SMOClust: Synthetic Minority Oversampling based on Stream Clustering for Evolving Data Streams

Chun Wai Chiu[1*] and Leandro L. Minku[2*]

[1*]School of Computer Science and Mathematics, Keele University, Keele, Staffordshire, ST5 5BG, United Kingdom.
[2*]School of Computer Science, University of Birmingham, Edgbaston, Birmingham, B15 2TT, United Kingdom.

*Corresponding author(s). E-mail(s): c.chiu@keele.ac.uk; l.l.minku@bham.ac.uk;

Abstract

Many real-world data stream applications not only suffer from concept drift but also class imbalance. Yet, very few existing studies investigated this joint challenge. Data difficulty factors, which have been shown to be key challenges in class imbalanced data streams, are not taken into account by existing approaches when learning class imbalanced data streams. In this work, we propose a drift adaptable oversampling strategy to synthesise minority class examples based on stream clustering. The motivation is that stream clustering methods continuously update themselves to reflect the characteristics of the current underlying concept, including data difficulty factors. This nature can potentially be used to compress past information without caching data in the memory explicitly. Based on the compressed information, synthetic examples can be created within the region that recently generated new minority class examples. Experiments with artificial and real-world data streams show that the proposed approach can handle concept drift involving different minority class decomposition better than existing approaches, especially when the data stream is severely class imbalanced and presenting high proportions of safe and borderline minority class examples.

Keywords: Data Streams, Class Imbalance, Concept Drift, Stream Clustering, Synthetic Data, Data Difficulty Factors

1 Introduction

In the past years, the volume and incoming speed of data have increased tremendously. Data frequently arrive continuously in the form of data stream rather than forming a single static data set. Therefore, *data stream learning*, which is able to learn incoming data upon arrival, becomes an increasingly important approach to extract knowledge from data. It has been widely used in real-world applications, such as credit card fraud detection [1], software defect prediction [2] and spam filtering [3]. There are many types of problems / tasks in data stream learning, for examples, classification, regression, clustering, anomaly detection etc. This work focuses on binary classification.

Concept drift is a common challenge in data streams. It is a change in the underlying distribution of the problem. Such a change can deteriorate the predictive performance of the data stream learning algorithm because the predictive model built previously may not be valid anymore. To deal with concept drift, data stream learning algorithms can be categorised to as explicit and implicit approaches [4, 5]. Explicit approaches employ a concept drift detection method to detect if there is a concept drift, and then adopt strategies to update predictive model to cope with such drift [4, 5]. Implicit approaches do not employ any concept drift detection method but continuously evolve themselves to reflect the current underlying concept, thus adapting to concept drifts [4, 5].

Data stream learning algorithms can also be categorised by their mode of operation: batch-based (chunk-based) learning and online learning [4, 6]. Batch-based learning refers to as learning the data stream by batches of new training data. It has the advantage of having more data to learn at a given time step, thus the learning approach can better capture the current underlying concept [4, 6]. In contrast, online learning has a stricter requirement which only allows the data stream learning approach to process each training example separately and then discard it [4, 6],rendering it applicable to problems with stricter time and memory requirements. To deal with concept drifts in a timely fashion, online learning usually is more preferable than batch-based learning which needs to wait for whole batches of training examples to arrive. Moreover, batch-based learning assumes that all training data within the same batch are drawn from the same underlying concept, which may not always be the case in most real-world applications. Thus, this work focuses on *online learning*.

Another challenge frequently present in real world data stream applications is that their class distribution is often skewed, an issue that is commonly referred to as *class imbalance* [7]. For example, in credit card fraud detection, there are always more genuine transactions than fraudulent transactions. In software defect prediction, there are typically more clean than defective components. When class imbalance exists, the data stream learning algorithms are likely to get biased towards the majority class, being likely to misclassify minority class examples. Yet, the minority class is usually the class of interest in the classification task, meaning that misclassifying minority class examples could lead to a high cost. This work focuses on binary classification, thus,

there is a majority class and a minority class when the data stream is class imbalanced.

To deal with class imbalance, a category of oversampling strategies has shown to be successful in data set learning (offline learning). They create synthetic examples to enrich the minority class, which causes less overfitting than reusing existing minority class examples [8–10]. Some recent work attempted to bring such a successful idea into the field of data stream learning [11, 12]. However, they usually cache all the minority class examples seen so far into the memory which is impractical for data stream learning. Moreover, reusing all past minority class examples may prevent these approaches from dealing with *concept drifts* (changes in the underlying probability distribution, a.k.a., *concept* [13]) affecting the minority class.

Dealing with the joint issue of concept drift and class imbalance is a challenging task. In particular, the relatively small number of minority class examples mean that it may be more difficult to detect or adapt to concept drifts affecting the minority class. Many studies have been proposed to deal with either class imbalance or concept drift. However, existing work to deal with their joint challenge remains little. Although a recent survey work [7] showed that class imbalance is a more dominant factor than concept drift in affecting the predictive performance, the effectiveness of the existing class imbalance techniques for data stream learning could potentially be compromised by concept drift as most of them are not prepared to deal with drifts. Recent work addresses this challenge by finding relevant past minority class examples for oversampling [14] or performing synthetic minority class oversampling based on the statistics of the minority class after drift detection [15]. These methods are not always ideal because relevant past minority class examples might not exist while relying on drift detection to reset minority class statistics could be detrimental, especially when the drift is gradual.

In addition, the method of storing past examples as proposed in [14] may be impractical for data stream learning when there are strict space constraints. Similarly, synthesising new examples based on simple statistics of past examples as proposed in [15] overlooks important data difficulty factors within the class. Specifically, this method does not consider the location of past examples in the feature space. These data difficulty factors include concept drifts involving different movements of the minority class sub-clusters, changing class imbalance ratio, and changing the ratios of different types of minority class examples. Existing work has shown that these factors are critical in learning from drifting class imbalanced data streams [16].

Therefore, new approaches are needed to better address concept drifts with multiple data difficulty factors in class imbalanced data streams. To fill this research gap, this paper aims to answer the following research questions:

- **RQ1)** How to produce minority class synthetic examples for oversampling so that we could explore the decision areas of the minority class to better consider data difficulty factors while adapting to concept drift?

- **RQ2)** How does the proposed approach perform compared to existing approaches in different types of concept drift affecting the minority class? For which types does it perform the best and worst? Why?
- **RQ3)** How does the proposed approach perform compared to existing approaches when applied to real-world data streams?

To answer RQ1, we propose a novel method to create synthetic minority class examples for oversampling based on stream clustering. The motivation is that stream clustering methods use a set of micro-clusters as the abstraction/compression of the examples they have seen so far. They usually retain micro-clusters by temporal order, which means old micro-clusters are forgotten. Therefore, the information they hold reflects the characteristics of the current underlying concept. Our novel method exploits this nature of stream clustering methods to track the current decision areas of the minority class. It then generates synthetic minority class samples for oversampling within the region where real minority class examples have been recently observed. With this strategy, the proposed method is less likely to produce noisy synthetic examples while being able to explore the decision areas of the minority class, better considering data difficulty factors when adapting to concept drift (RQ1).

The proposed approach is compared against five existing approaches through experiments on artificial data streams considering different data difficulty factors and class imbalance ratios, and real-world data streams (RQ2, RQ3). The results show that SMOClust handled concept drifts of different minority class sub-clusters movements better than existing approaches (RQ2, RQ3). It also performed better than existing approaches when the data stream is severely class imbalanced and presents high proportions of safe and borderline [17] minority class examples (RQ2, RQ3). Its major weakness is to handle data streams presenting large proportions of rare and outlier [17] minority class examples (RQ2, RQ3).

The rest of this paper is organised as follows. Section 2 discusses related work on synthetic minority class oversampling techniques and state-of-the-art approaches in dealing with class imbalance and concept drift in data stream learning. Section 3 presents the proposed approach. Section 4 presents the experimental study and discusses the results. Section 5 concludes this study and discusses the future work.

2 Related Work

This section first introduces class imbalance and existing resampling methods for class imbalanced learning in Section 2.1. Section 2.2 then discusses the state-of-the-art approaches to deal with class imbalance and concept drift in data stream learning. Table 1 summarises the main characteristics of the approaches discussed in this section. At the end of this table, we contrast these with SMOClust, the approach that we propose in Section 3 of this paper.

Table 1: Comparison of Principal Characteristics Between Related Works and SMOClust

Approach	Classifier Type	Approach Type	Concept Drift Adaptation	Class Imbalance Strategy
OnlineUnderOverBagging [11]	Online Ensemble	N/A	N/A	Undersampling + Oversampling
OnlineSMOTEBagging [11]	Not Strictly Online[a] Ensemble	N/A	N/A	SMOTE
Learn++.CDS [18]	Batch-based Ensemble	Implicit	Weighted Majority Ensemble	SMOTE
Learn++.NIE [18]	Batch-based Ensemble	Implicit	Weighted Majority Ensemble	Sub-ensemble Method
DWMIL [19]	Batch-based Ensemble	Implicit	Weighted Majority Ensemble	Undersampling
HUWRS.IP [14]	Batch-based Ensemble	Implicit	Hellinger Distance + Random Subspace Method	Instance Propagation
OOB [20]	Online Ensemble	Implicit[b]	Fading Factor	Oversampling
UOB [20]	Online Ensemble	Implicit[b]	Fading Factor	Undersampling
ESOS-ELM [21]	Online Ensemble	Explicit	Hypothesis testing + Weighted Majority Ensemble	Sub-ensemble Method
C-SMOTE [22]	Not Strictly Online[a] Meta-strategy	Explicit	ADWIN	SMOTE + ADWIN
VFC-SMOTE [15]	Online Meta-strategy	Explicit	Employ Drift Detector	Synthetic Minority Oversampling by Beta Distribution Sampling + Histogram-based Sketch
SMOTE-OB [23]	Online Ensemble	Explicit	Employ Drift Detector	Synthetic Minority Oversampling by Beta Distribution Sampling + Histogram-based Sketch
CSARF [24]	Online Ensemble	Explicit	Employ Drift Detector + Weighted Majority Ensemble	Cost-sensitive Weighting + Sub-ensemble Method
ROSE [25]	Not Strictly Online[a] Ensemble	Explicit	ADWIN + Weighted Majority Ensemble	Cost-sensitive Weighting + Undersampling
SMOClust (Proposed approach)	Online Meta-strategy	Explicit	Employ Drift Detector	Synthetic Minority Oversampling by Multivariate Skewed Gaussian Sampling + Stream Clustering

[a] Not Strictly Online: Whilst these approaches process training examples upon arrival, they store training examples for later use.
[b] Can only deal with P(Y) drift.

2.1 Resampling Methods for Class Imbalance

Class imbalance refers to the data set or data stream having at least one under-represented class (minority class). In this situation, the machine learning model tends to misclassify minority class examples more frequently than the majority class because there exists very little information about the minority class.

Approaches to address class imbalance are mainly divided into three categories: algorithm-level approach, ensemble approach, and data-level approach [7]. Algorithm-level approaches are often called cost-sensitive approaches, as they place a higher cost when misclassifying minority class examples than majority class examples. Ensemble approaches create different class balanced training subsets to train each ensemble member. Data-level approaches modify the class distribution using a resampling method, such that standard machine learning models can learn from both classes with the same amount of information. They can be applied during the data pre-processing phase. Due to this generic nature, this work focuses on data-level approaches.

Undersampling and *Oversampling* are two main types of data-level approaches [7]. Undersampling methods reduce the number of majority class examples for training, usually removing noisy examples or examples that are deemed to have a low impact on the decision boundary. Yet, it has the chance to cause important information loss. On the other hand, oversampling methods increase the number of minority class examples, by replication or synthesis. They will not cause any information loss but they cause longer training time and are likely to cause overfitting when training on the same examples multiple times.

Synthetic Minority Oversampling Technique (SMOTE) [8] is a very renowned oversampling technique in offline learning, which synthesises minority class examples for oversampling, thus balancing the data set. In practice, SMOTE first randomly chooses an existing minority class example from the data set, denoted as x_i. It then randomly chooses one of the k-nearest neighbours of x_i in the minority class, denoted as x_i'. After that, a difference vector between x_i and x_i' is calculated. To create a point along the line between x_i and x_i', each dimension of the difference vector is multiplied by a random number θ $(0 < \theta < 1)$, then the resulting difference vector is added to x_i dimension-wisely. SMOTE performs this procedure until the target oversampling rate M is reached. This oversampling rate M and the k for k-nearest neighbours are the hyper-parameters of the algorithm.

Many variants of SMOTE have been proposed in the last two decades. For example, Borderline-SMOTE [9] considers that examples close to the decision boundary are more difficult to learn, thus it synthesises minority class examples around this area. Gaussian-based SMOTE (G-SMOTE) [10] is a more recent approach which tend to synthesise examples very close to the existing minority class examples. Other well-known methods of synthetic minority oversampling include ADASYN [26], DBSMOTE [27], SWIM [28] etc.

On top of the class imbalance ratio, it has been pointed out that data difficultly factors also greatly impact the classification performance [17] . These

factors describe the characteristic of a given example (usually the minority class example) in the feature space:

- Safe: Surrounded by examples from the same class.
- Borderline: Located near the decision boundary.
- Rare: Located deep inside the decision region of the opposite class, together with handful examples from the same class.
- Outlier: Isolated and located deep inside the decision region of the opposite class.

The aforementioned methods can also be considered as taking the data difficulty factors into the account. For example, Borderline-SMOTE synthesises minority class examples around the borderline region, while G-SMOTE can be considered as synthesising minority class examples in the safe region.

However, these synthetic minority oversampling methods could not be applied to class imbalanced data stream learning directly as they cache the entire data set into memory, which is impractical in data stream learning. For example, OnlineSMOTEBagging [11] is one of this kind. It replaces simple oversampling with SMOTE in OnlineUnderOverbagging. In our preliminary experiments with the data streams used in this work, we attempted to run OnlineSMOTEBagging. However, OnlineSMOTEBagging consumed all the memory we had access to (8GB), resulting in failure to complete the run. Furthermore, the underlying concept of the data stream may change over time (concept drift). The cached examples may be from different concepts. Thus, synthesising minority class examples based on them may not always follow the current underlying concept.

Additionally, one recent work in the field of software effort estimation is also quite inspiring [29]. They enlarge the software project data set by adding Gaussian noise to the existing examples. This method could be particularly related to synthetic minority oversampling for class imbalanced data stream learning as it is memory efficient and fast to perform. The potential risk is that, if we apply it to the most recent minority class examples, it might cause overfitting to such a recent area.

2.2 Approaches for Class Imbalanced Data Stream Learning in the Presence of Concept Drift

This section discusses approaches that are closely related to the proposed approach. For a comprehensive survey on class imbalanced data stream learning, please refer to [30].

Broadly speaking, existing approaches to deal with class imbalance and concept drift have two main categories: explicit approach and implicit approach.

Explicit Approaches

Explicit approaches estimate whether a concept drift has happened, usually by employing a drift detector to monitor the predictive performance of the base learner / main ensemble. This drift detector can be any from the literature,

ideally using a class imbalance insensitive metric, such as DDM-OCI [31], LFR [32], PAUC-PH [33] etc.

Continuous-SMOTE (C-SMOTE) [12] is one of the pioneers who bring SMOTE to drifting class imbalanced data stream learning. It uses an Adaptive Window (ADWIN) [34] to store the most recent examples and applies SMOTE to the minority class examples inside the ADWIN for oversampling. Upon drift detection, the old window of ADWIN is dropped as it is deemed to belong to the old concept. However, when there is no concept drift detection, C-SMOTE keeps storing minority class examples which can cause memory issues. Besides, SMOTE does not take decision boundaries and data difficulty factors into consideration, thus noisy examples may be generated.

Very Fast Continuous-SMOTE (VFC-SMOTE) [15] was proposed to solve the issues faced by C-SMOTE. It uses a dynamic summary data structure, called "sketch", to summarise the statistics of past examples. It generates synthetic examples by Beta distribution sampling from a set of summaries in the sketch, where each summary has the information of one input feature of past examples. When generating synthetic minority class examples, VFC-SMOTE tends to choose summaries that represent more past examples, which means it tends to generate synthetic minority class examples in the dense area of minority class. Nevertheless, this method may generate considerably noisy synthetic examples because it samples each input feature individually and does not adopt mechanisms to try to respect decision boundaries.

SMOTE with Online Bagging (SMOTE-OB) [23] is another approach that is similar to VFC-SMOTE. It incorporates the strategy of generating synthetic minority class examples from VFC-SMOTE into OnlineUnderOverBagging [11]. With this design, SMOTE-OB combines three data-level re-balancing methods to combat class imbalance while training the base learners diversely [23]. However, as SMOTE-OB uses the same synthetic minority class examples generating strategy as VFC-SMOTE, it faces the same disadvantages in terms of potentially generating considerably noisy synthetic examples.

Ensemble of Subset Online Sequential Extreme Learning Machine (ESOS-ELM) [21] is another notable explicit approach for drifting class imbalanced data stream learning. It uses a sub-ensemble method to train each base learner with an approximately equal number of majority and minority class examples, thus dealing with class imbalance. To deal with concept drift, it uses a threshold-based strategy with hypothesis testing to detect any significant change in the predictive performance of the main ensemble, thus reporting concept drift. Meanwhile, it also uses a weighted majority vote system, based on G-Mean, to adapt to any potential concept drift that could not be detected by the aforementioned method. ESOS-ELM's sub-ensemble method is time efficient in dealing with class imbalance as it does not replicate or synthesise any examples. However, it does not provide additional information to explore the decision areas of minority class. Besides, ESOS-ELM is restrictive in terms of base learner type. It only allows to use ELMs.

Cost-sensitive Adaptive Random Forest (CSARF) [24] is an online, cost-sensitive sub-ensemble method designed to address the challenges of drifting class imbalanced data streams. It is a variant of the Adaptive Random Forest (ARF) [35] algorithm. It incorporates a drift detector and a weighted majority ensemble to handle concept drift. To deal with class imbalance, CSARF utilises the Matthews Correlation Coefficient (MCC), a class imbalance insensitive metric, to assign weights to internal decision trees and ensure that all trees are trained with examples from the minority class [24]. While CSARF offers speed and memory efficiency due to its cost-sensitive approach, it fails to consider factors related to data difficulty. Additionally, CSARF is limited to using only the Hoeffding Tree [36] as base learners.

Robust Online Self-Adjusting Ensemble (ROSE) [25] is a cost-sensitive ensemble method designed specifically for learning from drifting class imbalanced data streams. It employs ADWIN as a drift detector and uses a weighted majority ensemble to handle concept drift. To address class imbalance, ROSE employs self-adjusting λ bagging (where λ is determined based on estimated class sizes), and enforces the Hoeffding bound to improve predictive performance in the minority class. Furthermore, ROSE maintains sliding windows per class to store the most recent examples and to create a class balanced data set through undersampling. This class balanced data set is used to build new background base learners. However, similar to CSARF, ROSE does not consider data difficulty factors in its class imbalance adaptation strategy. Additionally, ROSE's strategy for building new background base learners may be prone to more extreme levels of class imbalance in non-stationary data streams because such a scenario would require using very old minority class examples to build new base learners, besides the sliding window initially taking time to get filled with minority class examples.

In short summary, most existing explicit approaches to deal with class imbalance and concept drift do not explore the decision areas of the minority class. Whilst a few recent work [12, 15, 23] attempted to fill this research gap, they did not strictly take decision boundaries and data difficulty factors, which are crucial in data stream learning, into consideration.

Implicit Approaches

Implicit approaches are usually ensemble learners. They do not actively detect concept drift but continuously update the voting weights of the base learners, thus adapting to any potential changes in the underlying concept. However, in class imbalanced data stream learning, the weighting strategy also needs to consider that the base learner may bias toward the majority class. To address this issue, one can place a higher penalty on the weight of the base learners performing poorly in the minority class (cost-sensitive approach). Another method is to employ a resampling method to reduce the learning bias (data-level approach).

