Feature Extraction over Multiple Representations for Time Series Classification

Dominique Gay(✉), Romain Guigourès, Marc Boullé(✉), and Fabrice Clérot

Orange Labs, 2, Avenue Pierre Marzin, 22307 Lannion Cedex, France
{dominique.gay,Romain.Guigoures,marc.boulle,Fabrice.Clerot}@orange.com

Abstract. We suggest a simple yet effective and parameter-free feature construction process for time series classification. Our process is decomposed in three steps: (i) we transform original data into several simple representations; (ii) on each representation, we apply a coclustering method; (iii) we use coclustering results to build new features for time series. It results in a new transactional (i.e. object-attribute oriented) data set, made of time series identifiers described by features related to the various generated representations. We show that a Selective Naive Bayes classifier on this new data set is highly competitive when compared with state-of-the-art times series classification methods while highlighting interpretable and class relevant patterns.

1 Introduction

Time series classification (TSC) has been intensively studied in the past years. The goal is to predict the class of an object (a time series or a curve) $\tau_i = \langle(t_1, x_1), (t_2, x_2), \ldots, (t_{m_i}, x_{m_i})\rangle$ (where $x_k, (k = 1..m_i)$ is the value of the series at time t_k), given a set of labeled training time series. TSC problems differ from traditional classification problems since there is a time dependence between the variables; in other terms, the order of the variables is crucial in learning an accurate predictive model. The increasing interest in TSC is certainly due to the wide variety of applications: from e.g., medical diagnosis (like classification of patient electrocardiograms) to the maintenance of industrial machinery. Other domains, where data might be time series, are also concerned: finance, meteorology, signal processing, computer network traffic, ... The diversity of applications has given rise to numerous approaches (see Sect. 2 for detailed related work). However, most efforts of the community have been devoted to the following three-step learning process: (i) choosing a new data representation, (ii) choosing a similarity measure (or a distance) to compare two time series and (iii) using the Nearest Neighbor (NN) algorithm as classifier on the chosen representation, using the chosen measure. Ding et al. [10] offer a survey of the various data representations and distances found in the literature and an extensive experimental study using the NN classifier. They conclude that NN classifier coupled with Euclidean distance (ED) or Dynamic Time Warping (DTW) show the highest predictive performance for TSC problems using the original time domain. Recently, Bagnall et al. [1] experimentally show that the performance of classifiers significantly increases when changing data representation (compared with original temporal domain); thus, for a given classifier, there is a high variance of performance depending on the data transformation at use. To alleviate this problem, an ensemble method TSC-ENSEMBLE [1] based on three data representations (plus the original data) and NN algorithm is suggested. The experimental results demonstrate the importance of representations in TSC problems and show that a simple ensemble method based on several data representations provides highly competitive predictive performance. However, with the good performance of NN-based approaches also come the drawbacks of lazy learners: i.e., there is no proper training phase, therefore the training set has to be entirely stored and all the computation time is postponed until deployment phase; these weaknesses do not meet the requirements of deployment in resource-limited and/or real-time applications. Another weakness of the NN approaches is the lack of interpretability; indeed NN only indicates the nearest series w.r.t. the used similarity measure.

The method we suggest takes the pros and leaves the cons of the methods listed above: we come back to the *eager*[1] paradigm, benefit from the combination of multiple representations, build and select valuable features from multiple representations. More precisely, in this paper, we suggest a parameter-free process for constructing valuable features over multiple representations for TSC problems. Our contribution is thus essentially methodological. The next section discuss further related work to give a wider view of existing solutions for TSC problems. Section 3 motivates and describes the three steps of our unsupervised process: (i) transformation of original data into several new data representations; (ii) coclustering on various data representations; (iii) the exploitation of coclustering results for the construction of new features for the data. The output of the process is then a traditional data set (i.e. labeled objects described by attributes) ready for supervised feature selection and classification. We report the experimental validation of our approach in Sect. 4 before concluding.

2 Related Work

In TSC problems, DTW-NN is recognized by the community as a hard-to-beat baseline and it is confirmed by our experiments (see Sect. 4). However, there exist alternative approaches: besides the numerous similarity measures coupled with Nearest Neighbor algorithm [10], very recent novel metric has been proposed [25] as well as fusion of distance measures [7] and NN ensembles over multiple representations [1]. Concerning the intra-class variance, to deal with the lack of objects that cover sub-class pattern, Grabocka et al. [13] suggest to create virtual transformed objects for the training set; it results in a significant improvement of SVM predictive performance.

[1] In contrast to lazy learning, eager learning has an explicit training phase and generally deploys faster.

For the sake of interpretability, an extension of decision tree has been proposed recently [15]. On the other hand, feature-based approaches have also been intensively studied. Feature-based approaches for TSC aim at extracting class-relevant characteristics of series so that a conventional classifier can used. A wide variety of features has been studied: e.g., global, trends [16], symbolic, intervals, distance-based [9], features coming from spectral transforms [20,32] or a combination of several types of features [11].

Shapelet-based approaches [12,30], a subtopic of feature-based approaches, have drawn much attention in recent years. Shapelets are time series subsequences that are representative of a class. First approaches have embedded extracted shapelets in a decision tree [12,30,31], others in a simple rule-based classifier [29], while very recently, Lines et al. [19] have designed a shapelet-based transform.

Our approach generates similarity-based features and histogram features over multiple representations (see next section); the former allows us to reach predictive performance comparable to the best similarity-based NN classifiers, with the latter we gain some insight in the data. The closest works are from the inspiring works of Bagnall et al. [1] who establish competitive predictive performance by combining multiple representations in an ensemble classifier.

3 Feature Construction Process

Notations. In TSC problems, we define a time serie as a pair (τ_i, y_i) where τ_i is a set of ordered observations $\tau_i = \langle (t_1, x_1), (t_2, x_2), \ldots, (t_{m_i}, x_{m_i}) \rangle$ of length m_i and y_i a class value. A time series data set is defined as a set of pairs $D = \{(\tau_1, y_1), \ldots, (\tau_n, y_n)\}$, where each time series may have a different number of observations (i.e. with different length). Notice that the time series of a data set may also have different values for $t_k, (k = 1..m_i)$. The goal is to learn a classifier from D to predict the class of new incoming time series $\tau_{n+1}, \tau_{n+2}, \ldots$ To achieve this goal, we suggest the feature construction process summarized as follows:

1. We transform original data into multiple data representations.
2. We process a coclustering technique on each representation.
3. We build a set of features from each coclustering result and obtain a new data set gathering the various sets of features.

