Machine Learning
https://doi.org/10.1007/s10994-023-06462-2

Temporal silhouette: validation of stream clustering robust to concept drift

Félix Iglesias Vázquez[1] · **Tanja Zseby**[1]

Received: 8 March 2023 / Revised: 26 July 2023 / Accepted: 17 October 2023

Abstract
Stream clustering is required in applications where data is generated continuously or periodically and must be processed considering its temporal nature. In the absence of a ground truth, internal validation is the only option to evaluate the quality of performances. Traditional internal validation is commonly used also in stream clustering, even in spite of the fact that it becomes inconsistent in the event of data evolution. Recent trends opt for incremental approaches, but these are closer to change detection rather than validation methods and limit themselves by imposing online validation on online analysis. In this work we study the impact of concept drift in the validation of stream clustering and propose the Temporal Silhouette index, therefore making internal validation conform to streaming data. We conduct tests with more than 200 datasets and contrast performances of four popular stream clustering algorithms with seven validation methods (three static internal, three incremental internal, one external) and the proposed index. Results show the suitability of the Temporal Silhouette index for stream clustering validation in the event of concept drift and different types of outliers. The demand for reliable unsupervised learning in applications that process data in streams is ever-increasing, and such reliability inevitably requires the use of validation. This fact highlights the significance of the novel approach proposed in this work.

Keywords Stream clustering · Clustering validation · Multivariate time series

1 Introduction

Streaming data analysis has become highly relevant due to the impact of big data and pervasive communications. Nowadays data is generated, transmitted and consumed on a massive scale. Streaming data analysis not only implies data being processed incrementally,

Editors: Dino Ienco, Robert Interdonato, Pascal Poncelet.

✉ Félix Iglesias Vázquez
felix.iglesias@tuwien.ac.at

Tanja Zseby
tanja.zseby@tuwien.ac.at

[1] Institute of Telecommunications, TU Wien, Gusshausstraße 25 / E389, 1040 Vienna, Austria

also emphasizes "time" and the "temporal nature" of the data as key aspects of the knowledge to be discovered through analysis.

Leaving univariate time series analysis aside, the maturity of theoretical knowledge and technological developments in data analysis are much more consolidated in static scenarios than in dynamic ones. In a recent paper, Bezdek and Keller (2021) point out that "useful analysis of real streaming data is in its infancy". The authors, who particularly focus on stream clustering, observe that most "state-of-the-art" algorithms are based on batch clustering, which may not be applicable in some cases (e.g., real-time responses and update of models is required, or process memory must be decoupled from the batch size and artifacts). Although their examples rather belong to anomaly detection (e.g., intrusion detection), most popular stream clustering are actually batch extensions of traditional static clustering.

Beyond aspects related to computational costs, the fundamental difference between stream and static clustering is nonstationarity, i.e., changes over time that the phenomena represented by the data may undergo. This often increases the analysis challenge, since, for instance, algorithms must be able to update their models to cope with the sudden appearance of new classes. Context changes in data are commonly referred to as *distribution shift* (Moreno-Torres et al., 2012) or *concept drift* (Gama et al., 2014).

In this respect, adaptation to the new or unknown is usually faced from unsupervised or semi-supervised approaches, particularly stream clustering and stream anomaly detection. Later on, the consistency of clustering is evaluated by means of clustering validity indices (CVI). Due to the inherent ambiguity of clustering, it has been remarked that it is to be assessed taking into account application goals and peculiarities (von Luxburg et al., 2012; Iglesias et al., 2020); however, context-independent validation is also useful in design phases to select most appropriate methods, for refinement and optimization, and as a support tool in descriptive analytics (Arbelaitz et al., 2013).

As for stream clustering, in the absence of benchmark partitions, the internal validation is commonly performed with traditional static validation, which is blind to concept drift. On the other hand, incremental validation indices (iCVI) have emerged and gained popularity in recent years. These perspectives are rather change detectors—similar to those proposed in multivariate time series analysis, (Kuncheva, 2013)—than validation techniques per se. They therefore share the limitations of incremental analysis, e.g., tied to forgetting factors, focused on the last arriving data points, unable to use future values, or hardly capable to provide global validations. Such limitations are unavoidable in iCVI as algorithms subscribe to the assumption—which, although useful in many applications, is otherwise arbitrary—that online analysis requires online validation.

Concept drift remains relatively unexplored in the field of stream clustering despite its impact (Silva et al., 2013). This is even more true for its validation. As posed by Gama et al. (2014): "in unsupervised learning over evolving data, and in case of delayed and on-demand labeling in supervised learning, validation of change detection and adaptation mechanisms only start to be investigated". Furthermore, clustering requires robust and detailed evaluation. Improving evaluation methods leads to higher reliability of algorithms, thus facilitating their integration into real-life AI. As claimed by von Luxburg et al. (2012): "What is missing is not 'better' clustering algorithms but a problem-centric perspective in order to devise meaningful evaluation procedures".

With a perspective focused on the problem of concept drift, in this paper we propose a new internal validation index—the Temporal Silhouette (TS)—that can cope with the variations that clusters may experience over time. Furthermore, it is robust to spatial outliers as well as to out-of-phase outliers (i.e., points that share the geometric space of a cluster, but

not its time of occurrence), the latter commonly ignored in the literature. The name "Temporal Silhouette" is inherited from the popular Silhouette validation (Rousseeuw, 1987), on whose fundamental equations our method is based. Robust algorithms capable of coping with concept drift are not only relevant because of specific application requirements, but also due to the need of learning models that operate autonomously while maintaining their quality and reliability over time.

The rest of the paper is organized as follows. Section 2 reviews main approaches to stream clustering and its validation. Section 3 presents concept drift and its different types. Section 4 explains the rationale behind TS, our proposal for stream clustering validation, and shows simple examples for each case of concept drift and outliers. In Sect. 5, TS is tested as a traditional (i)CVI with four popular datasets for clustering evaluation and facing perturbations related to underclustering, overclustering, slicing and impurity. The sensitivity and coherence of TS parameters is also checked with the same experimental setup. The response of TS against concept drift is evaluated in Sect. 6 with 200 datasets and with four state-of-the-art stream clustering algorithms. Two additional examples with real data are shown in Sect. 7. The paper ends with the conclusions in Sect. 8.

2 Stream clustering

Within streaming data analysis, stream clustering can be briefly defined as the task of on-the-fly clustering data obtained in a continuous process over time. Traditionally, this task has been performed by batch clustering, i.e., algorithms that deal with the incoming data in chunks, process each chunk sequentially and update their internal models accordingly. In their survey about stream clustering, Silva et al. (2013) collect some requirements that, although demanding, should be kept in mind for the design of future algorithms:

> (i) provide timely results by performing fast and incremental processing of data objects; (ii) rapidly adapt to changing dynamics of the data, which means algorithms should detect when new clusters may appear, or others disappear; (iii) scale to the number of objects that are continuously arriving; (iv) provide a model representation that is not only compact, but that also does not grow with the number of objects processed (notice that even a linear growth should not be tolerated); (v) rapidly detect the presence of outliers and act accordingly; and (vi) deal with different data types, for example, XML trees, DNA sequences, GPS temporal and spatial information.

Silva et al. mention the following algorithms as the most relevant: BIRCH, CluStream, ClusTree, D-Stream, DenStream, DGClust, ODAC, Scalable k-means, Single-pass k-means, Stream, Stream LSearch, SWClustering, and StreamKM++. We briefly introduce here the algorithms used in our experiments.

BIRCH (Zhang et al., 1996) is perhaps the oldest clustering algorithm that can be naturally used in streaming setups. It was designed with the idea of solving the problem of processing large datasets while minimizing computational costs. It processes data incrementally, updates its internal models, and can handle outliers. CluStream is an extension of BIRCH (Aggarwal et al., 2007) that relies on micro-clusters, which are structures similar to the *clustering features* in BIRCH but with additional time-related dimensions. DenStream (Cao et al., 2006) is a density-based algorithm; as CluStream, it uses micro-clusters, which here are classified based on their density before clustering them with DBSCAN (Ester et al., 1996). Finally, StreamKM++ (Ackermann et al., 2012) is an adaptation of *k*-means

to streaming environments that reduces the computational effort by using coresets and a tree structure to construct and save coreset information.

The reader will find another comprehensive work on stream clustering in (Nguyen et al., 2015). In both mentioned surveys, concept drift is seen as one of the main challenges to face, as well as the fact that most algorithms neglect it. New algorithms appear regularly, and concept drift is receiving increased attention. Recent examples are: EmCStream (Zubaroğlu & Atalay, 2022), in which concept drift is particularly addressed by a stream clustering method based on data stream embedding; also EvolveCluster, whose authors remark in this respect: "it is harder to determine if the data objects are outliers or if the streams' concepts are evolving as the stream evolves" (Nordahl et al., 2021).

