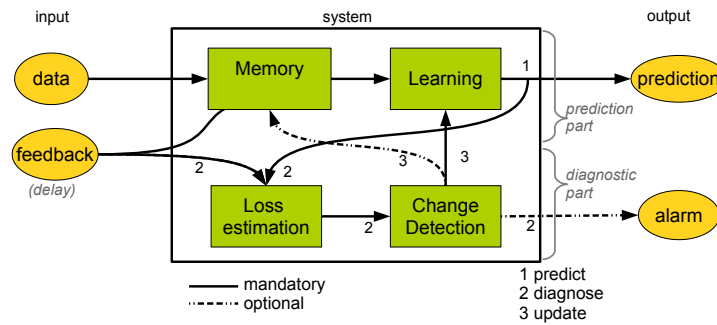


Fig. 3. A generic schema for an online adaptive learning algorithm.

2.5. Illustrative applications

The problem of concept drift has been recognized and addressed in multiple domains and application areas, including medicine [Kukar 2003; Tsymbal et al. 2006], industry [Pechenizkiy et al. 2009], education [Castillo et al. 2003], and business [Klinkenberg 2003]. Applications requiring adaptation can be grouped into four categories:

Monitoring and control. includes detection of anomalous behavior and adversary activities on the web, computer networks, telecommunications, financial transactions and other application areas where an abnormal behavior needs to be signaled, and it is often formulated as a detection task.

Management and strategic planning. includes predictive analytics tasks, such as evaluation of creditworthiness, demand prediction, food sales, bus travel time prediction, crime prediction by region.

Personal assistance and information. includes recommender systems, categorization and organization of textual information, customer profiling for marketing, personal mail categorization and spam filtering.

Ubiquitous environment applications. includes a wide spectrum of moving and stationary systems, which interact with changing environments, for instance moving robots, mobile vehicles, smart household appliances.

Next we discuss motivating application cases within each category to illustrate the demand of adaptive learning systems that can handle concept drift.

2.5.1. Monitoring and Control. In monitoring and control applications data is often presented in a form of time series. Two most typical learning tasks are time-series forecasting (regression task) or anomaly detection (classification task).

Online mass flow prediction in an industrial boiler [Pechenizkiy et al. 2009] is an example application in the monitoring and control category. Mass flow prediction would help to improve operation and control of the boiler. In steady operation, combustion is affected by the disturbances in the feed rate of the fuel and by the incomplete mixing of the fuel in the bed. Knowing the mass flow is important for boiler control. The system takes fuel that is continuously mixed inside and transferred from a container to the boiler. Scaling sensors located under the container provide streaming data. The task is to predict (estimate) the mass flow in real time.

Concept drift happens due to the following reasons. Fuel feeding is a manual and non standardized process, which is not necessarily smooth and can have short interruptions. Each operator can have different habits. The process characteristics may depend on the type and the quality of fuel used. The main focus for an adaptive learning algorithm is to handle two types of changes: an abrupt change to feeding and slower

but still abrupt switch to burning. One challenge for learning is that the feedback (the ground truth of mass flow) is not available at all, it can only be approximately estimated by retrospectively inspecting the historical data. An additional challenge is to deal with specific one-sided outliers that can be easily mistaken for changes.

Traditional approaches (such as ADWIN) for explicit change detection based on the monitoring of the raw sensor signal or streaming error of the regressors give reasonable results. They can be improved by considering the peculiarities of the application.

2.5.2. Management and strategic planning. The Smart Grid (SG) is an electric system that uses two-way digital information, cyber-secure communication technologies, and computational intelligence in an integrated fashion across heterogeneous and distributed electricity generation, transmission, distribution and consumption to achieve energy efficiency. A key and novel characteristic of SG's is the intelligent layer that analyzes the data produced by smart meters allowing companies to develop powerful new capabilities in terms of grid management, planning and customer services for energy efficiency. The advent of SG's has changed the way energy is produced, priced and billed. The key aspect of SG's is distributed energy production, namely renewable energies. The penetration of renewable energies (solar, wind, etc.) is increasing fast and power forecasting becomes an important factor in defining the operation planning policies to be adopted by a Transmission System Operator.

When observing the literature in wind power prediction [Monteiro et al. 2009], one realizes that most proposals are based on an off-line training mode, building a static model that is then used to produce predictions. This option rely in assumptions of stationarity of the wind electric power model, which must be strongly questioned [Bremnes 2004; Bessa et al. 2009]. Using real data from three distinct wind parks, [Bessa et al. 2009] presents the merits of on-line training against off-line training of neural networks. The authors point out the evolving nature of data and the presence of concept drift in wind pattern behavior.

2.5.3. Personal assistance and information. Text classification has been a popular topic in machine learning for decades. However, interesting applications related to the problem of concept drift appeared relatively recently. Examples of text stream applications include e-mail classification [Carmona-Cejudo et al. 2010], e-mail spam detection [Lindstrom et al. 2010] and sentiment classification [Bifet and Frank 2010]. Sentiment classification is a popular task in social media monitoring, customer feedback analysis and other applications.

The main source of concept drift in e-mail classification and spam filtering are due to changing e-mail content and presentation (virtual drift), as well as adaptive behaviour of spammers trying to overcome spam filters (may be virtual or real). Besides, users may change their attitude towards particular categories of e-mails starting or stopping to consider them spam (real drift). In sentiment classification the vocabulary used to express positive and negative sentiments may change over time. Since the collection of documents is not static (virtual drift, novelties), the feature space representing the current collection is dynamic that may require specific updates of the models.

Various adaptive learning strategies have been used in this domain, including individual methods like case-based reasoning, and ensembles, either evolving or with an explicit detection of changes by means of change detectors (Section 3.2).

Availability of feedback is a serious challenge in personal assistance and information. The dilemma is that if feedback is easily available, that implies no need for automated predictions. In e-mail classification we can hope that from time to time we will receive feedback from the user in case of misclassifications or can design an active learning system (e.g. [Zliobaite et al. 2013]), which from time to time asks the user to

provide labels on demand. However, when possible we need to aim at automatic ways for obtaining the true labels.

Suppose for monitoring the attitude of people towards a political party we want to classify the polarity or sentiment of tweets from Twitter. Labelling tweets manually as positive or negative is a laborious and expensive task. However, tweets may have author-provided sentiment indicators: changing sentiment is implicit in the use of various types of emoticons. Hence we may use these to label the training data. Smileys or emoticons are visual cues that are associated with emotional states. They are constructed using the characters available on a standard keyboard, representing a facial expression of emotion. By using emoticons, authors of tweets annotate their own text with an emotional state. Annotated tweets can be used to train a sentiment classifier.

Building a content-based filter for adaptive news access present rather different perspective on text classification in streaming settings. The goal is to learn incrementally and keep up to date a user model for news story classification. A simple yet effective approach has been proposed in [Billsus and Pazzani 2000]. For each user an adaptive learning system is built consisting of a simple ensemble with separate models for short-term and long-term interests of users. A stable Naive Bayes classifier is used for modelling the long term interests of a user and the Nearest Neighbour classifier captures the short term interests of the user. For the short-term interests model a fixed size window over the liked news stories is maintained and/or instances are weighed with respect to their age. No explicit change detection is used for monitoring either of the short-term or long-term interests. The true labels of some of the instances come naturally due to a positive relevance feedback, i.e. a user accessing a particular news item provides the signal that the item is relevant to his or her interests.

On the other hand, recommender systems is a broad application in the personal assistance and information category [Bobadilla et al. 2013; Adomavicius and Tuzhilin 2005]. Interests of the data mining community in recommender systems domain have been boosted by the NetFlix competition². One of the lessons learnt by the winning teams was that taking temporal dynamics into account substantially contributes towards building accurate models. Modelling user interests and handling concept drift were the other interesting aspects. In collaborative filtering, modelling of user interests relies primarily on the availability of other ratings already provide by the users. In a realistic application case, the data is highly imbalanced. Some movies are very popular, while most of the movies are not; some users rate many movies, but many other rate only a few. The rating matrix is high-dimensional and extremely sparse containing only about 1% of non-zero elements. Such properties make the application of most supervised learning techniques inapplicable and motivate the development of advanced collaborative filtering approaches. The sources and the nature of change can be diverse. Both items and users are changing over time. The item-side effects include first of all changing product perception and popularity. Popularity of some movies is expected to follow seasonal patterns. The user-side effects include changing tastes and preferences of users, some of which may be short-term or contextual and therefore likely reoccurring (mood, activity, company), changing perception of rating scale, possible change of rater within household and alike problems. The winning team developed an ensemble approach including multiple models for handling these various kinds of changes. As suggested in [Koren 2010] popular windowing and instance weighing approaches for handling concept drift are not the best choice for each kind of changing behaviour, simply because in collaborative filtering e.g. the relations between ratings is of the main importance for predictive modelling.

²www.netflixprize.com

2.5.4. Ubiquitous environment applications. Autonomous vehicle control is an application case in the ubiquitous environment category. DARPA Grand Challenge was a prize competition open for teams developing fully autonomous driverless vehicles. Stanley, the car of the winning team of the second long distance competition³ organized in 2005, had a highly reliable system on board that allowed the car to drive through different off-road environments at relatively high speeds. Among other intelligent components of the systems, Stanley had to perform identification and avoidance of obstacles and road finding [Thrun et al. 2006].

The Stanley team combined several ideas for automated terrain labelling using streaming laser sensor data for short and medium range obstacle avoidance and streaming data from the colour camera to learn the concept of the drivable surface and use it for velocity control (if the surface further ahead is non-drivable the car should slow down). That is the vision module had to maintain an accurate classifier for identifying drivable and non-drivable regions in the image stream. An adaptive learning approach was necessary for performing this task reliably because of many changing and not easily measurable factors such as surface material, lighting conditions or dust or dirt on the camera itself that effect the target concept.

The classification task was to model the colour of the drivable terrain. The Stanley team used an adaptive Mixture of Gaussians. The model had to adapt to slowly changing lighting conditions and to abruptly changing surface colours, e.g., moving from a paved to an unpaved road. For gradual adaptation the internal Gaussian was adjusted, for the rapid adaptation the Gaussians were replaced with new ones. The required speed of adaptation depended on the road conditions.

These real-world application cases present evidence that adaptive learning makes it possible to address complex learning problems which would not be feasible to tackle in the stationary settings. These examples also illustrate that it is important first to understand the nature and source of drift and only then engineer an effective adaptive learning strategy.

3. TAXONOMY OF METHODS FOR CONCEPT DRIFT ADAPTATION

In this section we propose a new taxonomy for adaptive algorithms that learn a predictive model from evolving data with unknown dynamics. The two main abstract concepts are *memory* and *forgetting* data and/or models. We discuss a set of representative and popular algorithms that implement adaptation strategies. The discussion is organized around the four modules of adaptive learning algorithms that were identified in Figure 3: *memory*, *change detection*, *learning*, and *loss estimation*. The main idea behind presenting the taxonomy in four separate modules rather than a tree of methods is to see adaptive learning systems as consisting of modular components. An overview of the taxonomy is presented in Figure 4, and the methods within each module are further detailed in Figures 5, 6, 7 and 8.

3.1. Memory

Learning under concept drift requires not only to update the predictive model with new information, but also forget the old information. We consider two types of memory: short term memory represented as data and long term memory represented as generalization of data – models. In this subsection, we analyze the short term memory under two dimensions as illustrated in Figure 5: *i) Data management* specifies which data is used for learning; and *ii) Forgetting mechanism* specifies how old data is discarded. The issues of long term memory will be discussed in Section 3.3.

³<http://archive.darpa.mil/grandchallenge05/>

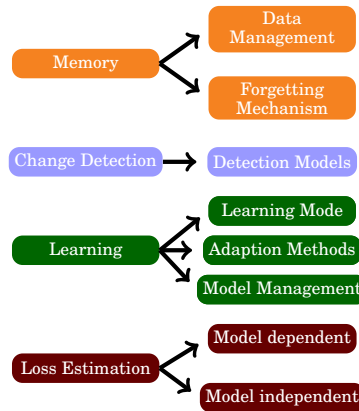


Fig. 4. Four modules of adaptive learning systems with a taxonomy of methods.

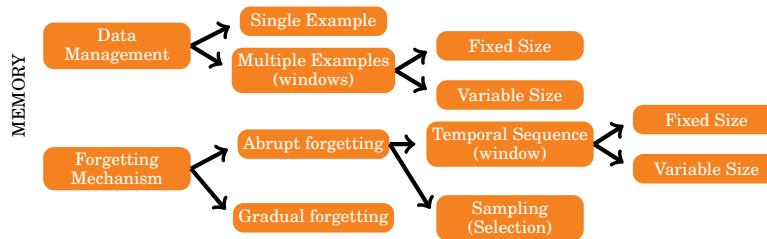


Fig. 5. Taxonomy of memory properties of methods.

3.1.1. Data Management. The assumption behind most of the adaptive learning algorithms is that the most recent data is the most informative for the current prediction. Hence, data management typically aims at learning from the most recent data: either a single example or multiple examples.

Single Example. Storing only a single example in memory has roots in online learning algorithms that learn from one example at a time. An online algorithm can process its input example by example in the order of arrival without keeping the entire training dataset in memory. Most online learners maintain a single hypothesis (which can originate from a complex predictive model) and model updates are error-driven. When a new example X_t is available, the current hypothesis makes a prediction \hat{y}_t . When the true target value y_t is received, loss is computed and the current hypothesis is updated if necessary. An example of such algorithms is *Winnow* [Littlestone 1987], which is a linear classifier system that uses a multiplicative weight-update scheme. The key characteristic of *Winnow* is its robustness to irrelevant features.

Online learning algorithms can be seen as naturally adaptive to evolving distributions, mainly due to the update mechanism that continuously updates the model with the most recent examples. However, online learning systems do not have explicit forgetting mechanisms. Adaptation happens only as the old concepts are diluted due to the new incoming data. Systems like *WINNOW* [Littlestone 1987], and *VFDT* [Domingos and Hulten 2000] can adapt to slow changes over time. The main limitation is their slow adaptation to abrupt changes, which depends on how sensible the model update with a new example is set to be. Setting these parameters requires a tradeoff between stability and sensitivity [Carpenter et al. 1991b]. Representative *single instance memory* systems explicitly dealing with concept drift include *STAGGER* [Schlim-

mer and Granger 1986], DWM [Kolter and Maloof 2003; 2007], SVM [Syed et al. 1999], IFCS [Bouchachia 2011a].

Multiple Examples. Another approach to data management is to maintain a predictive model consistent with a set of recent examples. The algorithm family FLORA [Widmer and Kubat 1996] is one of the first supervised incremental learning systems for evolving data. The original FLORA algorithm uses a sliding window of a fixed length, which stores the most recent examples in the *first-in-first-out (FIFO)* data structure. At each time step, the learning algorithm builds a new model using the examples from the training window. The model is updated following two processes: a learning process (update the model based on the new data) and a forgetting process (discard data that is moving out of the window). The key challenge is to select an appropriate window size. A short window reflects the current distribution more accurately, thus it can assure fast adaptation in periods with concept changes, but in stable periods too short window worsens the performance of the system. A large window gives a better performance in stable periods, but it reacts to concept changes slower.

In general the training window size can be *fixed* or *variable* over time.

Sliding windows of a fixed size. These methods store in memory a fixed number of the most recent examples. Whenever a new example arrives, it is saved to memory and the oldest one is discarded. This simple adaptive learning method is often used as a baseline in evaluation of algorithms.

Sliding windows of variable size. Here the number of examples in a window varies over time, typically depending on the indications of a change detector. A straightforward approach is to shrink the window whenever a change is singled such that the training data reflects the most recent concept, and grow the window otherwise.