The new data set is thus object-attribute oriented and ready for supervised classification phase. Since our main contribution is methodological, we will take some time to motivate each step of the process, and when necessary, to make the paper self-contained, we will recall the main principles of the tools used in each step.

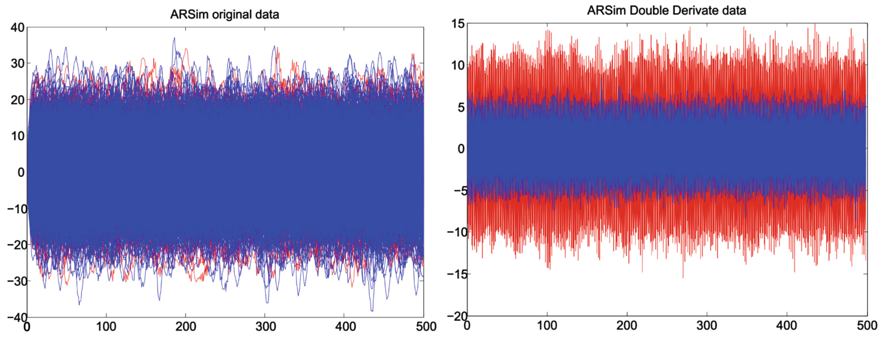
3.1 Transformations and Representations

Numerous data transformation methods for time series has been suggested in the literature: e.g., polynomial, symbolic, spectral or wavelet transformations, (see [10,18] for well-structured surveys on experienced data representations). The underlying idea of using data transformation is that transformed data might contain class-characteristic pattern that are easily detectable (i.e. patterns unreachable in the original time domain). The following example illustrates and confirms the relevance of using representations, and highlights simple interpretable features that might arise from data representations.

Motivating example. Graphs from Fig. 1 confirm the relevance of changing data representation: indeed, from original data (a) it is challenging to separate the two classes (blue/red). It has been shown [1] that the accuracy of NN-based classifiers on the original data is about 60 %. On the other hand a simple transformation, like double derivate (b) facilitate class discrimination. For example, after computing the double derivate transformation, we see that curves with some values above 6 (or below -6) are red whereas curves with most of its values between -6 and 6 are blue. On this data example, a simple transformation and two interpretable features are enough to characterize the two classes of curves.
To illustrate and instantiate our process, we use the original representation and we pick six representations among the numerous ones existing in the literature.

Derivatives: DV et DDV. We use derivatives and double derivatives of original time series (computed between time t et $t-1$). These transformations allow us to represent the local evolution (i.e., increasing/decreasing, acceleration/deceleration) of the series.

Cumulative integrals: IV et IIV. We also use simple and double cumulative integrals of the series, computed using the trapeze method. These transformations allow us to represent the global (cumulated) evolution of the series.



(a) (b)

Fig. 1. ARSim 2-class data: original data versus double derivate transformation (Color figure online).

Power Spectrum: PS. A time series can be decomposed in a linear combination of sines and cosines with various amplitudes and frequencies. This decomposition is known as the Fourier transform. And, the Power Spectrum is $PS(\tau_i) = \langle (f_1, a_1), \ldots, (f_{m_i}, a_{m_i}) \rangle$, where f_k represent the frequency domain and a_k the power of the signal (i.e. the sum of the Fourier coefficients squared). This transformation is commonly used in signal processing and plunges the original series into the frequency domain.

Auto-correlation function: ACF. The transformation by auto-correlation (ACF) is: $\tau_{i\rho} = \langle (t_1, \rho_1), \ldots (t_{m_i}, \rho_{m_i}) \rangle$ where

$$\rho_k = \frac{\sum_{j=1}^{j=m_i-k} (x_j - \bar{x}) \cdot (x_{j+k} - \bar{x})}{m \cdot s^2}$$

and where $\bar{x}$ and s^2 are the mean and variance of the original series. ACF transformation describes the correlation between values of the signal at different times and thus allow us to represent auto-correlation structures like repeating patterns in the time series.

We do not pretend that the chosen representations are suitable for all TSC problems; there are many other transformation techniques in the literature. Depending on the application, the domain expert remains the best to select potentially suitable representations for the problem at hand. Let us just recall that the time complexity for computing the chosen representations is at most sub-quadratic w.r.t. the number of points.

Thus, for a given time series data set D_{orig}, we build six new data representations: D_{DV}, D_{DDV}, D_{IV}, D_{IIV}, D_{PS} and D_{ACF} depending on the transformation used. In the following, for the sake of generality, an object from one of these representations will be called "curve" instead of time series since D_{PS} does not use the time domain.

3.2 Coclustering

In classification problems (also in TSC), there might exist intra-class variance, i.e. the variations between objects of the same class might be numerous and of various aspects. Using clustering as a pre-processing step to supervised classification is not new and is a solution to deal with intra-class variance. The idea is to pre-process the data set by grouping together similar objects and to highlight local patterns that might be class-discriminant: e.g., Vilalta et al. [26] suggest a pre-processing step by supervised (per-class) clustering using Expectation Maximization to enhance the predictive performance of Naive Bayes classifier. In order to be able to derive interesting features, we will use an unsupervised coclustering technique as described in the following.

A curve can be seen as a set of points (X, Y), described by their abscissa and ordinate values. A set of curves is then also a set of points (C_{id}, X, Y) where C_{id} is the curve identifier. This tridimensional representation (one categorical variable and two numerical variables) of a curve data set is needed to apply

coclustering methods. Indeed, the goal is to partition the categorical variable and to discretize the numerical variables in order to obtain clusters of curves and intervals for X and Y. The result is a tridimensional grid whose cells are defined by a group of curves, an interval for X and an interval for Y.
For that purpose, we use the coclustering method KHC [6] (Khiops Coclustering). Originally designed for clustering functional data [23], it is also suitable for the particular case of curve data as defined above and it is directly applicable for our pre-processing step. KHC method is based on a piecewise constant non-parametric density estimation and instantiates the generic MODL approach [4] (Minimum Optimized Description Length) – which is similar to a Bayesian Maximum A Posteriori (MAP) approach. The optimal model M, i.e. the optimal grid, is obtained by optimization of a Bayesian criterion, called *cost*. The *cost* criterion bets on a trade-off between the accuracy and the robustness of the model and is defined as follows:

$$\mathit{cost}(M) = -\log(\underbrace{p(M \mid D)}_{\text{posterior}}) = -\log(\underbrace{p(M)}_{\text{prior}} \times \underbrace{p(D \mid M)}_{\text{likelihood}})$$

