Randomizing the Self-Adjusting Memory for Enhanced Handling of Concept Drift

Viktor Losing*, Barbara Hammer†, Heiko Wersing* and Albert Bifet‡
*HONDA Research Institute Europe, Carl-Legien-Str. 30, 63073 Offenbach am Main
†Bielefeld University, Universitätsstr. 25, 33615 Bielefeld
‡LTCI, Telecom ParisTech, Universite Paris-Saclay

***Abstract*—Real-time learning from data streams in non-stationary environments gains ever more relevance due to the exponentially increasing amounts of generated data. Recently, the Self-Adjusting Memory (SAM) was proposed, an algorithm able to robustly handle heterogeneous types on the basis of two dedicated memories for the current and former concepts that continuously preserve consistency with explicit filtering. Yet, since the algorithm is restricted to one memory architecture, the variety of possible alternatives is limited by design in favor of an overall model consistency. Moreover, it does not actively detect drift, thus adapting with a relatively high delay in case of abrupt changes. We propose a dynamic ensemble on the basis of the SAM algorithm, which is triggered by both, the inherent passive adaptation of SAM and active drift detection. Further, since SAM is based on the stable k-Nearest-Neighbor algorithm, we investigate multiple approaches to obtain a high diversity in the ensemble, resulting in an effective overall strategy. The increased computational demand is countered on the basis of a parallel implementation. We extensively evaluate the method on numerous benchmarks, where it consistently achieves superior results in comparison to state-of-the-art methods.**

***Index Terms*—concept drift, incremental learning, ensembles, data streams**

I. Introduction

An ever growing field of real-world applications generates data in streaming fashion at increasing rate, requiring large-scale and real-time processing. Streaming data is prevalent in domains such as health monitoring, traffic management, financial transactions, social networks [1] and is the foundation of the Internet of Things [2] technology. At the same time, a stronger focus on personalized services and products [3] requires the adaptation to single user behavior, habits and environments which naturally change over time. As these developments are inevitable merging, the demand for robust learning algorithms in the particularly challenging domain of data stream learning in non-stationary environments is increasing. Here, algorithms are facing a wide variety of possible patterns and scales of drift under strict limitations in terms of processing time and memory consumption.

Currently, many drift-learners are designed to handle certain types of change such as *abrupt*, *incremental* or *reoccurring* one, but often fail for others.

In real-world applications, unpredictable changes of different types are even concurrently occurring at various rates. Losing et al. proposed the Self-Adjusting Memory (SAM) in [4], an algorithm able to robustly handle heterogeneous types of concept drift without the requirement of prior assumptions about the task at hand. It creates dedicated k Nearest-Neighbor (kNN) [5] models for the current and former concepts and ensures consistency by a cleaning operation. However, the focus on consistency within one single architecture prevents SAM to store diverse, possibly partially contradicting concepts which can occur over time. Moreover, SAM solely adapts locally, whereas a global model adaptation can be particularly effective in case of abrupt drift.

Machine learning algorithms based on Bootstrap Aggregating (Bagging) such as the popular Random Forests [6] are one of the most powerful state-of-the-art learning methods [7], [8], and they are often used for learning under drift as well [9], [10]. By aggregating weak learners, ensembles can drastically improve on their single performance. In the field of non-stationary stream learning, ensembles offer an efficient way to handle concept drift because of their flexibility to selectively add and remove learners.

In this paper, we propose the SAM Ensemble (SAM-E) algorithm and combine the advantages of Bagging-ensembles with the SAM algorithm to boost the performance further. Thereby, the novel contributions of our work are as follows:

(i) Bagging requires a diverse ensemble, i.e. unstable base learners [11]. We investigate different ways to randomize SAM as it is based on the stable kNN algorithm.

(ii) An ensemble of SAMs deals with concept drift implicitly by means of the local adaptation of its base models. We also integrate a global adaptation by utilizing a drift detection to trigger the replacement of unsuitable base learners. This addition does not only increase the adaptation speed of the overall model, but also selectively preserves learners with a suitable parameterization.

(iii) We provide an open-source parallel implementation to facilitate the comparison for other researchers [1].

(iv) An extensive evaluation on artificial and real-world benchmarks is performed, covering various types and rates of concept drift. All benchmarks are publicly available, providing transparent and easily reproducible results. First, we analyze the effects of each proposed algorithmic building block. Subsequently, we compare our approach with state-of-the-art methods.

[1]The source code is available at https://github.com/vlosing/SAM-E.

II. Framework

Our focus is on data stream classification under supervised learning, i.e., given a feature vector $\mathbf{x} \in \mathbb{R}^n$, predict the target variable $y \in \{1, \dots, c\}$. Data-stream learning is often evaluated in the following setting: a potentially infinite sequence $S = (s_1, s_2, \dots, s_t \dots)$ of tuples $s_i = (\mathbf{x}_i, y_i)$ arrives one after another. As t represents the current time stamp, the learning objective is to predict the corresponding label y_t for a given input $\mathbf{x}_t$ by the previously learned model h_{t-1}, resulting in $\hat{y}_t = h_{t-1}(\mathbf{x}_t)$. Afterward, the true label is revealed and a loss determined. A new model h_t is generated on the basis of the current tuple s_t and the previous model h_{t-1}, before proceeding with the next sample: $h_t = \text{train}(h_{t-1}, s_t)$. The interleaved test-train error (up to time t) is given by $E(S) = \frac{1}{t} \sum_{i=1}^{t} \mathbb{1}(h_{i-1}(\mathbf{x}_i) \neq y_i)$. Hence, algorithms face the challenges of anytime model adaption, one pass learning, and restrictions due to a fixed ordering of the samples, often violating the assumption of independent and identically distributed data.

A. Concept Drift

Concept drift [12] occurs when the distribution $P_t(\mathbf{x}, y)$ changes for at least two time steps t_0 and t_1:

$$\exists\, \mathbf{x} : P_{t_0}(\mathbf{x}, y) \neq P_{t_1}(\mathbf{x}, y),$$

The joint distribution can also be written as $P_t(\mathbf{x}, y) = P_t(\mathbf{x}) P_t(y|\mathbf{x})$, where $P_t(\mathbf{x})$ is the distribution of the features and $P_t(y|\mathbf{x})$ the posterior probability of the classes. The term *real drift* specifies that the relation between observation and targets $P_t(y|\mathbf{x})$ changes over time. *Virtual drift*, also named covariate shift, refers to a change of the feature distribution $P_t(\mathbf{x})$ that does not affect the posterior of the classes. The pattern in which drift is taking place is often categorized either as *abrupt*, resulting in a severe shift within the distribution, e.g. caused by a malfunctioning sensor, or *incremental*, an evolving change over time, e.g. evoked by a slowly degrading sensor. Repeating patterns of change are termed as *reoccurring* concept drift.

