Clustering in the Presence of Concept Drift

Richard Hugh Moulton[1](✉), Herna L. Viktor[1], Nathalie Japkowicz[2], and João Gama[3]

[1] School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON, Canada
{rmoul026,hviktor}@uottawa.ca

[2] Department of Computer Science, American University, Washington DC, USA
nathalie.japkowicz@american.edu

[3] Faculty of Economics, University of Porto, Porto, Portugal
jgama@fep.up.pt

Abstract. Clustering naturally addresses many of the challenges of data streams and many data stream clustering algorithms (DSCAs) have been proposed. The literature does not, however, provide quantitative descriptions of how these algorithms behave in different circumstances. In this paper we study how the clusterings produced by different DSCAs change, relative to the ground truth, as quantitatively different types of concept drift are encountered. This paper makes two contributions to the literature. First, we propose a method for generating real-valued data streams with precise quantitative concept drift. Second, we conduct an experimental study to provide quantitative analyses of DSCA performance with synthetic real-valued data streams and show how to apply this knowledge to real world data streams. We find that large magnitude and short duration concept drifts are most challenging and that DSCAs with partitioning-based offline clustering methods are generally more robust than those with density-based offline clustering methods. Our results further indicate that increasing the number of classes present in a stream is a more challenging environment than decreasing the number of classes. Code related to this paper is available at: https://doi.org/10.5281/zenodo.1168699, https://doi.org/10.5281/zenodo.1216189, https://doi.org/10.5281/zenodo.1213802, https://doi.org/10.5281/zenodo.1304380.

Keywords: Data streams · Clustering · Concept drift

The authors acknowledge that research at the University of Ottawa is conducted on traditional unceded Algonquin territory. This research was supported by the Natural Sciences and Engineering Research Council of Canada and the Province of Ontario. J. Gama is partially funded by the ERDF through the COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through the FCT as part of project UID/EEA/50014/2013.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-10925-7_21) contains supplementary material, which is available to authorized users.

1 Introduction

Data streams are challenging learning environments: their size is unbounded [3], the probabilities underlying the data stream can change [10,12,21] and labelled data is not readily available [20,22]. Clustering addresses these challenges as a means of summarization [17] and as an unsupervised learning technique. Lacking in the literature, however, is a discussion of how data stream clustering algorithms (DSCAs) are expected to perform. Since clustering is useful in dealing with data streams, understanding DSCA behaviour will help develop effective machine learning techniques for data streams. We use Webb et al.'s framework for quantitatively describing concept drift [23] to analyse DSCA performance.

We make two contributions in this paper. First, we propose a method for generating real-valued data streams with precise quantitative concept drift. This method uses mixture models and the Hellinger distance between concepts to mathematically model data streams. Second, we conduct quantitative analyses of DSCAs in experimental settings to determine the effect that different concept drifts have on the clusterings produced by different DSCAs. We also demonstrate how to use these findings to guide the selection of a DSCA in real-world applications. The research question we address is "how do the clusterings produced by different DSCAs change, relative to the ground truth, as quantitatively different types of concept drift are encountered?" Of particular interest is whether different DSCAs react differently in the presence of concept drift.

In the remainder of this paper we review the literature concerning concept drift and clustering in data streams (Sect. 2), describe the Mixture Model Drift Generator (Sect. 3), lay out our experimental framework (Sect. 4), present our results (Sect. 5) and identify potential future research in our conclusion.

2 Literature Review

In this section we describe the quantitative models used to formalize the data stream environment. We also discuss the types of concept drift identified in the literature and how they can be described using mathematical formalisms as well. Finally, we review the data stream clustering task and survey algorithms proposed for this purpose.

2.1 Concept Drift in Data Streams

Webb et al. describe data streams as data sets with a temporal aspect and generated by some underlying process. This process can be modelled as a random variable, χ, and the data stream's instances as objects drawn from this random variable. An object, o, is a pair $\langle x, y \rangle$ where x is the object's feature vector and y is the object's class label. Each is drawn from a different random variable, X and Y: $x \in \mathrm{dom}(X)$, $y \in \mathrm{dom}(Y)$ and $o \in \mathrm{dom}(X, Y) = \mathrm{dom}(\chi)$ [23].

Many authors have conceptualized concept drift qualitatively [12,25]. Abrupt concept drift is when one concept is immediately replaced by another, e.g. a computer network expands. Gradual concept drift is an alternation between concepts, e.g. a network's computers are slowly upgraded to a new OS. Incremental concept drift, instead sees a series of intermediate concepts, e.g. an office's computer usage evolves over a project's lifetime (Fig. 1).

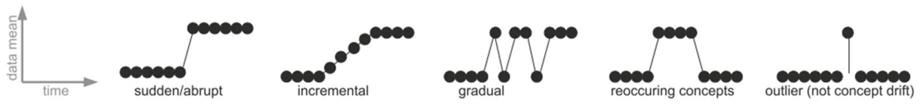


Fig. 1. Qualitatively different types of concept drift (from Gama et al. [12])

Described quantitatively, a data stream's concept at time t is the probability associated with its underlying generative process; Definition 1 [12,26]. Concept drift occurs between points of time t and u in the data stream when $P_t(X, Y) \neq P_u(X, Y)$. This could occur because of changes in $P(X)$ or in $P(Y|X)$. As class labels are not available for the clustering task, we are only concerned with the former. Concept evolution, when a novel pattern emerges, is a special case of concept drift [10,20]. Although a changing number of classes is usually discussed in the context of supervised learning [11,20], $P(X)$ is also likely to affected and so we consider concept evolution here. This might occur when certain activities only occur at certain times, e.g. system updates download overnight.