One of the first algorithms using an adaptive window size is the FLORA2 [Widmer and Kubat 1996]. Incoming examples are added to the window and the oldest ones are deleted. Addition and deletion keeps the window (and the predictive model) consistent with the current concept. Further versions of the algorithm deal with recurring concepts (FLORA3) and noisy data (FLORA4). A later study [Klinkenberg and Joachims 2000] presents a theoretically supported method to recognize and handle concept changes with Support Vector Machines. This method maintains a window over the training examples with an appropriate size. The key idea is to adjust the window-size based on the estimate of the generalization error. At each time step the algorithm builds a number of SVM models using various window sizes and selects the the window size that minimizes the leave-one-out error estimate as the training window.

Learning windows of variable length appear in [Maloof and Michalski 1995; Klinkenberg 2004; Gama et al. 2004; Kuncheva and Zliobaite 2009]. The assumption behind relying on windowing is that the recency of the data is associated with relevance and importance. Unfortunately, this assumption may not be true in every circumstance. For instance, when the data is noisy or concepts reoccur, recency of data does not mean relevance. Moreover, if slow change lasts longer than the window size, windowing may fail as well.

3.1.2. Forgetting Mechanism. The choice of a forgetting mechanism depends on the expectations that we have regarding changes of data and what tradeoff between reactivity of the system and robustness to noise is required. The more abrupt is forgetting, the higher the reactivity, but also the higher risk of capturing noise is. The most common approach to deal with evolving data generated from processes with unknown dynamics is forgetting the outdated observations.

Abrupt Forgetting. At each time, a set of observations defines a window of the information considered for learning. Abrupt forgetting, or partial memory, refers to the mechanisms where a given observation either is inside the window or outside the window. Several window models have been presented in the literature. Two basic types of sliding windows are [Babcock et al. 2002]: *i) Sequence based.* The size of the window is defined in terms of the number of observations. Two different models are *sliding windows* of size j and *landmark windows*; *ii) Timestamp based.* The size of the window is defined in terms of *duration*. A sliding window stores only the most recent examples in the *first-in-first-out (FIFO)* data structure; while a landmark window stores all the examples after the landmark timestamp. A landmark window is a window of variable size. A timestamp window of size t consists of all elements whose timestamp is within a time interval t of the current time period. Systems that use abrupt forgetting were discussed in the previous section.

One of the alternatives to overcome windowing, especially fixed windowing, is sampling. The goal is to capture/summarize the underlying features of the data stream over long periods of time such that the selected sample is drawn uniformly from the stream. One of well known algorithms is the Reservoir Sampling [Vitter 1985]. The goal of reservoir sampling is to obtain a sound representative sample for the stream so far seen. It operates as follows. When the i th item arrives, it is added to the reservoir with probability $p = k/i$ (k is the size of the reservoir). Then a randomly chosen item is discarded from the reservoir to free space for the new item if added. The i th item is discarded with probability $1 - p$. A number of sampling techniques have been investigated like in [Al-Kateb et al. 2007; Efraimidis and Spirakis 2006; Aggarwal 2006; Rusu and Dobra 2009]. Reservoir sampling has been discussed in some studies related to drift, change detection and as an alternative to windowing in [Ng and Dash 2008; Yao et al. 2012; Zhao et al. 2011].

Gradual Forgetting. Gradual forgetting is a full memory approach, which means that no examples are dropped out of memory. However, examples in memory are associated with a weight that reflects their age. Example weighting is based on a simple intuition that the importance of an example in the training set should decrease with its age. Suppose that at time i , the stored sufficient statistics is S_{i-1} and we observe an example X_i . Assuming an aggregation function $G(X, S)$, the new sufficient statistics is computed as $S_i = G(X_i, \alpha S_{i-1})$, where $\alpha \in (0, 1)$ is the fading factor, this way the oldest information becomes the least important.

Linear decay techniques can be found in [Koychev 2000; 2002], and a technique for exponential decay is presented in [Klinkenberg 2004]. The latter technique weights examples according to their age using an exponential ageing function $w_\lambda(X) = \exp(-\lambda j)$, where the example X appeared j time steps ago. The parameter λ controls how fast the weights decrease. For larger values of λ less weight is assigned to the examples and less importance they have. If $\lambda = 0$, all the examples have the same weight.

Memory methods are summarized in Table I.

3.2. Change Detection

The change detection component refers to the techniques and mechanisms for explicit drift and change detection. It characterizes and quantifies concept drift by indicating change-points or small time-intervals of change [Basseville and Nikiforov 1993]. We consider the following dimensions as illustrated in Figure 6: *i)* methods based on sequential analysis; *ii)* methods based on control charts; *iii)* methods based on differences between two distributions; *iv)* heuristic methods.

Table I. Categorization of Memory Techniques.

Data Management		
Single Example		[Schlimmer and Granger 1986],[Littlestone 1987] [Domingos and Hulten 2000], [Kuncheva and Plumptre 2008],[Kelly et al. 1999] [Bouchachia 2011a],[Ikonomovska et al. 2011]
Multiple Examples	Fixed Size	[Salganicoff 1997],[Widmer and Kubat 1996], [Syed et al. 1999],[Hulten et al. 2001],[Lazarescu et al. 2004], [Bifet and Gavalda 2006; 2007],[Gomes et al. 2011]
	Variable Size	[Malooof and Michalski 1995],[Klinkenberg 2004], [Gama et al. 2004], [Zhang et al. 2008],[Kuncheva and Zliobaite 2009]
Forgetting Mechanisms		
Abrupt Forgetting	Temporal Sequences	[Salganicoff 1997],[Widmer and Kubat 1996], [Forman 2006],[Klinkenberg 2004],[Pechenizkiy et al. 2009]
	Sampling	[Ng and Dash 2008],[Yao et al. 2012],[Delany et al. 2005] [Zliobaite 2011a],[Zhao et al. 2011],[Salganicoff 1993]
Gradual Forgetting		[Koychev 2000; 2002],[Helmbold and Long 1994], [Klinkenberg 2004],[Koren 2010]

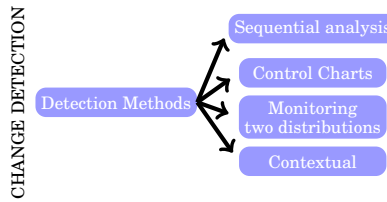


Fig. 6. Taxonomy of control properties of methods.

We already point-out that online learning systems, without any explicit change detection mechanism can adapt to evolving data. The advantage of explicit change detection is the information about the dynamics of the process generating data.

A typical control strategy monitors the evolution of the performance indicators [Widmer and Kubat 1996; Zeira et al. 2004] or raw data and statistically compares them to a fixed baseline. A seminal work in adaptive learning with monitoring performance indicators is the FLORA family of algorithms. The FLORA2 algorithm [Widmer and Kubat 1996] includes a window adjustment heuristic for a rule-based classifier. To detect concept changes, the accuracy and the coverage of the current model are monitored and the window size is adapted accordingly. In the context of information filtering monitoring the values of three performance indicators: *accuracy*, *recall* and *precision* has been proposed [Klinkenberg and Renz 1998]. The posterior of each indicator is compared to the standard sample errors of a moving average value.

3.2.1. Detectors based on Sequential Analysis. The *Sequential Probability Ratio Test* (SPRT) [Wald 1947] is the basis for several change detection algorithms. Let X_1^n be a sequence of examples, where the subset of examples X_1^w , $1 < w < n$ is generated from an unknown distribution P_0 , and the subset X_w^n is generated from another unknown distribution P_1 . When the underlying distribution changes from P_0 to P_1 at point w , the probability of observing subsequences under P_1 is expected to be *significantly* higher than that under P_0 : *Significant* means that the ratio of the two probabilities is no smaller than a threshold. Assuming that observations X_i are independent, the statistic for testing the hypothesis that a change point occurred at time w against the null hypothesis of no change at time w , is given by

$$T_w^n = \log \frac{P(x_w \dots x_n | P_1)}{P(x_w \dots x_n | P_0)} = \sum_{i=w}^n \log \frac{P_1[x_i]}{P_0[x_i]} = T_w^{n-1} + \log \frac{P_1[x_n]}{P_0[x_n]}, \quad (3)$$

and a change is detected if $T_w^n > L$, where L is a user defined threshold.

The *cumulative sum* (CUSUM) is a sequential analysis technique due to [Page 1954] that uses the SPRT rationale. It is often used for change detection. The test outputs an alarm when the mean of the incoming data significantly deviates from zero. The input for the test is a residual from any predictor, for instance, the prediction error from a Kalman filter. The CUSUM test is given by $g_t = \max(0, g_{t-1} + (x_t - \delta))$ ($g_0 = 0$), and the decision rule is if $g_t > \lambda$ then signal an alarm followed by setting $g_t = 0$. Here x_t stands for the current observed value, δ corresponds to the magnitude of changes that are allowed, t is the current time and λ is a user defined threshold. Since this expression only detects changes in the positive direction, when negative changes need to be found, the min operation should be used instead of max. In this case, a change is detected when the value of S is below the (negative) value of the threshold. The CUSUM test is memoryless, and its accuracy depends on the choice of parameters δ and λ . Both parameters control the trade-off between earlier detecting the true changes and allowing more false alarms. Low values of δ allow faster detection, at the cost of increasing the number of false alarms. CUSUM has been applied in stream mining, for example, by [Muthukrishnan et al. 2007].

The *Page-Hinkley* [Page 1954] test (PH) is a variant of CUSUM. It is a sequential analysis technique typically used for change detection in signal processing. It allows efficient detection of changes in the normal behaviour of a process established by a model. The PH test is a sequential adaptation of the detection of an abrupt change in the average of a Gaussian signal [Mouss et al. 2004]. The test variable m_T is defined as the cumulative difference between the observed values and their mean up until the current time: $m_T = \sum_{t=1}^T (x_t - \bar{x}_T - \delta)$, where $\bar{x}_T = \frac{1}{T} \sum_{t=1}^T x_t$ and δ specifies the tolerable magnitude of changes. The minimum m_T is defined as $M_T = \min(m_t, t = 1 \dots T)$. PH tests for the difference between M_T and m_T : $PH_T = m_T - M_T$. When this difference is greater than a threshold (λ) (user defined) a change is flagged. Larger λ will entail fewer false alarms, but might miss some changes.

Similar to the online versions of CUSUM test, and PH test presented here, Shiryaev & Roberts method, Shiryaev's Bayesian test [Shiryaev 2009; Tartakovsky and Moustakides 2010] rely on online thresholding, that is, as soon as the computed statistic exceeds a pre-set threshold ε , the change is diagnosed. The accuracy of such detection methods often relies on indicators like the false alarm rate and the mis-detection rate. Some of these methods are further explained below.

3.2.2. Detectors based on Statistical Process Control. Control charts, or Statistical Process Control (SPC), are standard statistical techniques to monitor and control the quality of a product during a continuous manufacturing. SPC considers learning as a process, and monitors the evolution of this process. Drift detection methods based on SPC appear in [Klinkenberg and Renz 1998; Lanquillon 2002; Gama et al. 2004; Gomes et al. 2011; Bouchachia 2011a].

Let pairs (X_i, y_i) form a sequence of examples. For each example, the model predicts \hat{y}_i , that can be either true ($\hat{y}_i = y_i$) or false ($\hat{y}_i \neq y_i$). For a set of examples, the error is a random variable from Bernoulli trials. The Binomial distribution gives a general form of the probability for the random variable that represents the number of errors in a set of n examples. For each point i in the sequence, the error-rate is the probability p_i of observing false with the standard deviation $\sigma_i = \sqrt{p_i(1-p_i)/i}$. The drift detector manages two registers during the model operation, p_{min} and σ_{min} . At time i after casting the prediction for the current example and verifying the prediction error, if $p_i + \sigma_i$ is lower than $p_{min} + \sigma_{min}$, then $p_{min} = p_i$ and $\sigma_{min} = \sigma_i$.

For a sufficiently large number of observations, the Binomial distribution is closely approximated by the Normal distribution with the same mean and variance. Consider-

ing that the probability distribution should not change unless a concept drift happens, the $1 - \delta/2$ confidence interval for p_i with $n > 30$ examples is approximately $p_i \pm \alpha \times \sigma_i$. The parameter α depends on the desired confidence level. A commonly used confidence level for Warning is 95% with the threshold $p_i + \sigma_i \geq p_{min} + 2 * \sigma_{min}$, and for Out-of-Control is 99% with the threshold $p_i + \sigma_i \geq p_{min} + 3 * \sigma_{min}$.

Suppose at time j an example (X_j, y_j) arrives and the model prediction leads to p_j and σ_j . The system is defined to be in one of the following three states:

- (1) In-Control while $p_j + \sigma_j < p_{min} + 2 \times \sigma_{min}$. The error of the system is stable. The example X_j is deemed to come from the same distribution as the previous examples.
- (2) Out-of-Control whenever $p_j + \sigma_j \geq p_{min} + 3 \times \sigma_{min}$. The error has increased significantly as compared to the recent past examples. With the probability $1 - \delta/2$ the recent examples come from a different distribution than the previous examples.
- (3) Warning state is in between of the two previous states. The error is increasing but has not reached Out-of-Control yet. This is a not a decisive state. The error may be increasing due to noise, drift, or due to a small deficiency of the predictive model. This state signals that more examples are required for confirming a drift.

SPC can be used to measure the *rate of change* as the time between Warning and Out-of-Control. Short times indicate fast changes, longer distances indicate slower changes. The *rate of change* can also be measured as the rate errors to the number of examples during Warning. SPC relies on the estimates of the error variance to define action bounds, which shrink as the confidence in the error estimates increases. SPC can be implemented inside incremental learning algorithms, or as a wrapper to batch algorithms. (Algorithm 3 in the Appendix).

The *exponentially weighted moving average* (EWMA) algorithm [Ross et al. 2012] advances on similar ideas. The EWMA computes a recent estimate of the error rate, μ_t , by progressively down-weighting older data: $Z_0 = \mu_0$ and $Z_t = (1 - \lambda)Z_{t-1} + \lambda e_t, t > 0$, where e_t is the error at the current example. It can be shown that, independently of the distribution of X_t , the mean and standard deviation of Z_t are: $\mu_{Z_t} = \mu_t$ and $\sigma_{Z_t} = \sqrt{\frac{\lambda}{2-\lambda}(1 - (1 - \lambda)^{2t})\sigma_E}$. Assume that before the change point that $\mu_t = \mu_0$. The EWMA estimator Z_t fluctuates around this value. When a change occurs, the value of μ_t changes to μ_c and Z_t will react to this by diverging from μ_0 towards μ_c . We can signal a change when $Z_t > \mu_0 + L\sigma_{Z_t}$. The parameter L , the control limit, determines how far from Z_t must diverge from μ_0 before a change is flagged.

3.2.3. Monitoring distributions on two different time-windows. These methods typically uses a fixed reference window that summarizes the past information, and a sliding detection window over the most recent examples. Distributions over the two windows are compared using statistical tests with the null hypothesis stating that the distributions are equal. If the null hypothesis is rejected, a change is declared at the start of the recent window. The windows can monitor univariate or multivariate raw data (separately for each class), evolving model parameters of performance indicators [Dries and Ruckert 2009]. The two windows can be of equal or progressive sizes and different window positioning strategies may be employed [Ade and Berthold 2013].

Comparing distributions on two detection windows⁴ in the context of data streams has been introduced in [Kifer et al. 2004]. Examples from two windows are compared with statistical tests based on the Chernoff bound to decide whether the two distributions are different. In the same line, the technique VFDTc [Gama et al. 2006] has

⁴Note that the detection windows are conceptually different from training windows. They may coincide in size. The detection windows are used for estimating data distribution and are typically paired. The training windows determine training data set for the learning algorithm when producing a model.

the ability to deal with concept drift, by continuously monitoring differences between two distributions classes of examples: the distribution when a node was a leaf and the weighted sum of the distributions in the children leaves of that node.

