Applied Intelligence
https://doi.org/10.1007/s10489-022-03826-4

Processing data stream with chunk-similarity model selection

Pawel Ksieniewicz[1]

Accepted: 29 May 2022

Abstract
The classification of data stream susceptible to the concept drift phenomenon has been a field of intensive research for many years. One of the dominant strategies of the proposed solutions is the application of classifier ensembles with the member classifiers validated on their actual prediction quality. This paper is a proposal of a new ensemble method – *Covariance-signature Concept Selector* – which, like *state-of-the-art* solutions, uses both the model accumulation paradigm and the detection of changes in the data posterior probability, but in the integrated procedure. However, instead of ensemble fusion, it performs a static classifier selection, where model similarity assessment to the currently processed data chunk serves as a concept selector. The proposed method was subjected to a series of computer experiments assessing its temporal complexity and efficiency in classifying streams with synthetic and real concepts. The conducted experimental analysis allows concluding the advantage of this proposal over *state-of-the-art* methods in the identified pool of problems and high potential in practical applications.

Keywords Data stream · Classifier selection · Classification · Pattern recognition

1 Introduction

An almost trivial introduction, but at the same time hard to deny, is the often overused statement that *the modern world is filled with data*. The difficult beginning of the third decade of the 21st century has irreversibly transferred the central axis of human existence to the global network of computer systems, which, almost like in W. Gibson's Neuromancer, spans the vast majority of our everyday lives. Most of the time, we organize and process subsequent portions of data in e-mails, instant messages, remote video calls, documents, reports, and notifications. Only to spend our free time accepting more portions of data in the form of *Netflix* or *Youtube* materials tailored to our taste, posts from our social bubble on *Twitter* or music served by *Spotify*. In such times *Machine Learning* drifts its intuitive meaning into a marketing slogan, eagerly taken up by companies such as *Google* or *Amazon*, which is to be a magic panacea for understanding the amount of data we produce and receive every day.

The reality behind this slogan is a much more prosaic complex of difficulties. Typical, classic recognition models target stationary problems, i.e., those describing a particular unchanging concept represented by a finite set of labeled problem instances [1]. The existence of highly numerous data sets, which do not allow for storing the entire dataset in the memory of a computer system, justifies the development of *inductive learning* paradigm with *incremental learning* [2]. Most often using iterative model update procedure based on upcoming data batches. In the case of high-dimensional data, prone to the *curse of dimensionality* [3], methods of *feature selection* and *extraction* are used [4], aimed at reducing the difficulty of the analyzed problem. In the case of data with a high cost of label acquisition, *active* and *semi-supervised learning* paradigms are used [5, 6], allowing the identification of the most challenging objects for the recognition model, thanks to which it is possible to reduce the involvement of human experts in the field. All these methods fit into the common domain of *Big Data* [7], offering solutions for problems characterized by a large amount of data (*volume*). They also have to deal with high speed of data processing (*velocity*), a great *variety* affecting the difficulty with the reliability of labeling (*veracity*) and the potential *value* for the end-user of the system.

✉ Pawel Ksieniewicz
pawel.ksieniewicz@pwr.edu.pl

[1] Department of Systems and Computer Networks, Wroclaw University of Science and Technology, Wybrzeże Stanisława Wyspiańskiego 27, Wroclaw, 50-370, Poland

An additional challenge for *pattern recognition* systems appears when the problems perfectly described by *5V* of *Big Data* also differ by the dynamics of the concepts contained in them. This subject has been dealt with for twenty years in the field of *data stream processing*, defining the characteristics of the variability of the problems *posterior probability* as the phenomenon of *concept drift* [8]. However, classification problems represented by data streams are rarely limited to describing a single point in time, and the systems dedicated to their processing must take into account the temporal changes of class definitions [9].

According to the established taxonomy, this variability may also have different characteristics. First of all, we can talk about *real* and *virtual drifts* [10]. In both of these cases, the distribution of objects in the feature space changes, but in *virtual drift* it does not affect the decision boundary of the model. A particular case of such a situation is the drift in prior probability [11], which apparently does not change the class definitions, but by changing their counts, often leads to noticeable changes in the classifier's decisions and its measurable quality [12]. This type of subject is more widely discussed in the recently popular subfield of imbalanced stream processing [13–15].

The characteristic of its dynamics is a much more frequently analyzed property of *concept drift* [16]. Literature distinguishes *sudden drifts* – consisting in an immediate change of one concept into another [17], *incremental drifts* – where the concept smoothly transforms its distribution [18], and *gradual drifts* – where we deal with a transition period in which two different definitions of the same class occur simultaneously in a changing proportion [19]. The most popular techniques used in dealing with the problems changing over time are built-in mechanisms that allow updating the recognition model taking into account the forgetting of old concepts [20]. There may also be distinguished pre-processing methods dedicated to streams [21] or – the most common in literature – ensemble approaches, using multiple recognition models in line with the wisdom of crowds paradigm [22, 23].

The oldest widely used ensemble method dedicated to processing data streams is the *Streaming Ensemble Algorithm* (SEA) proposed in 2001 by Street and Kim [24]. It is a relatively simple approach based on the accumulation of models built on successive data chunks until reaching the limit of predictors. Exceeding the limit forces the weakest model to be removed from the pool according to the quality calculated on the most recent portion of the labeled data. Further development of this idea led to the *Accuracy Weighted Algorithm* (AWE) [25], which proposes a metric for evaluating models and weighting their supports when obtaining the final ensemble prediction based on the mean square error of the classification. Later proposals most often further developed the paradigm of selecting the models of best quality, such as the *Accuracy Updated Ensemble* (AUE) [26] – correcting the metric from AWE and additionally introducing the possibility of updating the models in the pool, *Recursive Ensemble Approach* (REA) [27] – balancing data with the use of historical samples or *Weighted Aging Ensemble* (WAE) [28] – proposing advanced mechanisms to rejuvenate models and assess the diversity of the constructed pool.

In the current trends in the design of recognition models for the processing of data streams, the use of *Hoeffding Tree* (HTC) varieties based on the *Hoeffding boundary* [29] as the base model gained a significant advantage. It is a family of incremental classification methods that allow to reject the assumption of invariability of a posterior distribution while maintaining high processing efficiency with very large data sets [30]. This makes it a perfect fit for data streams prone to concept drift. Currently, the most frequently used variant of HTC is *Concept-adapting Very Fast Decision Tree learner* (CVFDT) [31], identical to *Very Fast Decision Tree* (VFDT) with moving window, but characterized by a much lower computational complexity. This is what the methods of the current *state-of-the-art* in the field are based on. These include *Leveraging Bagging Classifier* (LBC) [32] – an approach developing *Oza Bagging* [33], increasing resampling based on the Poisson distribution, using input and output detection codes and introducing the *Adaptive Windowing* (ADWIN) [34] drift detector as a pruning tool.

Another *state-of-the-art* approach, *Adaptive Random Forest Classifier* (ARF) [35] is actually fixed on HTC as a base classifier, using the ADWIN detector as well as the LBC, but in splits based on subspaces and building the *background trees*, preparing to recognize the new concept as the detector picks up a drift warning. An extremely interesting alternative to LBC and ARF – outperforming them in imbalanced data streams – is the *Kappa Updated Ensemble* (KUE) [36], also based on resampling using the Poisson distribution, building a diversified subspace band with a fixed size, but implementing pruning based on Kohen-Kappa metric – more sensitive to class imbalance.

Inclusion of methods like ADWIN in the body of *state-of-the-art* induction procedures highlights the importance of another topic of research into dynamic data streams – *concept drift detection* [37, 38]. The dominant majority of methods of this type use the base model that classifies incoming problem instances, analyzing changes in recognition quality. This paradigm has been common since the *Drift Detection Method* (DDM) [39], which analyzes online *error-rate*. The *Early Drift Detection Method* (EDDM) [40] introduced later proposes the *distance-error-rate*, measuring the averaged distances between successive classifier errors. Alternative methods often use tests such as the *Page-Hinkley Test* (PHT) [41] or CUSUM [42, 43], which compares the current quality of a classifier with its average so far [44], or the *Statistical Drift Detection Method* (SDDM) [45]

– which produces metrics based on prior and posterior probabilities. Also dependent on the base model are the popular algorithms such as ADWIN or *Paired Learners* (PL) [46] based on, successively, two windows of variable size while maintaining a single recognition model and implementing a similar mechanics with the strategy of using two models of different reactivity.

This paper proposes a novel algorithm being a product of approaches utilized both by *ensemble classification* algorithms for *data streams* and *drift detectors*. On the one hand, it replaces the *drift detection* with *concept identification*. On the other hand, it constructs an updatable pool of classifiers dedicated to the identified characteristics of the statistical relationship between the attributes of the problem changing in time. The premise of the proposed algorithm is the construction of a diverse pool of classification models, the purpose of which is not the mere assurance of diversity but rather an appropriate adjustment of the prediction to the diversity of concepts previously encountered by the recognition system. An additional guideline is an attempt to abandon the paradigm of model assessment by their quality common in *state-of-the-art* solutions and replace it with an assessment of the scale of similarity between the training and test data.

The main goal of the proposed approach is:

a) to reduce the use of labels in the induction procedure, which will use unsupervised method for the detection of changes in the concept,
b) to expand the potential of using data stream processing methods for other base classifiers than HTC,
c) to reduce the processing time of a single batch in the stream classification.

Although tree-based models are perfect for classifying synthetic problems based on legacy generators and a pool of real benchmark streams, according to Wolpert's theorem, it is impossible to recognize one classification model as universal for all recognition problems. Therefore, it is justified to propose alternative methods that better use the generalization potential of neural models for specific examples analyzed in this work. Such algorithms often allow for convergence in problems described only by quantitative attributes with a normal distribution in significantly less time than CVFDT.

The main contributions of the work are:

- A proposal of the *Covariance-similarity Concept Selector* (CSCS) algorithm dedicated to the classification of *data streams* containing *concept drift*.
- A proposal of procedure for generating data streams utilizing static data, enabling the proper assessment of the quality of the stream classifier in the environment of real concepts.
- A series of computer experiments assessing (*a*) the computational overhead and (*b*) the quality of classification of the proposed method with the comparison of *state-of-the-art* methods on the example of synthetic and real concepts.

2 Methods

The research hypothesis experimentally validated by this paper is that: *"It is possible to distinguish between separate data chunks belonging to the pool of concepts identified based on the signatures determined on their statistical properties"*. The key challenge here is, therefore, to find an appropriate method to establish the unambiguous and easy to cross-compare *signature of the concept*.

The considered method for determining the signature, due to the simple obtaining process for large matrices [47], will be the calculation of the auto-covariance matrix, denoted as K_{XX}. Statistics defines this structure as a square matrix describing the covariance between each pair of elements of a given random vector [48]. Suppose its calculation for a given portion of the training set. Such a case gives a symmetrical matrix with dimensions of $d \times d$. The diagonal of such representation contains information about the variance of each of the d attributes of the problem. Symmetrical rows and columns store generalized information about its notion in multiple dimensions.

It is essential to underline that such an approach – like any information retrieval method – employs the naivety of a particular assumption. In this case, the assumption is the quantitative nature of the attributes. Such problem description enables drifting variability both in the bias and the relationships between the features of the problem.

2.1 Concept signature

This subsection aims to present the processing procedure of the proposed method on a simplified example of a drifting data stream that allows for appropriate visualization. Figure 1 shows auto-covariance matrices computed on twelve consecutive chunks of six data streams containing concept drifts typically distinguishable by the taxonomy (*sudden*, *gradual* and *incremental drift* with both *recurring* and *nonrecurring* concepts). Each data stream contains two concept drifts whose central points lie between the third and fourth and the ninth and tenth chunks. For the clarity of the visualization, to allow the observation of differences between the calculated *concept signatures* with a bare eye, all problems have been simplified to streams with six dimensions, half of which is informative and the other half – repeated [49]. The matrices were visualized in grayscale, with eight-bit depth, to the range limited by the *standard deviation* of the values present in the historical signatures.

As can be observed, the structure of the relationship between the problem attributes, measured by their variance

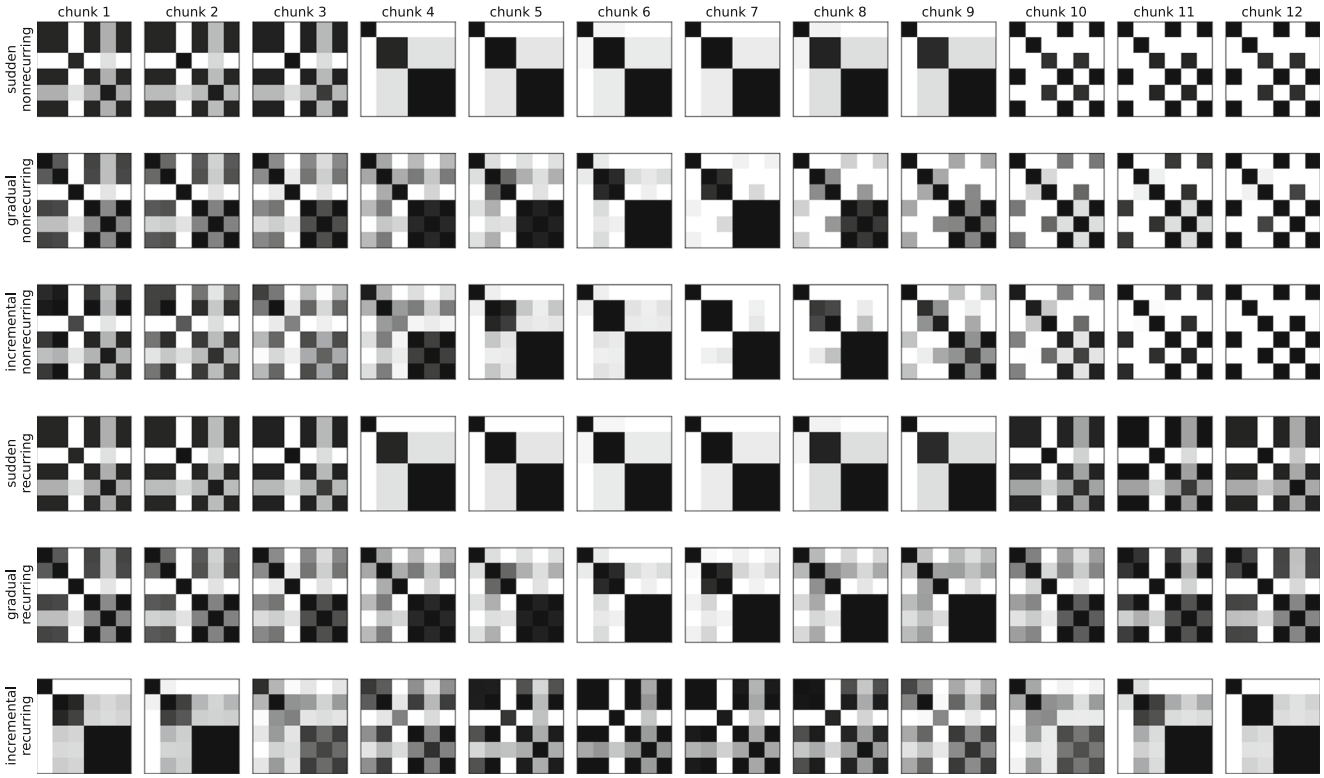


Fig. 1 Visualization of auto-covariance matrices computed on twelve consecutive chunks of six data streams containing concept drifts typically distinguishable by the taxonomy

and cross-variance, changes according to the concept drift dynamics. In the simplest case of *sudden drift*, concept signatures are legible and, in simplified visualization, allow for unequivocal identification of the current concept. On the other hand, in *incremental* and *gradual drifts*, the transition phase (*incremental drift*) or the co-presence phase (*gradual drift*) of two different concepts is noticed in the course of concept changes.