Oversampling-based and Undersampling-based Online Bagging (OOB and UOB) [20] are two pioneers of data-level approach for class imbalanced data

streams. Their idea is to incorporate random oversampling or random undersampling with Online Bagging (OB) [37]. They estimate the current class size based on an exponential smoothing function with a fading factor θ. Whenever a new example s_t with a class label y_t arrives, it is first used to calculate the class imbalance ratio of class y_t to the majority class (OOB) or the minority class (UOB). This ratio is used as the parameter λ of Poisson distribution in OB, thus deciding the number of times to train each ensemble member on s_t. While OOB and UOB are effective in addressing class imbalance with simple resampling methods, they can only deal with concept drifts that affect the posterior probability of the classes ($P(Y)$).

Learn++ for Concept Drift with SMOTE (Learn++.CDS) and Learn++ for Non-stationary and Imbalanced Environments (Learn++.NIE) [18] are two pioneer batch-based approaches in this category. They were both based on the well-known approach, Learn++ for Non-Stationary Environment (Lean++.NSE) [38]. Learn++.CDS uses SMOTE to balance the most recent batch of training data, while Learn++.NIE is a sub-ensemble method which bootstraps the majority class in the most recent batch of training examples to create different class balanced training sets. They both use weighted majority vote as a strategy to deal with concept drift where ensemble members performing well in the minority class have a higher weight. While they are both great methods to deal with class imbalance, they could struggle when the data stream is severely class imbalanced because there could exist some training batches which has no minority class examples.

Dynamic Weighted Majority for Imbalance Learning (DWMIL) [19] brought the renowned Dynamic Weighted Majority (DWM) into class imbalanced data stream learning. In general, it changes the weighting metric from accuracy to a class imbalance insensitive metric, such as G-Mean, while adopting UnderBagging [39], which is an offline learning approach, as the base learner to deal with class imbalance.

Heuristic Updatable Weighted Random Subspaces with Instance Propagation (HUWRS.IP) [14] is a batch-based learning approach to deal with drifting class imbalanced data streams. It is based on the approach of HUWRS [40] which was proposed to learn class balanced data streams. The main novelty of HUWRS.IP is the example selection mechanism, called Instance Propagation (IP), which selects relevant past minority class examples for oversampling the most recent train batch. However, these examples may not exist in the memory.

Shortly summarising, existing implicit approaches to deal with class imbalance and data stream learning either rely on sub-ensemble methods or reusing relevant past examples. These methods do not explore the decision areas of the minority class. They do not take data difficulty factors into account either. Besides, these approaches are batch-based approaches, thus they are unlikely to react to concept drift swiftly due to the need to wait for whole batches to arrive.

3 Proposed Approach

To answer the RQ1 posed in Section 1, we proposed a novel approach called Synthetic Minority Oversampling based on stream Clustering (SMOClust). The main novelty of this approach is to produce synthetic minority class examples for oversampling based on the information compressed by the stream clustering method. Most stream clustering methods represent this information in the form of micro-clusters, which summarise the statistics of past examples that are close together in the feature space. These statistics usually include the vectors of the dimensional-wise cumulative sum and squared sum. Thus, they do not need to cache all the past examples in the memory. Most importantly, this strategy could potentially deal with gradual drift involving different data difficulty factors because stream clustering methods continuously update themselves to reflect the characteristics of the current underlying concept.

SMOClust also employs a concept drift detector to monitor the predictive performance of the base learner, as a strategy to deal with abrupt drift. Thus, it is an explicit concept drift adaptation approach. Upon drift detection, the base learner will be reset. Although this strategy may not always be ideal [41, 42], this work focuses on investigating the effectiveness of the novel stream clustering based synthetic minority oversampling strategy in learning class imbalanced data streams with concept drift. So, it is intended to keep other components of SMOClust simple to analyse the characteristics of the proposed strategy.

Algorithm 1 presents the pseudo-code over-viewing SMOClust. The details of its working mechanism are described and explained as follows.

SMOClust is a data stream learning algorithm that uses a base learner B to learn from and make predictions to new examples. This base learner B could be any single learner, such as Hoeffding Tree [36], or an ensemble learner, such as Online Bagging [37]. SMOClust does not store past models. It uses stream clustering methods $SC[]$ to manage sets of micro-clusters that compress the information of past examples. There is one stream clustering method for each class of the problem (line 1, Algorithm 1). The stream clustering method can be arbitrary from the literature, such as Clustream [43], StreamKM++ [44], DenStream [45], Clustree [46] etc. In this work, Clustream was chosen because it is largely invariant for different types of concept drifts, meaning that it can effectively adapt to concept drift without compromising the quality of its clustering results [47]. The strategy of synthesising minority class examples for oversampling based on micro-clusters is explained in Section 3.1.

The most recent example s_t will be first used for concept drift detection (line 4, Algorithm 1). This concept drift detection method can be arbitrary from the literature, such as DDM [48], DDM-OCI [31], PAUC-PH [33], ADWIN [34] etc. Upon drift detection, the base learner B and time decay class sizes are reset but not the stream clustering methods $SC[]$ because they are prepared to adapt to concept drifts (line 5, Algorithm 1). That said, after concept drift detection, the stream clustering methods will still retain some knowledge belonging to the previous concept. This has two advantages: 1) In the case of

Algorithm 1 Synthetic Minority Oversampling based on Stream Clustering - SMOClust

Hyper-parameters: Base Learner(b), Stream Clustering Method(sc), Class Size Fading Factor(θ), Gaussian Noise Variance(v), Categorical Change Probability(P_c), k-Nearest neighbour(k), Drift Detector(d), Data Stream(S)
Variables: Base Learner(B), Stream Clustering Methods array($SC[]$)

1: Create a array of stream clustering methods ($SC[]$). Each of them corresponds to a class of the classification task.
2: **for** each new example s_t from the data stream S **do**
3: Update the drift detector by the prediction made by base learner B to s_t
4: **if** drift detection alarm is issued **then**
5: Reinitialise the base learner B
6: **end if**
7: Train the base learner B and update its estimated class size using s_t
8: Store the latest example of each class
9: $class_{maj}, class_{min} \leftarrow$ Determine the current majority and minority class based on the estimated class size by B
10: **while (**the minority class estimated by B is smaller than that of majority class **AND** all stream clustering methods can provide micro-clustering results**) OR** SMOClust has observed any minority class example **do**
11: **if** all stream clustering methods are ready to provide micro-clustering results **then**
12: $mCluster_{anchor} \leftarrow$ Randomly pick a frequently updated micro-cluster of $class_{min}$
13: **if** $mCluster_{anchor}$ is surrounded by micro-clusters of $class_{min}$ **then**
14: $synthInst^{Bin} \leftarrow$ create a synthetic example using Alg. 2
15: **else**
16: $synthInst^{Bin} \leftarrow$ create a synthetic example by Gaussian sampling $mCluster_{anchor}$
17: **end if**
18: Train $SC[class_{min}]$ using $synthInst^{Bin}$ (without class attribute)
19: Train the base learner B and update its estimated class size using $synthInst^{Bin}$
20: **else**
21: $synthInst \leftarrow$ create a synthetic example by adding Gaussian noise to latest minority class example [29]
22: Train $SC[class_{min}]$ using $synthInst$ (without class attribute)
23: Train the base learner B and update its estimated class size using $synthInst$
24: **end if**
25: **end while**
26: Use s_t (without class attribute) to train the stream clustering method that corresponds to the class value of s_t
27: **end for**

false positive drift detection, SMOClust can exploit the knowledge stored in the micro-clusters to train the base learner. 2) Knowledge of the pre-drift concept could help to learn the post-drift concept, especially when the drift has low severity [49].

After that, SMOClust uses s_t to train B and to update the time decay class sizes (line 7, Algorithm 1). The time decay class sizes estimate the current minority class and thus determine the oversampling rate. Equation 1 presents the calculation of the normalised class size of class c_m at time step t [20]:

$$classSize(c_m)^{(t)} = \begin{cases} \frac{1}{M}, & \text{if } t = f \\ \frac{[c_{s_t}=c_m]+\theta \times classSize(c_m)^{(t-1)} \times (t-f)}{t-f+1}, & \text{otherwise} \end{cases} \quad (1)$$

where $m \in M$ and $M = \{0, 1\}$, considering binary classification tasks and θ ($0 < \theta < 1$) is a predefined time decay factor. c_{s_t} is the true class of s_t. Thus, $[c_{s_t} = c_m] = 1$ if the true class of s_t is c_m, otherwise 0. f is the first

time step used in the calculation. Note that, unlike OOB and UOB [20] which estimate the current class sizes of the data stream, SMOClust estimates the class imbalance degree of the information seen by the base learner rather than the class imbalance degree of the data stream. Thus, synthetic examples are also used to update the class sizes. The reason behind this design is discussed together with the strategy of training the base learner with synthetic examples.

SMOClust first records the most recent examples from each class (line 8, Algorithm 1), then checks if the base learner has learnt from both classes equally (line 10, Algorithm 1). If not, SMOClust will generate synthetic minority class examples for oversampling based on the micro-clusters of the minority class (line 13-17, Algorithm 1), which is detailed in Section 3.1.

In the case that not all stream clustering methods can provide micro-clustering results and SMOClust has observed and recorded the most recent "real" example of the minority class (denoted as $s_{last\_minority}$), SMOClust will generate a synthetic minority class by adding Gaussian noise to $s_{last\_minority}$ for oversampling (line 21, Algorithm 1). This strategy follows the strategy proposed by [29], except SMOClust treats ordinal attributes as categorical attributes due to the limitation in MOA [50].

No matter the synthetic minority class example is generated based on micro-clusters or Gaussian noise, SMOClust will use it to train the base learner and the corresponding stream clustering method, and to update the class size immediately (line 18-19, 22-23, Algorithm 1). This strategy can prevent the base learner from biasing towards the majority class when there are no "real" minority class examples arrive for a long period, which is likely to happen when the data stream is extremely class imbalanced. Also, updating the class sizes with both "real" and synthetic examples allows us to estimate if the base learner has learnt from both classes equally. If not, SMOClust will then create synthetic minority class examples to train the base learner immediately.

In the case of none of the above conditions being satisfied, i.e., none of the conditions of the while-loop are satisfied (line 10, Algorithm 1), SMOClust will not perform any oversampling because this means either oversampling is not needed or there is no information about the minority class for SMOClust to generate synthetic examples. Lastly, a copy of the most recent example s_t is converted to a suitable format to train the stream clustering method, corresponding to the class value of s_t (line 26, Algorithm 1).

3.1 Generating a Synthetic Minority Class Example for Oversampling using Micro-clusters

This section presents the overview of generating a synthetic minority class example for oversampling based on micro-clusters. The general idea is to create synthetic minority class examples in one of the dense areas of the minority class. In this way, we can consolidate the knowledge learnt in the existing minority class areas without being greatly affected by noise. In the case where a dense area does not exist, SMOClust will pick one of the past minority class areas to explore the decision boundary around it.

Algorithm 2 presents the pseudo-code of this method. The details of generating a synthetic minority class example using micro-clusters can be described as follows.

Algorithm 2 Generate Synthetic Instance with k-NN Micro-Clusters

1: **function** GENSYNTHINSTBYKNN($SC[class_{min}]$, $mCluster_{anchor}$, $class_{min}$, k)
2: $sphere\_cluster \leftarrow$ combine $mCluster_{anchor}$ with its k nearest micro-clusters, using Alg. 3
3: $synthInst \leftarrow$ create a synthetic example by sampling $sphere\_cluster$, using Alg. 4
4: **return** $synthInst$
5: **end function**

First of all, SMOClust randomly takes one of the micro-clusters from the minority class as an anchor (denoted as $mc_{anchor}^{minority}$) (line 12, Algorithm 1). Micro-clusters that are created recently or are updated frequently and recently have higher chance to be chosen as this anchor. After that, SMOClust checks if $mc_{anchor}^{minority}$ is surrounded by the micro-clusters from the same class (line 13, Algorithm 1). If this condition is satisfied, SMOClust can consider such area is dense enough to create synthetic minority examples for oversampling. It will then make a copy of $mc_{anchor}^{minority}$ and then combine it with its k-Nearest micro-clusters (based on hull distance) in class $class_{min}$ to form a temporary micro-cluster mc_{temp} (line 2, Algorithm 2). We denote such set of k-Nearest micro-clusters as $MC^{kNN,minority}$, thus, $|MC^{kNN,minority}| = k$ and each k-Nearest micro-clusters is denoted as $mc_i^{kNN,minority} \in MC^{kNN,minority}$. The details of how to combine a set of micro-clusters into one are presented in Algorithm 3.

Algorithm 3 Combining a set of micro-clusters into one

1: **function** COMBINE($mClusters[]$)
2: $c_{new} \leftarrow$ compute the weighted average of the centres of micro-clusters in $mClusters[]$
3: **for** each micro-cluster $mClust \in mClusters[]$ **do**
4: $d \leftarrow$ compute the distance between c_{new} and the centre of $mClust$
5: $r_n \leftarrow r_n \cup$ (the radius of $mClust + d$)
6: **end for**
7: $r_{new} \leftarrow$ find the largest value in r_n
8: **return** a new micro-cluster with centre c_{new} and radius r_{new}
9: **end function**

To combine a set of micro-clusters into one, we first need to calculate the new centre c_{new} of the resulting micro-cluster mc_{temp}. This can be achieved by getting the weighted average of the centres of the original set of micro-clusters, dimensionwisely (line 2, Algorithm 3). After that, we set the radius r_{new} of the resulting micro-cluster mc_{temp} to as the distance between the new centre to the farthest hull (boundary) among all the original micro-clusters (line 3-7, Algorithm 3). Figure 1 illustrates an example of combining $mc_{anchor}^{minority}$ with its 3-nearest neighbours into one micro-cluster.

A synthetic minority class example will then be generated by sampling from this resulting micro-cluster with the highest chance near the centre of

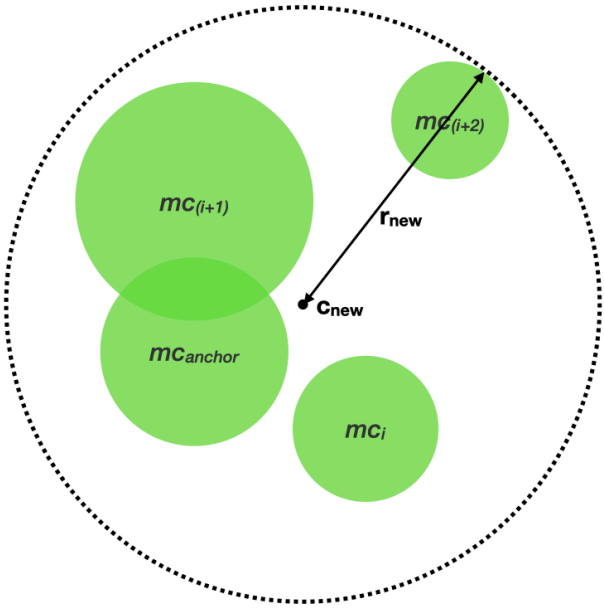


Fig. 1: Illustration of Combining $mc_{anchor}^{minority}$ with 3-nearest neighbours into one micro-cluster

$mc_{anchor}^{minority}$ (line 3, Algorithm 2). Figure 2 illustrates an example of sampling from a synthetic minority class example from mc_{temp}.

In Figure 2, the green circles are the micro-clusters belonging to the minority class while the blue circles are the micro-clusters belonging to the majority class.[1] The black circle line represents mc_{temp} and the red dashed lines are the contour of the probability density function to sampling a point. The closer to the centre of $mc_{anchor}^{minority}$, the higher the probability.

The reason for sampling a new synthetic minority class example close to $mc_{anchor}^{minority}$ is that this mc_{temp} could overlap with the micro-cluster from the other class. If we just sample from mc_{temp} randomly or by a multivariate Gaussian distribution with a mean at c_{new}, we will have a high chance to sample a point that is close to the region or the majority class. Therefore, sampling points as synthetic minority class examples from mc_{temp} but close to the centre of $mc_{anchor}^{minority}$ can reduce the risk of generating noisy examples while maintaining the ability to explore this dense region of the minority class.

Although Figure 2 only illustrates an example in two-dimensional feature space, this idea can be applied to any multi-dimensional space. This sampling strategy is further detailed in Section 3.2.

[1]Note that, the size and the number of micro-clusters in Figure 2 do not necessarily reflect the number of examples in each class. This figure just focuses on a particular region in the feature for explanation purposes.

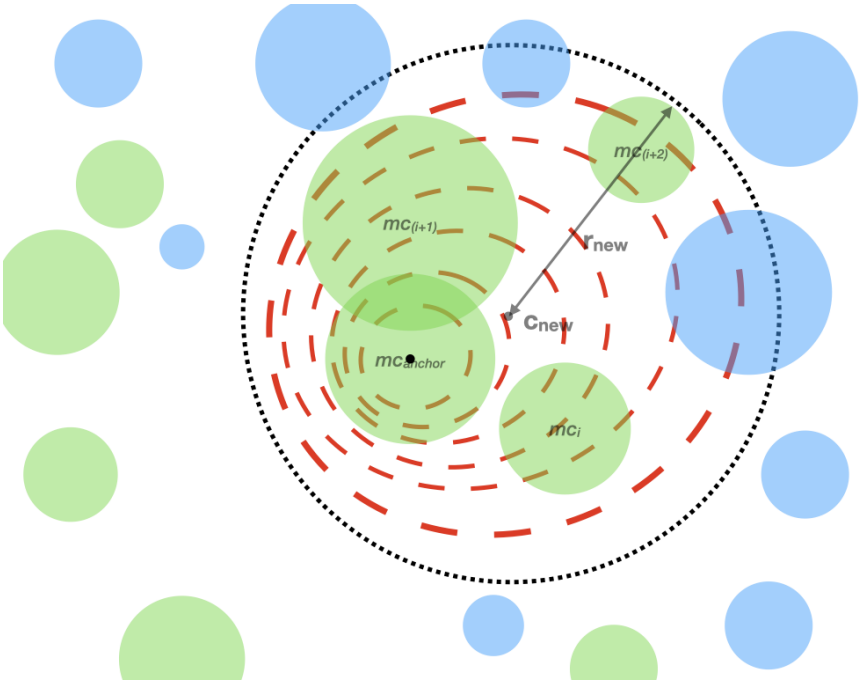


Fig. 2: Illustration of Sampling a Synthetic Minority Class Example from mc_{temp}

In the case that $mc_{anchor}^{minority}$ is not surrounded by the micro-clusters belonging to the same class, SMOClust will generate a synthetic minority class example by performing multivariate Gaussian sampling inside $mc_{anchor}^{minority}$ (line 16, Algorithm 1). For example, this will be the case when when $mc_{(i+2)}$ (top right green circle in Figure 2) is chosen to be the $mc_{anchor}^{minority}$. The the mean of the multivariate Gaussian distribution is the centre of $mc_{anchor}^{minority}$ and the standard deviation is set as a third of the radius of $mc_{anchor}^{minority}$ ($radius/3$). In other words, the boundary of $mc_{anchor}^{minority}$ is set at three units standard deviations (or standard score = 3) from the centre. Therefore, we have 99.9% of chance to sample a point within $mc_{anchor}^{minority}$. Gaussian distribution was chosen rather than uniform distribution in sampling $mc_{anchor}^{minority}$ because $mc_{anchor}^{minority}$ could partly overlap with the majority class region. Therefore, sampling a new point as synthetic minority class example close to the centre of $mc_{anchor}^{minority}$ is a safe strategy.

3.2 Sampling from a Micro-cluster with the Highest Probability at a Designated Location

This section present the strategy to sampling points from the temporary micro-cluster mc_{temp} which is formed by combining $mc_{anchor}^{minority}$ and $mc_{i}^{kNN,minority} \in$

$MC^{kNN,minority}$ with the highest probability at the centre of $mc_{anchor}^{minority}$. The general idea is to sample random points that are inside mc_{temp} and these points are likely to be close to the centre of $mc_{anchor}^{minority}$. The pseudocode of this sampling strategy is presented in Algorithm 4. Figure 3 illustrates the steps of this sampling strategy and it can be explained as follows.