III. Related work

Many approaches for stream learning with concept drift are either continuously growing incremental models [13]–[15] or utilize a sliding window to store a predefined number of recent examples [4], [16]. Algorithms of the first category often rely on incremental decision trees such as the Very Fast Decision Tree (VFDT) [17]. On the other hand, sliding windows are usually combined with instance-based learners such as kNN allowing to remove single examples in a straight-forward way.
A common technique to handle concept drift is to utilize a drift detection mechanism, which explicitly determines the time of change: ADaptive sliding WINdowing (ADWIN) [18] efficiently monitors the binary error history in a window starting from the last detected change. The window is repeatedly partitioned into two sub-windows of various size. Whenever the difference of their average error exceeds a threshold, depending on the size of the sub-windows and a confidence parameter, a change is detected and the older window dropped. Sliding window approaches often dynamically adapt the window size on basis of a drift detection. One example is the Probabilistic Adaptive Windowing (PAW) [19] which relies on the kNN classifier and ADWIN as drift detector. Furthermore, examples are randomly removed from the window to obtain a mix of recent and older instances. Incremental models that do not explicitly store examples typically reset their model as soon as a drift is detected. As such an approach erases all collected information until the detected time, the methods often rely on ensemble techniques to selectively reset only a few members, which preserves at least some information. Moreover, ensemble methods rank among the most powerful learning approaches [7]. Most ensembles are based on the online adaptation of Bagging proposed in [20]. Bifet et al. propose Leveraging Bagging (LVGB) [13], which utilizes higher instance weights to mitigate the slow learning of the VFDT. ADWIN is used as change detector for every tree, leading to the replacement of the worst classifier in case of drift. Gomes et al. proposed the Adaptive Random Forest (ARF) in [14], an ensemble of VFDTs, which uses ADWIN as change detector for every tree, leading to the replacement of the classifier in case of detected drift. Furthermore, it detects drift by means of two different sensitivity levels.
Methods which solely rely on drift detection are able to react quickly to abrupt drift, however, they struggle with incremental change, which may be not significant enough and remains undetected. Another weakness is that knowledge either slowly fades out of the model or is explicitly discarded. In cases where older data carries crucial information, such methods have to relearn these concepts from scratch.
The SAM algorithm realizes the idea of combining stable and reactive learners [21], [22] in a novel way leading to a robust handling of incremental and abrupt drift. Moreover, it explicitly preserves past information that is consistent with the present concept and, therefore, is able to handle reoccurring drift as well. Yet, due to a fixed data representation and smoothness of the underlying kNN model its intrinsic variability is limited. In particular, its definition of consistency strongly depends on the k parameter and controls the proportion of removed examples. In the following, we address this issue by proposing how to extend SAM to an ensemble which can naturally integrate different data representations (as given by different feature subsets) and smoothness of the classifier, as well as an active drift detection, to increase the adaptation speed of the overall method. Learning under concept drift was also considered in combination for imbalanced class distributions [23]–[25]. In our contribution, we do not explicitly tackle this challenge, however, various benchmarks that are evaluated in the experiments have skewed class distributions, providing a glimpse in the capabilities of the methods regarding this particular problem.

IV. Proposed algorithm

Before we describe the novel algorithm SAM-E, we briefly summarize the SAM algorithm, a complete and formal description is given in [4].

A. Self-Adjusting Memory (SAM)

SAM combines dedicated models for the current concept $P_t(\mathbf{x}, y)$ and all former ones $P_{t-1}(\mathbf{x}, y), \dots, P_1(\mathbf{x}, y)$ in such a way that the prediction accuracy is maximized. Two different memories are constructed: The Short-Term Memory (STM) summarizes data of the current concept and the Long-Term Memory (LTM) maintains knowledge of past concepts. Memories are represented by sets M_{ST}, M_{LT}, $M_{\text{C}} := M_{\text{ST}} \cup M_{\text{LT}}$. Each memory is a subset in $\mathbb{R}^n \times \{1, \dots, c\}$ of varying length, inducing a distance-weighted kNN classifier kNN : $\mathbb{R}^n \mapsto \{1, \dots, c\}$, referred to as $\text{kNN}_{M_{\text{ST}}}$, $\text{kNN}_{M_{\text{LT}}}$, $\text{kNN}_{M_{\text{C}}}$, where usually the Euclidean distance is used.

The prediction of SAM relies on the sub-model with the highest weight $(w_{\text{ST}}, w_{\text{LT}}, w_{\text{C}})$ and is defined for a given point $\mathbf{x}$ as:

$$\text{SAM} : \mathbf{x} \mapsto \begin{cases} \text{kNN}_{M_{\text{ST}}}(\mathbf{x}) & \text{if } w_{\text{ST}} \geq \max(w_{\text{LT}}, w_{\text{C}}) \\ \text{kNN}_{M_{\text{LT}}}(\mathbf{x}) & \text{if } w_{\text{LT}} \geq \max(w_{\text{ST}}, w_{\text{C}}) \\ \text{kNN}_{M_{\text{C}}}(\mathbf{x}) & \text{if } w_{\text{C}} \geq \max(w_{\text{ST}}, w_{\text{LT}}). \end{cases} \quad (1)$$

This model is incrementally adapted for every time t. Thereby, the adapted parameters are the size of the STM, the samples of the LTM, and the weights of the memories.

STM: The STM represents the current concept in a sliding window, which contains the most recent m examples. The length m is adapted by (i) adding the most recent example, and (ii) decreasing the length such that the resulting interleaved test-train error is minimized.