A preliminary experiment verifies whether the changes in the signatures reflect the real nature of stream dynamic. Its results are presented in Figs. 2 and 3. As in the case of auto-covariance matrix visualization, this experiment also uses simplified, six-dimensional data streams.

The streams were expanded to two hundred chunks of five hundred objects to analyze long-run behavior. Each of the six presented plots consists of two columns. The right column (*Covmats*) shows the current signature in the selected chunk. In contrast, the left column (*variance of difference*) shows the value vector of the variance of the difference between the current and each of the historical concept signatures. Dashed lines present actual values, while the red line shows its trend smoothed by the median filter. Additionally, in the form of a dotted line in the background, a *concept change curve* is presented, determined according to the *concept sigmoid spacing* parameter of the synthetic stream generator available in the *stream-learn* package [50]. The plots represent ten points in time, twenty chunks apart, starting with the eighth iteration.

Figure 2 shows the changes in the *variance of differences* vector for data streams with *non-recurring drift*, ie. introducing a new concept for each drift occurrence. As can be seen by observing the first row of visualization (*chunk 7*), the initial phase of the stream in which only one concept is present leads to changes in a much smaller range than the others. At the same time, the vector distribution seems (apart from a small number of observations) to be normal.

The observation of the second row of the visualization (*chunk 27*) allows for the verification of the curve value at the end of the first concept drift. The variance trend at the first drift in each of the dynamics almost perfectly mirrors the dynamics of the *concept change curve*.

The third line of the visualization (*chunk 47*) shows how the stable, equal trend curve of the signature difference variance reflects the last moment of stability of the second concept. Further rows confirm these observations, which is essential, also showing that the differences between the current signature and those for historical concepts differ but are still far from the signatures of the current concept. An interesting observation here is also the fact that in the case of *incremental drifts*, the differences in the transition phases are noticeably lower than in the case of *gradual drift*, which suggests a significant similarity in their transitional concepts.

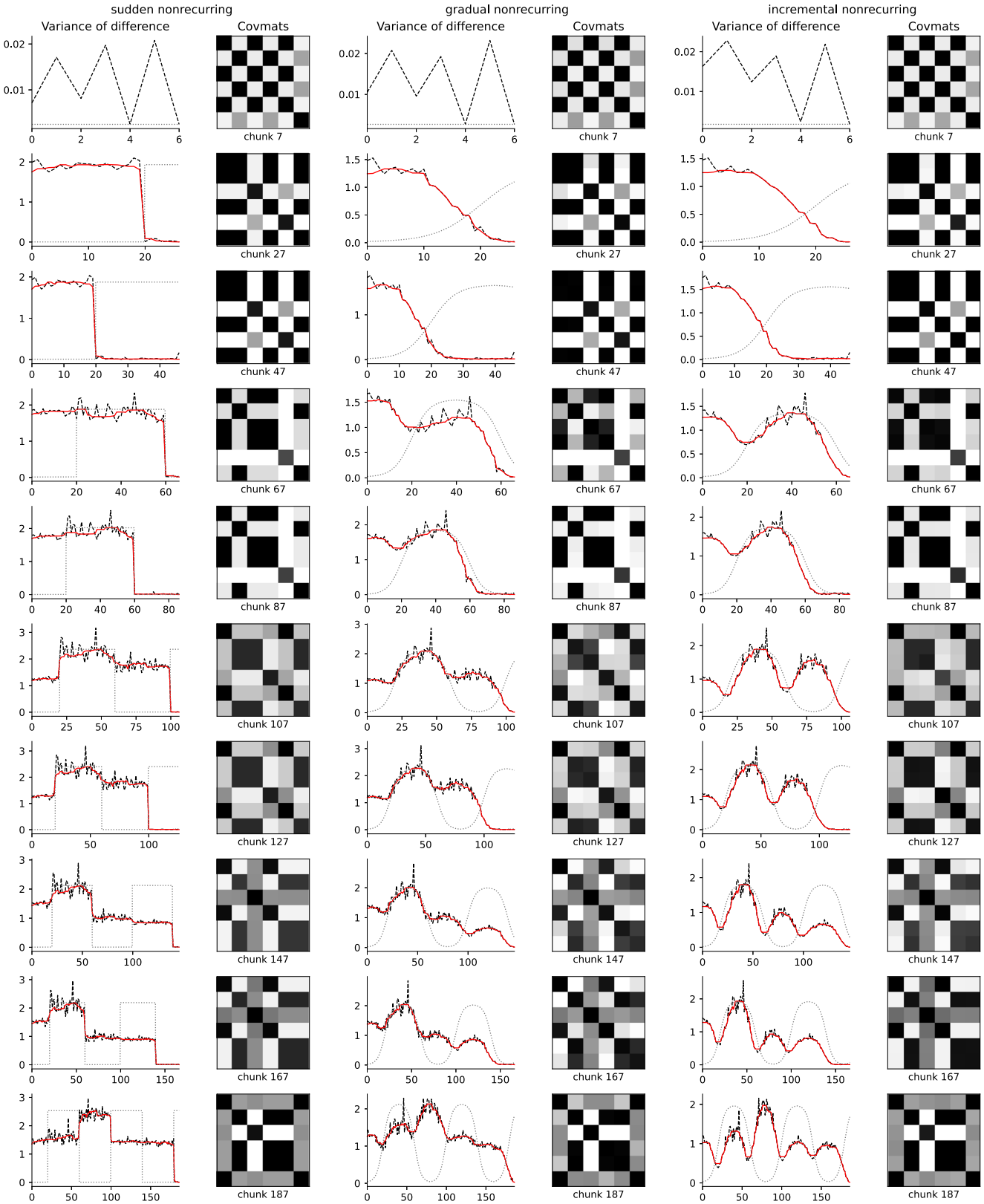


Fig. 2 The changes in the variance of differences vector for data streams with non-recurring drift

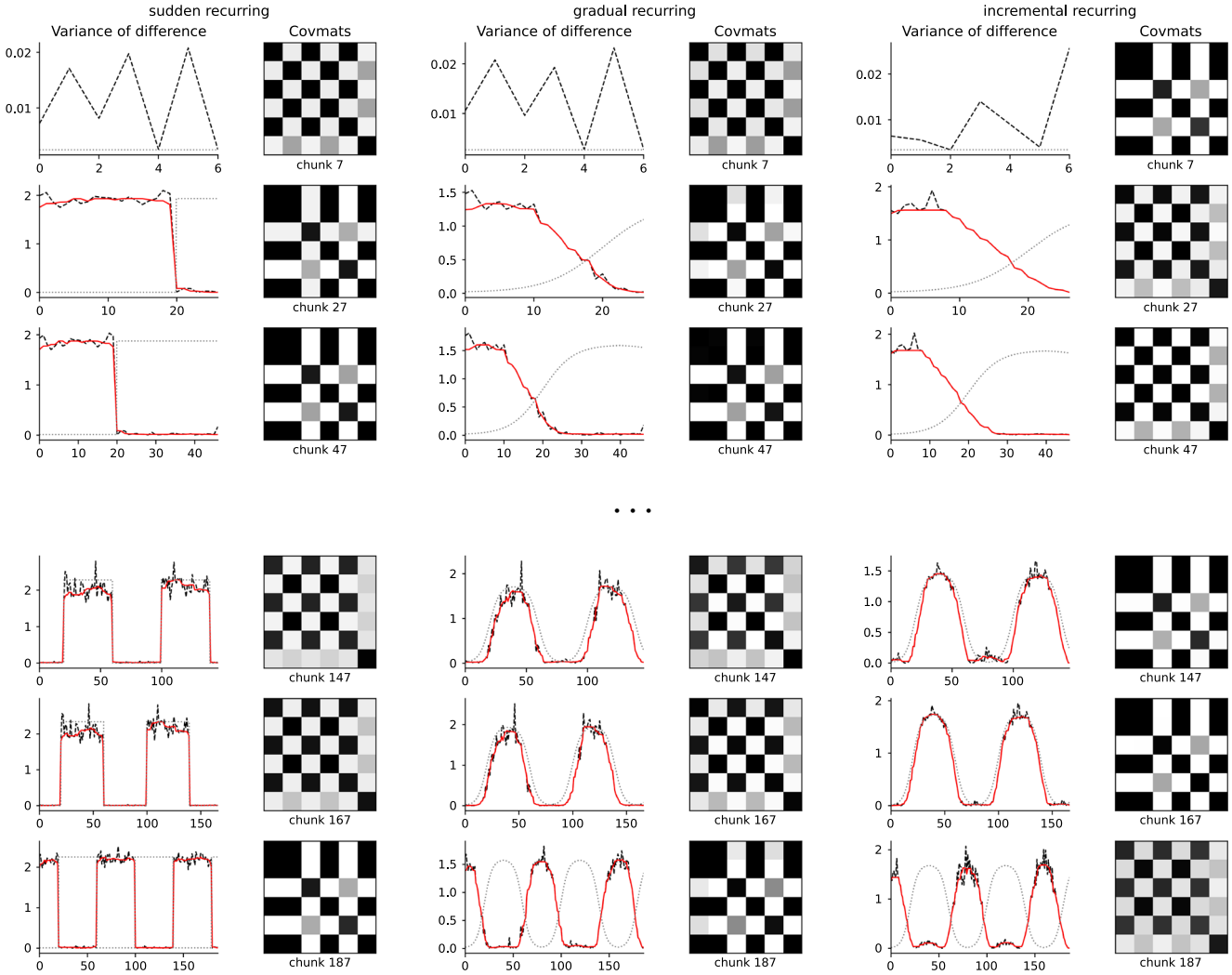


Fig. 3 The changes in the variance of differences vector for data streams with non-recurring drift (shortened to begining and ending of processing)

However, it creates a potential risk, which may lead to the generalization of the transition phase as a uniform concept in the future – in the worst case – leading to the generation and utilization of a random classifier during the drift.

Figure 3 shows the changes in the variance of the differences vector for data streams with *recurrent drift*, i.e., oscillating between the two concepts each time drift occurs. Its analysis makes it possible to replicate all the observations made so far, but without noticing a decrease in the variance in the transition phases between the concepts in the *incremental drift*. Such observation probably comes from a characteristic of *recurrent drifts*, where all transition phases are coherent and, in the context of class distribution, occur with the same or mirrored dynamics of the posterior distribution. *Recurring drift* also allows for an even more precise analysis of differences between the dynamics of transition and the values of the variance of the difference vector.

As we may observe, the red line on the plots almost perfectly reflects the concept change curve, which allows for the initial plausibility of the research hypothesis. Moreover, it leads to a proposal of a method using *concept signatures* – calculated as data chunks auto-covariance matrixes – for effective classification of data streams.

2.2 Covariance-signature concept selector

This subsection presents the context of using the *concept signature* described above in the construction of an effective method for data stream processing. The proposed recognition procedure of *Covariance-signature Concept Selector* (CSCS) uses the achievements of *state-of-the-art* ensemble methods such as ARF, LBC or KUE and builds a pool of classifiers, but, unlike the existing algorithms, does not integrate them, but conducts the selection of the most appropriate model for the currently predicted data chunk [51]. It means that, unlike in the rest of the considered methods, in the CSCS algorithm, a decision is always made by a single model, in line with the approach of static classifier selection.

What is extremely important, the calculation of the concept's signature without pattern labels potentially enables (a) the identification of the concept in the prediction procedure and (b) the selection of an appropriate classifier built on it. As mentioned in the Introduction, the paradigm of the qualitative assessment of the models present in the pool, which aimed at maintaining the highest possible quality of the prediction, is rejected here. Its replacement is the paradigm of the highest similarity between the predictor and the problem. Such change allows both the identification of new concepts and the selection of a model suitable for the current data chunk. The pseudo-code of the CSCS algorithm is presented in Algorithm 1.

Algorithm 1 Covariance-signature Concept Selector – CSCS.

Require:
$\mathcal{DS}$: data stream,
k: number of historical models to store,
n: number of historical covariance matrixes to store,
$\Psi()$: base classifier, $H()$: harmonic mean.
Ensure:
C: set of historical covariance matrixes,
Π: pool of historical models,
M: set of covariance matrixes for historical models.

1: $C, \Pi, M \Leftarrow \emptyset, \emptyset, \emptyset$
2: **for** $DS_i : \{X_i, y_i\}$ **in** $\mathcal{DS}$ **do**
3: $c \Leftarrow K_{X_i X_i}$ ▷ COVARIANCE FOR CURRENT CHUNK
4: $cv \Leftarrow [\mathbb{V}(c - c') \; \forall \; c' \in C]$ ▷ VARIANCES OF COVARIANCE DIFFERENCES
5: $mv \Leftarrow [\mathbb{V}(c - c') \; \forall \; c' \in M]$
6: $t \Leftarrow H(cv)$ ▷ NOVELTY TRESHOLD
7: **if none** $mv_j < t$ **or** $\|mv\| = 0$ **then** ▷ NOVEL CONCEPT
8: $\Pi \Leftarrow \Pi \cup \Psi(DS_i)$
9: $M \Leftarrow M \cup c$
10: **else** ▷ IDENTIFIED CONCEPT
11: **for any** $\Psi_j \in \Pi$ **where** $mv_j < margin$ **do**
12: update Ψ_j with X_i, y_i
13: **end for**
14: **end if**
15: **if** $\|M\| > k$ **then** ▷ MODEL PRUNING
16: remove oldest element from M
17: remove oldest element from Π
18: **end if**
19: **if** $\|C\| > n$ **then** ▷ COVMAT PRUNING
20: remove oldest element from C
21: **end if**
22: **end for**

Two pruning hyperparameters typical for this type of procedure control the processing: k – specifying how many models might construct the pool and n – specifying how many historical signatures should take part in comparisons [22]. For the correct operation of the method, it is required for the selected base classifier Ψ, like in all *state-of-the-art* algorithms since proposition of AUE, to be updating capable. In line with the method assumptions, it does not have to be able to adapt independently to changes in the concept. Both *Hoeffding Trees* and the *Multi-layer Perceptron* [52] as well as simple methods such as *Gaussian Naive Bayes* [53] can be used here. Alternatively, for non-updateable base classifiers, SEA algorithm may be employed here as a meta-classifier giving this ability to every model. Each model in the pool is dedicated to understanding and generalizing a single, given snapshot of the posterior probability present in the stream's course.

