Evolving Systems (2025) 16:82
https://doi.org/10.1007/s12530-025-09713-w

ORIGINAL PAPER

Evolving interval-based time series clustering for streaming industrial data

Žiga Stržinar[1,2] · Igor Škrjanc[2] · Boštjan Pregelj[1]

Received: 23 December 2024 / Accepted: 12 June 2025 / Published online: 28 June 2025

Abstract
Accurate clustering of time series data is crucial for extracting meaningful insights from streaming sensor data in industrial applications. To address the challenges of dynamic and unlabeled data streams, we introduce Interval ERAL (iERAL), an enhancement of the Error in Aligned Series (ERAL) framework. iERAL is a time series alignment and averaging method designed for online analysis, incorporating an interval band to represent variance in the underlying data. We pair iERAL with an evolving time series clustering algorithm, capable of automatically detecting, adapting to, and merging clusters in real-time. This evolving approach enables the algorithm to dynamically adjust to new patterns, promote or demote clusters based on their relevance, and handle data variability with interval-based analysis. Unlike previous methods, our approach not only computes the time series prototype for each cluster but also provides a variance band for interval-based analysis. We demonstrate the effectiveness of our method by applying it to line pressure measurements in a real-world industrial setting. The algorithm achieves promising results in clustering unlabeled data streams, highlighting its potential for anomaly detection and adaptive monitoring of industrial processes in evolving operating conditions.

Keywords Evolving clustering · Time series clustering · Time series prototype · Unsupervised learning · Industrial application

1 Introduction

The significance of time series analysis has grown with the increasing accessibility of sensor data across fields such as automotive, manufacturing, healthcare, and finance. Unlike many other types of data, the sequence of observations in time series data is critical. Consequently, specialized analytical approaches have been developed to respect this temporal structure.

Time series analysis encompasses several sub-disciplines, each addressing specific challenges. The most prominent include time series forecasting, clustering, and classification. Forecasting (Masini et al. 2023) predicts future values based on historical patterns. Clustering methods (Aghabozorgi et al. 2015; Paparrizos and Gravano 2015) group similar time series, while classification techniques (Bagnall et al. 2017; Dau et al. 2018; Abanda et al. 2019) assign time series to predefined categories. These approaches rely on algorithms tailored to the complexity and temporal nature of time-dependent data.

Time series clustering is applied when no labeled examples are available, and the objective is to identify natural groupings among time series. This technique has diverse applications, such as clustering stock market time series to aid portfolio diversification (Lim and Ong 2021; Shirota and Murakami 2021) and distinguishing power system events using clustering methods (Bariya et al. 2021).

Conventional clustering methods often assume access to a static dataset; however, real-world applications frequently require handling continuously streaming data. In such cases, a single-pass, online algorithm offers significant advantages

✉ Žiga Stržinar
ziga.strzinar@ijs.si

Igor Škrjanc
igor.skrjanc@fe.uni-lj.si

Boštjan Pregelj
bostjan.pregelj@ijs.si

[1] Department of Systems and Control, "Jožef Stefan" Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia

[2] Laboratory of Control Systems and Cybernetics, Faculty of Electrical Engineering, University of Ljubljana, Tržaška cesta 25, 1000 Ljubljana, Slovenia

(Škrjanc et al. 2019; Andonovski et al. 2018; Leite et al. 2020; Antić et al. 2021; Škrjanc et al. 2022, 2018; Blažič et al. 2014; Singh et al. 2023). Additionally, an evolving model capable of updating itself as new data arrives, can adapt to changing data distributions over time.

An evolving time series clustering approach is highly valuable in manufacturing environments for continuous monitoring of machine health. As sensor data is collected in real time, the model can dynamically cluster incoming time series to uncover patterns and identify anomalies before they escalate into equipment failures. By updating itself, the model remains effective as operating conditions change, such as by introducing new clusters for emerging patterns or adjusting existing clusters to reflect altered data distributions.

The present paper contributes to the time series analysis literature by proposing Interval ERAL (iERAL), a method for time series clustering that represents a cluster of time series using a single series – the cluster's prototype. Additionally, the paper introduces an evolving time series clustering algorithm that uses iERAL to process streaming data in a single pass. The algorithm can automatically determine the number of clusters required, add new clusters, and merge existing ones as the data evolves.

The paper is structured as follows. Section 2 presents related work on time series alignment and averaging methods, as well as time series clustering methods. Section 3 provides a detailed explanation of ERAL and sERAL, and introduces iERAL. Additionally, an evolving time series clustering algorithm, designed for use with iERAL, is described. Section 4 discusses experimental results, including a comparison of alignment and averaging methods, evolving clustering of unlabeled industrial data, and a comparison of evolving clustering results with ground truth labels. Section 5 analyses the results, and Sect. 6 concludes the paper and outlines future work.

2 Related work

A crucial component of many clustering algorithms is the calculation of a cluster prototype. Prototypes, or centroids, are representatives of clusters used in methods such as K-Means, K-Medoids, and Fuzzy C-Means. In time series clustering, the prototype is a time series that characterizes the cluster. Prototypes facilitate further analysis, including classification, anomaly detection, and visualization. They also help interpret clusters by revealing average behaviors or common patterns within the cluster.

2.1 Alignment and averaging methods

A set of time series analysis methods, known as alignment and averaging methods (Petitjean et al. 2011; Schultz and Jain 2018; Liu et al. 2023; Rasines et al. 2023; Morel et al. 2018; Liu et al. 2019; Stržinar et al. 2024), address the challenge of representing a set of time series. These methods map a set of time series $X = \{\underline{x_1}, \underline{x_2}, \ldots, \underline{x_n}\}$ to a single *prototype* time series.

The prototype time series $\underline{p}$ represents the set of time series X and is used for further analysis. Historically, these methods aimed to find the barycenter of the set using Dynamic Time Warping (DTW) Berndt and Clifford (1994) as the distance function. Early methods (Niennattrakul and Ratanamahatana 2009; Gupta et al. 1996) often performed poorly (Liu et al. 2023; Petitjean et al. 2011; Niennattrakul and Ratanamahatana 2007). However, many issues were alleviated by DTW Barycenter Averaging (DBA) Petitjean et al. (2011), which has become the standard for time series averaging. Recent methods primarily aim to improve DBA, focusing on algorithm speed or prototype quality. Notable advancements include the Stochastic Subgradient Method (SSG) Schultz and Jain (2018), ShapeDWA (Liu et al. 2023) and SoftDTW (Cuturi and Blondel 2017).

All the aforementioned methods are based on Dynamic Time Warping (DTW), a widely used distance function for time series. DTW aligns time series of different lengths and is robust to noise. It has been applied to classification (Dau et al. 2018; Stržinar et al. 2023), robotics in Rasines et al. (2023), seismology (Kumar et al. 2022), semiconductor manufacturing (He et al. 2018). It operates by finding the optimal alignment between two time series via the path with the lowest cost in a cost matrix, allowing local stretching and compressing of time series. While DTW is often criticized for its computational expense, various optimizations have been proposed to improve its efficiency (Ratanamahatana and Keogh 2004), Yi et al. (1998), Kim et al. (2001).

Despite the popularity of DBA, recent research highlights unintuitive solutions arising from DTW-based methods for alignment and averaging. For example, Liu et al. (2023); Rasines et al. (2023); Morel et al. (2018); Liu et al. (2019); Stržinar et al. (2024) demonstrates several cases where DTW-based methods fail to produce meaningful prototype time series, often resulting in artifacts such as phantom plateaus and spikes.

The limitations of DTW-based methods have led to the development of alternative approaches for alignment and averaging. One such method is Shape-based Distance (SBD) Paparrizos and Gravano (2015), which emphasizes the overall shape of the time series. Another example is Soft-DTW, which uses a differentiable approximation of DTW (Cuturi and Blondel 2017). Both methods have

demonstrated superior performance compared to DBA in specific scenarios.

2.2 ERAL and streaming ERAL

Recently, a method called Error in Aligned Series (ERAL) Stržinar et al. (2024) was introduced, based on optimal non-elastic alignment of time series. Applied to an industrial dataset (Stržinar et al. 2024), the method produced promising results, also validated using the UCR Archive (Dau et al. 2018). ERAL has been shown to outperform DBA and other DTW-based methods in speed while avoiding artifacts such as phantom plateaus or spikes, and capturing the underlying shape of the data.

ERAL employs an iterative process where a prototype candidate is repeatedly aligned with the dataset and incrementally adjusted. This process continues until the prototype converges. An error function called the ERAL score is used to compute the optimal alignment of each time series in the dataset and the current prototype candidate. The candidate is then updated based on the alignment results, with the process repeated for each iteration.

While ERAL generates high-quality prototypes, it is designed for batch processing, requiring the entire dataset to be processed before producing a prototype. This limitation is impractical in many clustering scenarios, where iterative updates are needed. To address this, Streaming ERAL (sERAL) Stržinar et al. (2024) was introduced. sERAL processes data incrementally, updating the prototype as new samples arrive. It produces prototypes similar to ERAL while enabling real-time data processing, making it suitable for online clustering applications.

sERAL introduces two internal structures: the Prototype Shape Vector (PSV) and the Prototype Confidence Vector (PCV). These structures accumulate information from time series processed in the past and can be used at any time to compute the prototype.

ERAL and sERAL form the foundation of our work in this paper. These methods will be discussed in detail in Sect. 3.

Table 1 provides a comparison of various alignment and averaging methods, highlighting their key features and performance differences.

2.3 Time series clustering

Clustering algorithms are fundamental tools in data analysis, enabling the exploration of unlabeled data. These algorithms are particularly useful in time series analysis for identifying patterns or grouping similar time series. Applications of time series clustering include training of coloborative robots (Rasines et al. 2023), semiconductor manufacturing and anomaly detection (He et al. 2018), portfolio diversification (Shirota and Murakami 2021).