[Vorburger and Bernstein 2006] present an entropy-based metric to measure the distribution inequality between two sliding windows including respectively older and more recent examples. If the distributions are equal, the entropy measure results in a value of 1, and if they are absolutely different the measure will result in a value of 0. The entropy measure is continuously monitored over time, and a concept drift is signalled when the entropy measure decreases below a given fixed user-defined threshold. Additional examples include the change detection methods proposed by [Dasu et al. 2006] and [Sebastião and Gama 2007]. Both use the Kullback-Leibler (KL) divergence to measure the distance between the probability distributions of two different windows (old and recent) to detect possible changes.

An illustrative example is presented in [Bach and Maloof 2008]. It uses two models: *stable* and *reactive*. The stable model predicts based on a long history, whereas the reactive model predicts based on a short recent time window. The technique uses the reactive model as an indicator of concept drift, and it uses the stable model to make predictions, since the stable model performs better than the reactive model when acquiring a target concept. The drift detection method uses the differences in accuracy between the two models to determine when to replace the current stable model, since the stable model performs worse than the reactive model when the target concept changes. Also [Nishida and Yamauchi 2007], consider two accuracies. The accuracy estimated over all the stream and the accuracy estimated over a sliding window of the most recent examples. A concept drift is signalled, whenever a significant decrease of the recent accuracy is observed.

The ADaptive sliding WINDOW (ADWIN) [Bifet and Gavalda 2006; 2007] is a change detector and estimator using a *detection* window. ADWIN takes as inputs a (possibly infinite) sequence of real values $x_1, x_2, x_3, \dots, x_t, \dots$ and a confidence parameter $\delta \in (0, 1)$. Each x_t is generated according to some distribution D_t , independently for every t and the value of x_t is available only at time t . Denote as μ_t the expected value of x_t when it is drawn according to D_t . Assuming that always $x_t \in [0, 1]$ by an easy re-scaling the algorithm can handle any case in which we know an interval $[a, b]$ such that $a \leq x_t \leq b$ with probability 1. ADWIN slides a fixed detection window W on the most recently read x_i . Let $\hat{\mu}_W$ denote the (known) average of the examples within W , and μ_W the (unknown) average of μ_t for $t \in W$. We use $|W|$ to denote the length of a (sub)window W . ADWIN is summarized in Figure 2 in the Appendix, it operates as follows. Whenever two *large enough* (sub)windows of W exhibit *distinct enough* means, the algorithm concludes that the expected values within those windows are different, and the older (sub)window is dropped. *Large enough* and *distinct enough* are defined by the Hoeffding bound, testing whether the average of the two (sub)windows is larger than ϵ_{cut} computed as $\epsilon_{cut} := \sqrt{\frac{1}{2m} \cdot \ln \frac{4|W|}{\delta}}$, where m is the harmonic mean of $|W_0|$ and $|W_1|$, i.e. $m = \frac{2}{1/|W_0| + 1/|W_1|}$. ADWIN does not assume any particular data distribution. δ is a user defined confidence parameter, the suggested default value is 0.2. ADWIN is equipped with compression using a variant of the exponential histogram [Datar et al. 2002]. Therefore, it does not need to store all the examples from the detection window W , it stores this data in only $O(\log W)$ memory and $O(\log W)$ processing time per item, rather than the $O(W)$.

The adaptation techniques discussed next have originally been coupled with specific change detectors. In general, a strategy can deploy any detector, since the main information that is needed from a detector is whether a change has occurred or not.

Table II. Complexity of change detection algorithms, W is the size of the detection window.

Approach	Memory at t	Processing at t	Example algorithms
sequential analysis	$O(1)$	$O(1)$	CUSUM, PH
statistical process control	$O(1)$	$O(1)$	SPC
monitoring two distributions	$O(\log W) - O(W)$	$O(\log W) - O(W)$	ADWIN

3.2.4. Contextual approaches. Splice system [Harries et al. 1998] presents a meta-learning technique that implements a context sensitive batch learning approach. Splice is designed to identify intervals with stable hidden context and to induce and refine local concepts associated with these contexts. The idea is to use a time-stamp of the examples as an input feature for a batch classifier. In a first stage, examples are augmented with a time-stamp feature, and a decision tree inducer learns a decision tree. If the decision tree finds splits on the time-stamp feature, the partitions on that feature suggest different contexts. In the second phase, C4.5 is applied to each partition to find interim (temporal) concepts. In the next stage, all examples and interim concepts are given a score based on the accuracy of the concepts in given fixed window (over the time-stamp). This is followed by clustering examples with the highest score for the same concept. Another application of C4.5 then creates new interim concepts.

Independently of use of any time windowing, very often robust approaches that yield a balance between incremental learning and forgetting are needed to deal with changing environments. This idea was applied in the IFCS algorithm [Bouchachia 2011a] in the context of classification. There three mechanisms, called staleness, penalization and overall accuracy, which are associated with the individual prototypes that form the classifier, are combined. The first two measures intend to tackle the problem of model complexity, but also gradual drift. If one of the two measure values falls below a very small threshold, called removal threshold, the prototype is removed. The last one is intended for handling gradual and abrupt drift. Staleness tracks the activity of the prototype in making decisions about the new input and such an activity indicates that the recent incoming new input emanate from the input space covered by the prototype. Stale prototypes tend to cover obsolete regions. To express the staleness mechanism, the following formula is used: $w(i) = \zeta^{t-a_t}$ where t and a_t indicate, respectively, the current time index and the last time the prototype i was a winner. Small values of the forgetting factor ζ accelerates the reduction in the weight. The weights are associated with the prototypes. Their value for stale prototypes decays and for active prototypes it reinforces. Clearly, if the staleness is long (that is $t - a_t$ is large enough, w_i diminishes and then the prototype vanishes.

The second mechanism is about tracking the accuracy of the decisions made and thus observing the evolution of the model in terms of consistency with the recent input patterns. The aim is to ensure that the accuracy does not deteriorate (at least significantly). The following formula has been used: $z_i = \lambda^{s_i}$ where s_i is number of errors made by the prototype since it has been created. The weight decreases exponentially as the number of errors increases. The smaller the value of λ , the more quickly are the forgetting speed and the model update. The third mechanism is the Statistical Process Control (SPC) criterion which is explained in Section 3.2.2 and which aims at handling gradual and abrupt changes based on the number of errors produced by the learning model during prediction.

Table II summarizes properties of the presented change detection algorithms. Overall, comparing two distributions for change detection requires more computational resources than detecting based on the evolution of indicators, but potentially may give more precise information about the location of change.

Table III. Categorization of Change Detection Techniques.

	Detection Methods
Sequential analysis	[Wald 1947], [Page 1954], [Mouss et al. 2004] [Muthukrishnan et al. 2007], [Ikonomovska et al. 2011]
Control Charts	[Lanquillon 2002],[Gama et al. 2004] [Kuncheva 2009], [Gomes et al. 2011],[Ross et al. 2012]
Monitoring two distributions	[Kifer et al. 2004], [Bifet and Gavalda 2006; 2007] [Vorburger and Bernstein 2006], [Leeuwen and Siebes 2008] [Gama et al. 2006], [Nishida and Yamauchi 2007], [Bach and Maloof 2008], [Dries and Ruckert 2009], [Kuncheva 2013]
Contextual	[Harries et al. 1998],[Klinkenberg 2004]

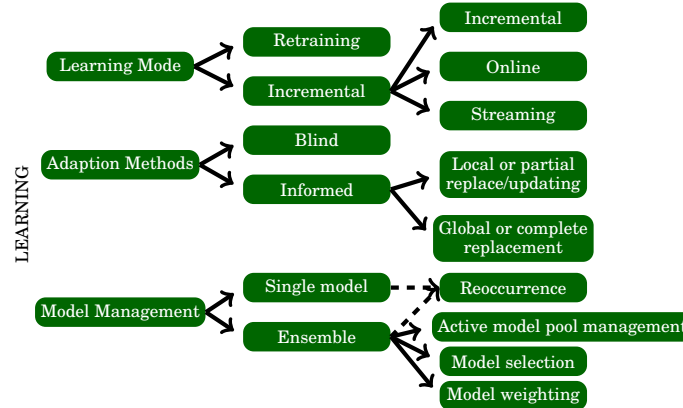


Fig. 7. Taxonomy of learning properties of methods.

3.3. Learning

The Learning component refers to the techniques and mechanisms for generalizing from examples and updating the predictive models from evolving data. This Section is organized as follows: *i) Learning mode* referring to model updating when new data points are available; *ii) Model adaptation* analyze the behaviour of predictive models in time-evolving data; *iii) Model management* refers to the techniques to maintain active predictive models.

3.3.1. Learning Mode. Whenever new labeled examples are available, the learning system might update the model. We consider two different learning modes: **Retraining** that discards the current model and builds a new model from scratch using buffered data; **Incremental** adaptation updates of the model.

Retraining approaches need some data buffer to be stored in memory. Retraining have been used to emulate incremental learning with batch learning algorithms [Gama et al. 2004]. At the beginning a model is trained with all the available data. Next, whenever new data arrives, the previous model is discarded, the new data is merged with the previous data, and a new model is learned on this data [Street and Kim 2001], [Zeira et al. 2004], [Klinkenberg and Joachims 2000].

Incremental approaches update the current model using the most recent data. **Incremental** algorithms process input examples one-by-one and update the sufficient statistics stored in the model after receiving each example. They might have access to previous examples or summaries of examples. This is the case of CVFDT [Hulten et al. 2001] described in the next section. The **Online** learning mode updates the current model with the most recent example. They are error-driven updating the current model depending on it misclassifies the current example. Examples include WINNOWER [Littlestone 1987], MBW [Carvalho and Cohen 2006]. MBW [Carvalho and Cohen 2006] refers

that well-known algorithms, including perceptron and *Multi-layer perceptrons*, that have been traditionally trained using several passes through the training data, when restricted to a single training pass over the data are particularly relevant for massive and streaming data [Carvalho and Cohen 2006]. The origins of online learning origin goes back to the late 60s, with Perceptron [Rosenblatt 1958], and early 2000 with the advent of the paradigm of "prediction with expert advice" [Cesa-Bianchi and Lugosi 2006]. The early work appeared in a number of seminal papers [Littlestone 1987; Littlestone and Warmuth 1994; Herbster and Warmuth 1998; Vovk 1998]. The model is updated with the current example. As time goes on, the newly arrived data tend to erase away the prior patterns. In models, such as artificial neural networks, learning is inevitably connected with forgetting. The ability to continuously learn from a stream of examples while preserving previously learned knowledge is known as the *stability-plasticity* dilemma [Carpenter et al. 1991a]. It is a dilemma because there needs to be a balance between being stable to handle noise and being able to learn new patterns. Some artificial neural networks completely forget the past patterns when exposed to a new set of patterns; this phenomenon is known as the *catastrophic forgetting* [French 1994; Polikar et al. 2001].

Streaming algorithms are online algorithms for processing high-speed continuous flow of data. Examples are processed sequentially and can be examined in only a few passes (typically just one). These algorithms use limited memory and control the available memory. For example Hoeffding trees [Domingos and Hulten 2000] and variants VFDTc [Gama et al. 2006] or FIMT-DD [Ikonovska et al. 2011], are able to freeze leaves when memory becomes scarce.

3.3.2. Adaptation Methods. The adaptation strategies manage adaptation of the predictive model. They fall into two major types: *blind* and *informed*.

The **blind** adaptation strategies adapt the model without any explicit detection of changes. **Blind** adaptation, typically use techniques as *fixed size sliding windows* that take a window size w as a parameter and periodically retrain the model with the latest w examples, and *example weighting* that consider the importance of training examples [Widmer and Kubat 1996; Klinkenberg and Renz 1998; Klinkenberg and Joachims 2000; Lanquillon 2002].

A special case of blind adaptation are incremental and online learning where the model evolves with data. Without any strategy to explicitly detect concept drift, the model adapts to the most recent data. A paradigmatic example is VFDT [Domingos and Hulten 2000]. In VFDT, new examples update statistics in the leaves of the current model. As the tree grows and new leaves are generated, the label in the these leaves reflect the most recent concepts. The blind strategies are proactive, they update the model based on the loss function. The main limitation of the blind approaches is slow reaction to concept drift in data. The blind approaches forget old concepts at a constant speed independently of whether changes are happening or not. When changes are happening it may be more beneficial to discard old data faster, and discard old data slower or not discard at all at times when changes are not happening.

The **informed** strategies are reactive, their actions depend on whether a trigger has been flagged [Bifet and Gavalda 2006; Hulten et al. 2001]. Triggers can be either change detectors (examples described in Section 3.2) or specific data descriptors that we will see in the *reoccurring concept management* techniques [Widmer and Kubat 1996]. Change detectors can be independent from the adaptation strategy (e.g. the *adaptive training window* technique) or they can be closely integrated with the adaptation strategy [Gama et al. 2006; Ikonovska et al. 2011] which we refer to as *model integrated detectors* [Gama et al. 2006].

The reaction to a drift signal might apply to the model as a whole, or might explore characteristics of the language used to represent generalization of examples.

Global Replacement. Informed adaptation in global models (such as linear regression, discriminant classifiers, naive Bayes) require full reconstruction of the model. This is the most radical reaction to a drift. The full model is deleted and a new model is started from scratch. This strategy has been widely used, for example in [Gama et al. 2004],[Street and Kim 2001], [Zeira et al. 2004], [Klinkenberg and Joachims 2000].

Local Replacement. In many cases changes occur only in some regions of the data space. For example, in spam filtering the spammers may start using combinations of words such as "Facebook support" that were previously associated with legitimate e-mails. This change in the concept of spam will affect only a small subset of all the incoming e-mails [Carmona-Cejudo et al. 2010].

Granular models, such as decision rules or decision trees, can adapt parts of the model. In a decision tree (or decision rules), each node (or rule) covers a hyper-rectangle in the data space. Thus, such decomposable models only need to adapt those nodes that cover the region of the data space affected by concept drift. CVFDT [Hulten et al. 2001] continuously monitors the quality of previous decisions (splitting features) with respect to a sliding window of fixed size over the most recent data stream. CVFDT maintains a window of training examples and keeps its learned tree up-to-date with this window by monitoring the quality of its old decisions as data moves into and out of the window. In particular, whenever a new example is read it is added to the statistics at all the nodes in the tree that it passes through, the last example in the window is forgotten from every node where it had previously had an effect, and the validity of all statistical tests are checked. If CVFDT detects a change it starts growing an alternate tree in parallel which is rooted at the newly-invalidated node. When the alternate is more accurate on new data than the original the original is replaced by the alternate and freed.

Each node in a Hoeffding tree captures statistics from a time window over the stream. The root node receives the oldest examples, the leaf nodes receive the most recent examples. Nodes near the root were generated using examples older than those that generated nodes near the leaves. This observation is on the basis of *Sequential regularization* [Gama et al. 2006; Ikononovska et al. 2011]. The technique compares the distribution of the errors at leaves to the distribution of the errors at upper nodes in the tree. Recall that the leaves contain the most recent information and the upper nodes contain older information. Thus, if the two error distributions are significantly different that is interpreted as a concept drift. In [Gama et al. 2006], when a concept change is detected, the system adapts the model, assuming that the most recent information contains the useful information about the current concept. The statistics of the most recent examples incorporated in the tree are stored in the leaves. Therefore, supposing that the change of concept was detected in the node i , the reaction method *pushes up* all the information of the descending leaves to the node i , namely the sufficient statistics and the classes distributions. The decision node becomes a leaf and the sub-tree rooted at the decision tree is pruned. This forgetting mechanism removes the outdated information.

3.3.3. Model Management. Ensemble learning maintains in memory an ensemble of multiple models that make a combined prediction. Adaptive ensembles are often motivated by the assumption that during a change, data is generated from a mixture distribution, which can be seen as a weighted combination of distributions characterizing the target concepts, each individual model models a distribution [Scholz and

Klinkenberg 2007]. The final prediction is typically a weighted average of the individual predictions, where the weight reflects the performance of the individual models on the most recent data. The weights change over time.