Algorithm 4 Sampling from a Hyper-Sphere by Skewed Gaussian with the Maximum of the Probability Density Function at a Designated Location

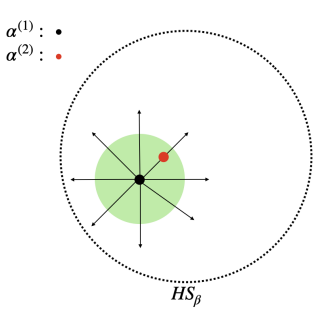
1: **function** SAMPLE_AROUND_TARGET($\alpha^{(1)}$, *sphere_cluster*)
2: $\beta \leftarrow sphere\_cluster.getCentre()$
3: $r \leftarrow sphere\_cluster.getRadius()$
4: $dimensions \leftarrow \beta.numOfDimensions()$
5: $\delta \leftarrow createArrayWithSize(dimensions)$
6: $\gamma \leftarrow createArrayWithSize(dimensions)$
7: $\alpha^{(2)} \leftarrow sample\_random\_from\_hypersphere(\alpha^{(1)}, 1)$ ▷ By Muller's Method [51]
8: $A \leftarrow 0;\ B \leftarrow 0;\ C \leftarrow 0$
9: **for** $i \in range(0..dimensions)$ **do**
10: $\delta[i] \leftarrow \alpha^{(2)}[i] - \alpha^{(1)}[i]$
11: $\gamma[i] \leftarrow \beta[i] - \alpha^{(1)}[i]$
12: $A \leftarrow A + (\delta[i] * \delta[i])$ ▷ $A = \sum_{i=0}^{n} \delta_i^2$
13: $B \leftarrow B + (\delta[i] * \gamma[i])$ ▷ $\sum_{i=0}^{n} \delta_i \gamma_i$
14: $C \leftarrow C + (\gamma[i] * \gamma[i])$ ▷ $\sum_{i=0}^{n} \gamma_i^2$
15: **end for**
16: $B \leftarrow B * -2$ ▷ $B = -2(\sum_{i=0}^{n} \delta_i \gamma_i)$
17: $C \leftarrow C - (r * r)$ ▷ $C = (\sum_{i=0}^{n} \gamma_i^2) - r^2$
18: **return** $(-B + sqrt(B * B - 4 * A * C))/(2 * A)$ ▷ $\frac{-B+\sqrt{B^2-4AC}}{2A}$
19: **end function**

Let us first denote the micro-cluster mc_{temp} as HS_β which is a hyper-sphere with radius r and centred at $\beta = (\beta_1, \beta_2, \beta_3, ..., \beta_n)$, where n is the number of dimensions of the input space of the problem, the equation of this hyper-sphere is:

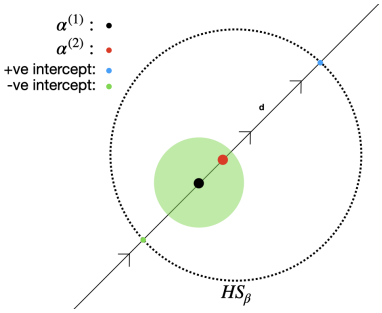
$$\sum_{i=0}^{n} (x_i - \beta_i)^2 = r^2 \tag{2}$$

Let us also denote the centre of $mc_{anchor}^{minority}$ to as $\alpha^{(1)} = (\alpha_1^{(1)}, \alpha_2^{(1)}, \alpha_3^{(1)}, ..., \alpha_n^{(1)})$ (the black dot in Figure 3(a)), which should always be inside HS_β. First of all, we need to pick a random direction from $\alpha^{(1)}$ (Figure 3(a)). This can be achieved by randomly and uniformly picking a point from a unit hyper-sphere centred at $\alpha^{(1)}$, using the Muller's method [51]. We then denote this point to as $\alpha^{(2)} = (\alpha_1^{(2)}, \alpha_2^{(2)}, \alpha_3^{(2)}, ..., \alpha_n^{(2)})$ (the red dot in Figure 3(a)) (line 7, Algorithm 4). Points $\alpha^{(1)}$ and $\alpha^{(2)}$ form an n-dimensional infinite long straight line (the line d in Figure 3(b)), whose parameterised equation is:

$$x_i = \alpha_i^{(1)} + t(\alpha_i^{(2)} - \alpha_i^{(1)}) \tag{3}$$

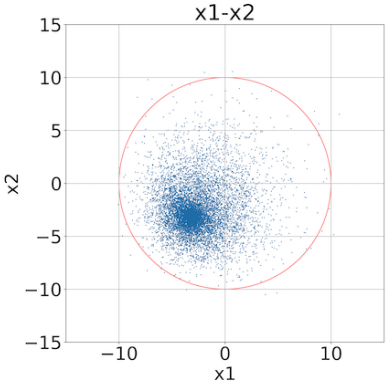


(a) Step 1



(b) Step 2

(c) Step 3



(d) Two-Dimensional Example of the Sampling Result

Fig. 3: Illustration of Sampling from mc_{temp}

where t is a scalar and $(\alpha_i^{(2)} - \alpha_i^{(1)})$ is the direction vector. To find the intercepts of this infinite long line to the hull of HS_β (the blue and green dots in Figure 3(b)), we can substitute Equation 3 into Equation 2[2]:

$$\sum_{i=0}^{n}((\alpha_i^{(2)} - \alpha_i^{(1)})t + (\alpha_i^{(1)} - \beta_i))^2 = r^2 \tag{4}$$

Let us denote $\delta_i = \alpha_i^{(2)} - \alpha_i^{(1)}$ and $\gamma_i = \beta_i - \alpha_i^{(1)}$ (line 10 and 11, Algorithm 4), then Equation 4 becomes:

$$\sum_{i=0}^{n}(\delta_i t - \gamma_i)^2 = r^2$$

[2]The idea is inspired by the discussion on https://math.stackexchange.com/questions/151064/calculating-line-intersection-with-hypersphere-surface-in-mathbbrn?rq=1

$$(\sum_{i=0}^{n} \delta_i^2)t^2 - 2(\sum_{i=0}^{n} \delta_i \gamma_i)t + (\sum_{i=0}^{n} \gamma_i^2) - r^2 = 0 \quad (5)$$

Let us denote $A = \sum_{i=0}^{n} \delta_i^2$ (line 12, Algorithm 4), $B = -2(\sum_{i=0}^{n} \delta_i \gamma_i)$ (line 13 and 16, Algorithm 4) and $C = (\sum_{i=0}^{n} \gamma_i^2) - r^2$ (line 14 and 17, Algorithm 4) to solve Equation 5 based on Bhaskara's equation:

$$t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

Here, we just take the positive root of t because it "follows" the direction vector, while the negative root "oppositely follows" the direction vector (the direction is denoted by the arrows on line d in Figure 3(b)). i.e.

$$t_{intercept} = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \quad (6)$$

Substituting $t_{intercept}$ into Equation 3 will obtain the intercept of the line and the hyper-sphere, following the direction vector (the blue dot in Figure 3(b)). Thus, to sample points within the HS_β, we can simply sample a scalar t_{sample} between 0 and $t_{intercept}$ (Figure 3(c)) and substitute it into Equation 3 to obtain the sampled point. As we want to sample this point with the highest chance at the target point $\alpha^{(1)}$, we can sample t_{sample} using Gaussian distribution with the mean = 0 and standard deviation = $\frac{t_{intercept}}{3}$. i.e.

$$g \sim N(0, (\frac{t_{intercept}}{3})^2)$$
$$t_{sample} = \mathrm{g}$$

At last, we substitute t_{sample} into Equation 3 to obtain the sample point.

The reason for setting the standard deviation to be $\frac{t_{intercept}}{3}$ is that we want the sampled point to be within the micro-cluster. Yet, the probability density function of the Gaussian distribution has no bounds. Thus, we set the $t_{intercept}$ at 3 standard score (z-score = 3), such that 99.9% area under the probability density function curve of the Gaussian distribution is between $-t_{intercept}$ and $+t_{intercept}$. Also, we want t_{sample} to "follow" the direction vector (i.e. we only interested in line segment between the black and the blue dots on d in Figure 3(b)), thus, we only accept the positive value of t_{sample}.

Figure 3(d) presents a two-dimensional example of using the aforementioned strategy to sample points in a hyper-sphere centred at (0,0) with a radius of 10. The points have the highest probability to be sampled at (-7,0)

4 Experiments to Evaluate the Predictive Performance of SMOClust

This section presents the design of the experiments to evaluate SMOClust. The predictive performance of SMOClust was first compared against five existing approaches from the literature on artificial data streams of different types

of drifts. This is to investigate for which types of drift SMOClust will be advantageous and disadvantageous, answering RQ2. SMOClust was then compared against the same set of existing approaches on real-world data streams to obtain a general idea of its performance in practical situations, answering RQ3. Massive Online Analysis (MOA) [50] was chosen to be the experimentation platform. Section 4.1 presents the details of artificial and real-world data streams used in the experiments. Section 4.2 presents the detailed setup of the experiments, including the procedure of hyper-parameter tuning and the evaluation method used in the experiments.

4.1 Data Streams

As discussed in Sections 1 and 2, data difficulty factors play a crucial role in class imbalanced data stream learning with concept drift. Therefore, it is important to evaluate class imbalance data stream learning approaches based on data streams with different data difficulty factors. In line with that, the artificial data stream generator proposed by [16] was adopted because it allows us to simulate concept drifts that affect different data difficulty factors, including the class imbalance ratio, movement of the clusters in the minority class, and the proportion of safe, borderline and rare minority class examples. We have generated a large variety of artificial data streams to avoid any bias in the evaluation and enable us to understand the conditions under which SMOClust performs well and the conditions under which it fails, as well as the reason for such behaviour.

Table 2 presents a summary of artificial data streams used in the experiments. Each of them has five numerical input attributes $\{x_i \in (-1, 1)\}_{i=1}^{5}$ and a class label $y_i \in \{0, 1\}$. They all consist of 200k examples where concept drift happens gradually from 70k to 100k time steps. The continuous movement of minority class sub-clusters in gradual drift scenarios creates a complex and dynamic environment for evaluation. We created thirty artificial data streams of each type with different random seeds. Each of the thirty streams is used to evaluate the data stream learning approaches in a single run. The evaluation method is detailed in Section 4.2

Following the default setting by [16], when the artificial data stream has no drift or no modifier specified, it is: 1) class balanced, 2) composed of a single cluster representing class 1, uniformly surrounded by the examples of class 0, and 3) examples only appear in safe regions. When the data stream is class imbalanced, class 1 is the minority class while class 0 is the majority class.

As shown in Table 2, we considered four groups of drift from [16]'s work in this study. The first group (Imbalance ratio drift) considers concept drift affecting the class imbalance ratio only. The second group (Single factor drift with static imbalance ratio) considers data streams with a static class imbalance ratio while the concept drift happens in the form of five factors, which were discussed by [16]: splitting, moving, merging clusters and decreasing the ratio of safe examples while increasing the ratio of borderline or rare examples. In the third (Double factor drift) and the fourth (Complex factor drift) groups,

Table 2: Summary of Artificial Data Streams

Imbalance Ratio Drift	Single Factor Drift with Static Imbalance Ratio
StaticIm10_Im1, StaticIm1_Im10, Im1, StaticIm1_Im50	StaticIm{30/10/1}_Split{3/7}, StaticIm{30/10/1}_Move{3/7}, StaticIm{30/10/1}_Merge{3/7}, StaticIm{30/10/1}_Borderline{20/100} StaticIm{30/10/1}_Rare{20/100}
Double Factor Drift	**Complex Factor Drift**
Im1+Rare100, Im10+Rare60, Split5+Im10, Im1+Borderline100, Im10+Borderline20	StaticIm10_Split5+Im1+Rare100, StaticIm10_Split5+Im1+Borderline100, Split5+Im10+Borderline40+Rare40, Split5+Im10+Borderline80, Im10+Borderline20+Rare20

- All artificial data streams have 200k examples, where a single concept drift occurs from 70k-th time step to 100k-th time step.
- "+" refers to the factors occurring simultaneously during the concept drift.
- StaticIm{N} refers to a static minority class ratio of N% throughout the entire stream.
- Im{N} refers to the minority class ratio of N% after the concept drift.
- Split{N}, Move{N}, Merge{N} refer to drifts which split, move and merge N clusters in the minority class respectively.
- Borderline{N}, Rare{N} refer to drifts changing N% of the minority class examples from appearing in a safe region of the clusters to being borderline region and rare cases respectively.

we have chosen ten artificial data streams (five for each group) with concept drift affecting two factors and a group of factors, respectively. These artificial data streams were chosen evenly across the lists of data streams from [16]'s work with double factor drift and complex factor drift in [16]'s work respectively. These lists were sorted by the average performance of the compared data stream learning approaches in their work. Thus, picking data streams evenly from these lists means that we are taking scenarios with different difficulty levels.

As the analysis which is presented in Section 4.3 shows that SMOClust performed well in severely imbalanced data streams, we performed additional experiments with the aforementioned single factor drift streams with more severe static class imbalance ratio to further evaluate SMOClust in extreme cases. These additional severely class imbalanced artificial data streams are summarised in Table 3. Note that, although we reused the static imbalance ratio of 1% minority class examples, we used another set of random seeds when performing these additional experiments.

Apart from experiments with artificial data streams, we also performed experiments with different real-world data streams to evaluate SMOClust in practical applications. These real-world data streams are summarised in Table 4 and their details are as follows.

The Luxembourg stream [52] was constructed from the European Social Survey from 2002 to 2007. The classification task is to predict whether internet usage is high or low. The NOAA stream [38] contains weather records collected over five decades (1949-1999). These records include temperature, pressure, wind speed, precipitation and other weather-related events. The classification task is to predict whether the next day will rain. The Ozone stream

Table 3: Summary of Single Factor Drift Artificial Data Streams with Severe Imbalance Ratio

Single Factor Drift with Severe Static Imbalance Ratio
StaticIm{5/3/1/07/05/03}_Split{3/7}, StaticIm{5/3/1/07/05/03}_Move{3/7}, StaticIm{5/3/1/07/05/03}_Merge{3/7}, StaticIm{5/3/1/07/05/03}_Borderline{20/100}, StaticIm{5/3/1/07/05/03}_Rare{20/100}

- All artificial data streams have 200k examples, where a single concept drift from 70k-th time step to 100k-th time step.
- StaticIm{N} refers to a static minority class ratio to be N% throughout the entire stream. StaticIm{0N} refers to a static minority class ratio to be 0.N% throughout the entire stream.
- Split{N}, Move{N}, Merge{N} refer to drifts which split, move and merge N clusters in the minority class respectively.
- Borderline{N}, Rare{N} refer to drifts changing N% of the minority class examples from appearing in a safe region of the clusters to being borderline region and rare cases respectively.

Table 4: Summary of Real-World Data Streams

Stream	#Examples (Pre)	#Examples (Actual)	#Nom. Attr.	#Num. Attr.	Imbalance Ratio (Pre)	Imbalance Ratio (Actual)
Luxembourg	190	1711	15	16	0.532:0.468	0.512:0.488
NOAA	1,815	16,344	0	8	0.698:0.303	0.685:0.315
Ozone	253	2,281	0	72	0.893:0.107	0.942:0.058
PAKDD2009	4999	44998	13	14	0.798:0.202	0.803:0.197
Covtype$_{(c_1=\{1\text{-}6\})}$	58,101	522,911	2	10	0.785:0.215	0.619:0.381
Covtype$_{(c_1=1)}$	58,101	522,911	2	10	0.595:0.405	0.524:0.476
Covtype$_{(c_1=2)}$	58,101	522,911	2	10	0.963:0.037	0.936:0.064
Covtype$_{(c_1=3)}$	58,101	522,911	2	10	0.963:0.037	0.999:0.001
Covtype$_{(c_1=4)}$	58,101	522,911	2	10	0.958:0.042	0.987:0.014
Covtype$_{(c_1=5)}$	58,101	522,911	2	10	0.963:0.037	0.971:0.029
Covtype$_{(c_1=6)}$	58,101	522,911	2	10	0.963:0.037	0.965:0.035
INSECTS$^{inc.}$	45,204	406,840	0	33	0.899:0.101	0.905:0.095
INSECTS$_{abr.}$	35,527	319,748	0	33	0.912:0.088	0.907:0.093
INSECTS$^{inc.}_{grad.}$	14,342	128,981	0	33	0.921:0.079	0.899:0.101
INSECTS$^{inc.}_{abr.\ re.}$	45,204	406,840	0	33	0.895:0.105	0.905:0.095
INSECTS$^{inc.}_{re.}$	45,204	406,840	0	33	0.895:0.105	0.905:0.095
Amazon	800	7,200	0	30	0.728:0.272	0.875:0.125
Twitter	909	8,181	0	30	0.814:0.186	0.846:0.154

- Total number of attributes = #Nominal attributes + #Numeric attributes + Class attribute.
- "Pre" refers to hyper-parameter tuning sets (i.e. the first 10% of the original data set). "Actual" refers to actual experiment sets (i.e. the remaining 90% of the original data set.
- Covtype$_{(c_1=x)}$: "c_1=x" refers to the class 1 is the class x in the original data set while the rest of the classes are combined to be the class 0 in the "Actual" experiment stream. "c_1=$\{x_0-x_n\}$" refers to the class 1 is the class x_0-x_n in the original data set combined while the rest of the classes are combined to be the class 0 in the "Actual" experiment stream.
- For all INSECTS data streams, "ae-albopictus" is the class 1. "inc." refers to incremental, "abr." refers to abrupt, "grad" refers to gradual, and "re." refers to recurring.

[53] consists of air measurements collected from 1998 to 2004. The task is to predict the ozone level eight hours ahead of time. The PAKDD2009 stream [54] consists of private label credit card application records and the task is to decide whether a given application should be approved. Forest Covertype

(Covtype) stream [55] contains the cartographic information about the forest of 30 × 30-meter cells and the task is to predict the cover type for a given cell. Covtype stream originally is a multi-class classification problem with seven forest cover types. To make it suitable for this study, it has been converted into seven binary classification streams. Each of them takes one of the forest cover types as one class while combining other forest cover types to be the other class. INSECTS streams [56] were constructed using a smart trap with optical sensors to collect the flying data of three different species of insects in a non-stationary environment for around three months. The temperature of the data collection environment was controlled to simulate concept drifts. INSECTS streams originally have six classes: three species of mosquitoes with two genders. We converted them into binary classification tasks by combining classes belonging to the species of ae-albopictus as the minority class while combining the rest of the classes as the majority class. Also, it has to note that [56] originally proposed seven INSECTS streams but we only adopted six of them which contain concept drifts and left the INSECT-out-of-control stream unused as it does not contain any concept drift. The Amazon stream [57] comprises reviews of books, DVDs, electronics, and kitchen appliances. Reviews with a rating greater than 3 were labelled as positive. The objective is to discern whether a review has a rating above 3. The Twitter stream [58] consists of labelled tweets about popular topics. The goal is to predict whether the sentiment of a given tweet is positive or negative.

To facilitate analysing the predictive performance of SMOClust, we also analysed the characteristics of the minority class of the real-world data streams, including the potential number of clusters, and the ratios of safe, borderline, rare and outlier examples. Note that we only analysed the portion of the real-world data streams used in the actual experiments, which excludes the first 10% of each original real-world data stream that was used for hyper-parameter tuning (see Section 4.2 for details the hyper-parameter tuning procedure). The procedure of this analysis follows the methodology proposed by [16] and is described as follows.

The characteristics of each real-world data stream are estimated in successive batched of examples. We followed [16] to use a batch size of 2000 examples for all data streams except for Luxembourg, NOAA, Amazon, and Twitter, where a batch size of 200 was used as these data streams have less than 10,000 examples. The class imbalance ratio and the ratios of each minority class type are estimated for each batch. It is worth noting that we only focused on analysing the class 1 because it is the global minority class of all the real-world data streams (see Table 4), even though this class could potentially become a majority during certain periods of the data stream, e.g., when there is potential concept drift affecting $P(Y)$, changing the roles of majority and minority classes temporarily. As for types of minority class examples, they were estimated using the method proposed by [17]. This method first finds the k-Nearest neighbours of each minority class example. Based on the

class ratios among these k-Nearest neighbours, it then categorises each minority class example as safe, borderline, rare, or outlier. Here, we followed [17] to adopt $k = 5$.

