Modeling Time Series Data for Supervised Learning

by

Mustafa Gokce Baydogan

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved August 2012 by the
Graduate Supervisory Committee:

George C. Runger, Chair
Robert Atkinson
Esma Gel
Rong Pan

ARIZONA STATE UNIVERSITY

December 2012

ABSTRACT

Temporal data are increasingly prevalent and important in analytics. Time series (TS) data are chronological sequences of observations and an important class of temporal data. Fields such as medicine, finance, learning science and multimedia naturally generate TS data. Each series provide a high-dimensional data vector that challenges the learning of the relevant patterns

This dissertation proposes TS representations and methods for supervised TS analysis. The approaches combine new representations that handle translations and dilations of patterns with bag-of-features strategies and tree-based ensemble learning. This provides flexibility in handling time-warped patterns in a computationally efficient way. The ensemble learners provide a classification framework that can handle high-dimensional feature spaces, multiple classes and interaction between features. The proposed representations are useful for classification and interpretation of the TS data of varying complexity.

The first contribution handles the problem of time warping with a feature-based approach. An interval selection and local feature extraction strategy is proposed to learn a bag-of-features representation. This is distinctly different from common similarity-based time warping. This allows for additional features (such as pattern location) to be easily integrated into the models. The learners have the capability to account for the temporal information through the recursive partitioning method.

The second contribution focuses on the comprehensibility of the models. A new representation is integrated with local feature importance measures from tree-based ensembles, to diagnose and interpret time intervals that are important to the model.

Multivariate time series (MTS) are especially challenging because the input consists of a collection of TS and both features within TS and interactions between TS can be important to models. Another contribution uses a different representation to produce computationally efficient strategies

that learn a symbolic representation for MTS. Relationships between the multiple TS, nominal and missing values are handled with tree-based learners.

Applications such as speech recognition, medical diagnosis and gesture recognition are used to illustrate the methods. Experimental results show that the TS representations and methods provide better results than competitive methods on a comprehensive collection of benchmark datasets. Moreover, the proposed approaches naturally provide solutions to similarity analysis, predictive pattern discovery and feature selection.

To my family

ACKNOWLEDGMENTS

I want to express my deep and sincere gratitude to my advisor, Dr. George Runger, for his encouragement and generous support throughout my study at ASU. He offered me great research opportunities, resources, and trust which allowed me to fully explore the research area. Without his brilliant guidance, this dissertation would not have been possible.

I would also like to convey thanks to my dissertation committee: Dr. Robert Atkinson, Dr. Esma Gel, and Dr. Rong Pan for their valuable comments on this dissertation. I also would like to extend my gratitude to my industry collaborators: Dr. Eugene Tuv, and Dr. Ben Nelson.

My family supported me in every step I have taken since the very beginning of my graduate studies. I am mostly grateful to my father Mehmet Baydogan, my mother Perihan Baydogan, my sister Banu Gokcen Baydogan for their love and support. Without them, this work could not have been completed.

I am deeply indebted to my friends, Baykal Hafizoglu, Nedim Yel, Muhsin Menekse, Ahmet Cemal Durgun, Kerem Demirtas, Aysegul Demirtas and Tulin Inkaya, for their help, stimulating suggestions and encouragement. My special thanks go to my friends Mustafa Yuksel and Caglar Ata for their support through the course of this dissertation. Thanks for giving me a shoulder to lean on whenever I need.

Finally, my recent happiest moments were all with you, Didem Yamak. Thank you for your love, trust, and understanding. Thank you for providing me the continued moral support and encouragement to pursue my dreams. I am grateful to our journey so far and I am excited about our adventure ahead.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER 1

INTRODUCTION

In the last decade, the increasing use of temporal data, especially time series data, has initiated a great deal of research and development attempts in the field of data mining. Time series data which is chronological sequences of observations is one of the important class of temporal data. Many data sources in different fields, such as in medicine, finance, multimedia and learning sciences naturally generate time series data. For example, an ElectroCardioGram (ECG) is used to identify temporal patterns in heart signals to identify abnormal heart rhythms [6]. Average electrical voltage produced by the beating of the hard muscle is measured over the human body. An ECG is visualized as a 2D plot, where x axis is the time and y axis is the average voltage measured by the electrodes. In the field of seismology, seismograms are used to identify seismic events. A seismogram is a record of the ground motion produced by an earthquake, explosion, or other ground-motion sources [7]. The ground motion is identified by a seismograph at a measuring station as a function of time. Nowadays, Electroencephalography (EEG) which is the recording of electrical activity along the scalp is used to understand the brain activity and connectivity under different experimental conditions. EEG visualizes the voltage fluctuations resulting from ionic current flows within the neurons of the brain over the time.