Definition 1 (Concept). $Concept = P_t(X, Y) = P_t(\chi)$

In the quantitative paradigm, a concept drift's magnitude is the distance between the concepts at times t and u as measured by a distribution distance function, D, (Definition 2) [23]. This is not captured by the types shown in Fig. 1, which do not have the language to describe how far apart two concepts are. Using drift magnitude allows us to distinguish cases with overlapping concepts from cases with divergent concepts.

Definition 2 (Drift magnitude). $Magnitude_{t,u} = D(P_t(\chi), P_u(\chi))$

The duration of a concept drift is the time during which the drift occurs (Definition 3). This distinguishes drifts of different lengths and illustrates that the boundary between abrupt and extended concept drift is a threshold value [23].

Definition 3 (Drift duration). $Duration_{t,u} = u - t$

2.2 Data Stream Clustering Algorithms

DSCAs cluster the instances within data streams as they occur. Barbará argued that they must compactly represent instances; process instances incrementally; and clearly identify outliers [4]. Most DSCAs produce clusterings with an online component to summarize the data stream's instances and an offline component similar to traditional clustering algorithms [14,21]. Although DSCAs use different approaches, we note four general methods of clustering data streams that are discussed in the literature: partitioning, density-based, hierarchical and grid-based. These match Han et al.'s methods of clustering static data sets [16, p. 450].

Partitioning. A partition is produced so that similar objects are in the same partition and dissimilar objects are in different partitions. Partitions can be defined by mean points, representative points, or map nodes. A common drawback is that these methods are only capable of producing hypersphere-shaped clusters and have difficulty representing arbitrarily shaped clusters.

Density-based. Clusters are produced as dense regions separated by less dense regions, allowing them to represent arbitrarily shaped clusters. For example, DenStream [6] models objects with microclusters. Depending on their density, these microclusters are labelled as either core, potentially core or outlier. A key to density-based methods is defining what constitutes a dense region; this is usually done by user-defined parameters.

Hierarchical. Objects are organized into a hierarchical structure, like a tree. In this way, objects are closer in the tree structure to objects that are similar and further away in the tree from those that are dissimilar. The hierarchical nature of the tree structure allows different clusterings to be produced by inspecting the tree at different levels, but may also require the computationally expensive rebuilding of the tree.

Grid-based. The feature space is divided into grid cells, which summarize the data stream objects by acting as bins. D-Stream takes advantage of this to put an upper bound on computations: no matter how many objects arrive, they are represented with a constant number of grids [7]. The grid's fineness (or coarseness) represents a trade off between precision of results and cost of computations; this is generally defined by user supplied parameters.

Silva et al. [21] characterize DSCAs using seven aspects. The first is whether the clustering task is object-based or attribute-based. We focus on the former as its applications are more commonplace [21]. The second is the number of user-defined parameters, e.g. window sizes, decay rates and thresholds. Most notably: some DSCAs require the number of clusters to find - k - which handicaps an algorithm's ability to deal with a data stream's dynamic behaviour [21].

Online Component. A DSCA's online component allows it to process new instances quickly and incrementally; it incorporates three of Silva et al.'s aspects [21]. The third aspect is the data structure used to represent the unbounded instances in a compact manner. One possibility is a feature vector which summarizes N instances, $x_1, x_2, ..., x_N$, as $\langle N, LS, SS \rangle$, where LS is the linear sum of those instances ($\sum_{i=1}^{N} x_i$) and SS is the square sum of those instances ($\sum_{i=1}^{N} x_i^2$) [24]. Prototype arrays summarize partitions using medoids or centroids; these prototypes can later be summarized themselves [21]. Nodes in a self-organizing map or neurons in a growing neural gas are also potential summaries [13]. Alternatively, coreset trees organize $2m$ instances into a tree from which a coreset of

m instances is extracted. Two coresets are reduced by building another coreset tree from their union [1]. Finally, dividing the feature space into grid cells allows each cell to summarize its respective objects [7].

Reasoning that recent objects are more relevant, the fourth aspect is the window model used to passively forget old concepts. Sliding windows store instances in a queue and remove the oldest instance every time a new one is added. Damped windows weight instances by age, often by using an exponential function, until they are forgotten. Meaningful time-based or stream-based landmarks can be used to break the stream into non-overlapping chunks - landmark windows [21].

The fifth aspect is the outlier detection mechanism. The algorithm must decide if a point that doesn't fit the existing clusters represents a new cluster or an outlier to be discarded [4]. Mechanisms to do so include buffering outliers until it is possible to include them in the summarization [6,24] and deleting microclusters or grids with below-threshold density or relevance [2,6,7].