Using a hierarchical prior (on the parameters of a data grid model) that is uniform at each stage of the hierarchy, we obtain an analytic expression for the *cost* criterion:

$$\begin{aligned}
\mathit{cost}(M) &= \log n + 2\log N + \log B(n, k_C) && (1)\\
&+ \log\binom{N+k-1}{k-1} + \sum_{i_C=1}^{k_C} \log\binom{N_{i_C}+n_{i_C}-1}{n_{i_C}-1} && (2)\\
&+ \log N! - \sum_{i_C=1}^{k_C}\sum_{j_X=1}^{k_X}\sum_{j_Y=1}^{k_Y} \log N_{i_C j_X j_Y}! && (3)\\
&+ \sum_{i_C=1}^{k_C} \log N_{i_C}! - \sum_{i=1}^{n} \log N_i! + \sum_{j_X=1}^{k_X} \log N_{j_X}! + \sum_{j_Y=1}^{k_Y} \log N_{j_Y}! && (4)
\end{aligned}$$

where n is the number of curves, N the number of points, k_C (resp. k_X, k_Y) is the number of clusters of curves (resp. the number of intervals for X and Y), k the number of cells of the data grid, n_{i_C} the number of curves in cluster i_C, N_i the number of points for curve i and N_{i_C} (resp. N_{j_X}, N_{j_Y}, $N_{i_C j_X j_Y}$) is the cumulated number of points for curves of cluster i_C (resp. for interval j_X of X, interval j_Y of Y, for cell (i_C, j_X, j_Y) of the data grid. Notice that $B(n, k_C)$ is the number of divisions of n elements into k subsets. The two first lines stand for the prior and the two last lines relates to the likelihood of the model. Intuitively, low *cost* means high probability ($p(M \mid D)$) that the model M arises from the data D. From an information theory point of view, according to [24], the negative logarithms of probabilities may be interpreted as code length. Thus, the *cost* criterion may also be interpreted as the code length of the grid model plus the code length of data D given the model M, according to the Minimum Description Length principle (MDL [14]). Here, low *cost* means high compression of the data using the model M.

The *cost* criterion is optimized using a greedy bottom-up strategy, (i) starting with the finest grained model, (ii) considering all merges between adjacent clusters or intervals, for the curve and dimension variables, and (iii) performs the best merge if the criterion decreases after the merge. The process loops until no further merge improves the criterion. The obtained grid constitutes a non-parametric estimator of the joint density of the curves and the dimensions of points.

KHC is parameter-free, robust (avoids over-fitting), handles large curve data sets with several millions of data points and its time complexity is $\Theta(N\sqrt{N}\log N)$ (sub-quadratic) where N is the number of data points: thus, KHC meets our problem needs (for full details, see [6]).

An example of visualization of coclustering results. The Fig. 2 shows an example of visualization of two clusters of curves of the optimal grid obtained on ARSim data set (in DDV representation). Figure (a) (resp. (b)) shows a cluster whose curves are essentially from class c_1 (blue in Fig. 1 of the motivation example), (resp. c_2, red in the same example). The optimal grid obtained with KHC is made up of 43 clusters of curves, 13 intervals for X and 12 intervals for Y (i.e. DDV values). The joint density estimation (i.e. the optimal grid) is much finer than needed by the classification problem. Indeed, the ARSim

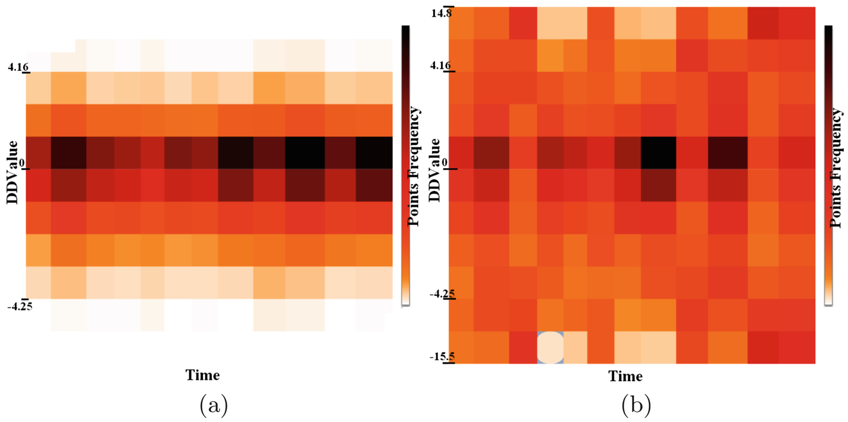


Fig. 2. Representation of the frequency of the cells for two clusters of curves obtained with KHC on ARSim data set (in DDV representation, DDV on y-axis and time on x-axis): (a) a cluster whose curves are mostly of class c_1; (b) a cluster whose curves are mostly of class c_2. For a given cell, stronger color indicates high point frequency.

data set is a 2-class classification problem and we found 43 groups of curves. This finer granularity gives us the potential for finer class characterization if the data representation is relevant for the task.

3.3 Feature Construction

Feature construction for TSC problems [21] aims at capturing class-relevant properties for describing time series. The generated features goes from simple ones like minimum, maximum, mean, standard deviation of time series to more complex ones like e.g., coefficients of spectral decompositions [20,32] or local pattern extracted from temporal abstractions of the series [2]. The main advantage of feature-based approaches is the final transactional (or vector) representation of the data which is suitable for conventional classifiers like Naive Bayes or decision trees. In our process, we generate features from coclustering results as follows.

For each coclustering result obtained with KHC on a data representation (D_{orig}, D_{DV}, D_{DDV}, D_{IV}, D_{IIV}, D_{PS}, D_{ACF}), we create a set of new features: $\mathcal{F}_{orig}, \mathcal{F}_{DV}, \mathcal{F}_{DDV}, \mathcal{F}_{IV}, \mathcal{F}_{IIV}, \mathcal{F}_{PS}, \mathcal{F}_{ACF}$. The new features are the descriptive attributes of the new data set whose objects are curves.

Let D_{rep} be one of the seven representations described above. Let $M_{rep} = KHC(D_{rep})$ be the tridimensional optimal grid obtained by coclustering with KHC on D_{rep}. We denote k_C the number of clusters of M_{rep} and k_Y the number of intervals of M_{rep} for dimension Y. We then create similarity-based features and histogram features.

Similarity-Based Features
Considering the good performance of (dis)similarity-based approaches (e.g., ED-NN and DTW-NN), we define a dissimilarity index based on the *cost* criterion.

