Data Min Knowl Disc (2015) 29:400–422
DOI 10.1007/s10618-014-0349-y

Learning a symbolic representation for multivariate time series classification

Mustafa Gokce Baydogan · George Runger

Received: 9 March 2013 / Accepted: 3 January 2014 / Published online: 16 February 2014

Abstract Multivariate time series (MTS) classification has gained importance with the increase in the number of temporal datasets in different domains (such as medicine, finance, multimedia, etc.). Similarity-based approaches, such as nearest-neighbor classifiers, are often used for univariate time series, but MTS are characterized not only by individual attributes, but also by their relationships. Here we provide a classifier based on a new symbolic representation for MTS (denoted as SMTS) with several important elements. SMTS considers all attributes of MTS simultaneously, rather than separately, to extract information contained in the relationships. Symbols are learned from a supervised algorithm that does not require pre-defined intervals, nor features. An elementary representation is used that consists of the time index, and the values (and first differences for numerical attributes) of the individual time series as columns. That is, there is essentially no feature extraction (aside from first differences) and the local series values are fused to time position through the time index. The initial representation of raw data is quite simple conceptually and operationally. Still, a tree-based ensemble can detect interactions in the space of the time index and time values and this is exploited to generate a high-dimensional codebook from the terminal nodes of the trees. Because the time index is included as an attribute, each MTS is learned to be segmented by time, or by the value of one of its attributes. The codebook is processed with a second ensemble where now implicit feature selection is exploited

Responsible editor: M. J. Zaki.

M. G. Baydogan (✉)
Department of Industrial Engineering, Boğaziçi University,
Bebek, Istanbul 34342, Turkey
e-mail: mustafa.baydogan@boun.edu.tr

G. Runger
School of Computing, Informatics & Decision Systems Engineering,
Arizona State University, Tempe, AZ, USA

to handle the high-dimensional input. The constituent properties produce a distinctly different algorithm. Moreover, MTS with nominal and missing values are handled efficiently with tree learners. Experiments demonstrate the effectiveness of the proposed approach in terms of accuracy and computation times in a large collection multivariate (and univariate) datasets.

Keywords Supervised learning · Codebook · Decision trees

1 Introduction

Multivariate time series (MTS) classification is a supervised learning problem in which each example consists of one or more time series (attributes). MTS data is common in fields such as medicine, finance and multimedia. For example, in motion capture studies humans perform a series of tasks and the positions of joints are tracked by markers (McGovern et al. 2011). Learning scientists are interested in electroencephalography (EEG), (recordings of electrical activity at different locations along the scalp) to understand the perceived difficulty of a puzzle-solving task in a learning environment. Moreover, in the domain of relational marketing, the behavior of a customer is observed through time, and the interactions and responses are represented as MTS. Also, Orsenigo and Vercellis (2010) described a telecommunication application where customer loyalty was analyzed from the transactions of each customer in time periods and described by duration, economic value and number of calls of different type (i.e., cell to cell, cell to landline etc.).

The problem has been studied in fields such as statistics, signal processing and control theory (Kadous and Sammut 2005). The most common approach is to obtain a rectangular representation of MTS by transforming the set of input series to a fixed number of columns using different rectangularization approaches (Orsenigo and Vercellis 2010). For example, singular value decomposition (SVD) was used by Li et al. (2006), Li et al. (2007), Weng and Shen (2008). Any supervised learner can be trained on the transformed data for classification. These approaches assume that the attributes are numerical. However, some attributes might be nominal or contain missing values.

Another strategy is to modify the similarity-based approaches used for univariate time series (UTS). For example, Akl and Valaee (2010), Liu et al. (2009) focused on gesture recognition based on dynamic time warping (DTW) distance. DTW (Sakoe 1978) provides a similarity measure independent of certain non-linear variations in the time dimension, and is considered a strong solution for time series problems (Ratanamahatana and Keogh 2005). Another similarity approach uses a kernel function determined by pairwise similarities between the attributes of a MTS. Thus, Chaovalitwongse and Pardalos (2008) used kernels based on DTW for brain activity classification and Orsenigo and Vercellis (2010) proposed a temporal discrete SVM for MTS classification. Similarity is based on a term that depends on the warping distances (Orsenigo and Vercellis 2010). However, an important limitation of these approaches is that MTS are not only characterized by individual attributes, but also by the relationships between the attributes (Bankó and Abonyi 2012) and such information is not captured by the similarity between the individual attributes (Weng and Shen 2008).

Another important challenge for MTS classification is the high dimensionality introduced by multiple attributes and longer series. For longer time series, higher-level representations are proposed Lin et al. (2007), such as Fourier transforms, wavelets, piecewise polynomial models, etc. (Lin et al. 2003). Also, symbolic aggregate approximation (SAX) (Lin et al. 2007, 2012; Shieh and Keogh 2008), is a simple symbolic representation for univariate series that segments the series into fixed-length intervals (and a symbol represents the mean of the values). This representation is similar to Piecewise Aggregate Approximation (PAA) (Chakrabarti et al. 2002). In the multivariate scenario, two alternative representations (illustrated in Fig. 1) are considered (Kuksa 2012). MTS with M attributes and T observations can be discretized to obtain a one-dimensional (1D) representation using vector quantization approaches similar to those used for univariate series (Kuksa 2012). Alternatively, each attribute of MTS can be discretized and combined to obtain a two-dimensional (2D) representation of MTS.

Here we provide a classifier based on new symbolic representation for MTS (denoted as SMTS) with several important elements. SMTS considers all attributes of MTS simultaneously, rather than separately, to extract information contained in relationships. The symbols are learned from a supervised algorithm that does not require pre-defined intervals, nor features. An elementary representation is used that consists of the time index, and the values (and first differences) of the individual time series as columns. That is, there is essentially no feature extraction (aside from first differences) and the local series values are fused to time position through the time index.

Each MTS is concatenated vertically and each instance is labeled with the class of the corresponding time series. A tree learner [e.g., CART (Breiman et al. 1984) or C4.5 (Quinlan 1993)] is applied to the simple representation and each instance is assigned to a terminal node of the tree. That is, a symbol is associated with each time index (instance), instead of intervals of the time series. With R terminal nodes, the dictionary contains R symbols. Because the time index is included as an attribute, each MTS may be segmented by time, or by the value of one of its attributes. Consequently, the time order of the data can be incorporated in the model. At the next level of the learning

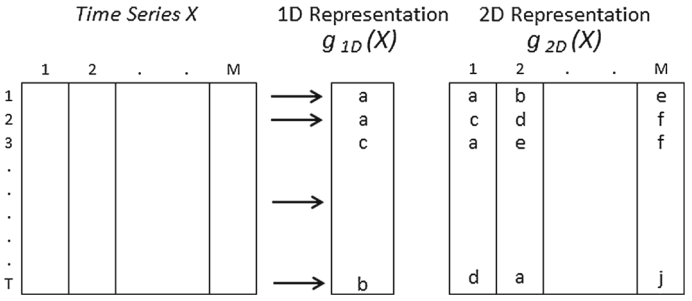

Fig. 1 Alternative representations for MTS (Kuksa 2012). MTS with M attributes and T observations are mapped to a 1D representation by the function g_{1D} (*left*) or a 2D representation in which each attribute of MTS is mapped to 1D representation by g_{2D} (*right*). Although the length of the symbolic representation is shown to be the same as T, it can be smaller based on the mapping strategy. Also, the 2D representation may have fewer columns than M depending on the mapping

algorithm, the set of instances from each MTS is summarized by the frequency over the R symbols. Because of the greedy nature of a single tree, the procedure to grow a tree is repeated to generate a tree ensemble (Breiman 2001). Consequently, each MTS is represented by a collection of distributions and the concatenated vector becomes the codebook for the second layer in the algorithm. A codebook is learned automatically from a simple representation of the raw data.