Offline Component. A DSCA's offline component is called when a clustering is required. The summarization is often treated as a static data set and traditional clustering algorithms are used. Silva et al.'s last two aspects are seen here: the offline clustering algorithm used and the resulting clusters' shape [21].

One popular approach for the offline clustering algorithm is the k-means family of clustering algorithms. This includes k-means applied to the statistical summary or to a weighted statistical summary, selecting medoids with k-medoids, and using an initial seeding with k-means++. As expected, DSCAs making use of these algorithms result in hypersphere-shaped clusters. The other popular approach is to use density-based clustering algorithms, such as DBSCAN, applied to feature vectors, grid cells or frequent states. DSCAs that use density-based clustering have the ability to find arbitrarily-shaped clusters [21].

3 The Mixture Model Drift Generator

In order to conduct experiments using real-valued data streams with precise controlled concept drift, we propose a Mixture Model Drift Generator[1] based on Webb et al.'s categorical data generator[2] [23]. The Mixture Model Drift Generator models the periods before and after concept drift - stable concepts - as mixture models with one distribution for each class and a probability vector for choosing between the classes. We use multivariate normal distributions (MVNDs), which are defined by a mean point and a covariance matrix.

3.1 Generating the Underlying Probabilities

The generator requires the number of classes present before, n_0, and after, n_1, the concept drift, the stream's dimensionality, a, the drift magnitude, m, the tolerance for the drift magnitude, ϵ, and the drift duration, d.

[1] Available https://doi.org/10.5281/zenodo.1168699.
[2] Available https://doi.org/10.5281/zenodo.35005.

```
**Data**: $n_0$, $n_1$, $a$, $m$, $\epsilon$
**Result**: Two mixture models $M_0$ and $M_1$
Generate $M_0$: a mixture model of $n_0$ $a$-dimensional MVNDs;
**do**
    Generate $M_1$: a mixture model of $n_1$ $a$-dimensional MVNDs;
**while** $H(M_0, M_1) \neq m \pm \epsilon$;
```

Algorithm 1. Mixture Model Drift Generator

Although any distribution distance functions could be used to measure drift magnitude, we use the Hellinger Distance because it is symmetrical and takes values between 0 and 1, inclusively [23]. The Hellinger distance between real valued probability density functions, $f(x)$ and $g(x)$, is shown in (1). From the last form of (1), the Hellinger distance is equal to 0 when the two functions are identical, $f(x) \equiv g(x)$, and equal to 1 when there is no overlap between them, i.e. $(f(x) \neq 0 \implies g(x) = 0) \wedge (g(x) \neq 0 \implies f(x) = 0)$.

$$H^2(f(x), g(x)) = \frac{1}{2} \int \left(\sqrt{f(x)} - \sqrt{g(x)} \right)^2 dx = 1 - \int \sqrt{f(x)g(x)} dx \tag{1}$$

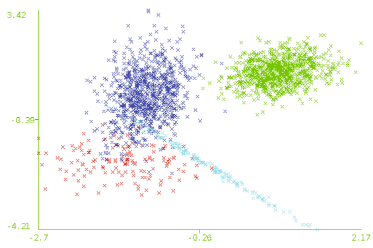
We are unaware of a method to solve the second mixture model's parameters given the first mixture model and m. Instead, mixture models are generated and their Hellinger distance from the first mixture model is calculated until an appropriate second mixture model is found.

3.2 Drawing Instances from the Generator

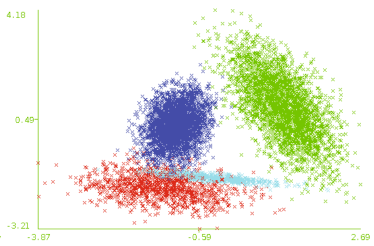
During a stable concept, sampling the mixture model provides the attributes, x, while the specific MVND that was selected provides the class label, y. Together, these form the data stream object $o \in Dom(\chi)$ as introduced in Sect. 2.

Figure 2 illustrates data streams produced by the Mixture Model Drift Generator. Figures 2a and c depict the initial stable concept for two different data streams; different classes are identified by different colours. These mixture models have identical parameters but the instances drawn from each are different. Figures 2b and d depict the final stable concepts for the data streams. The mixture models are separated by a Hellinger distance of 0.4 in the first case and are separated by a Hellinger distance of 0.8 in the second.

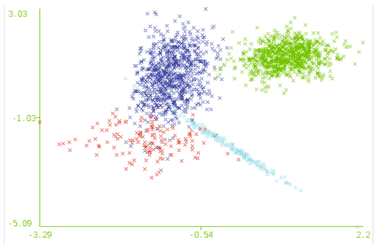
During concept drift, generating instances is based on the qualitative type of the drift. For gradual concept drift, the generator draws the instance from one of the stable concepts with the probability of selecting the original concept decreasing over time. For incremental concept drift, the generator draws instances of the same class from both concepts. These instances are weighted, with the original concept's weight decreasing over time, and returned as the object o.



