Applied Soft Computing 132 (2023) 109859

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Soft sensor for non-invasive detection of process events based on Eigenresponse Fuzzy Clustering

Žiga Stržinar [a,b,*], Boštjan Pregelj [a], Igor Škrjanc [b]

[a] "Jožef Stefan" Institute, Jamova cesta 39, Ljubljana, Slovenia
[b] Faculty of Electrical Engineering, University of Ljubljana, Tržaška cesta 25, Slovenia

ARTICLE INFO

Article history:
Received 19 July 2022
Received in revised form 22 October 2022
Accepted 18 November 2022
Available online 25 November 2022

Keywords:
Time series analysis
Classification algorithms
Fuzzy classification
Manufacturing
Event detection

ABSTRACT

Changes in process states and properties can be observed through measured variables. In this way, by classifying time series segments of measured data, changes in model parameters can be detected and the system state can be inferred. Time series classification methods are used in many fields, but the work presented here focuses mainly on the field of manufacturing. In the category of whole-series time series classifiers, the Nearest Neighbor classifier is often used. The aim of this work is to introduce an alternative supervised method for time series classification — Eigenresponse Fuzzy Clustering (EFC). We introduce class eigenresponses, which are time series prototypes of a class. We propose the learning eigenresponses for each class using a fuzzy clustering technique. Unlike some existing methods, we propose the use of multiple prototypes per class to better describe a wider range of values for each class. Moreover, the presented method is evaluated on several datasets. Using a dataset obtained on an industrial test bench on an e-bike drive assembly line, the method correctly classifies all time series. To further validate the performance, a set of publicly available datasets (UCR Archive) is used. For the category of datasets most similar to the target industrial application, an improvement over the benchmark approach is obtained.

1. Introduction

Nowadays, with Industry 4.0 standards [1] more prevalent, production monitoring and fault detection are becoming a required capacity in the manufacturing industry [2]. However, production machines, except for the most advanced ones, lack such capabilities. Therefore, a system that provides this capability would be of great importance [3,4]. The idea pursued in this work is that the above functionality can be achieved through an algorithm that enables event detection by analyzing omnipresent data, such as supply current or pneumatic line pressure.

To achieve this goal, a reliable, efficient and explainable method is required that performs pre-processing, segmentation and classification of the data. Segmentation of time series is an important step in time series analysis and several methods have been proposed [5–7]. However, for the work in this paper, a simpler derivative-based heuristic method is sufficient. In this paper, our main focus is on the classification of time series with limited length.

Time series classification (TSC) is an active area of research. Its growing importance is fostered by the explosion of data sources that generate time series: the Internet of Things (IoT), Industry 4.0, the financial sector [8], health and medical devices [9], weather stations and air quality monitors, GPS loggers, etc.

TSC applications include fault detection [10–12], object recognition, energy consumption analysis [13], traffic pattern analysis [14] and distracted driver detection [15]. In manufacturing, there are several use cases for time series classification: (1) workpiece defect detection, (2) machine monitoring for preventive and predictive maintenance, (3) detection of unexpected/faulty machine operations, (4) workpiece categorization. We see potential in an algorithm that is able to classify machine time series data into actions that are performed by the machine. The sequence of steps can then be analyzed by a downstream algorithm for possible deviations from normal behavior indicating irregular operation.

TSC techniques are categorized into five categories [16,17]: (1) whole series (distance based) approaches, (2) phase based intervals, (3) shapelet based, (4) dictionary based and (5) ensemble approaches. The classification of multivariate time series is discussed in [18].

In the whole series category of TSC methods, time series are compared using all data points, either by a special time series distance measure or by treating the time series as a high-dimensional vector and using traditional classification approaches. Several methods from this category are presented

[*] Corresponding author at: "Jožef Stefan" Institute, Jamova cesta 39, Ljubljana, Slovenia.
E-mail address: ziga.strzinar@ijs.si (Ž. Stržinar).

https://doi.org/10.1016/j.asoc.2022.109859

Table 1
Table of symbols and indices.

Symbol	Description
c	Number of clusters (FCM), number of prototypes per class (EFC)
k	Number of classes in the classification task, known a priori from the problem description
η	FCM fuzziness parameter
i	Cluster (FCM)/eigenresponse (EFC) index
j	Sample index
N	Number of samples
t	Class index
a	Index of sample awaiting classification

in [16], all of which use a combination of the nearest neighbor classifier (1NN) and an elastic distance measure. Alternative distance based methods are described in [19]. By far the most commonly used elastic distance measure is Dynamic Time Warping (DTW) [20]. Alternatives include FastDTW [21,22], TWED [23], WDTW [24], DD_{DTW} [25], MSM [26], DD_{DTW} [27]. Whole-series approaches to classifying time series are most appropriate when the distinguishing features are present throughout the time series, but may be subject to temporal shifts between individual time series. An example of such a dataset is 'FiftyWords' from the UCR archive [28].

The second category of TSC techniques are phase dependent interval approaches, where the time series are analyzed in intervals. This is advantageous when the distinguishing features of classes are only present in certain regions (intervals) of the entire time series. Examples of classifiers that use this approach are Time Series Forest [29], Time Series Bag of Features [30], and Learned Pattern Similarity [31].

Shapelet based algorithms focus on shapelets — short patterns that define a class and can occur anywhere in the series [32,33]. The defining feature of a class is the presence or absence of a shapelet. Several approaches have been developed: Fast Shapelets [34], Shapelet Transform [35], Learned Shapelets [36].

Dictionary based approaches focus on repetitive patterns in time series. Examples include: Bag of Patterns [37], SAXVSM [38], BOSS [39].

Ensemble approaches are very competitive in general classification problems and are also becoming popular in time series classification. Prominent examples are DTW_F and COTE [40], HIVE-COTE [41], HIVE-COTE v1.0 [42], HIVE-COTE v2.0 [43].

Above we have explained the characteristics of the different categories for classifying time series. For our task, a suitable category of TSC methods should be selected. In the case of continuous measurements of, for example, pneumatic pressure, segmented into meaningful segments ('meaningful' segments in this context are segments corresponding to the operations performed by the production machine, each segment only corresponding to exactly one operation), the task is to classify the segments and thus identify the exact series of operations performed by the machine. The whole series category seems to be the most appropriate, because it is expected that each operation results in a distinct response, the distinguishing feature of the time series segments is their overall shape. This will be demonstrated later in the article.

In the whole series TSC category, a number of researchers have made considerable efforts to develop different elastic distance measures, but the classifier has remained largely the same — 1NN. The use of 1NN comes with some disadvantages: (1) classification requires searching the entire training set and is therefore an expensive operation as the training set increases, (2) sensitivity to outliers in the sample, (3) the 'model' (the entire set of training time series) cannot be visualized.

In the past, a subset of whole series TSC methods called Distance features and Distance kernel methods, were presented [19]. The Distance features methods are of particular interest, they involve three steps: (1) computing a square distance matrix of the distances between training time series, (2) alternatively representing each training time series as distance matrix rows, (3) using well-known classifiers such as Support Vector Machine (SVM). Several methods have been proposed to reduce the computational cost of these methods [19].

Literature has explored the use of fuzzy clustering and classification in time series analysis. For clustering of time series, [44] lists Fuzzy C-means alongside Crisp partitioning methods. In [45], DTW and fuzzy clustering techniques are used, with the authors proposing their own fuzzy clustering algorithm. In [46], a fuzzy clustering algorithm is applied to switching time series. Recently, fuzzy methods have been applied to the analysis of COVID-19 data [47]. In [48], fuzzy class representation is used for a time series classification task. In [49], fuzzy clustering was applied to the change point detection problem (CPD [5]).

Our paper introduces Eigenresponse Fuzzy Clustering (EFC). Compared to the established combination of 1NN and elastic distances, the proposed approach aims to overcome the mentioned drawbacks of 1NN. Our approach differs from existing applications of fuzzy time series clustering and classification approaches by using multiple prototypes per class. The described approach also differs from the distance features category as it operates directly on raw time series and not in the features space (also called the dissimilarity space [50]).

The paper is structured as follows: Section 2 describes our contribution — the Eigenresponse Fuzzy Clustering (EFC) method. Section 3 describes the datasets used in this work to evaluate the algorithm. Section 4 evaluates the approach using the datasets defined in the previous Sections. Section 5 presents the conclusions.

Table 1 summarizes and describes the various symbols and indices used in this work. Throughout the paper, matrices (in capital letters) and vectors are shown in bold.

2. Eigenresponse Fuzzy Clustering

The contribution of this article is the introduction of an approach for time series classification — the Eigenresponse Fuzzy Clustering (EFC) method and the application of EFC.

When training a fuzzy model for time series data, usually one prototype per class is usually computed [44–46,48]. However, a single class can encompass a considerable range of values. It would be wise to model such a class with more than one class prototype. For this reason, Eigenresponse Fuzzy Clustering proposes the use of a configurable number of prototypes per class. This concept is illustrated in Fig. 1.