Following [16]'s procedure, we also estimated the number of minority class clusters in each batch, using the affinity propagation algorithm [59] and removing clusters with less than six minority class examples [16]. The affinity propagation algorithm was run thirty times with different random seeds for each batch. The average estimated number of minority class clusters is then recorded.

Lastly, we reported the ranges of the aforementioned characteristics across the different batched and their medians in Table 5. Note that we only performed analysis about types of minority class examples and the potential number of clusters on batched that contain at least six $(k + 1)$ minority class (class 1) examples. This is to prevent always categorising the minority class examples as rare cases or outliers when the total number of minority class examples in the batch is extremely low. The number of batches with less than size minority class examples is reported in brackets in the third column of Table 5.

As shown in Table 5, PAKDD2009 and NOAA streams usually present the most number of clusters of minority class examples, with medians of twenty-eight clusters, meaning that the minority class is split into several clusters in this data stream. INSECTS streams usually present fewer clusters of the minority class than PAKDD2009 and NOAA streams, which have medians ranging from thirteen to sixteen clusters. Luxembourg, Ozone, Covtype, Amazon and Twitter streams usually present the least number of clusters of the minority class, having medians ranging from zero to six.

As for the types of minority class examples, Table 5 shows that the Ozone, PAKDD2009, INSECTS, Amazon, and Twitter streams mainly consist of borderline, rare, and outlier minority class examples. Luxembourg and NOAA streams mainly consist of safe and borderline minority class examples. Most Covtype streams mainly consist of safe minority class examples. Regarding the minority ratios, most of them have a small range, indicating that the potential concept drifts only affect $P(Y)$ with mild severity. In contrast, $\text{Covtype}_{(c_1=\{1\text{-}6\})}$, $\text{Covtype}_{(c_1=1)}$ and $\text{Covtype}_{(c_1=2)}$ streams have a very large range, indicating that that they potentially present severe concept drifts affecting $P(Y)$. In particular, $\text{Covtype}_{(c_1=2)}$ presents a large range of minority class ratio with a very small median (1%). This may indicate that the severe concept drifts affecting $P(Y)$ could potentially be abrupt.

4.2 Experiment Setup

This section presents the procedure of hyper-parameter tuning and experiments. The following are the approaches from the literature that were considered in this study and the reason behind the choice. All of these approaches are strict online approaches, which do not require storage of any past data, so that the comparisons are fair.

Table 5: Characteristics of Real-World Data Streams (Values in the brackets are the median)

Stream	#Examples	#Batches (Uncounted)	Est. #Clust. (Median)	Minority Ratio	Safe	Borderline	Rare	Outlier
Luxembourg	1711	9 (0)	4-8 (6)	46%-53% (48%)	47%-62% (51%)	32%-48% (42%)	3%-9% (4%)	0%-3% (1%)
NOAA	16344	9 (0)	9-33 (28)	25%-37% (31%)	29%-45% (33%)	33%-50% (43%)	11%-18% (15%)	6%-13% (9%)
Ozone	2281	12 (5)	0-2 (0)	0%-18% (4%)	0%-22% (0%)	0%-61% (17%)	14%-58% (30%)	5%-70% (33%)
PAKDD2009	44998	23 (0)	19-32 (28)	17%-22% (20%)	0%-4% (2%)	27%-38% (32%)	33%-41% (36%)	22%-34% (30%)
$\text{Covtype}_{(c_1=\{1\text{-}6\})}$	522911	262 (8)	1-26 (1)	0%-76% (36%)	66%-99% (92%)	0%-26% (6%)	0%-8% (1%)	0%-5% (1%)
$\text{Covtype}_{(c_1=1)}$	522911	262 (2)	1-36 (1)	0%-91% (46%)	69%-100% (93%)	0%-25% (5%)	0%-5% (1%)	0%-4% (0%)
$\text{Covtype}_{(c_1=2)}$	522911	262 (149)	0-33 (5)	0%-89% (0%)	0%-100% (90%)	0%-78% (7%)	0%-14% (2%)	0%-11% (1%)
$\text{Covtype}_{(c_1=3)}$	522911	262 (240)	0-6 (1)	0%-3% (0%)	0%-62% (44%)	0%-67% (19%)	0%-38% (16%)	9%-67% (20%)
$\text{Covtype}_{(c_1=4)}$	522911	262 (108)	0-9 (3)	0%-10% (1%)	0%-100% (68%)	0%-79% (23%)	0%-56% (6%)	0%-50% (2%)
$\text{Covtype}_{(c_1=5)}$	522911	262 (159)	0-14 (3)	0%-28% (0%)	0%-99% (81%)	0%-61% (13%)	0%-38% (3%)	0%-29% (2%)
$\text{Covtype}_{(c_1=6)}$	522911	262 (110)	0-17 (2)	0%-21% (1%)	0%-100% (91%)	0%-93% (6%)	0%-33% (1%)	0%-22% (1%)
$\text{INSECTS}^{\text{inc.}}$	406840	204 (1)	0-27 (15)	0%-19% (9%)	0%-24% (5%)	0%-46% (32%)	0%-42% (29%)	14%-100% (33%)
$\text{INSECTS}_{\text{abr.}}$	319748	160 (0)	0-49 (13)	0%-39% (9%)	0%-50% (5%)	0%-53% (32%)	0%-46% (27%)	4%-100% (32%)
$\text{INSECTS}^{\text{inc.}}_{\text{grad.}}$	128981	65 (0)	7-27 (16)	5%-19% (10%)	0%-43% (8%)	17%-46% (33%)	5%-42% (27%)	8%-47% (28%)
$\text{INSECTS}^{\text{inc.}}_{\text{abr. re.}}$	406840	204 (1)	0-26 (15)	0%-18% (10%)	0%-29% (5%)	0%-57% (32%)	7%-41% (28%)	6%-93% (32%)
$\text{INSECTS}^{\text{inc.}}_{\text{re.}}$	406840	204 (1)	0-26 (15)	0%-17% (10%)	0%-27% (4%)	0%-58% (32%)	0%-41% (28%)	5%-100% (32%)
Amazon	7200	36 (20)	1-5 (4)	0%-34% (0%)	0%-13% (4%)	27%-55% (40%)	25%-46% (36%)	10%-38% (17%)
Twitter	8181	41 (0)	0-7 (2)	5%-41% (13%)	0%-22% (0%)	0%-52% (20%)	0%-59% (31%)	9%-90% (41%)

- $\text{Covtype}_{(c_1=x)}$: "c_1=x" refers to the class 1 is the class x in the original data set while the rest of the classes are combined to be the class 0 in the "Actual" experiment stream. "c_1=$\{x_0$-$x_n\}$" refers to the class 1 is the class x_0-x_n in the original data set combined while the rest of the classes are combined to be the class 0 in the "Actual" experiment stream.
- For all INSECTS data streams, "ae-albopictus" is the class 1. "inc." refers to incremental, "abr." refers to abrupt, "grad" refers to gradual, and "re." refers to recurring.

- $\text{OOB}_{(\text{d})}$ and $\text{UOB}_{(\text{d})}$ [20]: Baseline approaches that use simple oversampling or undersampling to deal with class imbalance in data stream learning.
- $\text{OnlineUnderOverBagging}_{(\text{d})}$ ($\text{oUnderOverB}_{(\text{d})}$) [11]: A simple existing approach which combines simple undersampling and oversampling for class imbalance data stream learning. We slightly modified it to use time decay class sizes with the "oversampling" equation from OOB to controlling the resampling rate. We chose to adopt the "oversampling" equation from OOB because the research paper [11] explicitly states that the resampling rate for OnlineUnderOverBagging should be greater than 1. On the other hand, the "undersampling" equation from UOB produces a fractional number, which is not suitable in this context.
- VFC-SMOTE [15]: An existing approach which addresses class imbalance by generating synthetic minority class examples using histogram-based summaries of past examples.
- SMOTE-OB [23]: An existing approach which incorporates the class imbalance adaptation strategy of VFC-SMOTE into OnlineUnderOverBagging[11].
- $\text{OnlineOversampling}_{(\text{d})}$ ($\text{oOS}_{(\text{d})}$): A variant of the proposed approach which always uses the most recently seen minority class example for oversampling. This approach is used as a baseline to support the investigation of when the proposed strategy of creating synthetic minority class examples for oversampling is advantageous / disadvantageous.
- SMOGauNoise: A variant of the proposed approach inspired by [29], which proposed a data augmentation method for software effort estimation. SMOGauNoise has the same learning and making prediction strategies as the proposed approach but it always creates synthetic minority class examples for oversampling by adding Gaussian noise to the most recent minority class example. Note that this is the first time to investigate [29]'s data augmentation method in the context of classification problems.

Approaches followed by "(d)" refers to these approaches that were not designed to handle concept drift originally[3]. We used a wrapper to enable them to use a concept drift detector. Their system reset upon concept drift detection.

For the evaluation method, we modified the periodic holdout test for the experiments with artificial data streams. This modified periodic holdout test takes the data difficulty factors into the consideration, which includes the position of the minority class clusters, class imbalance ratio, and the proportions of borderline and rare examples. During a single run, the data stream learning approach was tested on a holdout test set B_t^{test} of m examples after training on every n example. Its predictive performance in G-Mean was then recorded. The holdout test sets are class balanced and they follows the same underlying joint probability distribution (concept) at the evaluation time step t, where $t \mod n = 0$, i.e., $B_t^{test} \sim P_t(X, Y)$. At the end of the run, we summarised

[3]Except OOB and UOB can handle concept drift affecting $P(Y)$.

their performance across the stream by taking an average of their G-Mean performance on the test sets.

For hyper-parameter tuning purposes, an additional artificial data stream was created. It also consists of 200k examples where the concept drift happens from 70k to 100k time steps but the class imbalance ratio and the drift behaviour were randomly selected from the set of all combinations of drift factors used in [16]. We denote this data stream as the "hyper-parameter tuning stream". The set of hyper-parameter values of each approach that leads to the best ten runs average of G-Mean across this stream was then used in the experiments. In the experiments, we adopted thirty runs rather than ten runs to reduce the effect of randomness on the results.

Experiments with real-world data streams have a similar procedure. The first 10% or each real-world data stream was used for the hyper-parameter tuning purposes. The prequential evaluation was used because the underlying concepts are unknown in advance. The set of hyper-parameter values of each approach that leads to the best ten runs average of G-Mean across the first 10% of each real-world data stream was then chosen to be used in the experiment of the corresponding data stream which consists of the remaining 90% of examples. The time decayed G-Mean performance was sampled at every 500 time steps, except they were sampled at every fifty time steps for NOAA, Ozone, Amazon, Twitter streams and every ten time steps for Luxembourg stream due to the fact that these streams are a lot shorter than other streams (i.e., they have a lot fewer examples than other data streams). Thus, sampling at shorter intervals allows us to see how the performance of the approaches changes throughout these relatively short data streams. We adopted a time decay factor of 0.999 to make their past predictive performance less important to the current time step. We recorded their thirty runs average G-Mean performance across each stream for evaluation and comparative analysis.

At the end of the experiments, the predictive performance of the approaches was compared by different concept drift data difficulty factors. The corresponding rankings in the groups were then presented. Friedman test with a level of significance of 0.05 was applied to each group, confirming if there is any statistical significance between the predictive performance of different approaches. If there is, Nememyi post-hoc test was used to determine which approaches performed significantly different from the top-ranked approach. In the statistical tests, each group corresponds to a data stream learning approach while each observation within a group corresponds to the average predictive performance across a given data stream in a single run. The thirty runs average predictive performance of the approaches are also reported to facilitate us in analysing the margin of the performance difference.

4.3 Results with Artificial Data Streams

This section presents the analysis done to compare the predictive performance of SMOClust against existing approaches on artificial data streams which consider different drift difficulties in the minority class. General comparisons

are first given based on the Friedman rankings of average G-Mean of the approaches grouped by different drift difficulty factors, presented in Table 6. It is then followed by a detailed analysis of the behaviour of SMOClust in representative cases where it performed better and worse than existing approaches in Sections 4.3.1 and 4.3.2 respectively.

Table 6: Statistical (Friedman) Ranking of G-Mean on Artificial Streams Grouped by Factors

Groups	OOB	UOB	oOS	oUnder-OverB	OOB_d	UOB_d	oOS_d	oUnder-$OverB_d$	SMO-Gau-Noise	VFC-SMO-TE	SMO-TE-OB	SMO-Clust
Imbalance Ratio Drift	3.77	4.44	4.72	3.27	7.35	7.59	8.37	5.05	6.93	11.65	9.1	5.76
Double Factor	6.03	7.24	5.71	6.85	4.2	7.39	6.05	5.47	2.67	11.38	9.21	5.80
Complex Factor	5.16	8.2	5.27	6.4	4.27	6.09	6.88	4.99	2.94	11.71	10.4	5.68
Single Factor Drift with Static Imbalance Ratio												
StaticIm{*}[a]_Split	7.04	6.93	7.69	5.63	4.96	6.93	6.09	3.72	2.97	11.6	9.48	4.97
StaticIm{*}[a]_Move	5.13	6.97	4.75	4.65	5.93	8.04	6.66	5.34	3.19	11.84	9.96	5.53
StaticIm{*}[a]_Merge	5.40	6.84	4.70	3.84	6.38	8.01	6.60	4.74	3.30	11.77	10.01	6.41
StaticIm{*}[a]_Borderline	5.89	6.17	5.91	4.26	5.46	7.65	7.91	4.08	3.47	11.61	9.09	6.51
StaticIm{*}[a]_Rare	2.52	6.98	3.04	4.34	6.6	7.56	8.58	5.72	4.06	11.3	9.26	8.04
StaticIm30_{*}[b]	5.84	9.02	3.40	6.94	3.81	7.93	3.75	5.54	2.19	11.27	11.12	7.20
StaticIm10_{*}[b]	4.89	8.98	5.03	3.87	4.14	9.17	7.49	3.56	2.23	11.91	7.77	8.96
StaticIm1_{*}[b]	4.86	2.34	7.23	2.83	9.64	5.81	10.27	5.06	5.78	11.69	9.78	2.72
All	5.16	6.78	5.23	4.90	5.63	7.43	7.12	4.87	3.58	11.61	9.57	6.12

[a] "StaticIm{*}[a]" refers to StaticIm{30/10/1}, which means the group includes all artificial data streams of that type in static minority class ratio of 30%, 10%, and 1% respectively.
[b] "{*}" refers to Split/Move/Merge/Borderline/Rare, which means the group includes all artificial data streams of the above five types of drifts with the same static minority class ratio.
- Smaller values for the rankings are better values.
- The p-values of Friedman tests are all $\leq$2.2E-16.
- Highlighted ranks denote significant superior performance.
- Underlined ranks denote the corresponding approach's performance have no statistical significance with SMOClust.

Table 6 shows that SMOClust was one of the top-ranked approaches when the data stream is extremely class imbalanced (minority class ratio: 1%), indicating that SMOClust handled extremely class imbalanced data stream better than most existing approaches, while it performed similarly to UOB and OnlineUnderOverBagging. However, SMOClust was one of the low-ranked approaches in the group of rare cases, indicating that it could not handle rare cases very well. For other groups, although SMOClust was not one of the top-ranked approaches, it usually performed similarly to mid-ranked approaches.

As Friedman rankings only show the relative position of approaches' predictive performance but they do not provide any information about the margin of difference. To investigate how much did SMOClust performed better in

severely class imbalanced streams and worse in other groups of factors, we further compared their thirty runs average G-Mean on each artificial data stream. The results of their difference in average G-Mean are presented in the form of a heat-map in Figure 4. Green cells indicate results favourable to SMOClust, whereas red cells indicate results favourable to the compared approach. For a comprehensive table of the predictive performance of the approaches, please refer to the supplementary document.

Table 6 shows that SMOClust usually obtained lower rankings than other approaches in less severe class imbalanced data streams. However, Figure 4 reveals that the margin of the under-performance was usually small as we can rarely see saturated red cells in the table. In contrast, the high ranking achieved by SMOClust in the group of StaticIm1_{*} was supported by a lot of saturated green cells in the sector StaticIm1 of Figure 4, meaning that SMOClust performed a lot better than existing approaches in cases with severe class imbalanced ratio. Besides, Figure 4 further confirms that SMOClust could not handle rare minority class examples very well as we can see that cases involving Rare100 drift have lots of saturated red cells. In particular, OOB and OnlineUnderOverBagging handled rare minority class examples better than SMOClust.

One potential reason why SMOClust did not perform well in handling data streams with a large proportion of rare minority class examples is the conservative nature of the proposed synthetic example generation method, where most synthetic examples are generated in the dense area of the minority class. To address this, it might be helpful to generate synthetic examples in a more diverse manner. However, generating synthetic examples diversely can also introduce a significant amount of noise or even create artificial concept drifts. Moreover, it can be challenging to ensure that a certain area belongs to the minority class if there are no real minority class examples in that area. The proposed method is less prone to these risks and uncertainties, while overcoming the problems of existing work, which ignore data difficulty factors and rely on caching all (minority class) examples for synthetic minority class oversampling.

Comparing the predictive performance of SMOClust against UOB and OnlineUnderOverBagging in the group of StaticIm1_{*}, Table 6 shows that they performed similarly. Yet, the sector of StaticIm1 in Figure 4 reveals that SMOClust performed better than UOB by small margins (around 1-2% G-Mean, light green cell) in cases presenting concept drift of increasing rare minority class ratio, yet, it performed worse than UOB by medium-small margins (around 3% G-Mean, light red cells) in cases presenting concept drift of moving and merging minority class clusters. SMOClust performed better than OnlineUnderOverBagging by medium-small margins (around 2-3% G-Mean, light green cells) in cases presenting a concept drift of splitting minority class clusters. However, it performed slightly worse than OnlineUnderOverBagging (around 1% G-Mean, light red cell) in cases presenting concept drift of merging minority class clusters. It also performed worse than

Fig. 4: Difference in Average G-Mean Against SMOClust on Class Imbalanced Artificial Data Streams Based on 30 Runs (Green cells indicate SMOClust performed better; Red cells indicate SMOClust performed worse; Grey horizontal lines separate different groups of data streams, i.e., StaticIm{30/10/1}, Imbalance Ratio Drift, Double Factor, and Complex Factor)

OnlineUnderOverBagging by a large margin (around 7% G-Mean, saturated red cell) in StaticIm1_Rare100 case. In short, SMOClust performed similarly to both UOB and OnlineUnderOverBagging in most StaticIm1 cases, except OnlineUnderOverBagging performed a lot better in StaticIm1_Rare100 case.

When comparing the predictive performance of SMOClust against two approaches that also summarise past knowledge to support the generation of synthetic examples (VFC-SMOTE and SMOTE-OB), Table 6 and Figure 4 show that SMOClust performed better in most cases, especially in StaticIm1 cases. This indicates that the proposed synthetic minority oversampling strategy in SMOClust is superior.

Based on the aforementioned results, additional experiments were performed with the same set of single factor drift artificial data streams but enforced with extremely severe class imbalance ratios (minority class ratio 0.3% to 5%, summarised in Table 3) to further evaluate if SMOClust can usually perform better than existing approaches in extremely class imbalanced data streams.

Table 7 presents the Friedman rankings of average G-Mean by groups of different drift difficulty factors on the severely class imbalanced artificial data streams. It shows that SMOClust can indeed achieve higher rankings when the class imbalance ratio is very severe (minority class ratio $\leq 1\%$). Figure 5 presents the difference in average G-Mean (based on thirty runs) between the compared approaches and SMOClust on severely class imbalanced artificial data streams in the form of a heat-map with the same colour scheme as Figure 4. Similarly, please refer to the supplementary document for a comprehensive table of the predictive performance of the approaches. It supports the aforementioned deduction with a lot of saturated green cells in the cases of minority class ratio $\leq 1\%$, indicating the superior performance of SMOClust. The exception here is the comparison against UOB, with the margin of underperformance increasing as the severity of the class imbalance ratio increases by case. When compared against OnlineUnderOverBagging, SMOClust generally performed better in cases other than Rare100 drift, with the margin of superior performance increasing as the severity of the class imbalance ratio increases by case.