Ensemble methods for dynamically changing data can be categorized [Kuncheva 2004] into three types: (i) dynamic combination where the base learners are trained in advance and dynamically combined to respond to the changes in the environment by changing the combination rule (WINNOWER and weighted majority are often used [Littlestone 1987; Blum 1997; Tsymbal et al. 2006; Widmer and Kubat 1996]); (ii) continuous update of the learners such that the learners are either retrained in batch mode or updated online using the new data [Breiman 1999; Fern and Givan 2003; Oza 2001] (the combination rule may or may not change in the process); (iii) structural update where new learners are added (or existing ones are activated if they deactivated) and inefficient ones are removed (or deactivated) [Kolter and Maloof 2003; Street and Kim 2001; Bouchachia 2011b]. These three categories do not necessarily need to be mutually exclusive, an ensemble combining two or all three types of strategies is technically possible.

In connection to data drift and online learning, the application of ensemble learning has been the subject of some investigations over the recent past years. For instance, in [Elwell and Polikar 2011] a batch-based ensemble of classifiers, called Learn++.NSE, is proposed to deal with concept drift. The proposed algorithm aims at coping with drift regardless of the rate and type of drift and the number of classes present at any time. For each new batch of data, a new classifier is trained and combined using a dynamically weighted majority voting strategy.

In [Minku et al. 2010] diversity of learning ensemble is investigated in presence of different types of drifts. The study shows that before the drift occurs, ensembles with less diversity obtain lower test errors, but shortly after the drift occurs, highly diverse ensembles are better regardless of the type of drift. Longer after the drift, high diversity becomes less important.

The SEA algorithm [Street and Kim 2001] is one of the first techniques to handle concept drift with classifier ensembles learned from streaming data. It trains a separate classifier on each sequential batch of training examples. A trained classifier is added to a fixed-size ensemble, while the worst performing classifier is discarded. The final prediction is made using a simple majority voting.

A seminal work is the Dynamic Weighted Majority algorithm (DWM) [Kolter and Maloof 2003; 2007] that is an adaptive ensemble based on the Weighted Majority algorithm [Littlestone 1987]. It can be used with any on-line learning algorithm in time-changing problems with unknown dynamics. DWM maintains an ensemble of predictive models, each with an associated weight. Models are generated by the same learning algorithm on different batches of data. DWM dynamically creates and deletes experts in response to changes in performance. DWM makes predictions using a weighted-majority vote of these models, and the weights are dynamically changing. The weights of all the models that misclassified the current example are decreased by a multiplicative constant β . If the overall prediction is incorrect, a new expert is added to the ensemble with weight equal to 1. Finally, all the models are incrementally updated with the current example. To avoid creating an excessive number of models DWM prunes the ensemble by removing the poorly performing experts with a weight less than a threshold. A variant of DWM called AddExp [Kolter and Maloof 2005] is an extension for classification and regression that intelligently prunes some of the previously generated models.

A similar approach, but using a weight schema similar to boosting and explicit change detection appears in [Chu and Zaniolo 2004]. A boosting-like approach to train a classifier ensemble from evolving data streams is also proposed in [Scholz and

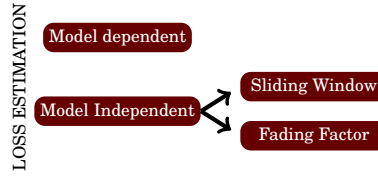


Fig. 8. Taxonomy of loss estimation properties of methods.

Klinkenberg 2007], where the dynamics of classifier weights is controlled by the *lift* measure instead of accuracy, which measures the correlation between predictions and their true label.

A general framework for mining concept-drifting data streams using weighted ensemble classifiers has been proposed in [Wang et al. 2003]. An ensemble of predictive models (e.g. C4.5, RIPPER, Naive Bayes) are trained on sequential batches of a data stream. Weighted voting is used to make the final prediction, the weights follow the expected predictive accuracy of each model. Suppose we have a training data set \mathcal{D} and a classifier \mathcal{L}_i . Let c be the true label of example X and $M_c(X)$ be the probability that X belongs to class c output by \mathcal{L}_i . The mean square error of the classifier is $MSE = \frac{1}{|\mathcal{D}|} \sum_{X \in \mathcal{D}} (1 - M_c(X))^2$. The weight of the classifier \mathcal{L} should be reversely proportional to its MSE, the weight reflects the benefit of using this individual model in comparison to a random classifier. The MSE of a random classifier is $MSE_r = \sum_c p(c) \times (1 - p(c))^2$, where $p(c)$ is the probability of observing class c . Thus, the weight w_i of a classifier \mathcal{L}_i is $w_i = MSE_r - MSE_i$.

A notable technique DDD [Minku and Yao 2011] equips a dynamic ensemble with a diversity control mechanism, and also uses an internal drift detection to speed up adaptation. Initially the model is composed of two ensembles: a low diversity ensemble and a high diversity ensemble. Both ensembles are trained with incoming examples, but only the low diversity ensemble is used for predicting. The weights of individual models are proportional to the prequential accuracy (described in Section 4). DDD assumes that, if there is no convergence of the underlying distributions to a stable concept, there is a drift. DDD then allows to use the high diversity ensemble for predictions. Online bagging (corresponding to sampling with replacement) is used to control diversity levels in the two ensembles.

Reoccurring concept management. FLORA3 [Widmer and Kubat 1996] is the first adaptive learning technique for the tasks where concepts may reoccur over time. More recent works discussing reoccurring concepts appear in [Widmer 1997; Yang et al. 2006; Katakis et al. 2010; Gama and Kosina 2011]. In such settings, instead of discarding outdated models, it might be useful to store the learned models in a sleeping mode. Following this idea, the work presented in [Gama and Kosina 2011], discuss a generic framework that identifies context using drift detection, characterizes contexts using meta-learning, and select the most appropriate predictive model for the incoming data using unlabeled examples. The proposed framework is based on a meta-learning schema which aims to recognize the area of applicability of the individual model.

3.4. Loss estimation

Supervised adaptive systems rely loss estimation based on environment feedback. The discussion in this section is close related with the performance related metrics discussed in Section 4.2.

Table IV. Categorization of Learning Techniques.

Learning Mode	
Retraining	[Street and Kim 2001], [Zeira et al. 2004], [Klinkenberg and Joachims 2000]
Incremental	[Schlimmer and Granger 1986],[Littlestone 1987], [Bifet et al. 2009],[Hulten et al. 2001],[Polikar et al. 2001]
Streaming	[Gama et al. 2006],[Ikononovska et al. 2011]
Adaptation Methods	
Blind	[Littlestone 1987], [Maloof and Michalski 2000], [Klinkenberg and Renz 1998] [Chu and Zaniolo 2004], [Bessa et al. 2009]
Informed	[Hulten et al. 2001], [Gama et al. 2006],[Ikononovska et al. 2011]
Model Adaptation	
Model Specific	[Hulten et al. 2001], [Gama et al. 2006], [Harries et al. 1998]
Model Independent	[Wald 1947], [Gama et al. 2004], [Wang et al. 2003] [Bifet and Gavalda 2006], [Kuncheva and Zliobaite 2009]
Model Management	
Single Model	[Hulten et al. 2001], [Gama et al. 2006], [Ikononovska et al. 2011]
	Recurrent [Widmer 1997], [Gama and Kosina 2011]
Ensemble	[Polikar et al. 2001], [Street and Kim 2001],[Kolter and Maloof 2005], [Gao et al. 2007], [Minku and Yao 2011], [Elwell and Polikar 2011]
	Recurrent [Yang et al. 2006], [Katakis et al. 2010], [Gomes et al. 2011]

Model Dependent. As we previously referred [Klinkenberg and Joachims 2000] recognize and handle concept changes using properties of Support Vector Machines. Their method maintains a window over the training examples that minimizes the leave-one-out error estimate. The key idea is to get an estimate of the generalization error using the so-called $\zeta\alpha$ -estimates, an efficient method for estimating the leave-one-out error of SVM. Leave-one-out estimators are accurate estimators but usually very expensive: one must run the learner so many times as the number of training examples. For SVM $\zeta\alpha$ -estimate can be computed for free once a SVM has been trained. It is computed as the percentage of training points that are between the margins. The theoretical properties of this estimator is discussed in [Joachims 2000]. Analytical loss estimation for the linear discriminant classifiers can be found in [Zliobaite and Kuncheva 2009].

Model Independent. As already referred, several works [Bach and Maloof 2008] [Nishida and Yamauchi 2007] [Gama et al. 2013], propose to detect changes using two sliding windows: a short window containing the most recent information and a large window, used as reference, containing a larger set of recent data including the data in the short window. The rationale behind this approach is that the short window is more reactive while the large window is more conservative. When a change occurs, statistics computed in the short window will capture the event faster than using the statistics in the larger window. Similarly, using fading factors, a smooth forgetting mechanism, a smaller fading factor will detect drifts earlier than larger ones. Based on this assumption, [Gama et al. 2013] propose to perform the PH test with the ratio between two error estimates: a long term error estimate (using a large window or a fading factor close to one) and a short term error estimate (using a short window or a fading factor smaller than the first one). A drift is signaled when the short term error estimator is significantly greater than the long term error estimator. The PH test monitors the evolution of ratio of both estimators and signals a drift when a significant increase of this variable is observed. The authors refer that the choice of α in fading factors and the window size is critical. Their experiments show that drift detection based on the ratio of fading estimates is somewhat faster than with sliding windows.

3.5. Discussion

Supervised adaptive learning algorithms rely on immediate arrival of feedback (true labels). In reality labels may become known immediately in the next time step after casting the prediction (e.g., food sales prediction). However, feedback can come with an

Table V. Categorization of Loss estimation Techniques.

Loss Estimation		
Model Dependent		[Klinkenberg and Joachims 2000], [Zliobaite and Kuncheva 2009]
Model Independent	Sliding Window	[Maloof and Michalski 1995], [Klinkenberg 2004], [Bach and Maloof 2008], [Nishida and Yamauchi 2007], [Gama et al. 2013]
	Fading Factors	[Koychev 2000] [Koychev 2002], [Gama et al. 2013]

uncontrollable delay, be unreliable, biased or costly. Labels may arrive within a fixed or variable time lag (in credit scoring typically the horizon of bankruptcy prediction is fixed, for instance, to one year, thus the true labels become known after one year has passed). Alternatively, the setting may allow to obtain labels on demand (e.g. in email spam categorization we can ask the user the true status of a given message).

4. EVALUATION

In order to perform an experimental evaluation of any machine learning technique, we need to consider first of all (i) performance evaluation metrics chosen according to the goal of a learning task and the operational settings, and (ii) a methodology allowing to compute the corresponding estimates in the streaming settings. Besides, we may need to find a justify whether one model (or technique) is superior to the other or not.

In this section we consider typical choices peculiar for evaluating an adaptive learning technique capable to handle concept drift. First, we discuss the performance evaluation metrics, then present possible experiment designs and conclude with pointers to the performance benchmarking.

4.1. Performance evaluation metrics

Performance evaluation metrics may be selected from the traditional accuracy measures, such as precision and recall or their weighed average in retrieval tasks, sensitivity and specificity or their weighed average in medical diagnostics, mean absolute scaled errors in regression or time-series prediction tasks, root mean square error in recommender systems.

It is important to consider appropriate reference points or baseline approaches in particular settings. For example, one common baseline in time series prediction is a moving average prediction, in its simplest form known as *tomorrow will look the same like today* prediction. Such a baseline gives reference points allowing to judge how much improvement a supposedly intelligent adaptive technique can achieve over a naive approach.

In addition, taking into account practical considerations of the streaming settings, we may consider the following measures:

A one-dimensional measure of the (computational) cost of the mining process. For example, in [Bifet et al. 2010b] the use of RAM-Hours was introduced as an evaluation measure of the resources used by streaming algorithms. Every GB of RAM deployed for 1 hour equals one RAM-Hour. It is based on rental cost options of cloud computing services.

A real-time unbalanced class measure of accuracy. The Kappa-Statistic may be a good measure for such cases. Besides taking into account class-imbalance it is also very efficient to compute as compared to e.g. the area under the ROC curve.

Besides evaluating the performance of the learning strategy as a whole we may like to estimate the accuracy of change detection for those strategies that employ explicit drift detection as part of the concept drift handling strategy. The following criteria are relevant for evaluating change detection methods.

Probability of true change detection. This measure characterizes the capacity of the learning system to detect and react to drifts when they occur.

Probability of false alarms. This measure characterizes the resilience to false alarms when there is no drift, that is detecting drift when there is no change in the target concept. Instead of reporting the commonly used false positive rate for change detections, in the streaming settings we may use the inverse of the *time to detection* or the *average run length*, the expected time between false positive detections as potentially more meaningful performance characteristic. This measure can be computed appropriately using data without drifts, then all the detections are counted as false alarms. Although, it could be possible to count false alarms in the case of gradual drifts, where more than one drift detection may appropriately occur, it is recommended to count false alarms only in the case that there is no drift in the data.

Delay in or time lag of detection. These measures give the estimate of how many new instances are required to detect a change after the actual occurrence of a change (or how much time would elapse before the change is detected).

Detection point accuracy. Besides the delay in detection we may want to know the accuracy of change localization. We can estimate it by counting the number of instances from the point that is detected as a change point to the true change point.

When we know how well the employed change detection methods perform we can also quantify the effect of particular error (or delay) in change detection on the overall performance of the adaptive model.

4.2. Experimental design

The most common procedure for estimating the performance of supervised learning techniques in the traditional settings with static data is cross-validation. In traditional batch learning the problem of limited data is overcome by analyzing and averaging multiple models produced with different random arrangements of training and test data. However, cross-validation is not directly applicable to the streaming settings with evolving data because it would mix the temporal order of data.

In the stream setting the problem of effectively unlimited data poses different challenges. One solution involves taking snapshots at different times during the induction of a model to see how much the model improves. The evaluation procedure of a learning algorithm determines which instances are used for training the algorithm, and which are used to test the model output by the algorithm. When considering what procedure to use in the data stream setting, one of the main concerns is how to build a picture of accuracy over time: one classifier may do well on the first half on the stream, and badly on the second. It is important to notice that streaming evaluation measures may be evolving over time.

4.2.1. Evaluation on time-ordered data. We discuss two common procedures peculiar to the evaluation of adaptive supervised learning techniques: *holdout* and *prequential* evaluation, and point to a recent idea of the controlled permutations.

Holdout. When traditional batch learning reaches a scale where cross-validation is too time consuming, it is often accepted to instead measure performance on a single holdout set. This is most useful when the division between train and test sets has been pre-defined, so that results from different studies can be directly compared. When testing a model at time t , the holdout set represents exactly the same context at that time t . The loss estimated in the holdout is an unbiased estimator. Unfortunately, it is not always possible to use holdout because it is not always possible to know for sure what examples belong to the concept that is active at time t .

Interleaved Test-Then-Train or Prequential. Each individual instance can be used to test the model before it is used for training, and from this the accuracy can be incrementally updated. When intentionally performed in this order, the model is always being tested on instances it has not seen before. This scheme has the advantage that no holdout set is needed for testing, making maximum use of the available data. It also ensures a smooth plot of accuracy over time, as each individual instance will become increasingly less significant to the overall average. The prequential error is computed based on an accumulated sum of a loss function between the prediction and observed values: $S = \sum_{t=1}^n f(\hat{y}_t, y_t)$.

There are three prequential evaluations: using a landmark window (*Interleaved Test-Then-Train*) a sliding window, or forgetting mechanism. The holdout evaluation gives a good estimation of the accuracy of the model on recent data. However, it requires recent test data that it is difficult to obtain for real datasets. In such a case a forgetting mechanism for estimating holdout accuracy [Gama et al. 2013] can be used, that is based on the prequential accuracy over a sliding window of size w with the most recent observations, or fading factors that weigh observations using a decay factor α . These mechanisms give an estimation of the accuracy that is approximate to the accuracy estimation obtained doing a holdout evaluation.