2.1 Stream clustering validation

Research on clustering validation is extensive, but also mainly focused on static setups. In the survey by Nguyen et al. (2015), the validation of stream clustering is addressed from a purely external perspective, i.e., by comparing results with Ground Truth (GT) partitions. In this line, some popular external validity measures used in clustering are: the Sum of Squared Errors, the Rand index (Hubert & Arabie, 1985), the Adjusted Mutual Information Score (Vinh et al., 2010), and the Cluster Mapping Measure (Kremer et al., 2011).

Nevertheless, clustering, being unsupervised, requires internal validation since GT is often not available. Also, validation is intended to evaluate theoretical properties that algorithms should maximize, e.g., intra-cluster compactness and inter-cluster separation. In this respect, internal validation seeks generality, while external validation is application-specific, meaning that it is compliant to the GT used, which is not forced to guarantee any kind of mathematical or topological property.

Many methods exist for CVI; to cite some widely used: the Silhouette index (Rousseeuw, 1987), the Davies-Bouldin index (Davies & Bouldin, 1979), the Xie-Beni index (Xie & Beni, 1991), and the Calinski-Harabasz index (Caliński & Harabasz, 1974). Liu et al. (2010) compare 11 popular CVIs when facing issues related to monotonicity, noise, density, subclusters and skewed distributions. Hassani and Seidl (2017), who observe that stream clustering is almost exclusively externally validated, compare the same 11 CVIs in stream clustering.

However, the trend in recent years has been the development of incremental versions of internal validation algorithms (iCVI). For instance, Moshtaghi et al. (2019) propose incremental versions of the popular Xie-Beni and Davies-Bouldin indices and use them to analyze stream clustering with Sequential k-means and Online Ellipsoidal Clustering. Ibrahim et al. (2018) study and extend the incremental Davies-Bouldin index and use it to validate performances of the Extended Robust Online Streaming Clustering algorithm. Later, the incremental version of the Partition Coefficient and Exponential Separation algorithm is presented and tested with the MU Streaming Clustering algorithm in (Ibrahim et al., 2019). Perhaps the most exhaustive work on incremental validation for clustering is the contribution by Brito Da Silva et al. (2020), in which 13 iCVIs are compared (seven of them proposed in the cited work). Insights of this study are mainly focused on over-partitioned and under-partitioned clustering and the capability of iCVIs to detect such undesired behaviors.

It seems logical to think that stream clustering, in principle aiming at online data analysis, also requires online validation. This is certainly a useful feature, but not mandatory. Validation can be offline, or not be forced to give an instantaneous response in the forefront of the analysis. The imposition of online validation makes the task more difficult and ends

up transforming proposed algorithms into change detectors rather than validation methods. This has already been pointed out by Moshtaghi et al. (2019): "In turn, the iCVIs we have derived are also misnamed, because they are not really CVIs; they are functions derived from batch CVIs that track computational performance, enabling the user to control and analyze the dynamic performance of the streaming algorithms to which they are attached. A better term for our iCVIs might be something like incremental Performance Monitors (iPMs)".

Thus incremental validation inherits the challenges associated with incremental learning. It implies adaptive models (Giraud-Carrier, 2000), which might require being configured with or without forgetting factors (Moshtaghi et al., 2019), and are also forced to estimate with little information whether the latest data points should be considered outliers, belonging to a new cluster or to an old cluster. This, for example, makes that clusters forming intermittently among other clusters are difficult to spot with iCVI if not detected before by the clustering algorithm (Ibrahim et al., 2019).

Finally, note that none of the previously mentioned papers about stream clustering validation explicitly address the issue of nonstationarity nor analyze the implications of concept drift. In fact, clusters in many of the datasets used in their experiments are stationary (broadly speaking) and stable in time. Since concept drift is a major challenge in stream clustering, validation methods should be tested with datasets showing different types of concept drift.

3 Concept drift

Concept drift adds the differentiating factor between static and streaming data analysis. In its absence, the problem would boil down to efficiently processing large datasets; static analysis methods would be equally valid for streaming data, and the temporal dimension could be simply ignored.

Gama et al. (2014) formally define *concept drift* as:

$$\exists X : p_{t_0}(X, y) \neq p_{t_1}(X, y), \tag{1}$$

where p_{t_0} and p_{t_1} stand for the joint distribution between the input set of features X and the target variable y in times t_0 and t_1 respectively. Equation 1 indicates that, in the lapse between t_0 and t_1, there is a variation in the mechanisms that generate data with regard to their hypothetical classification. From a Bayesian perspective, this is changes in the class prior probabilities $p(y)$, class conditional probabilities $p(X|y)$, and/or class posterior probabilities $p(y|X)$. Hence, two types can be defined: (a) *real concept drift*, which implies changes in $p(y|X)$ regardless of $p(X)$; and (b) *virtual drift*, meaning changes in $p(X)$ that do not affect $p(y|X)$.

Intuitively, we can say that *real drift* implies the modification of ideal classification boundaries, while *virtual drift* is a change in how data appear, without implying a change in how they should be classified. Differentiating between real or virtual drift is pertinent in supervised problems; however, it becomes elusive in unsupervised environments, since the decision on whether or not the classification boundaries should change is ambiguous and application-dependent.

3.1 Types of changes over time

Beyond *real drift* and *virtual drift*, concept drift can also be classified according to the types of change observed in data over time. Gama et al. (2014) differentiate:

- *Sudden drift*, which implies an abrupt shift in concepts, e.g., what was blue suddenly turns to red and remains.
- *Incremental drift*, which involves a slow or gradual transition from one state to another, e.g., what is blue gradually transforms into red in a transition that passes through violet.
- *Gradual drift*, which implies a discontinuous change between concepts that, during a period, coexist with evolving persistence, e.g., what was only blue appears red from time to time, more and more often, until only red remains.
- *Reoccurring concepts*, referring to concepts that intersperse in time, e.g., colors in a traffic light.

Gama et al. (2014) also consider *outliers* as "one of the challenges for concept drift handling algorithms"; in other words, outliers do not imply concept drift, but noisy information that disrupts algorithms. Here we differentiate two types of outliers that are congruent with streaming data:

– *Spatial outliers*, which match the traditional definition and include both far and local outliers. We can see them as data points that lie in areas of the feature space that do not belong to any class regardless of time.
– *Temporal outliers* (aka *contextual* or *out-of-phase outliers*), which are samples that fall in areas of the feature space corresponding to classes or clusters, but isolated in time or not at the expected time.

For a discussion of types of anomalies, novelties and outliers, we address the reader to the survey by Ruff et al. (2021). Due to their subjectivity, we set aside collective anomalies (interpretable as both anomalies or clusters, i.e., anomalous clusters). Note the difficulties to detect collective anomalies in evolving data—they are commonly addressed in univariate time series analysis (Fisch et al., 2022).

4 Temporal validation

The Temporal Silhouette (TS) index attempts to estimate the coherence of clusters relative to time.

Like the static Silhouette, it is an index that tends to 1 for perfect clustering and can be calculated element-wise, cluster-wise and solution-wise. It is formed by four components: the Distance to Temporal Centroid (α), the Temporal Inter-Distance (β), the Inter-Arrival Discrepancy (γ) and the Temporal Centroid Discrepancy (δ). We describe and combine them below after introducing some basic terms.

Given T as an ordered sequence of n real variables[1]:

[1] Regardless of the definition, for the sake of simplicity in our experiments simultaneity is not allowed and any consecutive points of a dataset are separated by a temporal unit.

$$T = \{t_1, ..., t_n\}, \quad t_i \in \mathbb{R} \tag{2}$$

A cluster A can be seen as a set of m multivariate data points (or multidimensional vectors) indexed in time order, i.e., $m = |A|$. Each data point in A takes an instant time from T, therefore:

$$A = \{x_{a,1}, ..., x_{a,m}\} \tag{3}$$

where A shows a one-to-one correspondence with the S_T^A subsequence of T, i.e., arrival times of data points in A.

$$S_T^A = \{t_{a,1}, ..., t_{a,m}\} \tag{4}$$

4.1 Distance to temporal centroid (α)

The objective of the Distance to Temporal Centroid component (α) is to estimate the deviation of data points from the inertia shown by their cluster over time. Clusters that remain or evolve progressively as a whole minimize α.

For a cluster A, if we define an observation window w as an even number of points, we can estimate a *temporal centroid* at instant t ($t \in S_T^A$) as:

$$c_t = \frac{1}{w} \sum_{i=o-w/2}^{o+w/2} x_i, \quad x_i \in A \tag{5}$$

where o is the index of t in A. Hence, the calculation of c_t can be seen as a multidimensional simple moving average (SMA). We define α as the mean of distances between a given data point in time t and its corresponding temporal centroid. Therefore, for cluster A:

$$\alpha_t = d(x_t, c_t), \quad t \in S_T^A \tag{6}$$

4.2 Temporal inter-distance (β)

The goal of the Temporal Inter-Distance (β) is to estimate the distance from data points to the closest-in-time data points from other clusters. Clusters far from each other or that do not overlap in space or time maximize β.