Interactions between the attributes of MTS are considered through the multivariate, tree-based symbol generation used in SMTS. Although our symbolic representation is of the same length as the time series, it does not generate a representation for each attribute of MTS. Therefore, SMTS scales well with a the number of attributes which makes it computationally efficient when compared to other approaches. Furthermore, an ensemble learner that scales well with a large number of attributes and long time series is used. SMTS can handle MTS examples of different lengths. Moreover, as discussed in a telecommunication application (Orsenigo and Vercellis 2010), data can be nominal (e.g., call type) for which similarity computations and other representations are not well-defined. MTS with nominal and missing values are handled efficiently with tree learners. Our experiments demonstrate the effectiveness of the proposed approach in terms of accuracy and computation times in both UTS and MTS datasets.

The remainder of this paper is organized as follows. Section 2 summarizes related work. Section 3 provides a brief background. Section 4 presents the problem and describes the framework. Section 5 demonstrates the effectiveness and efficiency of our proposed approach by testing on a full set of benchmark datasets from CMU (2012), Bache and Lichman (2013), Keogh et al. (2011) and Olszewski (2012). Section 6 provides conclusions.

2 Related work

The necessity of alternative representations for MTS classification was discussed by Kadous and Sammut (2005). A concept called a metafeature is introduced and used to represent a MTS. However, metafeatures must be defined by users in this approach. One of the metafeatures partitions the feature space using Voronoi tiling and selects these regions. The partitioning is not supervised in this approach and, as mentioned by Kadous and Sammut (2005), designing a good metafeature is not an easy task.

A 2D representation for MTS was considered by McGovern et al. (2011). Each attribute of MTS is discretized using SAX and salient attributes are identified from statistical performance measures. Patterns are identified on each individual attribute, without taking other attributes into account. Consequently, this approach is greedy because the relationships between the attributes may carry the important description of a complex system (Bankó and Abonyi 2012). Although the patterns are combined later, one may potentially miss a certain pattern that appears unimportant when time series are considered separately in the first step. Also, McGovern et al. (2011) required an approach for a multiclass extension since the rules were generated for binary classification.

Two MTS representations based on SAX to classify physiological data were presented by Ordonez et al. (2011). Multivariate words are generated by combining the symbols of each attribute at a particular time and a bag-of-patterns (BoP) approach is used to classify MTS. This considers the relationships between the time series by combining individual representations. However, the length of the words obtained by concatenating the symbols may increase substantially as the number of attributes increase. This potentially affects the effectiveness because longer words carry less information (i.e. curse of dimensionality). Also, this representation is not sensitive to warpings of the patterns because words are synchronized over time. Furthermore, the symbols are computed from a discretization of each time series separately, rather than from the multivariate distribution. Moreover, the fixed-length intervals have the potential to omit patterns that appear with different lengths (dilations) and be split across the time points (Fu 2011). A stacked Bags-of-Patterns was also proposed by Ordonez et al. (2011). This concatenates the representation of multiple univariate series into a single one. However, this representation does not take the relationships between attributes into account.

Time series similarity based on a BoW representation was also considered by Lin et al. (2012) for univariate time series classification. Time series were discretized by SAX (Lin et al. 2007) and time series were represented as words using the symbols generated by SAX. Similarity of the time series were then computed using the representations from document classification approaches. Unlike Lin et al. (2012), our approach generates symbols in a supervised manner and handles the multivariate case.

Also, Kudo et al. (1999) proposed an approach for multidimensional curve classification. Each attribute is discretized separately (as in the existing symbolic MTS methods). Also, each attribute of MTS is partitioned into rectangular regions of equal dimensions. Then classification rules are discovered based on the common regions through which only curves of one class pass. Their approach has similarities to McGovern et al. (2011) in terms of the discretization and rule generation. However, because the discretization does not consider the attributes simultaneously and the rules are discovered based on each individual attribute, there is a potential to miss important relationships between the attributes.

For image classification problems, Moosmann et al. (2008) trained trees on features extracted from image patches in a supervised manner and the terminal nodes were used as individual clusters. An image is represented as the frequency distribution of the cluster IDs from an image (visual codebook) and any supervised learner can be trained on the visual codebook. Unlike Moosmann et al. (2008), we do not sample patches and we do not generate features. Our simple representation is used to learn symbols directly. Furthermore, Moosmann et al. (2008) did not consider the location information (time index) as a feature during the tree learning as we do. Instead, they proposed to guide sampling to important locations of the images. They also use very few trees in the ensemble, whereas we consider a codebook of much higher dimension.

For univariate time series classification, a bag-of-features approach, TSBF, was proposed by Baydogan et al. (2013). TSBF summarizes time series with class probability estimates from tree ensembles. However, TSBF requires simple features (means, variances) to be extracted from subsequences of time series. Instead, SMTS does not

generate features. Each observation is taken as an instance and the symbols are learned. An extension of TSBF to the classification of MTS requires an appropriate integration of features and individual series as additional steps.

3 Background

Decision tree learners are comprehensible models with satisfactory accuracy that are successfully used in many applications. Univariate trees such as CART (Breiman et al. 1984) and C4.5 (Quinlan 1993) split data based on only one attribute at each node, and, thus, are limited to splits that are orthogonal to the attribute's axis (Brodley and Utgoff 1995).

Tree ensembles are proposed to mitigate the greedy nature of univariate trees. A random forest (RF) classifier (Breiman 2001) is used here to partition the feature space. A RF is an ensemble of J decision trees, $\{g_j, j = 1, 2, \ldots, J\}$. Each tree is constructed from a different bootstrap sample of the original data. The instances left out of a bootstrap sample and not used in the construction of a single tree are called out-of-bag (OOB) instances. At each node of each tree, a RF considers the best split based on only a random sample of features. Often, the sample size is $\sqrt{\nu}$, where ν is the number of features. The random selection reduces the variance of the classifier, and also reduces the computational complexity of a single tree from $O(\nu\eta \log \eta)$ to $O(\sqrt{\nu}\eta \log \eta)$ (assuming the depth of tree is $O(\log \eta)$ where η is the number of training (in-bag) instances). Therefore, for a large number of features and instances, a RF can be as computationally efficient as a single decision tree.

The prediction for instance x from tree g_j is $\hat{y}_j(x) = \text{argmax}_c\, p_j^c(x)$, where $p_j^c(x)$ is the proportion of class c in the corresponding leaf of the j-th tree, for $c = 0, 1, \ldots, C - 1$. Let $G(x)$ denote the set of all trees in the RF where instance x is OOB. The OOB class probability estimate of x is

$$p^c(x) = \frac{1}{|G(x)|} \sum_{g_j \in G(x)} I\left(\hat{y}_j(x) = c\right)$$

where $I(\cdot)$ is an indicator function that equals one if its argument is true and zero otherwise. The predicted class is $\hat{y}(x) = \text{argmax}_c\, p^c(x)$. The estimates computed from OOB predictions are easily obtained and have been shown to be good estimates of generalization error (Breiman 2001).

RF provides a number of desirable properties for the time series problem. High-dimensional feature spaces, nominal features, multiple classes, and missing values are handled. Nonlinear models and interactions between features are allowed. It is scale (unit) invariant and robust to outliers, and computations are reasonable even for large datasets.

4 Time series discretization using tree-based classifiers

A MTS, X^n, is an M-attribute time series each of which has T observations where x_m^n is the mth attribute of series n and $x_m^n(t)$ denotes the observation at time t. For

illustration purposes, time series are assumed to be of the same length, T, although our approach can handle time series of different length. MTS example X^n is represented by a $T \times M$ matrix as

$$X^n = \left[\mathbf{x}_1^n, \mathbf{x}_2^n, \ldots, \mathbf{x}_m^n, \ldots, \mathbf{x}_M^n\right] \tag{1}$$

where

$$\mathbf{x}_m^n = \left[x_m^n(1), x_m^n(2), \ldots, x_m^n(T)\right]'$$

is the time series in column m. There are N training MTS, each of which is associated with a class label $y^n \in \{0, 1, 2, \ldots, C-1\}$ for $n = 1, 2, \ldots, N$. Given a set of unlabeled MTS, the task is to map each MTS to one of the predefined classes.