Controlled permutations. Averaging the accuracy over time has a potential problem, it may mask the adaptation properties of adaptive learning algorithms. For example, if one algorithm does very well on the first half of data, and very badly on the second; while another algorithm would show an average, but consistent performance, taking an average over accuracies would mask that. Even the prequential evaluation may produce biased results towards the fixed order of data in a sequence, as it runs only one test in a fixed order of data. To reduce this risk, *controlled permutations* evaluation [Zliobaite 2011b] runs multiple tests with randomized copies of a data stream, in a theoretical way restricted, so that different distributions from the original data are approximately preserved in permutations. Randomization aims at keeping the instances, that were originally nearby in time, close together. The technique suits best to data streams with sudden drifts, but is also applicable to streams with other drifts. Controlled permutations allow to generate multiple test sets for testing adaptive techniques that enable assessing volatility and robustness of models, optimize their parameters and this way reduce the risk of overfitting the order of data in a sequence.

4.2.2. Cross-validation with aligned series of data. In many cases we use an adaptive learning technique to learn not a single adaptive model for the data stream associated with an individual object (e.g. predicting antibiotic resistance within a particular hospital), but to learn multiple models, one adaptive model per object, e.g. we want to predict food sales for each of the product in the stock or popularity of each moving in a movie renting store. Each object corresponds to a data stream on which we apply an adaptive model. Each data stream is an analogue of a dataset (when we compare the performance of different classification techniques over multiple datasets), but in this settings, different data streams would be still alike, i.e. sharing the same feature space or being of the same nature. In such cases it is logical to estimate the performance of the adaptive learning strategy across multiple objects.

This method is rather generic for many real problems when a need for a more exact evaluation setting justifies the use of large computational resources, for example e.g. sales, recommendations, advertising, sentiment classification across topics or sources. Figure 9 illustrates the estimation of the generalization performance of the technique(s) across multiple individual (but related) data streams by cross validating the data streams. For every individual data stream we can use one of the described (prequential, hold-out or controlled permutations) evaluation procedures. A subset of

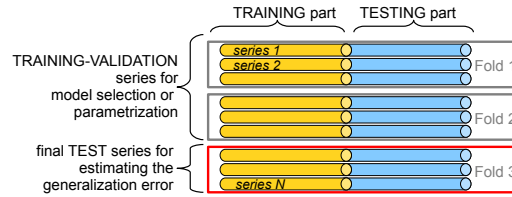


Fig. 9. Estimating the generalization performance of the technique(s) across multiple individual (but related) data streams by cross validating the data streams.

data streams can be used to run the the learning strategy with different parameterizations to identify to most reasonable parameter settings for the particular domain and then verify them on the remaining testing data streams. The values of the performance metrics can be averaged over data stream in the test basket (and averaged over the multiple cross-validation runs).

4.2.3. Statistical significance. Evaluating classifiers, we should be concerned with the statistical significance of the results at particular points of the learning process. Looking at the performance of only one classifier, it is convenient to give some insights about its statistical significance. We may use *confidence intervals*, that are interval estimates of parameters to indicate the reliability of an estimate. To do that, we can use Chernoff or Hoeffding bounds, sharper inequalities than Markov's or Chebyshev bounds.

When comparing two classifiers we need to distinguish between random and non-random differences in the experimental accuracies. The McNemar test [McNemar 1947] is a non-parametric test used in the stream mining literature to assess the statistical significance of differences in performance of two classifiers. This test needs to store and update two variables: the number of instances misclassified by the first classifier and not by the second one a , and b the number of instances misclassified by the second classifier and not by the first one. The McNemar statistic (M) is given as $M = \text{sign}(a - b) \times (a - b)^2 / (a + b)$. The test follows the χ^2 distribution. At 0.99 confidence it rejects the null hypothesis (the performances are equal) if $M > 6.635$.

When comparing more than two classifiers, the Nemenyi test [Demsar 2006] is used for computing significance: it is an appropriate test for comparing all classifiers to all classifiers over multiple datasets, being based on the average ranks of the algorithms across all datasets. The Nemenyi is recommended for use after a rejection of the null hypothesis by Friedman. The Nemenyi test is the following: two classifiers are performing differently if the corresponding average ranks differ by at least the critical difference $CD = q_\alpha \sqrt{k(k+1)/6N}$ where k is the number of learners, N is the number of datasets, and critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$. Other tests can be recommended when all classifiers are compared to a control classifier [Demsar 2006]. For example, the Dunnett test can be used [Dunnett 1955].

4.3. Performance benchmarking

To perform a benchmarking comparing several methods, we need to use some large datasets and software implementations of the algorithms. There are two types of dataset: artificial or real datasets. Artificial datasets are useful because they give the ground truth of the data, for example, all the points when the changes occur. However, real datasets are more interesting as they correspond to real-world applications where the algorithms usability is tested. Some authors use the term *real world dataset* to refer to data sets using real world data with forced drifts, that cannot be considered as completely real.

Appendix B presents popular artificial and real datasets that are publicly available. MOA is an open-source software to run data streaming experiments. It is written in Java and it is based on the experience of WEKA. It contains methods for classification, regression, clustering and frequent pattern mining. Online appendix C shows an example of evaluating classification under concept drift using MOA.

5. CONCLUSIONS

We presented the conceptual categorization of many existing adaptive learning strategies, capable of handling concept drift, along with the concrete state-of-the-art techniques. We highlighted the peculiarities of the evaluation methodology for experimenting with adaptive learning techniques in the data stream settings. The problem of concept drift has been recognized in different application domains. Further interest to adaptive learning has been boosted by the outcomes of several recently organized contests or challenges in AI (controlling driverless cars at the DARPA challenge), data mining and knowledge discovery (credit risk assessment competition at PAKDD'09), and recommender systems (Netflix movie recommendation) fields. Winning teams in each of these competitions emphasize that one of key factors in their success was due to addressing temporal dynamics and various (hidden) contexts affecting their concepts of interest. Therefore, we also referred to the conceptual categorization of typical application settings in which adaptive learning systems have to operate and discussed the lessons learnt from the real world cases of handling concept drift. We hope that besides serving as an introduction into the research area of adaptive learning under concept drift this article will help to position a new adaptive learning technique and application settings to which they apply.

Most of the work on concept drift assumes that the changes happen in hidden context that is not observable to the adaptive learning system. Hence, concept drift is considered to be unpredictable and its detection and handling is mostly reactive. However, there are various application settings in which concept drift is expected to reappear along the time line and across different objects in the modeled domain. Seasonal effects with vague periodicity for a certain subgroup of object would be common e.g. in food demand prediction [Zliobaite et al. 2012a]. Availability of external contextual information or extraction of hidden contexts from the predictive features may help to better handle recurrent concept drift, e.g. with use of a meta-learning approach [Gama and Kosina 2011].

Temporal relationships mining can be used to identify related drifts, e.g. in the distributed or peer-to-peer settings in which concept drift in one peer may precede another drift in related peer(s) [Ang et al. 2012]. In all these settings more accurate, more proactive and more transparent change detection may become possible.

The vast majority of the work on concept drift detection summarized in this survey does not address the problem of representation bias that is common to most of the adaptive systems that enforce or suggest particular type of behavior. Whenever there is a reinforcement feedback or a closed-loop control of the learning mechanism, we cannot evaluate and compare the performance of concept drift handling techniques by replaying historical data. Therefore, we can speculate that there will be more studies that try to embed concept drift handling technique in real operational settings for proper validation. While the majority of work on handling concept drift has focused on supervised settings with immediate availability of labels, the actual problem space is much wider. In unsupervised learning over evolving data, and in case of delayed and on-demand labeling in supervised learning, validation of change detection and adaptation mechanisms only start to be investigated.

Research on concept drift goes beyond the areas of machine learning, data mining, and pattern recognition in which the term was originally coined and studied most.

Thus, in Process Mining⁵ [van der Aalst 2012; 2011], the area of research dealing with the different kinds of analyses of (business) processes by extracting information from event logs recorded by an information system, handling concept drift has been recognized as an important problem [Carmona and Gavaldà ; Bose et al. 2013].

5.1. Next challenges

The next challenges for concept drift research include: improving scalability, robustness and reliability, moving from so called *black box* adaptation to more interpretable and explainable adaptation, reducing the dependence on timely and accurate feedback (true labels) and moving from adaptive algorithms towards adaptive systems that would automate full knowledge discovery process in addition to automating adaptation of the decision models. Some of these challenges have been discussed in [Zliobaite et al. 2012b].

Studying how to integrate expert knowledge in concept drift handling and how to interact with domain experts brings new challenges as well. Relying on non interpretable black-box models is not popular among the domain experts. They may need to trust that e.g. a control system is really going to react to changes when they happen and to understand how these changes are detected and what adaptation would happen. Furthermore, experts may have valuable knowledge how to improve the concept drift handling mechanism or to validate the system. The continuous or incremental learning nature of adaptive systems makes it challenging to come up with a strategy to incorporate expert knowledge into the adaptive learning process and to communicate with experts as the process evolves.

Studying how to perform localization and explanation of changes, e.g. by means of visualization or a rule-based approach would be helpful in improving usability and trust in adaptive learning systems.

REFERENCES

- ADAE, I. AND BERTHOLD, M. 2013. Eve: a framework for event detection. *Evolving systems* 4, 61–70.
- ADOMAVICIUS, G. AND TUZHILIN, A. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* 17, 6, 734–749.
- AGGARWAL, C. 2005. On change diagnosis in evolving data streams. *IEEE Trans. on Knowl. and Data Eng.* 17, 587–600.
- AGGARWAL, C. 2006. On biased reservoir sampling in the presence of stream evolution. In *Proc. of the 32nd int. conf. on Very large data bases*. VLDB. 607–618.
- AGRAWAL, R., GHOSH, S. P., IMIELINSKI, T., IYER, B. R., AND SWAMI, A. N. 1992. An interval classifier for database mining applications. In *Proc. of the 18th int. conf. on Very Large Data Bases*. VLDB. 560–573.
- AGRAWAL, R., IMIELINSKI, T., AND SWAMI, A. 1993. Database mining: A performance perspective. *IEEE Trans. on Knowl. and Data Eng.* 5, 6, 914–925.
- AL-KATEB, M., BYUNG SUK, L., AND WANG, X. 2007. Adaptive-size reservoir sampling over data streams. In *Proc. of Int. Conf. on Scientific and Statistical Database Management*. SSBDM. 22.
- ALBERG, D., LAST, M., AND KANDEL, A. 2012. Knowledge discovery in data streams with regression tree methods. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2, 69 – 78.
- ANG, H., GOPALKRISHNAN, V., M., I. Z., PECHENIZKIY, AND HOI, S. 2012. Predictive handling of asynchronous concept drifts in distributed environments. *IEEE Trans. on Knowl. and Data Eng.*
- BABCOCK, B., BABU, S., DATAR, M., MOTWANI, R., AND WIDOM, J. 2002. Models and issues in data stream systems. In *Proc. of the 21st SIGMOD-SIGACT-SIGART symp. on Princi. of database syst.* PODS. 1–16.
- BACH, S. H. AND MALOOF, M. A. 2008. Paired learners for concept drift. In *Proc. of the 8th IEEE Int. Conf. on Data Mining*. ICDM. 23–32.
- BACHE, K. AND LICHMAN, M. 2013. UCI machine learning repository. Tech. rep.
- BASSEVILLE, M. AND NIKIFOROV, I. 1993. *Detection of Abrupt Changes - Theory and Application*. online.

⁵<http://www.processmining.org>

- BESSA, R. J., MIRANDA, V., AND GAMA, J. 2009. Entropy and correntropy against minimum square error in off-line and on-line 3-day ahead wind power forecasting. *IEEE Trans. on Power Syst.* 24, 4, 1657–1666.
- BIFET, A. AND FRANK, E. 2010. Sentiment knowledge discovery in twitter streaming data. In *Proc. of the 13th int. conf. on Discovery science*. DS. 1–15.
- BIFET, A. AND GAVALDA, R. 2006. Kalman filters and adaptive windows for learning in data streams. In *Proc. of the 9th int. conf. on Discovery science*. DS. 29–40.
- BIFET, A. AND GAVALDA, R. 2007. Learning from time-changing data with adaptive windowing. In *Proc. of SIAM int. conf. on Data Mining*. SDM. 443–448.
- BIFET, A., HOLMES, G., KIRKBY, R., AND PFAHRINGER, B. 2011a. DATA STREAM MINING: A practical approach. Technical report, University of Waikato.
- BIFET, A., HOLMES, G., AND PFAHRINGER, B. 2010a. Leveraging bagging for evolving data streams. In *Proc. of the Eur. conf. on Mach. learn. and knowledge discovery in databases*. ECMLPKDD. 135–150.
- BIFET, A., HOLMES, G., PFAHRINGER, B., AND FRANK, E. 2010b. Fast perceptron decision tree learning from evolving data streams. In *Proc. of the 14th PA Conf. on Knowl. Discov. and Data Mining*. 299–310.
- BIFET, A., HOLMES, G., PFAHRINGER, B., KIRKBY, R., AND GAVALDA, R. 2009. New ensemble methods for evolving data streams. In *Proc. of the int. conf. on Knowl. discov. and data mining*. 139–148.
- BIFET, A., HOLMES, G., PFAHRINGER, B., READ, J., KRANEN, P., KREMER, H., JANSEN, T., AND SEIDL, T. 2011b. MOA: A real-time analytics open source framework. In *Proc. Eur. conf. on Mach. Learn. and Principles and Practice of Knowledge Discovery in Databases*. ECMLPKDD. 617–620.
- BILLSUS, D. AND PAZZANI, M. J. 2000. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction* 10, 147–180.
- BLUM, A. 1997. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Mach. Learn.* 26, 1, 5–23.
- BOBADILLA, J., ORTEGA, F., HERNANDO, A., AND GUTIÉRREZ, A. 2013. Recommender systems survey. *Know.-Based Syst.* 46, 109–132.
- BOSE, R. P. J. C., VAN DER AALST, W. M. P., ZLIOBAITE, I., AND PECHENIZKIY, M. 2013. Dealing with concept drift in process mining. (accepted to) *IEEE Trans. on Neur. Net. and Lear. Syst.*.
- BOUCHACHIA, A. 2011a. Fuzzy classification in dynamic environments. *Soft Comput.* 15, 5, 1009–1022.
- BOUCHACHIA, A. 2011b. Incremental learning with multi-level adaptation. *Neurocomp.* 74, 11, 1785–1799.
- BOUCHACHIA, A., PROSSEGGER, M., AND DUMAN, H. 2010. Semi-supervised incremental learning. In *Proc. of the IEEE Int. Conf. on Fuzzy Systems*. FUZZ-IEEE. 1–6.
- BREIMAN, L. 1999. Pasting small votes for classification in large databases and on-line. *Machine Learning* 36, 85–103.
- BREIMAN, L. ET AL. 1984. *Classification and Regression Trees*. Chapman & Hall, New York.
- BREMNES, J. 2004. Probabilistic wind power forecasts using local quantile regression. *Wind En.* 7, 1, 47–54.
- CARMONA, J. AND GAVALDÀ, R. Online techniques for dealing with concept drift in process mining. In *Advances in Intelligent Data Analysis XI - 11th Int. Symp.*
- CARMONA-CEJUDO, J., BAENA-GARCIA, M., DEL CAMPO-AVILA, J., BUENO, R., AND BIFET, A. 2010. GNUsmail: Open framework for on-line email classification. In *Proc. of the 19th Eur. conf. on Art. Intell.* ECAI. 1141–1142.
- CARPENTER, G., GROSSBERG, S., AND REYNOLDS, J. 1991a. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks* 4, 565–588.
- CARPENTER, G., GROSSBERG, S., AND ROSEN, D. 1991b. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks* 4, 6, 759–771.
- CARVALHO, V. AND COHEN, W. 2006. Single-pass online learning: performance, voting schemes and online feature selection. In *Proc. of the 12th ACM SIGKDD int. conf. on Kn. disc. and data min.* KDD. 548–553.
- CASTILLO, G., GAMA, J., AND BREDÀ, A. 2003. Adaptive bayes for a student modeling prediction task based on learning styles. In *Proc. of the 9th int. conf. on User modeling*. UM. 328–332.
- CESA-BIANCHI, N. AND LUGOSI, G. 2006. *Prediction, learning, and games*. Cambridge University Press.
- CHANDOLA, V., BANERJEE, A., AND KUMAR, V. 2009. Anomaly detection: A survey. *ACM Computer Surveys* 41, 15:1–15:58.
- CHU, F. AND ZANIOLO, C. 2004. Fast and light boosting for adaptive mining of data streams. In *Proc. of the 5th Pac.-Asia conf. on Knowledge Discovery and Data Mining*. PAKDD. 282–292.
- DASU, T., KRISHNAN, S., VENKATASUBRAMANIAN, S., AND YI, K. 2006. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Proc. of the 38th Symp. on the Interface of Statistics, Computing Science, and Applications*.