Given an arbitrary data point x_t in cluster A, we can find the set of k nearest-in-time neighbors that do not belong to A, we call it A'_k.

Hence, B_k is any subcluster of B within the A'_k subset (i.e., $B_k \subseteq B$ and $B_k \subseteq A'_k$). Therefore, B represents any cluster in the global solution that is not A. β_t is the distance between x_t and the closest B_k subcluster in A'_k.

$$\beta_t = \min_{B_k \neq A} \frac{1}{|B_k|} \sum_{i \in B_k} d(x_t, x_i), \quad x_i \in B_k, \, t \in S_T^A \tag{7}$$

4.3 Inter-arrival discrepancy (γ)

The objective of the Inter-Arrival Discrepancy component (γ) is to estimate the coherence between the time elapsed in consecutive points of the same cluster. A temporally well-formed cluster minimizes γ.

The set of temporal distances between consecutive data points of A is:

$$\Delta S_T^A = \{t_{a,2} - t_{a,1}, ..., t_{a,m} - t_{a,m-1}\} \tag{8}$$

Outliers in ΔS_T^A indicate either out-of-phase outliers or disruption patterns, therefore decreasing the temporal coherence of the cluster. Hence,

$$\gamma_A = \frac{|\{d_t \in \Delta S_T^A \mid d_t > 3 \text{ MAD } (\Delta S_T^A)\}|}{|A|} \tag{9}$$

which is the number of outliers in ΔS_T^A divided by the cardinality of A. Equation 9 uses the median absolute deviation (MAD) for finding outliers in a one-dimensional set as recommended in (Leys et al., 2013). Since the distribution of ΔS_T^A is not expected to be normal, we use a *consistency* constant based on the 0.75 quantile and consider values above 3 times the MAD as outliers.

4.4 Temporal centroid discrepancy (δ)

The Temporal Centroid Discrepancy (δ) shows some similarity with γ but in the spatial domain and with regard to temporal centroids. It is a factor that penalizes extreme jumps in the evolution of temporal centroids, since it is assumed that they must show either no variation or a smooth variation. For cluster A, the series of absolute distances from consecutive temporal centroids is:

$$\Delta c_A = \{|c_{a,2} - c_{a,1}|, ..., |c_{a,m} - c_{a,m-1}|\} \tag{10}$$

Again, we use the number of outliers of an univariate series to define δ_A:

$$\delta_A = \frac{|\{d_c \in \Delta c_A \mid d_c > 3 \text{ MAD } (\Delta c_A)\}|}{|\Delta c_A|} \tag{11}$$

Intuitively, δ_A is the proportion of big jumps in the temporal centroid evolution.

4.5 Temporal silhouette (*ts*, *TS*)

A *ts* score is calculated per data point by combining α, β, γ and δ components. α and β are related as in the traditional Silhouette score, whereas δ is added to diminish the quality of the cluster in the event of incoherent centroid evolution and γ in the event of inter-arrival discrepancies. A parameter ς weights the impact of γ.

The core part of the index (α, β) is translated two times (+1 and −1) to ensure that the penalties involved by γ and δ always make the score decrease. Therefore, for data points in cluster A:

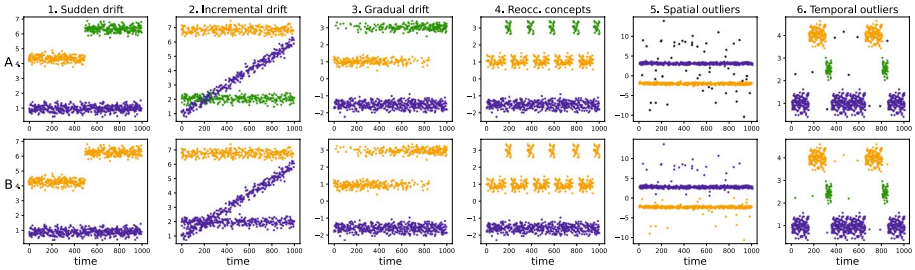

Fig. 1 Time-related challenges (x-axis: time, y-axis: 1D-data). Top and bottom rows show respectively correct and wrong clustering (Color figure online)

$$ts_t = \left(1 + \frac{\beta_t - \alpha_t}{\max(\beta_t, \alpha_t)}\right) \times \frac{1 - \delta_A}{1 + \varsigma\, \gamma_A} - 1, \quad t \in S_T^A \tag{12}$$

As with Silhouette, the following exception applies: $ts_t = 0$ if $|A| = 1$. The *ts* score for cluster A is calculated as the mean *ts* score of its data points:

$$ts_A = \frac{1}{|A|} \sum_t ts_t, \quad t \in S_T^A \tag{13}$$

To properly deal with outliers, when combining all cluster scores in a final TS index we want to represent their effective magnitude rather than their central tendency. The quadratic mean of cluster temporal silhouettes weighted with their respective cardinality and keeping the sign of each individual cluster score is an option that avoids losing representativeness. Therefore, if all data points are univocally split into the set of clusters $\{A, ..., Z\}$,

$$TS' = \frac{\text{sgn}(ts_A)\, ts_A^2 |A| + ... + \text{sgn}(ts_Z)\, ts_Z^2 |Z|}{|A| + ... + |Z|} \tag{14}$$

$$TS = \text{sgn}(TS') \sqrt{|TS'|} \tag{15}$$

For the calculation of the TS index we have defined three external parameters: w, k and ς. w and k are highly robust and have primarily a computational implication. While w affects the inertia of the SMA, k determines the external neighborhood to which each point is compared. In most of the experiments, there were no significant variations in validations with values for w between

[20, ..., 200] and k between [200, ..., 2000] (the robustness of these parameters is analyzed in Sect. 5.5). However, parameter ς decides the importance of time inconsistency, which depending on the application may or may not be desired. Empirically the default value $\varsigma = 1$ is set as a good trade-off. In Sect. 4.6 the effect of this parameter is shown with simple examples.

4.6 Examples of TS validation for concept drift types

The four types of changes over time plus the two types of outliers introduced in Sect. 3.1 are the main types of time-related challenges that stream clustering algorithms commonly face. They are shown with one-dimensional examples in Fig. 1. In the examples,

Table 1 TS scores for two clusterings, A and B, on the six types of temporal challenges and for validations with $\varsigma = 0$ and $\varsigma = 1$. Rows and columns match plots in Fig. 1

Case	A ($\varsigma = 1$)	B ($\varsigma = 1$)	A ($\varsigma = 0$)	B ($\varsigma = 0$)
1. Sud	**0.864**	0.733	**0.886**	0.754
2. Inc	**0.688**	0.667	**0.849**	0.742
3. Grad	**0.733**	0.563	**0.901**	0.663
4. Reoc	**0.790**	0.740	**0.902**	0.849
5. S.Out	**0.863**	0.704	**0.910**	0.762
6. T.Out	**0.824**	0.823	0.831	**0.848**

Bold values show the solution evaluated as "the best" by the TS index when A and B clustering are compared

two clustering are shown: correct in the top plots (case A) and wrong in the bottom plots (case B). We briefly explain the examples:

1. *Sudden drift*. In this example a cluster abruptly disappears at t=500 and is replaced by a new cluster. Case A interprets a new cluster, while clustering in B is blind to such change.
2. *Incremental drift*. In this scenario two clusters maintain their position over time while a third cluster varies its centroid linearly. Solution A distinguishes the three clusters, but solution B merges intersecting clusters.
3. *Gradual drift*. In this case the drift happens gradually: while one cluster slowly appears, a second cluster disappears progressively. Again, clustering A distinguishes them, while clustering B wrongly groups them together.
4. *Reoccurring concepts*. In this scenario two clusters intersperse their presence over time while a third cluster remains. Clustering A differentiates intermittent clusters, but clustering B sees them as the same one.
5. *Spatial outliers*. Here two clusters surrounded by outliers remain stationary. Outliers account for 5% data points and are local and far. Clustering A separates outliers into an independent group, while clustering B absorbs them into the two clusters according to their distance from the centroids.
6. *Temporal outliers*. In this scenario three clusters appear and disappear periodically. 1% of the data points are outside the normal periods of their clusters. In case A the algorithm identifies out-of-phase outliers, while clustering B absorbs them into clusters geometrically coherent.

Table 1 collects TS scores for all clusterings shown in Fig. 1. Scores for case A are always higher than scores for case B, thus confirming the suitability of TS in the given examples. Note the exception for the Scenario 6 and TS with $\varsigma = 0$. By canceling the inter-arrival discrepancy component, out-of-phase outliers are seen as inliers by the validation.

