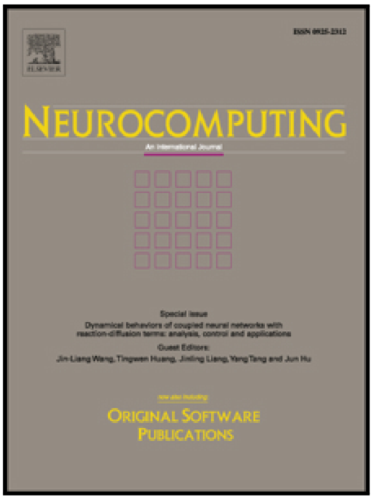
Communicated by Dr Xiaofeng Zhu


ISSN 0925-2312
NEUROCOMPUTING
An International Journal
Special issue
Dynamical behaviors of coupled neural networks with reaction-diffusion terms: analysis, control and applications
Guest Editors:
Jin-Liang Wang, Tingwen Huang, Jinling Liang, Yang Tang and Jun Hu
ORIGINAL SOFTWARE PUBLICATIONS

A Taxonomic Look at Instance-based Stream Classifiers

Iain A.D. Gunn, Álvar Arnaiz-González, Ludmila I. Kuncheva

PII: S0925-2312(18)30095-X
DOI: 10.1016/j.neucom.2018.01.062
Reference: NEUCOM 19264

To appear in: *Neurocomputing*

Received date: 23 February 2017
Revised date: 29 July 2017
Accepted date: 24 January 2018

A Taxonomic Look at Instance-based Stream Classifiers

Iain A. D. Gunn[a,1,*], Álvar Arnaiz-González[b], Ludmila I. Kuncheva[a]

[a]*School of Computer Science, Bangor University, Dean Street, Bangor LL57 1UT, UK*
[b]*Escuela Politécnica Superior, University of Burgos, 09006 Burgos, SPAIN*

Abstract

Large numbers of data streams are today generated in many fields. A key challenge when learning from such streams is the problem of concept drift. Many methods, including many prototype methods, have been proposed in recent years to address this problem. This paper presents a refined taxonomy of instance selection and generation methods for the classification of data streams subject to concept drift. The taxonomy allows discrimination among a large number of methods which pre-existing taxonomies for offline instance selection methods did not distinguish. This makes possible a valuable new perspective on experimental results, and provides a framework for discussion of the concepts behind different algorithm-design approaches. We review a selection of modern algorithms for the purpose of illustrating the distinctions made by the taxonomy. We present the results of a numerical experiment which examined the performance of a number of representative methods on both synthetic and real-world data sets with and without concept drift, and discuss the implications for the directions of future research in light of the taxonomy. On the basis of the experimental results, we are able to give recommendations for the experimental evaluation of algorithms which may be proposed in the future.

Keywords: Machine learning, stream classification, instance selection, prototype generation, concept drift

[*] Corresponding author

Email addresses: i.gunn@mdx.ac.uk (Iain A. D. Gunn), alvarag@ubu.es (Álvar Arnaiz-González), l.i.kuncheva@bangor.ac.uk (Ludmila I. Kuncheva)

[1] Present address: Department of Computer Science, Middlesex University, The Burroughs, London NW4 4BT

1. Introduction

Storing large data sets can be problematic, especially in stream learning, where data is continuously arriving. This issue is more relevant than ever in an era of "big data", where important problems involve data streams which cannot be stored in full [1, 2]. Many techniques have been suggested for forming reduced reference sets for instance-based classifiers, in particular the nearest-neighbour classifier [3, 4]. However, as we argued in a previous contribution [5], the taxonomy developed for describing offline algorithms for data editing is inadequate to describe algorithms for streaming data.

In summary, the offline taxonomy fails because many approaches developed for offline editing are inherently unsuitable for streaming data. For example, in the offline case, there are methods which only add instances to the reference set, never removing them; methods which only remove instances from the reference set (starting with all the training data), and never re-add them; and methods which both add and remove instances as they run. These are distinguished taxonomically as "incremental", "decremental", and "mixed" methods. Clearly, editing methods which can only add or only remove examples are unsuitable for dealing with unbounded data streams. Only "mixed" methods can be used in the streaming case, so the offline taxonomic distinction is useless for the streaming case. In the streaming case, all methods now being "mixed", the key taxonomic question of interest is the choice of processes by which instances are added to and removed from the reference set in response to the stream of arriving data, as it is here that the nature of the streaming problem forces a great difference in approach from the offline case.

In addition to the need for editing, a second key issue with stream learning is that data streams may typically be "non-stationary", that is, subject to "concept drift". We also found previously [5] that the established taxonomy developed to describe algorithms designed to deal with concept drift [6] cannot sensibly be used to classify instance-based algorithms. The existing taxonomy in this case used separate concepts of "Data Management" and "Memory" which could not be applied to lazy learners, for which memory simply *is* data retention.

This paper expands our previous study [5] on instance selection methods for drifting data streams. In addition to augmenting and refining the taxonomy of such methods, we carry out a numerical experiment to compare the performance of some modern algorithms, in light of the taxonomy.

The present work also gives greater consideration to prototype generation methods, typified by the learning vector quantisation (LVQ) family [7].

Note that our study considers the various algorithmic approaches for forming a reference set from streaming data, not the variety of instance-based classifiers which might use such a reference set. We do not compare alternative classifiers: we simply use the nearest-neighbour (1NN) classifier. (Differences between classification rules may be taxonomically considered as for the offline case.)

The rest of the paper is organised as follows. The formal problem of classification of a stream subject to concept drift, and related terminology, are introduced in section 2. Data-editing methods are introduced in section 3. Our refined taxonomy of instance-based methods for the concept drift problem is presented in section 4. The algorithms included in the experiment, and some other representative algorithms, are discussed in section 5. Our experimental set-up and results, with discussion, are presented in section 6. The conclusion section 7 contains recommendations for future experimental practice.

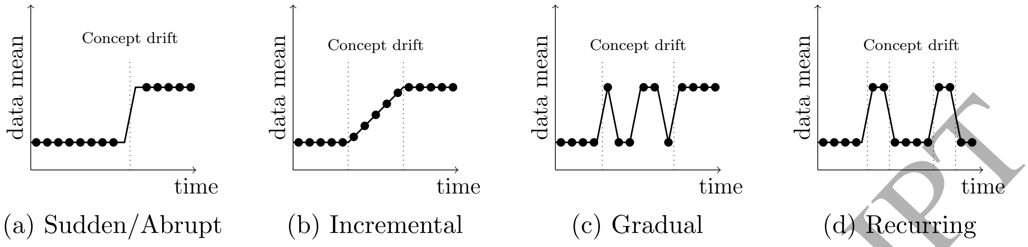
2. The concept-drift problem

The streaming version of the classification problem is typically posed thus:

- One data point $\mathbf{x} \in \mathbb{R}^d$ is received at time t.
- The class label of the point is not available at time t. The point is labelled by the classifier.
- The true label is then revealed before the next data point is classified.

The model can easily be altered to a batch-input form, in which a set of N points $X \subset \mathbb{R}^d$ is considered to arrive all at once at time t, and all N points must be labelled before the true labels are revealed and the next input batch arrives.

"Concept drift" is the generally accepted term for change in the probability distributions related to the problem, and the management of this problem is essential in streaming learning [8]. Occurrences of concept drift have been described in terms of the behaviour of the stream at the onset of the drift: see Figure 1 for an illustration of this idea. The terminology is taken from Bose et al. [9] and Gama et al. [6]. Concept drift may be sudden, or the



(a) Sudden/Abrupt (b) Incremental (c) Gradual (d) Recurring

Figure 1: Main concept drift types illustrated schematically as if for one-dimensional data. Adapted from Gama et al. [6].

underlying distribution may pass continuously and relatively slowly through intermediate states ("incremental drift"). The original concept may then be gone forever, or it may recur, briefly (in "gradual drift"), or indefinitely, in which case it is called a true "recurring concept". In general, it is to be expected that some algorithms deal better than others with certain forms of drift. For example, algorithms which explicitly maintain a library of former concepts have been so engineered in order to perform better when the stream includes recurring concepts, but can only be disadvantaged by this apparatus when applied to a stream containing only sudden, irrevocable concept shifts. (This approach of storing former concepts for re-use is typified by the FLORA3 algorithm [10], one of the first algorithms to explicitly address recurring concepts. It is part of the FLORA family of algorithms [11], [12], dating back to 1989.)

Whether such a collection of former concepts is maintained or not, an algorithm for handling concept drift will have both a learning mechanism of some sort and a *forgetting* mechanism of some sort, the latter being essential to ensure the classifier does not become stuck in some setting after seeing a large amount of data which exceeds its capacity for learning. Some methods use explicit *change detection* strategies, which allow the algorithm to make a suitable increase in learning and forgetting rates when a concept shift is detected. We refer the reader to the survey of Gama et al. [6] for a good recent review of concept-drift adaptation methods.