A common approach to time series clustering is the use of K-Means or K-Medoids algorithms paired with some time-series-specific distance measure, for example Dynamic Time Warping (DTW) Berndt and Clifford (1994) or Move-Split-Merge (MSM) Stefan et al. (2012). This approach has been thoroughly explored in Holder et al. (2024). Further, general-purpose clustering algorithms, such as DBSCAN can be used in conjunction with a time series distance measure, such as DTW, and applied to time series datasets (Schubert et al. 2017).

A notable specialized time series clustering algorithm is K-Shape, introduced in Paparrizos and Gravano (2015). This algorithm clusters time series by their shape, using a distance function called Shape-based Distance (SBD), which employs cross-correlation to compare the shapes of two time series. K-Shape has been applied in various domains, including portfolio diversification (Shirota and Murakami 2021) and energy sector (Bariya et al. 2021). However, K-Shape has several limitations: it cannot process time series of varying lengths, operates as a batch algorithm, and cannot handle data in a single pass. Additionally, prototypes generated by SBD often exhibit edge artifacts (Stržinar et al. 2024).

To address the batch processing limitation of K-Shape, K-ShapeStream was developed (Bariya et al. 2021). This algorithm processes data in a single pass, updating prototypes incrementally as new samples arrive. However, the other limitations of K-Shape, such as edge artifacts and

Table 1 Comparison of time series alignment and averaging methods

Method	Type	Underlying distance	Time-warped prototypes	Key benefit
DBA Petitjean et al. (2011)	Batch	DTW	Yes	Optimized for DTW
SBD Paparrizos and Gravano (2015)	Batch	SBD	No	No spikes and plateaus
ERAL Stržinar et al. (2024)	Batch	ERAL score	No	No spikes and plateaus
sERAL Stržinar et al. (2024)	Single-pass	ERAL score	No	Adds single pass to ERAL
iERAL (This work)	Single-pass	ERAL score	No	Adds interval band

inability to handle varying time series lengths, remain unresolved.

Neither K-Shape nor K-ShapeStream can automatically determine the number of clusters required to represent the data. This limitation is common among clustering algorithms and is often addressed by experimenting with varying user-defined numbers of clusters and selecting the optimal configuration. However, this approach is impractical for streaming data, where the number of clusters may dynamically change over time.

Evolving algorithms, which can automatically adjust the model structure (e.g., the number of clusters), offer a flexible and robust solution. Despite their potential advantages, these algorithms have not been widely adopted in time series clustering, possibly due to the computational challenges or lack of specialized implementations.

In this paper, we present an Evolving Time Series Clustering algorithm capable of detecting the number of clusters, adding and merging clusters dynamically, and processing all data in a single pass. This is achieved using the ERAL-based alignment and averaging method described in Sect. 3.

3 Material and methods

In time series analysis, warping the time axis is common, typically achieved using Dynamic Time Warping (DTW) or other elastic distance functions. However, in certain situations–such as analyzing data from industrial processes–time axis warping may not be desirable, since even small deviations in signal shape can indicate a fault, an unintended process change, or another anomaly. Elastic approaches may obscure such deviations, making non-elastic distance functions preferable in these contexts.

Another critical aspect of time series analysis is segmenting continuous sensor outputs into discrete segments, a process known as segmentation. Segmentation is crucial for analyzing industrial processes, as it determines the level of detail downstream algorithms can capture. For instance, fine-grained segmentation may detect simple patterns but lose the broader context if downstream algorithms lack memory of prior segments. Conversely, coarse segmentation might limit process insights. Selecting the appropriate segmentation level is challenging, especially when segment boundaries are not clearly defined–such as during gradual signal changes–making boundary decisions difficult and potentially unreliable.

These two aspects–non-elastic alignment and segmentation–motivated the development of Error in Aligned Series (ERAL) Stržinar et al. (2024). ERAL is designed to minimize excessive warping of time series and provide robustness against non-ideal segmentation.

3.1 Error in aligned series – ERAL

Error in Aligned Series (ERAL) is an iterative method designed to compute the prototype of a set of time series. It optimizes the prototype without excessive warping, ensuring that the result effectively represents the dataset.

Throughout this work, underlined lowercase letters (e.g., $\underline{x}, \underline{x}_j$) denote univariate time series composed of scalar elements. Subscripts such as x_k refer to the k-th scalar value in the time series $\underline{x}$. Expressions like $x_{k-\tau}$ indicate index-shifted access to these scalar values and should not be interpreted as item separation or as multiple distinct variables.

Let $\underline{x}$ and $\underline{y}$ be time series of lengths N and M, respectively. The lengths of $\underline{x}$ and $\underline{y}$ may differ. Unlike some other methods that require equal-length time series, ERAL accommodates this difference. The time series are represented as:

$$\underline{x} = [x_0, x_1, \ldots, x_{N-1}] \tag{1}$$

$$\underline{y} = [y_0, y_1, \ldots, y_{M-1}] \tag{2}$$

$\underline{x}_\tau$ is a time series shifted by some $\tau \in \mathbb{Z}$, as shown in Eq. (3). An example is shown in the top plot of Fig. 1, where $\underline{y}$ is shifted by $\tau = 7$:

$$\underline{x}_\tau = [x_{0-\tau}, x_{1-\tau}, \ldots, x_{N-1-\tau}] \tag{3}$$

No assumptions are made regarding the relative values of N and M; however, in the following examples, we assume $N > M$.

ERAL is an iterative method where, in each iteration, a candidate prototype is aligned with the dataset. To achieve this alignment, the *ERAL score* is calculated for all combinations of the current prototype and the time series in the dataset.

For $\underline{x}$ and $\underline{y}$, the ERAL score is defined as:

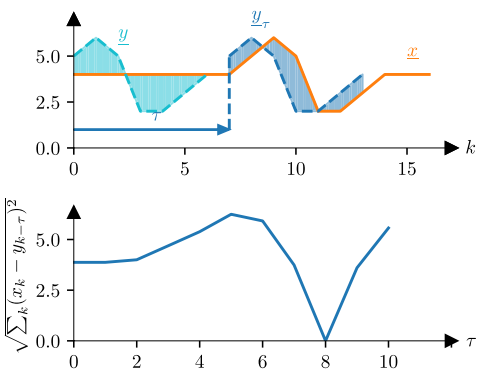


Fig. 1 Calculation of the ERAL score for two time series

$$\epsilon_{x,y}(\tau) = \nu_{x,y}(\tau)^{-1} \sqrt{\sum_k (x_k - y_{k-\tau})^2} \tag{4}$$

$$\nu_{x,y}(\tau) = \begin{cases} \sqrt{M - |\tau|} & \text{if } \tau < 0 \\ \sqrt{N - |\tau|} & \text{if } \tau > N - M \\ \sqrt{M} & \text{otherwise} \end{cases} \tag{5}$$

The ERAL score describes the error of aligning two time series at different lags and is computed for all valid τ:

$$\tau \in \left[-M + 1,\ N - 1 \right] \tag{6}$$

The example in Fig. 1 illustrates the ERAL score for two time series: $\underline{x} = [4, 4, 4, 4, 4, 4, 4, 5, 6, 5, 2, 2, 3, 4, 4, 4]$ and $\underline{y} = [5, 6, 5, 2, 2, 3, 4]$. A global minimum is observed at $\tau = 8$, where the two time series are optimally aligned.

The lag corresponding to the minimum ERAL score is called the optimal lag τ^*:

$$\tau^* = \arg\min_{\tau} \epsilon_{x,y}(\tau) \tag{7}$$

ERAL uses τ^* to align the prototype candidate with the dataset, ensuring that the candidate is optimally positioned for refinement.

$$\underline{x}_* = [x_{0-\tau^*}, x_{1-\tau^*}, \ldots, x_{N-1-\tau^*}] \tag{8}$$

In each iteration (indexed by j), a set of H time series $X = \{\underline{x_1}, \underline{x_2}, \ldots, \underline{x_H}\}$ is aligned to the current prototype candidate $\underline{p_j}$, resulting in $X_{*,j} = \{\underline{x_1}_{*,j}, \underline{x_2}_{*,j}, \ldots, \underline{x_H}_{*,j}\}$. These aligned time series are then used to update the prototype:

$$\underline{p_{j+1}} = \underline{p_j} + \lambda \sum_{i=1}^{H} w_{i,j} \cdot (\underline{x_i}_{*,j} - \underline{p_j}) \tag{9}$$

$$w_i = 1/\epsilon_{x_i,p}(\tau^*) \tag{10}$$

Figure 2 provides a simplified diagram of the ERAL process. The prototype candidate is aligned with the dataset, and an error vector is computed. The prototype candidate is adjusted using this error vector, and the process repeats over several iterations until the prototype converges. For complete details on ERAL, see Stržinar et al. (2024).