LTM: The LTM preserves all information which is not contained in the STM but may still be valuable in particular for reoccuring drift. Starting from the empty set, the samples points are obtained by two procedures: *(i) Cleaning and transfer:* Whenever the STM is reduced in size, information contradicting the current concept is filtered and the remaining information is transferred to the LTM. Filtering removes instances that are spatially close to those in the STM but have different labels. *(ii) Compression:* As soon as the size limit of the LTM is reached, information is condensed to a sparse knowledge representation via clustering, enabling the conservation of information for a long time period.

Weights: The weights w_{ST}, w_{LT}, w_C are representing the accuracy of the corresponding model on the recent data, where "recent" is defined by the size of the STM.

B. Self-Adjusting Memory Ensemble (SAM-E)

We describe the novel algorithm by breaking down all modifications that lead from a simple Bagging ensemble of SAM to the final algorithm SAM-E, that includes various randomization techniques and an active drift detection.
Typically, the success of Bagging mainly depends on accurate and diverse learners h^i, which are trained on bootstrap samples of the data, and averaged with a weighting according to its accuracy. Following the approach as introduced in [11], Bagging SAM refers to N independently trained SAM models SAM^i, $i = 1 \dots N$, which are trained separately with bootstrap samples chosen from the original training data. Every model is assigned a weight via $w_i := max(w^i_{\text{ST}}, w^i_{\text{LT}}, w^i_{\text{C}})$. Then the overall prediction function is given as the average

$$\hat{y}_t = \underset{\hat{c}\ \in \{1,\dots,c\}}{\arg\max} \sum_{i=1}^{N} w_i \cdot \mathbb{1}(\text{SAM}^i(\mathbf{x_t}) = \hat{c}), \quad (2)$$

where SAM^i is the prediction function of the ith submodel.

1) Online Bagging SAM – SAM-E$_{None}$: We use the online bagging approach as proposed in [20]. Concretely, a Poisson distribution is used as limit distribution of the multinomial distribution of Bagging, to determine the weight of each instance for each learner. In other words, at time step t, the ith learner, h^i_t, is adapted to the training sample $(\mathbf{x}_t, y_t)$ which is weighted with p, where p is distributed according to a Poisson distribution with parameter λ. In particular, the sample is dropped if $p = 0$. For kNN, we include the data point $(\mathbf{x}_t, y_t)$ whenever $p > 0$.
On average, 67% of the data are used by every learner if λ is set to 1. LVGB and ARF choose $\lambda = 6$, i.e. each learner uses 97% of the data. Even though this should lead to a lower diversity because all learners see more similar traning sets, it mitigates the slow learning speed of the VFDT, and outweighs any disadvantage. We also use $\lambda = 6$. In our setting, however, we do not use it to improve the learning speed of kNN, since instance-based models have naturally a high learning speed. Instead we target a high adaption speed to drift. In our method, the learners mostly handle drift themselves and a low λ leads to slower adaptation, since each of them has effectively less samples to react. An insightful analysis of the impact of diversity based on variations of λ is given in [26]. In contrast to the unstable decision tree algorithm, where Bagging alone creates enough diversity, kNN (and therefore SAM) is a stable method where additional diversity has to be induced. Hence, we propose further modifications that are essential for a performance gain as we will see in the experiments.

2) Varying the smoothness of kNN – SAM-E$_k$: The performance of kNN depends on the selected hyperparameter k [27] as it defines the smoothness of the classification prescription. For SAM, k plays an additional role: data in the LTM is kept consistent to the STM. Thereby, the consistency is established within an area of the feature space whose size depends on the neighborhood size k. Hence, large values k have the consequence that more data are checked for consistency leading to less preserved data in the LTM as compared to a smaller k value. We propose to increase diversity of the models SAM^i by initializing every new learner SAM^i with a randomized parameters k which is drawn as natural number from a uniform discrete distribution, $k \sim \mathcal{U}(a, b)$.

3) Varying the data representation – SAM-E$_{k,f}$: Another crucial role plays the applied metric. Dedicated metric learning schemes typically yield a significant computational burden [28]. Therefore, we rely on the simpler Random Subspace Method as introduced in [29]. Features are randomly selected to represent the data. Formally, a dimensionality $\hat{n} = \lceil \beta \cdot n \rceil$ with $\beta \leq 1$ is fixed. Then, $\hat{n}$ coefficients $(i_1, \dots, i_{\hat{n}})$ are

Algorithm 1 The SAM-E algorithm

```
Inputs:
    $S$ : data stream
    $N$ : ensemble size
    $a, b$ : bounds for randomization of $k$
    $\beta$ : relative size of the randomized subspace
    $r$ : proportion of replaced learners in case of detect drift
    $\delta$ : drift detection sensitivity threshold
    $\text{STM}_{\text{max}}, \text{LTM}_{\text{max}}$ : maximum bounds for the STM and LTM of
    SAM
Initialize:
    $C \leftarrow \text{CreateSAMs}(a, b, \beta, \text{STM}_{\text{max}}, \text{LTM}_{\text{max}}, N)$
    $W \leftarrow \{1/N, \dots, 1/N\}$
while $S$.hasNext() do
    $(x, y) \leftarrow S$.next()
    $\hat{y} \leftarrow \text{weightedPrediction}(x, C, W)$ (Equation 2)
    if driftDetected($\delta, \hat{y}, y$) then
        $C \leftarrow \text{replaceWorstClassifiers}(C, W, r)$
    $W \leftarrow \text{updateWeights}(C, y)$
    for all $i \in \{1, \dots, N\}$ do
        $p \leftarrow \text{Poisson}(\lambda = 6)$
        if $p > 0$ then
            $C_i$.train($x, y$)
```

sampled with replacement from $\{1, \dots, n\}$. Data are represented based on these coefficients only, i.e. input data have the form $\hat{\mathbf{x}} = (x_{i_1}, \dots, x_{i_n})$ where $\mathbf{x}$ comes from the data stream $\dots (\mathbf{x}_t, y_t) \dots$
By varying the subspace, the data representation changes significantly, since some coefficients are no longer regarded as relevant while others (if included as coefficients more than once) are emphasized. This variation also affects the consistency check of SAM because different examples are dubbed consistent, enabling the storage of different types of concepts.