The procedure starts with initializing the empty sets Π, C, and M. The set Π is to store a pool of up to k active classifiers. Set C holds a maximum of n most recent signatures and set M holds a signature for each Ψ currently contained in Π.

For each successive chunk $DS_i : \{X_i, y_i\}$, the concept signature c is first computed as the auto-covariance matrix of the set X_i. Then the vectors cv and mv are determined as the variances of the differences ($\mathbb{V}$) between c and each element of the sets C and M in the way described by Figs. 2 and 3.

In the next step, the *novelty threshold* t is determined, which is the *harmonic mean* of the vector cv. The DS_i concept is considered as yet unknown if none of mv elements are below the threshold t. For unknown concepts, the classifier trained on DS_i is added to the pool Π, and the set M is supplemented with the signature c. Otherwise, each Ψ_j model from Π for which the corresponding ms_j value was lower than the threshold t is considered coherent with the current concept and updated with DS_i. This approach allows the recognition system to adapt to new concepts and update stored models if the induced similarity between their concepts and the current batch of objects is detected.

After the proper part of the processing loop is completed, the procedure carries out the tidying up work. If the module M exceeds k, the oldest elements from sets M and Π are removed. Accordingly, if the module C exceeds n, the oldest element of the set C is removed. Such an approach ensures that the CSCS processing time is independent of the stream length and reduces the number of comparisons necessary to identify similarities between the concepts.

If less than k iterations detect less than n new concepts, the set M will still contain the signatures already removed from C, and the Π set – models paired with them. It is a deliberate and planned action to allow the CSCS algorithm to return to a temporarily obsolete model in the event of

recurrent drift. The purpose of the hyperparameter n is to determine how many concept signatures are present in the set C to correctly estimate the threshold t determining the identification of a new signature representing a concept that does not have a corresponding model in the Π pool. Such design aims to make the number of comparisons made at each method iteration independent of the current course of the stream, ensuring a relatively low time complexity of the proposed solution.

The CSCS prediction procedure for batch processing environment, described by Algorithm 2, is relatively simple and analogous to the training procedure. First, for the given test set X, signature c is determined. Next, the procedure computes the vector of variance of the signature differences of the stored models mv and returns the prediction of the selected model with the smallest mv value.

Algorithm 2 Prediction using CSCS for batch environment.

Require:
X: testing set,
Π: pool of models,
M: set of covariance matrixes for models.
Ensure:
y: set of predictions,

1: $c \Leftarrow K_{XX}$
2: $mv \Leftarrow [\mathbb{V}(c - c') \; \forall \; c' \in M]$
3: $y \Leftarrow \Pi_{argmin(mv)}(X)$

Algorithm 3 Prediction using CSCS for online environment.

Require:
$x \in X$: testing sample in a set of last m testing samples,
Π: pool of models,
M: set of covariance matrixes for models.
Ensure:
y: prediction,

1: $c \Leftarrow K_{XX}$
2: $mv \Leftarrow [\mathbb{V}(c - c') \; \forall \; c' \in M]$
3: $\Psi = \Pi_{argmin(mv)}$
4: $y \Leftarrow \Psi(x)$

In the presented prediction procedure, the decision is not made separately for each object but globally for the entire test set. In the case of using the CSCS method in online processing, it would be necessary to supplement the prediction procedure with a window mechanism, building a concept signature for a given horizon of recent test patterns. So, the online processing (Algorithm 3) procedure uses the last window of given m samples to determine c signature and gives prediction with the same approach to model identification.

3 Experimental evaluation

The principle of the CSCS algorithm may seem very similar to a typical *drift detection* procedure. While this method can detect changes in the concept, it cannot be identified as such a tool. The first difference is the use of an approach that does not rely on the assessment of the predictive capability of the system and does not harvest labels in any phase other than training a new model or updating existing ones. Second, the actual drift detections only occur here in the case of a data chunk describing a new, unknown concept and should not happen during any recurrent drift scenario.

Identification does not necessarily have to refer to an entirely new concept. In the case of incremental or gradual drifts, it may also occur several times in the course of changes, depending on their dynamics. Therefore, the planned experimental evaluation of the method will only concern its ability to construct a reliable recognition system, and it is in this context that the comparison with *state-of-the-art* methods will be prepared. For its needs, (a) 30 data streams containing synthetic concepts with quantitative features with different dynamics of change, (b) 15 data streams containing synthetic concepts with mixed quantitative-qualitative features, and (c) 8 data streams consisting of suddenly drifting real concepts with different dimensionalities were prepared.

3.1 Experimental scenarios and environment

3.1.1 Synthetic concepts from *stream-learn* generators

The relatively most superficial part of the experiment preparations was the generation of 30 synthetic data streams, carried out using the stream generator available in the *stream-learn* package. This generator always produces only quantitative attributes with a given dimension and proportion between informative, redundant, and random attributes. The shared part of the configuration of each of the streams prepared in this way was:

- 250 chunks for 250 objects each,
- 20 informative concept attributes,
- 10 concept drifts in the flow of a stream.

The six separate processing scenarios were flows diversified by type of drift:

1. non-recurring sudden drift,
2. recurring sudden drift
3. non-recurring gradual drift,
4. recurring gradual drift,
5. non-recurring incremental drift,
6. recurring incremental drift.

Each of the scenarios types of data streams generated a pool of problems in five replications differing in a randomly determined *random state* used so that – for optimal comparison – the same concepts were present in the following types of drift.

3.1.2 Synthetic concepts from *MOA* generators

In order to diversify the pool of synthetic problems, the experimental evaluation also used stream generators offered by the *MOA* package. These are solutions commonly used in most comparative experiments in data stream processing. The generators selected for the analysis were:

- *Random Radial Basis Function stream generator* (RBF) – a method for constructing problems with quantitative attributes, configurable in terms of the number of classes, attributes, and source centroids of the classes. Standard hyperparameterization was used here, i.e., binary problems with ten attributes and 50 class centroids.
- *Random Tree stream generator* (RTG) – a method proposed by Hulten and Domingos [54] based on a tree that randomly splits attributes and assigns random labels to leaves. It generates five quantitative and 25 qualitative attributes for a binary problem by default.
- *Agrawal stream generator* (AGR) – the most classical method of stream synthesis, proposed in 1993 by Agrawal et al. [55]. It always generates nine attributes, three of which are qualitative and six are discrete quantitative parameters from a uniform distribution.

Each of the stream generators prepared a pool of problems in five replications differing in a randomly determined *random states*.

The proposed method, due to decisions made to the structure of the concept signature calculation approach, should find its potential mainly in the RBF generator, where the characteristics of the attributes should allow for the correct identification of the concept by its signature. However, the predominance of qualitative or discrete quantitative attributes with a low range will probably impede proper identification, which seems to be a guideline against using this solution in streams similar to AGR or RTG.

3.1.3 Real concepts

The more complex challenge was to prepare a pool of problems with real concepts. In reference works there often appear a relatively short list of real streams, such as *electricity*, *poker-lsn* or *insects* to enumerate a few. Most often, however, they are very specific types of streams, characterized by (*a*) vanishing classes, (*b*) very low dimensionality, (*c*) categorical attributes only, or (*d*) strong class imbalance. Neither of these properties is the primary challenge of the research presented in this paper. Therefore, studies using them for the comparison of methods could be unreliable. For the purposes of the evaluation, a method of obtaining streams containing real concepts determined on the basis of static datasets was proposed.

The `@w4k2/benchmark_datasets` *Github* repository was used here, which contains a specially prepared, unified collection of classification datasets available for use in pattern recognition experiments built based on public UCI and KEEL repositories. The procedure for developing a stream containing real concepts consists of the following five steps:

1. From the available pool of datasets, select all binary problems, the dimensionality of which (number of attributes) is not less than d.
2. For each such dataset, draw 100 of d-dimensional subspaces, and then, using the full data set, the build GNB model on them.
3. If, in any of these cases, the model, when tested on the full training set, achieves at least 75% of *balanced accuracy score*, consider it as fit for use in a stream. Otherwise, ignore the dataset.
4. On all the sets considered as fit for use, perform oversampling using the SMOTE algorithm to obtain six thousand unique objects evenly distributed over both classes of the problem. Add such a set to the end of the stream.
5. If the data stream obtained in such a manner allows obtaining at least 100 chunks of 250 objects each, consider it as fit for the experiment.

The entire procedure of preparing streams with real concepts is carried out by a script publicly available in the `@w4k2/benchmark_datasets`[1] Github repository. It allowed the construction of eight data streams containing sudden drift, with real concepts described by 2 to 9 dimensions.

3.1.4 Experiments set-up

Computer experiments for the needs of the experimental evaluation have been fully implemented in the *stream-learn* environment, which is a set of tools compatible with the *scikit-learn* package. All the experimental, analytical scripts and implementation of the CSCS method are available in the public Github `@w4k2/cscs` repository.[2]

[1] https://github.com/w4k2/benchmark_datasets/blob/master/real_stream.py

[2] https://github.com/w4k2/cscs

In all experiments, two updatable classifiers were used under the configuration of hyperparameters:

- MLP – *Multi-layer Perceptron*:
  - 10 epochs per chunk in Experiments 1 and 2, which gives limit of 625,000 calls for criterion function,
  - 100 epochs per chunk in Experiment 3, which gives limit of 6,250,000 calls for criterion function,
  - 100 artificial neurons in single hidden layer,
  - *ReLU* activation function,
  - *adam* solver,
  - $L2$ penalty of 0.0001,
  - constant learning rate of 0.001,
- HTC – *Hoeffding Tree* or *Concept-adapting Very Fast Decision Tree*:
  - grace period of 250,
  - *information gain* split criterion,
  - split confidence of 0.0000001,
  - tie threshold of 0.05,
  - non-binary splits,
  - pre-pruning enabled,
  - *Naive Bayes Adaptive* leaf prediction.

where all the settings apart from the number of epochs per chunk for MLP, state the standard hyperparametrization. The ratio between the quality achieved by each of the classifiers at default 200 iterations of the optimizer was proportional to 10 iterations for stream-learn generated problems and 100 iterations for MOA generated and real problems, which preliminary research confirmed.

One of the main goals of the experimental evaluation was to verify the usefulness of various base classifiers in various ensemble classification methods in order to diversify the pool of identified *state-of-the-art* methods. Hence, apart from the currently most popular CVFDT, the potential of neural networks in implementing a simple MLP structure was analyzed, which allows for significantly faster inference than *Hoeffding trees*.

Eight standard comparative methods, indicated earlier in the Introduction and selected according to their popularity in the literature, were also used in the evaluation:

1. **REA** – with the limit of 10 classifiers in pool and pruning enabled,
2. **WAE** – with the limit of 10 classifiers in pool,
3. **SEA** – with the limit of 10 classifiers in pool and accuracy score metric.
4. **AWE** – with the limit of 10 classifiers validated with 5 splits and its original quality metric,
5. **AUE** – with the limit of 10 classifiers validated with 5 splits and its original quality metric.
6. **KUE** – with the pool size of 10 classifiers.
7. **ARF** – with the limit of 10 classifiers in pool (only for HTC comparisons).
8. **LBC** – with the limit of 10 classifiers in pool (only for HTC comparisons).

It should be underlined that the ARF and LBC algorithms are included only in comparisons using HTC as the base classifier. Such a decision comes from the limitation of ARF, which can create only decision tree models, while LBC hinders the effective construction of a neural network by relatively frequent resets of the base model.

For CSCS, to ensure a fair comparison, the k hyperparameter limiting pool size has also been set to 10, and hyperparameter n – limiting historical signature storage – was set to 100.

3.2 Experiment 1: computational overhead of streaming methods

The first experiment aimed to assess the temporal complexity of the analyzed recognition methods depending on the size of the problem described by a number of its features and the number of data stream chunks processed. It was supposed to allow for proper verification of the demand of commonly used recognition methods for the computing power of the systems that utilize them. All results on processing time were acquired with machine equipped with *Intel(R) Core(TM) i5 8265U* CPU with *3.7* GHz and *8* GB of *2666* MHz DDR4 RAM.

The previously declared streams pool was extended by an additional six for this experiment. It was not the quality of the constructed classification models that was important, but the processing time of individual methods. These data streams have been extended to 500 chunks representing dimensionality problems from 2 to 100 attributes (in 20 quants). There were ten drifts of the six types defined in previous subsections in each of them. As an additional element of analysis, a truncated version of the CSCS algorithm (CSCS-h) is present in processing for Experiment 1, which ignores n hyperparameter – responsible for the limit of historical signatures stored in the method memory.

Figure 4 shows, in the form of grayscale heatmaps, the average time in milliseconds of single chunk processing over these ranges by each of the compared methods with a neural network as a base classifier. As can be seen, the baseline MLP processing time here uses stable three milliseconds for any class of the problem. Processing time is not significantly dependent on the number of processed chunks or the dimensionality of a problem.

