Similarity Measure Selection for Clustering Time Series Databases

Usue Mori, Alexander Mendiburu, and Jose A. Lozano, *Member, IEEE*

Abstract—In the past few years, clustering has become a popular task associated with time series. The choice of a suitable distance measure is crucial to the clustering process and, given the vast number of distance measures for time series available in the literature and their diverse characteristics, this selection is not straightforward. With the objective of simplifying this task, we propose a multi-label classification framework that provides the means to automatically select the most suitable distance measures for clustering a time series database. This classifier is based on a novel collection of characteristics that describe the main features of the time series databases and provide the predictive information necessary to discriminate between a set of distance measures. In order to test the validity of this classifier, we conduct a complete set of experiments using both synthetic and real time series databases and a set of five common distance measures. The positive results obtained by the designed classification framework for various performance measures indicate that the proposed methodology is useful to simplify the process of distance selection in time series clustering tasks.

Index Terms—Time series, distance measures, clustering, multi-label classification

1 Introduction

In recent years, the increase in data collecting devices has enabled access to a large amount of temporal data (or time series data). This has generated a new type of databases where each instance consists of an entire time series. The main characteristics of this type of data are its high dimensionality, its dynamism, its auto-correlation and its noisy nature, all of which complicate the analysis to a large extent. In view of this, many researchers have focused on finding new methods or on adapting the existing data mining algorithms to obtain useful information from these databases. For example, tasks such as clustering have been successfully adapted to time series data in many application domains [1].

The clustering of time series databases typically requires the definition of a distance measure which will estimate the level of similarity or dissimilarity between time series. However, euclidean distance (ED) and other common measures used for non-temporal data are not always the most suitable methods to evaluate the similarity between time series, because they are unable to deal with noise and misalignments in the series [2]. For this reason, the scientific community has proposed a vast portfolio of measures for this specific type of data [3].

Experiments suggest that not all of these distance measures are appropriate for all time series databases [2]. This is probably due to the specific characteristics of each database, which make some distance measures more suitable than others. However, the choice of a similarity measure is not trivial because it is necessary to find a relationship between the characteristics of the time series databases and the properties of the different distance measures.

To the best of our knowledge, no formal methodology is available in the literature that supports the choice of one distance measure over another, in the context of clustering. As such, the common strategy is to experiment with a set of different distances and to choose one that is suitable one based on the obtained results. Due to the large number of available time series distance measures and the time complexity of many of them, this strategy is computationally very expensive and intractable in practice. Because of this, our main objective is to train a multi-label classifier that, given a specific unlabeled time series database, will automatically select the most suitable distance measures from a set of candidates. For this purpose, we propose to define a set of relevant features that describe each time series database and which will support the choice of one distance measure over another.

The rest of the paper is organized as follows. In Section 2, we present the set of features that will be used to describe the time series databases. Section 3 introduces our classification framework and Section 4 presents a method to define the class labels of the training set that will be used to build this classifier. We empirically validate our ideas in Section 5. Section 6 provides a summary of the most relevant results and finally, in Section 7, we extract the main conclusions and propose some insights on future work.

2 Time Series Database Characterization

In this section, we define a novel set of features that quantifies certain characteristics of time series databases and will be used as predictive information for choosing a suitable distance measure. A summary of these characteristics is

- *U. Mori and J.A. Lozano are with the Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, Gipuzkoa 20018, Spain. E-mail: {usue.mori, ja.lozano}@ehu.eus.*
- *A. Mendiburu is with the Department of Computer Architecture and Technology, University of the Basque Country UPV/EHU, Gipuzkoa 20018, Spain. E-mail: alexander.mendiburu@ehu.eus.*

Manuscript received 26 Mar. 2015; revised 21 July 2015; accepted 23 July 2015. Date of publication 29 July 2015; date of current version 3 Dec. 2015. Recommended for acceptance by E. Keogh.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2015.2462369

TABLE 1
Summary of All the Characteristics

Characteristic	Statistics	Quantification method/s	Number of variables
Dimension of database		Number of series in the database Mean length of series in database	2
Shift	Mean	Method based on cross correlations	1
Correlation	Mean, St Dev, %95Q, %5Q	Cross-correlations between series	4
Seasonality	Mean, St Dev, %95Q, %5Q % of series with high seasonality	STL decomposition	5
Trend	Mean, St Dev, %95Q, %5Q	STL decomposition	4
Trend	Mean	Number of peaks Mean distance in time between peaks Mean distance between values of peaks # of valleys Mean distance in time between valleys Mean distance between values of valleys # of jumps Mean size of jumps Moving average Exponential Smoothing	8
Noise	Mean, St Dev, %95Q, %5Q, Median	Fourier smoothing Non-linear Wavelet Shrinkage with D2 Non-linear Wavelet Shrinkage with C6	25
Outliers	Mean, St Dev, %95Q, %5Q, Median	Adjusted box-plots LOF adapted to time series	10
Skewness	Mean, St Dev, %95Q, %5Q	Skewness formula	4
Kurtosis	Mean, St Dev, %95Q, %5Q	Kurtosis formula	4
Autocorrelation	Mean, St Dev, %95Q, %5Q	Box-Pierce statistic	4
Total			**71**

%95Q and %5Q correspond to the %5 and %95 quantiles.

shown in Table 1. It should be noted that these characteristics do not describe a single time series but an entire database of time series. For this reason, for features describing a single series, we use a set of statistics to generalize the result to the entire database: we use the mean, the standard deviation and the 5 and 95 percent percentiles in most cases, together with some additional statistics for some particular features, which we comment on separately in each case.

The databases considered in this study only contain series of the same length so we will not study the cases with series of different lengths or sampling rates in detail. However, the calculation of the features can be extended directly to databases of this type in most cases. In the cases where specific techniques are necessary, we will make an explicit mention.

2.1 Dimensionality of the Database

It has been empirically demonstrated in the past few years that, in time series classification, the dimensionality of the time series has an impact on the accuracy of some similarity measures. For example, the accuracy of rigid distance measures such as euclidean distance improves as the number of time series in the database increases [2]. Besides, not all distance measures provide good results when working with very long time series [3]. These claims have not been assessed or analyzed in the clustering framework, but could also be relevant in some cases. As such, the number of time series in the database and the average length of these sequences are the first two variables that we will use to describe a time series database.

2.2 Shift

It is very common that the time series in a database are shifted in the time axis as shown in Fig. 1. Visually, anyone would confirm that the two time series in the figure are very similar to each other, but not all distance measures are able to deal with this type of similarity. For this reason, quantifying the level of "shift" in a time series database can be highly relevant.

To calculate the shift present in a database of time series $D = \{X_1, X_2, \ldots, X_N\}$, where X_i is a time series, we calculate the correlation of each pair of time series in the database at different lags and select the time lag in which this correlation yields its maximum. In this case, we define the correlation value of two series $X = \{x_0, x_1, \ldots, x_{n-1}\}$, $Y = \{y_0, y_1, \ldots, y_{n-1}\}$ at lag t using a slightly modified version of the common cross correlation formula, which avoids penalization at higher lags and considers both the positive and negative lags:

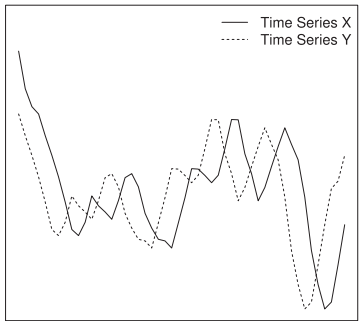


Fig. 1. Shifted time series.

$$r_t(X,Y) = \begin{cases} \frac{\frac{1}{n-t}\sum_{i=0}^{n-1-t}(x_i-\bar{x}_+)(y_{i+t}-\bar{y}_+)}{\sqrt{Var(x_+)}\sqrt{Var(y_+)}} & \text{if } t = 0,1,2,3,\ldots \\ \frac{\frac{1}{n-t}\sum_{i=0}^{n-1-t}(y_i-\bar{y}_-)(x_{i+t}-\bar{x}_-)}{\sqrt{Var(y_-)}\sqrt{Var(x_-)}} & \text{if } t = -1,-2,-3,\ldots \end{cases} \quad (1)$$

where $\bar{x}_+$ and $\bar{y}_+$ are the mean values of $X_+ = \{x_0, x_1, \ldots, x_{n-1-t}\}$ and $Y_+ = \{y_t, y_{t+1}, \ldots, y_{n-1}\}$ respectively, and $Var(x_+)$ and $Var(y_+)$ refer to the sample variances of these series. For negative lags, the same is applied for series $X_- = \{x_t, x_{t+1}, \ldots, x_{n-1}\}$ and $Y_- = \{y_0, y_1, \ldots, y_{n-1-t}\}$, respectively. It must be noted that Equation (1) is only directly applicable for series that are sampled in the same timestamps. Nevertheless, this calculation may be extended by applying more specific methods that obtain correlation values between unevenly sampled series [4].