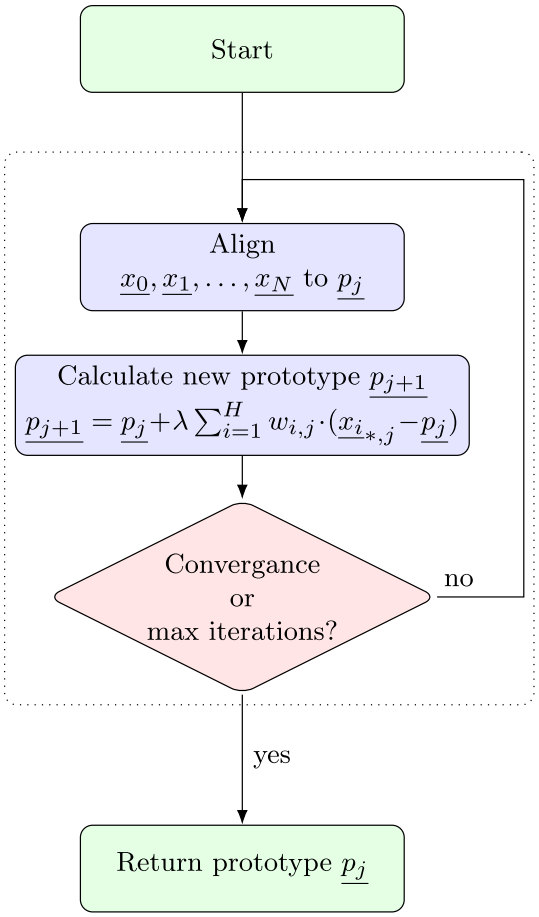


Fig. 2 Simplified diagram showing the ERAL process

3.2 Streaming ERAL - sERAL

While prototypes generated by ERAL are of high quality and avoid the issues seen in DBA, SSG, or other DTW-based methods, ERAL does not support online processing. This limitation poses challenges in real-world applications where data streams continuously and models must update in real-time. An example is monitoring industrial processes, where sensor data is collected and analyzed for anomalies in real time. Streaming ERAL (sERAL) Stržinar et al. (2024), described here, extends ERAL to support online processing.

With sERAL, two internal structures are introduced: the Prototype Shape Vector (PSV) and the Prototype Confidence Vector (PCV). Together, these structures store information about past time series and can be used to calculate the prototype at any time.

- **PSV**: A vector containing the average shape of all processed time series. Each incoming time series is aligned to the current prototype before updating the PSV. As new time series arrive, PSV evolves by expanding left or right to accommodate the aligned input series. The exact mechanism for calculating PSV will be detailed in Sect. 3.2.1.
- **PCV**: A vector of the same length as PSV, representing the proportion of past inputs used for each point in

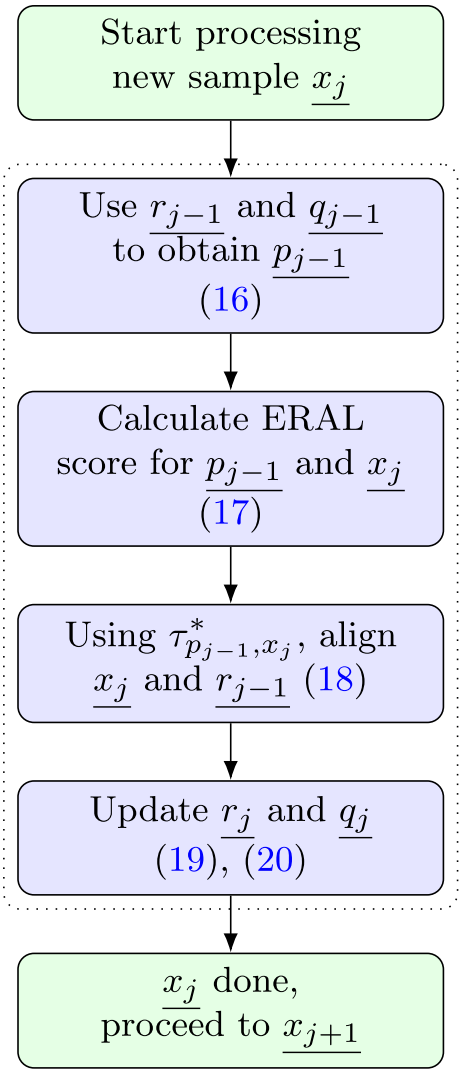


Fig. 3 Diagram illustrating the calculation of the sERAL prototype

PSV. This adjustment accounts for varying time series lengths (e.g., a short sample only updates a portion of PSV, increasing PCV in that region). PCV also reflects partial overlaps caused by non-zero alignments.

- **Prototype**: A continuous subset of PSV where PCV exceeds a certain threshold, which is a hyperparameter of the model. Based on our experience, a threshold of $50-80\%$ is a suitable starting point.

The concept of sERAL is illustrated in Fig. 3. Each incoming time series is aligned with the current prototype, and the PSV and PCV are updated accordingly. This process is repeated for every incoming time series.

3.2.1 Calculating PSV and PCV

Let PSV in the j-th iteration be $\underline{r_j}$, and PCV be $\underline{q_j}$. Similarly, the prototype is then $\underline{p_j}$:

$$\underline{r_j} = [r_{j,0}, r_{j,1}, \dots, r_{j,R_j-1}] \tag{11}$$

$$\underline{q_j} = [q_{j,0}, q_{j,1}, \dots, q_{j,R_j-1}] \tag{12}$$

$$\underline{p_j} = [r_{j,k}, r_{j,k+1}, \dots, r_{j,k+P_j}], \quad \text{where } 0 \le k \le R_j - P_j \tag{13}$$

$$i_{j,\min} = \min\{i \mid q_{j,i} > \alpha\} \tag{14}$$

$$i_{j,\max} = \max\{i \mid q_{j,i} > \alpha\} \tag{15}$$

$i_{j,\min}$ is analogous to k in Eq. (13), and $P_j = i_{j,\max} - i_{j,\min}$, resulting in:

$$\underline{p_j} = [r_{j,i} \mid i_{j,\min} \le i \le i_{j,\max}] \tag{16}$$

When a sample $\underline{x_j}$ is passed to sERAL, the ERAL score is calculated for $\underline{x_j}$ and the prototype from the previous iteration $\underline{p_{j-1}}$:

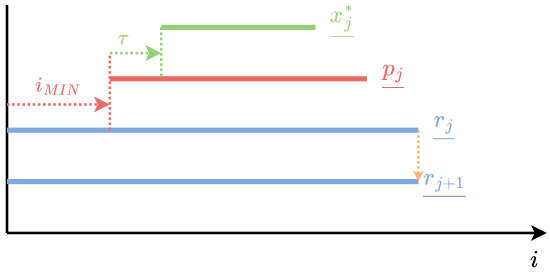
$$\epsilon_{x_j,p_{j-1}}(\tau) = \nu^{-1}_{x_j,p_{j-1}}(\tau)\sqrt{\sum_k (x_{j,k} - p_{j-1,k-\tau})^2} \tag{17}$$

Equations (5) and (6) are used for ν and the range of The resulting ERAL score $\epsilon_{x_j,p_{j-1}}(\tau)$ is then used to determine the optimal alignment:

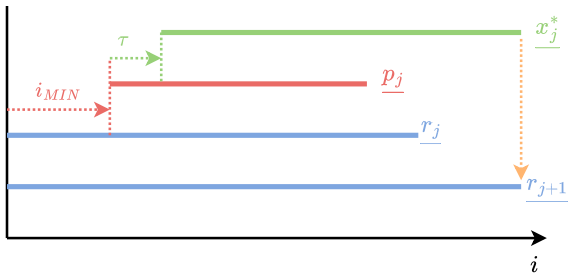
$$\tau^*_{x,p} = \arg\min_\tau \epsilon_{x,p}(\tau) \tag{18}$$

Let $\underline{x^*_j}$ be $\underline{x_j}$ shifted to align with $\underline{r_j}$ (see Fig. 4). After aligning the incoming time series and the existing prototype, PSV can be updated as defined in Eq. (19).

$$r_{j+1,i} = \begin{cases} \frac{j}{j+1} r_{j,i} + \frac{1}{j+1} x^*_{j,i} & \text{if } 0 \le i - \tau^*_{x,p} - i_{j,\min} \le N \quad \text{(overlap region)} \\ r_{j,i} & \text{otherwise} \end{cases} \tag{19}$$



(a) $\underline{x_j}$ is aligned to $\underline{p_j}$. $\underline{r_{j+1}}$ is of equal length as $\underline{r_j}$.



(b) $\underline{x_j}$ is aligned to $\underline{p_j}$. Aligned $\underline{x_j^*}$ extends past $\underline{r_j}$, therefore PSV $\underline{r}$ is extended - len($\underline{r_{j+1}}$) > len($\underline{r_j}$)

Fig. 4 Demonstration of $\underline{x_j^*}$ aligned to the prototype $\underline{p_j}$. If $\underline{x_j^*}$ extends past $\underline{r_j}$, PSV (as well as PCV and PIV) is extended accordingly

In the overlap regions (see Fig. 4), new input data is considered, increasing our confidence in the PSV values at those indices. Consequently, the confidence values expressed by PCV are updated as follows:

$$q_{j+1,i} = \begin{cases} \frac{j}{j+1} q_{j,i} + \frac{1}{j+1} & \text{if } 0 \le i - \tau^*_{x,p} - i_{j,\min} \le N \quad \text{(overlap region)} \\ \frac{j}{j+1} q_{j,i} & \text{otherwise} \end{cases} \tag{20}$$

As new data $\underline{x_j}$ is added, it may align such that it extends beyond the current lengths of $\underline{r_j}$ and $\underline{q_j}$. In such cases, PSV and PCV are extended by padding to the left and/or right. See Fig. 4b.

$$\text{len}(\underline{r_j}) + N > \text{len}(\underline{r_{j+1}}) > \text{len}(\underline{r_j}) \tag{21}$$

The newly added elements in PSV (padding) take their values from the aligned sample, while the newly added elements in PCV are initialized with $\frac{1}{j+1}$.

3.3 Interval ERAL - iERAL

The prototypes generated by sERAL are of high quality, generated in real-time, and avoid the issues seen in DBA, SSG, or other DTW-based methods. However, a simple extension presented in this paper may further enhance the utility of ERAL-based prototyping methods. In this section, we introduce Interval ERAL (iERAL), which extends sERAL by adding a variance band to the prototype.