From the reference methods, the SEA has the relatively smallest computational overhead, which allows processing from about 30 milliseconds for two-dimensional problems

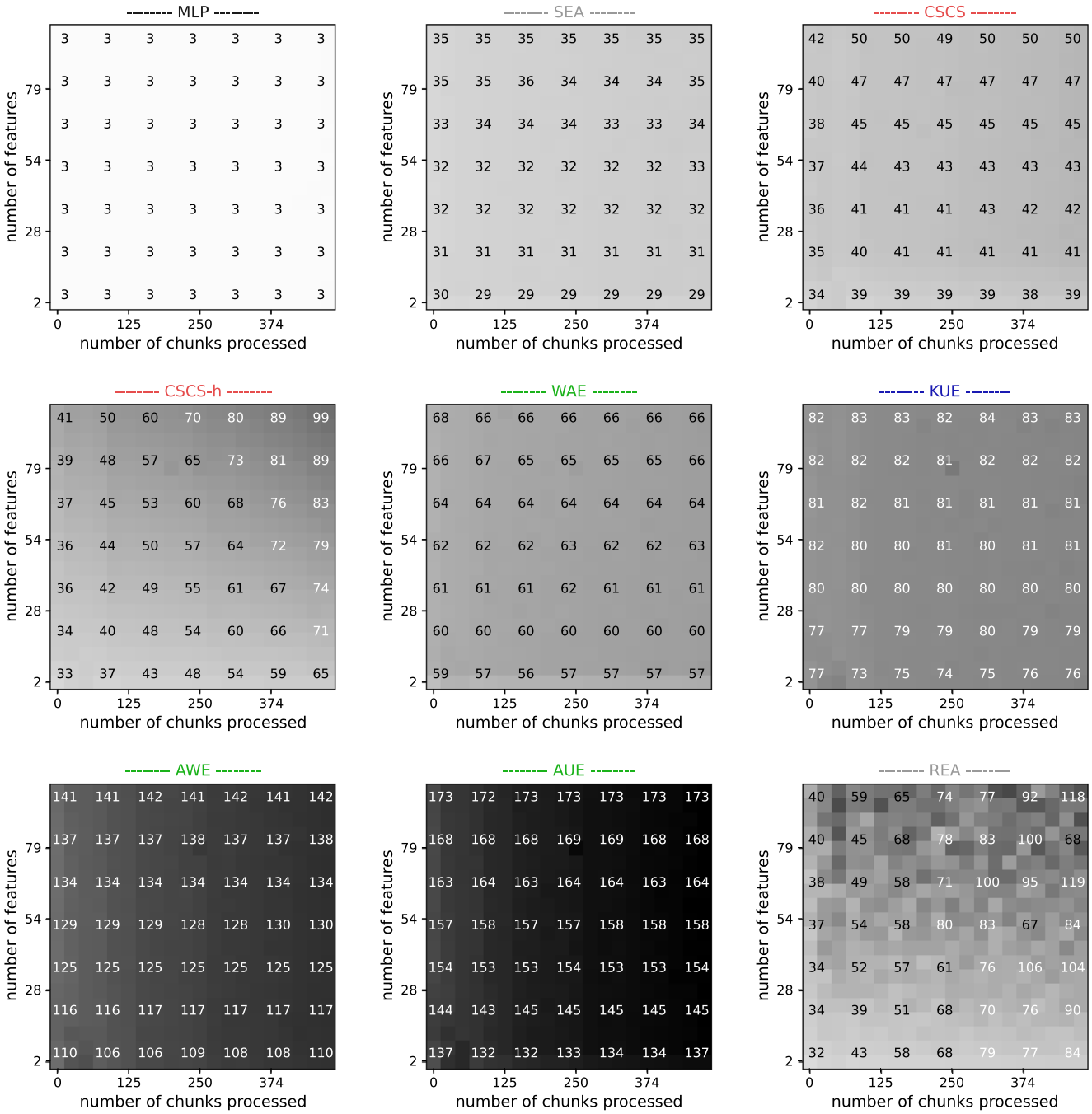


Fig. 4 Time (in milliseconds) to process a single chunk of data by different methods depending on the current run length and the number of attributes on *Multi-Layer Perceptron*

to about 35 milliseconds for 100-dimensional problems. Usage of WAE algorithm increases this time to range 56–68 and KUE to range 73–84. The AWE algorithm increases the processing time even further to about 105 milliseconds for planar problems and about 140 milliseconds for most complex cases. Likewise, the AUE algorithm extends this time to a range from about 130 to about 170 milliseconds. However, all these measurements turn out to be independent of the number of chunks processed so far apart from the very initial stage of processing. A completely separate category is built here by the REA algorithm, which shows strong dependencies on both the dimensionality of the problem and the number of processed chunks, reaching times ranging from 30 to 120 milliseconds.

In this comparison, the CSCS algorithm performs between the level of SEA – the simplest and WAE –

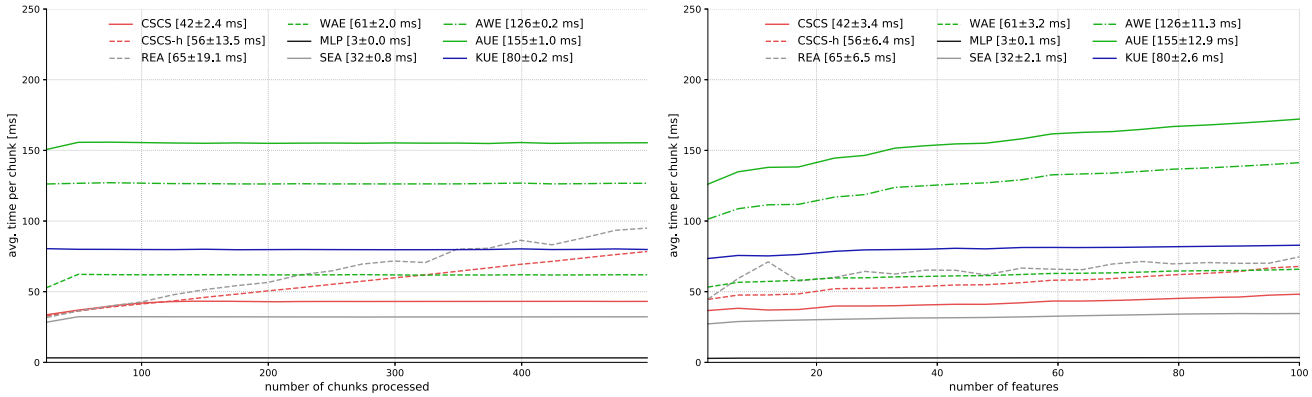


Fig. 5 Time course for processing a single chunk by different methods depending on the current number of chunks processed and the number of attributes on *Multi-Layer Perceptron*

the most optimized ensemble method for classification of data streams used in comparison. With processing times ranging from around 35 to 50 milliseconds, it provides a solution nearly or more than three times faster than the more complex AWE and AUE approaches. However, as can be seen in Fig. 5, in contrast to the competition, this method is not exactly time independent of the number of chunks processed so far. Instead of the complex procedure of assessing the models included in the pool, a less time-consuming statistical analysis of historical signatures is performed in it. As we can see, both on the CSCS-h curve and its heat map, conducting this analysis without limiting the signature horizon would lead to a strong correlation between the stream length and the processing time. At about 1000 chunks, it would equalize the average time needed for AWE and AUE while losing all the time advantage of the new stream analysis strategy. However, the introduction of this limit, after preliminary tests set for 100 signatures, leads to a solution that is slightly slower than SEA, much faster than AWE and AUE, and additionally, after exceeding 100 chunks, time independent of the stream length.

An additional observation that we can make in Fig. 5 is the trend expressed in the relationship between the dimensionality of the problem and the average processing time. As we can see, the standard deviation recorded in the study for the AWE method is 11.3 ms, and for the AUE method – 12.9 ms. Meanwhile, the CSCS method here maintains a standard deviation of 3.4 ms, giving a lower dynamics of processing time increase, which suggests that it may be able to preserve its effectiveness with potentially higher dimensionality.

The same visualizations prepared for the HTC algorithm as the base classifier gives completely different observations. The Hoeffding trees themselves show a strong dependence on the problem's dimensionality, increasing (with low dependence on the number of chunks processed) from about 15 milliseconds for planar problems to about 150 for problems of one hundred dimensions. Compared to the MLP with ten iterations, this gives five times the time for two dimensions and fifty times the time for one hundred dimensions. By making the comparison independent of the stream length (Fig. 7), it can be seen that a significant increase in MLP processing time, by order of magnitude from 10 to 100 iterations, leads to a more computationally demanding solution than HTC only for low-dimensional problems. With the problems typical of stream processing, represented by the stream scenarios, an MLP with 100 epochs is comparable in time to HTC, and an MLP with ten epochs much faster than it.

The computational resource demand hierarchy is also different here than in the case of MLP. The CSCS algorithm, as it may be observed in Fig. 6, generates relatively the lowest overhead here, processing in the range of about 20 to 160 milliseconds per chunk. However, during the time experiment, an upper processing limit was imposed – to filter out methods to slow to converge in the streaming environment – which interrupted learning when the given method exceeded a full second of processing per chunk. This limit was reached by all other ensemble algorithms, including even the relatively most unadorned SEA.

The ARF and LBC algorithms, which are *state-of-the-art* in the processing of balanced streams, reached this threshold in the earliest stages of processing, as can be seen in Fig. 8, achieving it with simple, eight-dimensional problems. The WAE and AUE algorithms remained functional for a bit longer, reaching the threshold around 30 dimensions. With 40 dimensions, the KUE algorithm has become ineffective, with 60 dimensions – the REA and AWE algorithms, and only with 90 - the SEA algorithm. Only the CSCS method and HTC's single base classifier met the limit under all conditions, never exceeding the 20 percent limit (200 milliseconds).

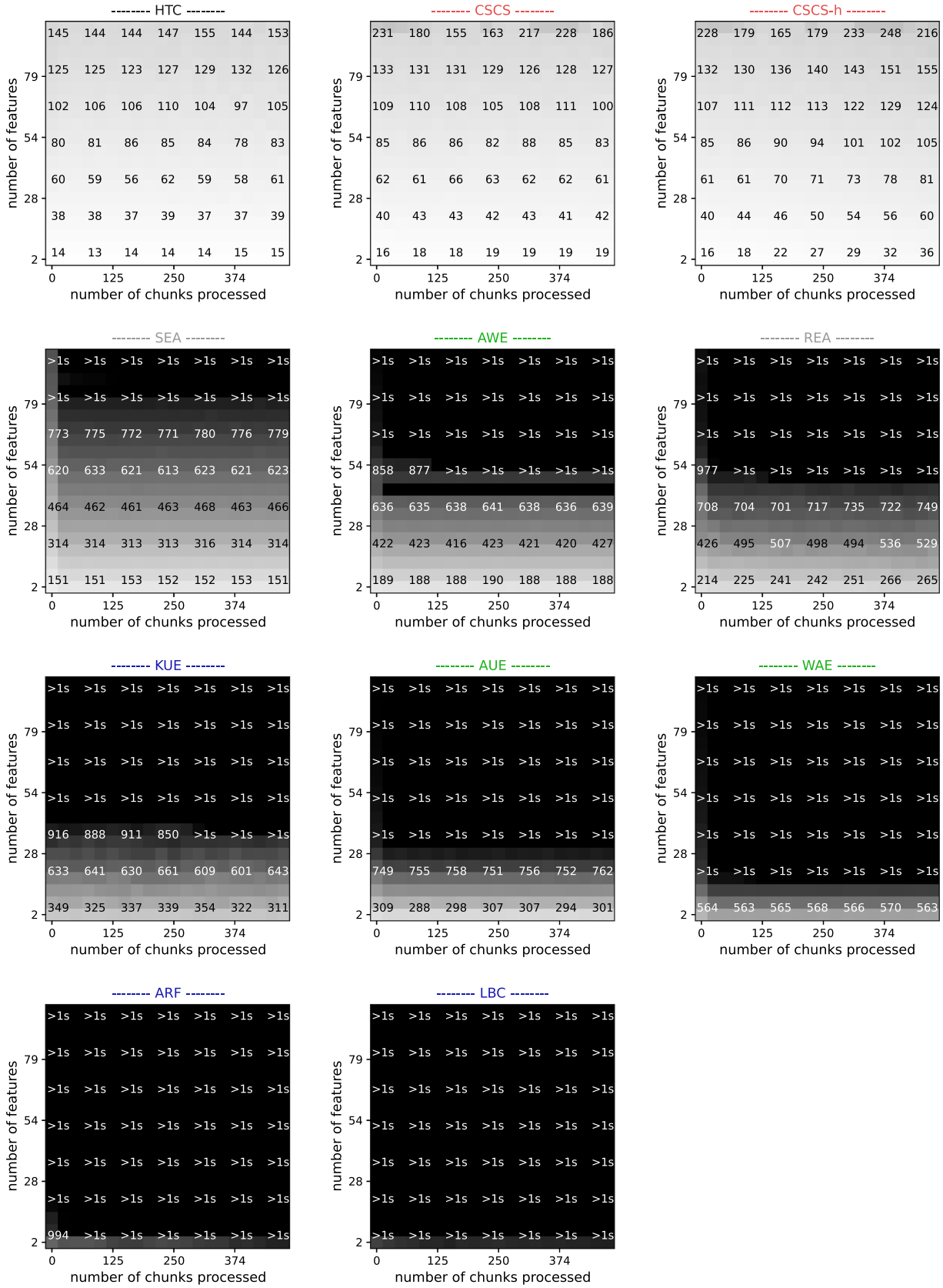


Fig. 6 Time (in milliseconds) to process a single chunk of data by different methods depending on the current run length and the number of attributes on *Hoeffding Tree*

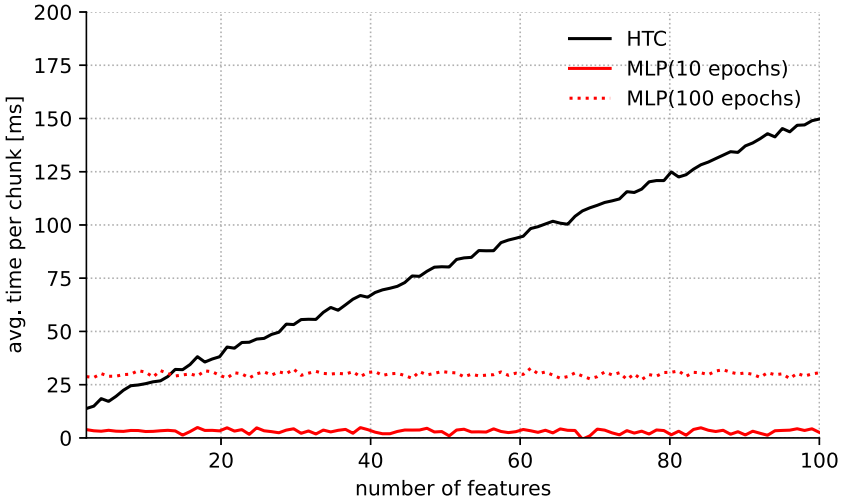


Fig. 7 Comparison of base classifiers training time as a function of dimensionality

3.3 Experiment 2: evaluation on synthetic concepts

The second experiment compared the proposed CSCS method with *state-of-art* algorithms for processing data streams with synthetic concepts. Figure 9 shows quality, as measured by *balanced accuracy score*, on exemplary – representative runs from three non-recurring processing scenarios from *stream-learn* generator for both MLP and HTC base classifier. The X-axis spreads the course of processing in successive chunks, and the Y-axis – quality, presented on a scale from the initial level of a random classifier (50%) to the perfect classification (100%), using the standard *Test-Then-Train* experimental protocol. Chunks in which the CSCS algorithm has detected new concepts are marked on the additional upper X-axis and with the use of dashed black lines. Each of the subfigures contains up to ten pairs of solid lines and hairlines. The solid lines represent the quality trend (the accumulative mean) and the hairlines – the same quality in the chunk for each compared method. The numerical values presented in the legend give the average quality of each method over the entire run of the data stream (Figs. 7 and 8).

For all non-recurring drift cases, with the usage of MLP, the CSCS method has a significant advantage over each of the reference algorithms ranging from 6% for incremental and sudden drifts to 7% for gradual drift. For the cases of gradual and incremental drift, the proportion between the qualities achieved by individual methods is identical. The CSCS algorithm achieves a clearly the best result, the SEA, AWE and AUE algorithms – results differing from each other at the level of thousandths, KUE achieves slightly lower scores, while the base MLP method and REA algorithm clearly stands out qualitatively downwards. In the case of sudden drift, there is a certainly noticeable difference in the SEA, AWE, AUE group, in which AUE achieves a higher quality than the competition. However, this advantage does not exceed one percent, and the result itself is still significantly worse than the one determined for the CSCS.