Definition 1 (Dissimilarity index). *The dissimilarity between a curve τ_i and a cluster c_j of the optimal grid M_{rep} is defined as:*

$$d(\tau_i, c_j) = cost(M_{rep|\tau_i \cup c_j}) - cost(M_{rep})$$

i.e., the difference of cost between the optimal model M_{rep} and the model $M_{rep|\tau_i \cup c_j}$ (the optimal grid in which we add the curve τ_i to the cluster of curves c_j).

Intuitively, d measures the perturbation brought by the integration of a curve into a cluster of curves of the optimal grid (i.e. according to the *cost* criterion used for grid optimization). In terms of code length, if a curve τ_i is similar to the curves of cluster c_j, the total code length of the data is not much different from the total code length of the data plus τ_i. Thus, small values of $d(\tau_i, c_j)$ indicate that τ_i is similar to the curves of c_j whereas high values of d ($d(\tau_i, c_j) \gg 0$) mean that τ_i does not look like the curves of c_j.
According to the dissimilarity index d, we generate the following features:

– k_C numerical features (one for each cluster c_j of curves of M_{rep}). The value for a curve τ_i is the difference $d(\tau_i, c_j)$. Thus, for a given curve τ_i, these features tell how τ_i is similar to the clusters of curves of the optimal grid (according to d).
– One categorical feature indicating the index j of the cluster of curves that is the closest to a curve τ_i according to the dissimilarity d defined above (i.e., $\arg\min_j d(\tau_i, c_j)$).

Histogram Features
Taking up the idea of interpretable features (see motivating example and Fig. 1), we also generate the following features:

– k_Y numerical features (one for each interval i_Y of Y from M_{rep}) whose value for a curve τ_i is the number of points of τ_i in interval i_Y.

These histogram features quantify the presence of a curve in intervals of Y obtained in the coclustering step.
For a given curve τ_i, we now have the following informations provided by the new features (for each representation): (i) the dissimilarity values between τ_i and all the clusters of curves, (ii) the index of the closest cluster of curves and (iii) the number of points of τ_i in each interval of Y.

3.4 Supervised Classification Algorithm

We saw that our feature construction process may generate hundreds of new features for each representation. The whole set of features $\mathcal{F}_{tot}$ for our new data set may contain thousands of attributes. Therefore, the classifier at the end of our process has to be capable of handling a large number of attributes but also selecting the relevant attributes for the classification task. At this stage, we could use conventional classifiers like decision trees or SVM. However, we choose the Selective Naive Bayes classifier (SNB) that is parameter-free, performs efficient feature selection and outperforms classical Naive Bayes [5]. Notice that SNB exploits pre-processing techniques that discretize numerical variables, group values of categorical variables, weight and select features w.r.t. class-relevance by using robust conditional density estimators and following the MODL approach (see [3,4]). Thus, the generated features benefit from these pre-processing techniques and preserve a potential of interpretability; we lead specific experiments in the next section to support this claim. Moreover, SNB is parameter-free, so is the whole feature construction process. Its time complexity is $\Theta(KM \log(KM))$, where M is the number of objects and K the number of features.

4 Experimental Validation

The implementation of the classification process is based on existing tools (KHC for coclustering and SNB for supervised classification[2]). Connections between the

[2] KHC and SNB are both available at http://www.khiops.com.

tools are scripted using MATLAB. The whole process is named MODL-TSC. The experiments are led to discuss the following questions:

$\mathcal{Q}_1$ Is MODL-TSC comparable with competitive contenders of the state-of-the-art in terms of accuracy?

$\mathcal{Q}_2$ MODL-TSC employs and combines several representations. Are they all useful? Do they all bring the same impact?

$\mathcal{Q}_3$ What kind of data insight do we gain using the coclustering-based features?

4.1 Protocol

We experiment our process on 51 time series data sets: 42 data sets are from UCR [17] and 9 new data sets introduced in [1]. A brief description of the data is given in Table 1. The benchmark data sets offer a wide variety in terms of application domains, number of series, length of series and number of class values. We lead experiments in a predefined train-test setting for each data set (see [17]). We compare the predictive performance of our process, called MODL-TSC, with a baseline, two of the most effective alternative approaches and a recently introduced interpretable classifier:

- ED-NN: the Nearest Neighbor classifier using the Euclidean distance. This approach is considered as a baseline.
- DTW-NN: the Nearest Neighbor classifier using the elastic distance Dynamic Time Warping, considered as hard to beat in the literature (see [28])
- TSC-ENSEMBLE [1] exploits multiple representations via an ensemble method and the NN algorithm. Its performance is comparable to DTW-NN
- FAST-SHAPELETS [22] mines shapelets (i.e., class relevant time series subsequences) that might be embedded in e.g., a decision tree

4.2 Results

For fair comparisons, we have rerun the experiments using implementations of ED-NN, DTW-NN, FAST-SHAPELETS and TSC-ENSEMBLE provided by E. Keogh, A. Bagnall and their teams. Performance results in terms of accuracy are reported in Table 1. The best result for each data set is written in bold. The last column indicates how many features per representation MODL-TSC has generated.

Comparisons with state-of-the-art. Firstly, global results (mean accuracy, number of wins and mean rank) show that MODL-TSC is very competitive compared to state-of-the-art methods. Proceeding the Friedman test [8] (at significance level $\alpha = 0.05$) and the Nemenyi post-hoc test lead us to the critical difference diagram in Fig. 3. Two groups of approaches emerge and we observe that MODL-TSC, TSC-ENS and DTW-NN perform significantly better than ED-NN and FAST-SHAPELETS; and there is no significant difference of performance

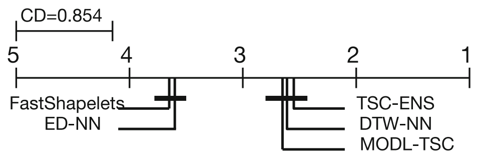


Fig. 3. Representation of difference of performance by critical difference diagram between MODL-TSC, ED-NN, DTW-NN, TSC-ENS and FAST-SHAPELETS.

inside each group. We also run Wilcoxon's sign rank test for pairwise comparisons (also with $\alpha = 0.05$) which confirms this result. This global view of performance results confirms that MODL-TSC is very competitive compared to two of the most effective contenders of the state-of-the-art and performs better than the baseline ED-NN and the very recent FAST-SHAPELETS approach.