Based on Equation (1), we can easily compute the maximum correlation value and its corresponding lag value:

$$r_{max}(X_i, X_j) = \max_t(r_t(X_i, X_j)), \quad (2)$$

$$lag(X_i, X_j) = \operatorname*{argmax}_t(r_t(X_i, X_j)). \quad (3)$$

At this point, it must be stressed that, while it is interesting to quantify the "shift" between a pair of similar time series, this operation does not make sense for series with completely different shapes. In this context, a simple method must be provided that attempts to discard the correlation and lag values obtained from dissimilar pairs of time series. The solution we propose is to simply consider the pairs of series with a high enough correlation value, and to provide the average of their absolute lag values as an approximation of the total shift level present in the database:

$$\text{Shift level} = \left(\frac{1}{|S_{\delta^*}|}\sum_{(X_i,X_j)\in S_{\delta^*}}|lag(X_i, X_j)|\right) \times \frac{100}{l}, \quad (4)$$

where l is the mean length of the time series of the database. As can be seen in the formula, we define the shift value of a database as a percentage of l, directly enabling the comparison of the values obtained for different datasets. Also, we define S_{δ^*} as follows:

$$S_{\delta^*} = \left\{(X_i, X_j) \mid i > j \wedge \frac{r_{max}(X_i, X_j) - r_*}{r^* - r_*} > \delta^*\right\}, \quad (5)$$

$$r^* = \max_{X_i,X_j\in D} r_{max}(X_i, X_j), \quad (6)$$

$$r_* = \min_{X_i,X_j\in D} r_{max}(X_i, X_j). \quad (7)$$

We note that in the calculation of S_{δ^*}, each pair of series in D is only considered once. The reason for this is that the value of r_{max} is equal for (X_i, X_j) and (X_j, X_i), and the value of lag only differs in the sign (see Equations (1), (2) and (3)). As can be seen in Equation (4), in the calculation of the shift feature, only the absolute phase difference between two series is of interest and, as such, it is not necessary to check the pairwise correlations between the series in both senses.

In order to define the threshold δ^* of Equation (5), we order the normalized correlation coefficients between the series in the database into an ascending sequence R. We recall that this study is set in a clustering framework and therefore the existence of several underlying groups in the databases is assumed. The series from the same cluster are supposed to be similar to each other, whereas those that belong to different clusters are dissimilar in some sense. In this context, if all the clusters are sufficiently different from each other, we will observe a clear jump that will separate the high correlations (intra-cluster) from the low correlations (inter-cluster), as shown in Fig. 2. Based on this, the first step is to calculate the first point after this jump occurs:

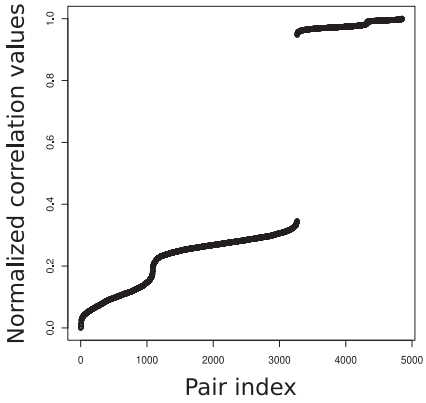


Fig. 2. Normalized correlations between pairs of series in a three-cluster database.

$$\delta = \max\{\hat{r}_i \in R \mid (\hat{r}_i - \hat{r}_{i-1}) > 0.01\}. \quad (8)$$

If the clusters are different enough from each other, δ^* will be directly defined by δ. However, the clusters in a time series database are not always very different from each other and in some cases the jump may be situated at very low correlation values or may not be present at all. Also, it is possible that the number of correlation values above this jump is too low to obtain significant information. As such, if δ is lower than 0.95, or if the pairs of series with a correlation value above δ are less than 5 percent of all the pairs of series, we set δ^* to a fixed value of 0.95:

$$\delta^* = \begin{cases} \delta & \text{if } \delta \geq 0.95 \text{ and } |S_\delta| \geq 0.05 \cdot \frac{N(N-1)}{2}, \\ 0.95 & \text{otherwise.} \end{cases} \quad (9)$$

The previous formulation includes some predefined thresholds and parameters such as the lower limit of δ^*, set to a fixed value of 0.95. It must be noted that we have set these parameters, and any others that appear in the formulations of the other characteristics defined in this section, based on the preliminary experiments explained later in Section 2.10.

It is easy to prove that the correlation, as formulated in Equation (1), between two identical but shifted linear series is always 1 and does not depend on the phase difference. Because of this, the procedure explained above is not useful to find the lag between two linear series. We consider a series linear if the Adjusted Root Mean Squared Error (ARMSE) of its fitted linear regression is lower than a user defined threshold, 0.1 in this case. If both of the series that are being compared (X and Y) are linear, the lag between them will be given by the median (med) of the difference

between their fitted linear regressions $\hat{X} = \{a \cdot t + b,\ t = 0, \ldots, n-1\}$ and $\hat{Y} = \{a' \cdot t + b',\ t = 0, \ldots, n-1\}$:

$$lag(X, Y) = med(\{(a - a')t + (b - b'),\ t = 0, 1, \ldots, n-1\}). \quad (10)$$

The correlation between the two original series controls whether they come from the same linear trend or not. As in the previous case, the correlation between a pair of series must be higher than δ^* in order to be considered, so we will only keep the pairs of linear series with a similar slope. In the special case of constant series, the correlation cannot be calculated. However, this is not problematic because constant series are invariant to shift, and do not give any useful information about the shift level of the database. As such, we will discard them before calculating the shift level of the database.

2.3 Correlation

Another feature that can be used to describe a time series database is the 95 percent percentile of the $r_{max}(X_i, X_j)$ values obtained in the calculation of the shift (see Section 2.2). This feature analyzes the level of correlation present in the database after removing the shift. It is a quite general feature that can be affected by various factors. For example, factors such as noise, outliers or local warp present in the database will be responsible for lower correlation values. Only some of the distance measures are able to deal with these issues and, therefore, this characteristic can be useful to discriminate between them.

In the same manner, the 5 percent percentile, the mean and standard deviation of the $r_{max}(X_i, X_j)$ values can be calculated. Contrary to the 95 percent percentile, these values provide information about the level of similarity between the clusters present in the database.

2.4 Seasonality

A time series has a seasonal component if it contains a pattern that repeats itself periodically. It is a basic characteristic of time series and it is interesting to study its influence in the performance of different distance measures. The methodology used to quantify this characteristic is based on the framework used in [5].

The first step is to find out whether the time series contain a seasonal component or not. If the collection frequency of the series is unknown, we use the spectral density function to find the frequency which has the greatest contribution to the variance of the series. If the contribution of this maximum value is higher than a user specified threshold (10, in this case), then the series has a seasonal component in that frequency. If no such frequency is found, we declare the seasonal component of the series null.

Finally, if the seasonal component is not null, we decompose the series using the Seasonal-Trend decomposition procedure based on local polynomial regression (STL) [6] and then quantify the seasonality in the following manner:

$$\text{Seasonality level} = 1 - \frac{Var(X_t - S_t - T_t)}{Var(X_t - T_t)}, \quad (11)$$

where X_t is the time series and S_t and T_t are its seasonal and trend components respectively.

To characterize an entire database, in addition to the basic statistics (mean, standard deviation and 5 and 95 percent percentiles) of all the seasonality levels, we save the percentage of series in the database that have a seasonality level higher than 0.5 as a feature. This last feature represents the number of series which have an important component of seasonality.

2.5 Trend

The trend of a time series is described as the long term behavior that does not include seasonal or random effects. The first way to characterize the trend level of a time series is by following the same methodology presented in the previous section:

$$\text{Trend level} = 1 - \frac{Var(X_t - S_t - T_t)}{Var(X_t - S_t)}. \quad (12)$$

This value will be calculated for each series and summarized using the four basic statistics (mean, standard deviation and the 5 and 95 percent percentiles).

Moreover, to describe the trend more in detail, we propose the following additional features: number of local maxima (peaks) and minima (valleys), mean distance between the peaks (and valleys) in the time domain normalized by the length of the time series, mean distance between the values of the peaks (and valleys) normalized by the maximum and minimum values of the series, number of jumps higher than $0.1 * (\text{max value of series—min value of series})$ and mean size of jumps normalized by the maximum and minimum values of the series. We identify the peaks and valleys in the time series by means of the *findPeaks* and *findValleys* functions in the *quantmod* package of R [7]. These functions are based on finding the timestamp in which the trend of the series changes from positive to negative, and outputting the next timestamp after the peak/valleys occurs. In order to obtain the exact peak value, we take the points immediately previous to those given by the *findPeaks* and *findValleys* functions.

To describe the entire database, we only consider the mean values in this case to avoid an excessive number of characteristics.

2.6 Noise

The amount of noise in the time series database may be relevant when choosing one distance over another, because some measures have a more robust behavior when noise is present [3]. To quantify the noise in a database, the idea is to remove it from each time series by applying several techniques reviewed in [8], and to calculate the standard deviation of this removed part. After calculating the noise level for each time series, we obtain an overall noise value by using the basic statistics (mean, standard deviation and the 5 and 95 percent percentiles). Additionally, we also save the median in order to reduce the effect of series with a strongly deviating noise level.

The first technique we have used to quantify the noise level of a time series database is a moving average filter. This method moves a fixed-sized window along the series and substitutes each point with the average of the values in the window. In order to try to find a balance between eliminating local fluctuations and over-smoothing the series, we

have chosen the window size to be 3 percent of the length of the series in the database. For short series, where %3 of the length is less than two points, we set the window size to a fixed value of two points.