As can be seen on the hairlines showing the precise quality in each chunk, both smaller drops in quality at the

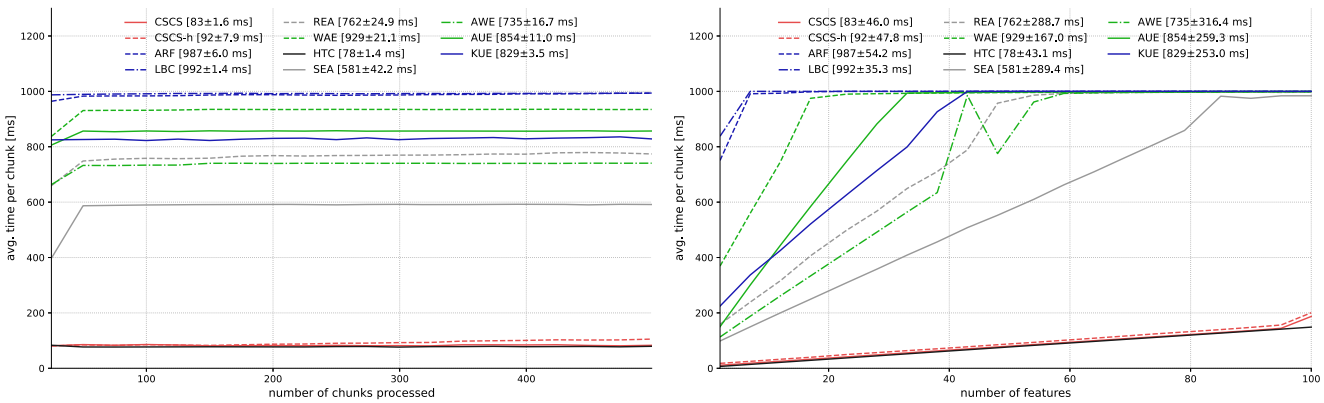


Fig. 8 Time course for processing a single chunk by different methods depending on the current number of chunks processed and the number of attributes on *Hoeffding Tree*

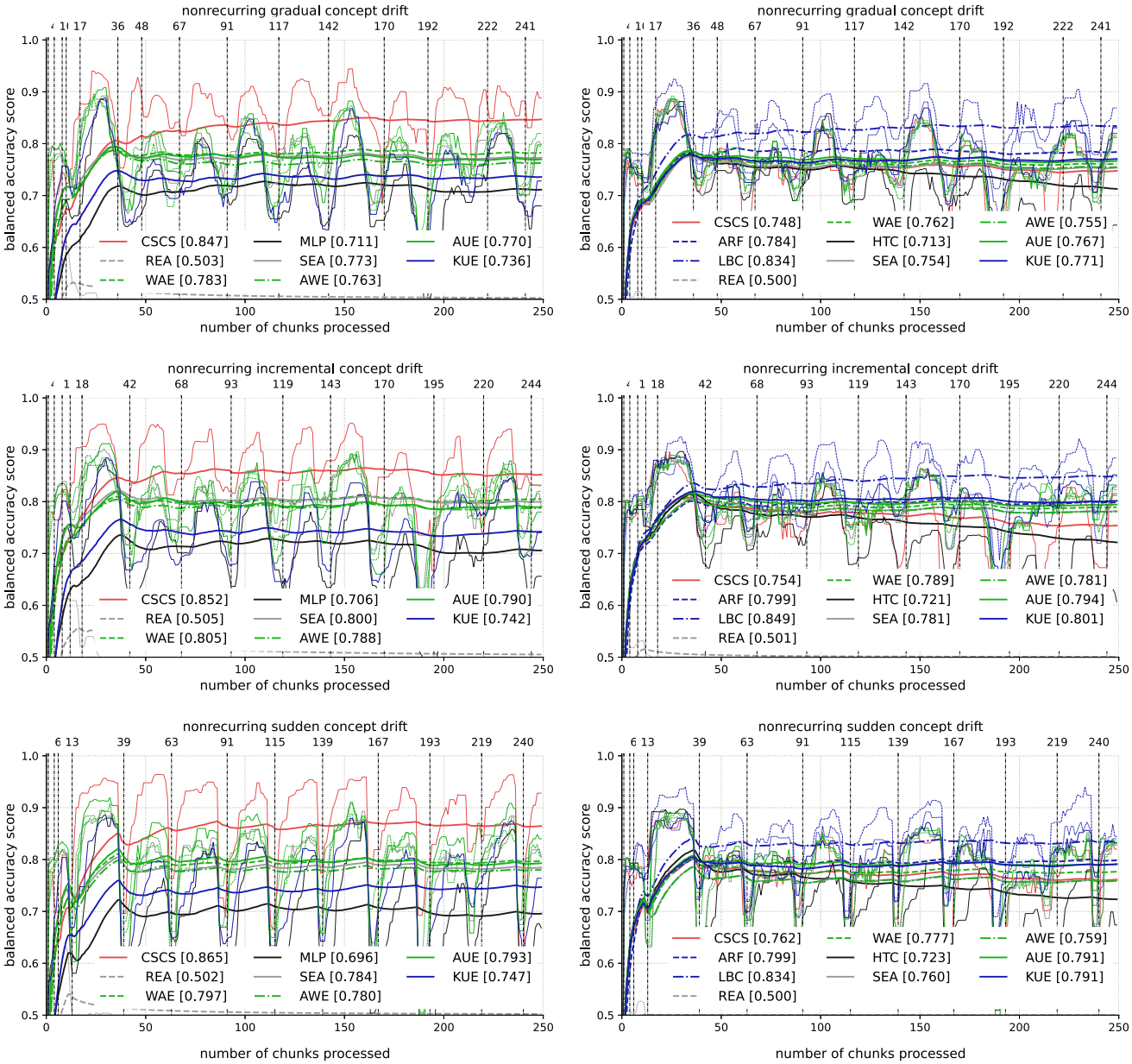


Fig. 9 *Balanced accuracy score* achieved by compared methods on exemplary - representative runs from three nonrecurring drift dynamics for *Multi-Layer Perceptron* (left) and *Hoeffding Tree* (right)

moment of drift and much faster convergence in the phases of the homogeneous concept characterize the CSCS method. What is particularly interesting, in the case of nonrecurrent gradual drift, there are also situations of smooth concept recognition, thanks to which in the initial phase of the drift – which should lead to a decrease in quality – it increases.

Using HTC as a base classifier gives noticeably lower results for these scenarios than MLP. Apart from the base model and mostly random REA, the most noticeable is a clear decline in the quality of CSCS, which seems to be an approach that performs poorly with HTC. A clear advantage over the competition is gained by the LBC algorithm, which, however, still always achieves slightly worse results than the CSCS-MLP pair. It is particularly evident in the exact quality of the grading at each point where MLP converges much faster and leads to higher scores. Two groups can be observed further. Slightly better ARF and KUE and slightly worse AWE, AUE and WAE. The results for REA, which oscillate around the random classifier in each problem, will not be mentioned in further analysis.

For recurrent drift scenarios (Fig. 10) with the MLP as base classifier, more diverse results are achieved. In the case of gradual drift, SEA, AWE and WAE methods create

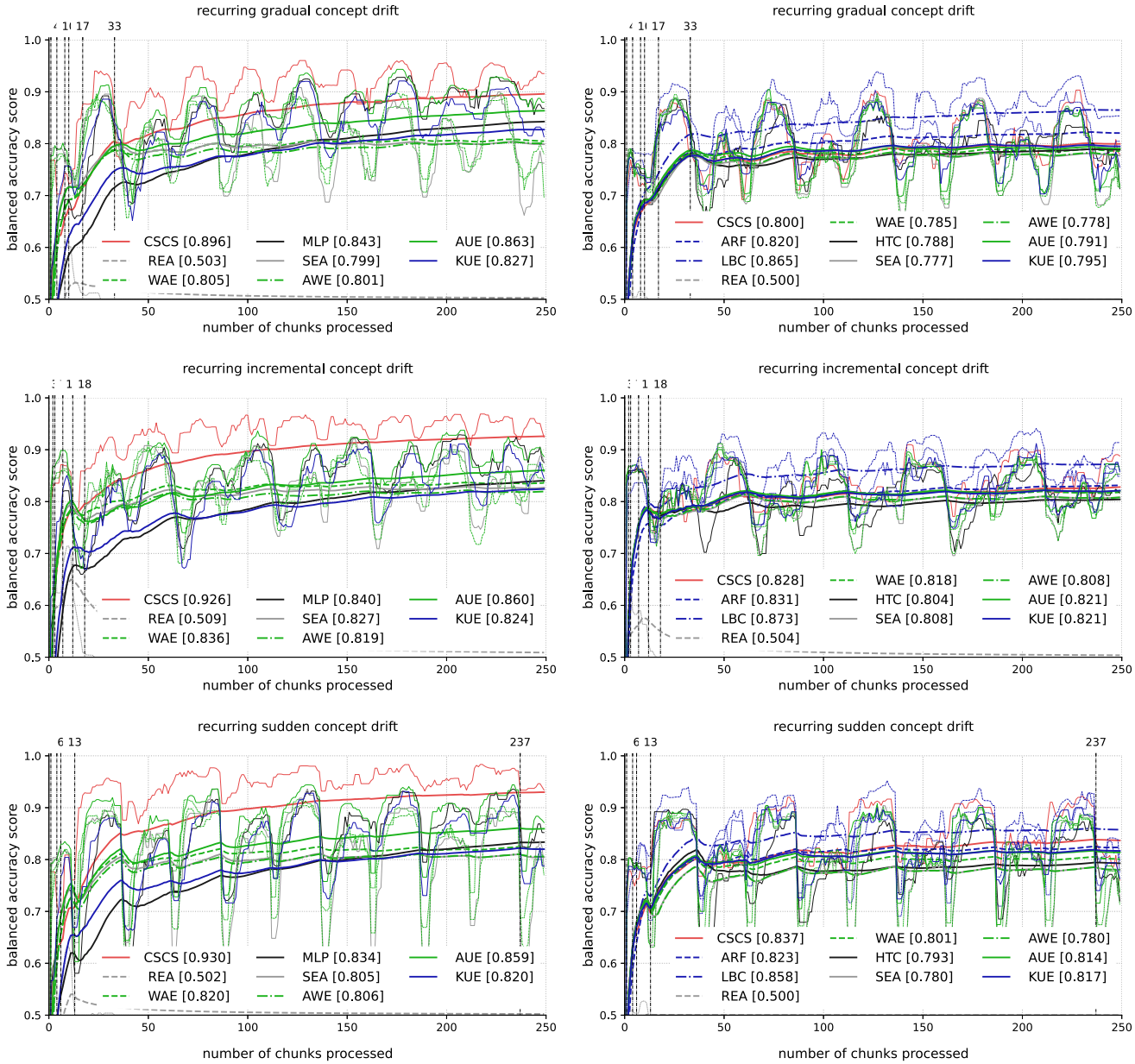


Fig. 10 *Balanced accuracy score* achieved by compared methods on exemplary - representative runs from three recurring drift dynamics for *Multi-Layer Perceptron* (left) and *Hoeffding Tree* (right)

the worst group of solutions, finally reaching about 80% of the balanced accuracy. The base MLP method is initially the worst. However, around the hundredth chunk, it exceeds the predictive capacity of the mentioned group and KUE, stabilizing at a quality level just below AUE and just above KUE. The AUE method, on the finite stream flow, shows its greater utility than the other reference methods and achieves the final quality with an average level of 86%. The CSCS algorithm, except for the initial phase of the first drift, here duplicates the efficiency achieved for nonrecurrent drifts, with (*a*) smaller quality drops throughout the stream course and (*b*) faster recovery after the drift. The results for incremental and sudden drift are similar. However, for sudden drift, after a good knowledge of both concepts contained in the stream after about 100 chunks, the CSCS algorithm gradually eliminates the phenomenon of quality loss after the drift occurs.

In the case of using HTC as the base classifier, the results are similar to those for non-recurring concept drifts. Again, we have a significant drop in CSCS quality to 80 percent, and again the LBC algorithm turns out to be the best in the comparison. Regardless of this, again the CSCS-MLP pair gives results noticeably higher than the *state-of-the-art* LBC-HTC.

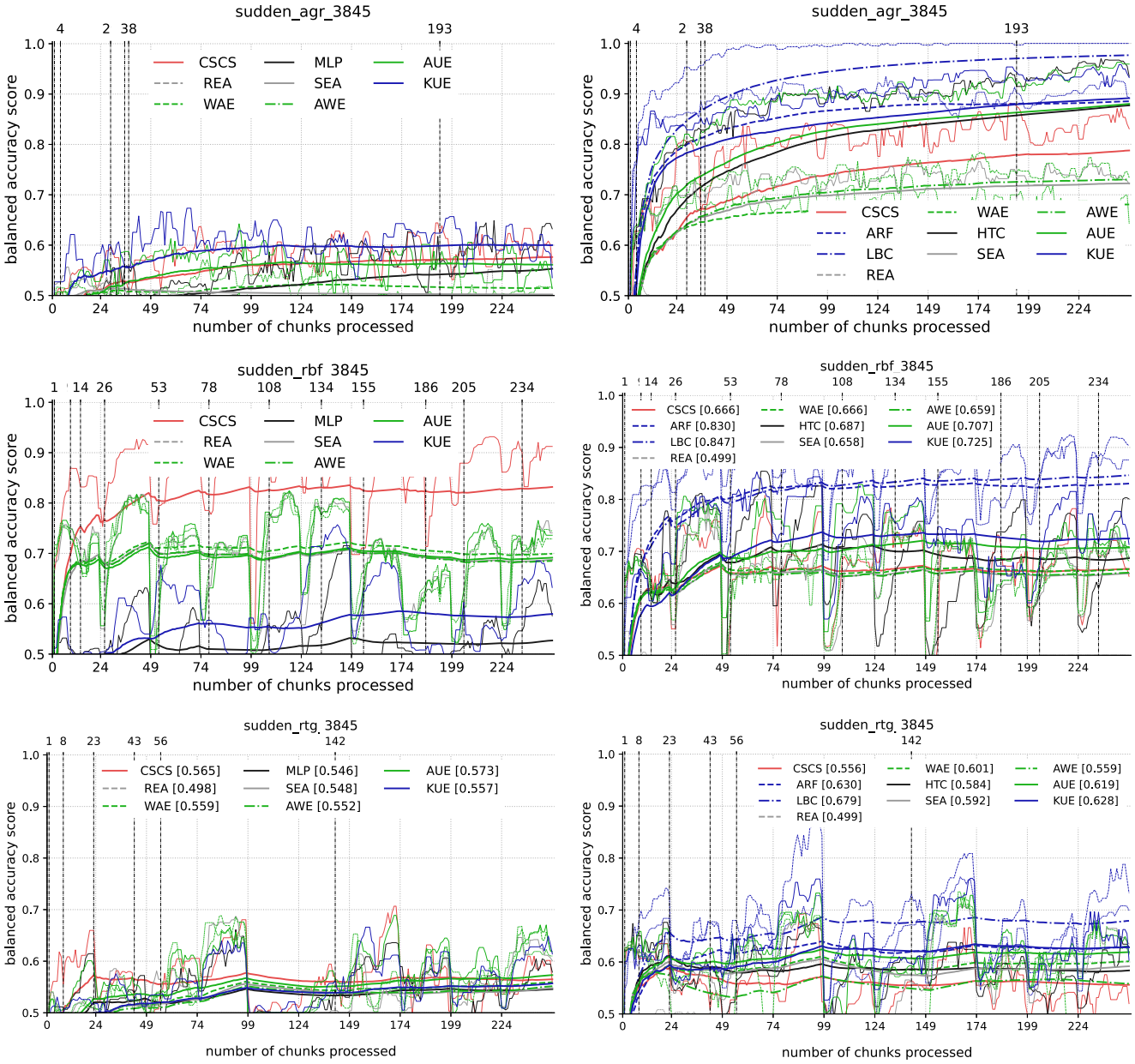


Fig. 11 *Balanced accuracy score* achieved by compared methods on exemplary - representative run from three selected scenarios generated by *MOA* for *Multi-Layer Perceptron* (left) and *Hoeffding Tree* (right)

The results achieved with the standard configuration of methods for flows generated with the use of the MOA packet are noticeably different (Fig. 11). As can be seen, the MLP base classifier achieves in each of these cases a much lower efficiency than HTC, which may explain the special popularity of the latter approach in most of the current research. Due to the low efficiency of neural networks, the LBC-HTC pair turns out to be by far the best solution in the entire rate, which in the case of the AGR generator gains even 10% advantage over the next, ARF-HTC, which achieves comparable generalization power for the RBF generator, but for RTG lays behind the LBC again.