Figure 5 also confirms that SMOClust usually does not handle rare minority class examples very well, especially when compared against OOB, OnlineUnderOverBagging and SMOGauNoise. However, an extremely severe class imbalance ratio may give advantage to SMOClust in dealing with Rare100 drift as cases involving Rare100 present less saturated red cells when the class imbalance ratio is $\leq 1\%$. In particular, the case of StaticIm03_Rare100 presents a row of saturated green cells. Anyhow, these results are consistent with previous results of experiments with less severe class imbalanced artificial data streams.

Besides, Table 7 also shows that SMOClust could not achieve high rankings in the groups concerning minority class ratio of 5% and 3%. This may due to the fact that the artificial data streams are long enough to have quite

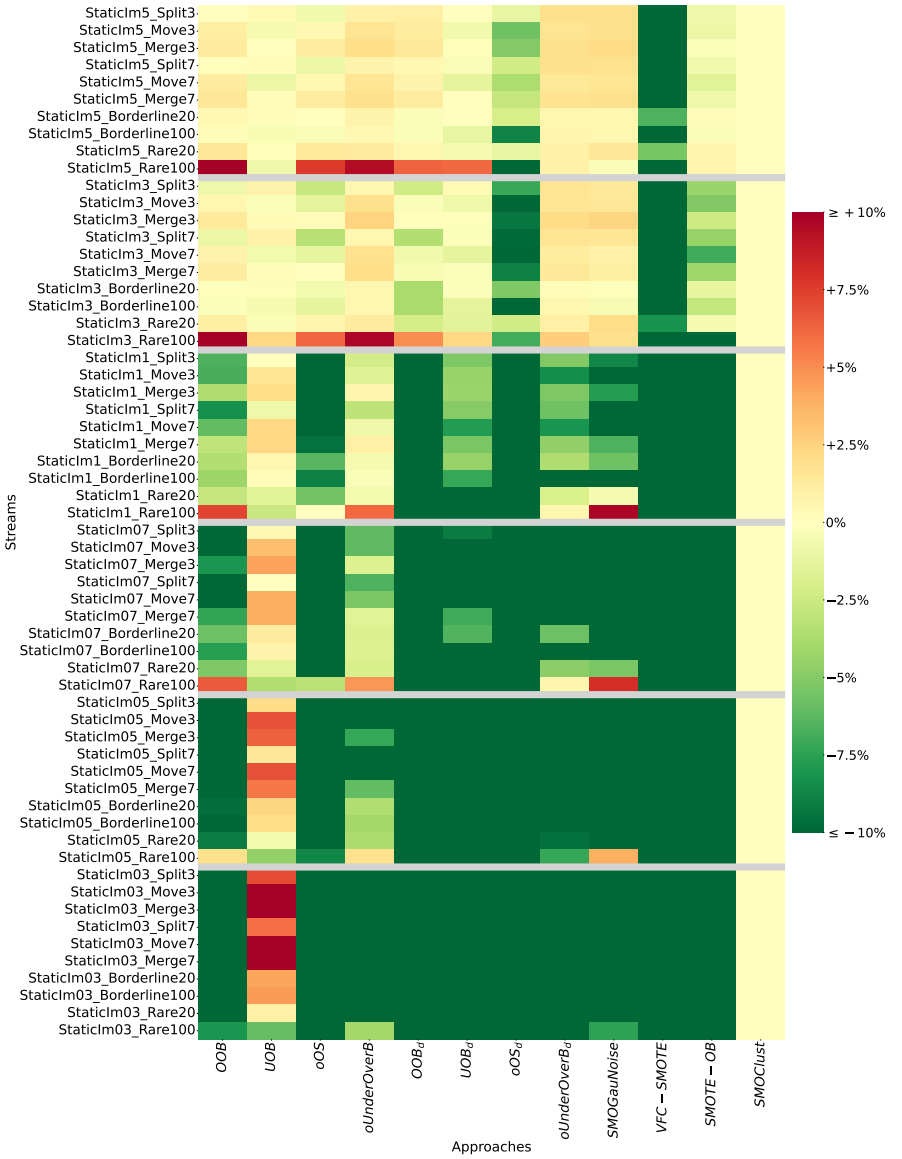


Fig. 5: Difference in Average G-Mean Against SMOClust on Severely Class Imbalanced Artificial Data Streams Based on 30 Runs (Green cells indicate SMOClust performed better; Red cells indicate SMOClust performed worse; Grey horizontal lines separate different groups of data streams, i.e., StaticIm{5/3/1/07/05/03})

Table 7: Statistical (Friedman) Ranking of G-Mean on Severely Class Imbalanced Artificial Streams Grouped by Factors

Groups	OOB	UOB	oOS	oUnder-OverB	OOB_d	UOB_d	oOS_d	oUnder-$OverB_d$	SMO-Gau-Noise	VFC-SMO-TE	SMO-TE-OB	SMO-Clust
StaticIm{*}[a]_Split	5.78	2.61	8.18	3.62	8.58	5.30	9.90	4.17	4.84	11.92	9.81	3.30
StaticIm{*}[a]_Move	4.80	3.20	7.53	2.82	7.62	6.25	10.25	4.76	5.13	11.96	10.12	3.57
StaticIm{*}[a]_Merge	4.78	3.09	7.19	2.64	8.15	6.06	10.26	4.66	5.14	11.80	10.03	4.20
StaticIm{*}[a]_Borderline	4.68	2.73	6.72	2.63	9.52	6.39	10.52	4.92	6.12	11.32	9.20	3.25
StaticIm{*}[a]_Rare	3.16	5.33	5.64	2.81	8.81	7.78	9.84	5.29	3.84	11.38	9.68	4.44
StaticIm5_{*}[b]	4.29	7.56	6.44	2.53	5.39	8.23	10.05	3.17	2.57	11.94	8.25	7.60
StaticIm3_{*}[b]	4.48	5.60	7.34	2.26	7.54	6.89	10.41	2.87	3.30	11.95	9.48	5.88
StaticIm1_{*}[b]	4.99	2.36	7.32	2.92	9.05	5.60	10.42	5.31	5.77	11.71	9.89	2.65
StaticIm07_{*}[b]	4.80	1.93	7.21	3.25	9.76	5.38	10.44	5.49	6.15	11.48	9.81	2.29
StaticIm05_{*}[b]	4.62	1.61	6.97	3.27	10.10	5.88	10.05	5.76	6.11	11.36	10.17	2.11
StaticIm03_{*}[b]	4.65	1.31	7.03	3.18	9.37	6.17	9.56	5.97	6.16	11.61	11.00	1.99
All	4.64	3.39	7.05	2.90	8.54	6.36	10.15	4.76	5.01	11.68	9.77	3.75

[a] "StaticIm{*}[a]" refers to StaticIm{5/3/1/07/05/03}, which means the group includes all artificial data streams of that type in static minority class ratio of 5%, 3%, 1%, 0.7%, 0.5%, and 0.3% respectively.
[b] "{*}" refers to Split/Move/Merge/Borderline/Rare, which means the group includes all artificial data streams of the above five types of drifts with the same static minority class ratio.
- Smaller values for the rankings are better values.
- The p-values of Friedman tests are all $\leq$2.2E-16.
- Highlighted ranks denote significant superior performance.
- Underlined ranks denote the corresponding approach's performance have no statistical significance with SMOClust.

a lot of minority class examples, despite the minority class ratios were low. Therefore, the advantage of SMOClust was not manifested. The sectors of StaticIm5 and StaticIm3 on Figure 5 show that SMOClust usually performed slightly worse than most existing approaches but it performed better than $OnlineOversampling_d$, VFC-SMOTE and SMOTE-OB.

Considering all cases in Figure 5, we can see that, when the minority class ratio decreases, SMOClust usually had a smaller margin of performance reduction than other approaches, except UOB. This shows that the aggressive nature of undersampling may be generally more advantageous than oversampling when the number of minority class examples in the data stream is extremely low. Yet, we can still see from Figure 5 that SMOClust performed better than UOB in most cases of Rare100 drift. This means that, when the minority class has extreme low number of examples and is difficult to learn, SMOClust still has more advantage than undersampling. One reason could be the fact that the compared approaches focus on learning the most recent decision areas of both classes, whereas SMOClust was designed to reinforce its knowledge in past minority class decision areas. This means that SMOClust is likely to have a better generalisation on the sub-areas of the minority class than existing approaches.

In the following sections, representative cases were chosen to discuss why SMOClust performed better and worse than existing approaches respectively, providing a more detailed understanding of the results.

4.3.1 Cases where SMOClust performed better

This section discusses why SMOClust performed better than most other approaches in artificial data streams with severe class imbalance ratio when the class imbalance ratio is extremely severe (minority class ratio $\leq 1\%$ throughout the stream). StaticIm1-Move7 stream was chosen from Figure 4 as the representative case to discuss the behaviour of SMOClust in details.

As mentioned in Section 4.1, the artificial data streams have five input attributes and a class label. Therefore, it is difficult to visualise the learnt decision areas of the approaches and understand their behaviour in details. Because of this, we created a version of the representative streams with two input attributes and a class label while preserving the characteristics which include the class imbalance ratio and the drift difficulty factors etc. Note that we only created a single copy of each two-dimensional representative stream, such that we can compare the data stream learning approaches with their median predictive performance in thirty runs on the same data stream. Also, the hyper-parameters of the approaches were tuned based on a separated random two-dimensional artificial data stream, following the procedure explained in Section 4.2.

Table 8 presents the their thirty runs average G-Mean on the two-dimensional version of StaticIm1_Move7 stream. It shows that SMOClust performed the best. These results are slightly inconsistent with the results of the corresponding five dimensional stream in Figure 4, where SMOClust performed slightly worse than UOB but similarly to OnlineUnderOverBagging. Yet, in general, SMOClust still performed better than other approaches in both two-dimensional and five dimensional versions of StaticIm1_Move7 stream. This may indicate that SMOClust tends to perform better in low-dimensional data stream. Anyhow, the detailed analysis presented in the following paragraphs can still explain the characteristics of SMOClust and why it performed better than most other approaches in this representative case.

Figure 6 presents the approaches' predictive performance over time steps of their median run[4]. To maintain readability, we omitted the predictive performance of OOB_d, UOB_d, oOS_d, $\text{oUnderOverB}_\text{d}$, VFC-SMOTE, and SMOTE-OB from Figure 6, as their performance fluctuates significantly throughout the stream. For the comparison of SMOClust against these approaches, please refer to the supplementary document. It shows that SMOClust performed the best in most time steps. In particular, SMOClust maintained the predictive performance to have at least 50% G-Mean on the class balanced holdout test sets during the concept drift (from 70k to 100k time steps) and recovered from

[4]Median run refers to the run that leads to the median of predictive performances averaged across time steps.

Table 8: 30 Runs Average G-Mean on Two-Dimensional Version of Representative Artificial Data Streams where SMOClust Performed Better

Stream	OOB	UOB	oOS	oUnder-OverB	OOB_d	UOB_d
StaticIm1_Move7	82.11%	76.3%	79.46%	85.26%	53.45%	56.94%

Stream	oOS_d	oUnder-$OverB_d$	SMO-GauNoise	VFC-SMOTE	SMOTE-OB	SMOClust
StaticIm1_Move7	76.88%	45.12%	82.94%	1.04%	33.09%	91.23%

- Based on the average G-Mean, cells are highlighted in lime / light grey when SMOClust performed better than the corresponding approach and cells are highlighted in orange / dark grey cells when SMOClust performed worse than the corresponding approach. The colour intensity scales with the absolute difference of average G-Mean between the SMOClust and the approach of the column and the intensity reaches the maximum when such difference is $\geq 10\%$.

the drift better than other approaches (the solid red line has a rapid recovery since 100k time steps). In contrast, other approaches usually dropped to around 0-20% G-Mean during the drift. This case showed the superior performance achieved by SMOClust in handling severely class imbalanced drifting data streams.

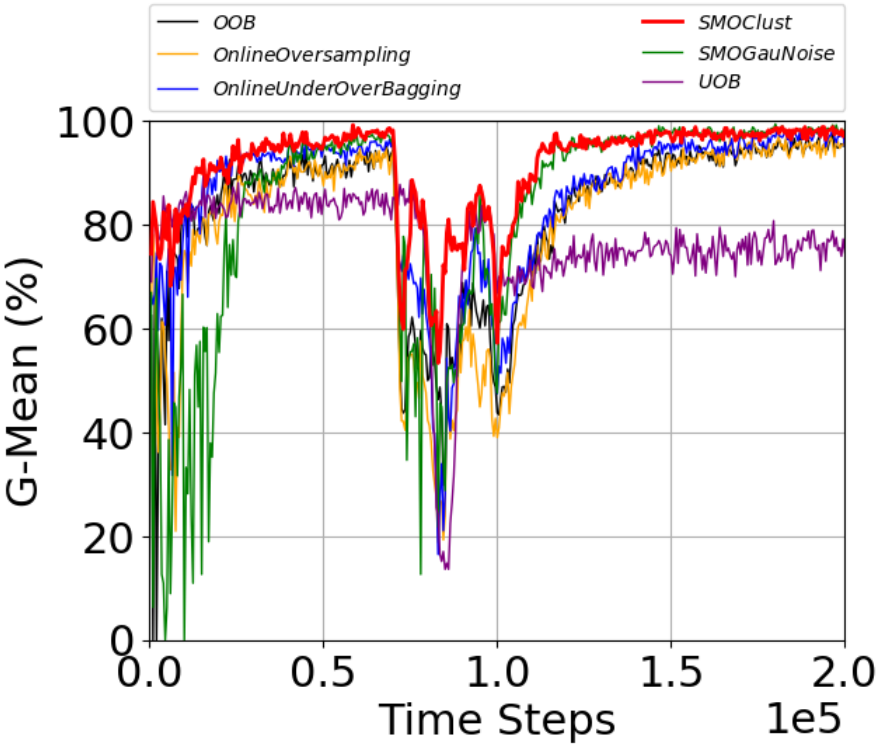
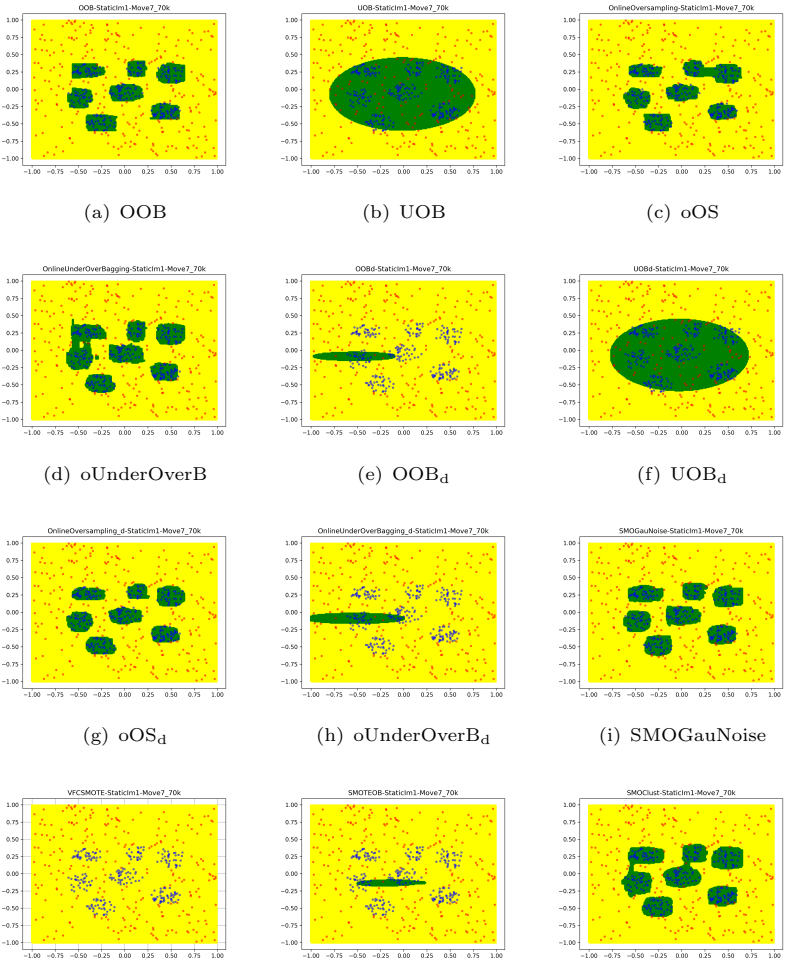


Fig. 6: Periodic Class Balanced Holdout Test G-Mean Against Time Steps in Two-Dimensional StaticIm1_Move7

Figures 7, 8 and 9 visualise the learnt decision areas of approaches at the time steps right before and after concept drift (70k and 100k time steps) and at the end (200k time steps) of the two-dimensional StaticIm1_Move7 stream respectively. The yellow and green regions represent their learnt decision areas of class 0 (majority class) and class 1 (minority class) respectively, while the red and blue dots are the class 0 (majority class) and class 1 (minority class) examples in the class balanced test set, corresponding to the time steps.



(a) OOB (b) UOB (c) oOS

(d) oUnderOverB (e) OOB_d (f) UOB_d

(g) oOS_d (h) $oUnderOverB_d$ (i) SMOGauNoise

(j) VFC-SMOTE (k) SMOTE-OB (l) SMOClust

Fig. 7: Decision Areas Against Class Balanced Test Set at 70k Time Steps (Before Drift) of Two-Dimensional StaticIm1_Move7

First of all, we compare the learnt decision areas of the approaches at the time steps right before concept drift (at 70k time steps). Figure 7 shows that OOB, OnlineOversampling, OnlineUnderOverBagging, SMOGauNoise and SMOClust had learnt decision areas which match the corresponding class balanced test set. This explains why they performed the best before the drift (0-70k time steps, Figure 6). Figure 7(i) and Figure 7(l) show that the learnt decision areas of SMOClust were similar to SMOGauNoise because they both have strategies to explore the minority class decision boundaries. The expansion by SMOClust was slightly more aggressive than SMOGauNoise, with some sub-areas linked together. Although the proposed synthetic minority oversampling strategy prioritises "safe" areas to generate synthetic minority class examples, the strategy of using synthetic examples to train the stream clustering methods may contribute to such aggressiveness in exploring the minority class decision boundaries.

Figures 7(a) and 7(c) show that OOB and OnlineOversampling learnt the most compact minority class decision areas because they reuse the existing minority class examples for oversampling. Figure 7(d) shows that the minority class decision areas of OnlineUnderOverBagging were slightly larger than that of OOB and OnlineOversampling. Particularly, there were two green areas linked together. This may be the result of using oversampling and undersampling together, which managed to cover the true minority class clusters while preserving some aggressiveness from undersampling. In contrast, Figures 7(b) and 7(f) show that UOB and UOB_d learnt a single cluster to aggressively cover most minority class areas, considering the small majority class areas in between as part of the minority class. This is likely to cost some predictive performance in the majority class. Thus, we can see that UOB and UOB_d performed slightly worse than the other approaches before the concept drift (0-70k time steps, Figure 6). However, Figure 5 shows that UOB performed slightly better than SMOClust in the five-dimensional StaticIm1_Move7 stream, indicating that the aggressive nature of undersampling may be an advantage in learning the minority class when the feature space is sparse and presents very few minority class examples. When the feature space is more compact, the proposed strategy in SMOClust is more advantageous.

Considering OOB_d, $\text{OnlineUnderOverBagging}_\text{d}$, VFC-SMOTE, and SMOTE-OB, Figures 7(e), 7(h), 7(j), and 7(k) show that their learnt minority decision areas were very small which only covered a small proportion of the true minority class areas. In the case of VFC-SMOTE, it predicted every example as majority class at 70k time steps. As previously mentioned, their predictive performance fluctuated a lot throughout the stream. So, it can be deduced that they were greatly affected by false-positive drift detections.