The variance band measures the spread of the underlying data and is calculated in parallel with PSV and PCV.

The benefits of adding the variance band include:

1. **Improved insight into the underlying data.** The variance band indicates the spread of the data, aiding in identifying outliers or detecting changes in data distribution.
2. **Enhanced distance functions.** Information on data distribution enables the use of interval-based and probability-based distance functions. These functions are more robust to noise and outliers, offering a more accurate measure of similarity between time series.
3. **Adaptability to non-stationary data.** Many processes generate non-stationary data, where the distribution changes over time. The variance band helps quantify these changes, detect anomalies, and react appropriately.

In sERAL, PSV and PCV retain information about the shape and alignment of the underlying data. In iERAL, a third component is introduced: the Prototype Interval Vector (PIV). PIV is a vector of the same length as PSV and PCV, containing the standard deviation of the data at each point in PSV. It is calculated in parallel with PSV and PCV and is used to compute the variance band.

The prototype is still derived from PSV and PCV, and it is unaffected by the variance band. This ensures full compatibility between iERAL and sERAL. The variance band serves as an additional component, enabling iERAL's use in a broader range of applications.

3.3.1 Calculating PIV

PSV and PCV are calculated online, necessitating an online formula for standard deviation to compute PIV. Welford's online algorithm (Chan and G.H.G., Leveque, R.J. 1983) is a widely used method for calculating the standard deviation of a data stream previously applied in Stržinar et al. (2022); Škrjanc (2021); Stržinar et al. (2024). It is based on the following formulas:

$$\mu_n = \mu_{n-1} + \frac{x_n - \mu_{n-1}}{n} \tag{22}$$

$$M_{2,n} = M_{2,n-1} + (x_n - \mu_{n-1})(x_n - \mu_n) \tag{23}$$

$$\sigma_n = \sqrt{\frac{M_{2,n}}{n}} \tag{24}$$

Welford's algorithm is suitable for calculating the standard deviation of a stream of scalars (see Fig. 5). However, in our case, we deal with a stream of time series. We propose an extension of Welford's algorithm to calculate the standard deviation of a stream of time series, as illustrated in Fig. 6.

PSV from Eq. (19) already implements an iterative calculation of the mean value at each index and can serve as an alternative to Eq. (22).

Welford's approach requires storing M_2 for each index. To achieve this, we define $\underline{m}$:

$$\underline{m_j} = [m_{j,0}, m_{j,1}, \ldots, m_{j,R_j-1}] \tag{25}$$

Here, $\underline{m_j}$ represents $\underline{m}$ at the j-th iteration. It has the same length R_j as $\underline{r}$ and $\underline{q}$ in Eqs. (11) and (12).

Equation (23) calculates the sum of squared differences between the data points and the mean, forming the core of Welford's approach. This calculation is repeated for every element of $\underline{m}$:

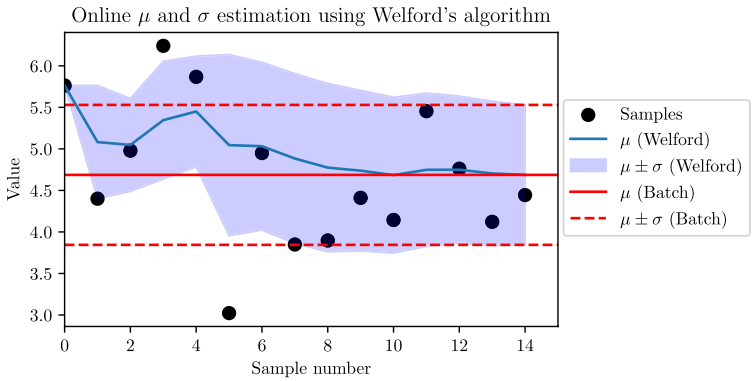


Fig. 5 Welford's algorithm for calculating the standard deviation of a stream of scalars

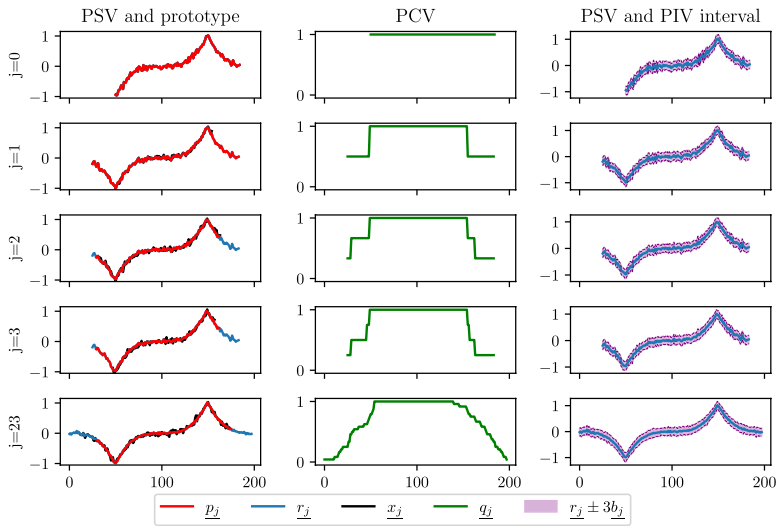


Fig. 6 Evolution of PSV, PCV, and PIV in iERAL. Each row represents the state after processing a new time series

$$m_{j+1,i} = \begin{cases} m_{j,i} + (x^*_{j,i} - r_{j,i})(x^*_{j,i} - r_{j+1,i}) & \text{if } 0 < i - i_{\min} - \tau^*_{x,p} < N \\ m_{j,i} & \text{otherwise} \end{cases} \tag{26}$$

Finally, the standard deviation formula (24) is extended to calculate the standard deviation at each index. In Eq. (24), the M_2 value is divided by the number of samples used to compute μ and M_2. PCV is used to determine the number of samples at each index:

$$b_{j,i} = \sqrt{\frac{m_{j,i}}{r_{j,i} \cdot (j+1)}} \tag{27}$$

As shown in Fig. 6, the standard deviation is calculated for each index of the time series and is not constant throughout the series.

3.3.2 PIV initialization problem

In the first row of Fig. 6, the standard deviation band is not zero. Using Eqs. (26) and (27), the standard deviation of a single sample would be zero. However, to enable the use of interval-based distance functions, the band should be non-zero even for the first sample. We propose initializing the band with the estimated internal noise level of the first sample.

Several methods exist for estimating the noise level of a time series. We propose using the standard deviation of the first derivative of the time series. Let $\sigma(\cdot)$ compute the standard deviation, then:

$$\Delta x = \{x_1 - x_0, x_2 - x_1, ..., x_N - x_{N-1}\} \tag{28}$$

$$b_{0,i} = \frac{\sigma(\Delta x)}{\sqrt{2}} \tag{29}$$

The derivation showing that $\sigma(\Delta x)/\sqrt{2}$ approximates the noise level is provided in Appendix A.

3.3.3 Merging with iERAL

In the context of evolving clustering, merging similar clusters is often desirable. This section describes the merging process for two iERAL clusters.

The first step in aligning two clusters is to compute the alignment of their prototypes. After the alignment is determined, the PSV, PCV, and PIV vectors of the two clusters are aligned accordingly, with padding added where necessary. The two clusters are then merged by:

- Weighted averaging of the PSV and PCV vectors.
- Summing the two PIV vectors.

The merging procedure is as follows:

1. Using Eqs. (14)–(16), obtain the two prototypes $\underline{p^1}$ and $\underline{p^2}$.
2. Align the two prototypes to compute the optimal lag:

$$\tau^*_{1,2} = \arg\min_\tau \left(\nu_{1,2}^{-1} \sqrt{\sum_k (p_{1,k} - p_{2,k-\tau})^2} \right) \tag{30}$$

3. Align $\underline{r^1}, \underline{q^1}, \underline{r^2}, \underline{q^2}, \underline{m^1}, \underline{m^2}$ using $\tau^*_{1,2}$. Let superscript * indicate aligned vectors (e.g., $\underline{r^{1*}}, \underline{q^{1*}}$, etc.).
4. Compute the combined PSV, PCV, and PIV similarly to Eqs. (19), (20), and (26):

$$r^M_i = \begin{cases} \frac{n^1}{n^1+n^2} r^{1*}_i + \frac{n^2}{n^1+n^2} r^{2*}_i & \text{if both } \underline{r^{1*}} \text{ and } \underline{r^{2*}} \text{ are defined at } i \\ r^{1*}_i & \text{if only } \underline{r^{1*}} \text{ is defined at } i \\ r^{2*}_i & \text{if only } \underline{r^{2*}} \text{ is defined at } i \end{cases} \tag{31}$$

$$q^M_i = \begin{cases} \frac{n^1}{n^1+n^2} q^{1*}_i + \frac{n^2}{n^1+n^2} q^{2*}_i & \text{if both } \underline{q^{1*}} \text{ and } \underline{q^{2*}} \text{ are defined at } i \\ \frac{n^1}{n^1+n^2} q^{1*}_i & \text{if only } \underline{q^{1*}} \text{ is defined at } i \\ \frac{n^2}{n^1+n^2} q^{2*}_i & \text{if only } \underline{q^{2*}} \text{ is defined at } i \end{cases} \tag{32}$$

$$m^M_i = \begin{cases} m^{1*}_i + m^{2*}_i & \text{if both } \underline{m^1} \text{ and } \underline{m^{2*}} \text{ are defined at } i \\ m^{1*}_i & \text{if only } \underline{m^1} \text{ is defined at } i \\ m^{2*}_i & \text{if only } \underline{m^{2*}} \text{ is defined at } i \end{cases} \tag{33}$$

$$n^M = n^1 + n^2 \tag{34}$$

The above procedure computes $\underline{r^M}$, $\underline{q^M}$, $\underline{m^M}$, and n^M, which define the new merged prototype.