The assumptions about the low efficiency of CSCS for the quality-based RTG and AGR generators were therefore confirmed. However, the behavior of the CSCS along with the MLP algorithm in the case of the RBF generator is interesting. Despite the final efficiency at a level much lower than that of the LBC, there is a significant 10% advantage of CSCS-MLP over other solutions using MLP. It is worth to recall here the observation from Fig. 7, that with about ten problem attributes (default RBF configuration), the HTC processing power demand is equal to the MLP chunk demand of 100 epochs. Meanwhile, the presented results relate to MLP with exactly 10 epochs on chunk.

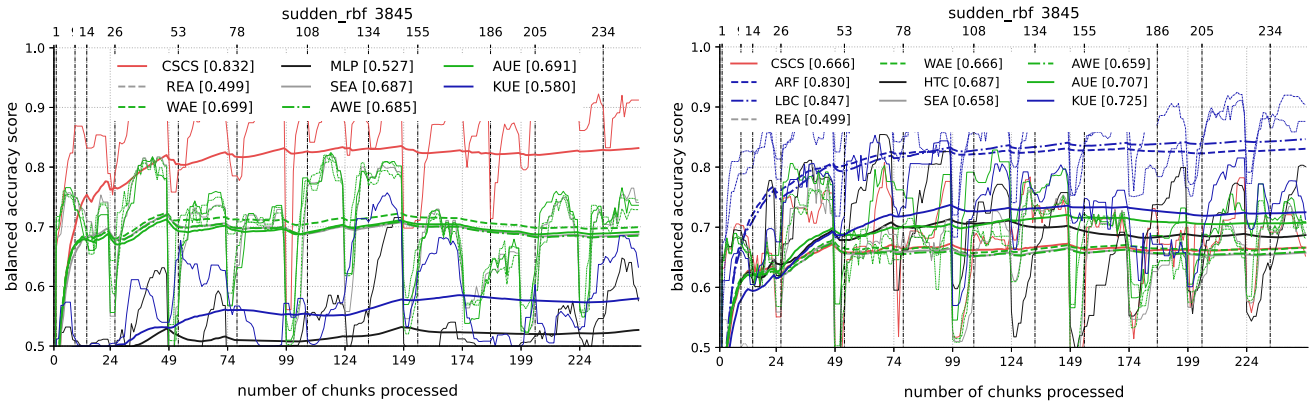


Fig. 12 *Balanced accuracy score* achieved by compared methods on exemplary - representative run from RBF scenario for *Multi-Layer Perceptron* with enhanced training time (left) and *Hoeffding Tree* (right)

Therefore, it would be potentially interesting to compare the effectiveness of HTC against the MLP of one hundred epochs.

This is shown in Fig. 12. As can be seen, after increasing the number of MLP epochs matching the computational demand with HTC, the results are quite different. The CSCS-MLP pair achieves results comparable to the ARF and only slightly weaker than the LBC. The AWE, AUE and WAE methods also benefited from a more complex base model, catching up with their HTC counterparts in terms of maximum efficiency, but showing higher learning dynamics. Only KUE model preserved its weak abilities. In connection with these observations, the MLP configuration with one hundred iterations on chunk was used for evaluation on data streams with real concepts (*Experiment 3*).

In addition to benchmarking the recognition efficiency, it is also important to verify the ability of the CSCS to identify new concepts. Figure 13 shows the concept identification capability in the prediction procedure of the CSCS algorithm for data streams with synthetic concepts generated with *stream-learn* package. The X-axis represents the stream course, and the Y-axis – the absolute identifier of the model included in the Π pool. The scatter plot superimposed on the plot space marks – with black dots – indicates which concept was used to predict which chunk. As in the case of Fig. 9, an additional upper X-axis marks the identification of chunks building the new model. In the background of the illustration, as in Fig. 2, the *concept change curve* of the data stream is highlighted.

As can be seen, the most problematic for the CSCS algorithm is the initial flow of the data stream, in which the identification of a new concept (based on the harmonic mean) is impossible due to the lack of comparative data from concepts other than the first one. In the neighborhood of chunk 25, where the second concept stabilizes, CSCS achieves the proper problem recognition capacity. Identifying new concepts here most often occurs near the end of a gradual or incremental drift or close to sudden drift. In the case of gradual and sudden drift, a momentary disturbance in identification is noticeable, which, after the end of the stable concept phase, temporarily indicates one of the historical models, but this quickly leads to the identification of a new concept and a stable shift of the entire recognition system to it. It is interesting to confirm the assumption in the description of the method that the transient phases of non-recurring incremental drift are very similar. Each transition phase was identified as a common concept #4. Nevertheless, the training phases of the identified model in the algorithm give only seemingly wrong selection, which in the course of drifts builds a transitional model capable of maintaining a higher predictive ability than *state-of-the-art* methods. It is confirmed by the observations made on Figs. 9 and 10.

Figure 14 shows the same analysis but for flows generated by MOA. As can be seen here, the assumptions about the low ability to identify concepts are confirmed in streams with the dominance of categorical attributes because for AGR and RTG, we can observe only a few detections. In the case of AGR streams, only the initial hyper-excitations and a single irrelevant detection downstream are observed. The graph for RTG streams presents slightly better recognition abilities – where there is a higher percentage of quantitative attributes – however, still not all significant changes are detected. Correct detection is achieved by CSCS in the case of RBF streams, which justifies the high CSCS-MLP result obtained after correcting the parameters of the neural network.

The observations made on the figures are also reflected in the numerical results presented in Tables 1 and 2. As we can read from it, the proposed CSCS-MLP method achieves the highest value of balanced accuracy in replication-average for each of the *stream-learn* scenarios. In all types of drift dynamics, it is competed only with the AUE method, which demonstrates greater discriminant power than other

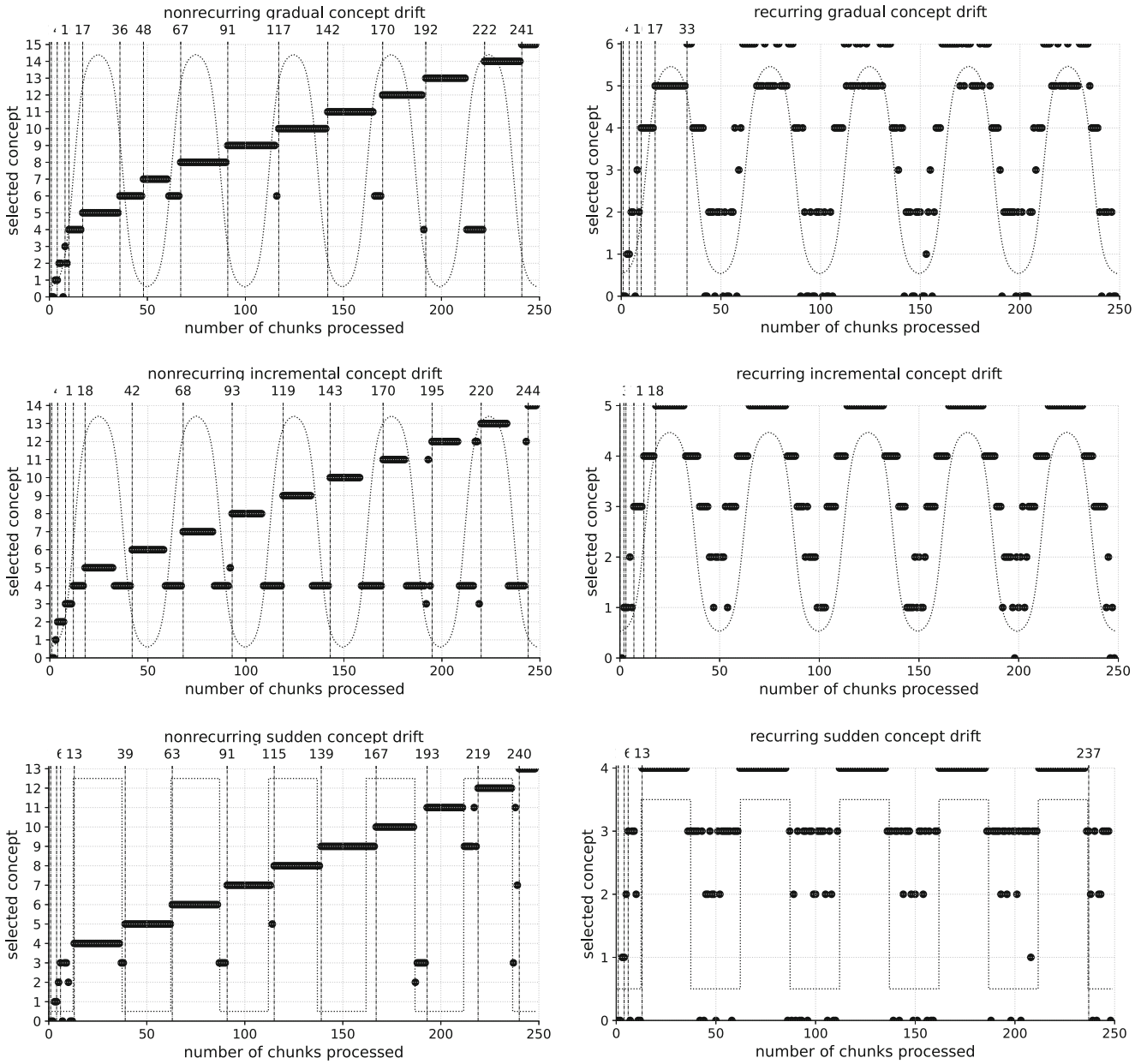


Fig. 13 Concept identification capability in the prediction procedure of the CSCS algorithm for data streams with synthetic concepts generated by *stream-learn* package

methods, but is still about seven percent worse than CSCS. In the case of recurrent drift, it is worth noting a fairly high result of the MLP base method, which shows its ability to adapt to several learnable concepts following the *lifelong-learning* paradigm. The SEA and REA methods, due to the lack of a model updating procedure, are not able to use this ability of the base classifier and, despite the ensemble approach, lead to ultimately worse recognition systems. The AUE algorithm, which is already capable of updating its models, allows a slight improvement over MLP. However, it is the CSCS algorithm – by its ability to dynamically switch between concepts – that makes the best use of this MLP property and achieves an average of 92 percent balanced accuracy on a stream with ten concept drifts.

In turn, solutions based on HTC show a higher generalization capacity for data synthesized by MOA generators. The difference here is particularly strong for AGR and RTG streams, in which the MLP was not able to significantly exceed the level of the random classifier. For the RBF generator, after adjusting the parameters of the base model (RBF*), a significant increase in the efficiency of neural networks can be seen, which in this case makes the CSCS-MLP pairing a method competitive with LBC-HTC and ARF-HTC.

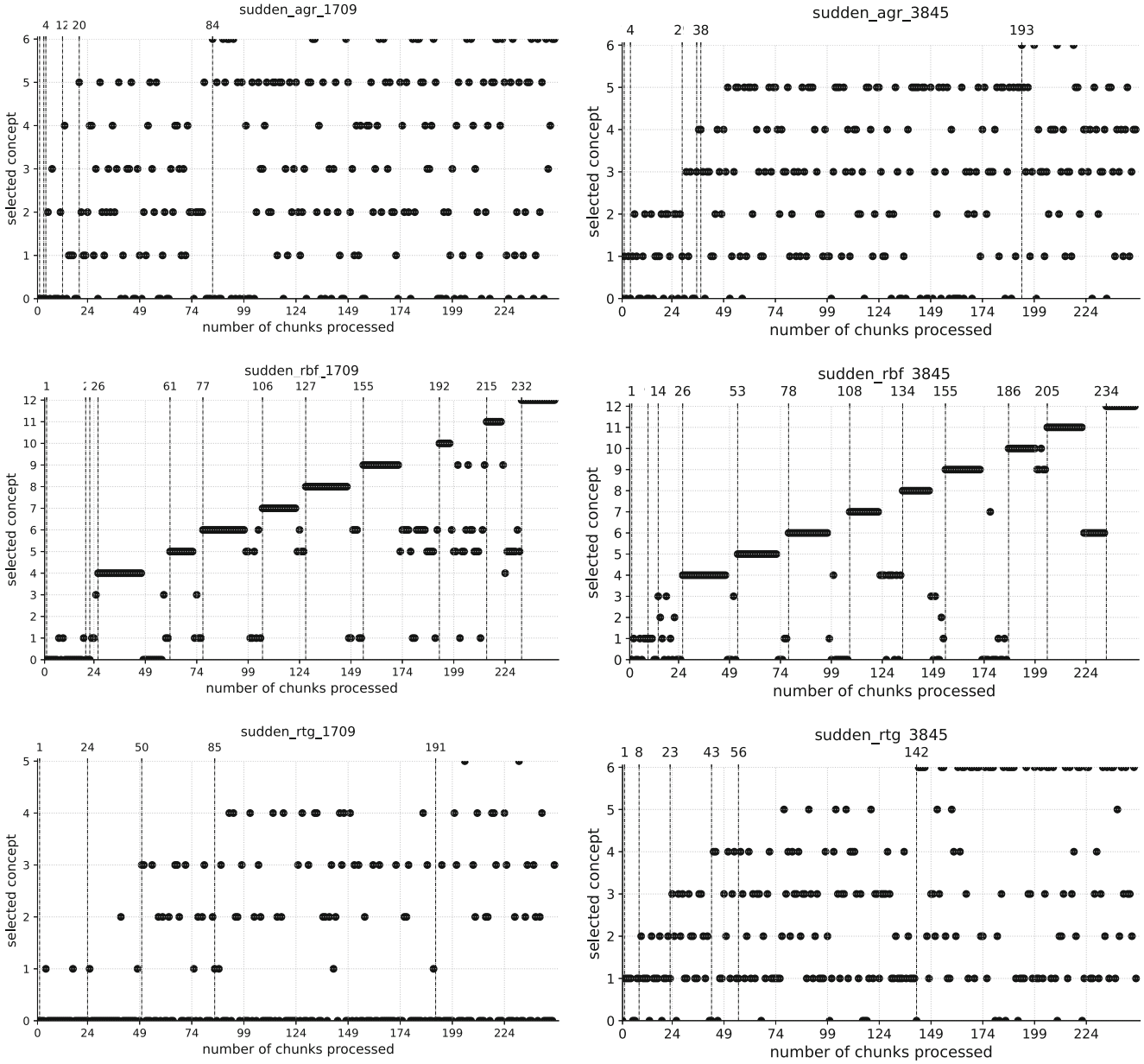


Fig. 14 Concept identification capability in the prediction procedure of the CSCS algorithm for data streams with synthetic concepts generated by *MOA* package

3.4 Experiment 3: evaluation on data streams with real concepts

The final experiment of the conducted evaluation is the analysis of the effectiveness of the compared methods in the processing of data streams with real concepts. Figure 15 presents obtained results analogous to those from Experiment 2. Again the X-axis shows the stream flow, and the Y-axis – accumulative balanced accuracy of the methods.