Secondly, we observe the remarkable performance of MODL-TSC on ARSim, ElectricDevices, FordA and OSULeaf data. On these data, we outperform DTW-NN and TSC-ENSEMBLE: the difference of test accuracy is at least 10. Here, the added-value of the data representations (i.e., the new features) is at work. TSC-ENSEMBLE (exploiting only three representations) and DTW-NN (working in time domain) obtain only poor accuracy results. Conversely the performance of MODL-TSC is very low on Coffee, DiatomSizeReduction, ECGFiveDays and OliveOil data. The difference of test accuracy (about 10 compared with DTW-NN and TSC-ENSEMBLE) is now to our disadvantage. We think that this poor performance might due to one reason: the training set size of these data sets is very small (less than 30 of curves) and it could be insufficient for either learning relevant coclusters or learning a predictive model without over-fitting. Indeed, e.g., for OliveOil data, there are only 30 training curves, no cluster of curves is found by KHC whatever the representation and most generated features from intervals of the Y-axis are considered irrelevant by the pre-processing step of SNB classifier.

Added-value of the representations. In Table 1, we also report accuracy results of SINGLE-MODL-TSC using only one representation. We observe that using a single representation provide poor average accuracy results. Almost always, MODL-TSC using several representations outperforms SINGLE-MODL-TSC on $\mathcal{F}_{orig}$ (resp. $\mathcal{F}_{DV}$, $\mathcal{F}_{DDV}$, $\mathcal{F}_{IV}$, $\mathcal{F}_{IIV}$, $\mathcal{F}_{PS}$, $\mathcal{F}_{ACF}$). In some cases (e.g., ItalyPowerDemand, MALLAT or MedicalImages), the good performance of MODL-TSC can be attributed to the combination of several representations. Indeed, the gap of test accuracy between any SINGLE-MODL-TSC and MODL-TSC is about 10; thus the combination of features coming from different views of the data improves accuracy results. In other cases (e.g., ARSim or wafer), the good performance seems to be due to only one (or at most two) representation while the other representations are ignored. As an example, for ARSim data, the DDV representation is the most relevant. KHC obtains 43 clusters of curves and 12 intervals for Y_{DDV}. Most of the clusters are almost pure (only one class of curves per cluster). Moreover, as we saw in Fig. 2(a) and (b), the number of points in

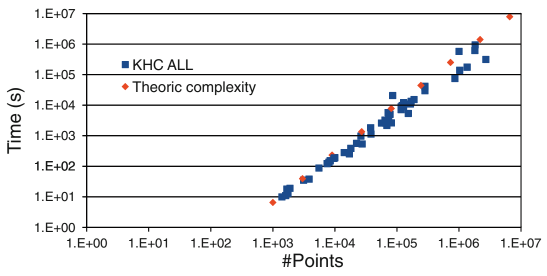


Fig. 4. Cumulative running time results for KHC on the seven studied representations and comparison with the announced theoretical complexity.

intervals generated by KHC above 4.16 and below −4.25 are class-discriminant since curves of class 1 almost never have points in these regions.

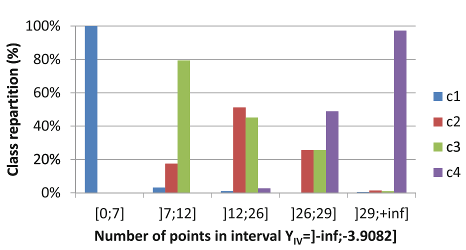
These experiments recall the very importance of representations in TSC problems and particularly in our feature construction process. Even the simple representations we chose to illustrate our process show good predictive performance. Depending on the application, we may still hope some improvement in performance if we could rely on expert domain knowledge to select relevant representations to use in our generic process.

4.3 Running Time Results

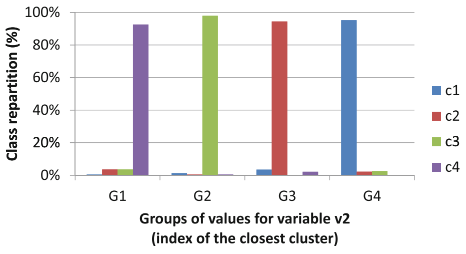
Among the three steps of our process, the coclustering step is the most demanding in terms of computational time. Moreover, for the largest data sets of our benchmark, other steps (building representations and features and learning the SNB classifier) are negligible compared with the coclustering step. For a better understanding of how much time KHC costs, we report in Fig. 4 the cumulative running time of KHC on the seven representations for each data set w.r.t. the number of points N in the training data set. We also draw the theoretical complexity announced for KHC in previous section: $\alpha N\sqrt{N}\log N$, where $\alpha = 3.10^{-5}$. We observe that for the most difficult data sets (ARSim, FordA and FordB, from 1 million to 1.8 million data points), KHC runs during one day for each representation to reach the optimal grid.

4.4 Interpretation: An Example

If we consider the cumulative integral (IV) representation of TwoPatterns data, the optimal grid obtained by KHC is made of 224 clusters of curves, 11 intervals for X and 9 intervals for Y_{IV}. According to the MODL pre-processing techniques, among all the attributes generated from all representations, the two most relevant attributes are from the IV representation:



(a)



(b)

Fig. 5. Histogram representation of class repartition for discretization of variable v_1 and value grouping of variable v_2.

1. v_1, the number of points in interval $I_{Y_{IV}} =]-\infty; -3.9082]$
2. v_2, the index of the closest cluster

In the supervised learning step, the discretization for v_1 and the value grouping for v_2 provide the following contingency tables represented as histograms (see Fig. 5(a) and (b)): We observe (Fig. 5(a)) that the number of points p of a curve in interval $I_{Y_{IV}}$ (i.e. the number of points with value less than -3.9082) is class relevant. Indeed, in the learning phase, curves such that $p \leq 7$ are of class c_1; when $p > 29$ (about 23 % of the points of the curve), curves are mostly of class c_4 and when $7 < p \leq 12$ they are mostly of class c_3. This type of feature is similar to the ones in the motivating example and Fig. 1: for a given representation, some regions of y-axis (delimited by intervals) will be class-discriminant and the number of points of an incoming curve in this interval will also be class-discriminant.