Concept drift is of interest to the extent that it affects adversely the future performance of the classifier and requires action: the occasional outlier or short abnormal event should simply be treated as noise and ignored.

3. Prototype selection and generation

One key distinction among data-editing methods must be introduced before the entire taxonomy is presented. This is the distinction between "prototype selection" and "prototype generation" families, which have been treated very similarly in taxonomies of their offline members [3, 4], but which we have argued [5] need very different treatment in their online incarnations.

The process of editing training data for use with the nearest-neighbour classifier, or similar instance-based classifiers, consists of replacing the set of training data, S, with a *smaller* reference set of what are called "prototypes". The meaning of "prototype" depends on the approach (selection or generation) chosen for the data editing[2]. In prototype selection, the reduced set of prototypes, S' is a subset of S (along with the labels of the objects). In prototype generation, the prototypes are allowed to be different points in the same space (or to be extended as other structures such as hyper-rectangles or hyper-ellipses). Generated prototypes in the original space can be created by various procedures for relabelling, merging or re-positioning members of an initial subset of S, such as by finding cluster centres.

Prototype generation is potentially the richer of the two data-editing approaches. In prototype generation, the entire space is available for the positioning of prototypes, allowing the approximation of any classification boundary with a specified precision. If instead the prototypes are constrained to be chosen from the finite set of points constituting the training data, the set of possible boundaries is correspondingly reduced.

The learning vector quantisation (LVQ) family of methods are exemplars of the Prototype Generation approach which are particularly popular and successful in the online case. Many algorithms of the LVQ type have been developed, and a taxonomy of this family has been suggested [7], in which the algorithms are distinguished by their theoretical approach and concept of distance. However, as we previously found for prototype selection algorithms [5], the taxonomy developed for offline algorithms is no longer appropriate in the streaming case. For example, if the data stream may be subject to concept drift, it certainly cannot be assumed that the data is drawn i.i.d. from a stationary distribution; this destroys the usual approach to likelihood-maximisation and hence the taxonomic approach which consid-

[2] Synonyms of prototype generation in the literature are prototype *construction*, *extraction*, *reduction* and *replacement*.

ers this theoretical basis. There is important work to be done in developing theoretical approaches for concept drift; but for now, online algorithms are heuristic and cannot be classified in terms of theoretical approach.

There are two main differences between streaming LVQ algorithms and offline ones. The first difference is in the initialisation process: LVQ algorithms need an initialisation step that is not straightforward in the streaming case. Second, in the offline case, LVQ does not require a mechanism for adding/removing prototypes because the number of prototypes is typically fixed in advance. It is to be expected that new prototypes would be added to a streaming LVQ algorithm only to cope with a sudden new concept, in contrast to a prototype selection algorithm for which the addition and removal of prototypes is part of the adjustment to incremental drift as well. Nevertheless, it may turn out that questions of the mechanism for adding and removing prototypes are at least as important for the classification of streaming LVQ algorithms as the details of the prototype update procedure, as the choice of mechanism may strongly affect how the algorithm behaves in the presence of concept drift.

On the prototype selection side of things, the nature of streaming data renders impossible the application of many established offline techniques for forming reduced reference sets. As mentioned in the Introduction, methods which proceed in a single "direction" by either starting with the empty set and adding prototypes to it, or starting with the set of all training data and removing prototypes from it, are not applicable to streaming data. Any algorithm designed to maintain an up-to-date reference set in response to an indefinite amount of streaming input data must necessarily have both a mechanism for adding new prototypes to the reference set, and a mechanism for removing prototypes from the reference set (even if the mechanism for prototype addition is as simple as to add all new examples as they come in). For this reason, our taxonomy differs substantially from the usual taxonomy of offline data reduction algorithms. We characterise the algorithms in terms of prototype addition and removal mechanisms, instead of the traditional terms of condensation vs editing, increasing vs decreasing "direction", or wrapper vs filter evaluation (for discussion of which see our previous contribution [5]).

4. A taxonomy of nearest neighbour methods for streaming data

The taxonomy we present here is an augmentation and refinement of that we created in our previous contribution [5], where we brought together the

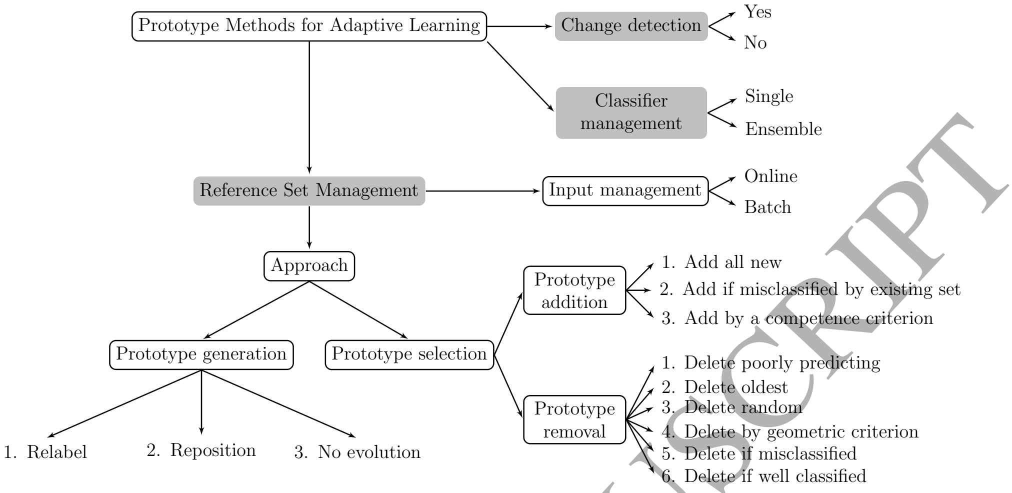


Figure 2: A taxonomy of nearest-neighbour methods for streaming data. The nodes in boxes show properties that should be specified, and their values are chosen among the leaves of the respective sub-tree. Shaded boxes are areas for which algorithms might be developed independently. For example, an ensembling method might be developed which is agnostic regarding the base classifier; the base classifier could then be any classifier trained on the chosen reference set.

taxonomies of instance selection/generation and concept-drift methods for the first time. Fig. 2 shows the proposed taxonomy. The shaded boxes correspond to three areas for which algorithms have separately been proposed: online maintenance of a reference set of prototypes; ensembling of classifiers; and change detection. A proposed method for classifying streaming data might be an algorithm for any of these three tasks, or might be presented as a "system" or "framework" with algorithms for these separate areas.

We do not seek here to develop a taxonomy of change-detection algorithms or of ensembling methods[3]. We have therefore left the options in Fig. 2 for these categories as simple yes-or-no choices, to indicate whether such algorithms are in use or not, although a richer description is possible and could be deemed important in future taxonomic studies. The most important and difficult first step is to develop a taxonomy for the methods of reference set management, because reference set management is the core of the area:

[3] We refer readers interested in taxonomies for change-detection to Webb et al. [13]. For ensembling methods we recommend the recent survey of Gomes et al. [14].

there can be no use for ensembling or change detection methods without a base classifier for them to work with.

4.1. Reference set management

The key distinction from the offline case is that concepts of "editing" or "condensing" must be replaced with a concept of "reference set management" for the streaming case. The potential presence of concept drift requires any instance-based algorithm to continue to be able to adopt new instances indefinitely; it must therefore also have the ability to continue to remove old instances indefinitely.

4.1.1. Input management

The "Input Management" category is used to make a distinction between algorithms which operate in a truly online manner on a single new data point at a time, and those for which the data stream is regarded as arriving in batches. This distinction is separate from the issue of which prototypes are "remembered" and for how long.

4.1.2. Approach: Prototype (instance) selection versus prototype generation

The distinction between these two broad families was introduced in section 3. Prototype selection methods in our taxonomy are classified according to the many different possibilities for adding and, especially, removing prototypes from the reference set. In contrast, prototype generation methods are typically thought of as updating persistent prototypes, though a mechanism for (occasionally) adding new prototypes will usually be necessary. These methods are therefore classified according to the prototype evolution mechanism: leaf 1 corresponds to the relabelling of prototypes; and leaf 2 corresponds to the repositioning of prototypes, as for LVQ methods. However, it is not quite universally true that prototype generation methods proceed by evolving prototypes; therefore, there is a third leaf to the prototype generation branch. It describes those methods which do not update their generated prototypes, but simply add and remove them to and from the reference set as the prototype selection methods do. (SyncStream [15] is the algorithm we consider which has this behaviour; see section 5.)