The basic observation here should be much greater diversity in the quality of various ensemble methods, which no longer group into hard-to-distinguish clusters in the case of real concepts. However, this is only apparent for MLP-based models, where HTC leads to slightly more consistent results. The different dynamics of the initial approach to neural network convergence are particularly interesting here, in which the CSCS algorithm shows the most significant dynamics. However, in some cases, as in the four-dimensional and seven-dimensional problem, the MLP model turns out to be the most effective in the presence of a first concept. However, this state is temporary, and as the concept changes, CSCS is gaining more and more advantages over the competition.

Table 1 Comparative analysis of ensemble methods for classification of data streams with synthetic concepts on *Multi-Layer Perceptron*

	Proposition	Baseline	Ensemble methods					
Drift dype	CSCS	MLP	SEA	REA	AWE	AUE	WAE	KUE
stream-learn generators								
gradual	0.871	0.765	0.770	0.511	0.765	0.810	0.777	0.769
incremental	0.883	0.759	0.796	0.519	0.788	0.816	0.803	0.776
sudden	0.894	0.752	0.778	0.514	0.775	0.814	0.792	0.770
recurring	0.916	0.820	0.781	0.514	0.777	0.843	0.790	0.806
nonrecurring	0.849	0.697	0.782	0.515	0.774	0.784	0.792	0.737
mean	0.883	0.759	0.781	0.514	0.776	0.814	0.791	0.771
MOA generators								
AGR	0.575	0.548	0.503	0.501	0.505	0.564	0.519	0.598
RBF	0.659	0.552	0.565	0.506	0.561	0.587	0.571	0.580
RBF*	0.836	*0.547*	*0.704*	*0.509*	*0.702*	*0.708*	*0.717*	*0.580*
RTG	0.578	0.538	0.544	0.500	0.548	0.570	0.552	0.555
mean	0.604	0.546	0.537	0.502	0.538	0.574	0.547	0.578

Underlined scores are the global best in comparison between MLP and HTC

With the MLP classifier, the AWE, WAE and AUE algorithms represents a uniform front, illustrated by green lines on the plots. They show a significantly higher quality than the KUE algorithm, but also – apart from the initial processing stage – they differ significantly from the CSCS algorithm. When using the HTC classifier, the ARF and LBC algorithms have the highest generalization ability. However, for several components of the stream (chunks 168-192 and 216-240 with four attributes and 192-240 with six attributes), HTC completely loses the potential for recognition, and all models based on it drop significantly to the level of a random classifier.

Interestingly, which was not noticeable in Experiment 2, in the presence of real concepts, the CSCS-MLP algorithm achieves its maximum recognition ability much faster than competing ensemble methods. Its learning curve for each

Table 2 Comparative analysis of ensemble methods for classification of data streams with synthetic concepts on *Hoeffding Tree*

	Prop.	Base.	Ensemble methods							
DT	CSCS	HTC	SEA	REA	AWE	AUE	WAE	KUE	ARF	LBC
stream-learn generators										
gra.	0.760	0.737	0.750	0.507	0.751	0.765	0.759	0.775	0.794	0.843
inc.	0.777	0.750	0.779	0.508	0.779	0.794	0.788	0.800	0.807	0.851
sud.	0.786	0.744	0.755	0.508	0.755	0.789	0.775	0.794	0.805	0.843
rec.	0.801	0.778	0.761	0.508	0.761	0.781	0.773	0.792	0.811	0.852
non.	0.748	0.709	0.762	0.508	0.763	0.784	0.775	0.788	0.794	0.840
mean	0.774	0.744	0.762	0.508	0.762	0.783	0.774	0.790	0.802	0.846
MOA generators										
AGR	0.799	0.901	0.721	0.502	0.727	0.895	0.678	0.872	0.890	0.979
RBF	0.675	0.708	0.675	0.511	0.675	0.725	0.685	0.756	0.840	0.865
RTG	0.559	0.587	0.608	0.501	0.581	0.634	0.618	0.634	0.636	0.684
mean	0.678	0.732	0.668	0.505	0.661	0.751	0.660	0.754	0.788	0.843

Underlined scores are the global best in comparison between MLP and HTC

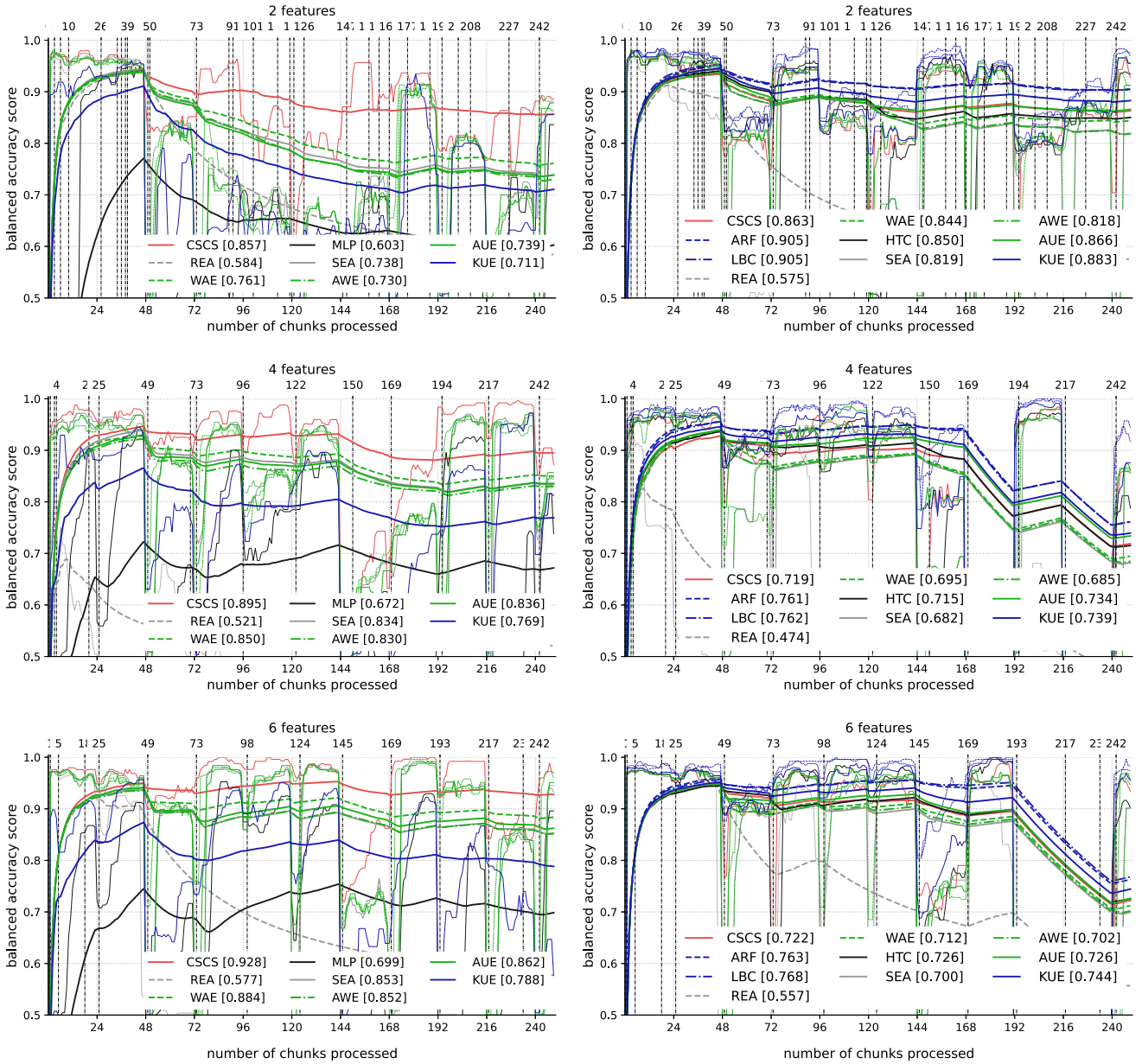


Fig. 15 *Balanced accuracy score* achieved by compared methods on runs from processing scenarios with real concepts

of the concepts contained in the stream is noticeably more dynamic and allows for faster achievement of neural network convergence. In most of the analyzed streams, the CSCS-MLP pairing is better than the other ensemble methods almost from the very beginning of processing, even despite the earlier problems in distinguishing between concepts.

Relatively low-dimensional real concepts also make it possible to evaluate the effectiveness of concept detection depending on the size of the representation of the chunk signature. Appropriate analysis can be made based on Fig. 16 showing the ability to identify the concept depending on the dimensionality of the stream. Problems with a small number of attributes build a small covariance matrix, so they should also hinder the proper recognition of their properties by the CSCS algorithm.

As can be seen in the case of the problem with two attributes, the CSCS algorithm is characterized by the high variation in concept identification, relatively often deciding to "jump" between models. However, the training of all models recognized as consistent with the current concept does not affect the final quality of recognition. Its only influence is observable in reducing the dynamics of the learning curve of such a recognition system. Moreover, it is done while still achieving a significantly better prediction than the competition. In the case of problems with four to six attributes, the recognition is unambiguous after the initial

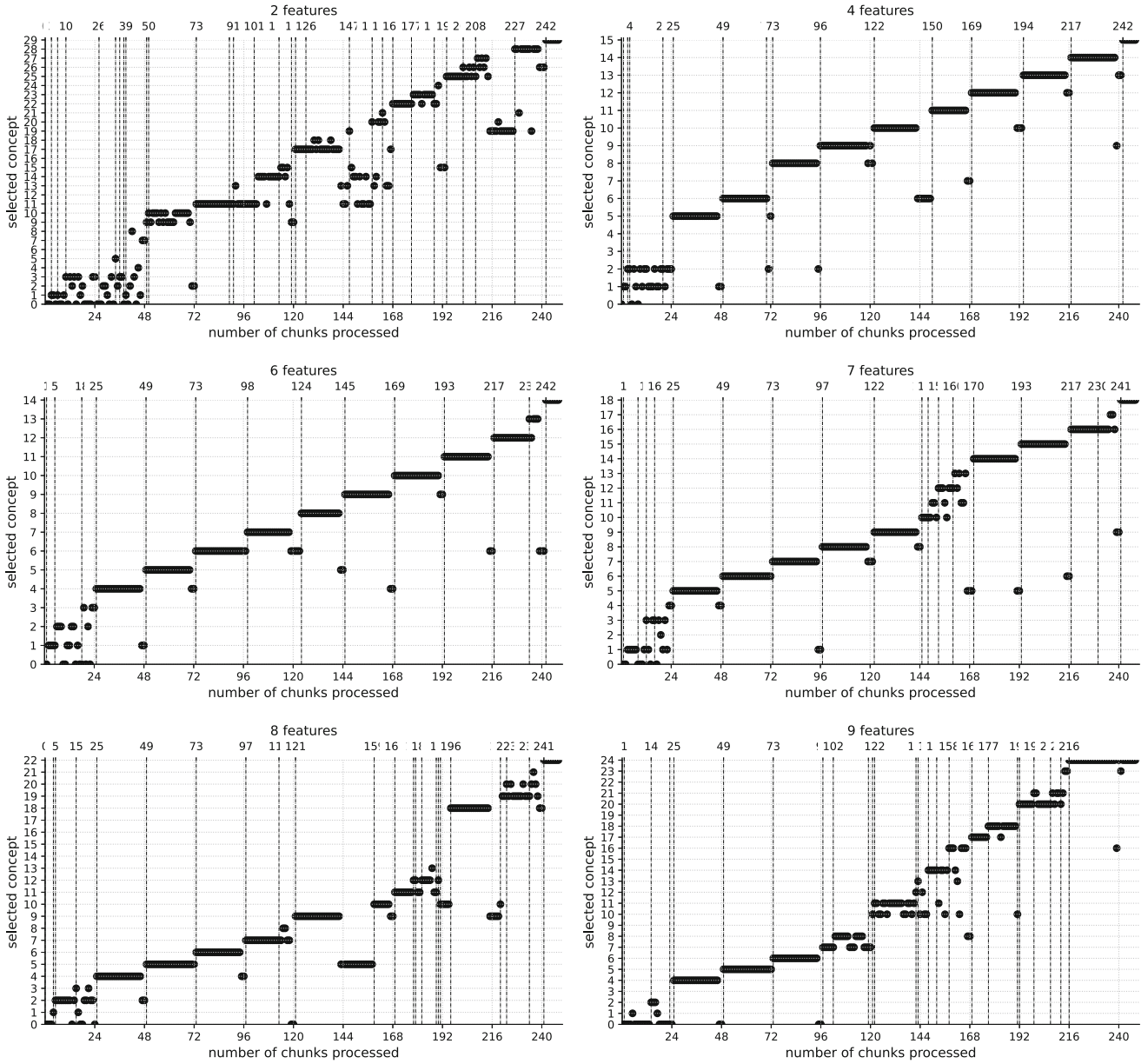


Fig. 16 Concept identification capability in the prediction procedure of the CSCS algorithm for data streams with real concepts

phase and allows a quick switch to a new concept. Apparent difficulties occur at a later stage of processing in the problems with seven, eight, and nine attributes. However, they are mainly due to the inclusion in the stream of concepts with similar dependencies between the attributes. The basis of which was the datasets available in the KEEL repository, diversified by combining classes from a multi-class problem into a dichotomy. However, it does not change the fact that also, in these cases, the CSCS algorithm shows a significant advantage in the quality of recognition over the competition.