- DATAR, M., GIONIS, A., INDYK, P., AND MOTWANI, R. 2002. Maintaining stream statistics over sliding windows. *SIAM J. Comput.* 31, 1794–1813.
- DELANY, S., CUNNINGHAM, P., TSYMBAL, A., AND COYLE, L. 2005. A case-based technique for tracking concept drift in spam filtering. *Knowledge-Based Systems* 18, 4-5, 187–195.
- DEMSAR, J. 2006. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* 7, 1–30.
- DOMINGOS, P. AND HULTEN, G. 2000. Mining high-speed data streams. In *Proc. of the 6th ACM SIGKDD int. conf. on Knowledge discovery and data mining*. KDD. 71–80.
- DRIES, A. AND RUCKERT, U. 2009. Adaptive concept drift detection. *Stat. Anal. Data Min.* 2, 5-6, 311–327.
- DUDA, R., HART, P., AND STORK, D. 2001. *Pattern Classification*. Wiley.
- DUNNETT, C. W. 1955. A multiple comparison procedure for comparing several treatments with a control. *Journal of the American Statistical Association*, 1096–1121.
- EFRAIMIDIS, P. AND SPIRAKIS, P. 2006. Weighted random sampling with a reservoir. *Inf. Proc. Lett.* 97, 5, 181–185.
- ELWELL, R. AND POLIKAR, R. 2011. Incremental learning of concept drift in nonstationary environments. *IEEE Trans. on Neural Networks* 22, 10, 1517–1531.
- FERN, A. AND GIVAN, R. 2003. Online ensemble learning: An empirical study. *Mach. Learn.* 53, 1-2, 71–109.
- FORMAN, G. 2006. Tackling concept drift by temporal inductive transfer. In *Proc. of the 29th int. ACM SIGIR conf. on Research and development in inf. retrieval*. SIGIR. 252–259.
- FRENCH, R. M. 1994. Catastrophic forgetting in connectionist networks: Causes, consequences and solutions. In *Trends in Cognitive Sciences*. 128–135.
- GABER, M. M., ZASLAVSKY, A., AND KRISHNASWAMY, S. 2005. Mining data streams: a review. *SIGMOD Rec.* 34, 18–26.
- GAMA, J. 2010. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC.
- GAMA, J., FERNANDES, R., AND ROCHA, R. 2006. Decision trees for mining data streams. *Intelligent Data Analysis* 10, 1, 23–46.
- GAMA, J. AND KOSINA, P. 2011. Learning about the learning process. In *Proc. of the 10th int. conf. on Advances in intelligent data analysis*. IDA. Springer, 162–172.
- GAMA, J., MEDAS, P., CASTILLO, G., AND RODRIGUES, P. 2004. Learning with drift detection. In *Proc. of the 17th Brazilian symp. on Artif. Intell.* SBIA. 286–295.
- GAMA, J., SEBASTIÃO, R., AND RODRIGUES, P. 2013. On evaluating stream learning algorithms. *Machine Learning* 90, 3, 317–346.
- GANTZ, J. AND REINSEL, D. 2012. IDC: The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *International Data Corporation, sponsored by EMC Corporation*.
- GAO, J., FAN, W., HAN, J., AND YU, P. S. 2007. A general framework for mining concept-drifting data streams with skewed distributions. In *Proc. of the 7th SIAM Int. Conf. on Data Mining*. SDM.
- GEHRKE, J., RAMAKRISHNAN, R., AND GANTI, V. 1998. RainForest - a framework for fast decision tree construction of large datasets. In *Proc. of the 24rd Int. Conf. on Very Large Data Bases*. VLDB. 416–427.
- GIRAUD-CARRIER, C. 2000. A note on the utility of incremental learning. *AI Commun.* 13, 4, 215–223.
- GOMES, J. B., RUIZ, E. M., AND SOUSA, P. A. C. 2011. Learning recurring concepts from data streams with a context-aware ensemble. In *Proc. of the ACM symp. on Appl. Comp.* SAC. 994–999.
- GRISOGONO, A.-M. 2006. The implications of complex adaptive systems theory for C2. In *State of the Art State of the Practice*. Vol. CCRTS.
- HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. 2009. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.* 11, 1, 10–18.
- HARRIES, M. 1999. SPLICE-2 comparative evaluation: Electricity pricing. Tech. rep., South Wales Univ.
- HARRIES, M., SAMMUT, C., AND HORN, K. 1998. Extracting hidden context. *Machine Learning* 32, 101–126.
- HELMBOLD, D. P. AND LONG, P. M. 1994. Tracking drifting concepts by minimizing disagreements. *Mach. Learn.* 14, 1, 27–45.
- HERBSTER, M. AND WARMUTH, M. 1998. Tracking the best expert. *Machine Learning* 32, 2, 151–178.
- HULTEN, G., SPENCER, L., AND DOMINGOS, P. 2001. Mining time-changing data streams. In *Proc. of the 7th ACM SIGKDD int. conf. on Knowledge Discovery and Data Mining*. KDD. 97–106.
- IKONOMOVSKA, E., GAMA, J., AND DZEROSKI, S. 2011. Learning model trees from evolving data streams. *Data Mining Knowledge Discovery* 23, 1, 128–168.
- JOACHIMS, T. 2000. Estimating the generalization performance of an svm efficiently. In *Proc. of the 17th Int. Conf. on Mach. Learn.* ICML. 431–438.

- KADLEC, P., GRBIC, R., AND GABRYS, B. 2011. Review of adaptation mechanisms for data-driven soft sensors. *Computers & Chemical Engineering* 35, 1, 1–24.
- KATAKIS, I., TSOUMAKAS, G., AND VLAHAVAS, I. 2010. Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems* 22, 3, 371–391.
- KELLY, M. G., HAND, D. J., AND ADAMS, N. M. 1999. The impact of changing populations on classifier performance. In *Proc. of the 5th ACM SIGKDD int. conf. on Knowl. disc. and dat. min.* KDD. 367–371.
- KIFER, D., BEN-DAVID, S., AND GEHRKE, J. 2004. Detecting change in data streams. In *Proc. of the 13th int. conf. on Very large data bases.* VLDB. 180–191.
- KLINKENBERG, R. 2003. Predicting phases in business cycles under concept drift. In *Proc. of the Ann. Worksh. on Machine Learning of the National German Computer Science Society.* LLWA. 3–10.
- KLINKENBERG, R. 2004. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis* 8, 3, 281–300.
- KLINKENBERG, R. AND JOACHIMS, T. 2000. Detecting concept drift with support vector machines. In *Proc. of the 17th int. conf. on Machine Learning.* ICML. 487–494.
- KLINKENBERG, R. AND RENZ, I. 1998. Adaptive information filtering: Learning in the presence of concept drifts. In *Workshop Notes of the ICML/AAAI-98 Workshop on Learning for Text Categorization.* 33–40.
- KOLTER, J. AND MALOOF, M. 2003. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Proc. of the 3rd IEEE Int. Conf. on Data Mining.* ICDM. 123–130.
- KOLTER, J. AND MALOOF, M. 2005. Using additive expert ensembles to cope with concept drift. In *Proc. of the 22th int. conf. on Machine Learning.* ICML. 449–456.
- KOLTER, J. AND MALOOF, M. 2007. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research* 8, 2755–2790.
- KOREN, Y. 2010. Collaborative filtering with temporal dynamics. *Commun. ACM* 53, 4, 89–97.
- KOSINA, P., GAMA, J., AND SEBASTIAO, R. 2010. Drift severity metric. In *Proc. of the 19th Eur. Conf. on Artificial Intelligence.* ECAI. 1119–1120.
- KOYCHEV, I. 2000. Gradual forgetting for adaptation to concept drift. In *Proc. of ECAI Workshop on Current Issues in Spatio-Temporal Reasoning.* 101–106.
- KOYCHEV, I. 2002. Tracking changing user interests through prior-learning of context. In *Proc. of the 2nd int. conf. on Adaptive Hypermedia and Adaptive Web-Based Systems.* 223–232.
- KUKAR, M. 2003. Drifting concepts as hidden factors in clinical studies. In *Proc. of AIME 2003, 9th Conference on Artificial Intelligence in Medicine in Europe.* 355–364.
- KUNCHEVA, L. 2008. Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. In *Proc. of the 2nd Workshop SUEMA 2008.*
- KUNCHEVA, L. AND ZLIOBAITE, I. 2009. On the window size for classification in changing environments. *Intelligent Data Analysis* 13, 6, 861–872.
- KUNCHEVA, L. I. 2004. Classifier ensembles for changing environments. In *Proc. of the 5th Int. Worksh. on Multiple Classifier Systems.* MCS. 1–15.
- KUNCHEVA, L. I. 2009. Using control charts for detecting concept change in streaming data. Tech. Rep. BCS-TR-001-2009, School of Computer Science, Bangor University, UK.
- KUNCHEVA, L. I. 2013. Change detection in streaming multivariate data using likelihood detectors. *IEEE Transactions on Knowledge and Data Engineering* 25.
- KUNCHEVA, L. I. AND PLUMPTON, C. O. 2008. Adaptive learning rate for online linear discriminant classifiers. In *Proc. of Int. Worksh. on Structural and Syntactic Pattern Recognition.* SSPR. 510–519.
- LANQUILLON, C. 2002. Enhancing text classification to improve information filtering. *Künstliche Intelligenz*, 16, 2, 37–38.
- LAZARESCU, M. M., VENKATESH, S., AND BUI, H. H. 2004. Using multiple windows to track concept drift. *Intelligent Data Analysis* 8, 1, 29–59.
- LEEUWEN, M. AND SIEBES, A. 2008. Streamkrimp: Detecting change in data streams. In *Proc. of the Eur. Conf. on Mach. Learn. and Knowledge Discovery in Databases.* ECMLPKDD. 672–687.
- LINDSTROM, P., DELANY, S. J., AND NAMEE, B. M. 2010. Handling concept drift in a text data stream constrained by high labelling cost. In *Proc. of the 23rd Int. Florida Art. Intell. Research Society Conf.*
- LITTLESTONE, N. 1987. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning* 2, 4, 285–318.
- LITTLESTONE, N. AND WARMUTH, M. 1994. The weighted majority algorithm. *Inf. Comput.* 108, 2, 212–261.
- MALOOF, M. AND MICHALSKI, R. 2000. Selecting examples for partial memory learning. *Machine Learning* 41, 27–52.

- MALOOF, M. AND MICHALSKI, R. 2004. Incremental learning with partial instance memory. *Artificial Intelligence* 154, 95–126.
- MALOOF, M. A. 2010. The AQ methods for concept drift. In *Advances in Machine Learning I: Dedicated to the Memory of Professor Ryszard S. Michalski*. 23–47.
- MALOOF, M. A. AND MICHALSKI, R. S. 1995. A method for partial-memory incremental learning and its application to computer intrusion detection. In *In Proc. of the 7th IEEE Int. Conf. on Tools with Artif. Intell.* 392–397.
- MARKOU, M. AND SINGH, S. 2003. Novelty detection: a review - part 1: statistical approaches. *Signal Processing* 83, 2481–2497.
- MASUD, M., GAO, J., KHAN, L., HAN, J., AND THURAISINGHAM, B. 2011. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE TKDE* 23, 6, 859–874.
- MCNEMAR, Q. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12, 2, 153–157.
- MEHTA, M., AGRAWAL, R., AND RISSANEN, J. 1996. SLIQ: A fast scalable classifier for data mining. In *Proc. of the 5th Int. Conf. on Extending Database Technol.: Advances in Database Technol.* EDBT. 18–32.
- MINKU, L., WHITE, A., AND YAO, X. 2010. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering* 22, 730–742.
- MINKU, L. AND YAO, X. 2011. DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering* 24, 4, 619–633.
- MONTEIRO, C., BESSA, R., MIRANDA, V., BOTTERUD, A., WANG, J., AND CONZELMANN, G. 2009. Wind power forecasting: State-of-the-art 2009. Tech. Rep. ANL/DIS-10-1, Argonne National Laboratory.
- MORENO-TORRES, J. G., RAEDER, T., ALAIZ-RODRIGUEZ, R., CHAWLA, N. V., AND HERRERA, F. 2012. A unifying view on dataset shift in classification. *Pattern Recognition* 45, 1, 521–530.
- MOUSS, H., MOUSS, D., MOUSS, N., AND SEFOUHI, L. 2004. Test of Page-Hinkley, an approach for fault detection in an agro-alimentary production system. In *Proc. of the Asian Control Conference*. 815–818.
- MUTHUKRISHNAN, S., VAN DEN BERG, E., AND WU, Y. 2007. Sequential change detection on data streams. In *Workshop Proc. of the 7th IEEE Int. Conf. on Data Mining*. ICDMW. 551–550.
- NG, W. AND DASH, M. 2008. A test paradigm for detecting changes in transactional data streams. In *Proc. of the 13th int. conf. on Database systems for advanced applications*. DASFAA. 204–219.
- NISHIDA, K. AND YAMAUCHI, K. 2007. Detecting concept drift using statistical testing. In *Proceedings of the 10th international conference on Discovery science*. DS'07. Springer-Verlag, Berlin, Heidelberg, 264–269.
- OZA, N. 2001. Online ensemble learning. Ph.D. thesis, University of California Berkeley.
- PAGE, E. S. 1954. Continuous inspection schemes. *Biometrika* 41, 1/2, 100–115.
- PECHENIZKIY, M., BAKKER, J., ZLIOBAITE, I., IVANNIKOV, A., AND KÄRKKÄINEN, T. 2009. Online mass flow prediction in CFB boilers with explicit detection of sudden concept drift. *SIGKDD Explor.* 11, 2, 109–116.
- POLIKAR, R., UDPA, L., UDPA, S., MEMBER, S., MEMBER, S., AND HONAVAR, V. 2001. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Tr. on Sys., Man and Cyber.C* 31, 497–508.
- ROSENBLATT, F. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65, 6, 386–408.
- ROSS, G. J., ADAMS, N. M., TASOULIS, D. K., AND HAND, D. J. 2012. Exponentially weighted moving average charts for detecting concept drift. *Pattern Recogn. Lett.* 33, 191–198.
- RUSU, F. AND DOBRA, A. 2009. Sketching sampled data streams. In *Proc. of the 2009 IEEE Int. Conf. on Data Eng. ICDE*. 381–392.
- SALGANICOFF, M. 1993. Density-adaptive learning and forgetting. In *Proc. of the Int. Conf. on Mach. Learn.* ICML. 276–283.
- SALGANICOFF, M. 1997. Tolerating concept and sampling shift in lazy learning using prediction error context switching. *Artificial Intelligence Review* 11, 1-5, 133–155.
- SCHLIMMER, J. AND GRANGER, R. 1986. Incremental learning from noisy data. *Mach. Learn.* 1, 3, 317–354.
- SCHOLZ, M. AND KLINKENBERG, R. 2007. Boosting classifiers for drifting concepts. *Intell. Data Ana.* 11, 1, 3–28.
- SEBASTIÃO, R. AND GAMA, J. 2007. Change detection in learning histograms from data streams. In *Progress in Artificial Intelligence: Proc. of the Portuguese Conf. on Art. Intell.* 112–123.
- SHAFFER, J. C., AGRAWAL, R., AND MEHTA, M. 1996. SPRINT: A scalable parallel classifier for data mining. In *Proc. of the 22th Int. Conf. on Very Large Data Bases*. VLDB. 544–555.