4) Adding active drift detection – SAM-E$_{k,f,d}$: Even though each sub model is able to handle drift by its own, adaptation to abrupt drift faces some delay, and models cannot account for possibly changed representations or a varying noise level. We tackle these issues by adding an explicit drift detection mechanism on top of the ensemble. As soon as drift is detected on the performance of the ensemble, we replace the classifier with the highest STM classification error with a new one that starts learning from scratch and has a randomized configuration with respect to the model parameters k and $\hat{n}$. This has two benefits. First, it mitigates the inert adaptation to concept drift, a natural consequence of model aggregation. Second, it creates a competition within the ensemble where only members with a suitable parameterization for the current situation are lasting. This process is a form of adaptive hyperparameter selection.
In our experiments, we use ADWIN [18] as a proven drift detector. However, other drift detectors such as the Page Hinkley test [30] could be used as well. Each time a drift is detected we replace $\lceil rN \rceil$ learners, where $0 < r < 1$ is a hyperparameter controlling the number of replacements. The pseudo code of the method is given in Algorithm 1.

V. Experiments

For a comparison of different configurations of SAM-E, we vary the degree of randomization to investigate its effects on the classification performance and diversity. Afterward, we evaluate the parallel versus the sequential implementation in terms of run time and Ram-hours. Finally, we compare SAM-E with state-of-the-art methods.
To test for significant differences, we rely on the approach suggested in [31]. Concretely, we use the non-parametric, rank-based Friedman test with $\alpha = 0.05$. If the null hypothesis is rejected, we proceed with the Nemenyi post-hoc test to identify the algorithms with significant differences. All experiments are executed within MOA. We use a cluster consisting of 24 Intel Xeon 2.60GHz cores with 32 GB RAM to perform the experiments.

TABLE I
Characteristics of the considered data sets. ART=Artificial, RW=Real-World, A=Abrupt, I=Incremental, R=Real, V=Virtual. As suggested in [13], we sort the instances of the *Poker* dataset by rank and suit to generate virtual drift.

Data set	#Samples	#Feat.	#Class	Type	Drift type
SEA Concepts	50K	3	2	ART	A-R
Rot. Hyperplane	200K	10	2	ART	I-R
Moving RBF	200K	10	5	ART	I-R
Inter. RBF	200K	20	15	ART	A-R
Moving Squares	200K	2	4	ART	I-R
Transient Chessb.	200K	2	8	ART	A-V
Random Tree	200K	200	25	ART	None
LED-Drift	200K	24	10	ART	A-R
Mixed Drift	600K	2	8	ART	various
Poker	829.2K	10	10	ART	A-V
Outdoor	4K	21	40	RW	?
Spam	9.3K	40K	2	RW	?
Weather	18.1K	8	2	RW	?
Electricity	45.3K	5	2	RW	?
Rialto	82.2K	27	10	RW	?
Airline	539.3K	7	2	RW	?
Cover Type	581K	54	7	RW	?
PAMAP	2.7M	52	18	RW	?
KDD99	4.9M	41	23	RW	?

A. Datasets

We use an exhaustive set of benchmarks for evaluation. These consist of artificial datasets with known types of drift and real-world data where such characteristics are unknown. Table I gives further information regarding the varying number of dimensions, classes and instances. Due to lack of space we omit a detailed description and refer the interested reader to [4], [14].

B. Evaluation of variants of SAM-E

First, we analyze the effects of the suggested randomizations as well as the top-level drift detection on the classification performance. Table II lists all compared variants and the corresponding error rates are given in Table III. A single SAM classifier achieves on the datasets an average error rate of 15.38. As it can be seen Bagging alone, which only randomizes over the presented data points, only slightly improves the performance with an average rate of 14.77. This confirms that additional randomization is required for kNN-based ensembles. In our experiments, the performance improves with more randomized components. However, obviously the randomization of each component, such as the size of the random subspace, can be adjusted as well. In general,

TABLE II
THE COMPARED VARIATIONS OF THE ALGORITHM WITH DIFFERENT DEGREE OF RANDOMIZATION AS WELL AS WITH DRIFT DETECTION AND WITHOUT.

Abbr.	Randomizing K	Random subspace	Drift detection
SAM-E$_{\text{None}}$	✗	✗	✗
SAM-E$_{\text{k}}$	✓	✗	✗
SAM-E$_{\text{k,f}}$	✓	✓	✗
SAM-E$_{\text{k,f,d}}$	✓	✓	✓

TABLE III
INTERLEAVED TEST-TRAIN ERROR RATES OF DIFFERENT VARIATIONS OF SAM-E WITH AN ENSEMBLE SIZE OF $N = 10$. THE EXPERIMENTS WERE REPEATED 30 TIMES. THE BEST RESULTS ARE MARKED IN BOLD.

Data set	SAM-E$_{\text{None}}$	SAM-E$_{\text{k}}$	SAM-E$_{\text{k,f}}$	SAM-E$_{\text{k,f,d}}$
SEA Concepts	12.61±0.03	12.31±0.22	12.28±0.09	**12.28**±0.07
Rot. Hyperplane	13.12±0.00	13.49±1.06	**12.41**±0.21	12.49±0.71
Moving RBF	12.02±0.00	**11.47**±0.08	11.98±0.11	11.86±0.09
Inter. RBF	3.32±0.01	**3.08**±0.03	3.37±0.01	3.30±0.01
Moving Squares	**2.41**±0.00	3.12±0.58	2.47±0.97	2.47±0.25
Transient Chessb.	10.92±0.01	10.5±0.26	**10.08**±0.14	10.30±0.09
Random Tree	35.36±0.01	34.47±0.05	32.72±1.19	**32.72**±0.77
LED-Drift	43.22±0.05	43.46±1.63	37.52±2.81	**35.48**±2.61
Mixed Drift	11.98±0.01	11.58±0.03	11.6±0.15	**11.58**±0.02
Poker	15.82±0.00	12.59±0.34	15.03±4.14	**8.79**±0.44
Artificial ∅	16.11	15.94	14.94	**14.13**
Outdoor	11.02±0.08	**8.48**±0.32	8.98±0.26	9.25±0.29
Weather	21.91±0.04	**21.6**±0.07	21.81±0.08	21.41±0.16
Electricity	17.41±0.02	**15.36**±0.25	16.66±0.23	16.36±0.19
Rialto	18.06±0.01	**15.65**±0.22	16.64±0.36	15.80±0.16
Airline	39.12±0.03	38.88±0.13	37.53±0.73	**35.51**±0.16
Cover Type	5.66±0.01	**3.78**±0.40	8.1±0.24	4.69±0.36
PAMAP	0.02±0.00	0.02±0.00	0.02±0.00	**0.02**±0.00
SPAM	6.67±0.09	**5.37**±0.24	5.38±0.31	5.61±0.23
KDD99	0.01±0.00	0.01±0.00	0.01±0.00	**0.01**±0.00
Real world ∅	13.57	12.17	13.02	**12.07**
Overall ∅	14.77	13.96	13.93	**13.15**
Overall ∅ rank	3.47	2.13	2.45	**1.95**