This is confirmed by the numerical results contained in the Tables 3 and 4. Methods dedicated to HTC (ARF and LBC) achieve the best results only for low-dimensional data consisting of two or three attributes. For each of the other cases, the balanced accuracy achieved by CSCS-MLP turns out to be much higher than for *state-of-the-art* methods. The difference between other MLP models is, in some cases, more than 10 percent in favor of CSCS, and with the extreme differences between MLP and HTC (7 attributes), it is almost 20 percent.

Table 3 Comparative analysis of ensemble methods for classification of data streams with real concepts on *Multi-Layer Perceptron*

	Proposition	Baseline	Ensemble methods					
N. of features	CSCS	MLP	SEA	REA	AWE	AUE	WAE	KUE
2	0.860	0.605	0.741	0.587	0.733	0.742	0.764	0.713
3	0.915	0.642	0.806	0.574	0.803	0.809	0.833	0.749
4	0.899	0.674	0.837	0.523	0.833	0.839	0.853	0.772
5	0.908	0.635	0.814	0.549	0.819	0.826	0.856	0.781
6	0.932	0.701	0.856	0.580	0.855	0.866	0.887	0.791
7	0.919	0.649	0.841	0.574	0.836	0.850	0.869	0.767
8	0.913	0.659	0.830	0.596	0.833	0.840	0.861	0.782
9	0.902	0.634	0.807	0.557	0.808	0.811	0.838	0.747
mean	0.906	0.650	0.817	0.567	0.815	0.823	0.845	0.763

Underlined scores are the global best in comparison between MLP and HTC

Globally, when analyzing streams with real concepts in the cross-section from 2 to 9 dimensions, the CSCS-MLP turns out to be the best solution, which achieves 91 percent balanced accuracy. Subsequent methods are WAE-MLP reaching 85 percent and the AUE-MLP group, LBC-HTC, SEA-MLP, ARF-HTC, and AWE-MLP averaging about 82 percent balanced accuracy.

4 Discussion

The evaluation carried out in three experiments makes it possible to substantiate the statement about the advantage of updating the models already available in the pool of classifiers over the obligatory building of a new model on each new data chunk for a broad group of problems described by quantitative features. Importantly, this allows for a significant reduction in the processing time (*E1*) with a simultaneous advantage in the quality of prediction resulting from ignoring obsolete models, which was observed both for synthetic streams (*E2*) and for streams with real concepts (*E3*). Despite the increase in the calculation time of the base model for the processing of streams with real concepts, the CSCS method – even in such a case – is characterized by significantly lower complexity than the current *state-of-the-art* solutions (*E1*).

The approach proposed in the CSCS algorithm – based on keeping obsolete models in a pool of classifiers – is a very promising strategy for recurrent concept drift. Using the neural network as the underlying processing model allows to use of its natural property to adapt to the problem, and by identifying the concept, it does not desensitize the base model to subsequent drifts [56]. In addition, this approach also allows for the potential use of training strategies using classifiers that do not natively follow the drift of the concept. Finally, referring to Wolpert's theorem, it should be noted

Table 4 Comparative analysis of ensemble methods for classification of data streams with real concepts on *Hoeffding Tree*

	Prop.	Baseline	Ensemble methods							
d	CSCS	MLP	SEA	REA	AWE	AUE	WAE	KUE	ARF	LBC
2	0.867	0.854	0.823	0.577	0.822	0.869	0.847	0.886	0.909	0.908
3	0.904	0.893	0.865	0.566	0.866	0.907	0.876	0.911	0.939	0.938
4	0.722	0.718	0.685	0.476	0.687	0.737	0.698	0.742	0.764	0.765
5	0.810	0.810	0.750	0.531	0.748	0.815	0.777	0.827	0.853	0.858
6	0.725	0.729	0.702	0.559	0.705	0.729	0.715	0.747	0.766	0.771
7	0.688	0.688	0.678	0.430	0.674	0.696	0.685	0.705	0.726	0.728
8	0.791	0.769	0.768	0.509	0.766	0.795	0.784	0.805	0.836	0.833
9	0.686	0.679	0.674	0.531	0.674	0.697	0.693	0.718	0.746	0.747
mean	0.774	0.768	0.743	0.522	0.743	0.780	0.760	0.793	0.817	0.818

Underlined scores are the global best in comparison between MLP and HTC

that this method allows the potential extension of stream processing applications to problems in which *Hoeffding trees* have significant difficulties with achieving high and stable generalization abilities.

5 Conclusion

The *Covariance-signature Concept Selector* algorithm – proposition of this article – allowed, through the integration of paradigms typical for ensemble learning and drift detectors, to construct models characterized by greater efficiency in the data stream classification than methods that are currently *state-of-the-art* in the field in the recognized group of problems with quantitative features. However, this does not change that it is a solution characterized by several elements that constitute a potential development area.

In the first place, CSCS algorithm allows for efficient concept identification only for streams where drifts occur. In the case of stationary streams, typical of incremental learning, any CSCS identification of the concept will only lead to a slowdown of the training procedure. Therefore, the proposed algorithm cannot be considered recommended in the above situations. Similarly, the CSCS is not an optimal solution in the initial stage of processing, when only one stable concept is available. Instead, it should be recommended for data streams with relatively frequent concept changes and streams with recursive concept drift.

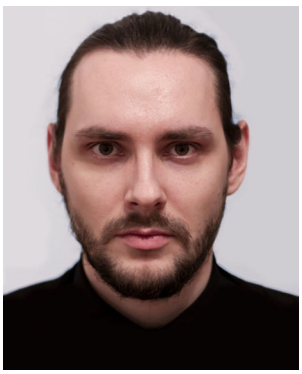
Additionally, in future work, it is possible to further reduce the processing time of the CSCS algorithm to or even below the SEA level by changing its implementation and introducing rather a matrix than the iterative computation of the variance of differences in the covariance matrix. Among additional future work, it will also be worth considering a proposed method for classifying *slow-drift* data streams. Due to its ability to recognize the transitional stages of a concept, this method could work for sets that have only one – gradual or incremental – concept drift starting its dynamics in the first chunk and ending in the last chunk.

Acknowledgements This work was supported by the Polish National Science Centre under the grant No. 2017/27/B/ST6/01325 as well by the statutory funds of the Department of Systems and Computer Networks, Faculty of Electronics, Wroclaw University of Science and Technology.

References

1. Alpaydin E (2020) Introduction to machine learning. MIT press
2. Wu Y, Chen Y, Wang L, Ye Y, Liu Z, Guo Y, Fu Y (2019) Large scale incremental learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 374–382
3. Köppen M (2000) The curse of dimensionality. In: 5th Online World conference on soft computing in industrial applications (WSC5), vol 1, pp 4–8
4. Khalid S, Khalil T, Nasreen S (2014) A survey of feature selection and feature extraction techniques in machine learning. In: 2014 Science and information conference. IEEE, pp 372–378
5. Ienco D, Bifet A, Žliobaitė I, Pfahringer B (2013) Clustering based active learning for evolving data streams. In: International conference on discovery science. Springer, pp 79–93
6. Berthelot D, Carlini N, Goodfellow I, Papernot N, Oliver A, Raffel C (2019) Mixmatch: A holistic approach to semi-supervised learning. arXiv:1905.02249
7. Zhou L, Pan S, Wang J, Vasilakos AV (2017) Machine learning on big data: opportunities and challenges. Neurocomputing 237:350–361
8. Žliobaitė I (2010) Learning under concept drift: an overview. arXiv:1010.4784
9. Gaber MM, Zaslavsky A, Krishnaswamy S (2007) A survey of classification methods in data streams. Data Streams, 39–59
10. Sobolewski P, Woźniak M (2013) Comparable study of statistical tests for virtual concept drift detection. In: Proceedings of the 8th international conference on computer recognition systems CORES 2013. Springer, pp 329–337
11. Ksieniewicz P (2021) The prior probability in the batch classification of imbalanced data streams. Neurocomputing 452:309–316
12. Komorniczak J, Zyblewski P, Ksieniewicz P (2021) Prior probability estimation in dynamically imbalanced data streams
13. Grzyb J, Klikowski J, Woźniak M (2021) Hellinger distance weighted ensemble for imbalanced data stream classification. J Comput Sci 51:101314
14. Ghazikhani A, Monsefi R, Yazdi HS (2013) Recursive least square perceptron model for non-stationary and imbalanced data stream classification. Evolv Syst 4(2):119–131
15. Zyblewski P, Sabourin R, Woźniak M (2019) Data preprocessing and dynamic ensemble selection for imbalanced data stream classification. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 367–379
16. Gama J (2012) A survey on learning from data streams: current and future trends. Progress Artif Intell 1(1):45–55
17. Manoj Kumar MV, Thomas L, Annappa B (2015) Capturing the sudden concept drift in process mining. Algorithms & theories for the analysis of event data (ATAED'15, Brussels, Belgium, June 22-23, 2015), p 132
18. Brzezinski D, Stefanowski J (2013) Reacting to different types of concept drift: the accuracy updated ensemble algorithm. IEEE Trans Neural Netw Learn Syst 25(1):81–94
19. Liu A, Zhang G, Lu J (2017) Fuzzy time windowing for gradual concept drift adaptation. In: 2017 IEEE International conference on fuzzy systems (FUZZ-IEEE). IEEE, pp 1–6
20. Krawczyk B, Woźniak M (2015) One-class classifiers with incremental learning and forgetting for data streams with concept drift. Soft Comput 19(12):3387–3400
21. Ramírez-Gallego S, Krawczyk B, García S, Woźniak M, Herrera F (2017) A survey on data preprocessing for data stream mining: Current status and future directions. Neurocomputing 239:39–57
22. Krawczyk B, Minku LL, Gama J, Stefanowski J, Woźniak M (2017) Ensemble learning for data stream analysis: a survey. Inform Fus 37:132–156
23. Kuncheva LI (2004) Classifier ensembles for changing environments. In: International workshop on multiple classifier systems. Springer, pp 1–15
24. Street WN, Kim Y (2001) A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining, pp 377–382

25. Wang H, Fan W, Yu PS, Han J (2003) Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining, pp 226–235
26. Brzeziński D, Stefanowski J (2011) Accuracy updated ensemble for data streams with concept drift. In: International conference on hybrid artificial intelligence systems. Springer, pp 155–163
27. Chen S, He H (2011) Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. Evolv Syst 2(1):35–50
28. Woźniak M, Kasprzak A, Cal P (2013) Weighted aging classifier ensemble for the incremental drifted data streams. In: International conference on flexible query answering systems. Springer, pp 579–588
29. Hoeffding W (1963) Probability inequalities for sums of bounded random variables. Amer. ž Statist Assoc J, 1329
30. Muallem A, Shetty S, Pan JW, Zhao J, Biswal B (2017) Hoeffding tree algorithms for anomaly detection in streaming datasets: a survey. J Inf Secur 8:4
31. Hulten G, Spencer L, Domingos P (2001) Mining time-changing data streams. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining, pp 97–106
32. Bifet A, Holmes G, Pfahringer B (2010) Leveraging bagging for evolving data streams. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 135–150
33. Oza NC, Russell SJ (2001) Online bagging and boosting. In: International workshop on artificial intelligence and statistics. PMLR, pp 229–236
34. Bifet A, Gavalda R (2007) Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM international conference on data mining. SIAM, pp 443–448
35. Gomes HM, Bifet A, Read J, Barddal JP, Enembreck F, Pfharinger B, Holmes G, Abdessalem T (2017) Adaptive random forests for evolving data stream classification. Mach Learn 106(9):1469–1495
36. Cano A, Krawczyk B (2020) Kappa updated ensemble for drifting data stream mining. Mach Learn 109(1):175–218
37. Gonçalves Jr PM, de Carvalho Santos SilasGT, Barros RobertoSM, Vieira DaviCL (2014) A comparative study on concept drift detectors. Expert Syst Appl 41(18):8144–8156
38. Barros RSM, Santos SGTC (2018) A large-scale comparison of concept drift detectors. Inf Sci 451:348–370
39. Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection. In: Brazilian symposium on artificial intelligence. Springer, pp 286–295
40. Baena-Garcıa M, del Campo-Ávila J, Fidalgo R, Bifet A, Gavalda R, Morales-Bueno R (2006) Early drift detection method. In: Fourth international workshop on knowledge discovery from data streams, vol 6, pp 77–86
41. Page ES (1954) Continuous inspection schemes. Biometrika 41(1/2):100–115
42. Alippi C, Roveri M (2006) An adaptive cusum-based test for signal change detection. In: 2006 IEEE international symposium on circuits and systems. IEEE, pp 4–pp
43. Severo M, Gama J (2006) Change detection with Kalman filter and cusum. In: International conference on discovery science. Springer, pp 243–254
44. Srivastava MS, Wu Y (1993) Comparison of Ewma, Cusum and Shiryayev-Roberts procedures for detecting a shift in the mean. Ann Stat, 645–670
45. Micevska S, Awad A, Sakr S (2021) Sddm: an interpretable statistical concept drift detection method for data streams. J Intell Inform Syst 56(3):459–484
46. Bach SH, Maloof MA (2008) Paired learners for concept drift. In: 2008 Eighth IEEE international conference on data mining. IEEE, pp 23–32
47. Bose A, Bhattacharjee M (2018) Large covariance and autocovariance matrices. CRC Press, USA
48. Park KI, Park M (2018) Fundamentals of probability and stochastic processes with applications to communications. Springer
49. Guyon I, Gunn S, Ben-Hur A, Dror G (2004) Result analysis of the nips 2003 feature selection challenge. Advances in Neural Information Processing Systems, 17
50. Ksieniewicz P, Zyblewski P (2020) stream-learn–open-source python library for difficult data stream batch analysis. arXiv:2001.11077
51. Zyblewski P, Sabourin R, Woźniak M (2021) Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams. Inform Fus 66:138–154
52. Hinton GE (1990) Connectionist learning procedures. 555–610
53. Chan TF, Golub GH, LeVeque RJ (1982) Updating formulae and a pairwise algorithm for computing sample variances. In: COMPSTAT 1982 5th symposium held at Toulouse 1982. Springer, pp 30–41
54. Domingos P, Hulten G (2003) A general framework for mining massive data streams. J Comput Graph Stat 12(4):945–949
55. Agrawal R, Imielinski T, Swami A (1993) Database mining: a performance perspective. IEEE Trans Knowl Data Eng 5(6):914–925
56. Ksieniewicz P, Woźniak M, Cyganek B, Kasprzak A, Walkowiak K (2019) Data stream classification using active learned neural networks. Neurocomputing 353:74–82



Pawel Ksieniewicz received the M.Sc. and Ph.D. degrees from the Wroclaw University of Science and Technology in 2013 and 2017, respectively. He is an Assistant Professor with the Wroclaw University of Science and Technology. Most of his papers concern classification of difficult data, focusing on processing data streams, multidimensional data representation, imbalanced data, and image processing.