- SHIRYAEV, A. 2009. On stochastic models and optimal methods in the quickest detection problems. *Theory Probab. Appl.* 53, 3, 385–401.
- STREET, W. AND KIM, Y. 2001. A streaming ensemble algorithm SEA for large-scale classification. In *Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*. KDD. 377–382.
- SYED, N., LIU, H., AND SUNG, K. 1999. Handling concept drifts in incremental learning with support vector machines. In *Proc. of the 5th ACM SIGKDD int. conf. on Knowl. disc. and data mining*. KDD. 317–321.
- TARTAKOVSKY, A. AND MOUSTAKIDES, G. 2010. State-of-the-art in bayesian changepoint detection. *Sequential Analysis* 29, 125–145.
- THRUN, S., M. MONTEMERLO, H. DAHLKAMP, D. STAVENS, A. ARON, J. DIEBEL, P. FONG, J. GALE, M. HALPENNY, G. HOFFMANN, K. LAU, C. OAKLEY, M. PALATUCCI, V. PRATT, P. STANG, S. STROHBAND, C. DUPONT, L. JENDROSSEK, ET AL. 2006. Stanley: the robot that won the darpa challenge. *J. Field Robot.* 23, 9, 661–692.
- TSYMBAL, A. 2004. The problem of concept drift: Definitions and related work. Technical report, Department of Computer Science, Trinity College: Dublin.
- TSYMBAL, A., PECHENIZKIY, M., CUNNINGHAM, P., AND PUURONEN, S. 2006. Handling local concept drift with dynamic integration of classifiers: Domain of antibiotic resistance in nosocomial infections. In *Proc. of 19th IEEE int. symp. on Computer-Based Medical Syst.* CBMS. 679–684.
- VAN DER AALST, W. M. P. 2011. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer.
- VAN DER AALST, W. M. P. 2012. Process mining. *Commun. ACM* 55, 8, 76–83.
- VITTER, J. 1985. Random sampling with a reservoir. *ACM Trans. Math. Softw.* 11, 1, 37–57.
- VORBURGER, P. AND BERNSTEIN, A. 2006. Entropy-based concept shift detection. In *Proc. of the 6th Int. Conf. on Data Mining*. ICDM. 1113–1118.
- VOVK, V. 1998. A game of prediction with expert advice. *J. Comput. Syst. Sci.* 56, 2, 153–173.
- WALD, A. 1947. *Sequential Analysis*. John Wiley and Sons, Inc.
- WANG, H., FAN, W., YU, P., AND HAN, J. 2003. Mining concept-drifting data streams using ensemble classifiers. In *Proc. of the 9th ACM SIGKDD int. conf. on Knowl. disc. and data mining*. KDD. 226–235.
- WIDMER, G. 1997. Tracking context changes through meta-learning. *Mach. Learn.* 27, 3, 259–286.
- WIDMER, G. AND KUBAT, M. 1993. Effective learning in dynamic environments by explicit context tracking. In *Proc. of the Eur. Conf. on Mach. Learn.* ECML. 227–243.
- WIDMER, G. AND KUBAT, M. 1996. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.* 23, 1, 69–101.
- YANG, Y., WU, X., AND ZHU, X. 2006. Mining in anticipation for concept change: Proactive-reactive prediction in data streams. *Data Mining and Knowledge Discovery* 13, 3, 261–289.
- YAO, R., SHI, Q., SHEN, C., ZHANG, Y., AND VAN DEN HENGEL, A. 2012. Robust tracking with weighted online structured learning. In *Proc. of the 12th Eur. Conf. on Computer Vision*. ECCV. 158–172.
- ZEIRA, G., MAIMON, O., LAST, M., AND ROKACH, L. 2004. Change detection in classification models induced from time-series data. In *Data Mining in Time Series Databases*. Vol. 57. 101–125.
- ZHANG, P., ZHU, X., AND SHI, Y. 2008. Categorizing and mining concept drifting data streams. In *Proc. of the 14th ACM SIGKDD int. conf. on Knowl. disc. and data mining*. KDD. 812–820.
- ZHANG, Z. AND ZHOU, J. 2010. Transfer estimation of evolving class priors in data stream classification. *Pattern Recogn.* 43, 9, 3151–3161.
- ZHAO, P., HOI, S., JIN, R., AND YANG, T. 2011. Online AUC maximization. In *Proc. of the 28th Int. Conf. on Machine Learning*. ICML. 233–240.
- ZLIOBAITE, I. 2009. Learning under concept drift: an overview. Technical report, Vilnius University.
- ZLIOBAITE, I. 2011a. Combining similarity in time and space for training set formation under concept drift. *Intelligent Data Analysis* 15, 4, 589–611.
- ZLIOBAITE, I. 2011b. Controlled permutations for testing adaptive classifiers. In *Proc. of the 14th Int. Conf. on Discovery Science*. DS. 365–379.
- ZLIOBAITE, I., BAKKER, J., AND PECHENIZKIY, M. 2012a. Beating the baseline prediction in food sales: How intelligent an intelligent predictor is? *Expert Syst. Appl.* 39, 1, 806–815.
- ZLIOBAITE, I., BIFET, A., GABER, M. M., GABRYS, B., GAMA, J., MINKU, L. L., AND MUSIAL, K. 2012b. Next challenges for adaptive learning systems. *SIGKDD Explorations* 14, 1, 48–55.
- ZLIOBAITE, I., BIFET, A., PFAHRINGER, B., AND HOLMES, G. 2013. Active learning with drifting streaming data. *IEEE Trans. on Neural Networks and Learning Systems*.
- ZLIOBAITE, I. AND KUNCHEVA, L. 2009. Determining the training window for small sample size classification with concept drift. In *Proc. of IEEE Int. Conf. on Data Mining Workshops*. ICDMW. 447–452.

Online Appendix to: A Survey on Concept Drift Adaptation

JOÃO GAMA, University of Porto, Portugal
INDRĚ ŽLIOBAITĚ, Aalto University, Finland
ALBERT BIFET, Yahoo! Research Barcelona, Spain
MYKOLA PECHENIZKIY, Eindhoven University of Technology, the Netherlands
ABDELHAMID BOUCHACHIA, Bournemouth University, UK

A. PSEUDOCODE OF CONCEPT DRIFT ALGORITHMS

ALGORITHM 1: Page-Hinkley Algorithm

input: Admissible change δ , Drift threshold λ , Loss at example t : e_t ;
output: $drift \in \{TRUE, FALSE\}$;
/* Initialize the error estimators */;
 $SR(0) \leftarrow 0$; $m_T(0) \leftarrow 0$; $M_T \leftarrow 1$;
/* Page Hinkley test */;
Let $SR(t) \leftarrow SR(t-1) + R(t)$;
Let $m_T(t) \leftarrow m_T(t-1) + R(t) - \frac{SR(t)}{t} - \delta$;
Let $M_T \leftarrow \min(M_T, m_T(t))$;
if $m_T(t) - M_T \geq \lambda$ **then**
| $drift \leftarrow TRUE$;
else
| $drift \leftarrow FALSE$;
end

ALGORITHM 2: The ADWIN change detection algorithm

begin
| Initialize Window W ;
| **foreach** $(t) > 0$ **do**
| | $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W) ;
| | **repeat**
| | | Drop elements from W
| | | **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \epsilon_{cut}$ holds for every split of W into $W = W_0 \cup W_1$;
| | **end**
| | Output $\hat{\mu}_W$
end

© 2013 ACM 0360-0300/2013/01-ART1 \$10.00
DOI 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

ALGORITHM 3: The SPC change detection algorithm

input: Current decision model \mathcal{L} ; X, y^T : Training Set with class $y \in Y$;
 α and β parameters

begin

 Let X_t, y_t be the current instance and $\hat{y}_t \leftarrow \mathcal{L}(X_t)$ be the prediction;
 Let $error_t \leftarrow E(\hat{y}_t, y_t)$, compute mean p_t and variance σ_t of with $error_t$;
 if $p_t + \sigma_t < p_{min} + \sigma_{min}$ **then**
 | $p_{min} \leftarrow p_t$ and $\sigma_{min} \leftarrow \sigma_t$;
 end
 if $p_t + \sigma_t < p_{min} + \beta \times \sigma_{min}$ **then**
 | In-control: Warning? $\leftarrow False$; update \mathcal{L} with the instance X_t, y_t ;
 else
 | **if** $p_t + \sigma_t < p_{min} + \alpha \times \sigma_{min}$ **then**
 | | **if NOT** Warning **then**
 | | | Warning: Warning? $\leftarrow True$; start buffer $\leftarrow \{X_t, y_t\}$;
 | | **else**
 | | | Warning: buffer \leftarrow buffer $\cup \{X_j, y_j\}$;
 | | **end**
 | **else**
 | | Out-control: train a new decision model using the instances in the buffer;
 | | Warning? $\leftarrow False$; re-start p_{min} and σ_{min} ;
 | **end**
 end
end

ALGORITHM 5: Algorithm for Dynamic Weighted Majority (DWM)

Input:
 $\{\vec{x}, y\}_n^1$: training data, feature vector and class label; $c \in \mathbb{N}^*$: number of classes $c \geq 2$; β :
factor for decreasing weights, $0 \leq \beta < 1$; θ : threshold for deleting experts; p : period between
expert removal, creation, and weight update; $\{e, w\}_m^1$: set of experts and their weights;
 $\Lambda, \lambda \in \{1, \dots, c\}$: global and local predictions; $\vec{\sigma} \in \mathbb{R}^c$: sum of weighted predictions for each
class.

begin

$m \leftarrow 1$
 $e_m \leftarrow$ Create-New-Expert()
 $w_m \leftarrow 1$
 foreach $i \leftarrow 1, \dots, n$ **do**
 | $\vec{\sigma} \leftarrow 0$
 | **foreach** $j \leftarrow 1, \dots, m$ **do**
 | | $\lambda \leftarrow$ Classify(e_j, \vec{x}_i)
 | | **if** $\lambda \neq y_i \wedge i \bmod p = 0$ **then**
 | | | $w_j \leftarrow \beta w_j$
 | | **end**
 | | $\sigma_\lambda \leftarrow \sigma_\lambda + w_j$
 | **end**
 | $\Lambda \leftarrow \text{argmax}_j \sigma_j$
 | **if** $i \bmod p = 0$ **then**
 | | $w \leftarrow$ Normalize-Weights (w)
 | | $\{e, w\} \leftarrow$ Remove-Experts ($\{e, w\}, \theta$)
 | | **if** $\Lambda \neq y_i$ **then**
 | | | $m \leftarrow m + 1$
 | | | $e_m \leftarrow$ Create-New-Expert()
 | | | $w_m \leftarrow 1$
 | | **end**
 | **end**
 | **foreach** $j \leftarrow 1, \dots, m$ **do**
 | | $e_j \leftarrow$ Train(e_j, \vec{x}_i, y_i)
 | **end**
 | **return** Λ
 end
end

ACM Computing Surveys, Vol. 1, No. 1, Article 1, Publication date: January 2013.

ALGORITHM 4: The CVFDT algorithm

input: S is a sequence of examples,
 X is a set of symbolic attributes,
 $G(\cdot)$ is a split evaluation function,
 δ is one minus the desired probability of
choosing the correct attribute at any given node,
 τ is a user-supplied tie threshold,
 w is the size of the window,
 n_{min} is the # examples between checks for growth,
 f is the # examples between checks for drift.

output: HT is a decision tree.

Procedure CVFDT($S, X, G, \delta, \tau, w, n_{min}$)
/ Initialize */*
Let HT be a tree with a single leaf l_1 (the root).
Let $ALT(l_1)$ be an initially empty set of alternate tree for l_1 .
Let W be the window of examples, initially empty.
Initialize sufficient statistics to compute G .
/ Process the examples */*
foreach *example* (x, y) *in* S **do**
 Sort (x, y) into a set of leaves L using HT and all trees in ALT of any node (x, y) passes through.
 Let ID be the maximum id of the leaves in L .
 Add $((x, y), ID)$ to the beginning of W .
 if $|W| > w$ **then**
 Let $((x_w, y_w), ID_w)$ be the last element of W
 ForgetExamples($HT, n, (x_w, y_w), ID_w$)
 Let $W = W$ with $((x_w, y_w), ID_w)$ removed
 end
 CVFDTGrow($HT, n, G, (x, y), \delta, n_{min}, \tau$)
 if *there have been* f *examples since the last checking of alternate trees* **then**
 CheckSplitValidity(HT, n, δ)
 end
end
return HT .

B. DATASETS FOR CONCEPT DRIFT**B.1. Synthetic**

Synthetic data has several benefits: they are easy to reproduce and bear little cost of storage and transmission. Seven popular changing data generators are summarized in Table VI and described below. Synthetic data provides an advantage of knowing the ground truth (e.g. where exactly concept drift happens, what is the drift, what are classification accuracies achievable on each concept). The main limitation of synthetic data is uncertainty whether corresponding drifts happen in reality.

SEA Concepts Generator. This data model due to [Street and Kim 2001] presents abrupt concept drifts (real). The data contains three independent real valued attributes in $[0, 10]$, only the first two attributes are relevant for prediction. The original data model produces four different concepts. The class decision boundary is defined as $x_1 + x_2 \leq \theta$, where x_1 and x_2 are the first two attributes and θ is a threshold value different for each concept: (1) $\theta = 9$, (2) $\theta = 8$, (3) $\theta = 7$ and (4) $\theta = 9.5$.

STAGGER Concepts Generator. This data model due to [?] presents abrupt concept drifts (real). The data contains three independent categorical attributes: size \in small, medium, large, color \in red, green, blue and shape \in square, circular, trian-

ALGORITHM 6: The AddExp algorithm for discrete classes

input: X, y^T A Training Set with class $y \in Y$
 $\beta \in [0, 1]$: factor for decreasing weights; $\tau \in [0, 1]$: loss required to add a new expert

begin

- Set the initial number of experts: $N_1 \leftarrow 1$;
- Set the initial expert weight: $w_{1,1} \leftarrow 1$;
- for** $t \leftarrow 1$ **to** T **do**
 - Get expert predictions: $\epsilon_{t,1}, \dots, \epsilon_{t,N_t} \in Y$;
 - Compute prediction: $\hat{y}_t = \operatorname{argmax}_{c \in Y} \sum_{i=1}^{N_t} w_{t,i} [c = \epsilon_{t,i}]$;
 - Update experts weights: $w_{t+1,i} \leftarrow w_{t,i} \beta^{[y_t \neq \epsilon_{t,i}]}$;
 - if** $\hat{y}_t \neq y_t$ **then**
 - Add a New Expert;
 - $N_{t+1} \leftarrow N_t + 1$;
 - $w_{t+1,N_{t+1}} \leftarrow \gamma \sum_{i=1}^{N_t} w_{t,i}$;
 - end**
 - Train each expert on instance X_t, y_t ;
- end**

ALGORITHM 7: The AddExp algorithm for continuous classes

input: X, y^T A Training Set with class $y \in [0 : 1]$; $\beta \in [0, 1]$: factor for decreasing weights
 $\gamma \in [0, 1]$: factor for new expert weight; $\tau \in [0, 1]$: loss required to add a new expert

begin

- Set the initial number of experts: $N_1 \leftarrow 1$;
- Set the initial expert weight: $w_{1,1} \leftarrow 1$;
- for** $t \leftarrow 1$ **to** T **do**
 - Get expert predictions: $\epsilon_{t,1}, \dots, \epsilon_{t,N_t} \in [0, 1]$;
 - Compute prediction: $\hat{y}_t = \frac{\sum_{i=1}^{N_t} w_{t,i} \epsilon_{t,i}}{\sum_{i=1}^{N_t} w_{t,i}}$;
 - Suffer loss $\|\hat{y}_t - y_t\|$;
 - Update experts weights: $w_{t+1,i} \leftarrow w_{t,i} \beta^{\|\epsilon_{t,i} - y_t\|}$;
 - if** $\|\hat{y}_t - y_t\| \geq \tau$ **then**
 - Add a New Expert;
 - $N_{t+1} \leftarrow N_t + 1$;
 - $w_{t+1,N_{t+1}} \leftarrow \gamma \sum_{i=1}^{N_t} w_{t,i} \|\epsilon_{t,i} - y_t\|$;
 - end**
 - Train each expert on instance X_t, y_t ;
- end**

Table VI. Models for synthetic data generation

Name	# of concepts	Task	Type of drift			
			real CD	virtual CD	priors	novelties
SEA	4	classification (2)	✓	–	✓	–
STAGGER	3	classification (2)	✓	–	✓	–
Rotating hyperplane	any	classification (2)	✓	–	–	–
RBF generator	any	classification (any)	✓	✓	✓	–
Function Generator	10	classification (2)	✓	–	✓	–
LED Generator	–	classification (10)	–	✓	–	–
Waveform Generator	–	classification (3)	–	✓	–	–

gular. A binary classification task is defined by a disjunct of conjuncts. There are three concepts: (1) positive class *if* size = small *and* color = red, (2) color = green *or* shape = circular, (3) size = medium *or* size = large.