Nemenyi significance: $\{$SAM-E$_{\text{k}}$, SAM-E$_{\text{k,f,d}}\} \succ$ SAM-E$_{\text{None}}$

the trade-off between the degree of randomization and the performance of the overall classifier mainly depends on the size of the ensemble, where large ensembles are able to exploit a higher degree of randomness, as well as the specific dataset. Most of the dimensions (17 out of 24) of *LED-Drift* are random noise. The comparably low error rate of SAM-E variants incorporating random subspaces for this dataset suggests that random projection mitigates the susceptibility to noisy dimensions, one major weakness of kNN models. The comparison of SAM-E$_{\text{k,f,d}}$ and SAM-E$_{\text{k,f}}$ allows the evaluation of the drift detection on top of the ensemble. Active drift detection turns out nearly always beneficial. One reason is the increased variability of the algorithm in case of drift, enabling a faster adaptation to the current concept. The versions SAM-E$_{\text{k,f,d}}$ and SAM-E$_{\text{k}}$ deliver results that are significantly better than simple Bagging (SAM-E$_{\text{None}}$).

Figure 1 depicts on the left the temporal course of the error rate for some data sets. Interestingly, in the most cases, the advantage of SAM-E$_{\text{k,f,d}}$ compared to the other variations increases

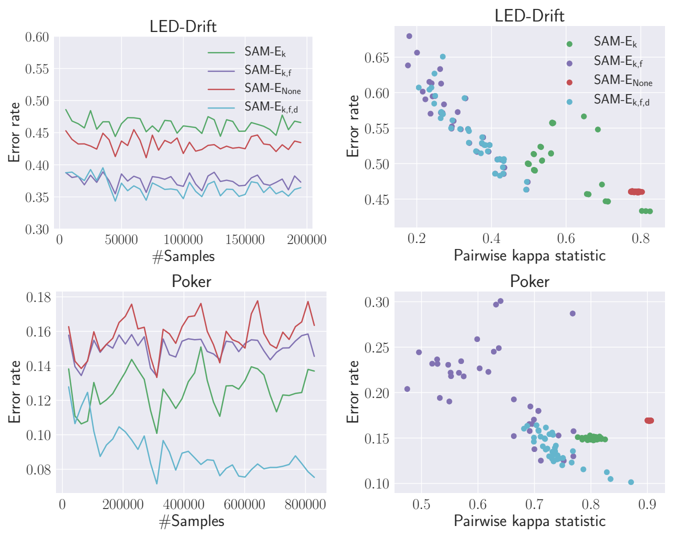

Fig. 1. The temporal course of the error rate as well as corresponding kappa-error diagrams. High diversity (low kappa statistic) coupled with a low error rate result in the best classification performance.

over time. Corresponding kappa-error diagrams [32] are shown on the right of Figure 1. These diagrams are commonly used to inspect the diversity of ensembles. The pairwise kappa-statistic is plotted against the average classification performance of both learners. Highly diverse learners (low pairwise kappa-statistic) with a low error rate lead to the best classification performance of the ensemble. Most of the time SAM-E$_{\text{k,f,d}}$ is able to achieve the best compromise. The relatively high kappa-statistics of SAM-E$_{\text{None}}$ attest a low diversity, which limits the leeway to reduce the error rate beyond those of the single classifiers. It clearly shows that Bagging alone is not enough to create diversity among kNN based classifiers. One example is the *LED-Drift* data set where its single learners have the lowest error rate, but the classification error of the ensemble is comparatively high.

The replacement of the worst learners, triggered by the drift-detection, decreases the error further which is especially pronounced in the task *Poker*. Due to the overall superiority of SAM-E$_{\text{k,f,d}}$, further experiments are solely based on it and we refer to this variant as SAM-E in the following.

1) Speedup of the parallel implementation: As Bagging yields completely independent learners, we provide a parallel implementation to speed-up the method. We measure the used resources by the sequential and parallel implementation in terms of average run-time and RAM-hours for all data sets. One RAM-hour equals one GB of RAM deployed for one hour. We provide results for different ensemble sizes. The interleaved test-train processing enforces the aggregation after each example and especially a completed training of the previous instance. Therefore, threads are very short and create a large overhead. However, this scheme is only necessary for the evaluation. In real-world application, it is often tolerable to buffer instances to small chunks and process them at once, leading to a reduced overhead. We also consider such a scheme

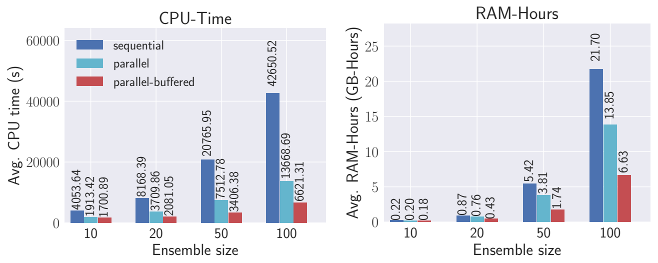

Fig. 2. Comparison of the parallel and sequential implementation in terms of average run-time and RAM-hours. Relaxing the test-train scheme to buffering 100 instances clearly increases the gain further.

and buffer chunks of 100 instances. Figure 2 depicts the results. The parallel implementation halves the run-time for an ensemble size of 10 and achieves a speedup of 3 for the largest tested ensembles with $N = 100$. The buffering mechanism is effective and doubles the gained speed-up. It can be seen that the processing time is significantly increased with additional classifiers. Hence, the parallel implementation is only partially able to counteract the increased computational demands of the ensemble.