In EFC, classes are represented compactly, retaining only the truly representative shapes — those that define a class. EFC implements this by representing each class with c prototypes. After training, only the prototypes are kept, not the entire training set. The prototypes of class i are learned from all training time series belonging to class i. Since learning is done per class, parallelization is possible when using multi-core CPUs. During classification, only the class prototypes – eigenresponses – are queried, not the entire training data set. The problem of outliers has been

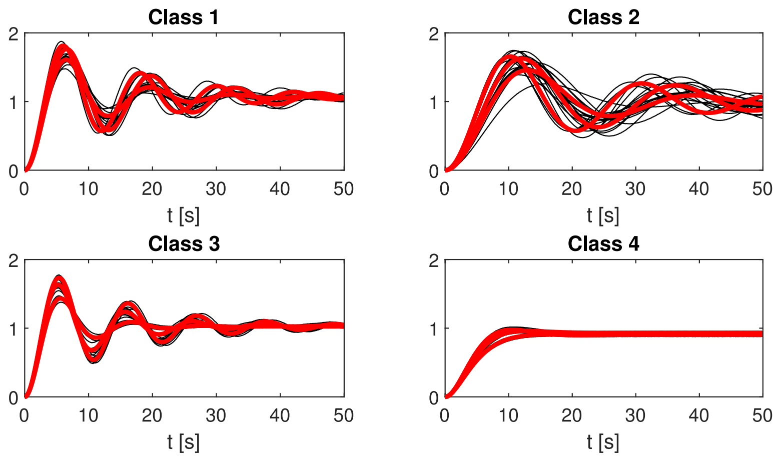


Fig. 1. Time responses of four classes, for each class three ($c = 3$) prototypes are determined.

mentioned. We distinguish two cases of outliers: (1) outlier time series, (2) outlier samples. In the first case, an unusual time series occurs in the training set. In our experience, such outliers have not occurred frequently so far. Resilience to such outliers could be increased by replacing the FCM algorithm with for example, possibilistic c-means; this could be an avenue for further research. The second type is the sample outlier, where a single sample in a time series is irregular. This can be caused by noise or a defective sensor. The proposed algorithm provides improved resilience to this type of outlier, as the outlier is not included in the eigenresponses. Since only a small number of eigenresponses are used to represent a class, visualization is possible and the user can gain insight into the learned model.

In the following sections, FCM is briefly explained (Section 2.1), followed by the introduction of EFC — training the model in Section 2.2 and use for classification in Section 2.3. Then in Section 2.4 the hyperparameters of EFC are discussed and in Section 2.5 the issue of distance measures is addressed.

2.1. Fuzzy C-means

Clustering is an unsupervised learning approach — a process that aims to group a set of data points (samples, objects) into multiple groups (clusters) [51]. Clustering methods can be divided into several categories. The category of partitioning methods aims to partition the sample space into regions associated with each cluster. There are generally two types: crisp and fuzzy.

In crisp methods, each sample x_j is assigned to exactly one cluster [51]. Let μ_i be the membership function to cluster C_i. In crisp methods, μ takes only from $\{0, 1\}$.

Fuzzy methods, on the other hand, allow fuzzy boundaries between clusters. The membership function can take any value in the interval $[0, 1]$. For c clusters ($C = \{C_1, \dots, C_c\}$):

$$\mu_i(\mathbf{x}_j) \in [0, 1]; \quad \sum_{C_i \in C} \mu_i(\mathbf{x}_j) = 1 \tag{1}$$

The Fuzzy C-Means clustering algorithm (FCM) was first introduced in [52], improved in [53], and frequently implemented, adapted, and revised since then.

The algorithm aims to minimize the following objective function

$$I_\eta = \sum_{C_i \in C} \sum_{j=1}^{N_i} \mu_{ij}^{\eta} d(\mathbf{x_j} - \mathbf{v_i}); \quad 1 < \eta < \infty \tag{2}$$

where η is a tuning parameter that determines the fuzziness of the clustering. i is the cluster index, j is the sample index. $\mathbf{v_i}$ is the ith cluster prototype. c is the number of prototypes. μ_{ij} is $\mu_i(x_j)$ - membership function value of sample $\mathbf{x_j}$ to cluster C_i. N is the number of all samples. $d(\mathbf{x_j} - \mathbf{v_i})$ is a distance function, for example squared euclidean $\|\mathbf{x_j} - \mathbf{v_i}\|^2$.

The objective function computes a sum of the weighted distances between all sample–cluster-prototype pairs. The weight is taken from the fuzzy partition matrix, where each element μ_{ij} describes the membership of the sample $\mathbf{x_j}$ to the ith cluster represented by the cluster prototype $\mathbf{v_i}$. The membership function is normalized to the sum of the distances from $\mathbf{x_i}$ to the prototypes of all clusters:

$$\mu_{ij} = \left(\sum_{C_l \in C} \left(\frac{d(\mathbf{x_j} - \mathbf{v_i})}{d(\mathbf{x_j} - \mathbf{v_l})} \right)^{\frac{1}{\eta - 1}} \right)^{-1} \tag{3}$$

FCM is an iterative algorithm that is repeated until it converges or the iteration limit is exceeded.

In FCM, initialization is an important step. The traditional approach is to use random initialization. This may result in more iterations of the algorithm being required or the solution reaching a local-minima solution [54]. We have observed that when using high-dimensional data (i.e., time series), random initialization causes the cluster prototypes to converge against the mean of the data set. Several alternative initialization approaches have been proposed, for example, Fuzzy C-means++ (FCM++) [54]. In FCM++, cluster prototypes are seeded using samples from the dataset. FCM++ aims to select diverse samples as the initial cluster prototypes.

2.2. Training phase

The first step of training is to group the training time series according to their class designations. Each group is then passed to the FCM clustering algorithm

The result of clustering for each class is a set of c cluster prototypes - **eigenresponses**. The total number of prototypes is $c \cdot k$ (k is the number of classes in the classification task). Fig. 1 shows the initial series (black lines) and the prototypes (red lines) for each class (one class per figure quarter).

The eigenresponses for each class are stored in the matrix $\mathbf{V_t}$ and later used to classify new time series. The entire procedure of the train phase is shown in Fig. 2 and in the algorithm:

```
Require: $c, \eta$
1: initialize EFC model
2: for $t = 1 : k$ do
3:     $\mathbf{X}_t \leftarrow \mathbf{X}_{train}$  if  $\mathbf{y}_{train} == t$ {Grouping by class}
```

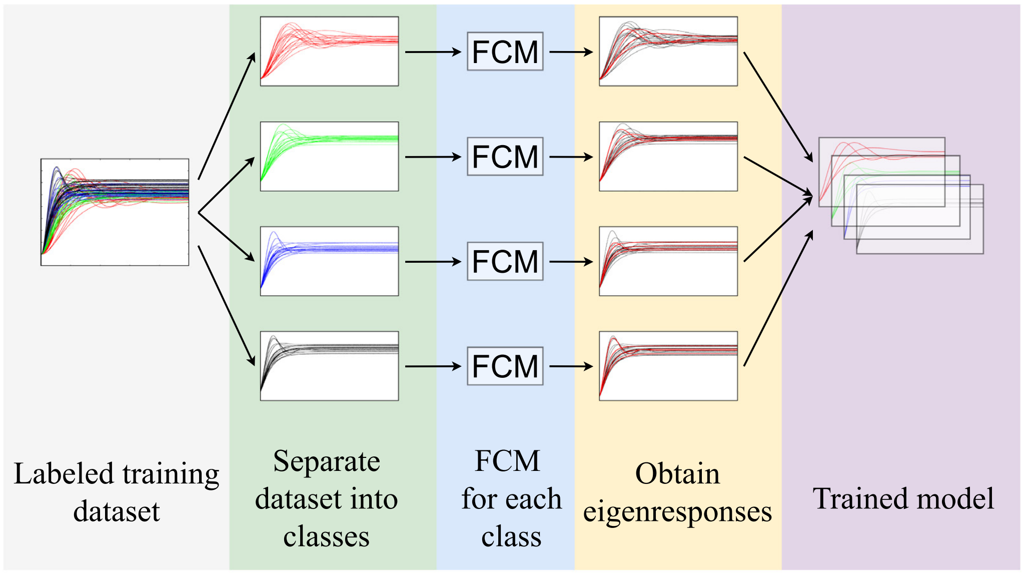


Fig. 2. Learning phase of EFC clustering of eigenresponses. One FCM clustering is performed per class (four classes in above Figure). The cluster prototypes of all FCM clusterings are saved as the trained EFC model — the prototypes are considered eigenresponses.

```
4:     $\mathbf{v}_{t,1}, \ldots, \mathbf{v}_{t,c} \leftarrow \mathrm{FCM}(\mathbf{X}_t, c, \eta)$
5:     $\mathbf{V}_t \leftarrow [\mathbf{v}_{t,1}, \ldots, \mathbf{v}_{t,c}]$
6: end for
7: return $\mathbf{V}_t; t = 1, \ldots, k$
```

Line 3 of the above algorithm is to be interpreted as a selection step, where the training samples of class t are extracted from $\mathbf{X}_{train}$ and assigned to $\mathbf{X}_t$. $\mathbf{X}_t$ is then used in the FCM (line 4).