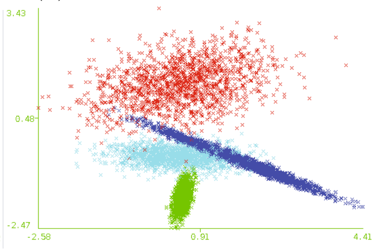
(a) Stream 1: Initial Concept



(b) Stream 1: Final Concept



(c) Stream 2: Initial Concept



(d) Stream 2: Final Concept

Fig. 2. Initial and final concepts for two data streams. $M_1 = 0.4$ and $M_2 = 0.8$

4 Experimental Evaluation

Our research question is “how do the clusterings produced by different DSCAs change, relative to the ground truth, as quantitatively different types of concept drift are encountered?” To answer this, we apply different DSCAs to synthetic real-valued data streams with different concept drifts. We then use these results to select a DSCA for a real word data stream.

4.1 Data Stream Clustering Algorithms

The five algorithms selected (Table 1) were listed by Silva et al. as among the 13 most relevant DSCAs [21]. Importantly, they also cover the four methods identified in Sect. 2.2.

The MOA 17.06 [5] implementation was used for each DSCA. A modified version of ClusTree[3] was used to properly implement the offline k-means clustering algorithm. A modified version of D-Stream[4] was used to permit the specification of grid widths for numerical attributes. Parameters chosen for each algorithm are included in the supplemental materials.

[3] Available https://doi.org/10.5281/zenodo.1216189.
[4] Available https://doi.org/10.5281/zenodo.1213802.

Table 1. DSCA characteristics with the online component summarized above and the offline component summarized below (adapted from Silva et al. [21])

DSCA	Data Structure	Window Model	Outlier Detection
CluStream	feature vector	landmark	statistical-based
ClusTree	feature vector tree	damped	-
D-Stream	grid	damped	density-based
DenStream	feature vector	damped	density-based
StreamKM++	coreset tree	landmark	-

DSCA	Clustering Algorithm	Cluster Shape	Approach
CluStream	k-means	hyper-sphere	Partitioning
ClusTree	k-means	hyper-sphere	Hierarchical
D-Stream	DBSCAN variant	arbitrary	Grid-based
DenStream	DBSCAN variant	arbitrary	Density-based
StreamKM++	k-means++	hyper-sphere	Partitioning

4.2 Synthetic Data Streams

Three experiments were conducted using synthetic data streams. One hundred two-dimensional data streams were produced for each experimental setting using the Mixture Model Drift Generator. For all data streams, the initial stable concept occurs for 2,000 instances to allow the DSCAs to achieve a stable clustering. The final stable concept occurs from the end of concept drift until the end of the data stream, again allowing a stable clustering.

The DSCA's clustering quality is the dependent variable in each experiment. Experiment A maintains four classes and has a drift duration of 1 – this is abrupt concept drift. Drift magnitude is the independent variable with values of 0.4, 0.5, 0.6, 0.7, 0.8 or 0.9. Experiment B has four classes before and after concept drift and a drift magnitude of 0.6. Drift duration is the independent variable with values of 1,000, 5,000 or 9,000 instances for both incremental and gradual concept drift. Experiment C involves a drift duration of 1, a drift magnitude of 0.6 and four class prior to concept drift. The number of post-concept drift classes represents concept evolution and varies between 2, 3, 5 or 6 classes.

4.3 Real World Data Streams

Four data streams were built using the ADFA-LD Anomaly Detection dataset, introduced by Creech et al. to replace the classic KDD 1999 Network Intrusion Detection dataset [8]. Haider et al. described four features they extracted from this dataset and showed a nearest neighbour approach using them for anomaly detection [15], suggesting that sensible clusters exist in this low-dimensional feature space. Our features, based on Haider et al.'s [15], are shown in Table 2.

We used the dataset's validation instances as the stream's normal behaviour. Attacks begin sporadically, dominate the data stream starting at instance 2000

Table 2. ADFA-LD Anomaly Detection features (based on Haider et al. [15])

	Description
1	System call that appears with the highest frequency in the trace
2	Lowest valued system call that appears in the trace
3	Highest valued system call that appears in the trace
4	Number of distinct valued system calls that appear in the trace
5	Ratio of the number of appearances of the most repeated system call in the trace to the total number of system calls in the trace
6	Ratio between the range from lowest frequency to highest frequency of appearance in the trace to the total number of system calls in the trace

and continue for 250–500 instances before returning to their original frequency. This is abrupt concept drift as the two underlying probabilities are swapped immediately. Concept evolution may also occur as the clusters that make up the normal and attack behaviours may not be present throughout.

4.4 Performance Measure

As we have access to each data stream's ground truth, we use an external measure of cluster quality. Many of these exist in the literature, including purity, Rand statistic and Cluster Mapping Measure (CMM). We choose CMM because it is constrained to $[0, 1]$ (0 is the worst clustering, 1 is the best), accounts for different kinds of faults and performed well in Kremer et al.'s experiments [19].

CMM builds a clustering's fault set, its missed points, misassigned points and noise points assigned to clusters, and evaluates each fault point's connectivity to its true and assigned clusters [19]. An object's connectivity to a cluster is the ratio of its average k-neighbourhood distance (*knhDist*, Definition 4) to the average k-neighbourhood distance of the cluster. Each object's penalty is exponentially weighted by its age [19]. Definitions 4–7 are adapted from Kremer et al. [19].