C. *Comparison with state-of-the-art methods*

We compare SAM-E to other ensemble methods specialized for data stream learning with concept drift and also include single classifiers such as SAM itself and VFDT. Table IV lists all algorithms as well as relevant hyperparameter settings. The SAM algorithm as well as each classifier in SAM-E were allowed to store 1000 samples but never more than 10% of the whole data set.
Learning under concept drift requires the highest degree of robustness. The temporal dependency of the data forbids a straight-forward optimization of the algorithmic meta-parameters on a separate validation set. Optimizing those in hindsight on the whole data would lead to overfitting and has very limited practical relevance. Hence, it is common practice to evaluate the methods out-of-the-box in their default configuration, chosen by the original author, without dataset-specific adaptation. We performed the experiments according to this practice. SAM-E default configuration for all experiments is as follows:

- k is uniformly drawn from the range $[1, \dots, 7]$.
- The random subspace of each learner $\hat{X} \in \mathbb{R}^{\hat{d}}$ is set to use 70% of the original number of features, i.e. $\hat{d} = 0.7 \cdot d$.
- Each time ADWIN fires, 10% ($r = 0.1$) of the worst performing learners are replaced, where the ensemble size is 10 or 100, respectively.

These parameters were set on basis of preliminary experiments on the artificial datasets *4CRE-V1*, *FG-2C-2D* published in [33].

1) Classification performance: The error rates are listed in Table V. SAM-E achieves the best classification performance on average. The fact that the single SAM algorithm delivers the second best results highlights its effectiveness for concept drift. SAM-E outperforms the single SAM algorithm in case of the datasets *Random Tree*, *LED-Drift*, *Outdoor*, and *Poker* quite distinctly. The two comparably poor performances of SAM-E in the tasks *Random Tree* and *LED-Drift* are due to the susceptibility of kNN based approaches in regard to noisy dimensions. However, the random subspaces enable SAM-E to mitigate the problem in comparison to the single SAM algorithm.
Both tree ensembles ARF and LVGB deliver clearly worse results, mostly due to their comparably slow learning speed (*Outdoor*) or limited adaptation ability in case of fast incremental drift (*Moving Squares*). Furthermore, they struggle with reoccurring drift as shown by the results for *Transient Chessboard* and *Rialto*. They are dominating at the *Random*

TABLE IV
THE COMPARED ALGORITHMS AND THEIR HYPERPARAMETER. ENSEMBLES WERE EVALUATED WITH 10 AND 100 MEMBERS. WE USED A WINDOW OF 1000 SAMPLES AND k WAS SET TO 5 FOR KNN-BASED METHODS. IN CASE OF SLIDING WINDOW APPROACHES, THE MAXIMUM WINDOW SIZE SHOULD BE SET AS HIGH AS IT IS ACCEPTABLE IN REGARD TO MEMORY AND PROCESSING-TIME DEMANDS. WE DISCUSS THE EFFECTS OF DIFFERENT WINDOW-SIZE PARAMETERS EXTENSIVELY IN [4].

Abbr.	Classifier	Parameter
VFDT	Hoeffding Tree (VFDT)	-
SAM	Self-Adjusting Memory with kNN	$w = 1000$
LVGB	Leveraging Bagging with VFDT	$n = \{10, 100\}$
ARF	Adaptive Random Forest with VFDT	$n = \{10, 100\}$
SAM-E	Self-Adjusting Memory Ensemble	$n = \{10, 100\}$ $w = 1000$

TABLE V
INTERLEAVED TEST-TRAIN ERROR RATES ACHIEVED BY SINGLE CLASSIFIERS AND ENSEMBLE MODELS. ENSEMBLES CONSISTED OF $N = 10$ MEMBERS. WE REPEATED THE EXPERIMENTS 30 TIMES FOR NON-DETERMINISTIC APPROACHES.

Data set	VFDT	SAM	ARF	LVGB	SAM-E
SEA Concepts	15.16	13.22	11.68±0.06	**11.68**±0.07	12.28±0.07
Rot. Hyperplane	15.02	15.22	17.35±0.15	12.73±0.02	**12.49**±0.71
Moving RBF	66.27	12.10	34.02±0.17	45.62±0.15	**11.86**±0.09
Inter. RBF	74.71	3.27	**2.68**±0.04	10.08±0.94	3.30±0.01
Moving Squares	66.73	2.64	36.84±1.49	11.74±0.03	**2.47**±0.25
Transient Chessb.	45.24	11.26	26.30±0.17	14.69±6.22	**10.30**±0.09
Random Tree	10.36	37.05	8.71±1.49	**3.93**±0.09	32.72±0.77
LED-Drift	26.30	45.99	27.39±0.33	**26.13**±0.02	35.48±2.61
Mixed Drift	55.42	12.27	19.87±0.06	25.97±0.10	**11.58**±0.02
Poker	25.88	16.86	19.23±0.17	17.93±0.40	**8.79**±0.44
Artificial ∅	40.11	16.99	20.41	18.05	**14.13**
Outdoor	42.68	11.58	29.70±2.03	39.28±0.25	**9.25**±0.29
Weather	26.49	22.31	21.87±0.46	22.18±0.08	**21.41**±0.16
Electricity	29.00	17.58	21.13±0.50	17.58±0.18	**16.36**±0.19
Rialto	76.19	18.27	24.08±0.10	40.46±0.07	**15.80**±0.16
Airline	34.94	39.84	**34.20**±0.11	36.89±0.02	35.51±0.16
Cover Type	21.85	5.76	8.33±0.03	8.54±0.06	**4.69**±0.36
PAMAP	1.22	0.02	0.03±0.00	0.11±0.01	**0.02**±0.00
SPAM	19.09	7.00	8.18±0.42	7.35±0.31	**5.61**±0.23
KDD99	0.10	0.01	0.03±0.00	0.03±0.00	**0.01**±0.00
Real world ∅	27.95	13.60	16.39	19.16	**12.07**
Overall ∅	34.35	15.38	18.51	18.57	**13.15**
Overall ∅ rank	4.47	2.76	3.00	3.08	**1.68**

Nemenyi significance: SAM-E $\succ$ VFDT

Tree task, which is designed for tree-based methods, as the task is to learn a tree-based model.
Figure 3 depicts the temporal course of the classification performance for some data set as well as corresponding kappa-error diagrams. SAM-E is able to outperform the other