5 Evaluation in stationary environments

In this section we study the response of TS and other validations when facing suboptimal clustering. We model three common sources of distortion (overclustering, undercluster-ing and purity) plus the most elemental perturbation in stream clustering: cluster slicing (or time overclustering). Perturbations are gradually added in the labels of four datasets

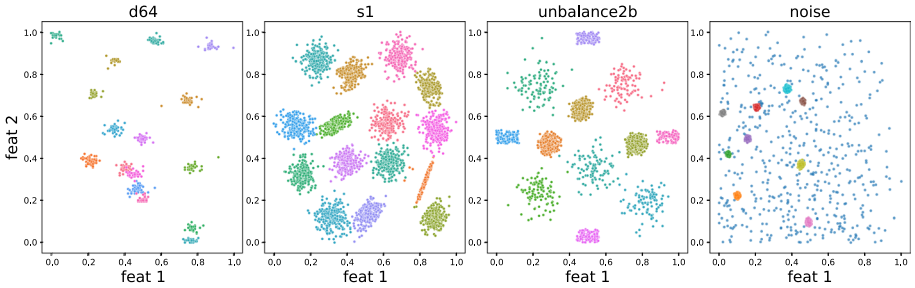

Fig. 2 Spatial view of the datasets used for the evaluation in stationary degradation and the sensitivity of parameters experiments. *d64* only shows 2 dimensions out of 64 (Color figure online)

that pose different space challenges. The sensitivity of TS parameters is later explored with the same experimental setup. The implementation of the TS index and all experiments and tests described in this work are freely available in our repository (TUWien - CN Group, 2023). A permanent, immutable version for repeatability is also available (Iglesias Vázquez, 2023).

5.1 Data

For these tests we use datasets obtained from the Clustering Basic Benchmark of the University of Eastern Finland (Fränti & Sieranoja, 2018), a popular data repository for clustering evaluation.[2] They are: *s1* (Fränti & Virmajoki, 2006), *unbalance2b* (based on the *umbalance2* dataset (Rezaei & Fränti, 2020) with four extra clusters), *d64* (Fränti et al., 2006). The *noise* dataset has been created with the MDCgen tool (Iglesias et al., 2019). Datasets have been selected to represent different spatial challenges: (a) *d64*: medium/high number of dimensions; (b) *s1*: space with very close clusters; (c) *unbalance2*: clusters with different density and cardinality; and (d) *noise*: clusters surrounded by noise. In all four cases, data points are shuffled and sequentially timestamped to give them a temporal dimension. Figure 2 shows a two-spatial-dimensional view of the datasets.

5.2 Perturbations

To simulate suboptimal clustering, GT labels of the datasets introduced in Sect. 5.1 are submitted to four different types of perturbation, each one of them in eight different levels of degradation, from 0 (no-degradation, equal to GT) to 7 (high-degradation). A summary is shown in Table 2. Perturbations are:

- *Overclustering*, meaning arbitrarily splitting proper clusters into glued clusters. This is done by halving clusters in one of their spatial dimensions chosen at random. Therefore, a low degradation (1) implies one extra cluster when compared to the GT partition, while the highest degradation (7) will show seven additional clusters.
- *Underclustering* means joining clusters that should be separated and assigning them the same label. In our experiments this is performed by merging the clusters that show

[2] https://cs.joensuu.fi/sipu/datasets/

Table 2 Perturbations and degradation levels added in the experiments. 'cl.' stands for 'clusters'. The GT is used for 0 degradation

	Degradation level							
	0	1	2	3	4	5	6	7
Overclustering	GT	+ 1 cl	+ 2 cl	+ 3 cl	+ 4 cl	+ 5 cl	+ 6 cl	+ 7 cl
Underclustering	GT	− 1 cl	− 2 cl	− 3 cl	− 4 cl	− 5 cl	− 6 cl	− 7 cl
Slicing	GT	+ 1 cl	+ 2 cl	+ 3 cl	+ 4 cl	+ 5 cl	+ 6 cl	+ 7 cl
Impurity	GT	4.2%	8.3%	12.5%	16.7.0%	20.8%	25.0%	29.2%

closest centroids. This process is repeated according to the degradation level, but always ensuring at least two clusters in the final solution. Therefore, if compared with the GT partition, a low degradation (1) implies one cluster less, while the highest degradation (7) will show seven clusters less.

- *Slicing* (or time overclustering) involves chopping up a legitimate cluster along the time dimension. We select the biggest cluster and subdivide it into two new clusters. The cutoff point is set randomly in a way that each subcluster gets between 40% and 60% of the data points of the original cluster. Again, the lowest degradation (1) will show one extra cluster with respect to the GT partition, whereas the highest degradation will have seven additional clusters.
- *Impurity* refers to data points randomly assigned to the wrong cluster. We increase the percentage of wrongly clustered data points based on the degradation level (Table 2).

5.3 Validation indices

In addition to TS, we also test other validation techniques. CVIs under comparison are: the *Silhouette* (Sil) (Rousseeuw, 1987), *Davies-Bouldin* (DB) (Davies & Bouldin, 1979) and *Calinski-Harabasz* (CH) (Caliński & Harabasz, 1974) indices, as implemented in the `scikit-learn 1.1.1` library (Pedregosa et al., 2011). From the same source is also the external validation index taken as benchmark, namely: the *Adjusted Mutual Information* (AMI) (Vinh et al., 2010).

iCVIs evaluated are: the incremental Xie-Beni index (iXB) (Moshtaghi et al., 2019), the incremental Partition Separation index (iPS) (Brito Da Silva et al., 2020), and the incremental representative Cross Information Potential (irCIP) (Brito Da Silva et al., 2020), as implemented in the Python `cvi 0.4.0` library.[3] TS is set with default parameters: $w = 100$, $k = 1000$, $\varsigma = 1$.

5.4 Results and discussion

Figures 3, 4, 5 and 6 show the performances of the studied indices when adding perturbations to GT labels to simulate suboptimal clustering for each one of the selected datasets.

[3] https://pypi.org/project/cvi/

Fig. 3 Performances of studied validation indices under different levels of overclustering, underclustering, slicing and impurity degradation for the **d64** dataset case. Scores are scaled and inverted (when required) to show 0 for the worst performance and 1 for the best performance in the set. The *y-axis* shows the *level of degradation* (Table 2)

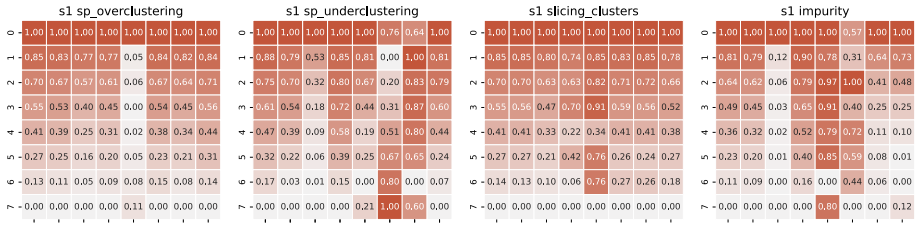

Fig. 4 Performances of studied validation indices under different levels of overclustering, underclustering, slicing and impurity degradation for the **s1** dataset case. Scores are scaled and inverted (when required) to show 0 for the worst performance and 1 for the best performance in the set. The *y-axis* shows the *level of degradation* (Table 2)

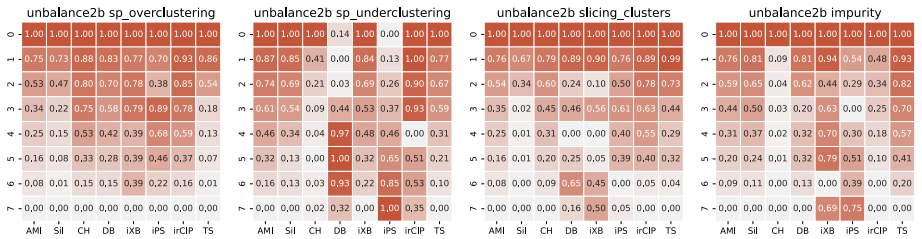

Fig. 5 Performances of studied validation indices under different levels of overclustering, underclustering, slicing and impurity degradation for the **unbalance2b** dataset case. Scores are scaled and inverted (when required) to show 0 for the worst performance and 1 for the best performance in the set. The *y-axis* shows the *level of degradation* (Table 2)

To facilitate the comparison among indices without altering dynamical differences among scores, given a dataset, a type of perturbation and a validation index, the set of scores for the different levels of degradation are min-max scaled to show 1 for the best performance and 0 for the worst performance. Since DB, iXB and irCIP indicate better clustering for lower index scores, they are previously inverted. Thus, a proper internal validation is expected to show either '1' for the 0-degradation case and '0' for the 7-degradation case or correlation with the external AMI index.