4.1 Representation of multivariate time series

Instead of extracting features from each time series, each row of X^n is considered to be an instance in our approach. This is achieved by creating a matrix of instances $D_{NT\times M}$ where

$$D_{NT\times M} = \begin{bmatrix} \mathbf{x}_1^1\, \mathbf{x}_2^1 \ldots \mathbf{x}_M^1 \\ \mathbf{x}_1^2\, \mathbf{x}_2^2 \ldots \mathbf{x}_M^2 \\ \vdots \\ \mathbf{x}_1^n\, \mathbf{x}_2^n \ldots \mathbf{x}_M^n \\ \vdots \\ \mathbf{x}_1^N\, \mathbf{x}_2^N \ldots \mathbf{x}_M^N \end{bmatrix} \tag{2}$$

Equation 2 is basically the concatenation of training examples X^n in equation 1. We assign the label of each instance to be the same as the time series.

We map $D_{NT\times M}$ to the feature space $\Phi_{NT\times(2M+1)}$ that adds the following columns: time index, first differences for each numerical attribute. The row of Φ for series n at time index t is

$$\left[t, x_1^n(t), x_1^n(t) - x_1^n(t-1), \ldots, x_M^n(t), x_M^n(t) - x_M^n(t-1)\right] \tag{3}$$

The differences provide trend information. A tree learner can capture this information if it relates to the class. This difference is not available for the first observation of MTS, which is assumed to be missing. If an attribute is nominal, first differences are not included. For both numerical and nominal values, missing values are handled by the tree learners. Also, the number of instances may differ across different MTS, but are assumed to be the same for attributes within an MTS. See Table 1 for an example representation.

Table 1 Sample data table for one univariate time series from each of three classes

Series	Time index	Attribute	Difference	Class
1	1	0.5	–	1
1	2	0.5	0	1
1	...	...	...	1
1	19	0.7	0.2	1
2	1	−0.4	–	3
2	...	...	...	3
2	19	0.1	−0.3	3
3	1	0	–	2
3	...	...	...	2
3	19	−0.2	0.1	2

To represent MTS, columns are generated for time index, each attribute, first differences for each numerical attribute, and class

A RF tree learner is trained on Φ assuming that each instance has the same class label as its time series. This is denoted as $RFins$ (for RF applied to the instances) with J_{ins} trees in the forest. Each instance of Φ is mapped to a terminal node of each tree g_j, $j = 1, 2, \ldots, J_{ins}$. Although the trees of $RFins$ without any modification are unpruned, we restrict the number of terminal nodes of each tree to R and that determines the alphabet size in our approach. The trees are trained in a breadth-first order. One level of the tree is built at a time and training stops when there are R terminal nodes. A second parameter is the number of trees J_{ins}. Each tree of $RFins$ provides a symbolic representation for the time series.

Figure 2 illustrates the representation from one tree for three univariate time series. The plot of raw data versus time index is illustrated in Fig. 2a (with the time index plotted on the horizontal axis and the value from a time series plotted on the vertical axis) and we refer to this as signal space. The terminal nodes for the example tree are shown in Fig. 2b to illustrate the partition used for the symbolic representation. These terminal nodes correspond to the partition of signal space shown in Fig. 2a. We omit the first differences in the example so that the partition and symbols can be illustrated on a 2D plot. The symbols are locally sensitive because the time index is a candidate feature to split a node (as illustrated in Fig. 2b). The bar chart of the symbol distribution is shown in Fig. 2c.

A simple way to view a time series is as a set of points in the two-dimensional signal space in Fig. 2a, denoted as S. If there are differences between the classes, there are regions in S where one class of points dominate. When the RF classifier is applied to Φ, boundaries in S are learned to partition points from different classes. Because time is used as a predictor variable, and because RFs can effectively handle interactions, complex regions in S where one class dominates can be detected. In this sense, the time ordering of the data is used. Rather than one tree, the RF ensemble is used to handle structures that are not parallel to the coordinate axes. Each tree of $RFins$ is trained on a random subsample of features and instances. Therefore, the final representation includes different views of the same time series. The representation maps a time series to the high-dimensional space of terminal nodes (that correspond to regions in S where one class dominates). Because trends

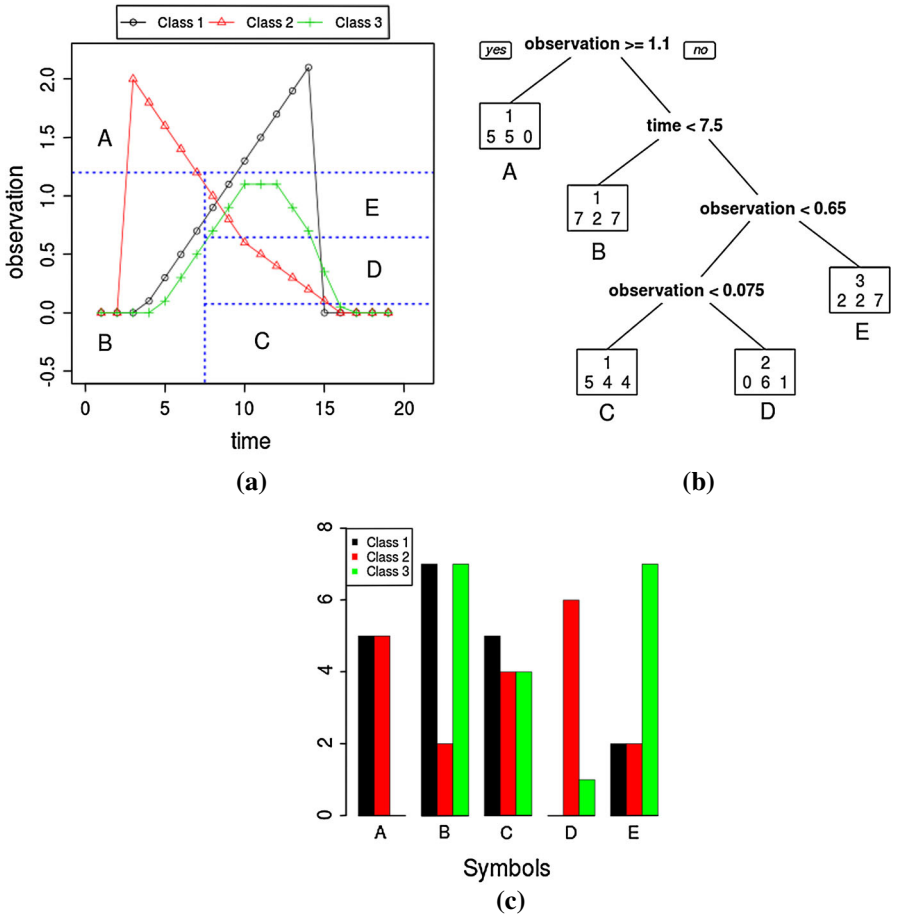

Fig. 2 **a** Time series from three classes are plotted in signal space. **b** A decision tree learned from the time series in signal space partitions the space and the terminal nodes provide symbols. **c** The symbol distribution for each series is shown in a bar chart

are often important properties in time series, the added first differences improve the models.

4.2 Classification

After the symbolic representation is generated from the trees in $RFins$, a bag-of-words (BoW) approach is used to classify the time series. Each symbol is simply considered to be a word and the relative frequency vector of the symbols from each tree are concatenated and used to classify the time series. This frequency vector from each tree is normalized by the number of instances in the time series to obtain the relative frequency vector. More details follow.