4.2. Change detection

This category distinguishes between approaches which attempt actively to determine whether the input stream is undergoing concept drift at a given time and adapt their strategy accordingly, and those whose adaptation is a purely passive consequence of continuing to update the reference set.

4.3. Classifier management

Ensembling of classifiers is a popular and successful technique, though less so for nearest-neighbour classifiers than for other classifiers such as decision trees. This is usually said to be a result of the "stability" of nearest-neighbour methods, which makes them less suitable than unstable methods as ensemble base methods. However, there may be promise in ensembles of nearest-neighbour methods whose base classifiers use different subsets of the features [16].

5. Algorithms considered

This section reviews some existing editing kNN methods for streaming data, for the purpose of illustrating the taxonomy. We consider this to be a representative collection, and do not claim a comprehensive coverage of the area. Table 1 shows the proposed classification under the new taxonomy of the algorithms reviewed in this section. A subset of these algorithms will subsequently be used in section 6 for our experimental study.

5.1. Prototype selection methods

Historically, Aha et al.'s Instance-Based Learning Algorithm IB3 was the first prototype selection technique capable of handling concept drift [17]. IB3 adds new instances to the reference set if they are misclassified by the existing reference set. For removal, IB3 uses a statistical test to determine which instances have a classification performance which is significantly poor; these are discarded.

PECS [18] takes a similar approach to IB3. The main differences, apart from the particular statistical methods, are that PECS immediately includes all new examples in the prototype set, and that it never truly deletes examples, only inactivates them (PECS is therefore not strictly a streaming algorithm, although it can handle concept drift). It is interesting that authors of PECS converged to this IB3-like approach, despite the algorithm being developed from the conceptually different LWF algorithm [19]. The LWF algorithm removes prototypes when they have been superseded by newer prototypes in the same region of the feature space. This is the reason for our "2/4" notation for this algorithm in Table 1: we seek to indicate that the age

Table 1: Prototype selection methods for streaming data with concept drift, described within the taxonomy of Fig. 2.

Method	Change detection	Approach			Data management
		PG	PS		
			+	−	
1NN	N		1	2	O
PECS	N		1	1	F
SimC	N		1	1	O
LWF	N		1	2/4	O
COMPOSE	N		1	4	B
AES	Y		1	2, 5, 6	O
IBL-DS	Y		1	2, 3, 5	O
IBLStreams	Y		1	2, 3, 5	O
IB3	N		2	1	O
Lu et al.	Y		3	1	B
SyncStream	Y	3	1	1	O
ILVQ	N	2			O
oiGRLVQ	N	2			O
ANNCAD*	N	1			O

Notes:

PG: Prototype generation: (1) Prototypes are regular grid in space; edit by relabelling; (2) Reposition; (3) Prototypes not evolved.

PS (+): Prototype selection (Prototype Addition): (1) Add all; (2) Add misclassified; (3) Add by a competence criterion.

PS (−): Prototype selection (Prototype Removal): (1) Remove poorly predicting; (2) Remove oldest; (3) Remove a random sample; (4) Remove by a geometric criterion; (5) Remove if misclassified (traditional editing); (6) Remove if correctly classified by many neighbours (condensing).

O/B/F : Online / Batch / Offline

* Ensemble method, rather than single classifier.

criterion is intertwined with a geometric criterion in this case, as opposed to age and geometric criteria being separately implemented.

In IBL-DS [20] and IBLStreams [21], the key mechanism for prototype removal is to remove prototypes which are misclassified according to other nearby prototypes, as in Wilson editing [22]. In these algorithms, if the class of a newly-arrived example is the same as the class of the majority of the youngest examples in that neighbourhood, then older examples of the opposite class in that neighbourhood are removed. There is also provision to remove prototypes purely based on age, to guarantee an upper bound on the reference set size. A mechanism to delete prototypes pseudorandomly is triggered when abrupt concept change is detected.

The AES [23] algorithm is presented in terms of an extended analogy with endocrine systems, which turns out in practice to mean a distance-weighted voting scheme. The location of the "cell" (reference prototype) with highest "hormone concentration" (the sum of the votes) is moved to the location of the newly-arrived example. In our way of thinking about maintaining a reference set of selected prototypes, this is equivalent to adding all new examples and deleting, upon the arrival of each new example, the example most strongly in agreement with its neighbours (as in traditional condensing).

The COMPOSE algorithm [24] is based on an idea that is mathematically elegant, but computationally very expensive: the set of reference prototypes for each class is thinned by removing examples associated with simplices which contain the outer faces of an α-shape[4] which defines the class boundary.

SimC [25] internally separates the reference instances of each class into a number of separate sets corresponding to different regions of the search space. New examples are added to the nearest suitable set, or are used to start a new set if no nearby set is suitable. Poorly performing examples are removed as new examples are added to their sets. Through this splitting of the reference set into multiple sub-sets with their own update criteria, SimC aims to achieve good "spatial and temporal relevance".

5.2. LVQ family

Kohonen [26] developed the Learning Vector Quantization technique, a seminal example of prototype generation. Many derivative algorithms have been developed over the years, but it is only relatively recently that LVQ-style

[4] α-shapes are a generalisation of the concept of a convex hull.

algorithms have begun to be designed for streaming applications.

In the oiGRLVQ algorithm [27], new examples are added to the reference set if they are sufficiently far from any existing prototype, according to the similarity measure which the algorithm uses. A pruning step removes a proportion of the prototypes in each class, removing those which have least often been the closest prototype to a new exemplar.

ILVQ [28] also adds new examples to the reference set when they are distant from existing elements of the reference set, and removes prototypes which are little-used in classification.

5.3. Non-LVQ generated-prototype methods

ANNCAD [29] discretises the feature space into a grid, an approach whose equivalence to a nearest-instance method may not be immediately obvious. However, the approach is equivalent to using the centre of each hyper-rectangular cell of the discretisation as a generated prototype. Note, though, that ANNCAD does not perform a true nearest-neighbour classification: to reduce computational demands, only those neighbours are queried which are also nearby branches in the tree structure which describes the various resolutions of the discretisation. ANNCAD is of further interest as the only exemplar of an ensemble approach in our selection. (The ensembling in the case of ANNCAD is baked into the instance handling algorithm: this is in contrast to the more common case where a base classifier can optionally be incorporated into an ensemble, or not.) ANNCAD uses a small ensemble of offset grids, to mitigate the problem of the neighbourhood relation being strongly dependent on the quantisation process.

SyncStream [15] is interesting in its use of two levels of reference set data. Strongly-performing instances are retained in one level; poorly-performing instances are deleted; and instances of mediocre performance are summarised by a clustering technique into a smaller number of generated prototypes. Hence, SyncStream is categorised both according to prototype selection and prototype generation techniques in Table 1, and is further unusual in that it makes use of generated prototypes but does not evolve them; prototypes once generated are preserved until their deletion, like selected prototypes.

5.4. Case-based reasoning family

We include in Table 1 the "concept drift-tolerant case-base editing technique" of Lu et al. [30], as a recent and generally applicable example of the "case-based reasoning" approach. This algorithm is designed for the concept-drift problem, and is interesting in that it seeks to add new prototypes in areas of the feature space where it believes concept drift to be taking place. For this algorithm, reference set management is inextricably bound up with change detection. (This algorithm takes the place of certain older case-based algorithms with a narrow application to spam filtering which we considered in our previous study [5].)

6. Experimental comparison

Clearly, a comprehensive experimental comparison demonstrating the strengths and weaknesses of all categories in the proposed taxonomy is infeasible. Our present experiment serves as an illustration; we are cautious of making strong claims. Nonetheless, we believe that this experiment can give useful insights. We may be able to identify traits in the taxonomy which have an effect on performance (for example, are methods which use change detection better/worse than methods which do not?) and we may be able to identify promising avenues for future research on that basis.

We performed three sets of sub-experiments. The first uses synthetic data sets, with no added noise: the results and discussion for these sub-experiments are given in subsection 6.4.2. The second uses synthetic data with various levels of noise added using a setting in the MOA system: the results and discussion are in subsection 6.4.3. The third uses real-world data, with concept drift introduced using a synthetic method: the results and discussion are in subsection 6.5. We draw our key insights from the first of these sub-experiments, in our discussion in 6.4.2. The principal function of the remaining two sub-experiments is to show that the results found in that section do not vanish in the presence of noise or for real-world data, though there is some further interest in these results.

6.1. Framework

The framework selected for the experiments was the Massive Online Analysis (MOA) system[5], version 2016.04. We created our own, non-optimised implementation of the following algorithms: ANNCAD [29]; PECS [18]; IB3 [17]; LWF [19]; and oiGRLVQ [27]. The algorithm IBLStreams [21]

[5] Available at `http://moa.cms.waikato.ac.nz/`

was already available in MOA[6]. A wrapper was necessary for the adaptation of the publicly available code of SimC[7] [25] and SyncStream[8] [15] to MOA. We have made all the implemented methods available online for general use: `https://github.com/alvarag/ConceptDriftMOA`.