Results shown in Figs. 3, 4, 5 and 6 reveal that CVIs are significantly more stable and consistent than iCVIs. This is not surprising and subscribes to the reasoning suggested in

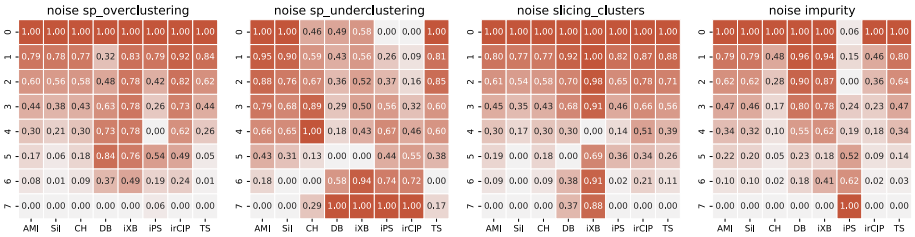

Fig. 6 Performances of studied validation indices under different levels of overclustering, underclustering, slicing and impurity degradation for the **noise** dataset case. Scores are scaled and inverted (when required) to show 0 for the worst performance and 1 for the best performance in the set. The *y-axis* shows the *level of degradation* (Table 2)

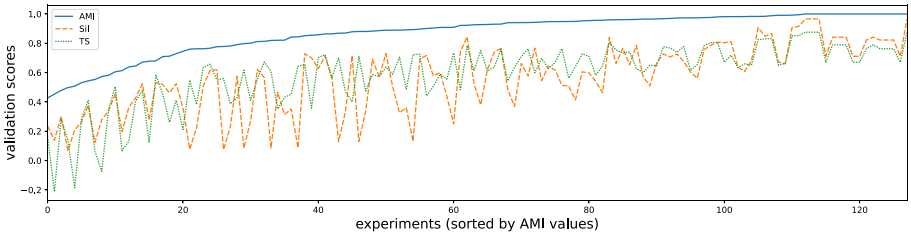

Fig. 7 AMI, sil and TS scores for the evaluation in stationary environments experiments. Scores are sorted based on AMI values (ascending)

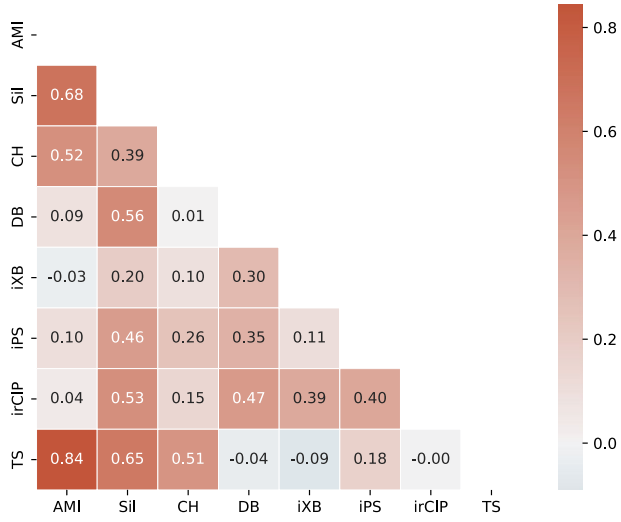

Fig. 8 Correlation among validation indices in the evaluation in stationary environments experiments

Sects. 1 and 2.1, since incremental indices are closer to change detectors rather than to validation indices per se. Note that, among the perturbations implemented in this section, only slicing actually introduces a significant temporal change.

TS and Sil indices are the most stable and correlated with the external reference index AMI, followed by CH and DB. To better visualize the correlation among AMI, Sil and TS,

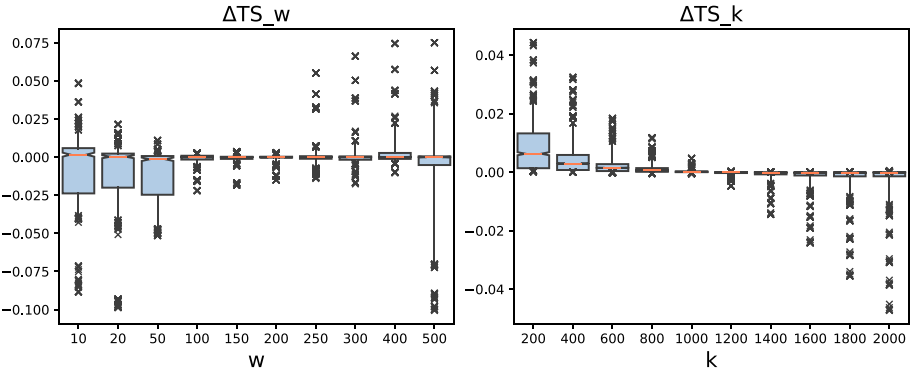

Fig. 9 Variations of TS scores for different values of w and k. Boxplot whiskers are set between the 0.5 and 0.95 quantiles

and, at the same time, assess the dynamical range of TS values, in Fig. 7 we sorted scores obtained from all 128 experiments (4 datasets, 4 perturbation-types, 8 degradation-levels) by taking AMI as reference. Additionally, Fig. 8 shows linear correlation coefficients among all studied indices (after inverting DB, iXB and irCIP).

Overall, experiments conducted in this section show that, when evaluating streaming data in stationary cases, TS has a stable behavior and is the closest to the external AMI benchmark. On the other hand, it largely outperforms DB and incremental options: iXB, iPS, irCIP.

5.5 Sensitivity of parameters

In Sect. 4 we anticipated the robustness of TS parameters. In this section, we perform a comprehensive sensitivity analysis by repeating the same experiments with different values of w and k. We omit ς since it is a parameter designed to be sensitive and give a greater or lesser penalty for temporal incoherence; i.e, variation in sigma induces strong variation in TS scores.

We evaluate values in W and K sets:

$$W = \{10, 20, 50, 100, 150, 200, 250, 300, 400, 500\}$$
$$K = \{200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000\}$$

Given a dataset, a perturbation-type and a degradation-level, we calculate TS for each combination of $w \in W$ and $k \in K$, obtaining 100 different TS scores. We then fix w and calculate the drift of TS respect the mean for the different values of k. We do the same by fixing k for the different values of w. Hence, ΔTS is the signed deviation from the mean of each parameter value with regard of all tested values for the same parameter.

Figure 9 shows boxplots with ΔTS for the different values of w and k. Each boxplot is calculated over 1280 values. For instance, when setting $k = 50$, this value is used 1280 times: 4 (datasets) $\times$ 4 (perturbation-types) $\times$ 8 (degradation-levels) $\times$ 10 (values of w) times. The same for each w in W and k in K.

Figure 9 shows negligible variations, which are slightly larger at the extreme values of the studied ranges, thus confirming the robustness of w and k in the experiments described.

w and k are parameters to be adjusted according to the expected and desired dynamics, and are easily adjustable with a minimum pre-knowledge of the scenario under study; i.e., while w is linked to the expected inertia of clusters, k determines the neighborhood around which each data point is evaluated. If clusters move over time, depending on their rate of change, variations in w will have a larger impact on *TS*.

6 Evaluation in concept drift

In this section we evaluate TS when facing scenarios submitted to concept drift. In contrast to Sect. 5, these experiments are not performed by corrupting labels; instead, we compare the performances of established stream clustering algorithms, thus also exploring their ability to cope with concept drift. Again, experiments and codes are available in our repository (TUWien - CN Group, 2023) and in a DOI-citable version for reproducibility in (Iglesias Vázquez, 2023).

6.1 Methodology

Experiments in this section conform to:

1. **Data** consists of 200 datasets generated to implement temporal challenges introduced in Sect. 3.
2. **Stream Clustering Algorithms** process data. We use four consolidated algorithms and the GT, which is added in the comparison as a *perfect clustering* competitor.
3. **Validation Indices.** All clustering solutions ($200 \times 5 = 1000$) are validated with TS, three CVIs and three iCVIs. Additionally, the external AMI index is provided to obtain a benchmark context-validation based on the GT partition.
4. **Finding the Best Validation.** The more reliable the validation method, the more often it will prefer the *perfect clustering* given by the GT rather than clusterings discovered by algorithms. We show such comparison as a whole and broken down by set of datasets. We also show comparisons without the GT *perfect clustering* by tentatively considering the best AMI as the best clustering (which might be biased or unreliable in cases with close scores).