Definition 4 (average k-neighbourhood distance). *The average distance of point p to its k neighbours in C is: $knhDist(p, C) = \frac{1}{k}\sum_{o \in knh(p,C)} dist(p, o)$. The average distance for a cluster C is: $knhDist(C) = \frac{1}{|C|}\sum_{p \in C} knhDist(p, C)$.*

Definition 5 (Connectivity). *Connectivity between object o and cluster C is:*

$$con(o, C) = \begin{cases} 1 & \textit{if } knhDist(o, C) < knhDist(C) \\ 0 & \textit{if } C = \emptyset \\ \frac{knhDist(C)}{knhDist(o,C)} & \textit{else} \end{cases}$$

Definition 6 (Penalty). *$Cl(\cdot)$ returns the ground truth class of the argument object and $map(\cdot)$ returns the ground truth class to which the argument cluster is mapped. The penalty for an object $o \in \mathcal{F}$ assigned to cluster C_i is:*

$$pen(o, C_i) = con(o, Cl(o)) \cdot (1 - con(o, map(C_i)))$$

Definition 7 (Cluster Mapping Measure). *Given an object set $\mathcal{O}^+ = \mathcal{O} \cup Cl_{noise}$, a ground truth $\mathcal{CL}^+ = \mathcal{CL} \cup \{Cl_{noise}\}$, a clustering $\mathcal{C} = \{C_1, ..., C_k, C_\emptyset\}$, and the fault set $\mathcal{F} \subseteq O^+$, the Cluster Mapping Measure between $\mathcal{C}$ and $\mathcal{CL}^+$ is defined using the point weight $w(o)$, overall penalty $pen(o, C)$ and connectivity $con(o, Cl(o))$ as:*

$$CMM(\mathcal{C}, \mathcal{CL}) = 1 - \frac{\sum_{o \in \mathcal{F}} w(o) \cdot pen(o, C)}{\sum_{o \in \mathcal{F}} w(o) \cdot con(o, Cl(o))}$$

and if $\mathcal{F} = \emptyset$, then $CMM(\mathcal{C}, \mathcal{CL}) = 1$.

We used the implementation of CMM in MOA 17.06 [5] to evaluate the clustering produced by a DSCA every 100 instances.

5 Results and Discussion

The results show the average CMM for each setting's 100 data streams.[5] Algorithms that require the number of clusters were given $k = 4$. We use the Friedman test to compare multiple algorithms across multiple domains because it is non-parametric, doesn't assume the samples are drawn from a normal distribution and doesn't assume the sample variances are equal [18, pp. 247–248].

If the Friedman test leads us to conclude that the difference in algorithm performance is statistically significant, we conduct post-hoc Nemenyi tests. This regime is recommended by Japkowicz and Shah [18, p. 256] as well as Demšar [9]; statistical testing used the *scmamp* package in R[6]. Post-hoc Nemenyi test results are shown graphically; algorithms that are not significantly different (at $p = 0.05$) are linked.

5.1 Experiment A - Abrupt Concept Drift

These data streams exhibited abrupt concept drift at instance 2000. Each algorithm's maximum change in cluster quality for a given setting was calculated using the 1500 instances after concept drift. This change was compared to the algorithm's change in quality for the baseline data streams, controlling for unrelated changes in cluster quality, e.g. StreamKM++'s characteristic decrease in quality. To ease interpretation, only magnitudes 0.0, 0.4, 0.6 and 0.8 are shown.

The algorithms' results (Fig. 3) divide into two qualitative groups. CluStream and ClusTree are largely invariant to abrupt concept drift for all magnitudes.

[5] Results for additional cases available: https://doi.org/10.5281/zenodo.1304380.
[6] https://cran.r-project.org/package=scmamp.

The other three algorithms' cluster quality changes due to concept drift, with DenStream and StreamKM++ sensitive when the magnitude of the concept drift is larger. All three algorithms' results behave the same, however: abrupt concept drift is met with a decrease in cluster quality, an extreme cluster quality is reached and then a new stable cluster quality is established for the remainder of the data stream. Larger concept drift magnitudes results in larger decreases in cluster quality for all three algorithms. We also note that the magnitude of the concept drift does not affect the stable cluster quality for the second concept.

Using Friedman's test we conclude that among the five algorithms there is a significant difference in the change of cluster quality due to concept drift ($p < 0.01$). Post-hoc Nemenyi test results are shown in Fig. 4 where the highest ranked algorithm had the highest (most positive) change in cluster quality and the lowest ranked algorithm had the lowest (most negative) change.