In Fig. 5(b), we firstly see that, for variable v_2 ("index of the closest cluster"), MODL pre-processing by supervised value grouping provide 4 groups: G_1, (resp. G_2, G_3 et G_4) made of 56, (resp. 53, 53 et 62) indexes of clusters that are mostly of class c_4 (resp. c_3, c_2, c_1). The attribute v_2 is then class-relevant. Indeed, for example, if j is the index of cluster, that is the closest to a curve τ_i, and belongs to G_2 (i.e. $j \in G_2$), then τ_i is considered very similar to curves of class c_3. Moreover, the variable "index of the closest cluster" is an indicator of the relevance of the representation in our process for the current TSC problem. In this example, attribute v_2 alone, is enough to characterize 95 % of the data, therefore, IV data representation is very relevant for characterizing the classes of TwoPatterns data. Conversely, for the original representation (D_V), the optimal grid obtained with KHC is made of 255 clusters of curves but MODL pre-processing indicates that the variable "index of the closest cluster" is not relevant to characterize the classes of TwoPatterns; as a consequence, SINGLE-MODL-TSC on $\mathcal{F}_{orig}$ shows bad test accuracy results.

Table 1. Description and characteristics of time series benchmark data sets. Comparisons of accuracy results for MODL-TSC, FASTSHAPELETS, DTW-NN, ED-NN, and TSC-ENSEMBLE. Accuracy results for SINGLE-MODL-TSC using each single representation and number of features generated per representation (last column).