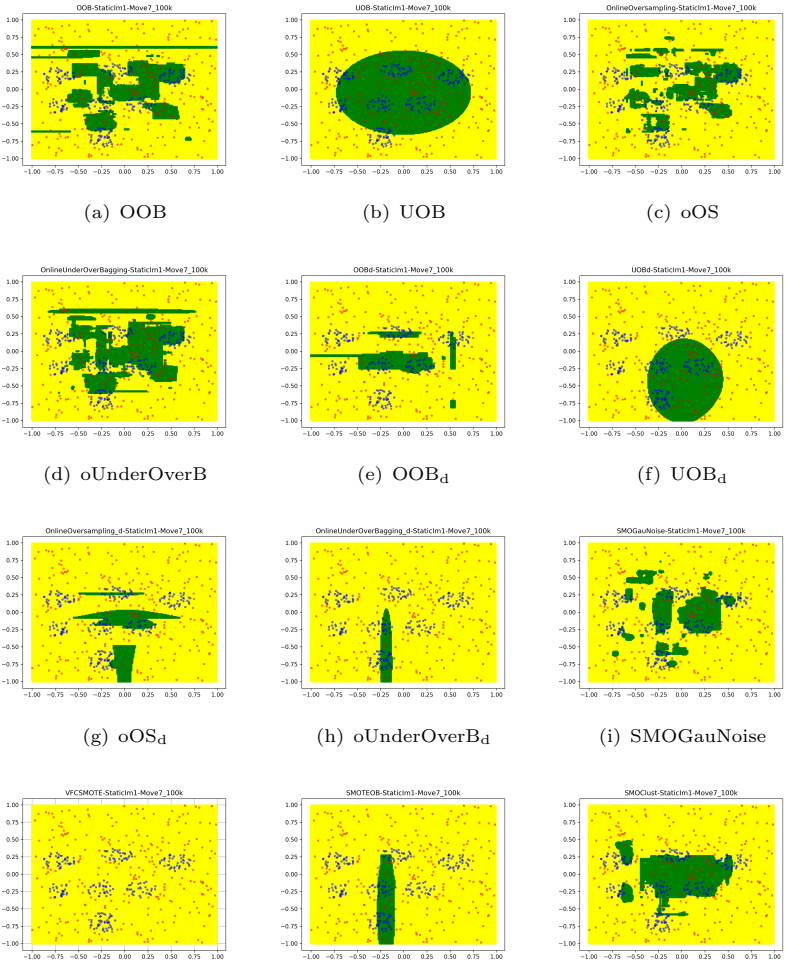
Over the next paragraphs, we compare the predictive performance and the decision boundaries of SMOClust against other approaches at the time steps right after concept drift (at 100k time steps) and at the end of the data stream (at 200k time steps), to understand how SMOClust handles concept drift of moving minority class sub-clusters when the data stream is severely class imbalanced.

Figure 6 shows that the predictive performance of SMOClust fluctuated during the drift (70k-100k time steps, Figure 6). Thus, it is likely that its base learner had been reset several times due to drift detection. Yet, it was the fastest approach to recovering predictive performance from the drift. Figure 8 presents the learnt decision boundaries right after the drift. It shows that SMOClust and SMOGauNoise made the best attempt in adapting the drift. They were able to cover most minority class sub-clusters at the new position, especially SMOClust. The potential reason is that, although the base learner of SMOClust is reset upon drift detection, the stream clustering methods are not reset as they are expected to be drift adaptable. Therefore, SMOClust is more robust to incremental and gradual drifts than SMOGauNoise, explaining its rapid predictive performance recovery from the drift.

On the other hand, Figures 8(a), 8(b), 8(c), and 8(d) show that the learnt minority class decision areas of OOB, UOB, OnlineOversampling and OnlineUnderOverBagging mainly retained at the pre-drift position because they are not concept drift adaptable. Their concept drift adaptable counterparts , VFC-SMOTE and SMOTe-OB did not handle the drift very well either. Figures 8(e), 8(f), 8(g), 8(h), 8(j), and 8(k) show that their learnt minority class decision areas only covered a few minority class sub-clusters at the post-drift position, which is likely because their base learners had been reset for several times caused by drift detection and they do not have any strategy to deal with incremental and gradual drifts. As the result, they struggled to recover their predictive performance from the drift, as shown in Figure 6.

Lastly, we compare the learnt decision areas of the approaches at the end of the two-dimensional StaticIm1_Move7 stream. Figure 9 shows that OOB_d, SMOGauNoise and SMOClust are the best approaches in converging to the post-drift position of minority class sub-clusters. In particular, a few green areas of SMOClust and SMOGauNoise were slightly less compact than OOB_d, showing that SMOClust and SMOGauNoise had slightly better generalisation than OOB_d.

Figures 9(a), 9(c), and 9(d) show that OOB, OnlineOversampling and OnlineUnderOverBagging managed to converge to the new concept after the drift. However, they also retained a small portion of green areas which corresponds to the pre-drift position of the minority class. This shows that OOB, OnlineOversampling and OnlineUnderOverBagging can adapt to concept drift involving minority class sub-cluster movement. However, they required a longer period to adapt as they were hindered by the knowledge acquired pre-drift. Meanwhile, Figures 9(e), 9(g), and 9(h) show that their concept drift adaptable counterparts adapted better, except $\text{OnlineUnderOverBagging}_\text{d}$. While resetting base learners helps to adapt to concept drift, $\text{OnlineUnderOverBagging}_\text{d}$ partly uses undersampling in its strategy to deal with class imbalance led to some over-generalisation between the learnt minority class areas. UOB and UOB_d use undersampling to deal with class imbalance, thus Figures 9(a) and 9(f) show that they had the greatest over generalisation due to the aggressive



(a) OOB (b) UOB (c) oOS

(d) oUnderOverB (e) OOB_d (f) UOB_d

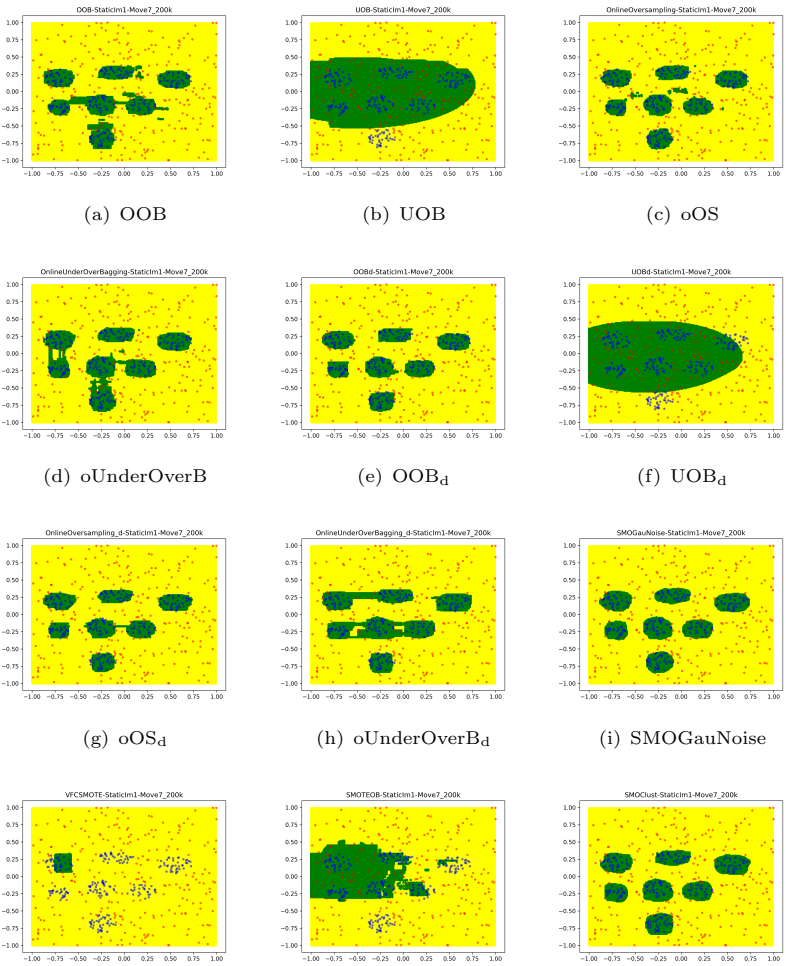
(g) oOS_d (h) $oUnderOverB_d$ (i) SMOGauNoise

(j) VFC-SMOTE (k) SMOTE-OB (l) SMOClust

Fig. 8: Decision Areas Against Class Balanced Test Set at 100k Time Steps (After Drift) of Two-Dimensional StaticIm1_Move7

nature of undersampling. VFC-SMOTE and SMOTE-OB continued to struggle, as shown in Figures 9(j) and 9(k), because of frequent false-positive drift detections.

Short Summary: Through the pre-drift analysis, the ability of SMOClust in handling stationary severely class imbalanced data streams presenting several minority class sub-clusters is validated. In particular, it shows that SMOClust was able to learn and explore the true decision boundaries despite the data stream presents very few minority class examples. The post-drift analysis shows that SMOClust was more robust in adapting incremental and gradual drift involving minority class



(a) OOB (b) UOB (c) oOS

(d) oUnderOverB (e) OOB_d (f) UOB_d

(g) oOS_d (h) $oUnderOverB_d$ (i) SMOGauNoise

(j) VFC-SMOTE (k) SMOTE-OB (l) SMOClust

Fig. 9: Decision Areas Against Class Balanced Test Set at 200k Time Steps (End of Stream) of Two-Dimensional StaticIm1_Move7

sub-clusters movement than existing approaches. Although most of the approaches converged to the new concept at the end of the data stream, SMOClust was the best and the fastest approach in recovering predictive performance from the drift. The inconsistent results between two and five-dimensional versions of this representative case indicate that SMOClust may be more advantageous in lower-dimensional data streams.

Table 9: 30 Runs Average G-Mean on Two-Dimensional Version of Representative Artificial Data Streams where SMOClust Performed Worse

Stream	OOB	UOB	oOS	oUnder-OverB	OOB_d	UOB_d
StaticIm10_Rare100	70.61%	63.65%	69.14%	68.19%	65.49%	68.17%
Stream	**oOS_d**	**oUnder-$OverB_d$**	**SMO-GauNoise**	**VFC-SMOTE**	**SMOTE-OB**	**SMOClust**
StaticIm10_Rare100	65.04%	64.98%	64.56%	54.64%	58.98%	70.32%

- Based on the average G-Mean, cells are highlighted in lime / light grey when SMOClust performed better than the corresponding approach and cells are highlighted in orange / dark grey cells when SMOClust performed worse than the corresponding approach. The colour intensity scales with the absolute difference of average G-Mean between the SMOClust and the approach of the column and the intensity reaches the maximum when such difference is $\geq 10\%$.

4.3.2 Cases where SMOClust performed worse

This section discusses the situations where SMOClust performed worse than other approaches, particularly in cases with concept drift leading to 100% rare minority examples. StaticIm10_Rare100 stream was chosen from Table 4 as the representative case to discuss the behaviour of SMOClust in detail. Following the method of analysis in Section 4.3.1, we also created a two-dimensional version of StaticIm10_Rare100 stream such that we can visualise and compare the learnt decision boundaries of the approaches to understand their behaviour.

Table 9 presents the approaches' thirty runs average G-Mean on the two-dimensional StaticIm10_Rare100 stream. It shows that SMOClust performed better than most other approaches. Figure 10, showing the G-Mean of the approaches in their median run[5] throughout the two-dimensional StaticIm10_Rare100 stream, also supports the results on Table 9. Note that, to improve readability, we have omitted the predictive performance of OOB_d, UOB_d, oOS_d, $oUnderOverB_d$, VFC-SMOTE and SMOTE-OB from Figure 10, similar to Figure 6, due to their values fluctuating significantly throughout the stream. For a comparison of SMOClust against these approaches, please refer to the supplementary document.

As these results are not consistent with the results of the five-dimensional StaticIm10_Rare100 stream, shown in Table 6 and 7, we preliminary checked if using a different set of random seeds or picking another case that involves drift leading to 100% rare minority class examples would yield results that are consistent with Table 6 and 7. Yet, it still shows that SMOClust performed similar to or better than other approaches in two-dimensional StaticIm10_Rare100 stream. Thus, in this analysis, we focus on why SMOClust can handle concept drift leading to 100% rare minority class examples than other approaches when the data stream has only two dimensions while attempting to deduce why it could not when the data stream has five dimensions.

Figures 11, 12 and 13 visualise the learnt decision areas of the approaches at the time steps right before and after concept drift (70k and 100k time steps)

[5]Median run refers to the run that leads to the median of predictive performances averaged across time steps.

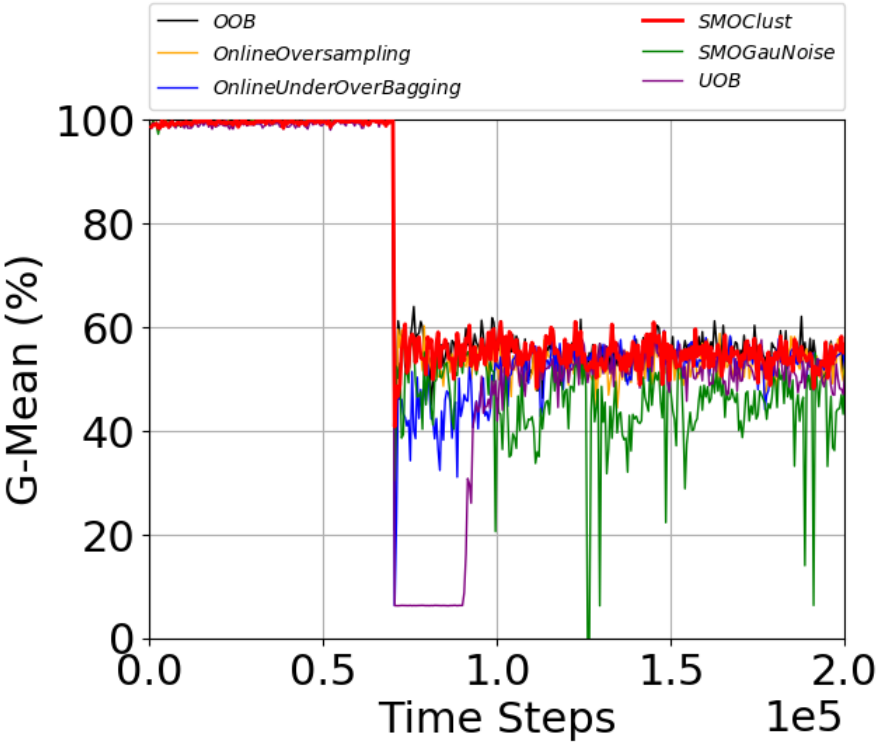
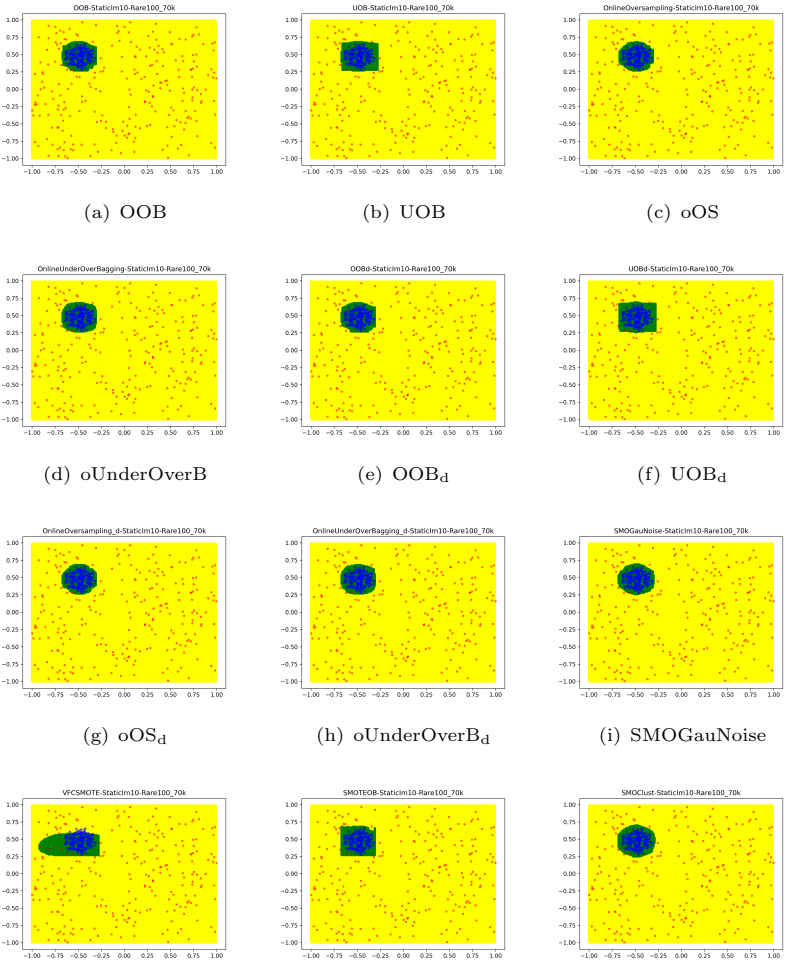


Fig. 10: Periodic Class Balanced Holdout Test G-Mean Against Time Steps in Two-Dimensional StaticIm10_Rare100

and at the end (200k time steps) of the two-dimensional StaticIm10_Rare100 stream respectively. The yellow and green regions represent their learnt decision areas of class 0 (majority class) and class 1 (minority class) respectively, while the red and blue dots are the class 0 (majority class) and class 1 (minority class) examples in the class balanced test set which corresponds to the time steps.

Figure 10 shows that all approaches performed very well during the pre-drift period (0-70k time steps). Figure 11 reveals that it is because they learnt the decision boundary of the pre-drift concept very well, as the minority class was just a single cluster. While most approaches learnt an oval shape decision boundary, UOB, UOB_d and SMOTE-OB learnt a rectangular shape, which could be due to the use of undersampling. VFC-SMOTE learnt a peculiar shape decision boundary which would cause more frequent false-positive drift detections. These may have been due to minority class examples generated by VFC-SMOTE with considerable amount of noise. Meanwhile, SMOTE-OB adopts the same strategy as VFC-SMOTE for generating synthetic minority class examples but simultaneously incorporating undersampling to address class imbalance. This integration of undersampling

might explain why SMOTE-OB more successfully circumvented the issue encountered by VFC-SMOTE.



(a) OOB (b) UOB (c) oOS

(d) oUnderOverB (e) OOB_d (f) UOB_d

(g) oOS_d (h) $oUnderOverB_d$ (i) SMOGauNoise

(j) VFC-SMOTE (k) SMOTE-OB (l) SMOClust

Fig. 11: Decision Areas Against Class Balanced Test Set at 70k Time Steps (Before Drift) of Two-Dimensional StaticIm10_Rare100

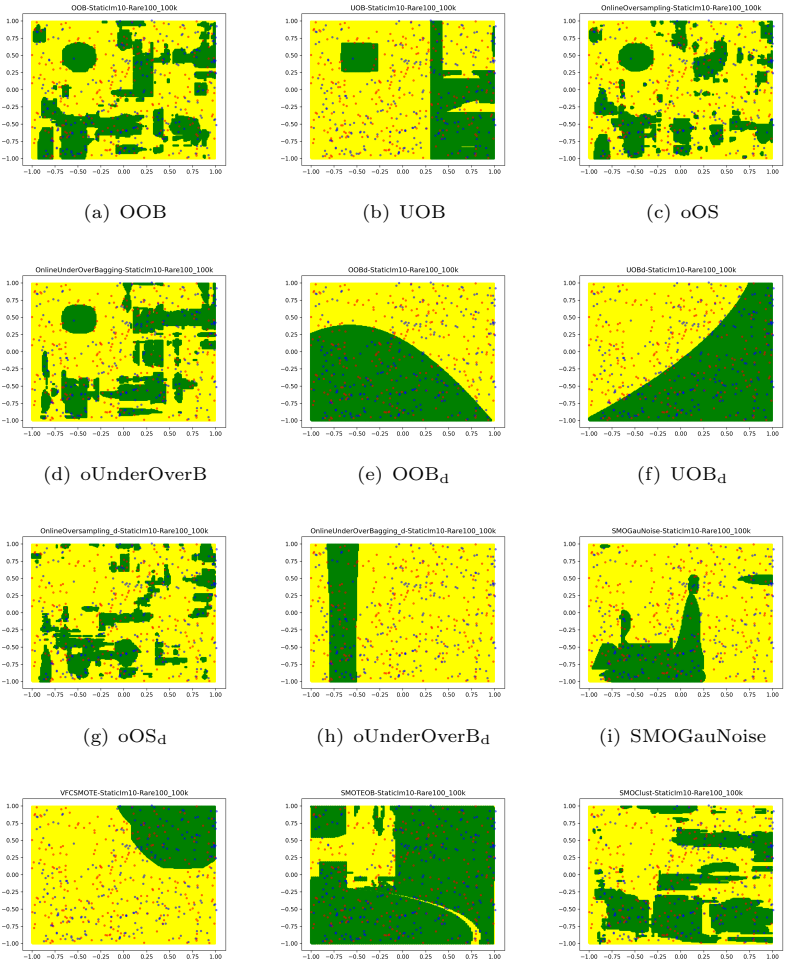
Figure 10 shows that the predictive performance of the approaches dropped to below 60% G-Mean and started to differ since the concept drift began (70k time steps). While most approaches' predictive performance fluctuated with large magnitude, SMOClust's predictive performance was relatively steady, bouncing between 50%-60% G-Mean. UOB performed poorly since the drift

began at 70k time steps until the drift was close to finishing at 100k time steps, indicating that undersampling struggled in dealing with this drift without the help of a concept drift detector.