Table 2 A visual example of the representation based on symbol frequencies

	Tree 1					Tree 2					. . .	Tree J_{ins}				
	A_1	B_1	C_1	D_1	E_1	A_2	B_2	C_2	D_2	E_2	. . .	.	.	.	.	.
1	0.26	0.37	0.26	0.00	0.11	0.11	0.53	0.21	0.00	0.16	. . .	0.63	0.00	0.11	0.26	0.00
2	0.26	0.11	0.21	0.32	0.11	0.26	0.26	0.00	0.21	0.26	. . .	0.11	0.53	0.21	0.00	0.16
3	0.00	0.37	0.21	0.05	0.37	0.16	0.16	0.21	0.47	0.00	. . .	0.42	0.42	0.00	0.16	0.00
.	.	.	.	.	.	.	.	.	.	.	. . .	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	. . .	.	.	.	.	.
N	0.42	0.11	0.16	0.21	0.11	0.16	0.37	0.21	0.00	0.26	. . .	0.21	0.00	0.21	0.37	0.21

Each column denotes a symbol from a tree of $RFins$ (with $R = 5$ terminal nodes each), and each row denotes a MTS. Each table entry is the proportion of time series indices (for the MTS corresponding to the row) in the terminal node

Let $H_j(X^n)$ be the $R \times 1$ relative frequency vector of the terminal nodes from tree g_j for MTS X^n. Let the function $q_j(\cdot)$ assign a row of Φ to a terminal node of tree g_j. Each entry of $H_j(X^n)$ is the proportion of time indices from MTS n assigned to a terminal node (say r) by tree g_j. That is, the rth element of $H_j(X^n)$ is

$$\frac{\sum_{t=1}^{T} I\left[q_j(\Phi_t^n) = r\right]}{T}$$

for $r = 1, 2, \ldots, R$, where $I(\cdot)$ is the indicator function and Φ_t^n denotes the row t of MTS n in Φ.

We concatenate the $H_j(X^n)$ (relative frequency vectors) from each of the J_{ins} trees g_j of $RFins$ to obtain the final representation of each time series, denoted $H(X^n)$, of length $R \times J_{ins}$. Table 2 illustrates the representation for symbol frequencies with the number of terminal nodes $R = 5$. A classifier is then trained on the $H(X^n)$. The cardinality of $H(X^n)$ might be large based on the setting of R and J_{ins}. Therefore, a scalable classifier that can handle interactions and correlations such as a RF is preferred for this task. This RF is referred as $RFts$ for which we train J_{ts} trees. To classify a test MTS, X^0, the frequency representation $H(X^0)$ is obtained and $RFts$ assigns the class.

5 Experiments and results

Although our focus is on MTS, SMTS is tested on several UTS and MTS datasets. These datasets are from domains such as speech recognition, activity recognition, medicine, etc. The UTS are from Keogh et al. (2011) and provide diverse characteristics such as the lengths of the series, the number of classes, etc. (as shown in the first columns in Tables 4 and 5). A total of 22 MTS from CMU (2012), Bache and Lichman (2013), Keogh et al. (2011) and Olszewski (2012) are used to illustrate the performance

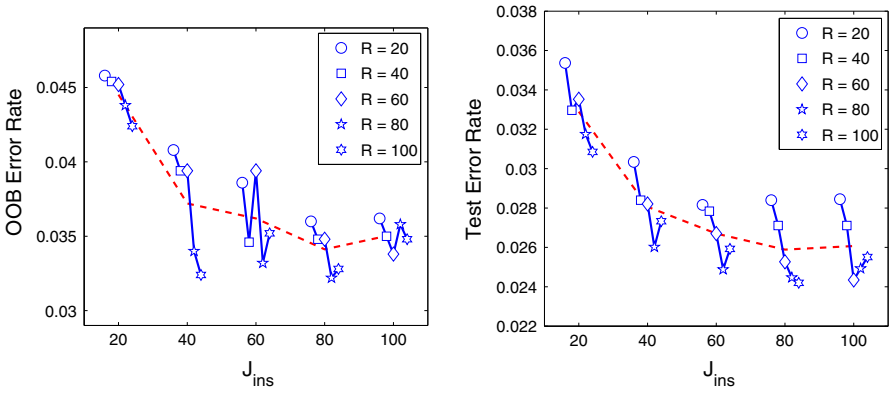

Fig. 3 Average OOB and test error rates with selected R and J_{ins} for *StarLightCurves* dataset (five replications). As representation size ($R \times J_{ins}$) increases, marginal improvement in OOB error rates decreases

of our approach on MTS data. The dataset characteristics are given in Table 6. We randomly selected train and test examples if no default train/test split is provided for the dataset.

5.1 Parameter settings

Our algorithm does not require the setting of many parameters and it is robust to the settings. Three parameters of our approach are R, J_{ins} and J_{ts}. RF is insensitive to both the number of trees and, also the number of candidate attributes scored to potentially split a node (Breiman 2001). The number of features evaluated at each node of a tree is set to the default that equals the approximate square root of the number of features. For $RFins$, there are $2 \times M + 1$ features (assuming that all attributes are numerical). For $RFts$, the number of features is $R \times J_{ins}$. Consequently, the only parameter of importance is the number of terminal nodes (alphabet size) R. Still, we allow for different settings for J_{ins} and J_{ts}.

The parameters R and J_{ins} are set in two steps. We first set R based on a mini experiment, because it determines the level of detail captured. In this experiment, we evaluate all R levels by running $RFins$ with 25 trees. Then $RFts$ with 50 trees is trained on each representation. The R value providing the best OOB error rate based on the training data is chosen. After setting R, we generate a representation for each level of J_{ins} in the second mini experiment. $RFts$ with 50 trees is trained on each representation to select the J_{ins} value to minimize the OOB error rate on the training data. Finally, J_{ts} is set based on the progress of OOB error rates at discrete number of tree levels (step size of 50 trees). If the OOB error rate does not improve more than a tolerance level (set to 0.05 times the OOB error rate from the previous step), we stop adding new trees to RF. We illustrate the behavior of SMTS on the *StarLightCurves* dataset with selected values for the representation size (R and J_{ins} levels) in Fig. 3. Error rates improve as the representation size is increased and stabilize after a certain level. Also, this confirms our strategy to select R first because the error rates are primarily affected by the R setting. If there is no concern about the training

time, J_{ins} can be set large. Based on our preliminary experiments, three values are evaluated for $R \in \{20, 50, 100\}$ and $J_{ins} \in \{20, 50, 100\}$. After the parameters are set from only the training data, the model is used for the classification of the test time series.

5.2 Classification accuracy

5.2.1 Univariate datasets

SMTS is compared to nearest neighbors (NN) classifiers with DTW. Two versions of DTW are considered: NNDTWBest (Ratanamahatana and Keogh 2004) searches for the best warping window, based on the training data, then uses the learned window on the test data, while NNDTWNoWin has no warping window. Note that DTW is a strong solution for time series problems in a variety of domains (Ratanamahatana and Keogh 2005). The results for NNDTW classifiers were obtained from Keogh et al. (2011). We also standardize each time series to zero mean and unit standard deviation. This adjusts for potentially different baselines or scales that are not considered to be relevant (or persistent) for a learner.

An unsupervised BoW approach with a codebook derived from K-means clustering is also considered for comparison. In the unsupervised approach, the Euclidean distance between each row of Φ is computed. K-means clustering with k selected from the set $k \in \{20, 50, 100, 250, 500\}$ is used to label the observations. Then, we use the frequency of the cluster assignments to generate the codebook. Similar to our approach, we train a RF on the codebook. The results are reported for the k level providing the minimum OOB error rate on the training data.

We also compare to a RF classifier applied directly to the times series values. That is, the number of features is T for each time series. This approach requires time series of the same length. For all RFs trained with alternative approaches, OOB error rates at a discrete number of tree levels are considered to set the number of trees. As before, the number of features evaluated at each node of a tree is set to the default which equals the approximate square root of the number of features.

The results for a nearest-neighbor classifier with Euclidean distance on a BoP representation (NNBoP) are from Lin et al. (2012). The results are reported for 20 of the UCR datasets with the best parameter setting combination selected by Lin et al. (2012). Consequently, we compare to NNBoP over these 20 datasets and compare to the other methods over 45 datasets.