6.2. Algorithms and parameters

A sliding window of 2 000 instances (with the nearest-neighbour classifier (1NN)) was selected as the baseline method. The list of all algorithms and parameter settings included in the experimental comparison is as follows:

- 1NN: sliding window of 2 000 instances.
- LWF: $\theta = 0.05$, $\beta = 0.04$, and $\tau = 0.08$.
- IB3: conf. accept. = 0.8 and conf. drop = 0.05.
- PECS: $p_{min} = 0.3$, $p_{max} = 0.7$, and $\beta = 0.4$.
- ANNCAD: ensemble size = 4, $\lambda = 0.98$, and shift = 0.1.
- SimC: default options.
- SyncStream: "statistical analysis" strategy used[9]; $\theta = 60$.
- IBLStreams: without adaptation of the size k of the neighbourhood, i.e. $k = 1$ always.
- oiGRLVQ: reduction = 20%, # mini-batch = 500.

We have kept most of the parameter settings close to the ones suggested by the creators of the respective algorithms, but it was necessary to alter some of the values in order to ensure that all of the algorithms stored a similar number of prototypes, around 2 000. The set of algorithms chosen for the experiment is necessarily only representative. It is not practical to consider all instance-based algorithms with all possible parameter values.

[6] Minor changes were made to run the algorithm in the current version of MOA. The original code is available at: `https://www-old.cs.uni-paderborn.de/fachgebiete/intelligente-systeme/software/iblstreams-moa-extension.html`.

[7] Available at `https://www.dropbox.com/s/s2t2ogaki1x1n4w/Weka.rar?dl=0`.

[8] Available at `https://github.com/kramerlab/SyncStream/`.

[9] The "statistical analysis" strategy is one of two options for concept-drift detection the authors of SyncStream propose for use with their algorithm [15].

6.3. Experimental setup

The performance of the different algorithms was evaluated by the technique known as *Interleaved Test-Then-Train* or *Prequential*: the model is tested with each incoming example and subsequently the new example is used for training [31]. A benefit of this common technique is that it makes maximum use of the available data [6]. We used the sliding-window version of the Prequential technique (with a window of width 100), as opposed to the main alternative of using a fade factor. The accuracy statistic was calculated and stored after every 100 instances: each accuracy value recorded is the percentage of instances correctly classified in the previous 100 instances.

6.4. Experiments using synthetic data sets

We wished to evaluate the behaviour of the algorithms in a controlled setting suitable as a benchmark. For this purpose, we used two popular [6, 30] generators in MOA, with and without concept drift. The key benefit of using a generator is its reproducibility: the stream can be replicated in MOA (or other software) directly.

6.4.1. Data sets used

All the sub-experiments involved binary classification problems. Except where noted, default options for the generators were adopted. Only numeric attributes were used because some of the algorithms cannot deal with nominal features. The two generators used were:

- Hyperplane: first used in [32]. The data stream is generated uniformly in a unit hypercube in a d-dimensional space (each dimension is a feature). A hyperplane is constructed to split the data into two classes and serve as the boundary. Concept drift is generated by varying the position of the plane with time. In our experiments, the feature space was defined by ten numeric attributes ($d = 10$).

- Random tree: the class labels are determined by means of a decision tree prepared in advance. The tree is constructed by choosing random attributes to split the space; a random class label is then assigned to each leaf. The random tree was generated with five numerical attributes, and a maximum depth of five was allowed. The same attribute may be picked more than once in the decision process.

Table 2: Settings used with the MOA generators.

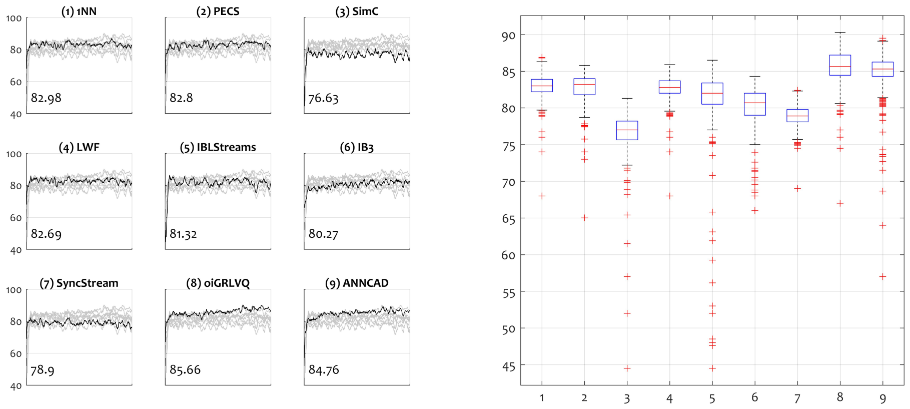
Generator	Seed	Concept drift		
		Position	Width	Seed
Hyperplane (no CD)	1	-	-	-
Hyperplane (gradual CD)	1	25 000	1 000	5
Hyperplane (abrupt CD)	1	25 000	1	5
Random tree (no CD)	1	-	-	-
Random tree (gradual CD)	1	25 000	1 000	5
Random tree (abrupt CD)	1	25 000	1	5

Time series of 50 000 instances were generated, with the concept drift introduced in the middle. Two different types of concept drift were tested, sudden and gradual. For the hyperplane generator, two versions of the experiment were performed: without and with noise added to the data. Where used, noise was added directly using the option provided in the MOA framework for the Hyperplane generator. The experiments without noise are discussed in subsection 6.4.2; the experiments with added noise are discussed in subsection 6.4.3.

The configuration of the generators, including seed values, is shown in Table 2.

6.4.2. Hyperplane and random tree generators without noise: results and discussion

Six sub-experiments were performed for synthetic data without the addition of noise: Hyperplane with (1) no concept drift, (2) gradual change, and (3) abrupt change; and Random tree with (4) no concept drift, (5) gradual change, and (6) abrupt change. The results for these six data streams are shown in Figs 3–8 respectively. Each of these figures shows accuracy as a function of time for all nine methods. In the (a) subfigures, the accuracies of all methods are plotted in grey, and each method is highlighted in black in its own sub-plot. This view allows for an instant evaluation of the success of the highlighted method. If the black curve runs higher than the grey curves, the method outperforms its competitors. The average accuracy for a method across the whole run is shown in the bottom-left corner of its subplot. To the right of these accuracy plots, in the (b) subfigures, we show boxplots for each of the nine methods, showing the distribution of the values of the accuracy



(a) Evolution of accuracy

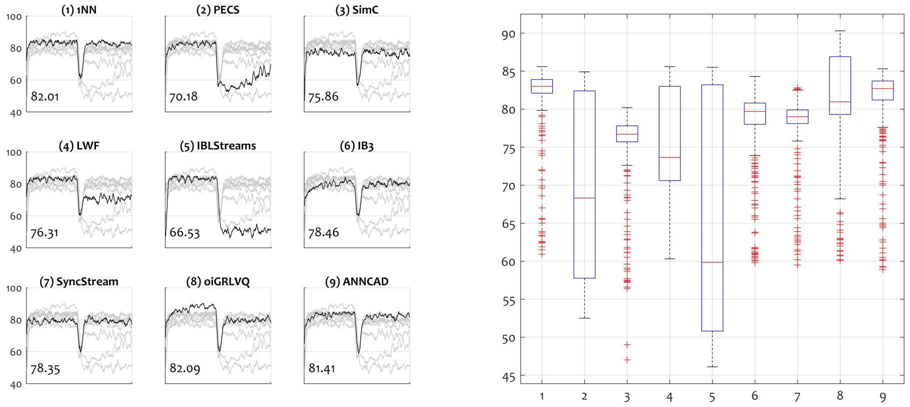
(b) Boxplot

Figure 3: Hyperplane generator: accuracy (in %) of the algorithms **without** concept drift. The evolution of the accuracy statistic is shown in subfigure (a), and the distribution of the instantaneous accuracy values for each algorithm is shown by a boxplot in subfigure (b). The average accuracy is shown in the bottom-left corner of each sub-plot in (a).

statistic calculated throughout the run. This gives further information about the stability of the methods' performance.

Tables 3 and 4 show respectively the means and the medians of the methods for the 6 sub-experiments. Each column represents one sub-experiment; they are grouped by the generator used and distinguished by the type or absence ("–") of concept drift. The best value of each column is highlighted in bold.