The second technique is an exponential moving average filter. In this case, each point is replaced by the weighted average of all its previous points in the time series. The weights are defined by an exponential function that decreases as the data points get older. We have chosen a parameter of 0.7 after preliminary experiments.

The third method is called linear Fourier smoothing. The idea is to calculate the Discrete Fourier Transform of the time series and to eliminate the components that correspond to frequency values higher than a user predefined cut-off frequency. By using the inverse transformation, we obtain the smoothed time series and the noise can be quantified. In this case, we remove components referring to the highest 10 percent of the frequencies to try to smooth only the fast fluctuations in the time series.

The last two denoising techniques applied are variations of the nonlinear wavelet shrinkage, which is based on the Discrete Wavelet transform. Unlike the Fourier Transform, the Wavelet transform uses wavelet functions, in this case D2 Daubechies and C6 Coiflets, which are wave-like oscillations but of finite length. We have implemented this denoising method with the *wavshrink* function of the *wmtsa* package [9] in R by modifying only the parameter corresponding to the choice of the wavelet functions. With its default parameters, this R function calculates the decimated discrete wavelet transform of the series and then, the coefficients whose magnitude is below a threshold $delta$ are set to 0. This noise threshold is calculated by using the "universal" threshold function which is defined so that if the original time series only contained Gaussian noise, then the hard thresholding function would correctly eliminate all the noise contained in the series. Once the shrinkage is performed, by means of a synthesis operation, the new wavelet coefficients are transformed again into the time domain, obtaining the new de-noised series.

2.7 Outliers

Outliers, or data points that deviate notably from the rest of the data, greatly affect some distance measures [3] and should be included in the characterization of time series databases. For this reason, we calculate the proportion of outliers in each series with two different methods. Next, we generalize the results to the entire database as in the case of the noise.

The first method consists of building a box-plot for each time series and calculating the percentage of points that lie outside the whiskers of the plot. In order to remove the effect of trend and seasonality in the distribution of the data, the application of this technique must be performed after the series is detrended and deseasonalized using the STL method as explained in Section 2.4. Standard box-plots are designed to represent symmetric distributions and time series do not always fulfill this condition. Therefore, we use the adjustment proposed in [10] and implemented in the *robustbase* package [11] in R to build the plots. This approach makes use of the medcouple coefficient (MC) to measure the skew and modify the limits and whiskers of the plot to better capture the behavior of a non-symmetric distribution. Precisely, the MC coefficient takes values between -1 and 1, where the positive values correspond to right-skewed distributions and the negative values to those that are left-skewed. Symmetric distributions take a MC value of 0. In the *robustbase* package, the whiskers of the box-plot are defined based on an exponential function of the MC coefficient. Specifically, all points outside the intervals shown in Equation (13) will be catalogued as outliers

$$\begin{cases} \left[Q1 - 1.5e^{4MC} \cdot IQR\,,\, Q3 + 1.5e^{3MC} \cdot IQR\right] & if\ MC \geq 0, \\ \left[Q1 - 1.5e^{3MC} \cdot IQR\,,\, Q3 + 1.5e^{4MC} \cdot IQR\right] & if\ MC < 0, \end{cases} \quad (13)$$

where Q1 and Q3 are the first and third quartiles respectively, MC is the medcouple coefficient and IQR is the interquartile range. For symmetric distributions, Equation (13) is reduced to the formulation of a common box-plot.

The second method is an adaptation of the Local Outlier Factor (LOF) method presented in [12] to time series data. LOF is based on calculating the local density of each point by using the proximity to its neighbors and finding points that have a notably lower density than their neighbors. The method builds a ranking of the data points, and the outliers are identified as the points that have high LOF factors. This algorithm is defined for regular datasets and, therefore, a neighborhood is defined as a set of points that have similar values. However, in time series, the points are temporally correlated with each other and the definition of neighborhood must capture this behavior. As such, we have combined the LOF method with a sliding window of size 11. In each window, the LOF technique is applied by using the *DMwR* package [13] in R and we identify the points with a score higher than 5 as outliers for that window. In order to discard jumps or sudden changes, we finally consider a point an outlier if we have identified it as such in the majority of the windows that contain it (90 percent in this case, based on exploratory trials).

2.8 Skewness and Kurtosis

These two characteristics are related to the shape of the probability distribution of the data and can provide some insights about the shape of the time series. Skewness is a measure of the asymmetry of the data around the mean value and is calculated as follows:

$$\text{Skewness} = \frac{\sum_{i=0}^{n-1}(x_i - \bar{x})^3}{n\sigma^3}. \quad (14)$$

$\bar{x}$ is the mean of the time series and σ is the standard deviation. On the contrary, kurtosis is a measure of how peaked or flat the probability distribution of the data is:

$$\text{Kurtosis} = \frac{\sum_{i=0}^{n-1}(x_i - \bar{x})^4}{n\sigma^4} - 3. \quad (15)$$

These features only represent one time series and, so, we generalize them in the usual way (mean, standard deviation and 5 and 95 percent percentiles).

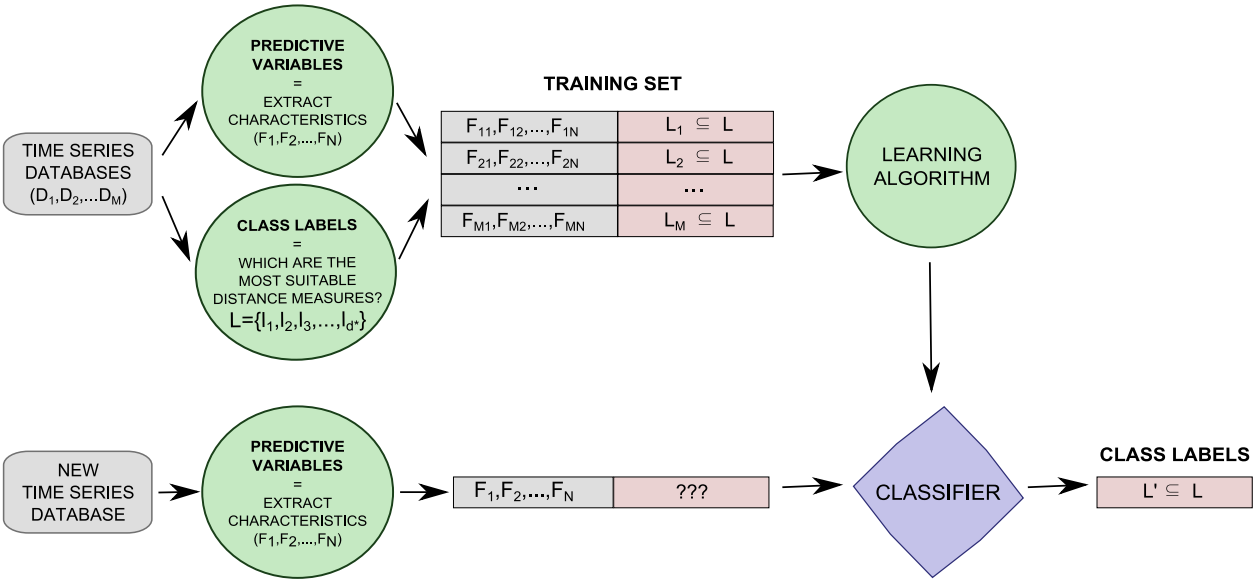


Fig. 3. Multi-label classification framework for the selection of the most suitable distance measure to cluster a time series dataset. If $d*$ candidate distance measures are considered and L is the label set that includes all these measures, a subset L_i of L is assigned to each time series database which represents the most suitable measures for the given dataset.

2.9 Autocorrelation

We obtain the level of autocorrelation of a single series X, from a modification of the Box-Pierce Statistic [5]:

$$AC(X) = \frac{1}{h}\sum_{i=1}^{h} r_i(X)^2, \tag{16}$$

where $r_i(X)$ is the autocorrelation value of the series at lag i, and h is chosen to be 15 percent of the length of the series. As the randomness in the series grows, the temporal correlation becomes weaker and the Box-Pierce statistic provides lower values. We generalize the autocorrelation values as usual (mean, standard deviation and 5 and 95 percent percentiles).

2.10 Parameter Tuning in the Definition of the Characteristics

As commented previously, most of the characteristics presented in this work to describe time series databases require the definition of some parameters or thresholds. We have chosen these parameters after a set of exploratory experiments carried out by using the Synthetic control [14] and the CBF [15] synthetic databases. Firstly, we have modified the characteristics of these two databases by tuning them synthetically and univariately and obtaining a set of databases with different and previously known features. The details on how to artificially tune the features of these time series databases can be studied in the supplementary material hosted in our website.[1] Secondly, we have measured the characteristics of these artificially generated databases by using the methods proposed in the previous sections, using some different parameter combinations. In a last step, we choose the parameters that yield the most accurate correspondence between the synthetically introduced "ground truth" characteristics and those that are measured. The combination of parameters that we have considered is not exhaustive and thus, the method does not ensure optimality in any way. In this context, more sophisticated parameter search strategies could be applied, which could improve the results.

3 Automatic Distance Selection for Time Series Databases

In this section we use the characteristics defined in Section 2 to create a multi-label classifier that is able to automatically choose the most suitable distances from a set of candidates. The proposed classification framework is shown in Fig. 3: the predictive variables are defined by the features described in Section 2 and the class labels declare which distance measures, from the set of options, are the most appropriate for clustering each database.