2.3. Classification phase

The classification is based on the eigenresponses obtained in the training phase (cluster prototypes).

Three approaches to classify a new sample $\mathbf{x_a}$ are proposed: (1) reconstruction-based, (2) distance-based, (3) membership-based. Each of these approaches is explained in the following Sections.

2.3.1. Reconstruction based classification

When a new sample $\mathbf{x_a}$ is received for classification, the stored eigenresponses ($\mathbf{V_t}$; $t = 1, \ldots, k$) are used to reconstruct the given sample. The reconstruction is performed in two steps:

1. By solving the (overdetermined) system of Eqs. (4) for $\mathbf{u_{t,a}}$, an optimal (in terms of Squared Error) weights vector $\mathbf{u_{t,a}}$ is obtained.

   $$\mathbf{x_a} = \mathbf{u_{t,a}}\, \mathbf{V_t} + \mathbf{e_{t,a}} \tag{4}$$

   $\mathbf{x_a}$ is the new sample vector to be classified, $\mathbf{V_t}$ is the matrix of prototypes of class t, $\mathbf{e_{t,a}}$ is the residual error term, $\mathbf{u_{t,a}}$ is the solution — an optimal weights vector for the reconstruction of $\mathbf{x_a}$ using the prototypes $\mathbf{V_t}$. The solution is optimal in minimizing $\mathbf{e_{t,a}}^T\mathbf{e_{t,a}}$. The optimal $\mathbf{u_{t,a}}$ is obtained for all classes $t = 1, \ldots, k$.
2. Among all generated $\mathbf{u_{t,a}}$, the one with the lowest squared reconstruction error $\mathbf{e_{t,a}}^T\mathbf{e_{t,a}}$ is chosen.

   $$\hat{t_a} = \underset{t=1,\ldots,k}{\arg\min}\, \mathbf{e_{t,a}}^T\mathbf{e_{t,a}} \tag{5}$$

   $\hat{t_a}$ is the result — the index of the class in which the given sample $\mathbf{x_a}$ is classified.

The weight vector $\mathbf{u_{t,a}}$ is optimal (in terms of Squared Error) only if the distance metric used for the evaluation is Euclidean distance. This approach should be used appropriately.

2.3.2. Distance based classification

This approach uses the 1-nearest neighbor (1NN) classifier given in algorithm:

```
Require: sample $\mathbf{x_a}$
Require: collection of labelled training samples $\mathbf{X_{train}}$
 1: best_dist $\leftarrow \infty$
 2: for all $\mathbf{x_{train,i}}$ in $\mathbf{X_{train}}$ do
 3:     dist $\leftarrow d(\mathbf{x_{train,i}}, \mathbf{x_a})$
 4:     if dist $<$ best_dist then
 5:         best_idx $\leftarrow i$
 6:         dest_dist $\leftarrow$ dist
 7:     end if
 8: end for
 9: $\hat{y}_a \leftarrow$ class_of($\mathbf{x_{train,best\_idx}}$)
10: return $\hat{y}_a$
```

During the learning phase, c prototype responses were determined for each of the k classes. In distance-based classification approach these prototypes are used as labeled (their class identifier) training samples for a 1NN classifier. For a test sample $\mathbf{x_a}$, the distances to all training samples (class prototypes = eigenresponses) are calculated and the sample is assigned to the class whose prototype was closest.

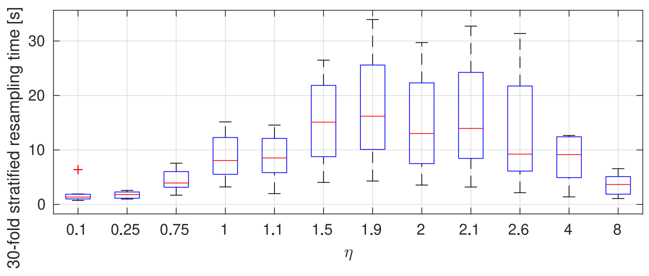
$$\hat{\mathbf{y}}_\mathbf{a} = \underset{t=1,\ldots,k}{\arg\min}\left[\min_{i=1,\ldots,c}\left[d(\mathbf{x_a}, \mathbf{v_{t,i}})\right]\right] \tag{6}$$

2.3.3. Membership-based classification

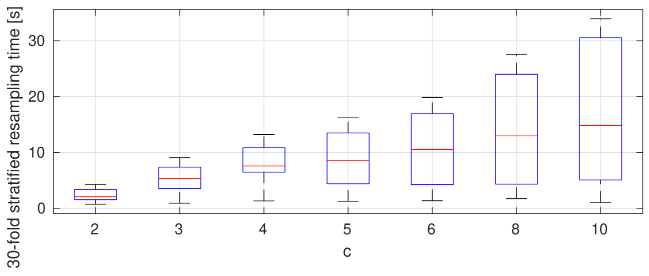
The FCM algorithm uses fuzzy memberships. EFC can make use of them in the classification phase. The membership of sample x_j to cluster i represented by prototype v_i is calculated using (3). For better readability, the equation is re-written here.

$$\mu_{ij} = \left(\sum_{l=1}^{c}\left(\frac{d(\mathbf{x_j}, \mathbf{v_i})}{d(\mathbf{x_j}, \mathbf{v_l})}\right)^{\frac{1}{\eta-1}}\right)^{-1} \tag{7}$$

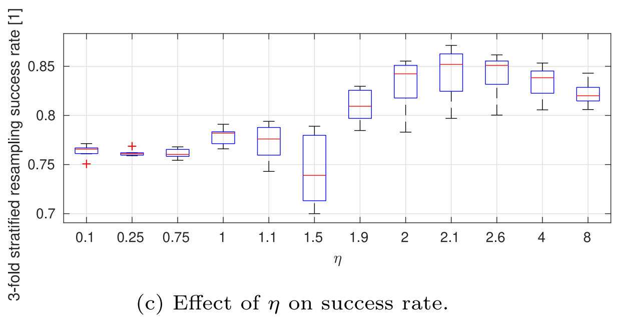
In the proposed EFC model, each class is represented by c prototypes. The prototypes – eigenresponses – can be used to classify a new sample $\mathbf{x_a}$ using appropriate membership functions. Three possible approaches have been developed by the authors of this paper and are presented here:



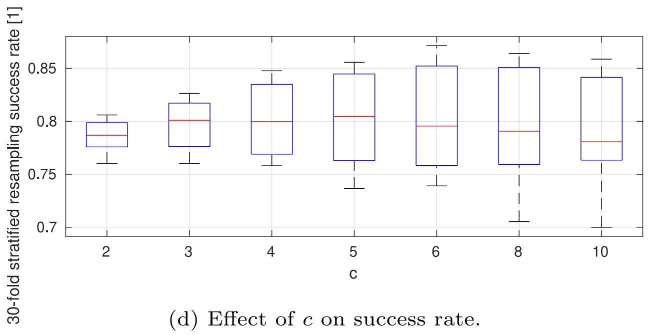
(a) Effect of η on training time.



(b) Effect of c on training time.



(c) Effect of η on success rate.



(d) Effect of c on success rate.

Fig. 3. Hyperparameters η and c effect on training time and success rate using 30 fold stratified resampling on UCR dataset 'ECG200'.