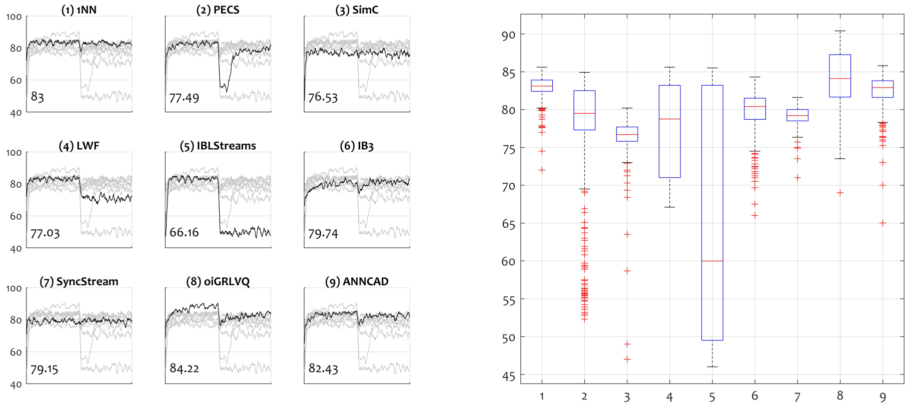
With these results in hand, we will now turn to address the possible insights we mentioned at the beginning of this section. First, a note of warning about the limitations on what can be inferred: we have so far (cf. following sections) used only artificial data generated from MOA, we restricted the number of prototypes to about 2 000, and we enforced the use of the 1NN rule for classification (in particular, no adaptation of k was allowed for IBLStreams, which uses a variable-neighbourhood kNN classification rule by design). Any of these choices may disadvantage some methods. Further, none of our datasets involved recurring concepts; hence, methods designed to detect recurring concepts do not appear here to their best advantage. For these reasons and more, our findings do not by any means invalidate the



(a) Evolution of accuracy

(b) Boxplot

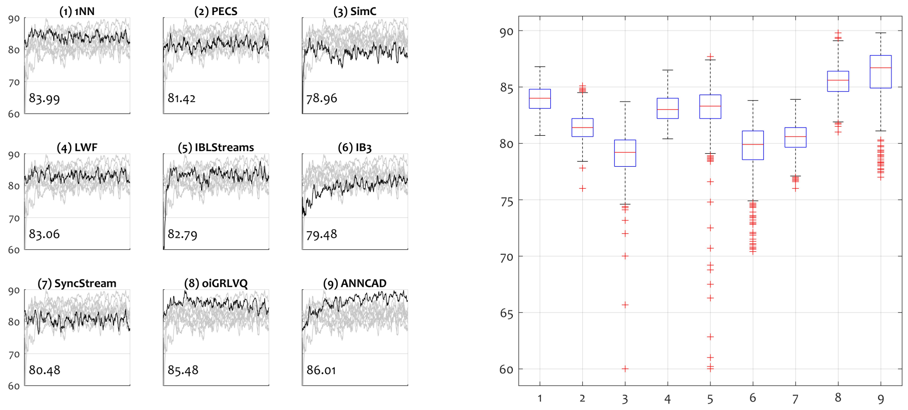
Figure 4: Hyperplane generator: accuracy (in %) of the algorithms with **gradual** concept drift. The average accuracy is shown in the bottom-left corner of each sub-plot in (a).



(a) Evolution of accuracy

(b) Boxplot

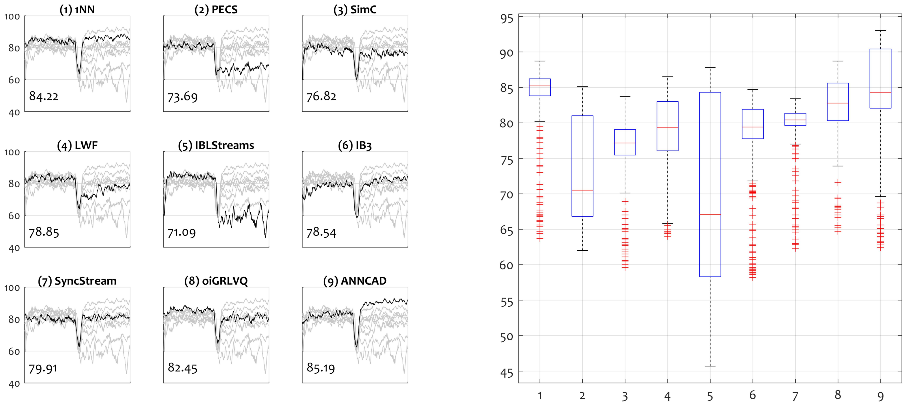
Figure 5: Hyperplane generator: accuracy (in %) of the algorithms with **abrupt** concept drift. The average accuracy is shown in the bottom-left corner of each sub-plot in (a).



(a) Evolution of accuracy

(b) Boxplot

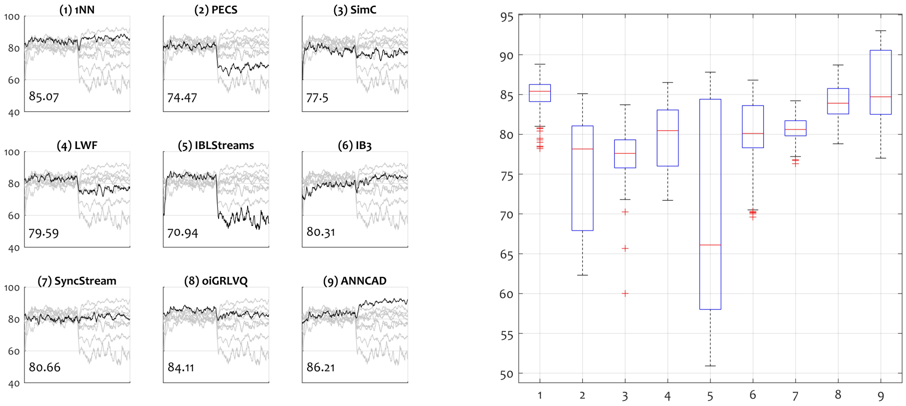
Figure 6: Random tree generator: accuracy (in %) of the algorithms **without** concept drift. The average accuracy is shown in the bottom-left corner of each sub-plot in (a).



(a) Evolution of accuracy

(b) Boxplot

Figure 7: Random tree generator: accuracy (in %) of the algorithms with **gradual** concept drift. The average accuracy is shown in the bottom-left corner of each sub-plot in (a).



(a) Evolution of accuracy

(b) Boxplot

Figure 8: Random tree generator: accuracy (in %) of the algorithms with **abrupt** concept drift. The average accuracy is shown in the bottom-left corner of each sub-plot in (a).

Table 3: Mean accuracy of the methods for the 6 sub-experiments using synthetic data without noise. Columns with a dash "–" in the header are for experiments with no concept drift. The best value of each column is highlighted in bold.

Method	Hyperplane			Random tree		
	–	gradual	abrupt	–	gradual	abrupt
1NN	82.98	82.01	83.00	83.99	84.22	85.07
PECS	82.80	70.18	77.49	81.42	73.69	74.47
SimC	76.63	75.86	76.53	78.96	76.82	77.50
LWF	82.69	76.31	77.03	83.06	78.85	79.59
IBLStreams	81.32	66.53	66.16	82.79	71.09	70.94
IB3	80.27	78.46	79.74	79.48	78.54	80.31
SyncStream	78.90	78.35	79.15	80.48	79.91	80.66
oiGRLVQ	**85.66**	**82.09**	**84.22**	85.48	82.45	84.11
ANNCAD	84.76	81.41	82.43	**86.01**	**85.19**	**86.21**

Table 4: Median accuracy of the methods for the 6 sub-experiments using synthetic data without noise. Columns with a dash "–" in the header are for experiments with no concept drift. The best value of each column is highlighted in bold.

Method	Hyperplane			Random tree		
	–	gradual	abrupt	–	gradual	abrupt
1NN	83.00	**83.00**	83.10	84.00	**85.20**	**85.40**
PECS	83.20	68.30	79.50	81.40	70.50	78.15
SimC	77.00	76.70	76.70	79.20	77.15	77.60
LWF	82.80	73.65	78.75	83.00	79.30	80.45
IBLStreams	82.00	59.84	59.99	83.30	67.05	66.09
IB3	80.70	79.70	80.40	79.90	79.40	80.10
SyncStream	78.90	79.00	79.20	80.60	80.40	80.60
oiGRLVQ	**85.65**	80.95	**84.10**	85.60	82.78	83.90
ANNCAD	85.30	82.70	82.90	**86.70**	84.30	84.70

methods which did not work well in our chosen setting.

We see from the plots that PECS, LWF and IBLStreams had difficulties recovering from either gradual or abrupt change. This can alternatively be seen from Tables 3 and 4. In contrast, SimC and SyncStream showed good resilience to concept drift, but at the expense of poor overall accuracy. A combination of resilience to drift and good overall accuracy was displayed by oiGRLVQ, by ANNCAD, and by the benchmark method, simple windowed 1NN.