The reason for choosing a multi-label framework is that if we compare the behavior of different distance measures when clustering a time series database, it is common to find measures that yield equally good clustering results. Instead of providing only one solution, the multi-label framework enables the selection of the entire set of most suitable distance measures for each time series database. This increases the value of the distance selection tool because it allows the user to choose between equally good solutions based on the computational efficiency of the distance measures or other possible reasons.

After various preliminary tests, we have chosen two algorithms to train the proposed multi-label classifier, although any other choice could be used within the framework directly. The first is based on the *Classifier Chain algorithm (CC)* [16], which decomposes the multi-label problem into an ordered chain of binary classification problems [17]. Each of these binary classification problems corresponds to one of the class labels, and incorporates the class predictions from all the previous classifiers in the chain into its feature space. Since the single stand-alone CC classifier is strongly affected by the order of the labels, an ensemble approach is used in this

1. http://www.sc.ehu.es/ccwbayes/isg/index.php?option=com_content&view=article& id=122&Itemid=100#timeseries

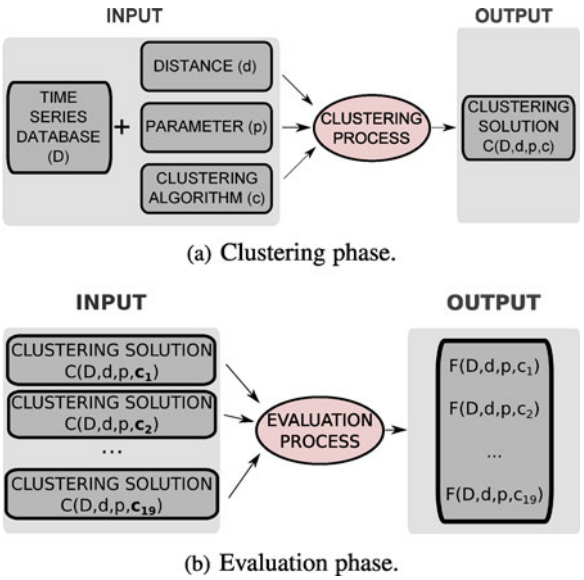


Fig. 4. Clustering and evaluation phases of the most suitable distance measure selection process.

work, as proposed in [16]: the *Ensemble Classifier Chain (ECC)* method. The second multi-label classifier used in this work is the *Random-k-labelsets classifier (RkL)* [18], which transforms the multi-label problem into a multi-class framework.

Both these methods are high-order problem transformation methods, because they transform the multi-label problem into a set of simpler more common classification tasks, which consider complex relations between all the labels. As such, they require the selection of an algorithm that solves the underlying classical classification problems. Pruned C4.5 trees [19] have been chosen because of their inherent feature selection ability, which adapts perfectly to this framework because it provides the means to automatically select a relevant set of features for each underlying classification problem.

In order to train these multi-label classifiers, we need the predictive variables and class labels of the training set of time series databases. We can obtain the predictive variables directly by simply calculating the features introduced in Section 2. However, the definition of the class labels is not so simple.

In a typical classification task, the class values of the set of training instances are usually provided by a set of experts or by the data generation process itself. In our case, recall that the class labels declare which distance measures are the most suitable to cluster each time series database. We do not have the labels of any publicly available time series datasets and, consequently, we will have to calculate them for a set of training instances by assessing and comparing the performance of the candidate set of distance measures when clustering them. This can be done if the ground truth clustering is known in advance, which is the case for the databases used in this study. However, to the best of our knowledge, there is no complete and statistically sound framework to perform this evaluation in the time series clustering context. In the following section, we propose a procedure to carry out this task.

4 Selection of the Most Suitable Distance Set

As mentioned in the previous section, in order to build the training set, we must define the class labels for each time series database by selecting the most suitable distance measures from the set of options. For this, we will replicate a common clustering procedure where different clustering solutions will be built, evaluated and compared. In the following sections, these three steps are explained in more detail:

4.1 Clustering Phase

In this first phase we cluster each training time series database in various ways by combining different options of distance measures, parameter settings and clustering algorithms (see Fig. 4a):

- *Distance measures*: We select a set of candidate distance measures from which the most suitable will be chosen. Note that, the larger the candidate distance set, the more complex the multi-label classification problem will become.
- *Parameters for the distance measures:* If the distance measures require the selection of a parameter, we perform the clustering separately for each parameter option.
- *Clustering algorithms:* We use several clustering algorithms to mitigate the influence that these may have on the performance of the distance measures. We choose the partitional K-medoids clustering algorithm with 15 different initializations and the agglomerative hierarchical algorithm with different options of linkage criteria: maximum or complete linkage, minimum or single linkage, mean linkage (UPGMA) or Ward's criterion.

Given the characteristics of the databases used in this study, the number of clusters is known beforehand and will be used directly.

4.2 Evaluation Phase

In this second phase, we evaluate all the clusterings performed in the previous stage in order to enable the comparison between them. As aforementioned, the ground truth clustering for all the databases that have been used in this study is known. This enables the calculation of the $\mathcal{F}$-measure for clustering, which is used as an evaluation metric in this process [20]:

$$\mathcal{F}(C^*, C) = \frac{1}{N}\sum_{i=1}^{k^*} N_i \max_{j=1,\ldots,k}\left\{\frac{2|C_i^* \cap C_j|}{|C_i^*| + |C_j|}\right\}, \qquad (17)$$

where N is the number of series in the database, N_i is the number of time series in the ith cluster of the ground truth solution, k^* is the number of clusters present in the ground truth clustering (C^*) and k is the number of clusters in the clustering that is to be evaluated (C). C_i^* and C_j represent the ith and jth cluster in the ground truth clustering and the clustering that is being evaluated respectively. In this study k^* and k are always equal.

We group the results as shown in Fig. 4b. For each database (D) and each combination of distance measure (d) and

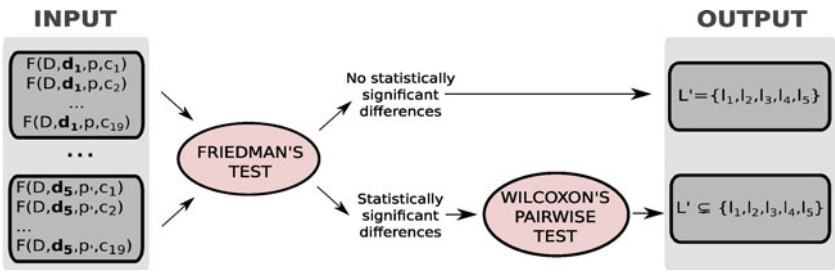


Fig. 5. Distance measure selection and label creation.

parameter (p), we obtain an array of $\mathcal{F}$-values that includes 15 values relative to the K-medoids algorithm ($\{c_1, \ldots, c_{15}\}$) and four values obtained from the variations of the hierarchical clustering algorithm ($\{c_{16}, \ldots, c_{19}\}$).

4.3 Comparison and Selection Phase

In this last phase, the aim is to choose the most suitable distances for the time series database. The different parameter options have a deep impact on the performance of the distance measures. In this context, before comparing the results of different measures, the most appropriate parameter has to be identified for each case. For each distance measure, we analyze all its $\mathcal{F}$-arrays (one for each parameter) together, comparing their 19 values row by row. The most suitable parameter is that which has the best $\mathcal{F}$-value for the highest number of rows. If there is a tie, we choose one of the winning parameter options randomly.

Once we have selected the best parameter for each distance measure, we discard all the other options and can focus on comparing the five different distance measures. In order to do this, we compare the $\mathcal{F}$-value arrays associated to each distance measure using various consecutive statistical tests (see Fig. 5).

First, we apply the Friedman rank sum test [21] to discover if there are any overall statistically significant differences between the performances of the distance measures. If the null hypothesis for this test is not rejected (p-value $> \alpha$), no significant differences can be assumed and we consider all the candidate distance measures equally suitable for the given time series database.

In contrast, if the null hypothesis is rejected (p-value $\leq \alpha$), we identify the first distance in the ranking built by the test, d, as one of the most suitable. In this case, the process continues with a comparison of every pair of distances using the Wilcoxon signed rank test [22]. In order to account for the multiple comparisons and control the family-wise error rate, we modify the p-values obtained from the Wilcoxon tests following the Holm post-hoc method. Finally, using these modified p-values, we determine the distance measures whose performance is statistically equal to d-s and add them to the set of most suitable distance measures. We set the significance level α to 0.05 in all the statistical tests applied in this phase.

5 Experimental Setup

In this section, the experiments carried out are summarized. The objective is to evaluate the accuracy of the automatic distance measure selection tool, built as explained in Section 3.

5.1 The Data

In this paper, we use two types of time series databases:

5.1.1 UCR Datasets

Specifically, the 46 databases from the UCR time series database archive [15]. This archive collects the majority of the publicly available synthetic and real time series databases and, since its creation, has provided the baseline for evaluating new classification, clustering and indexing proposals.

We have calculated the characteristics introduced in Section 2 for all these 46 databases to be used as predictive information. Furthermore, one of the advantages of the UCR archive is that the ground truth clustering of the databases is provided together with the data. This enables the calculation of the class variables, as explained in Section 4.