1. **Competing classes** approach is given in (9). For each class t, membership of $\mathbf{x_a}$ are computed to all cluster prototypes $\mathbf{v_{t,i}}$, $i = 1 \dots c$. Maximum memberships across classes are then compared and the class with the highest membership is selected as $\hat{y_a}$.

$$\mu_{t,i,a} = \left(\sum_{l=1}^{c} \left(\frac{d(\mathbf{x_a}, \mathbf{v_{t,i}})}{d(\mathbf{x_a}, \mathbf{v_{t,l}})} \right)^{\frac{1}{\eta-1}} \right)^{-1} \tag{8}$$

$$\hat{y_a} = \underset{t=1,\dots,k}{\arg\max} \left[\max_{i=1,\dots,c} \left[\mu_{t,i,a} \right] \right] \tag{9}$$

2. **Competing prototypes** approach is given in (10). In this approach, all prototypes are considered in the membership calculation. A new sample $\mathbf{x_a}$ is assigned to class t of the best matching $\mathbf{v_{t,i}}$.

$$\hat{y_a} = \underset{\hat{t}=1,\dots,k}{\arg\max} \left[\max_{i=1,\dots,c} \sum_{\substack{l=1 \\ t=1}}^{\substack{c \\ k}} \left(\left(\frac{d(\mathbf{x_a}, \mathbf{v_{\hat{t},i}})}{d(\mathbf{x_a}, \mathbf{v_{t,l}})} \right)^{\frac{1}{\eta-1}} \right)^{-1} \right] \tag{10}$$

3. **Competing aggregates** approach is based on (11). The approach is an extension of competing prototypes, with the difference that instead of individual prototype memberships, those representing the same class ($\mu_{t,i}, t = 1, \dots$) can be summed to represent the **aggregate membership** to a class $\mu_t(\mathbf{x_a}) = \sum_{i=1,\dots,c} \mu_{t,i}(\mathbf{x_a})$.

$$\hat{y_a} = \underset{\hat{t}=1,\dots,k}{\arg\max} \left[\sum_{i=1}^{c} \left(\sum_{\substack{l=1 \\ t=1}}^{\substack{c \\ k}} \left(\frac{d(\mathbf{x_a}, \mathbf{v_{\hat{t},i}})}{d(\mathbf{x_a}, \mathbf{v_{t,l}})} \right)^{\frac{1}{\eta-1}} \right)^{-1} \right] \tag{11}$$

Eqs. (5), (6), (9), (10), (11) describing the five classification approaches, are the fundamental equations of this paper.

Sections 2.2 and 2.3 have introduced the Eigenresponse Fuzzy Clustering algorithm. The trained model consists of $c \cdot k$ eigenresponses. The learnt eigenresponses are then used in classification. Five approaches to classification have been proposed.

2.4. Hyperparameters

The proposed approach includes two hyperparameters: the number of prototypes per class (c in (2)) and the fuzziness parameter (η in (2)). The values used in this paper are $c = 4$, $\eta = 2$. Our choice of c is guided by knowledge of the dataset. $\eta = 2$ is commonly used for the Fuzzy C-Means algorithm and is also used in our evaluation of EFC. A possible future research topic is a data-driven approach for hyperparameter selection, see Fig. 3(a)–3(d) for justification. Currently, the hyperparameters are constant across all classes, which may not be optimal.

2.5. Distance metrics

The algorithm described in Sections 2.2 and 2.3 uses a distance metric on several occasions:

1. the FCM algorithm requires a distance metric for computing distances between objects (samples),
2. the training phase of EFC applies the FCM algorithm,
3. the classification phase of EFC uses a distance metric in all classification approaches (reconstruction-based, distance-based and typicality based).

In this work, $d(\mathbf{x}, \mathbf{y})$ was used to indicate the distance between time series $\mathbf{x}$ and $\mathbf{y}$ of equal length. The time series community has invested considerable effort in developing various (elastic) distance metrics for use in classification. In addition to Euclidean distance, the Dynamic Time Warp distance is the most commonly used. Several authors have proposed additional distance metrics: TWED [23], WDTW [24], DD$_{DTW}$ [25], MSM [26], DD$_{DTW}$ [27], etc.

Since the distance formula is computed numerous times during the training and classification phases, its computational complexity is important. Many of the elastic measures are computationally intensive and therefore their computation is slow compared to, for example, Euclidean distance. We have implemented and used several elastic distances (DTW, cDTW, CID, dDTW, ...), but none of them have led to outstanding performance improvements. Therefore, only DTW, the most popular elastic distance, is discussed in this article. In Tables 5 and 6, we

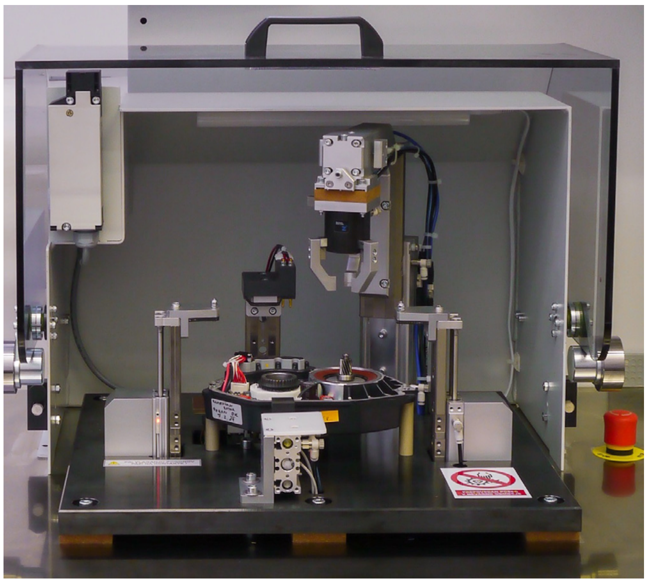


Fig. 4. Industrial machine dataset.

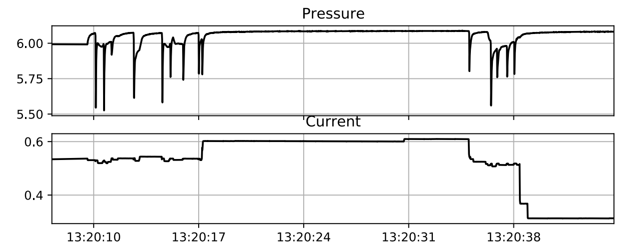


Fig. 5. Pressure and current measurements on EOL testing station — one run.

show that the use of DTW does not lead to improved performance compared to ED. Nevertheless, we include the results to show that distance measures other than ED can be used in combination with the proposed classifier. Researchers working on time series have proposed several elastic measures in recent years. Our inclusion of DTW is to show that other measures can be used and perhaps provide better results for specific use cases.

In the rest of this paper, we will mainly use ED and DTW. However, it should be noted that DTW and most other elastic measures are not in fact metrics, and should be used accordingly.

In this Section the EFC algorithm was introduced in two distinct phases — learning and classification. For classification, five approaches were developed and presented. In the next Section the datasets are presented, and later in Section 4 the algorithm is evaluated.

3. Datasets

The algorithm, described in Section 2 is evaluated on multiple datasets:

1. An industrial machine case study — 280 measurements from 14 different classes. Each class corresponds to an action on an industrial, automated, end-of-line test bench, on e-bike drive assembly line developed at the "Jožef Stefan" Institute [10]. Data of 20 test runs is used, the dataset is described in more detail in Section 3.1.
2. A selection of publicly available time series datasets, as specified in Section 3.2.

Table 2
Industrial machine dataset labels and descriptions.

Index	Label	Description
1	S1	Hood lock
2	S3	Center apply 1
3	S4	Restraint apply
4	S5	Contact apply
5	S6–9	Center remove 1, test current on, measure 1, test current off
6	S10	Center apply 2
7	S11	Holder apply
8	S12	Axis cylinder apply
9	S13	Bearing grip
10	S14–17	Axis cylinder remove, power supply connect, measure 2, power supply disconnect
11	S18–19	Bearing release, contact remove
12	S20	Center remove 2
13	S21	Holder remove
14	S22	Restraint remove

This Section describes the datasets used to evaluate the proposed classification algorithm. The 'industrial machine dataset' is the primary-target use case, and the UCR archive is used as a benchmark to compare the performance of the presented approach. Since the used industrial machine dataset has not yet been published elsewhere, the dataset is described in more detail, including the pre-processing steps used to prepare it (normalization, segmentation, etc.).

3.1. Industrial EOL machine dataset

The work presented in this paper is driven by the desire to detect events in a real-world industrial process using a ubiquitous signal such as supply current or pneumatic line pressure. With this in mind, the approach is evaluated using a real-world dataset of 280 time series. This Section describes the generating process and the preprocessing steps performed.

Originating process. An industrial end of line (EOL) testing station was developed at the "Jožef Stefan" Institute. The station is specialized in testing e-bike motors and is shown in Fig. 4. The main testing procedure is preceded and followed by a sequence of rapid actuator actions. These actions include clamping the device on multiple sides, lowering the test equipment onto the motor, etc. A total of 14 detectable actions were manually identified. These events are listed in Table 2.

The EOL station was equipped with additional sensors. The pressure sensor is of importance for this article. The observed actuators on the EOL testing station are electrically controlled but pneumatically driven. Therefore, every action of these actuators has an effect on the line pressure, which can be seen in the collected measurements — see Fig. 5. An extensive set of measurements was made, spanning several months. From these measurements, a set of 20 'runs' was randomly selected. A 'run' is an entire sequence of events necessary for the testing station to process (test) a single electric motor. Runs with incomplete or invalid data were ignored.

Normalization. The Z-normalization (13) is commonly used in machine learning applications, but for our data, the min–max normalization (12) was chosen for two reasons:

1. The distribution of pressure measurements is not normal, it is skewed, as shown in Fig. 6. There are long tails (see values < 6.0) and an additional peak is visible (between 6.1 and 6.2). The line pressure is usually bounded upward by the central pressure lines of the plant, resulting in the observed peak.