Following previous practice [3, 4], we visualise the methods' performance in relation to the taxonomic properties by plotting each property with a different marker: see Fig. 9. The marker shape indicates whether or not the method uses explicit change detection, and the filling distinguishes between prototype selection and prototype generation methods. The 9 methods are shown in the space of $x = (H_g + H_a)/2$ and $y = (R_g + R_a)/2$ where H_g is the median accuracy of the method for the hyperplane data with gradual change (sub-experiment 2), H_a is the median accuracy for the hyperplane data with abrupt change (sub-experiment 3), and R_g and R_a are the equivalents for the Random tree data (sub-experiments 5 and 6).

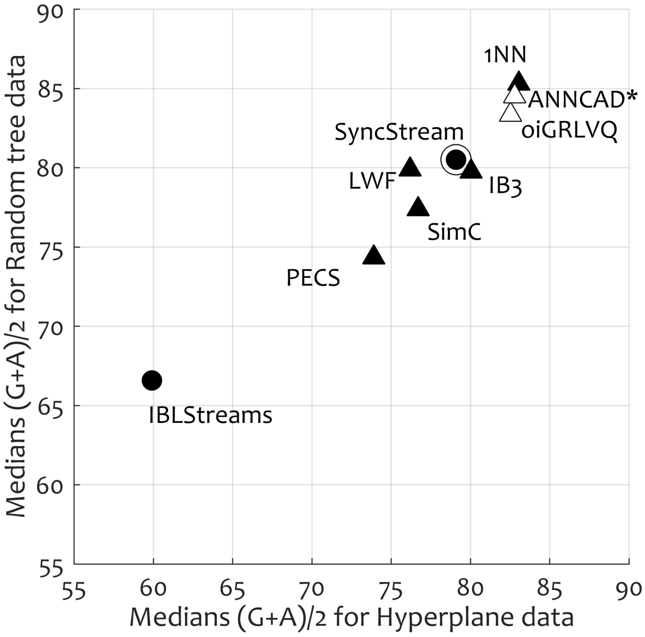


Figure 9: Scatterplot of the 9 methods in the space of median accuracies for the two synthetic data sets without noise. The significance of the markers is as follows: shape indicates whether or not the method uses explicit change detection: circle ○ – yes, triangle △ – no; and a filled shape indicates a prototype selection method while an empty shape indicates a prototype generation method. The filled marker for SyncStream is circled to indicate that this method belongs to both the prototype selection and prototype generation categories. The ensemble approach, ANNCAD, is indicated with an asterisk.

The results from this experiment favour prototype generation methods. This might be said to be expected on theoretical grounds, in that prototype generation can be seen as a generalisation of prototype selection, in which the instances are not limited to examples of the original data set. It might also be said that methods with no explicit change detector performed well, though few methods with explicit change detection were considered. Furthermore, the only ensemble-like method ANNCAD performed well, suggesting that an ensembling of nearest neighbour classifiers for streaming data could be an interesting research avenue. It should be noted that ANNCAD might be at an advantage when classifying the Random tree data, by virtue of using a tree structure internally. However, this is not the case for the hyperplane data, for which ANNCAD performs very nearly as strongly relative to the other algorithms. Conversely, it might have been thought that the hyperplane data would favour windowed 1NN, but it performs at least as well on the random tree data. The particularly poor performance of IBLStreams is most likely explained by its being particularly disadvantaged by our choice to impose the use of the 1NN classifier.

Perhaps the most exciting (though not entirely unexpected!) finding in this paper is the excellent performance of the baseline method: 1NN with a simple fixed-width sliding window. With no concept drift, the sliding window works on a par with the best of the other methods. However, when concept drift occurs, the sliding window gives high accuracy with a low variability, as evidenced by the boxplots in Figs 4, 5, 7, and 8. It also appears as the best method in Fig. 9.

The good performance of simple methods often raises the question of "value for money". Sophisticated designs are only justified if they demonstrably outperform the simple methods, both in terms of accuracy and consistency. Otherwise, practitioners will not have sufficient reason to opt for a complicated design. The humble windowed 1NN algorithm still poses a formidable challenge to those who would design new algorithms in this area, a challenge which must not be ignored.

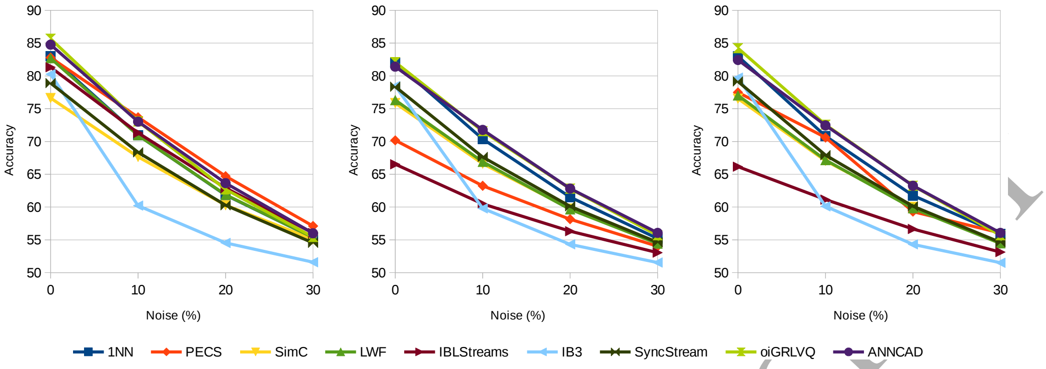
That being said, we must emphasise that our conclusions are based on limited experiments. We have not considered multi-class problems, imbalanced data sets, data sets with categorical features, and so on. There are many possible areas in which more-sophisticated algorithms may prove their worth!

Table 5: Mean accuracy of the methods for the 9 sub-experiments using synthetic data with **noise**. Columns with a dash "–" in the header are for experiments with no concept drift. The best value of each column is highlighted in bold.

Method	10% noise			20% noise			30% noise		
	–	gradual	abrupt	–	gradual	abrupt	–	gradual	abrupt
1NN	70.96	70.38	70.83	61.84	61.52	61.76	55.34	55.20	55.19
PECS	**73.70**	63.24	70.61	**64.70**	58.15	59.31	**57.11**	54.01	56.05
SimC	67.66	66.62	67.05	60.29	59.76	60.04	55.08	54.48	54.48
LWF	70.95	66.90	67.17	61.86	59.65	59.80	55.37	54.43	54.46
IBLStreams	71.31	60.49	61.14	62.90	56.32	56.66	56.05	53.05	53.14
IB3	60.22	59.79	60.10	54.53	54.32	54.31	51.59	51.54	51.51
SyncStream	68.35	67.64	67.92	60.27	60.12	60.13	54.54	54.62	54.42
oiGRLVQ	73.06	71.57	**72.56**	62.69	62.81	63.25	55.43	55.70	55.74
ANNCAD	73.03	**71.77**	72.45	63.65	**62.82**	**63.25**	56.00	**56.06**	**56.18**

Table 6: Median accuracy of the methods for the 9 sub-experiments using synthetic data with **noise**. Columns with a dash "–" in the header are for experiments with no concept drift. The best value of each column is highlighted in bold.

Method	10% noise			20% noise			30% noise		
	–	gradual	abrupt	–	gradual	abrupt	–	gradual	abrupt
1NN	71.00	71.00	71.10	61.90	61.70	61.70	55.30	55.40	55.30
PECS	**73.70**	59.75	71.35	**64.70**	56.30	60.00	**57.20**	54.10	**56.50**
SimC	67.80	67.20	67.30	60.30	60.10	60.20	54.80	54.50	54.40
LWF	71.00	66.50	66.90	61.90	59.85	59.80	55.45	54.60	54.50
IBLStreams	71.80	54.91	55.16	63.30	54.65	55.48	56.10	53.00	52.90
IB3	60.25	60.00	60.10	54.40	54.30	54.40	51.55	51.70	51.60
SyncStream	68.40	68.00	68.10	60.30	60.40	60.30	54.70	54.50	54.50
oiGRLVQ	73.20	71.10	72.05	62.70	62.60	62.90	55.50	55.70	55.77
ANNCAD	73.30	**72.55**	**72.60**	63.90	**63.60**	**63.65**	56.00	**56.30**	56.20



(a) No concept drift (b) Gradual concept drift (c) Abrupt concept drift

Figure 10: Accuracy of the different classifiers with and without noise on the hyperplane generator.

6.4.3. Hyperplane generator with noise: results and discussion

Real-life data sets are not perfect: outliers and noise are frequently present, negatively impacting the prediction capabilities of classifiers [33]. Moreover, instance-based classifiers (like kNN, and especially 1NN) are usually more sensitive to noise than other approaches. For these reasons, we performed further experiments in which noise was added to the synthetic data using a tool provided in the MOA framework. The hyperplane generator was used with various levels of added noise: 10%, 20%, and 30%.