5.1.2 Synthetic Datasets

Recall that each time series database becomes an instance in our classification framework and the number of databases in the UCR archive is not very large. In view of this, we have complemented the set of databases with 555 synthetic datasets, generated specifically for this study. We will not provide the details for the generation of the databases in this paper, due to the lack of space. Nevertheless, the code and specifications for generating them and the databases themselves are publicly available in the link titled "Generator of Synthetic Databases" hosted in the Software section of our webpage [23].

The synthetic databases used in this study are divided into eight distinct groups, depending on the baseline functions used to generate the series therein. The baseline functions range from simple piecewise constant functions or linear series to more complex ARMA formulations, combinations of sinusoidal functions, random functions, etc. From each type of database, several different instances have been generated by synthetically modifying and tuning the dimension of the database, the level of noise, outliers, shift and local warp. The idea is to introduce databases with different characteristics that will cover the feature space and will enable the construction of a general classifier, applicable to new databases.

Once we have generated the 555 synthetic databases, in order to use them as training instances of the classification framework presented in Section 3, the characteristics described in Section 2 have been calculated for each of them. Additionally, since the ground truth clustering of these artificial databases is directly obtained from the database generation process (see supplementary material, available online, mentioned previously for details), we obtain their corresponding class labels as explained in Section 4.

z-normalization is a procedure which is typically applied before clustering or classifying time series data, with the aim of removing scale differences that could exist within the classes or clusters. In this case, we have not z-normalized the datasets given the small, or almost non-existent, scale difference between the series therein. However, we will see in the next section that all the distance measures included in the experimentation (indeed, most of those presented in the literature [3]) are not invariant to scale. In this context, if there were large scale differences between the series, it would be beneficial to perform z-normalization before building or applying the classifier.

5.2 The Distance Measures

To validate our proposal, from all the time series similarity measures available in the literature, we have chosen five. We have made this selection with the idea of including measures with different characteristics and considering their performance in previous publications. However, the distance selection method that is proposed in Section 3 could be applied to other similarity measures, including metric learning proposals. In the following paragraphs we provide a brief introduction on the selected distances, but more detailed information can be found in [2], [3], [24].

Euclidean distance is one of the most common measures applied in data mining and the first one included in our experimentation. Although ED has been widely used in many time series application fields, various researchers have pointed out that it is not a suitable measure for time series [2]. First, it is only able to deal with series of equal length. Moreover, ED is highly susceptible to noise and outliers, which are common in temporal sequences. Finally, it is based on the comparison between points collected at the same time interval which is a drawback because time series frequently suffer transformations in the time axis while still maintaining a similar shape [25]. However, ED is a very efficient and simple metric, which has provided competitive results in some situations [2]. As such, an automatic distance selection tool which determines the cases in which this measure is suitable could be very helpful to avoid having to turn to more complex and computationally costly distance measures.

In order to deal with the disadvantages of ED, a vast portfolio of distance measures has been specifically defined for time series, and *Dynamic Time Warping (DTW)* [26] is probably the most popular. The objective of this measure is to find the optimal alignment between two series by searching for the minimal path in a distance matrix that defines a mapping between them. In this case, we use the euclidean distance between every pair of points for the mapping. DTW is able to deal with transformations such as local warping and shifting and, furthermore, it allows the comparison between series of different length. The main weakness of DTW is its complexity ($O(nm)$, for a couple of time series of lengths n and m, respectively) which is considerably higher than that of other more simple distances. To conclude, it must be noted that it is quite common to add an extra temporal constraint to DTW by limiting the number of vertical or horizontal steps that the path can take consecutively. This adjustment avoids the matching of points that are very far from each other in time and, in addition, it reduces the computation cost [2]. In the experiments of the following sections, we consider both the temporally constrained and unconstrained versions of DTW.

TABLE 2
Parameter Options for the Distance Measures

Distance Measure	Parameter	Min. value	Max. value	Step size
DTW	Window size (w)	$0.04 \cdot l$	$0.32 \cdot l$	$0.04 \cdot l$
EDR	Threshold (ϵ)	$0.20 \cdot sd$	sd	$0.20 \cdot sd$
TQuest	Threshold (τ)	$avg - sd$	$avg + sd$	$0.40 \cdot sd$
F	Number of coefficients (f)	$0.04 \cdot l$	$0.32 \cdot l$	$0.04 \cdot l$

l is the length of the time series in the database, and avg and sd are the average and the standard deviation of the series in the dataset.

The next distance measure considered is the *Edit Distance for Real Sequences (EDR)* [27], which has shown very positive results in previous works [3]. This distance measure also defines a mapping between the pair of series that are being compared but, in this case, the distance between the points is reduced to 0 or 1 by using a user specified threshold, ϵ. If two points are closer to each other, in the absolute sense, than ϵ, they will be considered equal. If they are farther apart, they will be considered distinct from each other and the distance between them will be considered 1. Once the mapping is built, the minimal path is sought, as in DTW. With high values of ϵ, EDR becomes more flexible and more capable of dealing with noisy data or outliers. Finally, as an additional property, EDR permits gaps or unmatched regions in the series but it penalizes them with a value proportional to their length.

TQuest was presented in [28] and is classified as a feature based distance in [3]. In this manner, instead of comparing the raw values of the series, it studies the similarity of a set of features extracted from them. Each time series is represented as the set of intervals in which its values are above a user defined threshold τ. The distance between two series is calculated based on the similarity between their interval sets. In some previous experiments, TQuest has shown to give worse results than ED or DTW in most datasets [2]. In addition, in [3], the authors state that the computational cost of this distance for comparing two series of length n is $O(n^2 log(n))$, which is larger than the computational cost associated to DTW or EDR. However, as suggested in [2], TQuest could be useful for highly specialized domains or time series where a certain threshold τ is of special interest.

Finally, the *Fourier coefficient based similarity measure* [29] is also a feature based distance [3] and, as its name indicates, the similarity calculation is based on comparing the first f Discrete Fourier Transform coefficients of the series. It is interesting to include this distance measure, mainly due to its computational efficiency, which can be a very positive characteristic when working with large databases.

All the distance measures introduced above, except ED, require the selection of a parameter. In these experiments, we have considered the options summarized in Table 2 for each distance. We have defined the range of the parameters based on the experimentation shown in [2], but have augmented the step size in all cases because of the extra

TABLE 3
Summary of Validation Scenarios

Evaluation Scenario	Training time series databases	Testing time series databases	Validation Technique
Experiment 1	Synthetic	Synthetic	10 x 5-fold CV
Experiment 2	Synthetic + UCR	Synthetic + UCR	10 × 5-fold CV
Experiment 3	UCR	UCR	10 × 5-fold CV
Experiment 4	Synthetic	UCR	Train/Test
Experiment 5	UCR	Synthetic	Train/Test

computational effort when working with the large number of databases included in this study.

5.3 Training and Evaluation of the Multi-Label Classifier

We have selected five different evaluation scenarios, summarized in Table 3, to ensure the validity of our proposal. In the first experiment we only use the synthetic databases to evaluate the multi-label classifiers, within a 10 × 5-fold cross validation framework, in order to ensure that the training and testing of the multi-label classifier is done with separate portions of data. In the second experiment we use both the synthetic databases and the databases from the UCR archive together, also within a 10 × 5-fold cross validation scheme. The third experiment concentrates on the databases of the UCR archive and discards the synthetic databases. Finally, the fourth and fifth experiments apply a train/test validation methodology where one group of databases (synthetic or UCR) is used for training and the other for testing.

In all these evaluation scenarios, we have implemented the RkL and ECC classifiers proposed in Section 3 with MEKA [30], a multi-label extension to WEKA. In the case of RkL, we employ the parameter combination recommended in [16]: $k = 3$ and $m = 2L$, where L is the number of possible labels. Finally, for the ECC classifier, since the training sets are not very large, we set the only parameter, the number of iterations, to 50 as proposed in [16]. In both these classifiers, we use a unique threshold for all labels by leaving the default option $PCut1$ in MEKA.

With respect to the parameters of the pruned C4.5 trees, in Experiments 3 and 5, we directly adopt the default parameters provided by the J48 function in WEKA, which implements this type of model. In these cases, the pruning strategy applied by WEKA is subtree raising with the confidence level set on a fixed value of 0.25. In order to obtain more generalizable results, *reduced error pruning* can be applied, which holds out part of the training set to estimate this confidence factor. However, this implies that the initial tree structure is learnt with only part of the training set. In Experiments 3 and 5, the training set is already quite small so this option is not appropriate. However, in Experiments 1, 2 and 4, we have enabled this option. All the other parameters have been left in default mode.

Moreover, to provide a baseline for comparison, we have included three additional naive multi-label classifiers in the evaluation. The first is a *Random Classifier (RC)* which divides the problem into five independent binary classification problems, each one associated to a distance measure. Each of these binary classifiers will output a value of 0 (distance not present in the label set) or 1 (distance present in the label set) randomly but respecting the univariate probabilities of presence/absence observed in the training set for each distance. The performance values of this classifier can be exactly calculated if the class distribution of the testing set is assumed to be equal to that of the training set. However, this condition does not necessarily hold in all cases. Based on this, and given the randomness of RC, we have simulated this classifier 1,000 times and then averaged the performance values of all rounds in order to provide reliable results. The second one is denominated *zeroR*, following the notation in MEKA, and uses a constant classifier that simply selects the most common label as the base classifier of the RkL algorithm. The third classifier, called *Majority label set (MLS)*, simply assigns the most common label set in the training set to all the test instances.