We provide the significance levels for Wilcoxon matched-pairs signed-ranks tests for SMTS and the number of wins/ties/losses against the algorithm in Table 3. Tables 4 and 5 provide detailed results from 10 replications of our algorithm on the test data.

The performance of SMTS is better than the BoW approach with K-means clustering on univariate datasets. This shows an advantage of our supervised signal space partition compared to an unsupervised one. Also, SMTS provides better results than RF on the values. This is expected, due to the problems of supervised learners trained

Table 3 Performance of SMTS (average over 10 replications) against competitor approaches, BoW approach with K-means clustering, TSBF (Baydogan et al. 2013), nearest-neighbor classifiers with dynamic time warping distance (NNDTWBest and NNDTWNoWin), RF applied directly to time series values and nearest-neighbor classifier with Euclidean distance on BoP representation (Lin et al. 2012)

		BoW	TSBF	NNDTW		RF	NN
		K-means		Best	NoWin		BoP
20 datasets	win/tie/lose	13/0/7	6/1/13	12/0/8	14/1/5	16/0/4	12/1/7
	Wilcoxon	0.124	0.934	0.131	0.011	0.001	0.102
45 datasets	win/tie/lose	34/2/9	18/1/26	28/0/17	32/1/12	38/0/7	[a]
	Wilcoxon	0.000	0.728	0.038	0.001	0.000	

[a] Reported the error rates over 20 datasets

Table 4 Error rates over 20 datasets, SMTS (average over 10 replications), BoW approach with K-means clustering, TSBF (Baydogan et al. 2013), nearest-neighbor classifiers with dynamic time warping distance (NNDTWBest and NNDTWNoWin), RF applied directly to time series values and nearest-neighbor classifier with Euclidean distance on BoP representation (Lin et al. 2012)

	# Of classes	Dataset size		Length	BoW		TSBF	NNDTW		RF	NN
		Train	Test		SMTS	K-means		Best	NoWin		BoP
50Words	50	450	455	270	0.289	0.308	**0.209**	0.242	0.310	0.348	0.466
Adiac	37	390	391	176	0.248	0.304	**0.245**	0.391	0.396	0.361	0.432
Beef	5	30	30	470	**0.260**	0.333	0.287	0.467	0.500	0.300	0.433
CBF	3	30	900	128	0.020	0.022	0.009	0.004	**0.003**	0.112	0.013
Coffee	2	28	28	286	0.029	0.107	**0.004**	0.179	0.179	0.007	0.036
ECG	2	100	100	96	0.159	0.200	0.145	**0.120**	0.230	0.184	0.150
Face (all)	14	560	1,690	131	0.191	**0.150**	0.234	0.192	0.192	0.190	0.219
Face (four)	4	24	88	350	0.165	0.284	**0.051**	0.114	0.170	0.211	0.023
Fish	7	175	175	463	0.147	**0.051**	0.080	0.160	0.167	0.221	0.074
Gun-Point	2	50	150	150	0.011	**0.007**	0.011	0.087	0.093	0.073	0.027
Lighting-2	2	60	61	637	0.269	0.213	0.257	**0.131**	**0.131**	0.244	0.164
Lighting-7	7	70	73	319	**0.255**	0.315	0.262	0.288	0.274	0.263	0.466
OliveOil	4	30	30	570	0.177	0.100	**0.090**	0.167	0.133	0.107	0.133
OSU L.	6	200	242	427	0.377	0.331	0.329	0.384	0.409	0.518	**0.256**
Swedish L.	15	500	625	128	0.080	0.106	**0.075**	0.157	0.210	0.126	0.198
Synt. Cont.	6	300	300	60	0.025	0.020	0.008	0.017	**0.007**	0.046	0.037
Trace	4	100	100	275	**0.000**	0.030	0.020	0.010	**0.000**	0.165	**0.000**
Two Pat.	4	1,000	4,000	128	0.003	0.011	0.001	0.002	**0.000**	0.158	0.129
Wafer	2	1,000	6,174	152	**0.000**	0.008	0.004	0.005	0.020	0.012	0.003
Yoga	2	300	3,000	426	**0.094**	0.173	0.149	0.155	0.164	0.191	0.170

Bold values indicate the best performance (i.e. minimum error rate)

on the features from fixed locations and dimensions as discussed by many authors (Baydogan et al. 2013; Geurts 2001). Obviously, our symbol generation process combined with our BoW representation helps to avoid this problem. SMTS performs

Table 5 Error rates over 25 datasets, SMTS (average over 10 replications), BoW approach with K-means clustering, TSBF (Baydogan et al. 2013), nearest-neighbor classifiers with dynamic time warping distance (NNDTWBest and NNDTWNoWin), RF applied directly to time series values

	# Of classes	Dataset size		Length	BoW		TSBF	NNDTW		RF
		Train	Test		SMTS	K-means		Best	NoWin	
ChlorineConc.	3	467	3,840	166	0.328	0.328	0.336	0.350	0.352	**0.291**
CinC_ECG_torso	4	40	1,380	1,639	0.100	0.236	0.262	**0.070**	0.349	0.250
Cricket_X	12	390	390	300	0.302	0.359	0.278	0.236	**0.223**	0.427
Cricket_Y	12	390	390	300	0.298	0.297	0.259	**0.197**	0.208	0.396
Cricket_Z	12	390	390	300	0.259	0.321	0.263	**0.180**	0.208	0.406
DiatomSize	4	16	306	345	0.094	0.144	0.126	0.065	**0.033**	0.093
ECG 5 days	2	23	861	136	0.190	0.214	**0.183**	0.203	0.232	0.210
FacesUCR	14	200	2,050	131	0.157	0.204	0.090	**0.088**	0.095	0.215
Haptics	5	155	308	1,092	**0.485**	0.513	0.488	0.588	0.623	0.551
InlineSkate	7	100	550	1,882	**0.562**	**0.562**	0.603	0.613	0.616	0.665
ItalyPowerDemand	2	67	1,029	24	0.038	0.090	0.096	0.045	0.050	**0.033**
MALLAT	8	55	2,345	1,024	0.052	**0.034**	0.037	0.086	0.066	0.082
MedicalImages	10	381	760	99	0.273	0.345	0.269	**0.253**	0.263	0.277
MoteStrain	2	20	1,252	84	**0.051**	0.126	0.135	0.134	0.165	0.119
SonyRobot	2	20	601	70	0.208	0.394	**0.175**	0.305	0.275	0.321
SonyRobotII	2	27	953	65	0.151	0.197	0.196	**0.141**	0.169	0.197
StarLightCurves	3	1,000	8,236	1,024	0.025	0.026	**0.022**	0.095	0.093	0.052
Symbols	6	25	995	398	0.035	0.104	**0.034**	0.062	0.050	0.148
TwoLeadECG	2	23	1,139	82	**0.027**	0.036	0.046	0.132	0.096	0.268
uWaveGesture_X	8	896	3,582	315	0.180	0.217	**0.164**	0.227	0.273	0.245
uWaveGesture_Y	8	896	3,582	315	0.257	0.293	**0.249**	0.301	0.366	0.314
uWaveGesture_Z	8	896	3,582	315	0.242	0.275	**0.217**	0.322	0.342	0.290
WordsSynonyms	25	267	638	270	0.399	0.409	0.302	**0.252**	0.351	0.439
Thorax1	42	1,800	1,965	750	**0.099**	0.126	0.138	0.185	0.209	0.123
Thorax2	42	1,800	1,965	750	**0.071**	0.102	0.130	0.129	0.135	0.090

Bold values indicate the best performance (i.e. minimum error rate)

better for most of the datasets when compared to NN approaches. We are performing slightly better than NNBoP approach over the 20 datasets considered. TSBF outperforms SMTS in most of the UTS datasets, but there is not a significant difference. The good performance of TSBF lies in its own representation, because each time series is represented with multiple subsequences from which features are extracted. This allows for mechanisms to handle the possible problems related to noise, translation and dilation of patterns. We mentioned previously that additional steps would be needed to extend TSBF to multivariate series which are the focus here, and this is discussed in a following section.