6.2 Data

Data is taken from *Data for Evaluation of Stream Data Analysis Algorithms* (Iglesias, 2021). This collection of datasets are formed by 9 sets of 20 datasets, generated with MDCStream (Iglesias et al., 2020a) and designed to fit different challenges related to concept drift, outliers and data geometries. Our experiments are conducted on the sets that address the types of concept drift introduced in Sect. 3, plus the *base* set, which is taken as a baseline. The sets are:

– *Base*. These datasets have between 3 and 30 dimensions and contain from 2 to 10 clusters with different cardinality. The point appearance of each cluster shows a stable high frequency, making them stationary. Outliers are below 5%. Cluster centroids do not move over time and there is no cluster overlap. Two different distributions and distribution coefficients are allowed to set point locations in spatial dimensions. This fact,

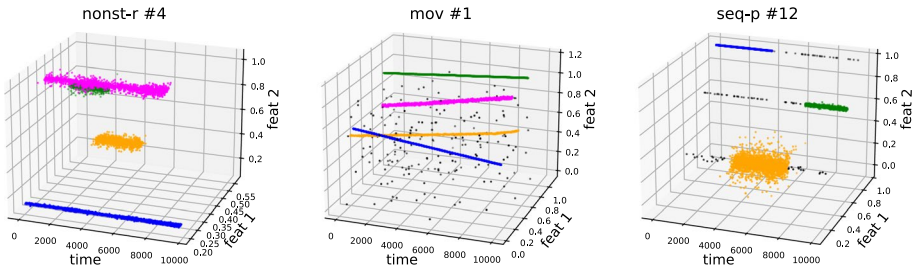

Fig. 10 Plots of two features over time for three random datasets used in the experiments. Data points take colors depending on the cluster to which they belong (outliers are shown in black). Note that plot resolution may give a false sense of distance between data points (Color figure online)

together with the diverse cluster cardinality and the stationary condition, offers a considerable variation of spatio-temporal cluster densities.

– *Nonstationary*. While keeping the same configuration as *base* datasets, *nonstationary* scenarios allow clusters randomly appear, disappear, coexist and reappear. From the types of concept drift above, these datasets satisfy the cases of *sudden drift*, *gradual drift* and *reoccurring contexts*.
– *Moving*. In these datasets cluster centroids are forced to move linearly over time, thus implementing *incremental drift*. Other characteristics are as in the *base* set.
– *Sequential*. Compared to the *nonstationary* set, here clusters do not coexist, so they are forced to appear sequentially. This set focuses exclusively on the *sudden drift* type.

All datasets—as retrieved from the original source—are formed by 10000 data points, which include spatial outliers (extreme values and local outliers). To test the effect of both types of outliers discussed in Sect. 3, we created dataset versions by removing outliers, and versions in which 1% of the data points were transformed into temporal outliers after removing spatial outliers. Injecting out-of-phase outliers is only possible in cases with temporal gaps before, after or in between clusters, i.e., *nonstationary* and *sequential* sets. Therefore, our experiments run over 10 sets of datasets, namely:

1. *Base* (`base`).
2. *Base*, spatial outliers removed (`base-r`).
3. *Nonstationary* (`nonst`).
4. *Nonstationary*, spatial outliers removed (`nonst-r`).
5. *Nonstationary*, spatial outliers removed, 1% of temporal outliers (`nonst-p`).
6. *Moving* (`mov`).
7. *Moving*, spatial outliers removed (`mov-r`).
8. *Sequential* (`seq`).
9. *Sequential*, spatial outliers removed (`seq-r`).
10. *Sequential*, spatial outliers removed, 1% of temporal outliers (`seq-p`).

Figure 10 shows three random datasets. Given the difficulties to plot multidimentional time-evolving spaces, it only draws two spatial dimensions and time.

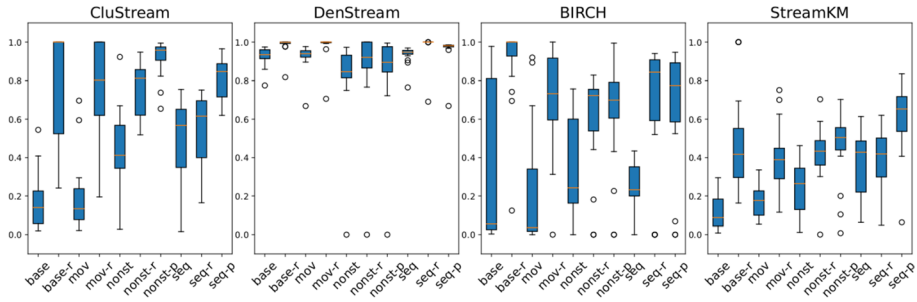

Fig. 11 Boxplots showing AMI performances of the streaming clustering used in the experiments. '1' means *perfect clustering*

6.3 Stream clustering algorithms

Stream clustering algorithms used in the experiments are:

1. *CluStream* (Aggarwal et al., 2003) from the ClusOpt library (Oliveira, 2020). Hyperparameters are: the number of micro-clusters m is 10 times the number of clusters, the horizon h is set to 5000 data points, whereas the radius t is 2. The version used does not discriminate outliers.
2. *DenStream* (Cao et al., 2006) from the implementation in the DenStream repository (Memari, 2020). Hyperparameters are: DBSCAN epsilon, $eps = 0.2$; the forgetting factor, $lambd = 0.1$, the outlier factor, $beta = 0.2$, and the weight of data points to define a core cluster, $mu = 11$.
3. *Birch* (Zhang et al., 1996) from the implementation in the scikit-learn 1.1.1 library (Pedregosa et al., 2011). Hyperparameters are: threshold radius for merging subclusters, $th = 0.5$.
4. *StreamKM++* (Ackermann et al., 2012) from the the ClusOpt library (Oliveira, 2020). Hyperparameters are: the coreset-size is 10 times the number of clusters and the length of the window is set to 5000 data points. The version used does not natively discriminate outliers.

The *GT* is added as a fifth algorithm performing *perfect clustering*. Algorithms are fed with chunks of 200 data points. With all datasets having a total of 10000 data points equispaced one time unit, this results in 500 chunks. Except for DenStream, other algorithms take the *expected* number of clusters as an external parameter.

6.4 Validation indices

We use the same validation índices as in Sect. 5.3. TS is adjusted with the default parameters: $w = 100$, $k = 1000$, $\varsigma = 1$.

Table 3 On the left, matches for each index rating *perfect clustering* (GT) as the best. On the right, once removed the GT from the comparison, matches for each index rating the clustering with the highest AMI (*best suboptimal solution*) as the best

	GT as ref.							*best AMI as ref.*						
	Sil	CH	DB	iXB	iPS	irCIP	TS	Sil	CH	DB	iXB	iPS	irCIP	TS
base	20	12	3	11	4	3	20	20	19	16	16	12	14	20
base-r	20	18	20	20	16	14	20	20	20	18	17	14	14	20
mov	20	9	1	10	3	5	20	18	20	18	18	12	14	20
mov-r	20	19	16	18	11	14	20	19	19	18	17	8	12	20
nonst	20	13	3	11	4	4	20	20	20	17	15	14	14	18
nonst-r	20	17	20	18	13	13	20	20	20	15	10	8	10	18
nonst-p	6	8	1	12	0	0	17	14	16	13	8	6	6	15
seq	20	18	0	5	2	0	20	20	20	20	18	17	18	20
seq-r	20	20	20	17	10	15	20	20	20	19	16	9	14	20
seq-p	4	2	0	3	3	3	20	17	18	17	16	15	17	17
all	170	136	84	125	66	71	**197**	188	**192**	171	151	115	133	188

6.5 Results and discussion

Figure 11 shows a boxplot summary with the performances of the different algorithms used. This evaluation applies the external metric AMI, for which the value '1' indicates perfect clustering, i.e., an exact match with the GT partition. In the figure we see that algorithms show dissimilar behaviors depending on the dataset type. Although DenStream is not inputted with the number of clusters to discover, it obtains the best overall clustering, slightly dropping performances in the *nonstationary* sets. In fact, in streaming environments, due to concept drift, externally setting the number of clusters might be a drawback for poorly adaptive algorithms, since some clusters may be absent during several batches. The *base* set without outliers (`base-r`) and *sequential* datatsets (i.e., no more than one cluster at a time) tend to be the best solved in general. While it is true that all algorithms work better in the absence of outliers (`-r` cases), in BIRCH and CluStream such effect is stronger. Out-of-phase outliers have a minor impact in performances, and even improve them in some cases. Note that these outliers only account for 1% of the data; also, when they improve clustering, it is because they help keep main clusters in memory, even though they are incorrectly absorbed by them. StreamKM obtains the worst performances as it tends to arbitrarily forget and reassign clusters as well splitting them over time into different subclusters.

Table 3 shows the comparison among CVIs and iCVIs. Sil and TS get the highest scores by correctly highlighting perfect clustering over suboptimal clustering, with TS clearly standing out. As expected based on previous work, e.g., (Liu et al., 2010), DB and CH indices are affected by outliers, particularly DB.