For a detailed discussion on merging PCV and PSV, refer to Stržinar et al. (2024).

3.3.4 Availability

Full source code with examples for iERAL is available at https://repo.ijs.si/zstrzinar/ieral. Additionally, a Python package for iERAL is available at https://pypi.org/project/ieral/.

3.4 Evolving time series clustering

In our work (Stržinar et al. 2024, 2024, 2023, 2024), we analyze data streams from industrial machines. By segmenting the stream of measurements, a series of time series segments is obtained. The objective of segmentation is to ensure that

each segment represents a single action performed by the machine. The number of possible actions and their correct sequence are not known in advance. Applying an evolving time series clustering algorithm to the sequence of segments reveals groups of similar actions. Analyzing cluster assignments provides insights into the machine's actions, enabling the detection of erroneous sequences, defective actuators causing changes in time series signatures, and other anomalies.

Therefore, we apply the segments to an evolving time series clustering algorithm. Common clustering algorithms for time series data, such as K-Shape and K-ShapeStream, are insufficient for our requirements, as discussed in Sect. 2.

We propose an intuitive yet powerful application of evolving principles to time series clustering, using iERAL as the underlying alignment and averaging method.

The requirements of our evolving clustering algorithm are:

1. **Single-pass processing.** The algorithm must process data in a single pass, updating the model as new data arrives.
2. **Automatic cluster detection.** The algorithm must automatically detect the number of clusters needed to represent the data.
3. **Support for varying time series lengths.**
4. **Handling temporal shifts.** Since segmentation is not perfect, misalignment of key features may occur. The algorithm must handle such cases effectively.
5. **Cluster merging.** As data evolves, clusters may need to merge when they become similar.
6. **Distinguishing outliers.** The algorithm must differentiate between outlier samples and full clusters.

We propose the following algorithm for evolving time series clustering:

1. Use the first sample to initialize the first cluster.
2. Estimate the noise level of the first sample and use it to initialize the variance band of the first cluster.
3. For each subsequent sample:
   (a) Compute the distances between the new sample and all clusters.
   (b) If the distance to the closest established cluster is below a certain threshold, add the sample to the cluster.
   (c) Update the PSV, PCV, and PIV vectors of the cluster upon adding the sample.
   (d) If no established clusters are close enough, check the outlier cluster distances. If any distances are below a threshold, add the sample to the outlier cluster, updating PSV, PCV, and PIV.
   (e) If the sample is not added to any cluster, establish a new outlier cluster with the sample, initializing new PSV, PCV, and PIV vectors. Initialize PIV with the estimated noise level of the sample.
   (f) After adding the sample, check the cluster for merging with other clusters. Merging can occur between any cluster types (established or outlier).
   (g) Check clusters for changes in established-outlier status. Convert an established cluster to an outlier cluster if it has too few samples. Convert an outlier cluster to an established cluster if it has enough samples. The threshold is set as a percentage of the total processed samples, determined by domain knowledge.

3.4.1 iERAL-based distance calculation

As with most machine learning applications, the choice of the distance function is critical. In time series analysis, the Euclidean distance has been shown to be problematic (Fu 2011; Aghabozorgi et al. 2015). Elastic and edit-based distance measures, such as DTW and TWED, have been proposed. However, as described in Sect. 2, these methods have their own limitations. The ERAL score provides a non-elastic distance measure.

Having expanded ERAL with the variance band, we propose the use of an interval-based distance function. A simple interval-based distance can be computed as the proportion of the sample that falls outside the variance band of the prototype. This proportion can then be corrected by the overlap factor, similar to the distance presented in Stržinar et al. (2024).

In summary, the distance function is calculated as:

$$d(\underline{x}, C) = 1 - \frac{\sum_{i=i_{\min}+\tau}^{i_{\min}+\tau+N} \mathbb{1}(r_i - b_i \leq x_i \leq r_i + b_i)}{n} \tag{35}$$

Here, C is the cluster, and $\underline{x}$ is the sample. $\underline{r}$ represents the PSV of C, and $\underline{b}$ represents the PIV of C.

3.4.2 Evolving clustering demonstration

In order to demonstrate the evolving clustering procedure, Fig. 7 provides a demonstration of clustering as 14 time series are sequentially processed by the algorithm. Each

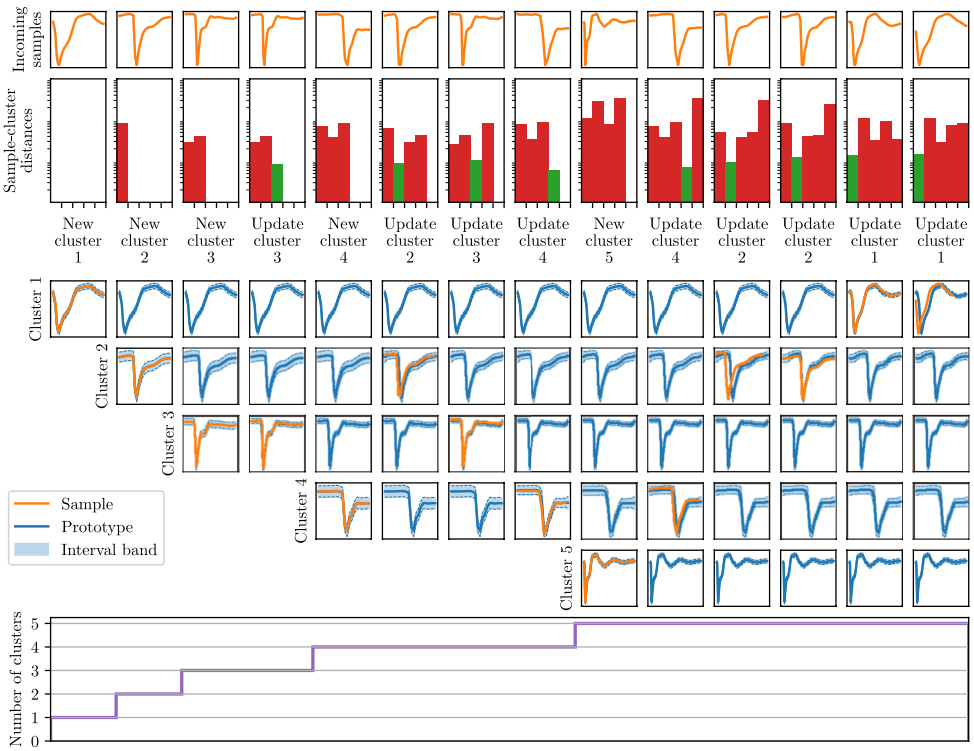


Fig. 7 Demonstration of evolving clustering for 14 time series passed to the algorithm. In total 5 clusters are initialized. Each column represents the processing of a single time series. First row plots the incoming samples. Second row depicts the distances between sample and all clusters. The central part demonstrates the evolution of the 5 clusters. Plot at the bottom shows progression of the number of clusters

column corresponds to a single input time series and its effect on the current clustering state.

The top row shows the incoming time series. The evolving clustering algorithm processes them one at a time. The first time series is used to initialize the first cluster (see central part of the figure). The noise estimate as calculated using Eq. (29) is used to initialize the PIV interval band. The total number of clusters is 1 (bottom plot of figure).

The second time series is processed (Fig. 7, second column) by computing the distance to the existing cluster (second row). The distance is too high (red bar), and a new cluster is initialized. Again, the PIV interval band is initialized by Eq. (29).

The third time series also initializes a new cluster, however, the fourth time series is close enough to the prototype of the third cluster that Cluster 3 is updated (green bar in second row).

The process is repeated for all incoming time series.

The second row of Fig. 7 visualizes the distances between the current sample and all existing clusters. A green bar indicates assignment to the closest cluster. If all the bars are red, the distance to the closest cluster exceeds the threshold, and a new cluster is created.

The central matrix of the demonstrative figure shows the temporal evolution of up to five clusters, with each cluster visualized using its current prototype (line) and the surrounding interval band (shaded area). Clusters are updated when a new sample is added, or remain unchanged otherwise.

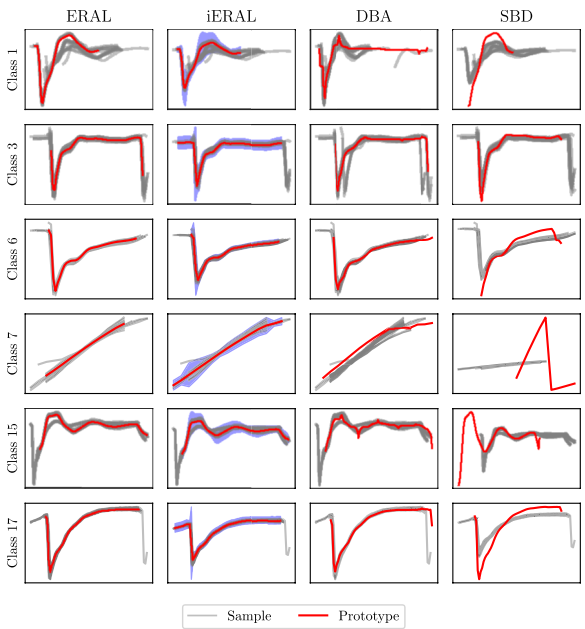


Fig. 8 Comparison of DBA, SBD, ERAL, and iERAL

The bottom plot tracks the total number of clusters over time, helping to highlight cluster initialization and merging behavior.