Although not shown here, SMTS experiments with and without difference features were also conducted. SMTS with difference features provided better results for 39 datasets of the 45 datasets and this illustrates the benefits of the difference information.

Table 6 Characteristics of MTS: number of classes, number of attributes, length of time series, number of training instances and number of testing instances

	# of Classes	# of Attributes	Length	Dataset size		CV	Source
				Train	Test		
AUSLAN	95	22	45–136	1140	1425	Tenfold	Bache and Lichman (2013)
Pendigits	10	2	8	300	10692		
Japanese Vowels	9	12	7–29	270	370		
Robot Failure							
LP1	4	6	15	38	50	Fivefold	Bache and Lichman (2013)
LP2	5	6	15	17	30		
LP3	4	6	15	17	30		
LP4	3	6	15	42	75		
LP5	5	6	15	64	100		
ECG	2	2	39–152	100	100	Tenfold	Olszewski (2012)
Wafer	2	6	104–198	298	896		
CMU_MOCAP_S16	2	62	127–580	29	29	Tenfold	CMU (2012)
ArabicDigits	10	13	4–93	6600	2200	×	Bache and Lichman (2013)
CharacterTrajectories	20	3	109–205	300	2558	×	
LIBRAS	15	2	45	180	180	×	

Test performance is also reported for all datasets. Column "CV" indicates if comparisons are also based on cross-validation. The source of the datasets are in the last column

5.2.2 Multivariate datasets

The MTS datasets used to evaluate SMTS are commonly used by other MTS classification studies. However, some researchers downsample certain datasets to fewer classes or instances due to the high number of classes [e.g., Weng and Shen (2008) used instances from 25 classes of AUSLAN]. Moreover, some algorithms preprocess the data for different purposes such as smoothing or obtaining an appropriate representation [e.g., Bankó and Abonyi (2012), Weng and Shen (2008) truncated some datasets to obtain time series of same length]. We compare SMTS to the approaches which do not preprocess the data.

Most of the MTS studies follow different strategies for experimentation which complicates the comparisons. For instance, Lin et al. (2012) and Orsenigo and Vercellis (2010) evaluated the performance using cross-validation (CV). To have a fair comparison with the competitor algorithms, we also follow their experimental strategy. The datasets for which CV is performed are given in Table 6. For the CV, we combine the training and test data to obtain a single dataset. We replicate the CV five times and report average error rates.

The motif discovery approach by McGovern et al. (2011) works on binary classification problems and the critical success index (CSI) by Schaefer (1990) is used to assess the performance of the approach. This measure evaluates success as the ratio of the number of true positives to the number of instances that are predicted to be positive. In order to make this approach comparable to SMTS, we modified the proposed approach and use the accuracy as the performance measure. As a binary classifier, we are able to run the approach on only three datasets in Table 6. There are five parameters of the approach: alphabet size $\in \{3, 4, \ldots, 8\}$, word size $\in \{2, 3\}$, the number of time steps averaged into piecewise aggregation $\in \{1, 2, \ldots, 5\}$, minimum probability of detection (POD) as 0.9 and maximum false alarm ratio (FAR) as 0.8. This yields 60 different parameter combinations for the experiment. The error rates are reported for the parameter setting that provides the best accuracy on training data over tenfold CV. This approach required an average of 18 hours per dataset with the same hardware used for SMTS. Computational times are discussed in Sect. 5.3.

We also compare SMTS with 1-NN classifiers with DTW and a multivariate extension of TSBF (MTSBF) (Baydogan et al. 2013) on the test data. For DTW calculations, each time series is standardized to have a mean of zero and a standard deviation of one before distance computations. The DTW distance between two MTS is taken to be the sum of the DTW distances between the associated univariate time series. TSBF generates a representation based on the features extracted from univariate time series in its original implementation. For MTSBF, we generate a representation for each attribute of MTS and then concatenate these representations columnwise to obtain the final representation for MTS. Therefore, this requires running TSBF on each attribute. The only important parameter of TSBF is the subsequence length factor (z in TSBF's notation) which determines the minimum length of the subsequences to be used for feature extraction. The other parameters are kept as recommended by Baydogan et al. (2013). Evaluated z values are $z \in \{0.25, 0.5, 0.75\}$. For each univariate series, this parameter is selected based on the procedure described by Baydogan et al. (2013). The codes for MTSBF are available on Baydogan (2013). MTSBF generates a representation for each attribute of MTS which requires running TSBF M times. This may not be adequate for MTS with a large number of attributes.

Table 7 summarizes the results from our CV experiments and reported error rates from other references. SMTS performs consistently better when compared to the classification approaches considered by Orsenigo and Vercellis (2010). The best error rate for *AUSLAN* reported by Kadous and Sammut (2005) is from an ensemble of 11 different classifiers trained on the extracted metafeatures. SMTS performs equally well for this particular dataset. Also, Lin et al. (2012) reported error rates for two versions of NN classifiers with DTW: NNDTW-Best and NNDTW-NoWin. SMTS outperforms the similarity-based approaches for these two datasets.

For binary classification problems, SMTS outperforms the predictive motif discovery method from McGovern et al. (2011) with an approach that is conceptually and operationally quite simple. Our experiments show that the motif discovery approach is sensitive to the setting of the parameters and has problems with overfitting. The parameters are set based on the suggestions and experiments in McGovern et al. (2011). It has the potential to perform better with different settings and how parameters should set to avoid overfitting is further discussed by McGovern et al. (2011).

Table 7 CV error rates for SMTS (average over five replications)

	SMTS	Orsenigo and Vercellis (2010)			NNDTW (k=3) Lin et al. (2012)		Tclass Kadous and Sammut (2005)	McGovern et al. (2011)
		TDVM	SVM_{DTW}	$1NN_{WD}$	NoWin	BestWin	Voting	Motif
Japanese Vowels	0.035	0.034	0.054	0.077				
Pendigits	0.010	0.037	0.066	0.055				
Robot failure								
LP1	0.062	0.148	0.182	0.182				
LP2	0.384	0.362	0.362	0.404				
LP3	0.189	0.319	0.342	0.383				
LP4	0.063	0.145	0.128	0.137				
LP5	0.261	0.329	0.379	0.348				
ECG	0.134				0.189	0.172		0.241
Wafer	0.010				0.091	0.066		0.130
AUSLAN	0.027						0.021	
CMU_MOCAP_S16	0.007							0.480

Best CV error rates reported by other MTS classification papers

Table 8 Test error rates for SMTS (average over 10 replications), nearest-neighbor classifier with dynamic time warping distance, and a multivariate version of a bag of features method

	SMTS	NNDTW NoWin	MTSBF	Other
CMU_MOCAP_S16	0.003	0.069	0.003	
AUSLAN	0.053	0.238	0.000	
ArabicDigits	0.036	0.092	–	0.069 by Hammami and Bedda (2010)
Japanese Vowels	0.031	0.351	–	0.032 by Bicego et al. (2009), Geurts (2001), 0.059 by Kudo et al. (1999)
Robot Failure				
LP1	0.144	0.280	–	
LP2	0.240	0.467	–	
LP3	0.240	0.500	–	
LP4	0.105	0.187	–	
LP5	0.349	0.480	–	
Wafer	0.035	0.023	0.015	
CharacterTrajectories	0.008	0.040	0.033	
uWaveGestureLibrary	0.059	0.071	0.101	
LIBRAS	0.091	0.200	0.183	
ECG	0.182	0.150	0.165	
Pendigits	0.083	0.088	–	

Best error rates reported by other MTS classification papers

The error rates on the test data are given in Table 8. The datasets are sorted (descending) based on the number of attributes to illustrate the effectiveness of SMTS. SMTS provides better results for the datasets with larger number of attributes, while the performance is comparable for the remaining datasets when compared to similarity-based approaches.