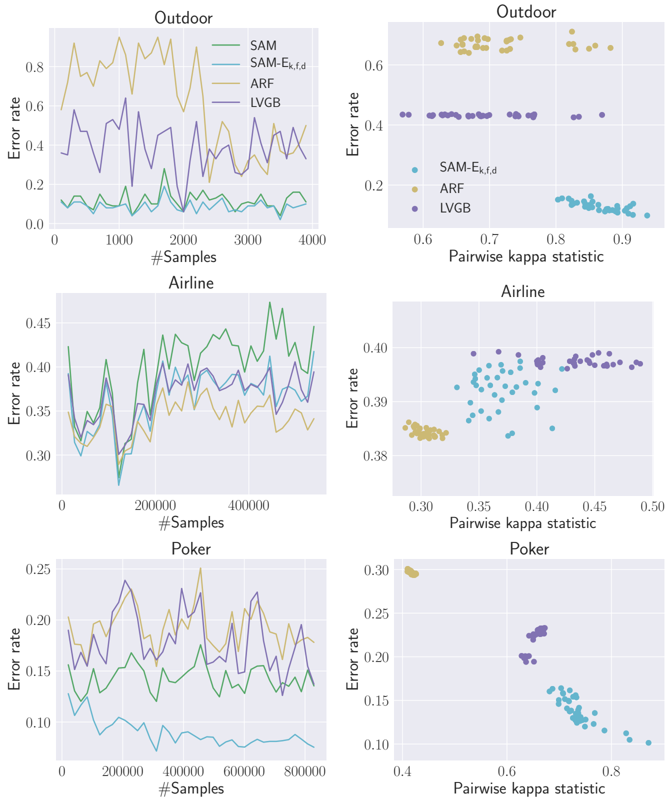

Fig. 3. The temporal error rate (on the left) as well as corresponding kappa-error diagrams (on the right). For the sake of clarity, only the four best methods are depicted.

methods due to the higher accuracy of its base classifiers. Its diversity is comparable to those of the tree ensembles, creating enough leeway for model aggregation to improve on the single learner performance.
The influence of some metaparameters are depicted in Figure 4. Naturally, the importance of the values depend on the particular dataset. Nonetheless, our default setting ($\beta = 0.7$, $b = 7$) that has been determined in preliminary experiments appears to be a reasonable choice accross different datasets.

2) Run time: The measured run time is given in Table VI. Tree-based methods are most of the times clearly faster than those relying on kNN. Particularly, in case of data sets with high dimensions such as *SPAM*, the evaluation complexity $\mathcal{O}(\log n)$ of the decision tree versus $\mathcal{O}(n)$ of kNN leads to a distinct difference. Furthermore, the learning complexity of incremental decision trees depends on the classification performance, whereas those of NN-methods is constant. A decision tree is only growing in case of ambiguous labels within the leaves, therefore tasks with a high classification performance such as *PAMAP* or *KDD* cause a very slow growth and quick processing. However, large and noisy data sets lead to continuously growing trees, which eventually become slow and inefficient.
On average, SAM-E requires distinctly more processing time than the tree-based ensembles, which was expected considering the run time of the VFDT and the single SAM algorithm. In case of tasks with high-dimensional data and a high priority

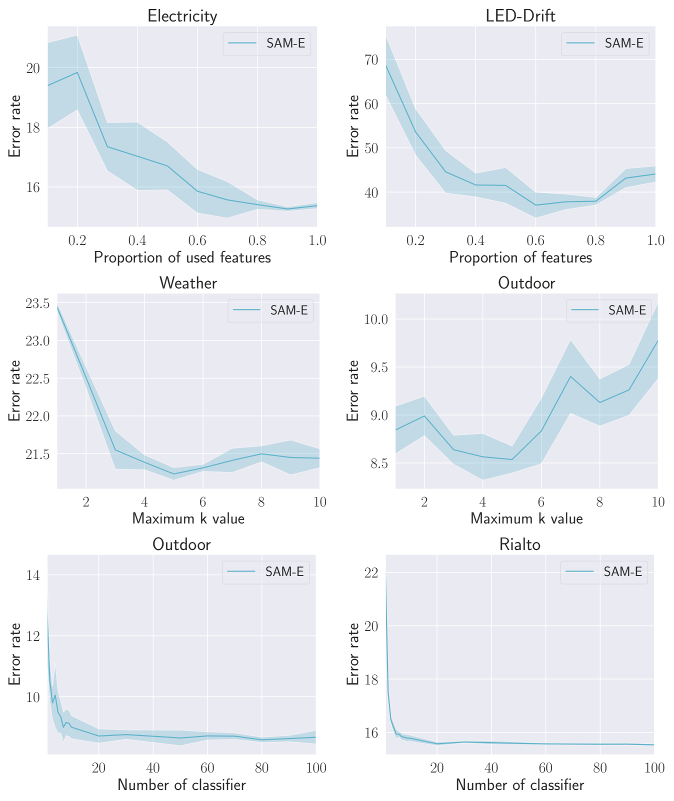

Fig. 4. The influence of the metaparameters on the results. Next to the number of classifiers, the varied parameters are the size of the random subspace (β) and the maximum k value ($k \sim \mathcal{U}(1, b)$).

TABLE VI
THE RUN TIMES (S) OF THE EXPERIMENTS. ENSEMBLES CONSISTED OF $N = 10$ MEMBERS. THE BEST RESULTS ARE MARKED IN BOLD.