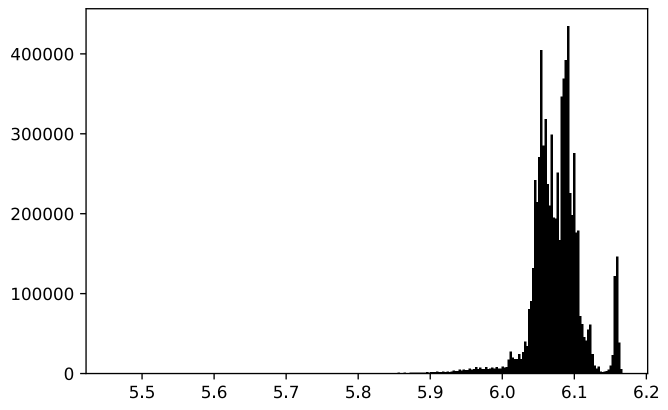


Fig. 6. Histogram of raw pressure measurements. All available measurements used (not only those included in the dataset). Number of bins = 250.

Table 3
Industrial machine dataset duration distribution.

Index	Name	Mean [s]	Std [s]	% of total (mean)
1	S1	0.498	0.045	0.57%
2	S3	0.541	0.021	0.62%
3	S4	0.485	0.004	0.56%
4	S5	1.446	0.034	1.66%
5	S6–9	1.867	0.025	2.14%
6	S10	0.541	0.029	0.62%
7	S11	0.829	0.006	0.95%
8	S12	1.009	0.005	1.16%
9	S13	0.222	0.006	0.25%
10	S14–17	76.879	10.224	88.12%
11	S18–19	1.421	0.007	1.63%
12	S20	0.396	0.005	0.45%
13	S21	0.642	0.005	0.74%
14	S22	0.469	0.007	0.54%

2. A run contains the actions of the actuators before and after the test that one wants to analyze further. However, it also contains measurements taken during the test. These vary greatly in length (see Table 3) and make up a large portion of the measurements of the entire run, and would have a disproportionate effect if one were to use z-normalization of the entire run data — see long period of stationary conditions in Fig. 5 and row 10 (S14–17) in Table 3.

It is of note that at this point the entire run is normalized, not responses to individual actuator actions (classes)

$$x(t) = \frac{x_{raw}(t) - min(x_{raw})}{max(x_{raw}) - min(x_{raw})} \tag{12}$$

$$x(t) = \frac{x_{raw}(t) - mean(x_{raw})}{std(x_{raw})} \tag{13}$$

Segmentation. After normalization, each run is divided into the 14 actions. A significant drop in line pressure indicates an action and is suitable for segmentation. Out of 20 runs, 280 segments are retrieved. Each segment is labeled with an appropriate label. The entire dataset was manually reviewed to ensure correct segmentation and labeling. Fig. 7 shows the derivative-based segmentation. The segments are shown in alternating colors.

The segmentation level can be chosen as follows:

1. calculate the derivative of the data,
2. determine the noise level in the measurements,
3. calculate the standard deviation of the measurements whose derivative is greater than the noise level,
4. the calculated standard deviation can be used as the segmentation threshold.

Segmentation is a fundamental part of data processing prior to segment classification. In real-world scenarios, this is a difficult task. The broader problem of time series segmentation is beyond the scope of this paper, but should be addressed in further research in this area.

Length correction. The raw sensor data was sampled at 100 Hz. The segments obtained vary in length. Most time series analysis works (best) on time series of equal length, so all segments are length corrected. They are resampled to 500 data points. Where necessary, missing data points are obtained through linear interpolation of the two adjacent data points.[1]

Nomenclature. In the remainder of this paper, the segments obtained here are referred to as 'responses', and their associated labels are 'task' or 'class' indicators.

The final dataset is shown in Fig. 8.

Indistinguishable classes. The classes labeled S3 and S10 are indistinguishable, so they are considered one interchangeable class for the sake of this article. This brings the total number of classes from 14 to 13.

Train test breakdown. The following Section introduces the UCR Archive datasets. They are all pre-split into training and test datasets. To simplify the work with the Industrial machine dataset, it is also split into two sets. In the original dataset, all classes are equally represented. This equal representation is also ensured in the generated training and test datasets. Apart from the requirement of equal representation, the selection of the training and test sets is done randomly.

A single split between training and test series was made, and this split is used throughout this paper.

3.2. UCR archive

In 2002, researchers at the University of California, Riverside, published an archive (referred to here as the UCR Archive, or simply the archive) of 16 datasets for use in time series classification. Over the years, the archive has grown and been expanded several times, most recently in 2018 [28,55]. The archive is now very diverse and includes 128 datasets.

All datasets are already divided into training and test sets. The authors of the archive also provide detailed descriptions of most of the data sets.

The archive includes a wide range of time series types: Shape contours (e.g., 'ArrowHead'), spectral measurements (e.g., 'Coffee', 'Beef'), simulated (e.g., 'TwoPatterns'), medical (e.g. 'TwoLeadECG', 'NonInvasiveFetalECGThorax1'), traffic count (e.g. 'DodgerLoopGame, 'MelbournePedestrian'), power consumption (e.g. 'ItalyPowerDemand', 'LargeKitchenAppliances'), and many more. Some of these categories may surprise the reader (outlines of shapes, spectral measurements) because they are not temporal data. However, the defining characteristic of time series over other types of data is that the attributes are ordered. Whether the order is temporal or not is in fact irrelevant [16]. The shape contours, also called the outlines or, simply 'images' category is the most similar to the time series relevant to our ultimate goal of detecting machine operations based on shapes.

In URC Arhive, individual datasets contain between 2 and 128 classes. Some datasets are long (2844 samples per time series in

[1] From the Table 3 it can be seen that the segments are very well distinguishable by their length alone. However, this is usually not the case in other real-world applications. After length correction, the information about the original length of a segment is no longer present in the dataset, and the evaluation results in Section 4 are therefore not biassed by this property of the raw data.

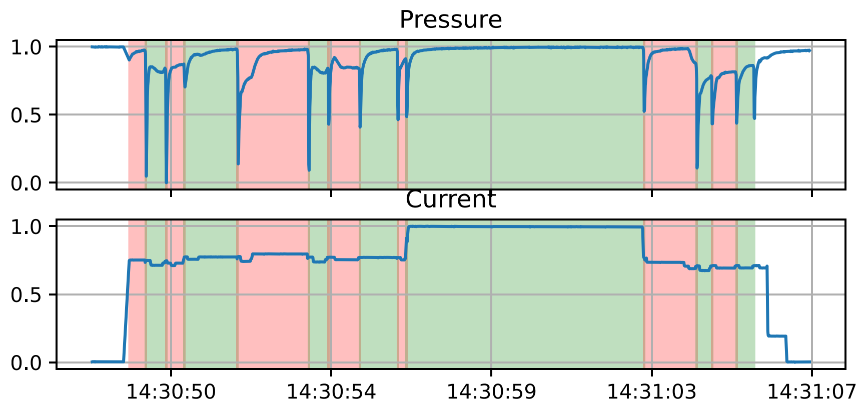


Fig. 7. Example of a segmented time series in Industrial machine dataset.

the 'Rock' record), others are very short (15 samples per time series in the 'SmoothSubspace' dataset). The archive also contains data sets with different time series lengths and even data sets with missing values.

All 128 datasets were analyzed and 48 of them were selected for further evaluation. The selection process could be described as one of elimination. A list of conditions was established. If a record did not meet one of these conditions, it would not be included in the final selection:

- The size of the training and testing datasets must be 500 or less. Larger sets lead to very long learning and evaluation times and were therefore not practical for us, especially because of the inclusion of edit-distance based distance measures.
- data sets with time series longer than 512 samples each were not included. The reason for this omission was that longer time series lead to time-consuming calculations of certain edit-distance-based distance measures.
- The algorithm as proposed in this paper defines 4 (our value for c) eigenresponses per class. Datasets with fewer than 4 exemplars of each class in the training set were therefore discarded. This condition must be satisfied for all classes represented in the training data.

The elimination process discarded a large number of UCR datasets, leaving 48 datasets for evaluation.

4. Results

In the previous Sections EFC was discussed. In this Section the algorithm is evaluated and compared to the most commonly used classifier for whole series classification.

4.1. Setup and methodology

This Section explains the experimental setup, special conditions, and the preprocessing used.

4.1.1. Z-normalization

Z-normalization (13) is commonly used in machine learning applications. In Section 3.1, min–max normalization was used instead. However, it was noted that min–max normalization was only used per run. In this Section z-normalization is applied to each individual time series — each class response. This normalization is used for both the Industrial machine dataset and all UCR datasets used.

Table 4
Performance on Industrial machine dataset.