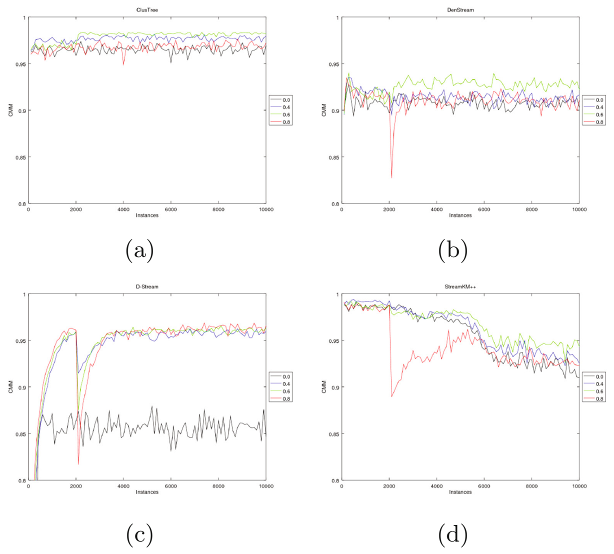


Fig. 3. Experiment A - Data streams with abrupt concept drift

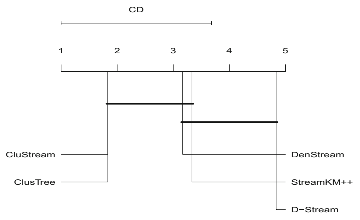


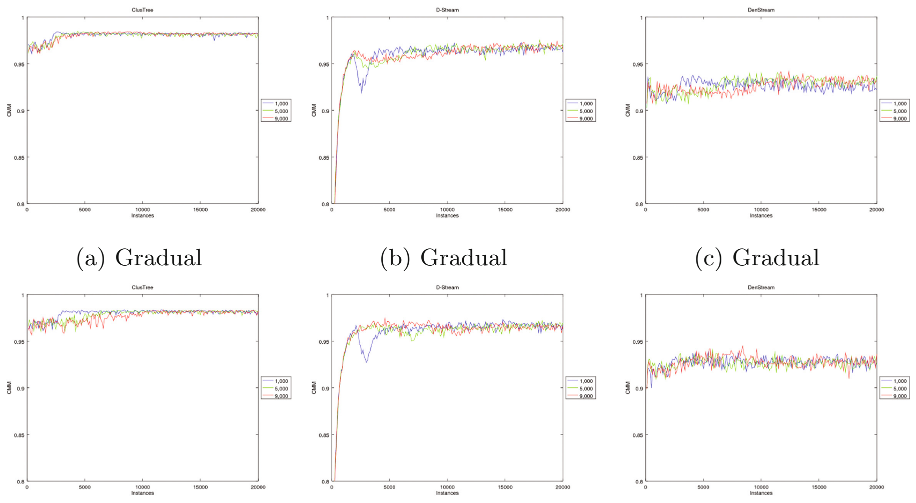
Fig. 4. Nemenyi test results for Experiment A

5.2 Experiment B - Extended Concept Drift

These data streams exhibited either gradual concept drift or incremental concept drift starting at instance 2000 and continuing for the specified duration. Each algorithm's time to reach and time to recover from its extreme CMM value was determined using average performance. Results are shown in Fig. 5.

Qualitatively, the algorithms' results are divided into two groups. CluStream and StreamKM++ are invariant for all durations and for both types. ClusTree, DenStream and D-Stream show changes in cluster quality that are affected by the concept drift's duration and type. DenStream and D-Stream exhibit the same general behaviour from Experiment A: concept drift results in a decrease in cluster quality, an extreme cluster quality is reached and then a new stable cluster quality is established. Longer concept drift durations soften this effect, as seen when comparing the 1,000 duration drift with the 9,000 duration drift for both DenStream and D-Stream; this effect is also dampened when facing incremental concept drift. In contrast, ClusTree's small changes in cluster quality take longer to reach the new stable cluster quality during longer concept drifts and for incremental compared to gradual drift.

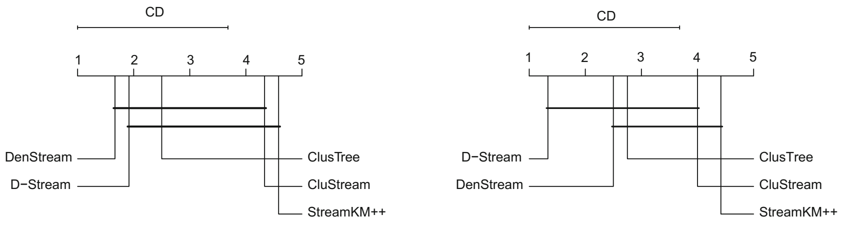
Using Friedman's test we conclude that there is a significant difference among the five algorithms in the time to reach an extreme value due to concept drift ($p < 0.01$) and in the time to recover from that extreme value to a new stable quality ($p < 0.01$); post-hoc Nemenyi test results are shown in Fig. 6.



(a) Gradual (b) Gradual (c) Gradual

(d) Incremental (e) Incremental (f) Incremental

Fig. 5. Experiment B - Data streams with extended concept drift



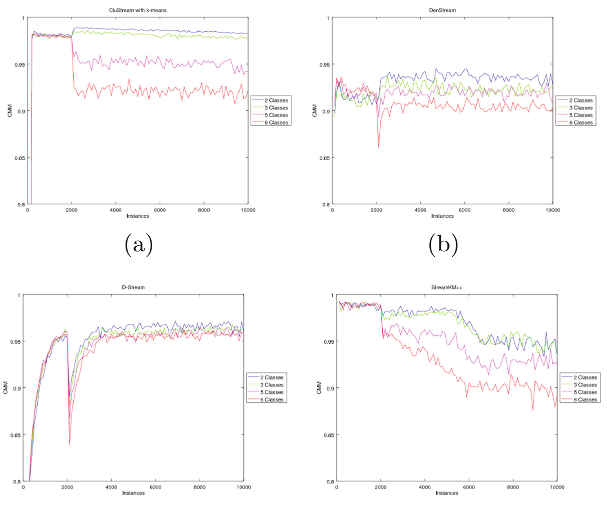
(a) Time to reach extreme cluster quality (b) Time to recover

Fig. 6. Experiment B - Nemenyi test results.