Data	#Train	#Test	Length	#Classes	FASTSHAPELETS	DTW-NN	ED-NN	TSC-ENS	MODL-TSC	$\mathcal{F}_{orig}$	$\mathcal{F}_{DV}$	$\mathcal{F}_{DDV}$	$\mathcal{F}_{IV}$	$\mathcal{F}_{IIV}$	$\mathcal{F}_{PS}$	$\mathcal{F}_{ACF}$	#features
50words	450	455	270	50	44.29	**69.01**	63.08	63.96	68.57	60.22	59.78	59.12	47.03	34.51	28.35	51.65	645/23/56/646/293/30/830
Adiac	390	391	176	37	48.59	59.65	51.10	62.15	**64.19**	49.10	48.34	34.53	19.95	17.90	34.27	02.05	27/30/29/32/34/31/63
ARSim	2000	2000	500	2	91.40	60.36	61.13	67.85	**99.95**	61.05	93.20	99.95	50.00	50.00	98.55	54.90	25/14/8/29/34/18/25
Beef	30	30	470	5	55.33	50.00	53.33	**60.00**	53.33	46.67	30.00	20.00	36.67	33.33	40.00	50.00	15/3/3/18/22/7/15
CBF	30	900	128	3	94.71	**99.67**	85.22	82.44	96.56	98.00	33.11	33.11	66.56	78.44	33.11	53.11	22/24/22/38/52/20/28
ChlorineConcentration	467	3840	166	3	58.31	64.84	65.00	**67.76**	58.15	56.56	57.79	57.11	53.26	53.26	55.00	53.26	56/72/104/66/82/33/64
CinC_ECG_torso	40	1380	1639	4	82.64	65.07	89.71	**94.57**	87.61	48.04	78.91	71.59	29.86	33.33	67.25	43.48	15/11/13/21/22/15/22
Coffee	28	28	286	2	**93.21**	82.14	75.00	**82.14**	64.29	53.57	53.57	53.57	53.57	64.29	53.57	53.57	113/74/81/102/76/25/85
Cricket_X	390	390	300	12	47.32	**77.69**	57.44	60.26	64.62	49.23	36.41	34.10	33.33	27.69	37.18	34.87	113/37/27/112/87/28/81
Cricket_Y	390	390	300	12	49.52	**79.23**	64.36	65.90	74.87	63.33	20.26	29.74	44.10	35.13	31.54	39.23	105/26/25/113/80/26/72
Cricket_Z	390	390	300	12	45.27	**79.23**	62.05	62.56	65.64	56.67	25.13	31.03	31.54	30.26	41.54	33.08	12/15/20/15/19/25/19
DiatomSizeReduction	16	306	345	4	88.30	**96.73**	93.46	94.44	80.39	30.07	59.15	76.14	76.80	71.24	70.59	30.07	17/14/14/21/28/15/22
Earthquakes	322	139	512	2	**73.38**	70.50	69.78	**73.38**	71.94	71.94	74.82	74.82	74.82	74.82	74.82	74.82	16/9/7/23/23/14/16
ECG200	100	100	96	2	77.30	77.00	88.00	**89.00**	79.00	79.00	80.00	75.00	69.00	66.00	73.00	73.00	544/43/50/195/151/6/27
ECGFiveDays	23	861	136	2	**99.59**	76.77	79.67	**98.72**	64.46	62.72	49.71	49.71	63.76	53.89	49.71	73.64	124/61/80/163/109/126/137
ElectricDevices	8953	7745	96	7	48.68	67.02	54.41	62.21	**73.92**	66.06	64.62	65.77	60.39	53.78	57.90	59.61	33/29/21/42/65/24/24
FaceAll	560	1690	131	14	58.93	**80.77**	71.36	71.60	70.47	63.61	58.58	35.86	45.74	25.50	31.78	52.78	16/12/29/22/34/11/18
FaceFour	24	88	350	4	**91.02**	82.95	78.41	86.36	89.77	90.91	15.91	37.50	38.64	15.91	34.09	72.73	21/17/18/28/46/21/21
FacesUCR	200	2050	131	14	67.17	**90.49**	76.93	83.71	74.88	66.63	48.54	35.02	42.44	33.66	34.24	48.63	25/36/26/31/44/48/47
Fish	175	175	463	7	80.28	**83.43**	78.29	79.43	77.14	70.29	68.57	65.14	42.29	27.43	35.43	12.57	185/87/49/233/259/156/78
FordA	3571	1320	500	2	84.77	72.42	68.64	84.85	**98.64**	80.23	90.68	95.38	73.26	62.50	77.73	83.79	111/67/64/236/274/147/70
FordB	3601	810	500	2	72.59	65.93	59.63	**73.46**	67.28	62.84	64.32	66.05	59.38	54.32	62.72	63.83	29/15/13/20/25/23/20
GunPoint	50	150	150	2	93.93	90.67	91.33	94.67	**98.00**	84.00	94.00	85.33	77.33	71.33	79.33	88.67	76/365/308/94/118/126/83
HandOutlines	1000	300	2709	2	87.33	**88.14**	86.25	87.57	87.30	85.95	69.73	66.22	75.68	76.49	68.92	78.38	57/30/37/49/62/62/50
Haptics	155	308	1092	5	38.44	37.66	37.01	40.26	**42.86**	41.56	41.56	31.82	32.47	37.01	35.39	37.01	119/24/27/101/94/45/99
InlineSkate	100	550	1882	7	25.91	**38.36**	34.18	31.82	34.55	24.91	29.64	28.91	23.45	22.91	32.18	22.73	12/7/8/13/14/7/13
ItalyPowerDemand	67	1029	24	2	90.51	95.04	**95.53**	94.95	81.05	68.80	49.85	49.85	70.26	72.89	49.85	70.75	74/21/20/49/45/24/51
Lightning2	60	61	637	2	70.49	86.89	75.41	77.05	73.77	65.57	68.85	65.57	65.57	63.93	67.21	62.30	56/21/21/38/36/22/39
Lightning7	70	73	319	7	59.73	72.60	57.53	69.86	**73.97**	68.49	53.42	50.68	67.12	52.05	45.21	53.42	22/24/23/33/44/29/27
MALLAT	55	2345	1024	8	**96.72**	93.39	91.43	86.35	92.92	81.02	68.96	12.32	76.42	65.20	68.57	81.96	47/29/29/36/38/42/41
MedicalImages	381	760	99	10	56.70	**73.68**	68.42	70.00	67.24	51.32	53.16	51.58	53.16	57.63	54.34	53.16	20/14/6/16/18/15/12
MoteStrain	20	1252	84	2	78.28	83.47	87.86	86.34	**90.89**	73.80	92.01	53.91	76.20	68.69	53.91	74.36	64/40/34/101/147/83/79
NIFECG_Thorax1	1800	1965	750	42	75.43	79.03	82.90	80.61	**87.58**	75.78	64.58	41.02	56.69	41.53	61.32	70.03	70/42/39/102/142/88/81
NIFECG_Thorax2	1800	1965	750	42	78.94	86.46	87.99	87.99	**89.72**	79.80	77.20	52.77	65.65	48.09	68.04	78.37	104/107/30/105/102/31/64
OliveOil	30	30	570	4	78.67	**86.67**	**86.67**	80.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	45/29/16/40/56/26/46
OSULeaf	200	242	427	6	64.09	59.09	51.65	57.85	**74.38**	49.59	60.33	72.73	38.43	35.12	42.98	48.35	18/59/62/12/17/7/11
SonyRobotSurface	20	601	70	2	68.55	72.55	69.55	**74.21**	68.05	60.40	72.21	81.53	55.07	53.24	42.93	42.93	140/76/82/11/18/11/11
SonyRobotSurface2	27	953	65	2	78.52	83.11	85.94	86.15	**87.51**	53.83	76.50	73.66	74.61	73.14	86.25	61.70	148/515/839/103/122/142/79
StarLightCurves	1000	8236	1024	3	93.68	90.66	84.88	93.99	**95.84**	95.35	95.65	93.37	89.78	87.21	78.91	87.35	26/29/32/33/46/19/30
SwedishLeaf	500	625	128	15	73.07	79.20	78.88	84.80	**90.56**	80.64	79.52	69.12	56.64	38.08	57.44	76.16	24/137/27/22/36/27/24
Symbols	25	995	398	6	93.24	**94.97**	89.95	92.46	86.83	86.63	75.08	64.62	77.39	70.35	56.08	84.72	19/14/14/25/36/13/22
SyntheticControl	300	300	60	6	91.90	**99.33**	88.00	91.33	97.67	92.67	54.00	39.00	90.00	86.33	49.67	64.00	32/13/5/35/39/22/35
Trace	100	100	275	4	99.80	**100.00**	76.00	81.00	**100.00**	98.00	91.00	19.00	89.00	63.00	85.00	72.00	12/10/135/14/18/13/14
TwoLeadECG	23	1139	82	2	**90.97**	90.43	74.71	89.03	84.81	74.36	64.35	84.81	63.48	62.77	49.96	75.24	765/42/40/234/156/17/38
TwoPatterns	1000	4000	128	4	88.65	**100.00**	90.68	90.10	98.83	25.70	53.70	25.88	98.33	83.98	30.95	51.65	136/98/64/90/64/48/91
uWaveGestureLibrary_X	896	3582	315	8	70.68	72.75	73.93	74.65	**80.85**	73.06	55.83	46.34	67.59	60.25	35.68	51.73	312/686/47/155/122/72/81
uWaveGestureLibrary_Y	896	3582	315	8	60.83	63.40	66.16	67.17	**69.96**	64.10	45.37	46.01	60.66	53.43	34.79	42.24	276/542/53/126/102/66/74
uWaveGestureLibrary_Z	896	3582	315	8	63.56	65.83	64.96	66.36	**74.06**	67.59	59.32	52.40	63.12	53.88	36.24	52.01	298/251/54/137/114/74/87
wafer	1000	6174	152	2	99.64	97.99	99.55	99.72	**100.00**	97.84	100.00	100.00	97.31	96.24	96.71	97.92	149/183/178/52/74/23/47
WordsSynonyms	267	638	270	25	40.61	**64.89**	61.76	62.54	61.44	50.94	53.76	49.37	39.97	30.72	25.39	39.81	194/179/77/118/89/55/129
yoga	300	3000	426	2	73.07	83.63	83.03	**83.67**	73.33	66.40	70.37	72.00	68.27	65.03	61.33	61.00	50/54/54/72/84/38/41
Mean Acc					73.23	78.06	73.89	77.45	77.44	65.98	61.20	55.30	59.26	53.48	53.27	57.54	
#wins					5	18	2	8	19								
MODL-TSC wins vs.					35	25	34	29	-	49	47	49	50	50	50	49	
Average rank					3.6667	2.6078	3.5784	2.5294	2.6176								

5 Conclusion and Perspectives

We have suggested MODL-TSC, a simple yet effective and generic feature construction process for time series classification problems (TSC). Our process is parameter-free, easy to use and the generated features offer a high potential of interpretation. The three main steps of the process are: (i) transforming data for generating multiple data representations; (ii) coclustering on each representation; (iii) constructing new features from coclustering results. The new data set is made of objects (time series identifiers) and descriptive attributes from the various representations. To predict the class of new incoming time series, we use the Selective Naive Bayes classifier (SNB). The time complexity of our process is sub-quadratic, thus time-efficient. Experimental results show that the performance of MODL-TSC is highly competitive and comparable with two of the most accurate approaches of the state-of-the-art (namely, DTW-NN and TSC-ENS). In addition, MODL-TSC embraces the eager paradigm and unlike the lazy approaches (ED-NN, DTW-NN and TSC-ENS), our approach has a proper learning phase and can deploy fast enough for real-world applications. Moreover, a qualitative study has shown that the generated features give some insight in the data representations: we are able to qualify the adequacy of a representation for solving the TSC problem at hand and to identify class-discriminating regions of values from data representations embedded in our process.