Data set	VFDT	SAM	ARF	LVGB	SAM-E
SEA Concepts	**1.0**	4.7	8.0	3.9	20.6
Rot. Hyperplane	**3.8**	23.1	45.6	26.6	85.3
Moving RBF	**4.7**	35.0	37.6	41.6	135.8
Inter. RBF	**13.7**	42.2	58.8	171.8	149.1
Moving Squares	**2.0**	11.7	44.1	20.2	59.2
Transient Chessb.	**2.4**	16.5	31.8	15.7	75.3
Random Tree	**4.8**	17.5	25.2	30.5	77.8
LED-Drift	**3.9**	59.9	26.3	32.3	190.8
Mixed Drift	**6.5**	69.4	102.5	62.4	463.3
Poker	**10.6**	91.1	204.8	86.7	502.0
Outdoor	0.9	**0.7**	3.4	8.7	1.8
Weather	**0.5**	2.5	4.1	2.9	9.1
Electricity	**1.1**	4.4	11.2	5.7	19.2
Rialto	**4.4**	17.4	27.8	42.7	68.3
Airline	**8.6**	37.0	322.3	598.3	235.8
Cover Type	**18.8**	201.9	148.2	191.6	1195.7
PAMAP	**144.8**	1223.3	288.0	390.9	8506.9
SPAM	**266.8**	2142.9	2737.0	1510.3	15651.0
KDD99	**119.0**	1520.6	456.5	511.8	8908.3
Overall $\sum$	**618.1**	5521.8	4583.1	3754.5	36355.0

TABLE VII
INTERLEAVED TEST-TRAIN ERROR RATES OF THE ENSEMBLES USING $N = 100$ LEARNERS. ONLY THE AVERAGED PERFORMANCE ARE REPORTED DUE TO THE LACK OF SPACE.

	ARF	LVGB	SAM-E
Overall ∅	16.83	19.56	**12.92**
Overall ∅ rank	2.11	2.42	**1.47**

on a low processing time, kNN-based ensembles such as SAM-E may not be a viable option.

D. Scalability

The results achieved with 100 learners are given in Table VII. Increasing the number of members improves the performance of SAM-E only slightly. Noteworthy, LVGB performs worse with the larger ensemble because it only replaces one learner at a time in case of detected drift, which has a decreasing effect when the number of learners is increased. ARF profits the most from the increased number of classifiers. In contrast to LVGB, it replaces each learner for which a drift has been detected. Furthermore, it has a hgigher diversity due to the random subspace. However, increasing the size even further is not beneficial[2].

VI. CONCLUSION

In this paper, we presented our approach SAM-E which combines the robust drift-handling algorithm SAM with the advantages of a highly diversified ensemble. The diversity is ensured by randomizing the underlying kNN-search as well as the feature space of each learner. Additionally, a drift detection, monitoring the performance of the ensemble, enables a faster adaptation in case of drift and explicitly filters learners with improper parameterization for the current situation. As SAM itself, our method is easy to use in practice, since the few hyperparameters can be robustly set in general without the necessity of data set specific tuning. We provide a parallel implementation which is open-source, facilitating the comparison for other researchers. We compared SAM-E with state-of-the-art methods where it most of the times delivered the lowest error rates and overall a high robustness.

REFERENCES

[1] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Netw. and Appl.*, vol. 19, no. 2, 2014.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] H. Yu, C. Miao, C. Leung, and T. J. White, "Towards ai-powered personalization in mooc learning," *Science of Learning*, vol. 2, no. 1, p. 15, 2017.

[4] V. Losing, B. Hammer, and H. Wersing, "Knn classifier with self adjusting memory for heterogeneous concept drift," in *International Conference on Data Mining (ICDM)*, Dec 2016, pp. 291–300.

[5] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule," *Transactions on Systems, Man and Cybernetics*, no. 4, pp. 325–327, 1976.

[6] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001.

[2]We repeated the experiments for ARF with $n = 250$, resulting in an average error rate of 16.81.

[7] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *Journal of Machine Learning Research*, vol. 15, pp. 3133–3181, 2014.

[8] V. Losing, B. Hammer, and H. Wersing, "Incremental on-line learning: A review and comparison of state of the art algorithms," *Neurocomputing*, vol. 275, 2018.

[9] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Computing Surveys*, vol. 50, no. 2, pp. 23:1–23:36, Mar. 2017.

[10] P. Almeida, L. Soares de Oliveira, A. de Souza Britto Jr, and R. Sabourin, "Adapting the dynamic classifier selection for concept drift scenarios," *Expert Systems with Applications*, vol. 104, pp. 67–85, 08 2018.

[11] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug 1996.

[12] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[13] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Machine learning and knowledge discovery in databases*, 2010, pp. 135–150.

[14] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9, pp. 1469–1495, Oct 2017.

[15] S.-T. Chen, H.-T. Lin, and C.-J. Lu, "An online boosting algorithm with theoretical justifications," in *International Conference on Machine Learning (ICML)*, USA, 2012, pp. 1873–1880.

[16] R. Klinkenberg and T. Joachims, "Detecting concept drift with support vector machines." in *International Conference on Machine Learning (ICML)*, 2000, pp. 487–494.

[17] P. Domingos and G. Hulten, "Mining high-speed data streams," in *KDD*, 2000, pp. 71–80.

[18] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing." in *International conference on data mining (SDM)*, vol. 7, 2007, p. 2007.

[19] A. Bifet, B. Pfahringer, J. Read, and G. Holmes, "Efficient data stream classification via probabilistic adaptive windows," in *Symposium on Applied Computing*, 2013, pp. 801–806.

[20] N. C. Oza, "Online bagging and boosting," in *International conference on Systems, man and cybernetics*, vol. 3, 2005, pp. 2340–2345.

[21] S. H. Bach and M. A. Maloof, "Paired learners for concept drift," in *ICDM*. IEEE, 2008, pp. 23–32.

[22] C. Alippi, G. Boracchi, and M. Roveri, "Just-in-time classifiers for recurrent concepts," *TNNLS*, vol. 24, no. 4, pp. 620–634, 2013.

[23] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 10, pp. 2283–2301, 2012.

[24] B. Mirza, Z. Lin, and N. Liu, "Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift," *Neurocomputing*, vol. 149, pp. 316–329, 2015.

[25] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4802–4821, 2018.

[26] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Transactions on knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742, 2009.

[27] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, New York, NY, USA, 2001.

[28] A. Bellet, A. Habrard, and M. Sebban, "A survey on metric learning for feature vectors and structured data," *Computing Research Repository*, vol. abs/1306.6709, 2013.

[29] T. K. Ho, "The random subspace method for constructing decision forests," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, Aug. 1998.

[30] E. S. Page, "Continuous inspection schemes," *Biometrika*, pp. 100–115, 1954.

[31] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, Dec. 2006.

[32] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *ICML*, vol. 97, 1997, pp. 211–218.

[33] V. M. A. Souza, D. F. Silva, J. Gama, and G. E. A. P. A. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," in *Proceedings of SIAM International Conference on Data Mining (SDM)*, 2015, pp. 873–881.