We have evaluated the performance of all these classifiers using three metrics. The first is the Exact Match (EM) metric, which measures the proportion of correctly classified instances:

$$\text{Exact Match} = \frac{1}{N_{Te}} \sum_{i=1}^{N_{Te}} I(\hat{L}_i = L_i), \tag{18}$$

where N_{Te} is the total number of instances in the testing set, L_i is the true set of class labels for instance i and $\hat{L}_i$ is the predicted set of labels for this same instance. I takes a value of 1 if the condition is true and 0 otherwise. It must be noted that EM is the strictest among the evaluation metrics in the multi-label framework because it only considers the instances whose predicted label set is identical to the ground truth label set [17].

The second metric is the accuracy, which is not as strict as the EM metric and is calculated as [17]:

$$\text{Accuracy} = \frac{1}{N_{Te}} \sum_{i=1}^{N_{Te}} \frac{|L_i \bigcap \hat{L}_i|}{|L_i \bigcup \hat{L}_i|}. \tag{19}$$

The third performance measure we have chosen is the macro F1-measure calculated as [16]:

$$F1_{macro} = \sum_{i=1}^{L} F1(i), \tag{20}$$

where L is the maximum number of possible labels and F1 (i) is the F1-score calculated for label i as:

$$F1 = \frac{2 \cdot TP}{2 \cdot TP + FN + FP}, \tag{21}$$

where TP is the number of true positives and FP and FN are the number of false positives and false negatives, respectively.

As can be seen in Equations (18) and (19), the exact match and the accuracy are example-based metrics, where the performance of the classifier is calculated individually for all the instances in the database and then averaged to obtain a global value [17]. On the contrary, the F1-macro measure is a label-based metric, where the performance is evaluated for each label separately and then the average is calculated across all the class labels. By using these two types of

TABLE 4
Exploratory Analysis of Label Distribution in the Synthetic and UCR Databases

	UCR datasets	Synthetic datasets
Number of labels per instance		
Mean	1.370	1.986
Standard Deviation	0.771	1.228
Presence of each class label (%)		
DTW	50.00	68.30
EDR	45.70	69.20
TQUEST	19.60	7.40
ED	10.90	22.00
F	10.90	3.17
Concurrent presence of two class labels (%)		
ED & F	8.70	21.40
DTW & TQUEST	6.50	6.50
DTW & F	6.50	26.30
ED & DTW	4.30	19.80
ED & EDR	2.20	18.20
ED & TQUEST	2.20	3.60
EDR & TQUEST	2.20	6.70
EDR & F	2.20	24.90
TQUEST & F	2.20	4.30
DTW & EDR	1.30	41.40

measures we will analyze the classifiers from two different points of view.

As additional metrics, and in order to study the behavior of some classifiers more specifically, we calculate the multi-label precision and recall [17]:

$$\text{Precision} = \frac{1}{N_{Te}} \sum_{i=1}^{N_{Te}} \frac{|L_i \bigcap \hat{L}_i|}{|\hat{L}_i|}, \tag{22}$$

$$\text{Recall} = \frac{1}{N_{Te}} \sum_{i=1}^{N_{Te}} \frac{|L_i \bigcap \hat{L}_i|}{|L_i|}. \tag{23}$$

The comparison of these two metrics will provide some insights into the relation between the cardinality of the predicted class labels and the true class labels.

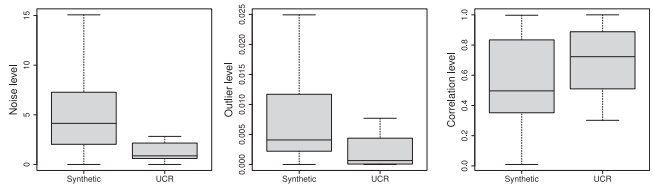
6 Results

In this section, we summarize the results obtained from the experimentation.

6.1 Exploratory Analysis of the Data

Before dealing with the results obtained from the experimentation, it is important to have some information about the characteristics of the databases that have been used in the study. The goal is to analyze the features and the labeling of the databases and to find the similarities and differences that could exist between the synthetic databases and the databases from the UCR. To begin with, in Table 4, some statistics and percentages are shown that provide information about the label distribution in the synthetic databases and the databases from the UCR archive.

It is obvious that the distribution of labels differs from one set of databases to the other. While the synthetic databases tend to have more than one label, the databases from the UCR archive generally admit only one distance as the most suitable. As evidence of this, the mean number of labels per instance is much closer to one in the case of the UCR distances than with the synthetic databases. Furthermore, the standard deviation is much lower in the UCR databases, and the concurrent appearances of pairs of distances are much less frequent in this case.



(a) Noise level. (b) Outlier level. (c) Correlation level.

Fig. 6. Comparison of the mean noise level, mean outlier level and mean correlation level in the synthetic and UCR databases. Noise level obtained by the moving average method and outlier level calculated using the adjusted box-plot method.

Taking into account that the distance selection process is based on statistical tests, this suggests that many of the databases from the UCR archive have very specific characteristics that only fit with the properties of one distance measure. For example:

- Many of the databases from the UCR archive have very similar clusters, some are even very difficult to discern by visual criteria. By setting very small values to the ϵ parameter of the EDR distance, this measure is able to magnify the small differences between the series, becoming an ideal distance measure for this type of databases.
- TQuest is among the most suitable distance measures much more frequently with the UCR databases than with the synthetic databases. As commented initially, this distance measure is not usually suitable for general databases and only provides good results in some specific cases where a certain threshold has a special relevance.

In relation to the feature space, after an exploratory analysis, it can be said that the synthetic databases cover a larger part than the databases from the UCR. As can be seen in Figs. 6a and 6b, the noise and outlier levels are restricted to very small values in the databases of the UCR. Also, as commented previously, there is a large presence of databases with very similar clusters in the UCR archive. This results in a much higher mean correlation value in the databases of the UCR in comparison with the synthetic databases (see, Fig. 6c).

Finally, variables related to trend, such as number of peaks, number of jumps, kurtosis and skewness, are generally higher in the synthetic databases. However, these variables are strongly influenced by the presence of randomness, noise and outliers and it is complicated to extract additional conclusions from exploratory univariate plots.

6.2 Evaluation of the Classifiers

We will now focus on the results obtained by the classifiers proposed in Section 3: in Table 5, the exact match, accuracy and F1-macro values are provided.

The first clear conclusion is that, in the first two validation scenarios, the classifiers that use the characteristics

TABLE 5
Exact Match, Accuracy and F1-Macro Values for the Multi-Label Classifiers

	ECC	RkL	RC	MLS	zeroR
EXACT MATCH					
Experiment 1	0.45	0.46	0.10	0.25	0.17
Experiment 2	0.42	0.44	0.10	0.24	0.16
Experiment 3	0.28	0.28	0.13	0.29	0.11
Experiment 4	0.07	0.13	0.09	0.30	0.09
Experiment 5	0.23	0.23	0.10	0.23	0.17
ACCURACY					
	ECC	**RkL**	**RC**	**MLS**	**zeroR**
Experiment 1	0.70	0.71	0.40	0.41	0.56
Experiment 2	0.68	0.69	0.39	0.41	0.55
Experiment 3	0.45	0.44	0.28	0.36	0.41
Experiment 4	0.39	0.40	0.32	0.37	0.44
Experiment 5	0.48	0.48	0.30	0.39	0.56
F1-macro					
	ECC	**RkL**	**RC**	**MLS**	**zeroR**
Experiment 1	0.71	0.64	0.40	0.16	0.33
Experiment 2	0.69	0.62	0.39	0.16	0.32
Experiment 3	0.30	0.26	0.27	0.12	0.23
Experiment 4	0.38	0.30	0.31	0.13	0.26
Experiment 5	0.33	0.30	0.31	0.16	0.33

TABLE 6
Precision and Recall Values for the Multi-Label Classifiers

	ECC	RkL	RC	MLS	zeroR
PRECISION					
Experiment 1	0.79	0.80	0.54	0.69	0.68
Experiment 2	0.77	0.78	0.53	0.67	0.67
Experiment 3	0.53	0.51	0.33	0.46	0.46
Experiment 4	0.46	0.47	0.36	0.46	0.48
Experiment 5	0.67	0.70	0.45	0.68	0.69
RECALL					
	ECC	**RkL**	**RC**	**MLS**	**zeroR**
Experiment 1	0.85	0.84	0.60	0.41	0.81
Experiment 2	0.84	0.82	0.59	0.41	0.81
Experiment 3	0.58	0.55	0.41	0.36	0.66
Experiment 4	0.74	0.67	0.56	0.37	0.76
Experiment 5	0.60	0.58	0.41	0.39	0.81

introduced in Section 2 clearly outperform the other three naive classifiers for all the three performance measures. Furthermore, the high performance values obtained confirm the usefulness of the two classifiers as automatic distance selection tools.