5.3 Experiment C - Concept Evolution

These data streams exhibited concept evolution. That is, concept drift caused the number of classes present in the data stream to either increase or decrease. The change in cluster quality was measured the same way as for Experiment A; results are shown in Fig. 7.

CluStream and StreamKM++, the algorithms with a k parameter, suffered from increasing the number of classes, though neither suffered from decreasing the number of classes, i.e. representing two classes by splitting them across four clusters still resulted in better clusters than attempting to merge six classes into four clusters. DenStream and D-Stream, both algorithms without a k parameter, were also affected by concept evolution.



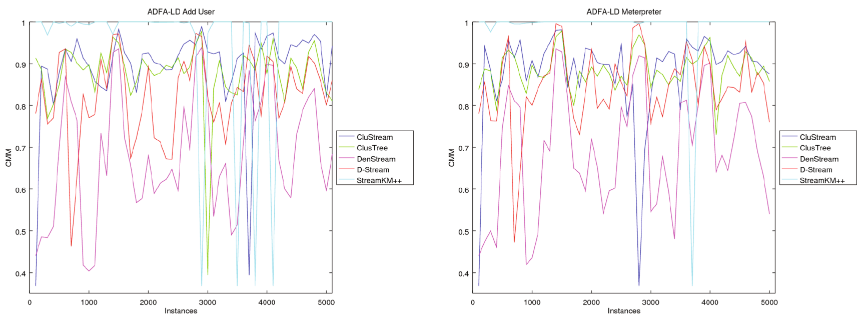
(a) (b) (c) (d)

Fig. 7. Experiment C - Data streams exhibiting concept evolution

Compared to Experiment A, the change in cluster quality was decreased when the number of classes present decreased ($\mu_{\Delta \mathrm{CMM}} = -0.01169$) and increased when the number of classes present increased ($\mu_{\Delta \mathrm{CMM}} = -0.05016$). Using Welch's unequal variances t-test, the difference in cluster quality changes between decreasing the number of classes and increasing the number of classes was found to be statistically significant ($p = 0.04278$).

5.4 Real World Data Streams

Algorithms that required the number of clusters had the parameter k set to 2 by inspection. Cluster quality was generally similar to the quality obtained for the synthetic data streams, with the exception of DenStream. Though more volatile, each algorithm's overall performance is similar to its performance on synthetic data streams with abrupt drift, in line with how we described these data streams in Sect. 4.3. ClusTree, CluStream and StreamKM++ each show stable quality in the face of the first concept drift, although each becomes more volatile after the return to normal behaviour. D-Stream exhibits some volatility throughout and produces, on average, slightly lower quality clusters. DenStream produces consistently lower quality clusters than the other algorithms, though its volatility makes further interpretation difficult (Fig. 8).



(a) (b)

Fig. 8. ADFA-LD Intrusion Detection data streams

We note that DSCA performance is largely the same for all types of attack. This is a promising characteristic as an Intrusion Detection System must be robust against a wide range of known and unknown attacks. Based on these results, a sample recommendation would be to incorporate ClusTree as the DSCA chosen for an Intrusion Detection System. ClusTree does not require k, is generally invariant to concept drift and it consistently produces high quality clusters.

6 Conclusion

The length, speed and dynamic behaviour of data streams each present challenges for learning. Although clustering can be used to address these challenges, this paper highlighted the literature's gap in understanding the behaviour of DSCAs for concept drifting data streams. We proposed a method for generating real-valued data streams with precise quantitative concept drift and used these to conduct an experimental study.

Concept drifts with larger magnitudes and shorter durations were the most difficult for DSCAs, with each resulting in a larger decrease in cluster quality. We observe that the key factor in determining how a DSCA will respond to concept drift is the algorithm it uses in the offline component. DSCAs that incorporate partitioning-based clustering methods are more robust to concept drift while those that incorporate density-based clustering methods are more sensitive. We also observe that an increase in the number of classes present in a data stream is more challenging for DSCAs than a decrease and that this was true even for DSCAs that did not require an explicit k parameter.

Applying these DSCAs to a real world data stream in the domain of intrusion detection, we observed similar behaviour as in our experiments with synthetic real-valued data streams. This provides evidence that these findings can be taken from laboratory settings and used to predict behaviour in real world domains. As discussed, however, the dimensionality and data types of the data stream will also play a role in the DSCAs' abilities to produce quality results.

Future work using these findings could include: studying the performance of DSCAs when faced with complex concept drift; using these insights to develop an anomaly detection framework that incorporates clustering; and using an internal measure of cluster quality as a concept drift detection method.