	1NN	EFC
20-fold cross validation accuracy	97.5%	100%

4.1.2. Ensemble of EFC classifiers

Section 2.3 presents five classification approaches: reconstruction-based, distance-based, competing prototypes, competing classes, and competing aggregates. When processing a new dataset split into training and testing data, the training dataset can be used to determine the best performing approach. Assuming that the performance in the test and training datasets is highly correlated, the best approach can be selected as the best for that dataset.

The classifier that uses a classification approach determined in the manner described above is referred to as the "EFC best" in the Results section.

Other advanced ensembling techniques, such as weighted voting, should be explored in the future.

4.2. Industrial machine dataset performance

The Industrial machine dataset described in Section 3.1 was used to evaluate the performance of the proposed algorithm. Both the 1NN and EFC classifiers were used, both with ED. For EFC, $c = 4, \eta = 2$ were used. Results were obtained by performing 20-fold cross-validation.

On the industrial dataset, the EFC classifier achieves 95.4% accuracy. Under the same conditions, the 1NN classifier achieves 88.9%. However, closer inspection shows that the EFC classifier makes errors between classes S3 and S10, which are indistinguishable from each other, see Section 3.1. If S3 and S10 are considered interchangeable, as described in Section 3.1, the EFC achieves 100% accuracy and the 1NN achieves 97.5% accuracy. The results are summarized Table 4. The accuracy of 100% is crucial in the field of manufacturing.

Thus, the proposed algorithm achieves its main goal by correctly classifying all time series from our case study on industrial machinery. This leads us to compare the EFC with other more general datasets. In the next section, we therefore further evaluate the algorithm on a set of publicly available datasets from the UCR archive.

4.3. UCR-archive performance

This section evaluates the proposed approach and compares it to the 1NN classifier using a number of datasets. The hypothesis

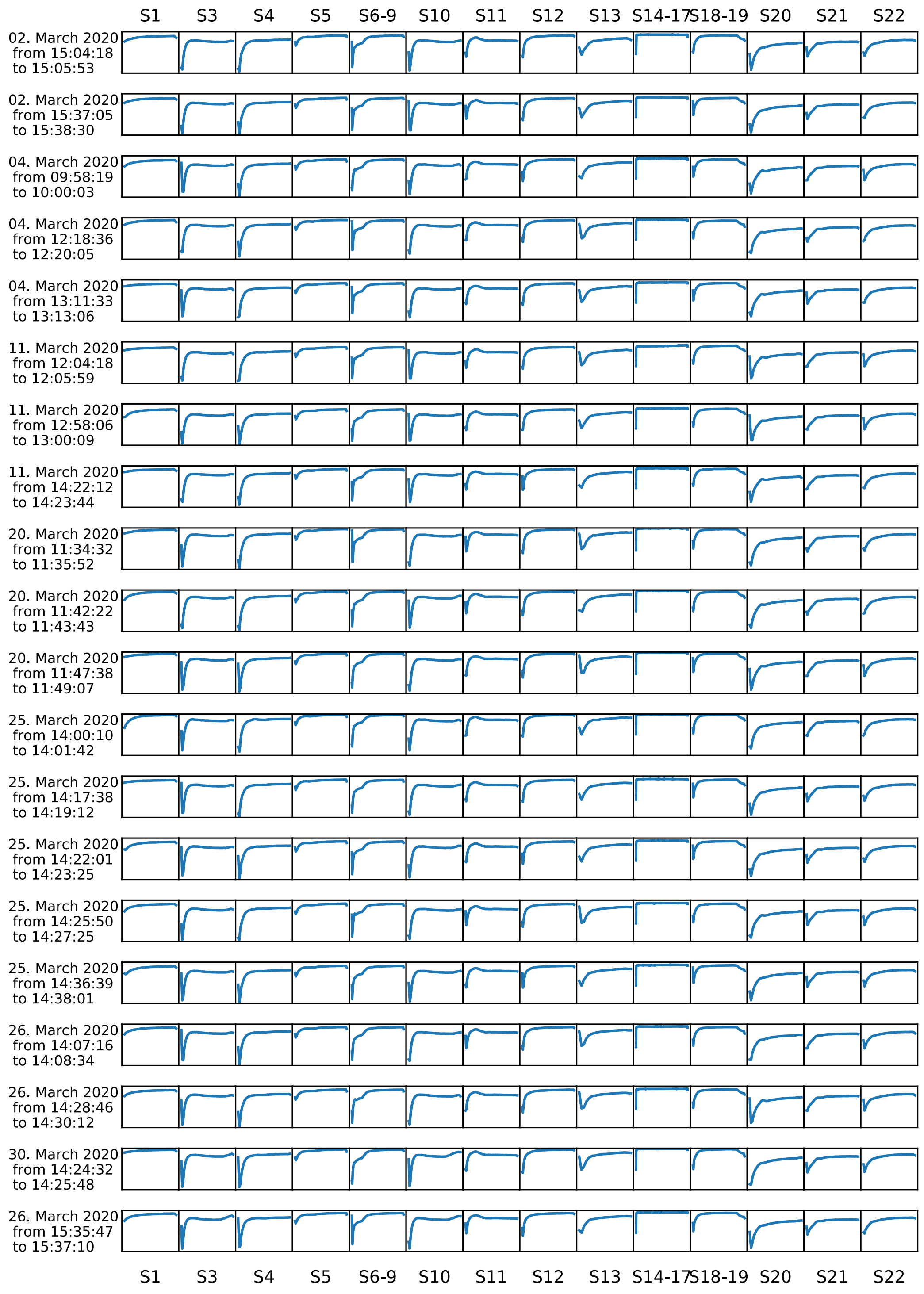


Fig. 8. Industrial machine dataset — 20 runs.

tested here is that there are a number of time series datasets where the proposed approach outperforms the most commonly used whole series approach, the 1NN classifier. The comparison made in this section focuses on the two distance measures most commonly used in time series classification: the Euclidean distance (ED) and the Dynamic Time Warp distance (DTW).

Fig. 9 shows performance using ED or DTW, the two most commonly used distance metrics in time series classification. The red dots show the results obtained using ED and the magenta dots represent DTW. Each dot shows the results for one dataset. Fig. 9 shows that the proposed approach performs better in 46.9% datasets with ED. Moreover, Fig. 10 that there is a correlation between training and testing results. This allows us to correctly predict whether 1NN or the proposed approach will perform better. The Tables 5 and 6 show whether test performance for a single dataset was predicted correctly (TP, TN) or incorrectly (FP,

Table 5
1NN vs EFC 'best', Euclidean distance.

Dataset	UCR category	Performance
Industrial machine dataset	–	TP
ECG200	ECG	TN
ArrowHead	Shape/Image	TP
BeetleFly	Shape/Image	TP
BirdChicken	Shape/Image	TP
DistalPhalanxOutlineAgeGrp	Shape/Image	TP
DistalPhalanxTW	Shape/Image	TP
Fish	Shape/Image	TP
Herring	Shape/Image	TP
MiddlePhalanxOutlineAgeGrp	Shape/Image	TP
MiddlePhalanxTW	Shape/Image	TP
OSULeaf	Shape/Image	TN
ProximalPhalanxOutlineAgeGrp	Shape/Image	FP
ProximalPhalanxTW	Shape/Image	TP
CricketX	Motion	TN
CricketY	Motion	TN
CricketZ	Motion	TN
GunPoint	Motion	TN
GunPointAgeSpan	Motion	TN
GunPointMaleVersusFemale	Motion	TN
GunPointOldVersusYoung	Motion	TN
ToeSegmentation1	Motion	FP
ToeSegmentation2	Motion	FP
PowerCons	Power	TN
DodgerLoopDay	Sensor	FP
DodgerLoopGame	Sensor	TP
DodgerLoopWeekend	Sensor	TP
Earthquakes	Sensor	FP
GesturePebbleZ1	Sensor	TP
GesturePebbleZ2	Sensor	TN
Lightning7	Sensor	FN
PickupGestureWiimoteZ	Sensor	TP
Plane	Sensor	TP
ShakeGestureWiimoteZ	Sensor	TP
Trace	Sensor	TN
BME	Simulated	FP
ShapeletSim	Simulated	FP
SmoothSubspace	Simulated	TN
SyntheticControl	Simulated	TP
UMD	Simulated	FP
Beef	Spectro	TP
Coffee	Spectro	TP
Ham	Spectro	NT
Meat	Spectro	FP
Wine	Spectro	FN
Chinatown	Traffic	TP
GestureMidAirD1	Trajectory	FP
GestureMidAirD2	Trajectory	TN
GestureMidAirD3	Trajectory	TN

Table 6
1NN vs EFC 'best', DTW distance.