SMTS performs better than MTSBF for some datasets where MTSBF significantly outperforms for the others. TSBF generates a representation for each attribute without taking the others into consideration. However, as discussed in Sect. 2, this approach is greedy because the relationships between the attributes may describe the class. Depending on the problem characteristics, individual handling of the attributes may provide inferior results. For example, certain classes from *uWaveGestureLibrary* are defined by the patterns over multiple time series as shown by Baydogan (2012) and MTSBF provides worse results for this particular dataset. SMTS performs equally well on the *ArabicDigits* and *Japanese Vowels* dataset when compared the other studies in the literature.

5.3 Computational time analysis

Here we empirically evaluate the runtime of SMTS with different settings of problem characteristics and parameters. SMTS is implemented in both C and R Software and our experiments use an Ubuntu 12.04 system with 16 GB RAM, quad core CPU

(i7-3620M 2.7 GHz). Although the CPU can handle eight threads in parallel, only two threads are used. The parallel implementation of SMTS is also available from Baydogan (2013).

The overall computational complexity of SMTS is mainly determined by $RFins$ and $RFts$. The time complexity of building a single tree in RF is $O(\sqrt{\nu}\eta\beta)$, where ν is the number of features, η is the number of instances, and β is the depth of the tree. For $RFins$, $\nu = 2M+1$ is the number of features, $\eta = (N \times T)$ is the number of training instances and $\beta = R-1$ is the depth of the tree in the worst case assuming that the depth takes the largest possible value. Thus, complexity is $O[J_{ins}\sqrt{2M+1}(NT)(R-1)]$ in the worst case. For $RFts$, $\eta = N$, $\nu = R \times J_{ins}$ and $\beta = \log N$ (assuming the depth of tree is $O(\log N)$) which is $O(J_{ts}\sqrt{R \times J_{ins}}N \log N)$.

The *StarLightCurves* dataset from Keogh et al. (2011) is used to demonstrate the effect of the parameters J_{ins}, N, T and R on the computation times. For the multivariate case, SMTS's performance as a function of the number of attributes M, is illustrated on the *AUSLAN* dataset from Bache and Lichman (2013). For each dataset, we randomly selected $\delta \in \{0.2, 0.4, \ldots, 1\}$ proportion of the number of instances (δ_N), number of observations (δ_T) and number of attributes (δ_M). The levels considered for R and J_{ins} are $R \in \{20, 40, 60, 80, 100\}$, $J_{ins} \in \{20, 40, 60, 80, 100\}$. Here 10 replications are conducted for each setting combination.

We first illustrate the computation time required to generate symbols. Figure 4a, b schematize the average symbol generation time across a variety of settings. The computation times with changing N and T are analyzed for fixed values of $R = 100$ and $J_{ins} = 100$. If both N and T increase, the symbol generation time is expected to increase quadratically (because the number of instances for $RFins$ is $N \times T$). However, the symbol generation works faster in practice. This is due to the nature of the time series. The time to sort all observations in each dimension prior to building the tree does not increase drastically. Since the values are in some fixed range, a faster computation time is achieved by avoiding the comparisons during the sorting process. We also analyzed the symbol generation times with selected values of R and J_{ins} with fixed values of $\delta_N = 1$ and $\delta_T = 1$. From the computation times in Fig. 4b, the practical complexity of symbol generation is approximately linear on both R and J_{ins} settings.

The complexity of training is also evaluated with changing R and J_{ins} where $\delta = 1$ for the remaining parameters. Here R and J_{ins} determine the representation size. Figure 4c illustrates the average training time with these parameters. The time for training increases linearly with the increase in R and J_{ins} which is consistent with the complexities of $RFins$ and $RFts$. Furthermore, the linear behavior of the training time with the increase in the representation size (i.e. both R and J_{ins}) is an advantage of the proposed approach. This behavior is due to the selection of the square root of the number of features to evaluate at each split node by $RFts$.

SMTS symbol generation times for selected numbers of attributes are illustrated in Fig. 4d when other parameters are fixed. The runtime increase with the number of attributes is consistent with $O(\sqrt{2M+1})$ from $RFins$. The discrete levels of M evaluated are $M \in \{4, 8, 13, 17, 22\}$ which explains the near-linear behavior of the symbol generation times.

The median training time is 16.4 s per dataset with a minimum of 0.8 s and a maximum of 1381.9 s (for *SonyRobot* and *Thorax2*, respectively) for univariate datasets.

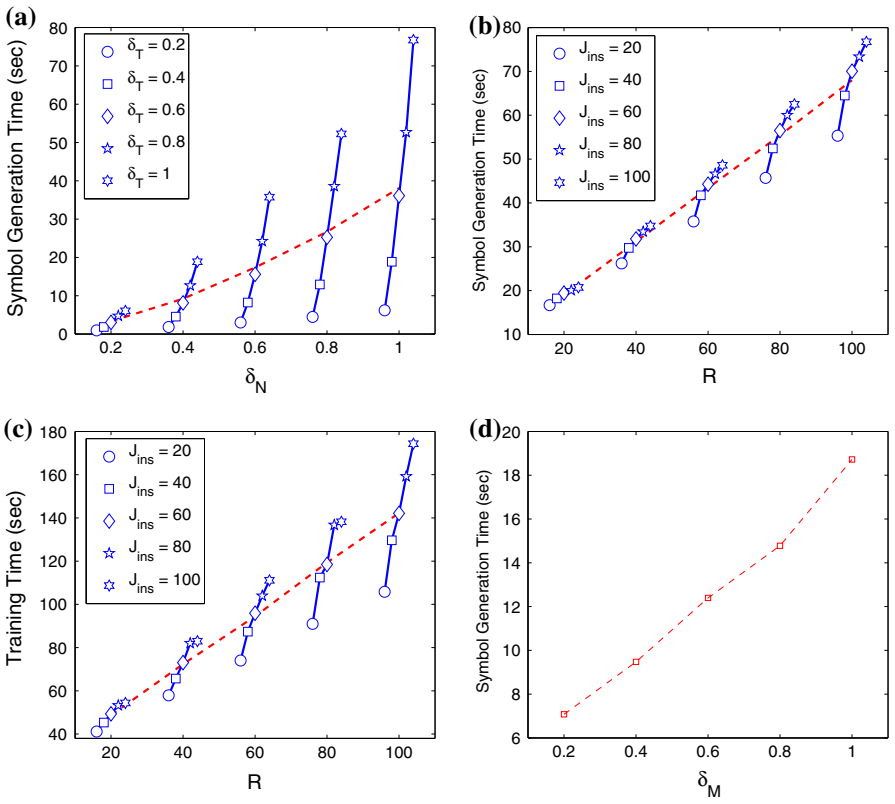

Fig. 4 Average symbol generation times with changing N and T ($R = 100$, $J_{ins} = 100$) and R and J_{ins} ($\delta_N = 1$, $\delta_T = 1$) (*top*) and with changing M ($R = 100$, $J_{ins} = 100$, $\delta_N = 1$, $\delta_T = 1$) (*bottom left*). Overall training time with changing R and J_{ins} ($\delta_N = 1$, $\delta_T = 1$) (*bottom right*)

For multivariate datasets, the median training time is 9.8 s per dataset with a minimum of 0.6 s and a maximum of 2025.7 seconds (for *LP2* and *ArabicDigits*, respectively). We did not evaluate the time for testing since our approach is very fast in classification of the series (takes less than a millisecond). It only requires the traversal of the trees from $RFins$ and $RFts$ after feature representation. SMTS is very fast and convenient for real-time classification of time series data.

5.4 Missing values

For time series with missing values, missing values are usually interpolated. However, the estimation method itself adds an additional parameter to the time series problem. Our proposed approach naturally handles the data with missing values, without any additional steps, because tree-based learning implicitly handles the attributes with missing values (Breiman et al. 1984).