4 Experiment and results

4.1 Comparison of alignment and averaging methods

One of the contributions of this paper is iERAL, an alignment and averaging method. The performance of iERAL is illustrated in Fig. 8. The figure shows the prototypes generated by DBA, SBD, ERAL, and iERAL for six sets of time series corresponding to six classes of data from (Stržinar et al. 2024, 2024).

Analyzing DBA, we observe unwanted spikes and plateaus in the prototypes. These artifacts, previously reported by several researchers (Liu et al. 2023; Rasines et al. 2023; Morel et al. 2018; Liu et al. 2019; Stržinar et al. 2024), can be attributed to the use of DTW as the underlying alignment method.

The rightmost column of Fig. 8 shows SBD prototypes. Since these are not based on DTW, the plateaus and spikes observed in DBA are absent. However, some scaling issues and prototype shortening are evident. Compared to the other three methods, SBD prototypes appear too short relative to the underlying data. Although the zero-mean tendency at the edges, as reported in (Stržinar et al. 2024), is not observed in this dataset, it is likely to occur in others given the properties of SBD.

Comparing ERAL and iERAL, we observe similar prototypes; however, iERAL better captures the shape at the edges. Given the excellent results of ERAL in Stržinar et al. (2024), iERAL demonstrates comparable performance, achieved in a single pass of the data. Additionally, the inclusion of the standard deviation band adds significant value by capturing data variability.

The confidence band in iERAL is observed to vary in width, confirming the hypothesis that the standard deviation of the data is not constant throughout the time series. This variability underscores the need for a mechanism to accurately capture and represent this characteristic.

4.2 Evolving clustering of unlabeled industrial data

The ultimate goal of this work is the unsupervised clustering of a stream of otherwise unlabeled sensor readings from industrial machines. By segmenting the data in the vicinity

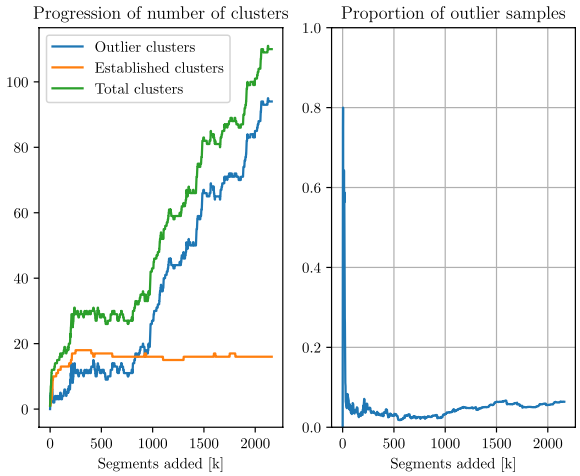
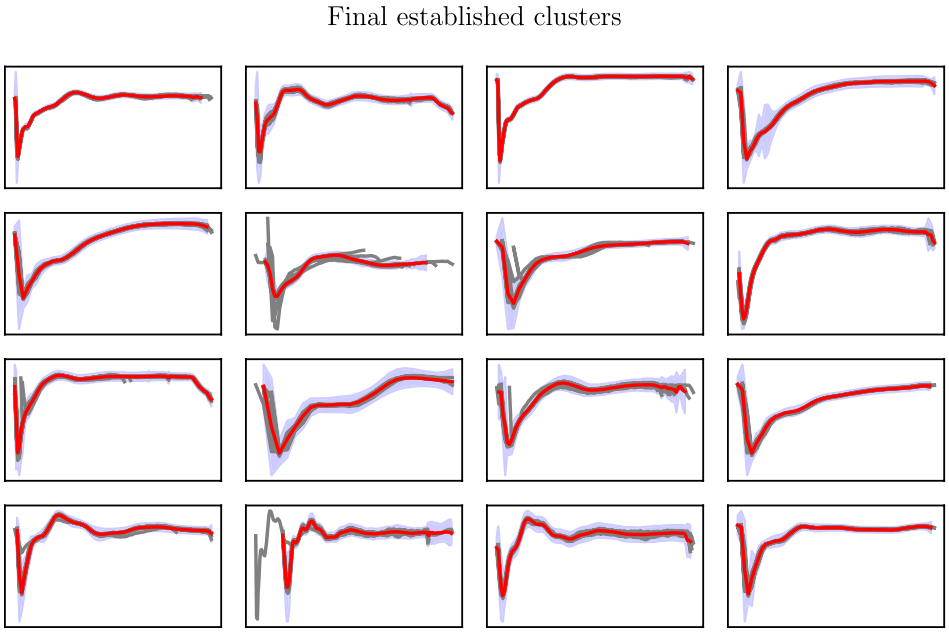


Fig. 10 Evolving clustering of unlabeled industrial data–progression of the number of clusters

Fig. 9 Evolving clustering of unlabeled industrial data



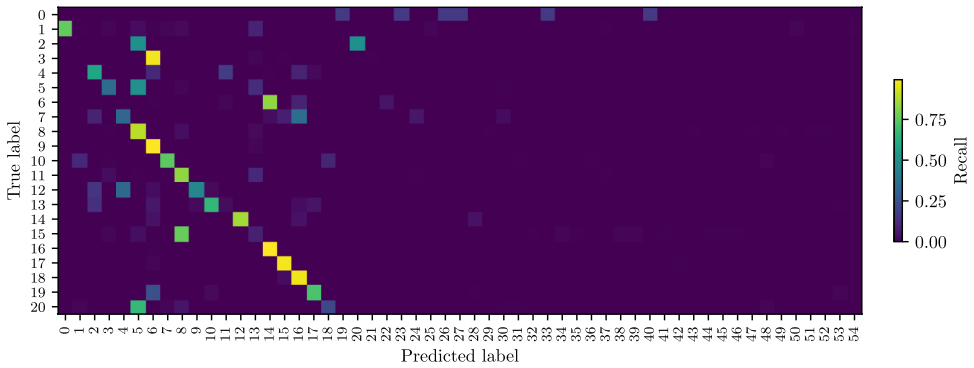


Fig. 11 Comparison of evolving clustering results and ground truth labels

of large drops in the (pressure) signal, a sequence of time series segments is obtained. These segments are passed to iERAL, and the results are demonstrated in Fig. 9. The figure shows the final established clusters–clusters containing more than 0.5% of all samples passed to the clustering algorithm.

Observing the established clusters in Fig. 9, we notice that the clusters are well-separated. Each cluster represents a distinct shape, and the shapes are consistent within each cluster. The confidence band clearly envelopes the data without being too wide to obscure the shape. The good separation of clusters indicates an effective choice of distance function–the interval-based approach described in Eq. (35).

In Fig. 10, we observe the progression of the number of clusters as the data is processed. Established clusters (those accounting for more than 0.5% of past samples at a given time) are shown in orange, while outlier clusters are shown in blue.

The number of established clusters (16) stabilizes early, after around 400 processed samples. Subsequently, only minor additions, promotions (outlier to established), demotions (established to outlier), and merges are observed. The number of outlier clusters is high, but analysis of Fig. 10 shows that this number remained relatively stable during the first part of the experiment. After processing roughly 900 samples, the number of outlier clusters begins to increase, likely due to the algorithm detecting new patterns in the data. These results are consistent with prior studies on this dataset (Stržinar et al. 2024). This behavior suggests the algorithm's capability to detect anomalous machine behavior, marking new patterns as outliers.

4.3 Comparison of evolving clustering results and ground truth labels

The industrial dataset used in the previous experiment is also available in a pre-segmented and labeled format. This enables a comparison of the unsupervised clustering algorithm proposed in this paper with ground truth labels. The results are shown in the confusion matrix in Fig. 11.[1]

The confusion matrix reveals a strong diagonal, indicating good performance of the clustering algorithm. The clusters identified correlate strongly with the actual class labels. The clustering algorithm has also detected several outlier clusters, which explains why the number of columns in the confusion matrix is greater than the number of rows (corresponding to ground truth labels). As expected, the outlier clusters are mostly empty.

4.4 Comparison to other time series clustering algorithms

To further evaluate the effectiveness of our proposed algorithm, we conducted a comparative study against several established time series clustering techniques. We compare the proposed algorithm to K-Shape Paparrizos and Gravano (2015), K-Means and K-Medoids clustering. We include results using both Dynamic Time Warping (DTW) Berndt and Clifford (1994) and Move-Split-Merge (MSM) Stefan et al. (2012) distances, as suggested in Holder et al. (2024). Additionally, we include DBSCAN Schubert et al. (2017) with DTW in our comparison. A Python library for time series analysis, the AEON toolkit (AEON 2025), also includes two additional clustering algorithms, which we include in our comparison: CLARA and CLARANS (Ng and Han 2002). We also include K-ShapeStream Bariya et al. (2021) as the streaming implementation of K-Shape.

The experiments were performed on a labeled industrial dataset, which enabled quantitative evaluation of clustering quality using the Adjusted Rand Index (ARI) and the V-score.

The Adjusted Rand Index (ARI) is a measure of similarity between two data clusterings, adjusted for chance. It

[1] The predicted labels in Fig. 11 have been permuted using the Hungarian algorithm to obtain the optimal diagonal. Permuting class labels in an unsupervised learning algorithm does not affect the validity of the results.

Table 2 Comparison of time series clustering algorithms on the labeled industrial dataset

Algorithm	Distance	V-score	ARI
Our solution	ERAL	0.67	0.39
K-Shape	SBD	0.42	0.12
K-ShapeStream	SBD	0.44	0.11
K-Means	DTW	0.50	0.20
K-Means	MSM	0.55	0.23
K-Medoids	DTW	0.47	0.17
K-Medoids	MSM	0.52	0.23
DBSCAN	DTW	0.64	0.22
CLARA	DTW	0.31	0.06
CLARANS	DTW	0.48	0.17

evaluates how well the predicted clustering agrees with the ground truth labels. The ARI ranges from -1 to 1, where 1 indicates perfect agreement, 0 indicates random labeling, and negative values suggest worse than random agreement.