Dataset	UCR category	Performance
Industrial machine dataset	–	TP
ECG200	ECG	TN
ArrowHead	Shape/Image	TP
BeetleFly	Shape/Image	TP
BirdChicken	Shape/Image	TN
DistalPhalanxOutlineAgeGrp	Shape/Image	TN
DistalPhalanxTW	Shape/Image	FP
Fish	Shape/Image	TN
Herring	Shape/Image	FP
MiddlePhalanxOutlineAgeGrp	Shape/Image	FP
MiddlePhalanxTW	Shape/Image	TP
OSULeaf	Shape/Image	TN
ProximalPhalanxOutlineAgeGrp	Shape/Image	TN
ProximalPhalanxTW	Shape/Image	TN
CricketX	Motion	TN
CricketY	Motion	TN
CricketZ	Motion	TN
GunPoint	Motion	FP
GunPointAgeSpan	Motion	TN
GunPointMaleVersusFemale	Motion	TN
GunPointOldVersusYoung	Motion	FN
ToeSegmentation1	Motion	FP
ToeSegmentation2	Motion	TN
PowerCons	Power	FN
DodgerLoopDay	Sensor	FP
DodgerLoopGame	Sensor	FP
DodgerLoopWeekend	Sensor	FP
Earthquakes	Sensor	TN
GesturePebbleZ1	Sensor	TN
GesturePebbleZ2	Sensor	TN
Lightning7	Sensor	TN
PickupGestureWiimoteZ	Sensor	FP
Plane	Sensor	TN
ShakeGestureWiimoteZ	Sensor	FP
Trace	Sensor	TN
BME	Simulated	FP
ShapeletSim	Simulated	FP
SmoothSubspace	Simulated	TN
SyntheticControl	Simulated	TN
UMD	Simulated	FP
Beef	Spectro	TP
Coffee	Spectro	TP
Ham	Spectro	FN
Meat	Spectro	FP
Wine	Spectro	FN
Chinatown	Traffic	TP
GestureMidAirD1	Trajectory	FP
GestureMidAirD2	Trajectory	TN
GestureMidAirD3	Trajectory	FP

FN) by observing the training results. For Euclidean distance, the F1 score is 0.778 and for DTW 0.424.

4.3.1. Shape outlines

The UCR archive contains datasets that are divided into several categories. One of the categories is outlines. Time series of outlines are obtained by mapping the outline of an object to polar coordinates. The resulting time series has similar properties to the Industrial machine dataset in Section 3.1. 12 of the datasets used in the evaluation belong to the category 'outlines of shapes', also called 'images': ArrowHead, BeetleFly, BirdChicken, DistalPhalanxOutlineAgeGroup, DistalPhalanxTW, Fish, MiddlePhalanxOutlineAgeGroup, MiddlePhalanxTW, OSULeaf, Herring, ProximalPhalanxOutlineAgeGroup, ProximalPhalanxTW. Out of the 12 datasets, our proposed method using ED improved the 1NN classifier in 10 datasets. The results corresponding to the 12 datasets are shown as outlined dots in Fig. 9.

5. Conclusion

This paper presents an approach for time series classification — Eigenresponse Fuzzy Clustering (EFC). EFC proposes learning multiple time series prototypes per class to better model the wider range of each class. Training of the EFC model is performed by fuzzy clustering with Fuzzy C-Means. In this paper, we have presented five methods for classification using the multiple-prototypes-per-class stored by the EFC model.

The developed EFC algorithm primarily aims to detect events in production and enables monitoring and fault detection. Autonomous fault detection is becoming an essential task in production monitoring. Therefore, an efficient and reliable method is of great importance. The results on a real-world dataset show that the method is very accurate for the target application.

EFC is further evaluated on a selection of publicly available datasets, the UCR Archive. This further evaluation shows that the algorithm is competitive on some more complex datasets. However, as mentioned in the previous sections, there is still room for improvement. Among the UCR Archive datasets, the

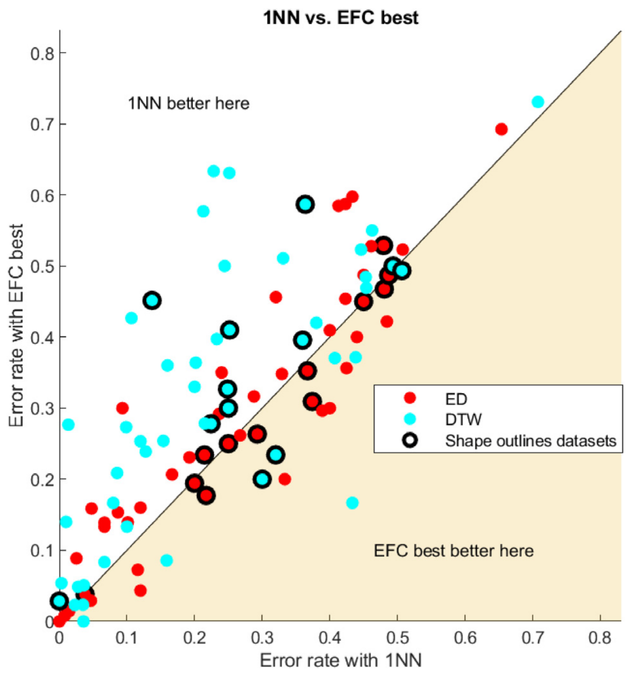


Fig. 9. Comparison of EFC 'best' and 1NN. Each point is a single dataset–distance combination. The two axes are the error rates of 1NN and EFC. The colors of the points distinguish between the Euclidean distance and the Dynamic Time Warp distance. Points below the diagonal line indicate dataset–distance combinations where the EFC approach performs better than 1NN. The outlined dots indicate the shape outlines category of the UCR Archive. For this category, EFC-ED gives better performance than 1NN- ED.

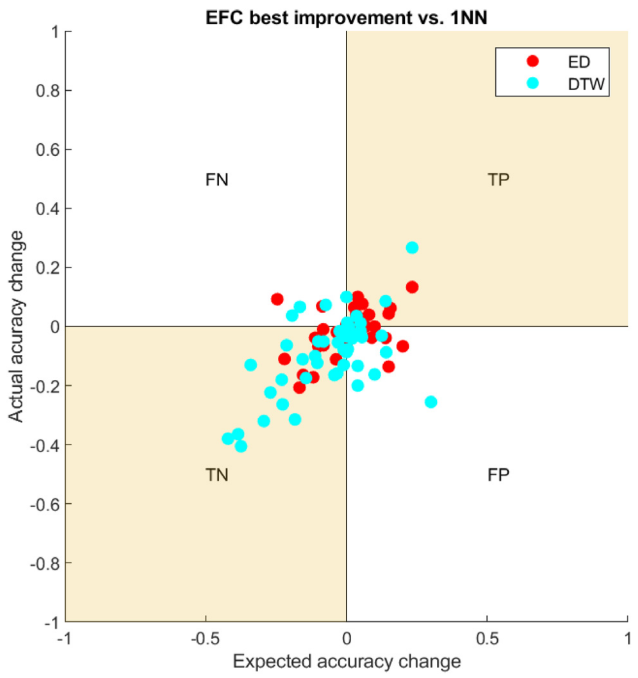


Fig. 10. Comparison of EFC 'best' and 1NN — a texas sharpshooter plot (FN = False Negative, FP = False Positive, TN = True Negative, TP = True Positive).

category of shape outlines (images) stands out as being the most similar to the desired application of the proposed method. In this category, the method outperforms the most commonly used whole time series classification approach, the nearest neighbor.

CRediT authorship contribution statement

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

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

The authors acknowledge the research core funding No. P2-0001 that was financially supported by the Slovenian Research Agency.