Also, iCVIs obtain significantly worse performances than CVIs since they fail to capture an overall perspective of the whole clustering. Among them, iXB stands out as the best option. This is consistent with the recommendations given by Moshtaghi et al. (2019), yet Brito Da Silva et al. (2020) find iXB less informative than other iCVIs. On the other hand, except for TS, all validation methods are confused in cases with out-of-phase outliers, since they fail to see them and do not penalize the clustering. In the tests, the most commonly

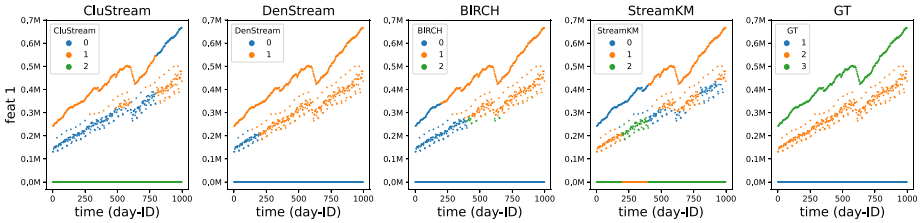

Fig. 12 Evolution of clusters in the retail sales collection dataset according to the studied stream clustering algorithms and the GT (Color figure online)

Table 4 Validation scores of the clustering performances for the retail sales collection dataset. Arrows in CVIs indicate whether the best performance corresponds to the highest or lowest index value

algorithm	AMI ↑	Sil ↑	CH ↑	DB ↓	iXB ↓	iPS ↑	irCIP ↓	TS ↑
CluStream	0.66	0.48	1649.56	1.94	0.21	1.80	0.65	0.79
DenStream	0.57	**0.70**	**3457.57**	**0.38**	**0.09**	1.68	**0.13**	0.57
BIRCH	0.34	0.21	1237.54	0.61	0.48	0.72	1.19	0.32
StreamKM	0.31	0.17	485.94	10.51	68916.33	1.26	1.80	0.36
GT	**1.00**	0.48	2630.33	0.80	0.12	**2.00**	0.63	**0.90**

Bold values show the clustering solution evaluated as "the best" by each validation index (column)

observed time disturbance was making algorithms chop a single cluster over time into false sub-clusters i.e., *slicing*. This effect is easily detectable by non-time-sensitive validation (i.e., CVIs), since it is processed as a spatial overlap. Even in the particular case of moving clusters, these are seen as elongated clusters that—features being multidimensional—very rarely intersect. These are the reasons why Sil still obtains good scores. When GT is not used as a benchmark, it is often difficult to establish best clustering. The better match between CH and the best AMI compared to TS (192 vs 188) is because TS penalizes spatial incongruity more than temporal incongruity (since it assumes that clusters can evolve), while other CVIs do the opposite (AMI is also not sensitive to the instant at which data points are compared).

7 Examples with real data

To better understand the usefulness of TS and its suitability with respect to traditional CVIs and iCVIs, we show two examples with real data that allow visual validation.

7.1 Retail sales collection

The Retail and Retailers Sales Time Series Collection dataset belongs to the U.S. Census Bureau and is maintained and updated by the Federal Reserve Economic Database (FRED). It is publicly available in the kaggle repository.[4] The collection consists of 23 one-dimensional timeseries, from which we select the RETAILIMSA, RETAILIRSA, and

[4] https://www.kaggle.com/datasets/census/retail-and-retailers-sales-time-series-collection.

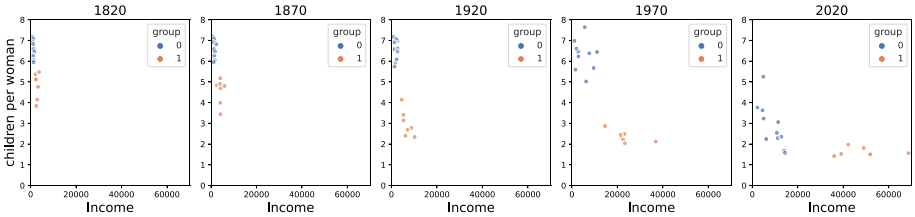

Fig. 13 Captures of GDP per capita vs children per woman in 18 countries for different years. European (orange) and non-European (blue) countries are separated by colors (Color figure online)

Table 5 Validation scores of the clustering performances for the fertility vs income dataset. Arrows in CVIs/iCVIs indicate whether the best performance corresponds to the highest or lowest index value

algorithm	AMI ↑	Sil ↑	CH ↑	DB ↓	iXB ↓	iPS ↑	rCIP ↓	TS ↑
CluStream	0.92	**0.43**	**3154.06**	**0.92**	0.10	1.57	1.51	0.75
DenStream	0.40	−0.01	2410.79	6.42	17.19	0.33	4.20	0.49
BIRCH	0.18	−0.09	311.20	1.87	0.73	0.45	4.21	0.38
StreamKM	0.11	0.12	634.12	2.07	5.55	**1.72**	9.78	0.53
GT	**1.00**	0.42	3032.29	0.94	**0.10**	1.60	**1.23**	**0.77**

Bold values show the clustering solution evaluated as "the best" by each validation index (column)

RETAILSMNSA to evaluate the performance of the stream clustering algorithms and the CVIs/iCVIs. We joined the three series into a single series and ordered data points based on the timestamp, obtaining a final dataset of size 1x999 with three clusters. Selected algorithms are expected to differentiate the three series. Evaluation methods are the same as in Sect. 5 and Sect. 6, while algorithm parameterizations were adjusted to optimize clustering.

Figure 12 and Table 4 show the performances of algorithms and CVIs/iCVIs respectively. In Fig. 12 we clearly see how established algorithms for stream clustering have problems adapting to the temporal displacement of clusters even in very simple scenarios. On the other hand, the commonly used CVIs and iCVIs are not able to differentiate the best solution either, showing a tendency to support the solution obtained by Denstream. Only TS and iPS identify GT as the best solution.

7.2 Fertility vs income

The Fertility vs Income dataset consists of yearly data from 1800 to 2022 related to the average babies per woman and the GDP (Gross Domestic Product) per capita in constant PPP dollars of a set of world countries. Data used to create this dataset belongs to the Gapminder data repository.[5] Gapminder is an independent foundation whose mission is to identify systematic misconceptions about relevant global trends through the collection and open publication of data from reliable sources, e.g., the UN (United Nations), WPP (World Population Prospects), the World Bank, the Maddison Project Database.

Among all available countries, we selected 18 that can be split into two groups with clearly distinguishable trends, i.e., 11 non-European (blue) and 7-European (orange)

[5] https://www.gapminder.org/data/

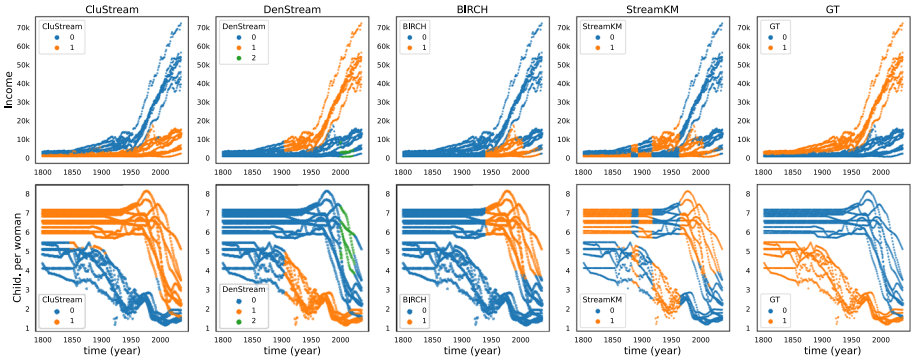

Fig. 14 Evolution of clusters in the fertility vs income dataset according to the studied stream clustering algorithms and the GT. Upper plots show income vs time, while lower plots show children per woman vs time (Color figure online)

countries. Thus, the dataset contains 4014 two-dimensional data points, timestamped with the year and identified in the two mentioned classes. Figure 13 shows five snapshots corresponding to 1820, 1870, 1920, 1970 and 2020. These plots reveal how, regardless of their respective evolution, both groups appear separated over time. Therefore, stream clustering algorithms are expected to fully or partially match the labeling. As in the previous example, we tuned algorithms to obtain optimal clustering.

Figure 14 and Table 5 show the performances of algorithms and CVIs/iCVIs. This time CluStream was able to infer consistent clusters, significantly close to the GT and the human intuition. Other stream clustering algorithms show partitions with different kinds of deviation. Note that traditional, static CVIs tend to consider CluStream as the best clustering, whereas the majority of iCVIs and TS correctly prefer the clustering given by the GT.