Tables 5 and 6 show respectively the means and the medians of the methods for the 9 sub-experiments (each column represents one sub-experiment). There is one sub-experiment for each noise level and, as previously, for each type of concept drift: no drift; gradual; and abrupt drift. The best value of each column is highlighted in bold.

Figure 10 shows the performance of the various classifiers in the presence of noise, under our three concept-drift scenarios. The IB3 algorithm immediately stands out as having the worst tolerance to noise, with or without concept drift. On the other hand, ANNCAD and oiGRLVQ continue their good showing from the noise-free case. Again, and more surprisingly in this case, windowed 1NN outperforms many of the more sophisticated classifiers.

6.5. Real-world data sets

For a more complete comparison of the algorithms, we have performed a similar set of experiments on some real-world data sets, into which we

Table 7: Description of data sets used for experiments in section 6.5: name; number of binary and numeric features; and number of instances.

Data set	# attributes		Size of the stream
	Binary	Numeric	
Electricity	7	7	45 312
Forest Covertype	44	10	49 514
Poker-hand	20	5	50 000

have introduced concept drift using the method proposed by Shaker and Hüllermeier [34].

We selected three popular data sets available on the MOA webpage[10]: Electricity, Forest Covertype, and Poker-Hand. Each of these has been used in several publications concerning learning from streams [15, 25, 29, 35, 36]. These data sets are highly imbalanced, have several classes and are relatively large. Moreover, some of their attributes are nominal, and hence cannot be handled by all of the algorithms in our selection. Therefore, we preprocessed the data sets to make them suitable for the experiment: the nominal attributes were converted into sets of binary attributes, only instances belonging to one of the two majority classes were selected, and random undersampling was performed to reduce the size of the sets. Table 7 describes the main characteristics of the processed data sets. In the relevant sub-experiments, concept drift was introduced starting at the 25 000th instance, with the random seed set to 1. As in the experiments with synthetic data sets, the gradual drift had a width of 1 000 and the abrupt drift had a width of 1.

Following Shaker and Hüllermeier [34], a drifting stream was created from each processed data set along the following lines: the original set was split, and one of the subsets is used without change; the drift sample is created by means of inverting the class of the other subset.

The results of the experiments on these data sets are summarized in the Tables 8 and 9. As for the previous tables for the experiments with synthetic data, these tables show the means and the medians of the methods, respectively, for each of the 9 sub-experiments (three data sets and three

[10] Available at `https://moa.cms.waikato.ac.nz/datasets/`.

Table 8: Mean accuracy of the methods for the 9 sub-experiments on real-world data. Columns with a dash "–" in the header are for experiments with no concept drift. The best value of each column is highlighted in bold.

Method	Forest Covertype			Electricity			Poker-hand		
	–	gradual	abrupt	–	gradual	abrupt	–	gradual	abrupt
1NN	**98.26**	**97.64**	**98.23**	86.37	86.01	86.42	71.45	70.59	70.55
PECS	96.28	95.70	96.13	82.96	83.10	83.30	74.92	**74.59**	**74.72**
SimC	97.45	96.82	97.49	**90.13**	**89.67**	**90.20**	71.58	70.86	70.88
LWF	97.66	96.94	97.65	86.32	86.27	86.69	71.92	73.25	73.51
IBLStreams	96.97	92.12	93.12	86.67	72.80	73.42	73.88	66.78	67.05
IB3	95.61	94.58	95.57	70.91	71.71	71.57	71.18	70.38	70.64
SyncStream	98.13	97.55	98.13	81.35	81.07	81.33	71.59	70.39	70.36
oiGRLVQ	96.21	95.43	96.00	72.72	72.21	72.15	68.00	65.75	66.68
ANNCAD	96.86	96.93	97.49	77.38	81.91	82.19	**75.19**	73.40	73.65

Table 9: Median accuracy of the methods for the 9 sub-experiments on real-world data. Columns with a dash "–" in the header are for experiments with no concept drift. The best value of each column is highlighted in bold.

Method	Forest Covertype			Electricity			Poker-hand		
	–	gradual	abrupt	–	gradual	abrupt	–	gradual	abrupt
1NN	**98.30**	**98.20**	**98.30**	86.20	86.30	86.50	71.80	70.40	70.30
PECS	96.00	95.80	95.80	83.00	83.40	83.40	74.55	**74.30**	**74.20**
SimC	97.50	97.40	97.55	**90.30**	**90.30**	**90.50**	71.90	71.05	71.05
LWF	97.70	97.30	97.60	86.30	86.60	86.80	71.80	73.40	73.95
IBLStreams	97.45	96.90	97.15	87.30	83.80	83.90	74.30	65.55	66.45
IB3	96.00	95.55	95.80	70.30	71.20	71.00	71.25	70.70	70.95
SyncStream	98.20	98.10	98.20	81.80	81.62	81.80	71.90	70.60	70.60
oiGRLVQ	97.00	96.90	96.90	75.40	72.40	73.40	67.50	64.95	66.10
ANNCAD	97.00	97.40	97.40	77.60	82.30	82.40	**75.05**	73.75	73.90

configurations for each).

For the real-world data, the question of which algorithm is best strongly depends on the data set. The baseline windowed 1NN classifier was the best on Forest Covertype, and SimC was the best on Electricity, for all three subexperiments (with and without concept drift). For the Poker-hand data set, when drift was present the PECS classifier was the best, whereas when drift was not present ANNCAD achieved the best results. This is a remarkable contrast to the performance of PECS on the noisy synthetic data, where PECS was the best-performing algorithm in the absence of drift, but its performance strongly degraded in the presence of gradual drift. The overall poor performance of all the methods on the poker-hand data is explained by the complex and abstract nature of the relationship between a poker hand and its constituent cards: the learning task in this case really calls for some sort of symbolic rule extraction, rather than usual type of machine-learning techniques based on delineating regions in a feature space.

7. Conclusion

We have presented an augmented and refined taxonomy for nearest-neighbour methods for the classification of data streams subject to concept drift. A numerical experiment, interpreted in the light of the new taxonomy, gave the following insights.

For the problem of classifying data streams subject to concept drift, methods based on prototype *generation* may be a more promising avenue for future research than further refinements of the many prototype *selection* methods which have been proposed. The theoretical observation that prototype generation methods draw from a richer hypothesis space than prototype selection methods is borne out, by our limited experiment, as greater practical performance by prototype generation algorithms: see Fig. 9.

We regard the strikingly strong performance of the simple windowed 1NN baseline method as a valuable result. We strongly recommend that experimental evaluations of streaming classification algorithms proposed in the future should include this simple baseline as a point of reference. Further, when choosing state-of-the-art algorithms to which to compare the performance of proposed new instance-based algorithms, the algorithms ANNCAD [29] and oiGRLVQ [27] might be valuable additions to the pool of standard algorithms.

Acknowledgement

This work was done under project RPG-2015-188 funded by The Leverhulme Trust, UK, and TIN 2015-67534-P from the Spanish Ministry of Economy and Competitiveness. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731593.

References

[1] Q. Yang, X. Wu, 10 challenging problems in data mining research, International Journal of Information Technology & Decision Making 05 (04) (2006) 597–604. doi:10.1142/S0219622006002258.

[2] B. Krawczyk, J. Stefanowski, M. Wozniak, Data stream classification and big data analytics, Neurocomputing 150, Part A (2015) 238 – 239, selected papers from the 16th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2012), Selected papers from the 6th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA 2013). doi:10.1016/j.neucom.2014.10.025.

[3] S. Garcia, J. Derrac, J. R. Cano, F. Herrera, Prototype selection for nearest neighbor classification: Taxonomy and empirical study, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (3) (2012) 417–435. doi:10.1109/TPAMI.2011.142.

[4] I. Triguero, J. Derrac, S. Garcia, F. Herrera, A taxonomy and experimental study on prototype generation for nearest neighbor classification, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42 (1) (2012) 86–100. doi:10.1109/TSMCC.2010.2103939.

[5] L. I. Kuncheva, I. A. D. Gunn, A concept-drift perspective on prototype selection and generation, in: 2016 International Joint Conference on Neural Networks (IJCNN), 2016, pp. 16–23. doi:10.1109/IJCNN.2016.7727175.

[6] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (4) (2014) 44:1–44:37. doi:10.1145/2523813.

[7] D. Nova, P. A. Estévez, A review of learning vector quantization classifiers, Neural Computing and Applications 25 (3) (2014) 511–524. doi:10.1007/s00521-013-1535-3.

[8] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, F. Herrera, A survey on data preprocessing for data stream mining: Current status and future directions, Neurocomputing (2017) – doi:10.1016/j.neucom.2017.01.078.