Rotating Hyperplane. This data model was first used to test CVFDT against VFDT in [Hulten et al. 2001]. Data is generated uniformly from a hyperplane d -dimensional space. The decision boundary is defined as $\sum_{i=1}^d w_i x_i = w_0 = \sum_{i=1}^d w_i$, where where x_i , is the i^{th} attribute. Examples for which $\sum_{i=1}^d w_i x_i \geq w_0$ are labeled positive, and examples for which $\sum_{i=1}^d w_i x_i < w_0$ are labeled negative. Concept changes are introduced by modifying the weights w_i (they need to satisfy the constraint $w_0 = \sum_{i=1}^d w_i$ to keep the priors fixed). Hyperplanes are useful for simulating time-changing concepts, because we can change the orientation and position of the hyperplane in a smooth manner by changing the relative size of the weights. Noise can be added by randomly swapping class labels.

Random RBF Generator. This generator was devised to offer an alternate complex concept type that is not straightforward to approximate with a decision tree model. The RBF (Radial Basis Function) [Bifet et al. 2009] generator works as follows. A fixed number of random centroids are generated. Each center has a random position, a single standard deviation, class label and weight. New examples are generated by selecting a center at random, taking weights into consideration so that centers with higher weight are more likely to be chosen. A random direction is chosen to offset the attribute values from the central point. The length of the displacement is randomly drawn from a Gaussian distribution with standard deviation determined by the chosen centroid. The chosen centroid also determines the class label of the example. This effectively creates a normally distributed hypersphere of examples surrounding each central point with varying densities. Only numeric attributes are generated. Drift is introduced by moving the centroids with constant speed.

Function Generator. It originates from [Agrawal et al. 1992], and was a popular data model for early work on scaling up decision tree learners [Agrawal et al. 1993; Mehta et al. 1996; Shafer et al. 1996; Gehrke et al. 1998]. The generator produces a stream containing nine attributes, six numeric and three categorical. Although not explicitly stated by the authors, a sensible conclusion is that these attributes describe hypothetical loan applications. There are ten functions defined for generating binary class labels from the attributes, presumably meaning approval or the loan. The original data model has no drift. Concept drift may be introduced by switching between functions to generate class labels.

LED Generator. This data source originates from the CART book [Breiman et al. 1984] and an implementation in C is available from the UCI repository [Bache and Lichman 2013]. The goal is to predict the digit displayed on a seven-segment LED display, where each attribute has a 10% chance of being inverted. The optimal Bayes classification accuracy is 74%. The original data model has no drift. Drift may be introduced by swapping the positions of attributes.

Waveform Generator. It shares its origins with LED, and is available from the UCI repository. The goal of the task is to distinguish three classes of a waveform, each of which is generated from a combination of two or three base waves. The optimal Bayes classification accuracy is 86%. There are two versions of the problem, wave21 which has 21 numeric attributes, all of which include noise, and wave40 which introduces an additional 19 irrelevant attributes. The original data model has no drift. Drift may be introduced by swapping the positions of attributes.

Implementations of these data models are available in MOA [Bifet et al. 2011b].

B.2. Real-World Data

Nowadays, it is easier than in the past, to find large real-world datasets for public benchmarking with concept change. The UCI machine learning repository [Bache and Lichman 2013] contains some real-world benchmark data for evaluating machine learning techniques, however they are not large datasets.

The main real data domains of the large datasets mentioned in the *Concept Drift* page of Wikipedia⁶ are the following ones.

Text mining. Documents of text that contains words or combinations of words as the features to use to process the data. A collection of text mining datasets with concept drift is maintained by I. Katakis⁷. Data recollected from Twitter may be also be considered for text mining.

Electricity. A widely used dataset is the Electricity Market Dataset introduced in [Harries 1999]. This time series based data was collected from the Australian New South Wales Electricity Market, available from J. Gama⁸. In this market, the prices are not fixed and are affected by demand and supply of the market. The prices in this market are set every five minutes. The *ELEC2* dataset contains 45,312 instances. Each example of the dataset refers to a period of 30 minutes, i.e. there are 48 instances for each time period of one day. The class label identifies the change of the price related to a moving average of the last 24 hours. The class level only reflect deviations of the price on a one day average and removes the impact of longer term price trends.

Email Spam. Datasets of email messages used to predict if they are unsolicited messages or not. For example, *ECUE Spam* datasets are two datasets each consisting of more than 10,000 emails collected over a period of approximately 2 years referring to a one user compiled by S.J.Delany⁹.

Business oriented. Datasets that contains data used in decision management systems in companies. *PAKDD'09 competition* dataset¹⁰ is used for a credit evaluation task. It is collected over a five year period, unfortunately the true labels are released only for the first part of the data. Another dataset called *Airline*, contains approximately 116 million flight arrival and departure records (cleaned and sorted) and it is compiled by E. Ikonomovska¹¹.

Games. Datasets obtained from online or non-online games: for example, a dataset *Chess.com* (online games) compiled by I.Žliobaitė¹².

C. EVALUATION EXAMPLE

We give an example on evaluation of data stream classification, using the MOA software framework [Bifet et al. 2011b]. MOA is an open-source framework for dealing with massive evolving data streams. MOA is related to WEKA [Hall et al. 2009], the Waikato Environment for Knowledge Analysis, which is an award-winning open-source workbench containing implementations of a wide range of batch machine learning methods.

⁶http://en.wikipedia.org/wiki/Concept_drift retrieved 12/11/2012

⁷http://mlkd.csd.auth.gr/concept_drift.html

⁸<http://www.liaad.up.pt/kdus/products/datasets-for-concept-drift>

⁹http://www.comp.dit.ie/aigroup/?page_id=729

¹⁰<http://sede.neurotech.com.br:443/PAKDD2009/arquivo.do?method=load>

¹¹http://kt.ijs.si/elena_ikonovska/data.html

¹²<https://sites.google.com/site/zliobaite/resources-1>

MOA enables evaluation of data stream classification algorithms on large streams, in the order of tens of millions of instances under explicit memory limits. Any less than this does not actually test data stream algorithms in a realistically challenging setting.

MOA is written in Java. The main benefits of Java are portability, where applications can be run on any platform with an appropriate Java virtual machine, and the strong and well-developed support libraries. Use of the language is widespread, and features such as automatic garbage collection help to reduce programmer burden and error.

MOA contains stream generators, classifiers and evaluation methods. Figure 10 shows the MOA graphical user interface. A command line interface is also available.

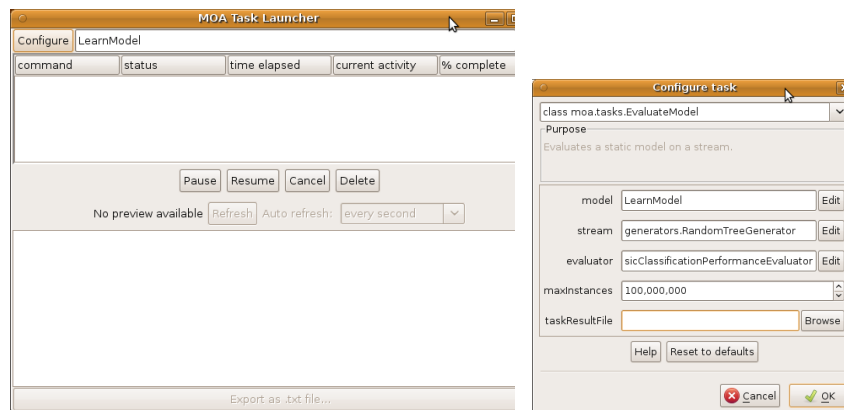


Fig. 10. MOA Graphical User Interface

Considering data streams as data generated from pure distributions, MOA models a concept drift as a weighted combination of two pure distributions that characterizes the target concepts before and after the drift. Within the framework, it is possible to define the probability that instances of the stream belong to the new concept after the drift using the sigmoid function, as an elegant and practical solution [Bifet et al. 2009].

MOA contains the popular data generators described in Appendix B. MOA streams can be built using generators, reading ARFF files, joining several streams, or filtering streams. They allow for the simulation of a potentially infinite sequence of data. The following generators are currently available: Random Tree Generator, SEA Concepts Generator, STAGGER Concepts Generator, Rotating Hyperplane, Random RBF Generator, LED Generator, Waveform Generator, and Function Generator.

MOA contains several classifier methods such as: Naive Bayes, Decision Stump, Hoeffding Tree, Hoeffding Option Tree, Adaptive Hoeffding Tree, Bagging, Boosting, Bagging using ADWIN [Bifet et al. 2009], and Leveraging Bagging [Bifet et al. 2010a].

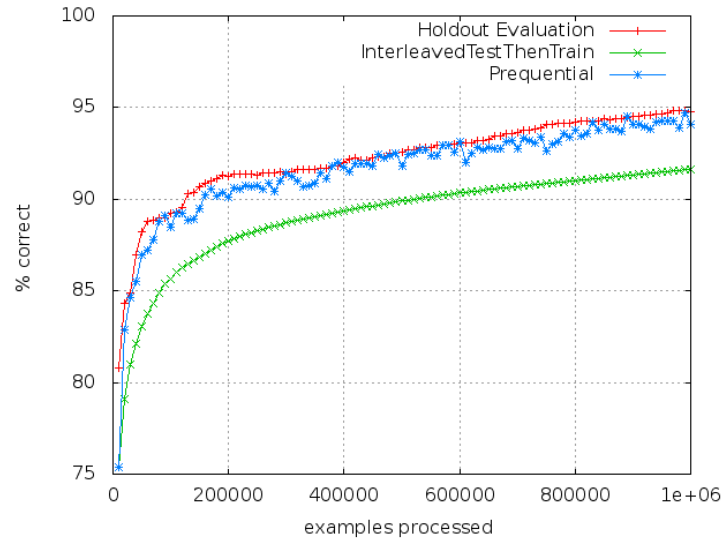


Fig. 11. Evaluation for a stream of a million of instances comparing hold-out, prequential with landmark window and prequential with sliding window.

Figure 11 shows a comparison between a hold-out evaluation, a prequential evaluation using a landmark window (*InterleavedTestThenTrain*), and a prequential evaluation using a sliding window of size 1000. We observe that the prequential evaluation using a sliding window is a good approximation to the hold-out evaluation.

We run also on MOA the following experiment simulating a concept drift scenario: a prequential evaluation using a sliding window of size 1000 for a stream of a million of instances generated by the Random RBF Generator, with the following learners: Hoeffding Tree, Adaptive Hoeffding Tree, and ADWIN Bagging and Leveraging Bagging. The stream is evolving and the centroids are moving with constant speed 10^{-4} : this speed is defined as the distance moved each new instance arrives and it is initialized by a drift parameter.

Figure 12 shows accuracy, Kappa statistic and RAM-Hours for this experiment. We observe that the Hoeffding Tree is the method with lower capacity of adaption. Ensemble methods perform better than single classifiers, but they have a higher cost in RAM-Hours. Leveraging Bagging is the method with higher accuracy and Kappa statistic, but the number of resources that it needs is considerably larger. Data stream evaluation is a two-dimensional process with a trade-off between accuracy results and resource costs.

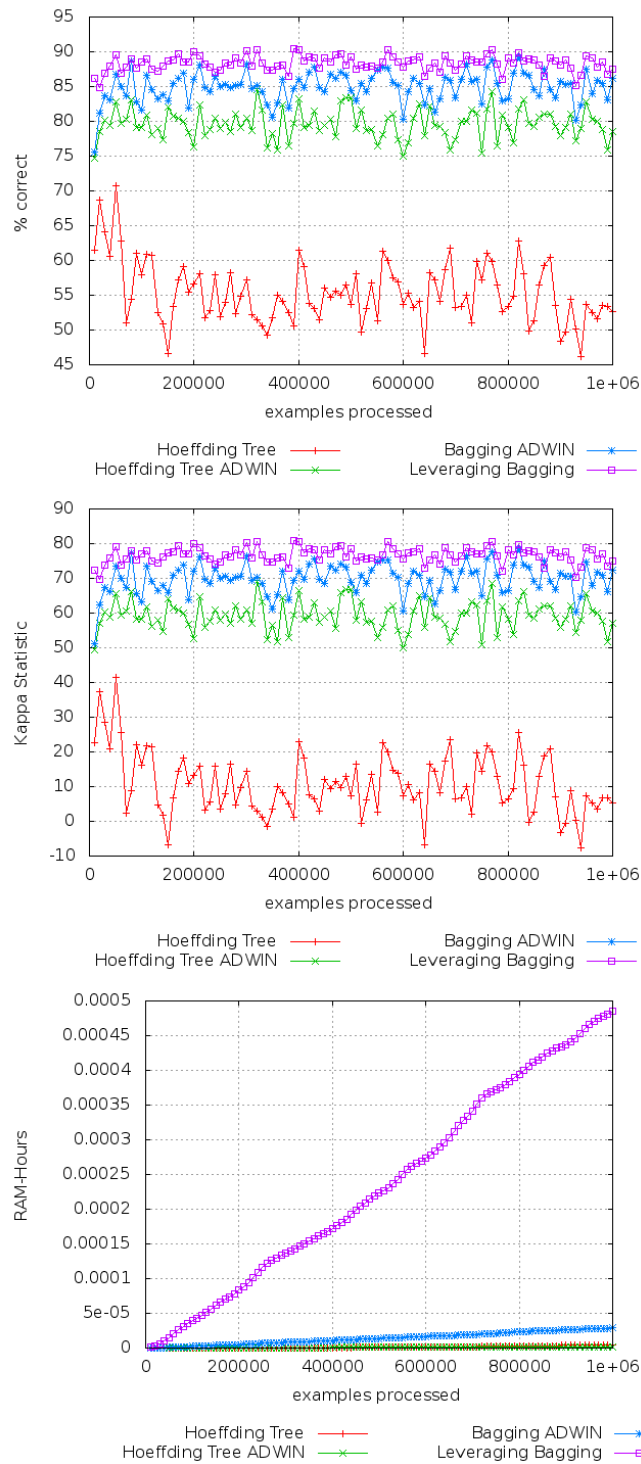


Fig. 12. Prequential Evaluation for a RBF stream of a million of instances where the centers are moving with a speed of 10^{-4} .