References

1. 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**(2), 2–4 (2012). https://doi.org/10.1145/2133803.2184450
2. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: 29th Very Large Database Conference, p. 12, Berlin (2003)
3. Ahmed, M., Naser Mahmood, A., Hu, J., Mahmood, A.N., Hu, J.: A survey of network anomaly detection techniques. J. Netw. Comput. Appl. **60**, 19–31 (2016). https://doi.org/10.1016/j.jnca.2015.11.016
4. Barbará, D.: Requirements for clustering data streams. ACM SIGKDD Explor. Newsl. **3**(2), 23–27 (2002). https://doi.org/10.1145/507515.507519
5. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
6. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: Ghosh, J., Lambert, D., Skillicorn, D., Srivastava, J. (eds.) Proceedings of the 2006 SIAM International Conference on Data Mining, pp. 328–339, Bethesda (2006). https://doi.org/10.1137/1.9781611972764.29

7. Chen, Y., Tu, L.: Density-based clustering for real-time stream data. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 133–142, San Jose, USA (2007)
8. Creech, G., Hu, J.: Generation of a new IDS test dataset: time to retire the KDD collection. In: 2013 IEEE Wireless Communications and Networking Conference (WCNC), pp. 4487–4492. IEEE (2013)
9. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006). https://doi.org/10.1016/j.jecp.2010.03.005
10. Faria, E.R., Gonçalves, I.J., de Carvalho, A.C., Gama, J.: Novelty detection in data streams. Artif. Intell. Rev. **45**(2), 235–269 (2016). https://doi.org/10.1007/s10462-015-9444-8
11. de Faria, E.R., de Leon, P., Ferreira Carvalho, A.C., Gama, J.: MINAS: multiclass learning algorithm for novelty detection in data streams. Data Min. Knowl. Disc. **30**(3), 640–680 (2015). https://doi.org/10.1007/s10618-015-0433-y
12. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4), 1–37 (2014). https://doi.org/10.1145/2523813
13. Ghesmoune, M., Azzag, H., Lebbah, M.: G-Stream: growing neural gas over data stream. In: Loo, C.K., Yap, K.S., Wong, K.W., Teoh, A., Huang, K. (eds.) ICONIP 2014. LNCS, vol. 8834, pp. 207–214. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12637-1_26
14. Ghesmoune, M., Lebbah, M., Azzag, H.: State-of-the-art on clustering data streams. Big Data Anal. **1**(1), 13 (2016). https://doi.org/10.1186/s41044-016-0011-3
15. Haider, W., Hu, J., Xie, M.: Towards reliable data feature retrieval and decision engine in host-based anomaly detection systems. Proceedings of the 2015 10th IEEE Conference on Industrial Electronics and Applications, ICIEA 2015, pp. 513–517 (2015). https://doi.org/10.1109/ICIEA.2015.7334166
16. Han, J., Pei, J., Kamber, M.: Data Mining: Concepts and Techniques, 3rd edn. Morgan Kaufmann Publishers, San Francisco (2011)
17. Hoplaros, D., Tari, Z., Khalil, I.: Data summarization for network traffic monitoring. J. Netw. Comput. Appl. **37**(1), 194–205 (2014). https://doi.org/10.1016/j.jnca.2013.02.021
18. Japkowicz, N., Shah, M.: Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press, New York (2011). https://doi.org/10.1017/CBO9780511921803
19. Kremer, H., Kranen, P., Jansen, T., Seidl, T., Bifet, A., Holmes, G.: An effective evaluation measure for clustering on evolving data streams. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2011), pp. 868–876. ACM Press, New York (2011)
20. Masud, M., Gao, J., Khan, L., Han, J., Thuraisingham, B.M.: Classification and novel class detection in concept-drifting data streams under time constraints. IEEE Trans. Knowl. Data Eng. **23**(6), 859–874 (2011). https://doi.org/10.1109/TKDE.2010.61
21. Silva, J.A., Faria, E.R., Barros, R.C., Hruschka, E.R., De Carvalho, A.C., Gama, J.: Data stream clustering: a survey. ACM Comput. Surv. **46**(1), 1–31 (2013). https://doi.org/10.1145/2522968.2522981

22. Souza, V.M.A., Silva, D.F., Gama, J., Batista, G.E.A.P.A.: Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: Proceedings of the 2015 SIAM International Conference on Data Mining, pp. 873–881. Society for Industrial and Applied Mathematics, Philadelphia, PA, June 2015. https://doi.org/10.1137/1.9781611974010.98
23. Webb, G.I., Hyde, R., Cao, H., Nguyen, H.L., Petitjean, F.: Characterizing concept drift. Data Min. Knowl. Disc. **30**(4), 964–994 (2016). https://doi.org/10.1007/s10618-015-0448-4
24. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: an efficient data clustering databases method for very large. ACM SIGMOD Rec. **25**(2), 103–114 (1996). https://doi.org/10.1145/233269.233324
25. Žliobaitė, I.: Learning under concept drift: an overview. Vilnius University, Technical report (2010)
26. Žliobaitė, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with drifting streaming data. IEEE Trans. Neural Netw. Learn. Syst. **25**(1), 27–39 (2014). https://doi.org/10.1109/TNNLS.2012.2236570