[9] R. P. J. C. Bose, W. M. P. van der Aalst, I. Žliobaitė, M. Pechenizkiy, Handling Concept Drift in Process Mining, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 391–405. doi:10.1007/978-3-642-21640-4_30.

[10] G. Widmer, M. Kubat, Machine Learning: ECML-93: European Conference on Machine Learning Vienna, Austria, April 5–7, 1993 Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 1993, Ch. Effective learning in dynamic environments by explicit context tracking, pp. 227–243. doi:10.1007/3-540-56602-3_139.

[11] M. Kubat, Floating approximation in time-varying knowledge bases, Pattern Recognition Letters 10 (4) (1989) 223 – 227. doi:10.1016/0167-8655(89)90092-5.

[12] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Machine Learning 23 (1) (1996) 69–101. doi:10.1007/BF00116900.

[13] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, F. Petitjean, Characterizing concept drift, Data Mining and Knowledge Discovery 30 (4) (2016) 964–994. doi:10.1007/s10618-015-0448-4.

[14] H. M. Gomes, J. P. Barddal, F. Enembreck, A. Bifet, A survey on ensemble learning for data stream classification, ACM Comput. Surv. 50 (2) (2017) 23:1–23:36. doi:10.1145/3054925.

[15] J. Shao, Z. Ahmadi, S. Kramer, Prototype-based learning on concept-drifting data streams, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, ACM, New York, NY, USA, 2014, pp. 412–421. doi:10.1145/2623330.2623609.

[16] N. García-Pedrajas, D. Ortiz-Boyer, Boosting k-nearest neighbor classifier by means of input space projection, Expert Systems with Applications 36 (7) (2009) 10570 – 10582. doi:10.1016/j.eswa.2009.02.065.

[17] D. W. Aha, D. Kibler, M. K. Albert, Instance-based learning algorithms, Machine Learning 6 (1) (1991) 37–66. doi:10.1007/BF00153759.

[18] M. Salganicoff, Tolerating concept and sampling shift in lazy learning using prediction error context switching (1997) 133–155doi:10.1007/978-94-017-2053-3_5.

[19] M. Salganicoff, Density-adaptive learning and forgetting, in: Proceedings of the Fifth International Conference on Machine Learning, Morgan Kaufmann, Amherst, MA, 1993, pp. 276–283.

[20] J. Beringer, E. Hüllermeier, Efficient instance-based learning on data streams, Intelligent Data Analysis 11 (6) (2007) 627–650.

[21] A. Shaker, E. Hüllermeier, IBLStreams: a system for instance-based classification and regression on data streams, Evolving Systems 3 (4) (2012) 235–249. doi:10.1007/s12530-012-9059-0.

[22] D. L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transactions on Systems, Man, and Cybernetics SMC-2 (3) (1972) 408–421. doi:10.1109/TSMC.1972.4309137.

[23] L. Zhao, L. Wang, Q. Xu, Data stream classification with artificial endocrine system, Applied Intelligence 37 (3) (2012) 390–404. doi:10.1007/s10489-011-0334-8.

[24] K. B. Dyer, R. Capo, R. Polikar, COMPOSE: A semisupervised learning framework for initially labeled nonstationary streaming data, IEEE Transactions on Neural Networks and Learning Systems 25 (1) (2014) 12–26. doi:10.1109/TNNLS.2013.2277712.

[25] D. Mena-Torres, J. S. Aguilar-Ruiz, A similarity-based approach for data stream classification, Expert Systems with Applications 41 (9) (2014) 4224 – 4234. doi:10.1016/j.eswa.2013.12.041.

[26] T. Kohonen, The self-organizing map, Proceedings of the IEEE 78 (9) (1990) 1464–1480. doi:10.1109/5.58325.

[27] I. Cruz-Vega, H. J. Escalante, An online and incremental grlvq algorithm for prototype generation based on granular computing, Soft Computing 21 (14) (2017) 3931–3944. doi:10.1007/s00500-016-2042-0.

[28] Y. Xu, F. Shen, J. Zhao, An incremental learning vector quantization algorithm for pattern classification, Neural Computing and Applications 21 (6) (2012) 1205–1215. doi:10.1007/s00521-010-0511-4.

[29] Y.-N. Law, C. Zaniolo, An adaptive nearest neighbor classification algorithm for data streams, in: A. M. Jorge, L. Torgo, P. Brazdil, R. Camacho, J. Gama (Eds.), Knowledge Discovery in Databases: PKDD 2005: 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, Porto, Portugal, October 3-7, 2005. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 108–120. doi:10.1007/11564126_15.

[30] N. Lu, J. Lu, G. Zhang, R. L. de Mantaras, A concept drift-tolerant case-base editing technique, Artificial Intelligence 230 (2016) 108–133. doi:10.1016/j.artint.2015.09.009.

[31] J. Gama, R. Sebastião, P. P. Rodrigues, Issues in evaluation of stream learning algorithms, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, ACM, New York, NY, USA, 2009, pp. 329–338. doi:10.1145/1557019.1557060.

[32] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, ACM, New York, NY, USA, 2001, pp. 97–106. doi:10.1145/502512.502529.

[33] Álvar Arnaiz-González, J. F. Díez-Pastor, J. J. Rodríguez, C. I. García-Osorio, Instance selection for regression: Adapting DROP, Neurocomputing 201 (2016) 66 – 81. doi:10.1016/j.neucom.2016.04.003.

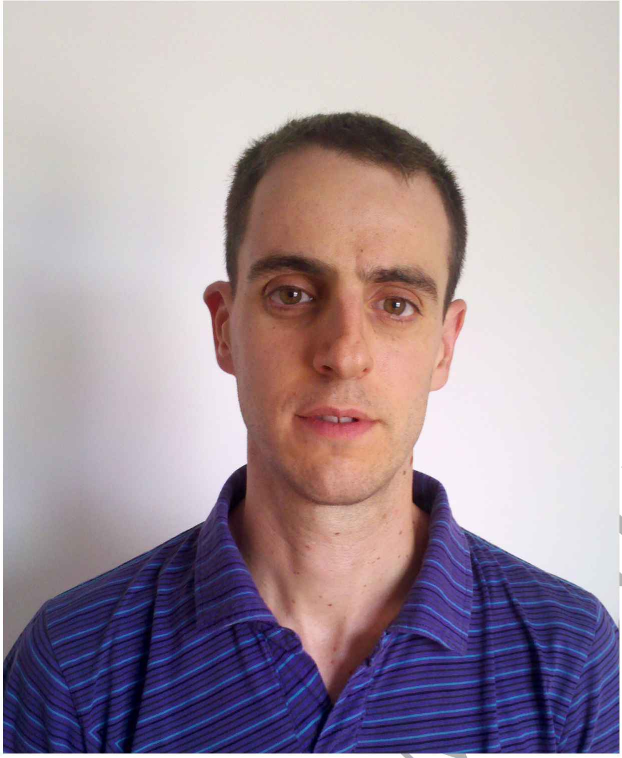
[34] A. Shaker, E. Hüllermeier, Recovery analysis for adaptive learning from non-stationary data streams: Experimental design and case study, Neurocomputing 150 (2015) 250 – 264, bioinspired and knowledge based techniques and applications The Vitality of Pattern Recognition and

Image Analysis Data Stream Classification and Big Data Analytics. doi:10.1016/j.neucom.2014.09.076.

[35] I. Žliobaitė, Combining similarity in time and space for training set formation under concept drift, Intell. Data Anal. 15 (4) (2011) 589–611. doi:10.3233/IDA-2011-0484.

[36] G. Ditzler, R. Polikar, Incremental learning of concept drift from streaming imbalanced data, IEEE Transactions on Knowledge and Data Engineering 25 (10) (2013) 2283–2301. doi:10.1109/TKDE.2012.136.

Iain Gunn is a Research Fellow at the Department of Computer Science, Middlesex University, London. He received the B.A., M.Math., M.Phil., and Ph.D. degrees from the University of Cambridge. He has research interests in machine learning and neural-symbolic integration.


Álvar Arnaiz González received his M.S. degree in Computer Engineering from the University of Burgos, in 2010. He is currently a Ph.D. candidate at the Department of Civil Engineering of Burgos University and he is a member of the Advanced Data Mining Research And Bioinformatics Learning group. His main research interests include Machine Learning, Data Mining, ensemble classifiers and instance selection.

Ludmila (Lucy) I. Kuncheva is a Professor of Computer Science at Bangor University, UK. Her interests include pattern recognition, and specifically classifier ensembles. She has published two monographs and over 200 research papers. Lucy has won two Best Paper

Awards (2006 IEEE TFS and 2003 IEEE TSMC.) She is a Fellow of IAPR and a Fellow of the Learned Society of Wales.