The results of this work are promising and also confirm the importance of representations in TSC problems. Indeed, depending on the application domain, a particular transformation will facilitate the discovery of class relevant patterns. Moreover, the combination of multiple representations with MODL-TSC leads to highly competitive predictive performance. We have used only a few simple representations in the time, frequency and correlation domains to demonstrate that our feature construction approach is well-founded. The literature offers plenty of relevant data representations (see [27] for a wide view). Notice also that designing new representations is still a hot topic (see e.g., [19]). It gives a large potential of improvement for MODL-TSC on data sets and applications where we are less performant than DTW-NN and TSC-ENSEMBLE since our methodology allows us to use a large spectrum of representations.

Acknowledgments. We wish to thank Anthony Bagnall and his team from University of East-Anglia for providing TSC-ENSEMBLE prototype, Eamonn Keogh and his team from University of California Riverside for providing prototypes of DTW-NN and FAST-SHAPELETS.

References

1. Bagnall, A., Davis, L.M., Hills, J., Lines, J.: Transformation based ensembles for time series classification. In: SDM'12, pp. 307–318 (2012)
2. Batal, I., Sacchi, L., Bellazzi, R., Hauskrecht, M.: Multivariate time series classification with temporal abstractions. In: FLAIRS'09 (2009)

3. Boullé, M.: A bayes optimal approach for partitioning the values of categorical attributes. J. Mach. Learn. Res. **6**, 1431–1452 (2005)
4. Boullé, M.: MODL: a bayes optimal discretization method for continuous attributes. Mach. Learn. **65**(1), 131–165 (2006)
5. Boullé, M.: Compression-based averaging of selective naive Bayes classifiers. J. Mach. Learn. Res. **8**, 1659–1685 (2007)
6. Boullé, M.: Functional data clustering via piecewise constant nonparametric density estimation. Pattern Recogn. **45**(12), 4389–4401 (2012)
7. Buza, K.A.: Fusion methods for time-series classification. Ph.D. thesis, University of Hildesheim (2011)
8. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
9. Rodríguez, J.J., Alonso, C.J., Boström, H.: Learning first order logic time series classifiers: rules and boosting. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 299–308. Springer, Heidelberg (2000)
10. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.J.: Querying and mining of time series data: experimental comparison of representations and distance measures. PVLDB **1**(2), 1542–1552 (2008)
11. Eruhimov, V., Martyanov, V., Tuv, E.: Constructing high dimensional feature space for time series classification. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 414–421. Springer, Heidelberg (2007)
12. Geurts, P.: Pattern extraction for time series classification. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, p. 115. Springer, Heidelberg (2001)
13. Grabocka, J., Nanopoulos, A., Schmidt-Thieme, L.: Invariant time-series classification. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part II. LNCS, vol. 7524, pp. 725–740. Springer, Heidelberg (2012)
14. Grünwald, P.: The Minimum Description Length Principle. MIT Press, Cambridge (2007)
15. Hidasi, B., Gáspár-Papanek, C.: ShiftTree: an interpretable model-based approach for time series classification. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part II. LNCS, vol. 6912, pp. 48–64. Springer, Heidelberg (2011)
16. Kadous, M.W., Sammut, C.: Classification of multivariate time series and structured data using constructive induction. Mach. Learn. **58**(2–3), 179–216 (2005)
17. Keogh, E., Zhu, Q., Hu, B., Hao. Y., Xi, X., Wei, L., Ratanamahatana, C.A.: The UCR time series classification/clustering page (2011). http://www.cs.ucr.edu/~eamonn/time_series_data/
18. Liao, T.W.: Clustering of time series data - a survey. Pattern Recogn. **38**(11), 1857–1874 (2005)
19. Lines, J., Davis, L.M., Hills, J., Bagnall, A.: A shapelet transform for time series classification. In: KDD'12, pp. 289–297 (2012)
20. Mörchen, F.: Time series feature extraction for data mining using DWT and DFT. Technical report, Philipps Univeristy Marburg (2003)
21. Nanopoulos, A., Alcock, R., Manolopoulos, Y.: Feature-based classification of time-series data. In: Mastorakis, N., Nikolopoulos, S.D. (eds.) Information Processing and Technology, pp. 49–61. Nova Science (2001)
22. Rakthanmanon, T., Keogh, E.: Fast shapelets: a scalable algorithm for discovering time series shapelets. In: SIAM DM'13 (2013)

23. Ramsay, J., Silverman, B.: Functional Data Analysis. Springer, New York (2005)
24. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal (1948)
25. Stefan, A., Athitsos, V., Das, G.: The move-split-merge metric for time series. Trans. Knowl. Data Eng. **25**, 1425–1438 (2013)
26. Vilalta, R., Rish, I.: A decomposition of classes via clustering to explain and improve naive bayes. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 444–455. Springer, Heidelberg (2003)
27. Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., Keogh, E.: Experimental comparison of representation methods and distance measures for time series data. Data Min. Knowl. Disc. **26**(2), 275–309 (2013)
28. Xi, X., Keogh, E.J., Shelton, C.R., Wei, L., Ratanamahatana, C.A.: Fast time series classification using numerosity reduction. In: ICML'06, pp. 1033–1040 (2006)
29. Xing, Z., Pei, J., Yu, P.S., Wang, K.: Extracting interpretable features for early classification on time series. In: SDM'11, pp. 247–258 (2011)
30. Yamada, Y., Suzuki, E., Yokoi, H., Takabayashi, K.: Decision-tree induction from time-series data based on a standard-example split test. In: ICML'03, pp. 840–847 (2003)
31. Ye, L., Keogh, E.J.: Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Min. Knowl. Disc. **22**(1–2), 149–182 (2011)
32. Zhang, H., Ho, T.-B., Lin, M.-S.: A non-parametric wavelet feature extractor for time series classification. In: Dai, H., Srikant, R., Zhang, Ch. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 595–603. Springer, Heidelberg (2004)

http://www.springer.com/978-3-319-08406-0

New Frontiers in Mining Complex Patterns
Second International Workshop, NFMCP 2013, Held in Conjunction with ECML-PKDD 2013, Prague, Czech Republic, September 27, 2013, Revised Selected Papers
Appice, A.; Ceci, M.; Loglisci, C.; Manco, G.; Masciari, E.; Ras, Z. (Eds.)
2014, XII, 261 p. 80 illus., Softcover
ISBN: 978-3-319-08406-0