In the third validation experiment, the RkL and the ECC classifiers also obtain superior results for all performance measures except the exact match, which is slightly superior for the MLS classifier. The reason for this is that, the labels of the databases in the UCR archive are very unbalanced. The most common labelset in the UCR archive only contains one label, the EDR distance, and this choice appears in almost a third of the databases. As such, the MLS classifier obtains good results in exact match, but very low values for the F1-macro measure, because it only guesses the label of the EDR distance. This same behavior also appears in experiment 4, where the MLS classifier obtains very high exact match values but low values in other performance measures.

In the fourth experiment, the classifiers obtained by the ECC and RkL algorithms obtain low exact match values. A first possible explanation for this phenomenon is that there are large differences between the feature space of the databases from the UCR and the synthetic databases (see Section 6.1). In view of this, the results obtained in the fourth experiment are somewhat as expected.

Moreover, some additional conclusions can be obtained if the precision and recall values are studied (see Table 6). In the fourth experiment, the classifiers obtained from the RkL and ECC algorithms obtain much higher values of recall than precision, which suggests that these classifiers tend to assign more labels than necessary. This behavior is expected because, as observed in Section 6.1, the databases from the UCR generally admit fewer class labels than the synthetic databases. In this context, although the accuracy and precision values for the ECC and RkL classifiers do not differ too much from those obtained from the MLS classifier, the exact match values become worse on account of the excess of labels. Nevertheless, given the high results that the ECC and RkL classifiers obtain for the F1-macro and accuracy measures in this experiment, we can say that, although they are more conservative, these classifiers are still useful to reduce the size of the candidate distance set.

Finally, in the fifth experiment, the ECC and RkL methods obtain exact match values which are similar (and slightly superior in the case of ECC) to the MLS classifier, and very superior to those obtained by the Random and ZeroR classifiers. With respect to the other metrics, only the zeroR classifier obtains slightly higher results than the ECC and RkL classifiers.

The reason for this is that, in all the experiments, the zeroR classifier always tends to assign the two most common distances, DTW and EDR, to all the test instances. Given the unbalanced nature of the label sets (see Table 6.1), especially those of the UCR databases, the F1-macro score and the accuracy are quite high for this classifier in all the experiments. However, the exact match value is very low in all cases. If we study Table 6, we can see that the recall values are generally very high for the zeroR classifier. In contrast, the precision values are very similar to or even lower than those obtained by other classifiers such as RkL or ECC. The big difference between these two performance scores implies that this classifier always tends to assign more labels than necessary. Moreover, since the two distance measures that the zeroR classifier systematically selects are the most expensive in terms of computational cost, we can say that this classifier is not very useful as a distance selection tool.

In summary, from the results shown, it may be concluded that the characteristics proposed to describe the time series databases are useful to discriminate between distance measures. This is deduced from the fact that the proposed classifiers obtain overall good performances for all evaluation scores in comparison to the naive classifiers, which obtain better results only on some occasions and for some specific evaluation scores. Furthermore, in view of the results obtained we can say that the proposed framework is useful to automatically select the most suitable distance measure for new time series databases.

As a real case study, and in order to illustrate the applicability of our method in a real context, we have performed

some preliminary experiments with the PAMAP dataset [31], [32], obtained from monitoring the physical activity of various subjects. These experiments have not been included in the paper due to the lack of space, and because the special typology of the data and the specific pre-process necessary to cluster this dataset require a more extensive explanation. However, we have included them in a file of supplementary material, available online, attached to this paper.

6.3 Visual Assessment of the Results

In this section we focus on the databases from the UCR and visually represent some of the results obtained by our method. To do this, first, we obtain a label prediction for each dataset in the UCR archive using a ECC multi-label classifier trained using all the synthetic sets and all the UCR datasets except the one of interest. Then, we plot the obtained results.

The Texas sharpshooter plot [25] was designed to evaluate distance measures in the context of time series classification. The aim of this plot is to assess when a new distance measure is better than a given baseline (ED, generally), and, additionally, if the databases in which it performs better can be identified in advance. For this, the expected gain in accuracy obtained by the proposed distance is calculated by using only a training set and then compared to the true gain, obtained by using a testing set.

In our case, we do not propose a unique distance measure, but a distance measure selecting tool. Furthermore, we focus on clustering, where there are no training and testing sets. In this context, the Texas sharpshooter plot is not directly applicable. However, we retain the underlying idea, take the ED as the baseline distance and calculate these two values for each database:

$$F_{obtained} = \frac{\text{Best F-measure obtained by the distance measures selected by the multi-label classifier}}{\text{F-measure obtained by using the euclidean distance}}, \tag{24}$$

$$F_{true} = \frac{\text{Best F-measure obtained by the true most suitable distance measures}}{\text{F-measure obtained by using the euclidean distance}}. \tag{25}$$

Note that the true most suitable distance measure set is obtained as explained in Section 4. Once we have obtained these values, we plot them and obtain the following conclusions:

- *Points close to the diagonal:* The points that lie close to the diagonal represent databases for which our method obtains a F-measure similar to the F-value obtained by the most suitable distance measure(s). Specifically, for the databases that are situated exactly on the diagonal, our method has provided at least one of the most suitable distances among its choices. Note that there are a few databases that obtain a higher F.Obtained value than the F.True value (points below the diagonal). The reason for this is that the most suitable distance sets are selected by means of statistical tests (see Section 4) whereas in Fig. 7 we focus only on the individual highest F-values. There are only three cases that deviate notably from the diagonal: the "FaceFour", "TwoPatterns" and "fish" databases. In the first case, our classifier chooses the Fourier distance instead of the EDR distance, which is the most suitable for this dataset. The database is quite noisy and both these distance measures are able to deal with noise, to some extent. Furthermore, the shift in the "FaceFour" dataset is very small, which is possibly why the multi-label classifier chooses the F distance over the EDR distance. In the case of "Two patterns", the classifier chooses the EDR distance when the DTW distance is the most suitable. In this database, some sections of the series are noisy (generated by a Gaussian distribution), which is probably why the EDR distance was chosen. However, these noisy sections are alternated with piecewise constant functions, which are essentially the parts which provide relevant information about the different classes. This behavior can not be detected by the common denoising methods and so, it is difficult for the multi-label classifier to predict the correct class label. Finally, in the "fish" database, the DTW distance is chosen by the classifier, when the EDR is the most suitable. In this case, it is difficult to extract any intuitive reasons for the choice made by the classifier as it is probably due to a complex combination of features.
- *Points to the right of the vertical line:* The datasets that lie to the right of the vertical line are those in which our method has provided useful information, resulting in an improvement over the ED distance. It is obvious that most of the UCR databases shown in the plot lie in this area.

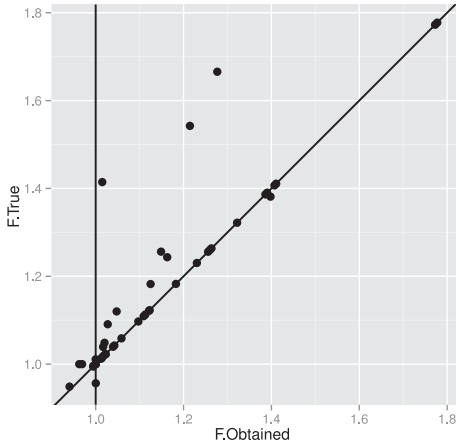


Fig. 7. Variation of the Texas sharpshooter plot for the databases in the UCR archive.

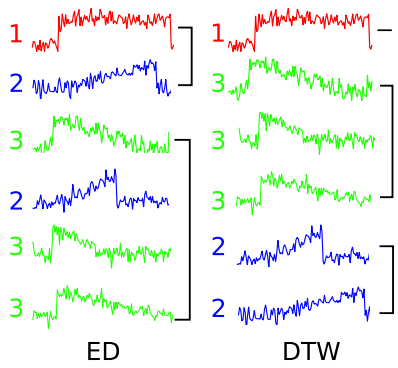


Fig. 8. The cluster solution obtained for a subset of the "CBF" dataset by using ED and the distance selected by our multi-label classifier. The true cluster of each series is represented by the number on its left and the clustering obtained by the distance measure is represented by the lines on the right.

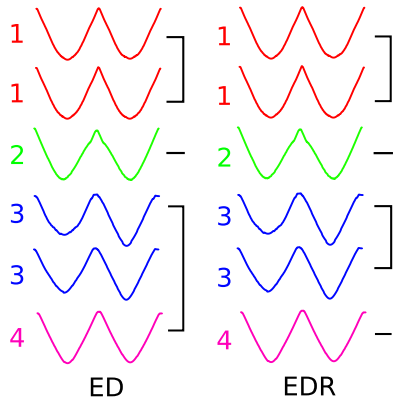


Fig. 9. The cluster solution obtained for a subset of the "DiatomSizeReduction" dataset by using ED and the distance selected by our multi-label classifier.

- *Points to the left of the vertical line:* These are the cases in which the ED performs better than any of the distance measures selected by our multi-label classifier. For the five databases that lie in this area, the accuracy obtained by the selected distance is not much lower than that obtained by ED. However, the ED is computationally very cheap, so using a more complex measure to cluster our database when ED provides good results, is a waste of computational effort. For two of the databases in this area, our classifier chooses the Fourier distance, which is not computationally very expensive. However, in the other three databases, DTW or EDR are chosen. In these cases, clustering the databases with the distances chosen by our classifier will yield similar accuracy results, but will result in a significant increase in computational cost compared to applying ED.