Figure 12 presents the learnt decision boundaries of the approaches right after the drift (100k time steps). It shows that OOB, OnlineOversampling, OnlineUnderOverBagging, OnlineOversampling$_{\mathrm{d}}$, SMOGauNoise, SMOTE-OB and SMOClust learnt very complex decision areas, indicating that they made great efforts to learn all the areas that spawn rare minority class examples belonging to the post-drift concept. However, only approaches with a concept drift detector were able to forget the old area of the minority class at the top left corner. This shows that, although this drift was gradual, concept drift detection was important in helping the system to forget irrelevant past knowledge. In contrast, approaches without a drift detector retained the oval minority class cluster at the top left corner which belongs to the pre-drift concept. Most of them struggled to perform well since the drift started at 70k time steps, as shown in Figure 10. OOB was an exception in terms of predictive performance. However, the fact that it retained the knowledge about the pre-drift minority class areas makes it disadvantageous in dealing with other types of drift, as discussed in Section 4.3.1.

Comparing the learnt decision areas of SMOClust against other approaches with drift detector (OOB$_{\mathrm{d}}$, UOB$_{\mathrm{d}}$, OnlineOversampling$_{\mathrm{d}}$, OnlineUnderOverBagging$_{\mathrm{d}}$, VFC-SMOTE, SMOTE-OB and SMOGauNoise), it can be observed that the learnt minority class areas of SMOClust were complex and covered the feature space spawning minority class examples the most. While OnlineOversampling$_{\mathrm{d}}$'s, SMOGauNoise's and SMOTE-OB's were also complex (see Figures 12(g), 12(i) and 12(k)), they either did not cover the feature space spawning minority class examples as much as SMOClust's did or exhibited over-generalisation. The fact that OnlineOversampling$_{\mathrm{d}}$ only reuses the recently seen minority class example for oversampling likely leads to overfitting to such most recent area. SMOGauNoise also has a strategy to explore the decision boundaries of the minority class, but such strategy only explores the area around the recently seen minority class example. This could be disadvantageous when false-positive drift detections were triggered, resetting the base learner. SMOTE-OB's over-generalisation could be explained by the use of undersampling and noisy minority class examples generated. SMOClust, on the other hand, does not have this disadvantage because the stream clustering methods are not reset upon drift detection. This makes it more robust to false-positive drift detections than other approaches. As the drift was gradual, OOB$_{\mathrm{d}}$, UOB$_{\mathrm{d}}$ and OnlineUnderOverBagging$_{\mathrm{d}}$ likely also suffered from multiple drift detection, as Figures 12(e), 12(f) and 12(h) show that the learnt a simple decision boundary right after the drift.

Figure 13 presents the learnt decision boundaries of the approaches at the end of the two-dimensional StaticIm10_Rare100 stream (at 200k time steps). While most approaches continued to further improve their learnt decision boundaries since the drift had finished, Figures 13(h) and 13(l) show that



(a) OOB (b) UOB (c) oOS

(d) oUnderOverB (e) OOB_d (f) UOB_d

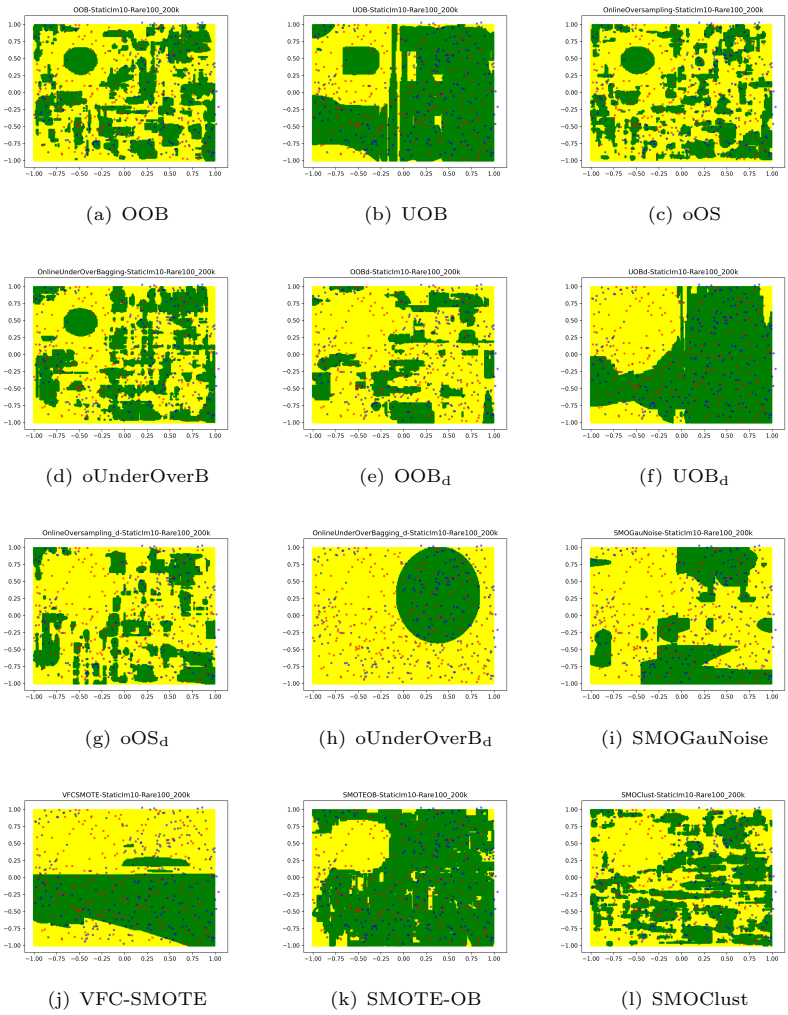
(g) oOS_d (h) $oUnderOverB_d$ (i) SMOGauNoise

(j) VFC-SMOTE (k) SMOTE-OB (l) SMOClust

Fig. 12: Decision Areas Against Class Balanced Test Set at 100k Time Steps (After Drift) of Two-Dimensional StaticIm10_Rare100

$OnlineUnderOverBagging_d$ and SMOGauNoise did not improve as much as other approaches, meaning that they suffered from false-positive drift detections during the post-drift period. Besides, UOB, UOB_d, and SMOTE-OB exhibited an extensive and predominantly continuous decision area for the minority class, demonstrating the aggressiveness of undersampling. However, in the case of SMOTE-OB, the approach's synthetic minority class generation strategy exacerbates this aggressiveness.

From this analysis, it has been shown that SMOClust managed to forget the pre-drift concept and adapt to drift leading to 100% rare minority



(a) OOB (b) UOB (c) oOS

(d) oUnderOverB (e) OOB_d (f) UOB_d

(g) oOS_d (h) $oUnderOverB_d$ (i) SMOGauNoise

(j) VFC-SMOTE (k) SMOTE-OB (l) SMOClust

Fig. 13: Decision Areas Against Class Balanced Test Set at 200k Time Steps (End of Stream) of Two-Dimensional StaticIm10_Rare100

class examples and more robust to false-positive drift detections than other approaches in the two-dimensional StaticIm10_Rare100 stream. However, the experiment with the five-dimensional StaticIm10_Rare100 stream presents different results (Figure 4). It shows that SMOClust only performed better than $OnlineOversampling_d$ but worse or similar to most other approaches. One potential reason is the fact that two-dimensional space is more compact than five-dimensional space, the rare minority class examples have a lot less space to randomly spawn, which means they are likely to spawn at the locations that had already been learnt and covered by SMOClust using micro-clusters.

Therefore, SMOClust can predict their class label correctly. However, five-dimensional space is sparser than two-dimensional space, meaning that new rare minority class examples are less likely to spawn at previous locations. Therefore, SMOClust struggled to make correct predictions to new rare minority class examples. Another potential reason is that the stream clustering method may be less effective in data streams with more dimensions. For example, it may create some minority class micro-clusters that overlap with the majority class region because of the sparsity of the feature space. Therefore, the aforementioned advantage of SMOClust in dealing with drift could not be manifested. Anyhow, future work is needed to further confirm whether SMOClust tends to perform better in data streams with fewer dimensions.

Short Summary: This analysis shows that SMOClust managed to adapt to concept drift leading to 100% rare minority class examples and was robust to multiple drift detection during gradual drift as well as false-positive drift detections when the data stream has only two dimensions. However, the experiments with the corresponding five-dimensional stream present a different set of results, as the stream clustering methods used by SMOClust might not perform well when the data stream has more dimensions.

4.3.3 Results with Two-Dimensional Artificial Data Streams

To investigate whether SMOClust performs better in lower-dimensional data streams, we performed additional experiments on the same artificial data streams presented in Section 4.1, but with only two input features. We also created a randomised two-dimensional data stream for the purpose of hyper-parameter tuning, following the procedure described in Section 4.3.

Figure 14 presents the difference in average G-Mean (based on thirty runs) between compared approaches and SMOClust on two-dimensional artificial data streams in the form of a heat-map. Green cells indicate results favourable to SMOClust, whereas red cells indicate results favourable to the compared approach. For a comprehensive table of the predictive performance of the approaches, please refer to the supplementary document. Compared to Figure 4, there are fewer red cells in this figure, indicating that SMOClust generally performed better in the lower-dimensional version of the same set of data streams. In particular, the sections of the heat-map corresponding to StaticIm30 and StaticIm10 data streams, which were mostly reddish in Figure 4, are mostly greenish in Figure 14.

Figure 14 also confirms the trend shown in Figure 4, showing that SMOClust tends to outperform other approaches in severely class-imbalanced data streams. To further validate this trend in lower-dimensional data streams, we performed further experiments on the same set of single factor drift artificial data streams, but with enforced extremely severe class imbalance ratios (minority class ratio 0.3% to 5%, as summarised in Table 3). The results are presented in Figure 15 in the form of a heat-map, using the same colour scheme as Figure 14. Similarly, please refer to the supplementary document for a comprehensive table of the predictive performance of the approaches.

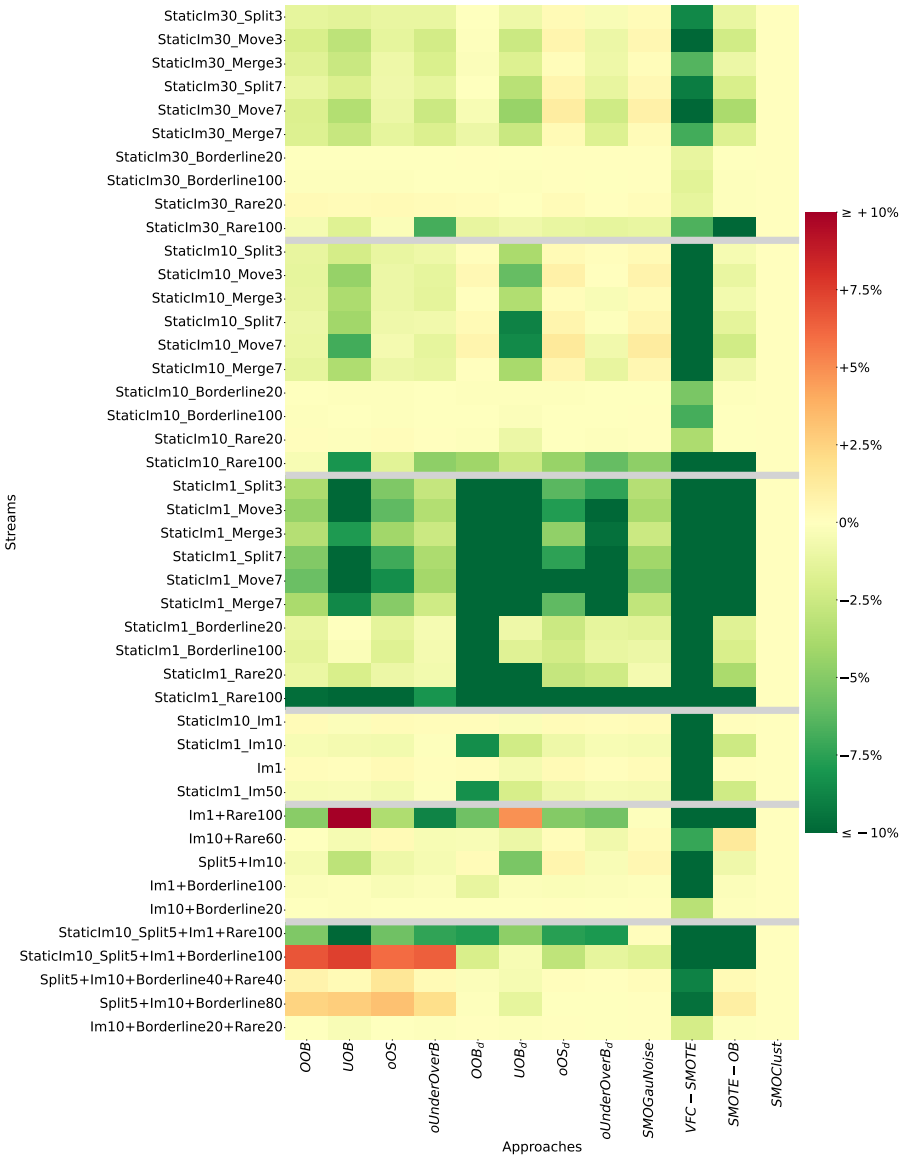


Fig. 14: Difference in Average G-Mean Against SMOClust on Two-Dimensional Class Imbalanced Artificial Data Streams Based on 30 Runs (Green cells indicate SMOClust performed better; Red cells indicate SMOClust performed worse; Grey horizontal lines separate different groups of data streams, i.e., StaticIm{30/10/1}, Imbalance Ratio Drift, Double Factor, and Complex Factor)

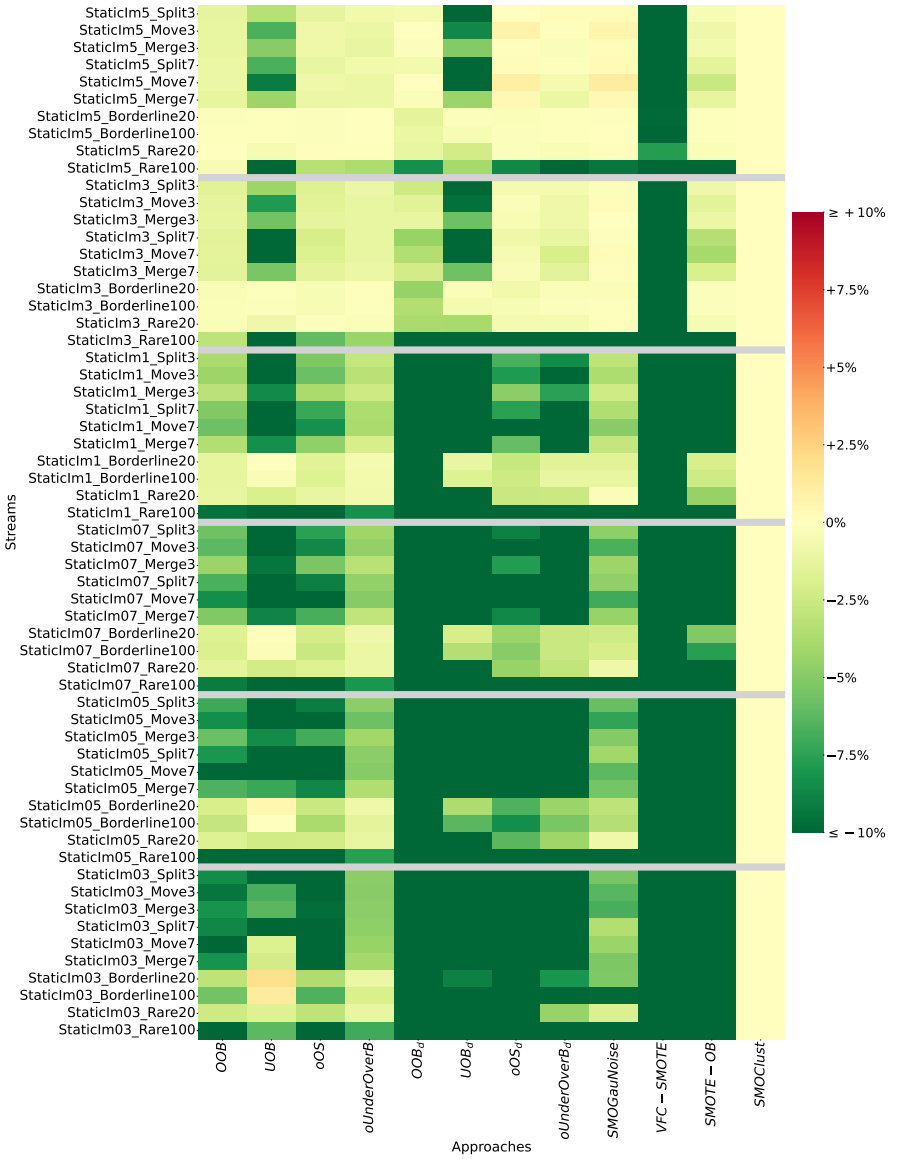


Fig. 15: Difference in Average G-Mean Against SMOClust on Two-Dimensional Severely Class Imbalanced Artificial Data Streams Based on 30 Runs (Green cells indicate SMOClust performed better; Red cells indicate SMOClust performed worse; Grey horizontal lines separate different groups of data streams, i.e., StaticIm{5/3/1/07/05/03}

Figure 15 presents more solid green cells than Figure 14, indicating that SMOClust performed better than other approaches in extremely severe class-imbalanced data streams, even in the lower-dimensional case. Additionally, the fact that Figure 15 has more green cells than Figure 5 supports the conclusion that SMOClust tends to perform better in lower-dimensional data streams.

4.4 Results with Real-world Data Streams

This section presents the analysis done to compare the predictive performance of SMOClust against nine existing approaches in real-world data streams. Experiments with real-world data streams allow us to obtain a general idea of SMOClust's predictive performance in practical applications, where the class imbalance ratio, the position and the type of the concept drifts are unknown. Table 10 presents the Friedman rankings of approaches' G-Mean on real-world data streams group by factors.