The V-score is an entropy-based external cluster evaluation metric. It is the harmonic mean of *homogeneity* (each cluster contains only members of a single class) and *completeness* (all members of a given class are assigned to the same cluster). It is defined as:

$$V = 2 \cdot \frac{\text{homogeneity} \cdot \text{completeness}}{\text{homogeneity} + \text{completeness}}$$

Both homogeneity and completeness are derived from conditional entropy calculations and lie in the range [0, 1], hence $V \in [0, 1]$. Higher values indicate better clustering performance.

A summary of the results is presented in Table 2.

Our algorithm outperformed the other tested methods in both V-score and ARI, even though it was not provided with prior knowledge of the true number of clusters. In contrast, the competing algorithms were all initialized with the true number of clusters. Furthermore, our algorithm operates in a single-pass fashion, a significant advantage for streaming data applications, whereas the others, with the exception of K-ShapeStream, are inherently batch algorithms. K-ShapeStream was run in micro-batch mode with batches of 30 samples.

This combination of strong performance and practical suitability for evolving data streams underlines the value of our algorithm in real-world industrial settings, where prior knowledge of the number of clusters is rarely available and online processing is often required.

5 Discussion

The results shown in Sect. 4 demonstrate the ability of iERAL so produce meaningful cluster prototypes. In combination with the proposed clustering algorithm clusters corresponding to the underlying process steps are found.

Due to the single-pass nature of iERAL, it is much faster when applied to industrial applications where data is streamed and clusters must be updated in real time. As shown by the computational complexity analysis, any batch-processing solution is not suitable. iERAL provides a suitable alternative.

The prototypes generated by iERAL are similar to those by ERAL. Both methods outperform DTW-based methods such as DBA by not introducing spike and plateau artifacts. This is especially important in industrial applications where slight changes in the shape of the underlying signal can indicate important faults which must be diagnosed.

By adding the interval band, iERAL captures not only the shape but also the variance of the underlying data, adding a second dimension to any downstream analysis task.

6 Conclusion and future work

The key contributions of our paper are twofold: 1) the introduction of the variance band to the ERAL framework through iERAL, 2) and the evolving time series clustering algorithm.

The proposed iERAL method enables single-pass calculation the prototype of a set of time series. The addition of the variance band, enables additional insights into the data and improved cluster interpretability.

iERAL combined with the proposed evolving time series clustering algorithm, enables the analysis of evolving data streams in real time. Unlike previous methods, the generated prototypes avoid common artifacts such as spikes and plateaus. The shape of streaming data is preserved by iERAL and the clusters generated by the evolving clustering algorithm correspond well to the underlying industrial process.

Our experiments have demonstrated how iERAL and the clustering algorithm can capture the patterns in streamed industrial data. The found clusters match the underlying industrial process steps.

Given the promising results presented in this paper, further industrial datasets should be obtained for additional evaluation of the proposed methods.

Since the distance function is a critical part of any time series analysis, the work presented here could be expanded

by further research into interval-based and probabilistic distance measures.

A noise estimation

In this section we examine the noise estimation proposal used in Sect. 3.3.2 to initialize the Prototype Inverval Vector.

Let the time series $\underline{x} = \{x_1, x_2, \dots, x_n\}$ be modeled as:

$$x_i = s_i + n_i \tag{36}$$

where s_i represents the smooth underlying signal, and n_i represents zero-mean, independent, and identically distributed (i.i.d.) noise with $E[n_i] = 0$ and $\text{Var}(n_i) = \sigma^2$.

The first-order difference of the time series is defined as:

$$\Delta x_i = x_{i+1} - x_i \tag{37}$$

Substituting the model $x_i = s_i + n_i$, we have:

$$\Delta x_i = (s_{i+1} - s_i) + (n_{i+1} - n_i) \tag{38}$$

$$\Delta x_i = \Delta s_i + \Delta n_i \tag{39}$$

where $\Delta s_i = s_{i+1} - s_i$ and $\Delta n_i = n_{i+1} - n_i$.

The variance of Δx_i is given by:

$$\text{Var}(\Delta x_i) = E[(\Delta x_i - E[\Delta x_i])^2] \tag{40}$$

Since the noise n_i is zero-mean ($E[n_i] = 0$) and the smooth signal is assumed to vary around a local mean ($E[\Delta s_i] = 0$), we have:

$$E[\Delta x_i] = E[\Delta s_i] + E[\Delta n_i] = 0 \tag{41}$$

Thus:

$$\text{Var}(\Delta x_i) = E[(\Delta x_i)^2] \tag{42}$$

Substituting $\Delta x_i = \Delta s_i + \Delta n_i$, we expand:

$$\text{Var}(\Delta x_i) = E[(\Delta s_i + \Delta n_i)^2] \tag{43}$$

Using the linearity of expectation:

$$E[(\Delta s_i + \Delta n_i)^2] = E[(\Delta s_i)^2] + 2E[\Delta s_i \Delta n_i] + E[(\Delta n_i)^2] \tag{44}$$

Assuming the smooth signal Δs_i and noise Δn_i are independent, we have:

$$E[\Delta s_i \Delta n_i] = 0 \tag{45}$$

This reduces the variance expression to:

$$\text{Var}(\Delta x_i) = E[(\Delta s_i)^2] + E[(\Delta n_i)^2] \tag{46}$$

For the noise term $\Delta n_i = n_{i+1} - n_i$:

$$E[(\Delta n_i)^2] = E[(n_{i+1} - n_i)^2] \tag{47}$$

Expanding the square:

$$E[(n_{i+1} - n_i)^2] = E[n_{i+1}^2] + E[n_i^2] - 2E[n_{i+1} n_i] \tag{48}$$

Since n_i are i.i.d., $E[n_{i+1}^2] = E[n_i^2] = \sigma^2$, and $E[n_{i+1} n_i] = 0$, we get:

$$E[(\Delta n_i)^2] = \sigma^2 + \sigma^2 = 2\sigma^2 \tag{49}$$

The term $E[(\Delta s_i)^2]$ represents the variance of the smooth signal differences. For a sufficiently smooth signal, this variance is small, and we approximate:

$$E[(\Delta s_i)^2] \approx 0 \tag{50}$$

Substituting back, we have:

$$\text{Var}(\Delta x_i) \approx E[(\Delta n_i)^2] = 2\sigma^2 \tag{51}$$

Taking the square root to compute the standard deviation:

$$\text{std}(\Delta x_i) = \sqrt{\text{Var}(\Delta x_i)} \approx \sqrt{2}\sigma \tag{52}$$

Rearranging for σ, the noise level is:

$$\sigma \approx \frac{\text{std}(\Delta x_i)}{\sqrt{2}} \tag{53}$$

Thus, the standard deviation of the noise can be estimated from the first-order differences of the time series as:

$$\text{std}(\underline{n}) \approx \frac{\text{std}(\text{diff}(\underline{x}))}{\sqrt{2}} \tag{54}$$

Acknowledgements This work has been supported by the Slovenian Research and Innovation Agency research programs P2-0001, P2-0219, research project L2-4454 abd Horizon Europe project AI REDGIO 5.0 (Grant agreement ID: 101092069).

Author Contributions Žiga Stržinar: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. Igor Škrjanc: Conceptualization, Funding acquisition, Project administration, Resources, Supervision, Validation, Writing – review and editing. Boštjan Pregelj: Conceptualization, Data curation, Funding acquisition, Project administration, Resources, Supervision, Validation, Writing – review and editing.

Data Availability The industrial dataset used in this paper is available at https://data.mendeley.com/datasets/ypzswhhzh9. Key results shown in this paper can be generated using the source code available at https://repo.ijs.si/zstrzinar/ieral.

Declarations

Declaration of Generative AI and AI-assisted technologies in the writing process During the preparation of this work the authors used ChatGPT in order to improve the readability of individual text paragraphs. After

using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

Abanda A, Mori U, Lozano JA (2019) A review on distance based time series classification. Data Min Knowl Disc 33(2):378–412. https://doi.org/10.1007/s10618-018-0596-4

AEON toolkit. https://www.aeon-toolkit.org/en/stable/. Accessed 21 Mar 2025

Aghabozorgi S, Shirkhorshidi AS, Wah TY (2015) Time-series clustering—a decade review. Inf Syst 53:16–38. https://doi.org/10.1016/j.is.2015.04.007

Andonovski G, Mušič G, Blažič S, Škrjanc I (2018) Evolving model identification for process monitoring and prediction of non-linear systems. Eng Appl Artif Intell 68:214–221. https://doi.org/10.1016/j.engappai.2017.10.020

Antić M, Zdešar A, Škrjanc I (2021) Depth-image segmentation based on evolving principles for 3d sensing of structured indoor environments. Sensors 21(13):4395. https://doi.org/10.3390/s21134395

Bagnall A, Lines J, Bostrom A, Large J, Keogh E (2017) The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min Knowl Disc 31(3):606–660. https://doi.org/10.1007/s10618-016-0483-9

Bariya M, Meier A, Paparrizos J, Franklin MJ (2021) k-shapestream: probabilistic streaming clustering for electric grid events. In: 2021 IEEE Madrid PowerTech, pp 1– 6 . https://doi.org/10.1109/PowerTech46648.2021.9494830 . IEEE

Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: Proceedings of the 3rd International conference on knowledge discovery and data mining, pp 359– 370. AAAI Press