8 Conclusions

We have introduced the Temporal Silhouette index, an internal validation method for stream clustering designed to provide reliable evaluations even in the event of concept drift. Our index has shown satisfactory validation when comparing two real-life cases, four datasets for clustering evaluation submitted to 32 different forms-levels of degradation, and 200 scenarios to implement the different types of concept drift identified in the literature, as well as cases with spatial and temporal outliers. As a general rule, Temporal Silhouette showed better performances than both traditional and incremental validation.

The Temporal Silhouette index fills a gap within the methods for the internal validation of unsupervised algorithms in streaming data analysis, which to date is performed either with static validation, opaque to concept changes, or with incremental approaches, which are closer to techniques for the online detection of context changes.

Author Contributions Conceptualization, Methodology, Formal analysis and Investigation, and Writing - Original Draft Preparation: FIV; Writing - Review and Editing, Resources and Supervision: TZ.

Funding Open access funding provided by TU Wien (TUW). The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme.

Data availability All experiments, data and codes used in the paper are available for reuse and replication in a Figshare repository with a DOI-citable version: Iglesias Vázquez, F. (2023). Temporal Silhouette for Stream Clustering Validation - Evaluation Tests (2.0.0). TU Wien. https://doi.org/10.48436/ss6a3-3r720

Code availability All experiments, data and codes are also available in an open Github repository updated and maintained by the authors in: TUWien - CN Group (2023). Temporal Silhouette (Python). URL: https://github.com/CN-TU/py-temporal-silhouette.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethical approval Not applicable.

Consent to participation Not applicable.

Consent for publication Not applicable.

References

Ackermann, M. R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., & Sohler, C. (2012). Streamkm++: A clustering algorithm for data streams. *ACM J Exp Algorithmics, 17*, 1–2.

Aggarwal, C.C., Han, J., Wang, J., & Yu, P.S. (2003). A framework for clustering evolving data streams. In Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29, VLDB Endowment, VLDB '03, p 81–92.

Aggarwal, C.C., Han, J., Wang, J., & Yu, P.S. (2007). On clustering massive data streams: A summarization paradigm. In Data Streams, Springer, pp 9–38.

Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Pérez, J. M., & Perona, I. (2013). An extensive comparative study of cluster validity indices. *Pattern Recognition, 46*(1), 243–256.

Bezdek, J. C., & Keller, J. M. (2021). Streaming data analysis: Clustering or classification? *IEEE Trans on Systems, Man, and Cybernetics: Systems, 51*(1), 91–102.

Brito Da Silva, L. E., Melton, N. M., & Wunsch, D. C. (2020). Incremental cluster validity indices for online learning of hard partitions: Extensions and comparative study. *IEEE Access, 8*, 22025–22047.

Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics, 3*(1), 1–27.

Cao, F., Estert, M., Qian, W., & Zhou, A. (2006). Density-Based Clustering over an Evolving Data Stream with Noise, pp 328–339.

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI, 1*(2), 224–227.

Ester, M., Kriegel, H.P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, KDD'96, pp. 226–231

Fisch, A. T. M., Eckley, I. A., & Fearnhead, P. (2022). A linear time method for the detection of collective and point anomalies. *Statistical Analysis and Data Mining: The ASA Data Science Journal, 15*(4), 494–508.

Fränti, P., & Sieranoja, S. (2018). K-means properties on six clustering benchmark datasets. *Applied Intelligence, 48*(12), 4743–4759.
Fränti, P., & Virmajoki, O. (2006). Iterative shrinking method for clustering problems. *Pattern Recognition, 39*(5), 761–765.
Fränti, P., Virmajoki, O., & Hautamäki, V. (2006). Fast agglomerative clustering using a k-nearest neighbor graph. *IEEE Trans on Pattern Analysis and Machine Intelligence, 28*(11), 1875–1881.
Gama, J., Zliobaite, I., Bifet, A., & Pechenizkiy, M. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR), 46*(4), 1–37.
Giraud-Carrier, C. (2000). A note on the utility of incremental learning. *AI Communications, 13*(4), 215–223.
Hassani, M., & Seidl, T. (2017). Using internal evaluation measures to validate the quality of diverse stream clustering algorithms. *Vietnam Journal of Computer Science, 4*(3), 171–183.
Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification, 2*(1), 193–218.
Ibrahim, O.A., Keller, J.M., & Bezdek, J.C. (2018). Analysis of streaming clustering using an incremental validity index. In IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–8.
Ibrahim, O.A., Keller, J.M., & Popescu, M. (2019). A new incremental cluster validity index for streaming clustering analysis. In IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–8.
Iglesias, F. (2021). Data for evaluation of stream data analysis algorithms. *Mendeley Data*. https://doi.org/10.17632/c43kr4t7h8.1
Iglesias, F., Zseby, T., Ferreira, D., & Zimek, A. (2019). Mdcgen: Multidimensional dataset generator for clustering. *Jour of Classification, 36*(3), 599–618.
Iglesias, F., Ojdanic, D., Hartl, A., & Zseby, T. (2020a). Mdcstream: Stream data generator for testing analysis algorithms. In Proceedings of the 13th EAI International Conference on Performance Evaluation Methodologies and Tools, Association for Computing Machinery, New York, NY, USA, VALUETOOLS '20, pp. 56–63.
Iglesias, F., Zseby, T., & Zimek, A. (2020). Absolute cluster validity. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42*(9), 2096–2112.
Iglesias Vázquez, F. (2023). Temporal Silhouette for Stream Clustering Validation - Evaluation Tests (2.0.0) https://doi.org/10.48436/ss6a3-3r720, tU Wien
Kremer, H., Kranen, P., Jansen, T., Seidl, T., Bifet, A., Holmes, G., & Pfahringer, B. (2011). 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, Association for Computing Machinery, New York, NY, USA, KDD '11, pp. 868–876.
Kuncheva, L. I. (2013). Change detection in streaming multivariate data using likelihood detectors. *IEEE Transactions on Knowledge and Data Engineering, 25*(5), 1175–1180.
Leys, C., Ley, C., Klein, O., Bernard, P., & Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology, 49*, 764–766.
Liu, Y., Li, Z., Xiong, H., Gao, X., & Wu, J. (2010). Understanding of internal clustering validation measures. *IEEE International Conference on Data Mining* (pp. 911–916). New Jersey: IEEE.
von Luxburg, U., Williamson, R.C., & Guyon, I. (2012). Clustering: Science or art? In Guyon I, Dror G, Lemaire V, Taylor G, Silver D (eds) Proceedings of ICML Workshop on Unsupervised and Transfer Learning, PMLR, Bellevue, Washington, USA, Proceedings of Machine Learning Research, vol 27, pp. 65–79.
Memari, I. (2020). DenStream (Python). https://github.com/issamemari/DenStream, GitHub repository (Accessed on Jun, 2022).
Moreno-Torres, J. G., Raeder, T., Alaiz-RodríGuez, R., Chawla, N. V., & Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern Recogn, 45*(1), 521–530.
Moshtaghi, M., Bezdek, J. C., Erfani, S. M., Leckie, C., & Bailey, J. (2019). Online cluster validity indices for performance monitoring of streaming data clustering. *International Journal of Intelligent Systems, 34*(4), 541–563.
Nguyen, H. L., Woon, Y. K., & Ng, W. K. (2015). A survey on data stream clustering and classification. *Knowledge and Information Systems, 45*(3), 535–569.
Nordahl, C., Boeva, V., Grahn, H., & Persson Netz, M. (2021). Evolvecluster: An evolutionary clustering algorithm for streaming data. Evolving Systems pp. 1–21.
Oliveira, G. (2020). ClusOpt Core (Python). https://github.com/giuliano-oliveira/clusopt_core, GitHub repository (Accessed on Jun, 2022).
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot,

M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research, 12*, 2825–2830.
Rezaei, M., & Fränti, P. (2020). Can the number of clusters be determined by external indices? *IEEE Access, 8*, 89239–89257.
Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics, 20*, 53–65.
Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., Dietterich, T. G., & Müller, K. R. (2021). A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE, 109*(5), 756–795.
Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., Carvalho, A. C. P. LFd., & Ja, Gama. (2013). Data stream clustering: A survey. *ACM Computing Surveys, 46*(1), 1–31.
TUWien - CN Group. (2023). Temporal Silhouette (Python). https://github.com/CN-TU/py-temporal-silhouette, GitHub repository.
Vinh, N.X., Epps, J., Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance 11:2837–2854.
Xie, X. L., & Beni, G. (1991). A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 13*(8), 841–847.
Zhang, T., Ramakrishnan, R., Livny, M. (1996). Birch: An efficient data clustering method for very large databases. Association for Computing Machinery, New York, NY, USA, SIGMOD '96, pp. 103–114.
Zubaroğlu, A., & Atalay, V. (2022). Online embedding and clustering of evolving data streams. *Statistical Analysis and Data Mining: The ASA Data Science Journal, 16*(1), 29–44.