References

[1] D. K. E. Deutsche Kommission Elektrotechnik, German standardization roadmap industrie 4.0., 4th edition, 2020, https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/Normungsroadmap_I4.0.html.
[2] Commission for economic policy, E. U., The future of industry in Europe, 2017, https://cor.europa.eu/en/engage/studies/Documents/The%20future%20of%20industry%20in%20Europe/future-of-industry.pdf.
[3] A.V. Bogoviz, A.A. Kurilova, T.E. Kozhanova, A.A. Sozinova, Artificial intelligence as the core of production of the future: Machine learning and intellectual decision supports, in: Advances in Mathematics for Industry 4.0, Elsevier, 2021, pp. 235–256.
[4] P. Burggräf, J. Wagner, B. Koke, M. Bamberg, Performance assessment methodology for AI-supported decision-making in production management, Procedia CIRP 93 (2020) 891–896.
[5] C. Truong, L. Oudre, N. Vayatis, Selective review of offline change point detection methods, Signal Process. 167 (2020) 107299.
[6] S. Gharghabi, Y. Ding, C.-C.M. Yeh, K. Kamgar, L. Ulanova, E. Keogh, Matrix profile VIII: Domain agnostic online semantic segmentation at superhuman performance levels, in: 2017 IEEE International Conference on Data Mining, ICDM, IEEE, 2017, pp. 117–126.
[7] Y. Matsubara, Y. Sakurai, C. Faloutsos, Autoplait: Automatic mining of co-evolving time sequences, in: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, 2014, pp. 193–204.
[8] T.-c. Fu, F.-l. Chung, R. Luk, C.-m. Ng, Representing financial time series based on data point importance, Eng. Appl. Artif. Intell. 21 (2) (2008) 277–300.
[9] E.J.d.S. Luz, W.R. Schwartz, G. Cámara-Chávez, D. Menotti, ECG-based heartbeat classification for arrhythmia detection: A survey, Comput. Methods Programs Biomed. 127 (2016) 144–164.
[10] B. Pregelj, A. Debenjak, G. Dolanc, J. Petrovčič, B. Benedičič, Diagnostic system for end-of-line quality control of pedalec electric bike drives, in: Procceedings of the 11. AIG Conference, Maribor, Slovenia, 2019.
[11] Ž. Stržinar, Modelling and fault detection in heating, ventilation, and air conditioning systems, University of Ljubljana, Faculty of Electrical Engineering, 2017, Original title: Modeliranje in zaznavanje napak v klimatskih sistemih.
[12] G. Andonovski, S. Blažič, I. Škrjanc, Partial cloud-based evolving method for fault detection of HVAC system, in: 2018 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE, IEEE, 2018, pp. 1–6.
[13] B. Ellert, S. Makonin, F. Popowich, Appliance water disaggregation via non-intrusive load monitoring (NILM), in: Smart City 360°, Springer, 2016, pp. 455–467.
[14] V. Guralnik, J. Srivastava, Event detection from time series data, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 33–42.
[15] R. Mercer, S. Ucar, E. Keogh, Shape-based telemetry approach for distracted driving behavior detection, in: 2021 IEEE Conference on Standards for Communications and Networking, CSCN, IEEE, 2021, pp. 118–123.
[16] A. Bagnall, J. Lines, A. Bostrom, J. Large, E. Keogh, The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Discov. 31 (3) (2017) 606–660.

[17] Ž. Stržinar, B. Pregelj, I. Škrjanc, Overview of some methods in the field of time series analysis, in: Proceedings of the 29. International Electrotechnical and Computer Science Conference ERK 2020, PortoroŽ, Slovenia, 2020, Original title: Pregled nekaterih metod na področju analize časovnih vrst.
[18] A.P. Ruiz, M. Flynn, J. Large, M. Middlehurst, A. Bagnall, The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Discov. 35 (2) (2021) 401–449.
[19] A. Abanda, U. Mori, J.A. Lozano, A review on distance based time series classification, Data Min. Knowl. Discov. 33 (2) (2019) 378–412.
[20] D.J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series., in: KDD Workshop, Vol. 10, Seattle, WA, USA, 1994, pp. 359–370.
[21] S. Salvador, P. Chan, Toward accurate dynamic time warping in linear time and space, Intell. Data Anal. 11 (5) (2007) 561–580.
[22] R. Wu, E.J. Keogh, FastDTW is approximate and generally slower than the algorithm it approximates, IEEE Trans. Knowl. Data Eng. (2020).
[23] P.-F. Marteau, Time warp edit distance with stiffness adjustment for time series matching, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2) (2008) 306–318.
[24] Y.-S. Jeong, M.K. Jeong, O.A. Omitaomu, Weighted dynamic time warping for time series classification, Pattern Recognit. 44 (9) (2011) 2231–2240.
[25] T. Górecki, M. Łuczak, Using derivatives in time series classification, Data Min. Knowl. Discov. 26 (2) (2013) 310–331.
[26] A. Stefan, V. Athitsos, G. Das, The move-split-merge metric for time series, IEEE Trans. Knowl. Data Eng. 25 (6) (2012) 1425–1438.
[27] T. Górecki, M. Łuczak, Non-isometric transforms in time series classification using DTW, Knowl.-Based Syst. 61 (2014) 98–108.
[28] H.A. Dau, E. Keogh, K. Kamgar, C.-C.M. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, Hexagon-ML, The UCR time series classification archive, 2018, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.
[29] H. Deng, G. Runger, E. Tuv, M. Vladimir, A time series forest for classification and feature extraction, Inform. Sci. 239 (2013) 142–153.
[30] M.G. Baydogan, G. Runger, E. Tuv, A bag-of-features framework to classify time series, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2796–2802.
[31] M.G. Baydogan, G. Runger, Time series representation and similarity based on local autopatterns, Data Min. Knowl. Discov. 30 (2) (2016) 476–509.
[32] J. Large, A. Bagnall, S. Malinowski, R. Tavenard, On time series classification with dictionary-based classifiers, Intell. Data Anal. 23 (5) (2019) 1073–1089.
[33] J. Faouzi, Time series classification: A review of algorithms and implementations, in: K. Kotecha (Ed.), Machine Learning (Emerging Trends and Applications), Proud Pen, 2022, URL https://hal.inria.fr/hal-03558165.
[34] T. Rakthanmanon, E. Keogh, Fast shapelets: A scalable algorithm for discovering time series shapelets, in: Proceedings of the 2013 SIAM International Conference on Data Mining, SIAM, 2013, pp. 668–676.
[35] A. Bostrom, A. Bagnall, Binary shapelet transform for multiclass time series classification, in: International Conference on Big Data Analytics and Knowledge Discovery, Springer, 2015, pp. 257–269.
[36] J. Grabocka, N. Schilling, M. Wistuba, L. Schmidt-Thieme, Learning time-series shapelets, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 392–401.
[37] J. Lin, R. Khade, Y. Li, Rotation-invariant similarity in time series using bag-of-patterns representation, J. Intell. Inf. Syst. 39 (2) (2012) 287–315.
[38] P. Senin, S. Malinchik, Sax-vsm: Interpretable time series classification using sax and vector space model, in: 2013 IEEE 13th International Conference on Data Mining, IEEE, 2013, pp. 1175–1180.
[39] P. Schäfer, The BOSS is concerned with time series classification in the presence of noise, Data Min. Knowl. Discov. 29 (6) (2015) 1505–1530.
[40] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with COTE: The collective of transformation-based ensembles, IEEE Trans. Knowl. Data Eng. 27 (9) (2015) 2522–2535.
[41] J. Lines, S. Taylor, A. Bagnall, Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification, in: 2016 IEEE 16th International Conference on Data Mining, ICDM, IEEE, 2016, pp. 1041–1046.
[42] A. Bagnall, M. Flynn, J. Large, J. Lines, M. Middlehurst, On the usage and performance of the hierarchical vote collective of transformation-based ensembles version 1.0 (hive-cote v1. 0), in: International Workshop on Advanced Analytics and Learning on Temporal Data, Springer, 2020, pp. 3–18.
[43] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, A. Bagnall, HIVE-COTE 2.0: A new meta ensemble for time series classification, Mach. Learn. 110 (11) (2021) 3211–3243.
[44] A. Javed, B.S. Lee, D.M. Rizzo, A benchmark study on time series clustering, Mach. Learn. Appl. 1 (2020) 100001.
[45] H. Izakian, W. Pedrycz, I. Jamal, Fuzzy clustering of time series data using dynamic time warping distance, Eng. Appl. Artif. Intell. 39 (2015) 235–244.
[46] M. Ji, F. Xie, Y. Ping, A dynamic fuzzy cluster algorithm for time series, in: Abstract and Applied Analysis, Vol. 2013, Hindawi, 2013.
[47] P. D'Urso, L. De Giovanni, M. Disegna, R. Massari, Fuzzy clustering with spatial–temporal information, Spatial Stat. 30 (2019) 71–102.
[48] P. Ravikumar, V.S. Devi, Fuzzy classification of time series data, in: 2013 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE, IEEE, 2013, pp. 1–6.
[49] S. Policker, A.B. Geva, Nonstationary time series analysis by temporal clustering, IEEE Trans. Syst. Man Cybern. B 30 (2) (2000) 339–343.
[50] B.K. Iwana, V. Frinken, K. Riesen, S. Uchida, Efficient temporal pattern recognition by means of dissimilarity space embedding with discriminative prototypes, Pattern Recognit. 64 (2017) 268–276.
[51] J. Han, J. Pei, M. Kamber, Data Mining: Concepts and Techniques, Elsevier, 2011.
[52] J.C. Dunn, A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters, Taylor & Francis, 1973.
[53] J.C. Bezdek, R. Ehrlich, W. Full, FCM: The fuzzy C-means clustering algorithm, Comput. Geosci. 10 (2–3) (1984) 191–203.
[54] A. Stetco, X.-J. Zeng, J. Keane, Fuzzy C-means++: Fuzzy C-means with effective seeding initialization, Expert Syst. Appl. 42 (21) (2015) 7541–7548.
[55] H.A. Dau, A.J. Bagnall, K. Kamgar, C.M. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, E.J. Keogh, The UCR time series archive, 2018, CoRR abs/1810.07758, arXiv:1810.07758.