To finish with our visual analysis, we show two examples (Figs. 8 and 9) of the improvement that can be obtained by using the distances proposed by our classifier instead of the baseline ED. In the "CBF" database, the shifting and warping between the series in the same class is quite large, and in this context, DTW gives much better results than ED. In the "DiatomSizeReduction" database, the similarity between the clusters makes EDR the most suitable distance measure because, by choosing a very small epsilon, it is able to magnify the tiny differences between the clusters, that ED is not able to capture correctly.

7 Conclusions and Future Work

In this paper, a multi-label classifier for the automatic similarity measure selection has been proposed for the task of clustering time series databases. The classifier receives a set of characteristics that describe the database as input and returns the set of most suitable distance measures from a set of candidates. The positive results obtained in the experimentation for various multi-label classification performance measures demonstrate that this tool is useful to simplify the distance measure selection process, crucial to the time series database clustering task.

An important by-product of this work is the introduction of the labeling process introduced in Section 4. With the definition of this process, we have proposed a distance measure evaluation method based on statistical tests for the task of clustering. We believe that, a method of this type has not been proposed before.

The first obvious future research direction is to include new distance measures in the proposed framework. In this line, a more extensive study could be performed introducing new features, that would describe other aspects of the time series databases that have not been considered in this paper. For this purpose, some of the features presented in [33] could be considered.

Another proposal for future work includes an optimization of the temporal costs associated with the calculation of the characteristics. Some of the features introduced in this study, such as the shift, are computationally quite expensive to calculate, which could be an inconvenience when working with particularly large databases. Since only means, medians, standard deviations and other general statistics are calculated, strategies such as sampling the time series database could be applied to reduce this computational cost. In the same line, reducing the number of parameters associated to the characteristics could also improve the applicability of the proposal.

Finally, some insights into the definition of the parameters of the distance measures have been included throughout the paper, but no extended experimentation has been carried out on this topic. Studying the relationship between the characteristics of the databases and the parameters that define each distance could be useful to simplify the selection of a distance measure even more.

Acknowledgments

The authors would like to thank the UCR archive and Prof. Keogh for providing the data used in the study. This work has been partially supported by the IT-609-13 program (Basque Government), TIN2013-41272P (Spanish Ministry of Science and Innovation) and by the NICaiA Project PIRSES-GA-2009-247619 (European Commission). Usue Mori holds a grant from the University of the Basque Country UPV/EHU.

References

[1] T. W. Liao, "Clustering of time series data: A survey," *Pattern Recog.*, vol. 38, no. 11, pp. 1857–1874, Nov. 2005.

[2] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining Knowl. Discovery*, vol. 26, no. 2, pp. 275–309, Feb. 2012.
[3] P. Esling and C. Agon, "Time-series data mining," *ACM Comput. Surveys*, vol. 45, no. 1, pp. 1–34, Nov. 2012.
[4] K. Rehfeld, N. Marwan, J. Heitzig, and J. Kurths, "Comparison of correlation analysis techniques for irregularly sampled time series," *Nonlinear Processes Geophysics*, vol. 18, no. 3, pp. 389–404, Jun. 2011.
[5] X. Wang, K. Smith, and R. Hyndman, "Characteristic-based clustering for time series data," *Data Mining Knowl. Discovery*, vol. 13, no. 3, pp. 335–364, May 2006.
[6] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "STL: A Seasonal-trend decomposition procedure based on loess," *J. Official Statist.*, vol. 6, no. 1, pp. 3–73, 1990.
[7] J. A. Ryan. (2013). Quantmod: Quantitative financial modelling framework. r package version 0.4-0 [Online]. Available: http://CRAN.R-project.org/package=quantmod
[8] T. Köhler and D. Lorenz. (2005). A comparison of denoising methods for one dimensional time series. Tech. Rep. [Online]. Available: http://www.math.uni-bremen.de/zetem/DFG-Schwerpunkt/preprints/orig/lore nz20051dreport.pdf
[9] W. Constantine and D. Percival. (2012). WMTSA: Wavelet methods for time series analysis [Online]. Available: http://cran.r-project.org/package=wmtsa
[10] M. Hubert and E. Vandervieren, "An adjusted boxplot for skewed distributions," *Comput. Statist. Data Anal.*, vol. 52, no. 12, pp. 5186–5201, Aug. 2008.
[11] P. Rousseeuw, C. Croux, V. Todorov, A. Ruckstuhl, M. Salibian-Barrera, T. Verbeke, M. Koller, and M. Maechler. (2013). Robustbase: Basic robust statistics [Online]. Available: http://cran.r-project.org/package=robustbase
[12] M. M. Breunig, H.-p. Kriegel, and R. T. Ng, "LOF: Identifying density-based local outliers," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2000, pp. 93–104.
[13] L. Torgo. (2010). *Data Mining With R, Learning with Case Studies.* London, U.K.: Chapman & Hall [Online]. Available: http://www.dcc.fc.up.pt/ ltorgo/DataMiningWithR
[14] D. T. Pham and A. B. Chan, "Control chart pattern recognition using a new type of self-organizing neural network," *Proc. Institution Mech. Eng. Part I-J. Syst. Control Eng.*, vol. 212, no. 2, pp. 115–127, 1998.
[15] E. Keogh, Q. Zhu, B. Hu, H. Y., X. Xi, and C. A. Wei, L. Ratanamahatana. (2011). The UCR time series classification/clustering homepage [Online]. Available: www.cs.ucr.edu/eamonn/time_series_data/
[16] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Mach. Learning*, vol. 85, pp. 333–359, 2011.
[17] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1819–1837, Aug. 2014.
[18] G. Tsoumakas and I. Vlahavas, "Random k-Labelsets: An ensemble method for multilabel classification," in *Proc. 18th Eur. Conf. Mach. Learning*, 2007, pp. 406–417.
[19] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1993.
[20] S. Wagner and D. Wagner. (2007). Comparing Clusterings - An overview. Universität Karlsruhe (TH), Karlsruhe, Germany, Tech. Rep. 2006-04 [Online]. Available: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000011477
[21] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Ann. Math. Statist.*, vol. 11, no. 1, pp. 86–92, 1940.
[22] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, no. 6, pp. 80–83, 1945.
[23] University of the Basque Country UPV/EHU. (2014). Intelligent systems group [Online]. Available: http://www.sc.ehu.es/ccwbayes/isg/
[24] J. Serrà and J. L. Arcos, "An empirical evaluation of similarity measures for time series classification," *Knowl.-Based Syst.*, vol. 67, pp. 305–314, Sep. 2014.
[25] G. E. Batista, X. Wang, and E. J. Keogh, "A complexity-invariant distance measure for time series," in *Proc. SDM*, 2011, pp. 699–710.
[26] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. KDD Workshop*, 1994, pp. 359–370.
[27] L. Chen, M. T. Ozsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2005, pp. 491–502.
[28] J. Aßfalg, H.-p. Kriegel, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz, "Similarity search on time series based on threshold queries," in *Proc. 10th Int. Conf. Adv. Database Technol.*, 2006, pp. 276–294.
[29] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Proc. 4th Int. Conf. Found. Data Orga. Algorithms*, 1993, pp. 69–84.
[30] J. Read. (2012). MEKA: A multi-label extension to WEKA [Online]. Available: http://meka.sourceforge.net/
[31] A. Reiss and D. Stricker, "Towards global aerobic activity monitoring," in *Proc. 4th Int. Conf. PErvasive Technol. Related Assistive Environ.*, Jul. 2011, pp. 1.
[32] A. Reiss, M. Weber, and D. Stricker, "Exploring and extending the boundaries of physical activity recognition," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 2011, pp. 46–50.
[33] B. D. Fulcher and N. S. Jones, "Highly comparative feature-based time-series classification," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 1–20, Dec. 1, 2014.

Usue Mori received the BSc degree in mathematics, the MSc degree in mathematical modeling, statistics and computing, and the MSc degree in computational engineering and intelligent systems from the University of the Basque Country UPV/EHU, Spain, in 2010, 2011, and 2012, respectively. Since 2013, she has been working toward the PhD degree in the University of the Basque Country UPV/EHU. Her main research interests include clustering and classification of time series.

Alexander Mendiburu received the BSc degree in computer science and the PhD degree from the University of the Basque Country, Spain, in 1995 and 2006, respectively. Since 1999, he has been a lecturer in the Department of Computer Architecture and Technology, University of the Basque Country UPV/EHU. His main research areas are evolutionary computation, probabilistic graphical models, and parallel computing.

Jose A. Lozano received the MSc degree in mathematics, the MSc degree in computer science, and the PhD degree in computer science from the University of the Basque Country UPV/EHU, Spain, in 1991, 1992, and 1998, respectively. Since 2008, he has been a full professor at the University of the Basque Country UPV/EHU, where he leads the *Intelligent Systems Group*. He is the coauthor of more than 50 ISI journal publications and a coeditor of three books. His major research interests include machine learning, pattern analysis, evolutionary computation, data mining, metaheuristic algorithms, and real-world applications. He is an associate editor of the *IEEE Transactions on Evolutionary Computation* and a member of the editorial board of another five journals. He is a member of the IEEE.