Blažič S, Škrjanc I, Matko D (2014) A robust fuzzy adaptive law for evolving control systems. Evol Syst 5:3–10. https://doi.org/10.1007/s12530-013-9084-7

Chan Tony F, GHG, Leveque RJ (1983) Algorithms for computing the sample variance: analysis and recommendations. Am Stat 37(3):242–247. https://doi.org/10.1080/00031305.1983.10483115

Cuturi M, Blondel M (2017) Soft-DTW: a differentiable loss function for time-series. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proc Mach Learn Res 70:894–903 PMLR (https://proceedings.mlr.press/v70/cuturi17a.html)

Dau HA, Bagnall AJ, Kamgar K, Yeh CM, Zhu Y, Gharghabi S, Ratanamahatana CA, Keogh EJ (2018) The UCR time series archive. CoRR **abs/1810.07758** (2018) arXiv:1810.07758

Dau HA, Keogh E, Kamgar K, Yeh C-CM, Zhu Y, Gharghabi S, Ratanamahatana CA, Yanping Hu B, Begum N, Bagnall A, Mueen A, Batista G, Hexagon-ML (2018) The UCR time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/

Fu TC (2011) A review on time series data mining. Eng Appl Artif Intell 24(1):164–181. https://doi.org/10.1016/j.engappai.2010.09.007

Gupta L, Molfese DL, Tammana R, Simos PG (1996) Nonlinear alignment and averaging for estimating the evoked potential. IEEE Trans Biomed Eng 43(4):348–356. https://doi.org/10.1109/10.486255

He H, et al (2018) Applications of reference cycle building and K-shape clustering for anomaly detection in the semiconductor manufacturing process. PhD thesis, Massachusetts Institute of Technology . https://dspace.mit.edu/handle/1721.1/120246

Holder C, Middlehurst M, Bagnall A (2024) A review and evaluation of elastic distance functions for time series clustering. Knowl Inf Syst 66(2):765–809. https://doi.org/10.1007/s10115-023-01952-0

Kim S-W. Park S, Chu WW (2001) An index-based approach for similarity search supporting time warping in large sequence databases. In: Proceedings 17th International conference on data engineering, pp 607– 614 (2001). https://doi.org/10.1109/ICDE.2001.914875 . IEEE

Kumar U, Legendre CP, Zhao L, Chao BF (2022) Dynamic time warping as an alternative to windowed cross correlation in seismological applications. Seismol Soc Am 93(3):1909–1921. https://doi.org/10.1785/0220210288

Leite D, Škrjanc I, Gomide F (2020) An overview on evolving systems and learning from stream data. Evol Syst 11(2):181–198. https://doi.org/10.1007/s12530-020-09334-5

Lim T, Ong CS (2021) Portfolio diversification using shape-based clustering. J Finan Data Sci 3(1):111. https://doi.org/10.3905/jfds.2020.1.054

Liu Y-T, Zhang Y-A, Zeng M (2019) Adaptive global time sequence averaging method using dynamic time warping. IEEE Trans Signal Process 67(8):2129–2142. https://doi.org/10.1109/TSP.2019.2897958

Liu Y, Zhang Y-A, Zeng M, Zhao J (2023) A novel shape-based averaging algorithm for time series. Eng Appl Artif Intell 126:107098. https://doi.org/10.1016/j.engappai.2023.107098

Masini RP, Medeiros MC, Mendes EF (2023) Machine learning advances for time series forecasting. J Econ Surv 37(1):76–111. https://doi.org/10.1111/joes.12429

Morel M, Achard C, Kulpa R, Dubuisson S (2018) Time-series averaging using constrained dynamic time warping with tolerance. Pattern Recogn 74:77–89. https://doi.org/10.1016/j.patcog.2017.08.015

Ng RT, Han J (2002) Clarans: A method for clustering objects for spatial data mining. IEEE Trans Knowl Data Eng 14(5):1003–1016. https://doi.org/10.1109/TKDE.2002.1033770

Niennattrakul V, Ratanamahatana CA (2009) Shape averaging under time warping. In: 2009 6th International conference on electrical engineering/electronics, computer. Telecommun Inf Technol 2:626–629. https://doi.org/10.1109/ECTICON.2009.5137128. (**IEEE**)

Niennattrakul V, Ratanamahatana CA (2007) Inaccuracies of shape averaging method using dynamic time warping for time series data. In: Computational Science–ICCS 2007: 7th International conference, Beijing, China, May 27-30, 2007, Proceedings, Part I 7, pp 513– 520 . https://doi.org/10.1007/978-3-540-72584-8_68 . Springer

Paparrizos J, Gravano L (2015) k-shape: efficient and accurate clustering of time series. In: Proceedings of the 2015 ACM SIGMOD International conference on management of data, pp 1855– 1870 https://doi.org/10.1145/2723372.2737793

Petitjean F, Ketterlin A, Gançarski P (2011) A global averaging method for dynamic time warping, with applications to

clustering. Pattern Recogn 44(3):678–693. https://doi.org/10.1016/j.patcog.2010.09.013
Rasines I, Remazeilles A, Prada M, Cabanes I (2023) Minimum cost averaging for multivariate time series using constrained dynamic time warping: a case study in robotics. IEEE Access. https://doi.org/10.1109/ACCESS.2023.3300720
Ratanamahatana CA, Keogh E (2004) Making time-series classification more accurate using learned constraints. In: Proceedings of the 2004 SIAM International conference on data mining, pp 11– 22 . https://doi.org/10.1137/1.9781611972740.2 . SIAM
Schubert E, Sander J, Ester M, Kriegel HP, Xu X (2017) Dbscan revisited, revisited: why and how you should (still) use dbscan. ACM Trans Database Syst (TODS) 42(3):1–21. https://doi.org/10.1145/3068335
Schultz D, Jain B (2018) Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces. Pattern Recogn 74:340–358. https://doi.org/10.1016/j.patcog.2017.08.012
Shirota Y, Murakami A (2021) Long-term time series data clustering of stock prices for portfolio selection. In: 2021 IEEE International conference on service operations and logistics, and informatics (SOLI), pp 1– 6 (2021). https://doi.org/10.1109/SOLI54607.2021.9672407 . IEEE
Singh T, Kalra R, Mishra S, Satakshi, Kumar M (2023) An efficient real-time stock prediction exploiting incremental learning and deep learning. Evol Syst 14(6):919–937. https://doi.org/10.1007/s12530-022-09481-x
Škrjanc I (2021) eGAUSS+ evolving clustering in classification. In: 2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI), pp 533– 538 .https://doi.org/10.1109/SACI51354.2021.9465615 . IEEE
Škrjanc I, Andonovski G, Ledezma A, Sipele O, Iglesias JA, Sanchis A (2018) Evolving cloud-based system for the recognition of drivers' actions. Expert Syst Appl 99:231–238. https://doi.org/10.1016/j.eswa.2017.11.008
Škrjanc I, Iglesias JA, Sanchis A, Leite D, Lughofer E, Gomide F (2019) Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A survey. Inf Sci 490:344–368. https://doi.org/10.1016/j.ins.2019.03.060
Škrjanc I, Andonovski G, Iglesias JA, Sesmero MP, Sanchis A (2022) Evolving gaussian on-line clustering in social network analysis. Expert Syst Appl 207:117881. https://doi.org/10.1016/j.eswa.2022.117881
Stefan A, Athitsos V, Das G (2012) The move-split-merge metric for time series. IEEE Trans Knowl Data Eng 25(6):1425–1438. https://doi.org/10.1109/TKDE.2012.88
Stržinar Ž, Pregelj B, Petrovčič J, Škrjanc I, Dolanc G (2024) Pneumatic Pressure and Electrical Current Time Series in Manufacturing. Mendeley Data (2024). https://doi.org/10.17632/ypzswhhzh9.2 . https://data.mendeley.com/datasets/ypzswhhzh9/2
Stržinar Ž, Pregelj B, Petrovčič J, Škrjanc I, Dolanc G (2024) Time series insights from the shopfloor: A real-world dataset of pneumatic pressure and electrical current in discrete manufacturing. Data in Brief, 110619 https://doi.org/10.1016/j.dib.2024.110619
Stržinar Ž, Pregelj B, Škrjanc I (2024) Analysis of time series alignment and averaging methods. In: Proceedings of the 33. International electrotechnical and computer science conference ERK 2024, Portorož, Slovenia
Stržinar Ž, Škrjanc I, Pratama M, Pregelj B (2024) Evolving clustering of time series for unsupervised analysis of industrial data streams. Available at SSRN 5026151 https://doi.org/10.2139/ssrn.5026151
Stržinar, Ž., Škrjanc, I (2022) Self-tuned model-based predictive control using evolving fuzzy model of a non-linear dynamic process. In: Explainable AI and Other Applications of Fuzzy Techniques: Proceedings of the 2021 Annual Conference of the North American Fuzzy Information Processing Society, NAFIPS 2021, pp. 406– 421 . https://doi.org/10.1007/978-3-030-82099-2_37 . Springer
Stržinar Ž, Pregelj B, Škrjanc I (2023) Soft sensor for non-invasive detection of process events based on eigenresponse fuzzy clustering. Appl Soft Comput 132:109859. https://doi.org/10.1016/j.asoc.2022.109859
Stržinar Ž, Pregelj B, Škrjanc I (2024) Non-elastic time series fuzzy clustering for efficient analysis of industrial data sets. Appl Soft Comput 167:112398. https://doi.org/10.1016/j.asoc.2024.112398
Yi B-K, Jagadish HV, Faloutsos C (1998) Efficient retrieval of similar time sequences under time warping. In: Proceedings 14th International conference on data engineering, pp 201– 208 . https://doi.org/10.1109/ICDE.1998.655778 . IEEE