Table 10: Statistical (Friedman) Ranking of prequential G-Mean on Real-World Streams Grouped by Factors

Groups	OOB	UOB	oOS	oUnder-OverB	OOB_d	UOB_d	oOS_d	oUnder-$OverB_d$	SMO-Gau-Noise	VFC-SMO-TE	SMO-TE-OB	SMO-Clust
Luxum-bourg	5.43	8.83	2.03	7.33	5.43	10.27	2.03	7.33	2.97	11.87	6.23	8.23
NOAA	1.80	5.80	5.00	1.93	6.10	8.27	10.73	6.97	12.00	9.67	3.13	6.60
Ozone	3.27	2.68	9.25	3.93	6.10	2.82	9.02	7.53	10.70	12.00	2.30	8.40
PAKDD-2009	5.90	2.43	1.20	6.43	8.00	6.97	2.37	7.67	10.70	10.30	4.03	12.00
Covtype	2.79	6.60	9.34	3.33	5.44	5.23	11.56	6.40	9.95	9.89	2.81	4.66
INSECTS	5.59	9.39	7.26	6.28	1.23	8.58	3.75	2.54	5.91	11.29	4.77	11.41
Amazon	1.93	8.43	4.23	11.93	4.57	3.43	6.50	6.93	7.93	10.47	1.07	10.57
Twitter	1.90	5.37	8.43	3.87	2.57	4.07	8.50	4.97	10.53	10.47	5.33	12.00
All	3.7593	7.0417	7.325	5.0074	4.2778	6.4083	7.712	5.4963	8.5574	10.5778	3.6444	8.1926

- The p-values of Friedman tests are all $\leq$2.2E-16.
- Highlighted ranks denote significant superior performance.
- Underlined ranks denote the corresponding approach's performance have no statistical significance with SMOClust.

Table 10 shows that the overall top-ranked approaches on real-world data streams are OOB, OOB_d and SMOTE-OB whereas SMOClust usually achieved low rankings. SMOClust only achieved a relatively better ranking in Covtype streams than in other streams. Considering all real-world data streams, SMOClust performed similarly to $OnlineOversampling_d$ and SMO-GauNoise. Following the analysis method in Section 4.3, we also compared the thirty runs average prequential G-Mean of the approaches on each real-world data stream in Figure 16 to further evaluate the predictive performance of SMOClust in real-world data streams.

Figure 16 shows that SMOClust usually performed similar or better than other approaches in NOAA and Covtype streams while it performed worse than other approaches in Ozone, PAKDD2009, INSECTS, Amazon, and Twitter streams. Recalling the discussion in Section 4.1 on estimated characteristics

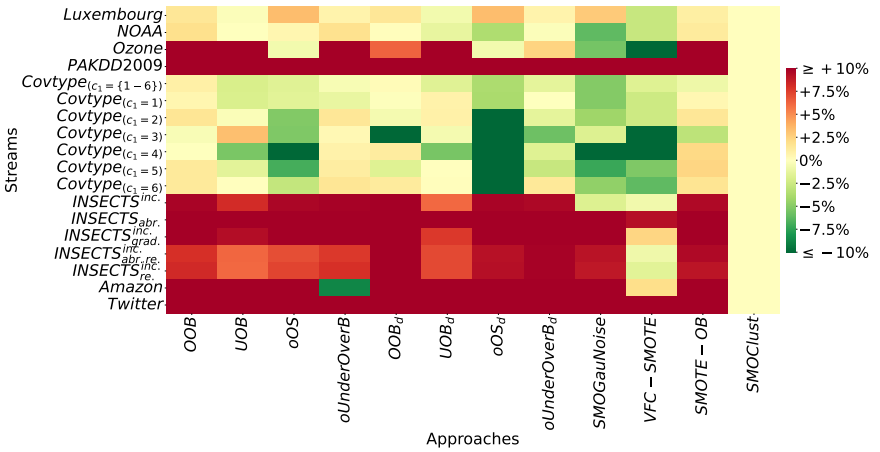


Fig. 16: Difference in Average G-Mean Against SMOClust on Real-World Data Streams Based on 30 Runs (Green cells indicate SMOClust performed better; Red cells indicate SMOClust performed worse)

of real-world streams, NOAA and Covtype streams mainly consist of safe and borderline minority class examples with different movements of minority class clusters and the minority class ratios throughout Covtype streams are usually very low (except Covtype$_{(c_1=\{1\text{-}6\})}$ and Covtype$_{(c_1=1)}$). As discussed in Section 4.3, these are the characteristics of a data stream that SMOClust is likely to perform similar or better than other approaches, especially when the class imbalance ratio is severe, such as Covtype$_{(c_1=3)}$ stream. Thus, we can see from Figure 16 that the rows of NOAA and Covtype streams mainly consist of saturated green cells and pale red cells.

On the other hand, Table 5 shows that Ozone, PAKDD2009, INSECTS, Amazon, and Twitter streams consist of large proportions of rare and outlier minority class examples. Based on the discussion in Section 4.3.2, SMOClust could not handle rare and outlier minority class examples very well, except when the dimensionality of the data stream was low or compact. Thus, it is not surprising to see a lot of red cells on these data streams.

To summarise the result of experiments with real-world data streams, the advantage of the proposed synthetic minority class oversampling strategy in SMOClust is manifested in severely class imbalanced data streams with high proportions of safe and borderline minority class examples with concept drifts of different movements of minority class sub-clusters. On the downside, SMOClust could not handle rare and outlier minority class examples very well. These findings are consistent with the result of experiments with artificial data streams.

5 Conclusion

The main contribution of this work is the proposed stream clustering based synthetic minority oversampling approach, called SMOClust (RQ1). This method helps the learning system to strategically explore different decision areas of the minority class and to be robust to false-positive drift detections (RQ1). To evaluate the predictive performance and the characteristics of SMOClust, experiments with artificial data streams concerning different types of concept drift difficulties were performed. The results show that SMOClust performed particularly well in severely class imbalanced data streams with high proportions of safe and borderline minority class examples (RQ2). It also handles concept drifts of different movements of minority class clusters better than other existing approaches (RQ2). However, when the data stream presents high proportions of rare and outlier minority class examples, SMOClust becomes disadvantageous (RQ3).

To further understand the reason behind the experiment results on artificial data streams, additional experiments with representative two-dimensional artificial data streams were performed. However, it shows that SMOClust managed to handle rare minority class examples better than other approaches in these two-dimensional cases. This indicates that the reason why SMOClust could not handle rare cases very well on the corresponding five-dimensional stream was likely because of the stream clustering methods did not perform well in higher-dimensional space. In other words, SMOClust may be more advantageous when the dimensionality of the data stream is not high.

Lastly, we validated the performance of SMOClust on different real-world data streams. To facilitate the analysis of the experiment results of this part of the study, we estimated the characteristics of the real-world data streams, following the procedure adopted by [16]. Based on the estimated characteristics and the experiment results, we concluded that the SMOClust behaved similarly to the experiments with artificial data streams (RQ3).

As for future work, an investigation of new strategies to better handle large proportions of rare and outlier minority class examples is one potential direction. For example, strategies to generate synthetic minority examples for oversampling in a more diverse manner without introducing a significant amount of noise or creating artificial concept drifts could be proposed. Additionally, extending the idea of SMOClust to deal with multi-class classification tasks could also be an area to investigate in the future. Furthermore, the proposed synthetic minority oversampling strategy in this work could be adapted for use with other complex data stream learning systems easily as it is a drift adaptable data-level method to address class imbalance in data stream learning. For example, it could be incorporated into an explicit drift handling approach which exploits relevant past knowledge to handle concept drifts [41, 42] or an ensemble approach which evolves themselves to adapt to concept drifts [60, 61]. Apart from these, a comprehensive study to compare SMOClust against more approaches for learning drifting class imbalanced data streams

(e.g., CSARF [24], ROSE [25] etc.) and with more data sets could also be a potential future work.

Declarations

- **Funding:** This work was partly supported by EPSRC Grant No. EP/R006660/2. This work was conducted while Chun Wai Chiu was a PhD student at the School of Computer Science, University of Birmingham, UK, under a School Scholarship provided in support of this grant.
- **Conflict of interest:** The authors declare that they have no conflict of interest.
- **Ethics approval:** Not Applicable.
- **Consent to participate:** Not Applicable.
- **Consent for publication:** Not Applicable.
- **Availability of data and materials:** The data sets used in the experiments are available at: https://github.com/michaelchiucw/SMOClust
- **Code availability:** The implementation of the proposed approach is available at: https://github.com/michaelchiucw/SMOClust
- **Authors' contributions:** All authors contributed to the study conception, design, experiment results analysis, drafting and editing the manuscript. Chun Wai Chiu implemented the proposed approach and performed the experiments.

References

[1] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., Bontempi, G.: Credit card fraud detection: A realistic modelling and a novel learning strategy. IEEE Transactions on Neural Networks and Learning Systems, 1–14 (2017)

[2] Tabassum, S., Minku, L.L., Feng, D., Cabral, G., Song, L.: An investigation of cross-project learning in online just-in-time software defect prediction. In: ICSE, pp. 554–565 (2020)

[3] Delany, S.J., Cunningham, P., Tsymbal, A., Coyle, L.: A case-based technique for tracking concept drift in spam filtering. Knowledge-Based Systems **18**, 187–195 (2005)

[4] Ditzler, G., Roveri, M., Alippi, C., Polikar, R.: Learning in nonstationary environments: A survey. IEEE Computational Intelligence Magazine **10**, 12–25 (2015)

[5] Žliobaitė, I.: Learning under concept drift: an overview. CoRR **abs/1010.4784** (2010)

[6] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Survey **46**(4) (2014). https://doi.org/10.1145/2523813

[7] Wang, S., Minku, L.L., Yao, X.: A systematic study of online class imbalance learning with concept drift. IEEE Transactions on Neural Networks and Learning Systems **29**(10), 4802–4821 (2018). https://doi.org/10.1109/TNNLS.2017.2771290

[8] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research **16**(1), 321–357 (2002)

[9] Han, H., Wang, W.-Y., Mao, B.-H.: Borderline-smote: A new over-sampling method in imbalanced data sets learning, vol. 3644, pp. 878–887 (2005). https://doi.org/10.1007/11538059_91

[10] Lee, H., Kim, J., Kim, S.: Gaussian-based smote algorithm for solving skewed class distributions, vol. 17, pp. 229–234 (2017). https://doi.org/10.5391/IJFIS.2017.17.4.229

[11] Wang, B., Pineau, J.: Online bagging and boosting for imbalanced data streams. IEEE Transactions on Knowledge and Data Engineering **28**(12), 3353–3366 (2016). https://doi.org/10.1109/TKDE.2016.2609424

[12] Bernardo, A., Gomes, H.M., Montiel, J., Pfahringer, B., Bifet, A., Valle, E.D.: C-smote: Continuous synthetic minority oversampling for evolving data streams, pp. 483–492 (2020). https://doi.org/10.1109/BigData50022.2020.9377768

[13] Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J., Woźniak, M.: Ensemble learning for data stream analysis: A survey. Information Fusion **37**, 132–156 (2017)

[14] Hoens, T.R., Chawla, N.V.: Learning in non-stationary environments with class imbalance. KDD '12, pp. 168–176. Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2339530.2339558. https://doi.org/10.1145/2339530.2339558

[15] Bernardo, A., Della Valle, E.: Vfc-smote: very fast continuous synthetic minority oversampling for evolving data streams. Data Mining and Knowledge Discovery **35** (2021). https://doi.org/10.1007/s10618-021-00786-0

[16] Brzezinski, D., Minku, L.L., Pewinski, T., Stefanowski, J., Szumaczuk, A.: The impact of data difficulty factors on classification of imbalanced and concept drifting data streams. Knowl. Inf. Syst. **63**, 1429–1469 (2021)

[17] Napierala, K., Stefanowski, J.: Types of minority class examples and their influence on learning classifiers from imbalanced data. Journal of Intelligent Information Systems **46**(3), 563–597 (2015). https://doi.org/10.1007/s10844-015-0368-1

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

[19] Lu, Y., Cheung, Y.M., Tang, Y.Y.: Dynamic weighted majority for incremental learning of imbalanced data streams with concept drift. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pp. 2393–2399 (2017). https://doi.org/10.24963/ijcai.2017/333. https://doi.org/10.24963/ijcai.2017/333

[20] Wang, S., Minku, L.L., Yao, X.: Resampling-based ensemble methods for online class imbalance learning. IEEE Transactions on Knowledge and Data Engineering **27**(5), 1356–1368 (2015). https://doi.org/10.1109/TKDE.2014.2345380

[21] Mirza, B., Lin, Z., Liu, N.: Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. Neurocomputing **149**, 316–329 (2015). https://doi.org/10.1016/j.neucom.2014.03.075. Advances in neural networks Advances in Extreme Learning Machines

[22] Bernardo, A., Gomes, H.M., Montiel, J., Pfahringer, B., Bifet, A., Valle, E.D.: C-smote: Continuous synthetic minority oversampling for evolving data streams. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 483–492 (2020). https://doi.org/10.1109/BigData50022.2020.9377768

[23] Bernardo, A., Valle, E.D.: Smote-ob: Combining smote and online bagging for continuous rebalancing of evolving data streams. In: 2021 IEEE International Conference on Big Data (Big Data), pp. 5033–5042 (2021). https://doi.org/10.1109/BigData52589.2021.9671609

[24] Loezer, Lucas and Enembreck, Fabrício and Barddal, Jean Paul and de Souza Britto, Alceu: Cost-sensitive learning for imbalanced data streams. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing. SAC '20, pp. 498–504. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3341105.3373949. https://doi.org/10.1145/3341105.3373949

[25] Cano, Alberto and Krawczyk, Bartosz: Rose: robust online self-adjusting ensemble for continual learning on imbalanced drifting data streams. Machine Learning **111**(7), 2561–2599 (2022). https://doi.org/10.1007/

s10994-022-06168-x

[26] He, H., Bai, Y., Garcia, E.A., Li, S.: Adasyn: Adaptive synthetic sampling approach for imbalanced learning, pp. 1322–1328 (2008). https://doi.org/10.1109/IJCNN.2008.4633969

[27] Bunkhumpornpat, C., Sinapiromsaran, K., Lursinsap, C.: Dbsmote: Density-based synthetic minority over-sampling technique. Applied Intelligence - APIN **36** (2011). https://doi.org/10.1007/s10489-011-0287-y

[28] Bellinger, C., Sharma, S., Japkowicz, N., Zaïane, O.: Framework for extreme imbalance classification: Swim—sampling with the majority class. Knowledge and Information Systems **62** (2020). https://doi.org/10.1007/s10115-019-01380-z

[29] Song, L., Minku, L.L., Yao, X.: A novel automated approach for software effort estimation based on data augmentation. ESEC/FSE 2018, pp. 468–479. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3236024.3236052. https://doi.org/10.1145/3236024.3236052

[30] Aguiar, G., Krawczyk, B., Cano, A.: A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework (2022)

[31] Wang, S., Minku, L.L., Ghezzi, D., Caltabiano, D., Tino, P., Yao, X.: Concept drift detection for online class imbalance learning, pp. 1–10 (2013). https://doi.org/10.1109/IJCNN.2013.6706768

[32] Wang, H., Abraham, Z.: Concept drift detection for streaming data, pp. 1–9 (2015). https://doi.org/10.1109/IJCNN.2015.7280398

[33] Brzezinski, D., Stefanowski, J.: Prequential auc for classifier evaluation and drift detection in evolving data streams, vol. 8983 (2014). https://doi.org/10.1007/978-3-319-17876-9_6

[34] Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing, vol. 7 (2007). https://doi.org/10.1137/1.9781611972771.42

[35] Gomes, H.M., Bifet, A., Read, J., Barddal, J.P., Enembreck, F., Pfahringer, B., Holmes, G., Abdessalem, T.: Adaptive random forests for evolving data stream classification. Machine Learning **106**, 1–27 (2017). https://doi.org/10.1007/s10994-017-5642-8

[36] Domingos, P., Hulten, G.: Mining high-speed data streams. KDD '00, pp. 71–80. Association for Computing Machinery, New York, NY, USA (2000). https://doi.org/10.1145/347090.347107.

https://doi.org/10.1145/347090.347107

[37] Oza, N.C.: Online bagging and boosting, vol. 3, pp. 2340–23453 (2005). https://doi.org/10.1109/ICSMC.2005.1571498

[38] Elwell, R., Polikar, R.: Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks **22**(10), 1517–1531 (2011). https://doi.org/10.1109/TNN.2011.2160459

[39] Wang, S., Yao, X.: Diversity analysis on imbalanced data sets by using ensemble models, pp. 324–331 (2009). https://doi.org/10.1109/CIDM.2009.4938667

[40] Hoens, T.R., Chawla, N.V., Polikar, R.: Heuristic updatable weighted random subspaces for non-stationary environments, pp. 241–250 (2011). https://doi.org/10.1109/ICDM.2011.75

[41] Chiu, C.W., Minku, L.L.: Diversity-based pool of models for dealing with recurring concepts. 2018 International Joint Conference on Neural Networks (IJCNN), 1–8 (2018). https://doi.org/10.1109/IJCNN.2018.8489190

[42] Chiu, C.W., Minku, L.L.: A diversity framework for dealing with multiple types of concept drift based on clustering in the model space. IEEE Transactions on Neural Networks and Learning Systems **33**(3), 1299–1309 (2022). https://doi.org/10.1109/TNNLS.2020.3041684

[43] Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29, 81–92 (2003)

[44] Ackermann, M.R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., Sohler, C.: Streamkm++: A clustering algorithm for data streams. ACM J. Exp. Algorithmics **17** (2012). https://doi.org/10.1145/2133803.2184450

[45] Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise, vol. 2006 (2006). https://doi.org/10.1137/1.9781611972764.29

[46] Kranen, P., Assent, I., Baldauf, C., Seidl, T.: The clustree: indexing micro-clusters for anytime stream mining. Knowl. Inf. Syst. **29**(2), 249–272 (2011)

[47] Moulton, R.H., Viktor, H.L., Japkowicz, N., Gama, J.: Clustering in the presence of concept drift. In: ECML/PKDD (2018)

[48] Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection, vol. 8, pp. 286–295 (2004). https://doi.org/10.1007/978-3-540-28645-5_29

[49] Minku, L.L., Yao, X.: Ddd: A new ensemble approach for dealing with concept drift. IEEE Transactions on Knowledge and Data Engineering **24**, 619–633 (2012)

[50] Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: Massive online analysis. Journal of Machine Learning Research **11**(52), 1601–1604 (2010)

[51] Muller, M.E.: A note on a method for generating points uniformly on n-dimensional spheres. Communications of the ACM **2**(4), 19–20 (1959). https://doi.org/10.1145/377939.377946

[52] Žliobaitė, I.: Combining similarity in time and space for training set formation under concept drift. Intelligent Data Analysis **15**, 589–611 (2011). https://doi.org/10.3233/IDA-2011-0484

[53] Zhang, K., Fan, W., Yuan, X., Davidson, I., Li, X.: Forecasting skewed biased stochastic ozone days: Analyses and solutions, vol. 14, pp. 753–764 (2006). https://doi.org/10.1007/s10115-007-0095-1

[54] Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.): Advances in Knowledge Discovery and Data Mining, 13th Pacific-Asia Conference, PAKDD 2009. Lecture Notes in Computer Science, vol. 5476. Springer, Bangkok (2009). https://doi.org/10.1007/978-3-642-01307-2

[55] Blackard, J.A., Dean, D.J.: Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. Computers and Electronics in Agriculture **24**, 131–151 (1999). https://doi.org/10.1016/S0168-1699(99)00046-0

[56] Souza, V.M.A., dos Reis, D.M., Maletzke, A.G., Batista, G.E.A.P.A.: Challenges in benchmarking stream learning algorithms with real-world data. Data Mining and Knowledge Discovery **34**, 1805–1858 (2020). https://doi.org/10.1007/s10618-020-00698-5

[57] Blitzer, J., Dredze, M., Pereira, F.: Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 440–447. Association for Computational Linguistics, Prague, Czech Republic (2007). https://aclanthology.org/P07-1056

[58] Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., Stoyanov, V.: SemEval-2016 task 4: Sentiment analysis in Twitter. In: Proceedings of the 10th International Workshop on Semantic Evaluation

(SemEval-2016), pp. 1–18. Association for Computational Linguistics, San Diego, California (2016). https://doi.org/10.18653/v1/S16-1001. https://aclanthology.org/S16-1001

[59] Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science (New York, N.Y.) **315**, 972–6 (2007). https://doi.org/10.1126/science.1136800

[60] Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: A new ensemble method for tracking concept drift. IEEE International Conference on Data Mining, 123–130 (2003)

[61] D. Brzezinski and J. Stefanowski: Combining Block-based and Online Methods in Learning Ensembles from Concept Drifting Data Streams. Information Sciences **265**, 50–67 (2014)