CBF, GunPoint, MoteStrain, Symbols and ECG 5 days datasets are selected to illustrate the performance of SMTS when there are missing values. We randomly

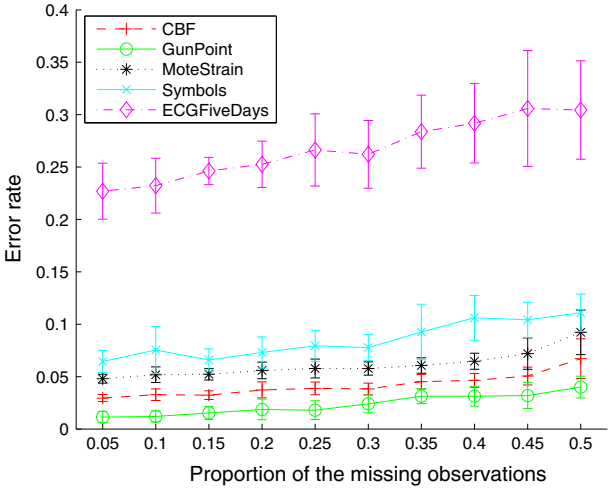

Fig. 5 The test error rates with different proportions of missing values in the training observations for five univariate datasets from Keogh et al. (2011). SMTS does not require a specific mechanism to handle the missing values and it is insensitive to large proportions of missing values

removed $\delta \in \{0.05, 0.1, \ldots, 0.5\}$ proportion of the values for each time series. The error rates over 10 replications are provided in Fig. 5. For these datasets, SMTS performs reasonably well even with large proportions of missing values.

6 Conclusions

The representation of the complex data in MTS is an important challenge for many methods. The SMTS here does not require pre-defined time intervals or features. Instead, the symbolic representation is learned, and all attributes of MTS are considered simultaneously during a supervised process. Thus, relationships between the individual attributes are taken into account. Furthermore, the initial representation of raw data (and first differences) is quite simple conceptually and operationally. Still, a RF can detect interactions in the space S of time index and time value and this is exploited to generate a codebook. The codebook is processed with a second RF where now the implicit feature selection is exploited to handle the high-dimensional input. The constituent properties yield an approach quite different from current methods. Moreover, MTS with nominal and missing values are handled efficiently with tree learners.

Ensemble learners that scale well with large number of attributes and long time series make SMTS computationally efficient. Our experiments demonstrate the effectiveness of the proposed approach in terms of accuracy and computation times for MTS, and that the approach is approximately equal to the best alternatives for univariate time series. Although not explored here, the proposed representation can be used for similarity analysis, and tasks such as clustering.

Acknowledgments This research was partially supported by ONR Grant N00014-09-1-0656.

References

Akl A, Valaee S (2010) Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, compressive sensing. In Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on, pp 2270–2273, March

Bache K, Lichman M (2013) UCI machine learning repository. University of California, School of Information and Computer Science, Irvine, CA. http://archive.ics.uci.edu/ml

Bankó Z, Abonyi J (2012) Correlation based dynamic time warping of multivariate time series. Expert Systems with Applications 18(5):231–241

Baydogan MG (2012) Modeling time series data for supervised learning. PhD thesis, Arizona State University, Dec.

Baydogan MG (2013) Multivariate time series classification. homepage: www.mustafabaydogan.com/multivariate-time-series-discretization-for-classification.html

Baydogan MG, Runger G, Tuv E (2013) A bag-of-features framework to classify time series. Pattern Analysis and Machine Intelligence, IEEE Transactions on 35(11):2796–2802

Bicego M, Pekalska E, Tax DMJ, Duin RPW (2009) Component-based discriminative classification for hidden Markov models. Pattern Recognition 42(11):2637–2648

Breiman L (2001) Random forests. Machine Learning 45(1):5–32

Breiman L, Friedman J, Olshen R, Stone C (1984) Classification and Regression Trees. Wadsworth, Belmont, MA

Brodley C, Utgoff P (1995) Multivariate decision trees. Machine Learning 19(1):45–77

Chakrabarti K, Keogh E, Mehrotra S, Pazzani M (2002) Locally adaptive dimensionality reduction for indexing large time series databases. ACM Trans. Database Syst. 27(2):188–228

Chaovalitwongse W, Pardalos P (2008) On the time series support vector machine using dynamic time warping kernel for brain activity classification. Cybernetics and Systems Analysis 44:125–138

Fu T-C (2011) A review on time series data mining. Engineering Applications of Artificial Intelligence 24:164–181

Geurts P (2001) Pattern extraction for time series classification. Principles of Data Mining and Knowledge Discovery, volume 2168 of Lecture Notes in Computer ScienceSpringer, Berlin / Heidelberg, pp 115–127

Hammami N, Bedda M (2010) Improved tree model for arabic speech recognition. In Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, volume 5, pages 521–526, July

Kadous MW, Sammut C (2005) Classification of multivariate time series and structured data using constructive induction. Machine Learning 58:179–216

Keogh E, Zhu Q, Hu B, Y. H, Xi X, Wei L, Ratanamahatana CA (2011) The UCR time series classification/clustering. homepage: www.cs.ucr.edu/~eamonn/time_series_data/

Kudo M, Toyama J, Shimbo M (1999) Multidimensional curve classification using passing-through regions. Pattern Recognition Letters 20(1113):1103–1111

Kuksa PP (2012) 2d similarity kernels for biological sequence classification. In ACM SIGKDD Workshop on Data Mining in Bioinformatics

Li C, Khan L, Prabhakaran B (2006) Real-time classification of variable length multi-attribute motions. Knowledge and Information Systems 10:163–183

Li C, Khan L, Prabhakaran B (2007) Feature selection for classification of variable length multiattribute motions. In Multimedia Data Mining and Knowledge Discovery, pages 116–137. Springer London

Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series, with implications for streaming algorithms. In In Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp 2–11. ACM Press

Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing SAX: a novel symbolic representation of time series. Data Mining and Knowledge Discovery 15:107–144

Lin J, Khade R, Li Y (2012) Rotation-invariant similarity in time series using bag-of-patterns representation. Journal of Intelligent Information Systems, pp 1–29

Lin J, Williamson S, Borne K, DeBarr D (2012) Pattern recognition in time series. In Advances in Machine Learning and Data Mining for Astronomy, Chapman & Hall, To appear.

Liu J, Wang Z, Zhong L, Wickramasuriya J, Vasudevan V (2009) uWave: Accelerometer-based personalized gesture recognition and its applications. Pervasive Computing and Communications, IEEE International Conference on 0:1–9

McGovern A, Rosendahl D, Brown R, Droegemeier K (2011) Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction. Data Mining and Knowledge Discovery 22:232–258

Moosmann F, Nowak E, Jurie F (2008) Randomized clustering forests for image classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 30:1632–1646

Olszewski RT (2012) http://www.cs.cmu.edu/~bobski/. accessed: June 10

Ordonez P, Armstrong T, Oates T, Fackler J (2011) Using modified multivariate bag-of-words models to classify physiological data. In Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops, ICDMW '11, pages 534–539, Washington, DC, USA, IEEE Computer Society.

Orsenigo C, Vercellis C (2010) Combining discrete svm and fixed cardinality warping distances for multivariate time series classification. Pattern Recognition 43(11):3787–3794

Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann,

Ratanamahatana C, Keogh E (2004) Making time-series classification more accurate using learned constraints. In Proceedings of SIAM International Conference on Data Mining (SDM04), pp 11–22

Ratanamahatana C, Keogh E (2005) Three myths about dynamic time warping data mining. In Proceedings of SIAM International Conference on Data Mining (SDM05), volume 21, pp 506–510

Sakoe H (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing 26:43–49

Schaefer JT (1990) The Critical Success Index as an Indicator of Warning Skill. Weather and Forecasting 5(4):570–575

Shieh J, Keogh E (2008) isax: indexing and mining terabyte sized time series. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '08, pages 623–631, New York, NY, USA, ACM.

CMU Graphics Lab Motion Capture Database. Homepage: mocap.cs.cmu.edu, 2012

Weng X, Shen J (2008) Classification of multivariate time series using locality preserving projections. Knowledge-Based Systems 21(7):581–587