Time series data is characterized by its numerical and continuous nature [8]. Time series are considered as a whole instead of individual numerical fields because of the temporal ordering in the data. This makes time series analysis different from other data analysis problems, in which there is no natural ordering of the observations. Moreover, another problem is that each series provide a high-dimensional data vector that challenges the analysis. The high-dimensionality can be handled by dimensionality reduction techniques such as feature selection when the temporal ordering is not important. However, entire series should

be considered as a vector in time series analysis problems since the relations between the certain time points may be of interest. Therefore, traditional dimensionality reduction techniques may not work well for the time series data. Real-world time series data is often high-dimensional, contains nonlinear relationships between its variates, and has long-range dependencies. Due to these complexities, time series data mining has received great interest over the past decade.

Time series data mining approaches focus on various problems. The major tasks considered in this context are pattern discovery and clustering, classification, rule discovery and summarization [8]. Although these tasks are presented separately, they are not independent. For instance, clustering result on time series may be useful to a classification task. Therefore, a study on one particular task may provide solutions to other tasks.

A fundamental problem in time series data mining approaches is how to represent the time series data. The representation is important to discover the useful information from the high-dimensional data efficiently rather than analyzing or finding statistical properties on the whole series. High-level representation of the original raw data is generally used as a feature extraction step, or simply to make the storage, transmission, and computation of massive dataset feasible in these approaches [9]. The time series representation strategies are categorized into two classes [9]: data adaptive (adaptive basis representation) and nondata adaptive (fixed basis representation). Examples of data adaptive approaches are Singular Value Decomposition (SVD) [10], Piecewise Linear and Piecewise Constant models (PAA) [11] and Symbolic Aggregate Approximation (SAX) [12]. Nondata adaptive approaches represent the time series in the transformation domain using mostly Discrete

Fourier Transform (DFT) [13] and Discrete Wavelet Transform (DWT) [14]. This thesis explores new adaptive basis representations for time series classification.

Time series classification is a supervised learning problem in which the input consists of a set of training examples and associated class labels, where each example is formed by one or more time series (variables) and the aim is to label test examples to predefined classes. Time series classification is an important task with many challenging applications including finance, science, natural language processing and medicine. For example, a cardiologist might be interested in analysis of ECG signals from different patients in order to see whether a particular patients, e.g., patients with a history of some disease, have different temporal s in their heart signals than a control group [6]. Seismologist aim at discriminating the nature of the seismic waves to classify events such as earthquakes, mining explosions or nuclear explosions [7]. Moreover, EEG records are used in a learning environment to understand the perceived difficulty by classifying the EEG signals based on the puzzle difficulty. Effective and efficient data mining methods are required for the knowledge extraction in such applications.

The algorithms proposed for time series classification can be divided into instance-based and feature-based methods in general. Instance-based classifiers predict a test instance based on its similarity to the training instances. For example, nearest neighbor (NN) classifiers classify objects based on the closest training examples in the feature space and one-nearest-neighbor classifiers with Euclidean (NNEuclidean) or a dynamic time warping distance (NNDTW) have been widely, and successfully used [15–19] in time series classification.

One-nearest-neighbor (NN) classifiers with Euclidean distance do not work well if the patterns of interest translate or dilate over time. DTW [20] is a method that allows a measure of the similarity of time series independent of certain non-linear variations in the time dimension. The idea of DTW is illustrated in Figure 1. Euclidean distance is computed by matching the observation at the same time points. Conversely, DTW aligns the observations using a dynamic programming approach that maximizes the similarity of the time series while satisfying the time ordering of the observations. Therefore, DTW recognizes the similarity of the time series better than the Euclidean distance.

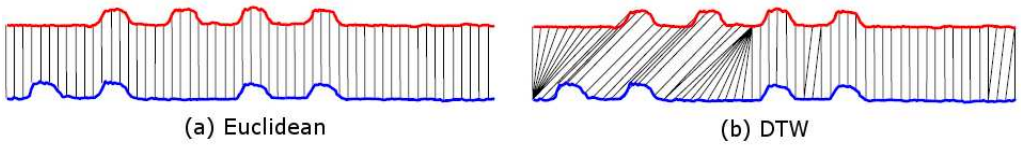


Figure 1. Euclidean and Dynamic Time Warping distance computation [1]. The grey lines indicate that distance is computed over the observations at either end of the line. Alignment of two time series by DTW recognizes the similarity of the series better than the Euclidean Distance

The majority of the NN classifiers works on the raw (observed) data. On the other hand, there are studies based on alternative time series representations. These studies search for similarity on features instead of the raw data. For example, Symbolic Aggregate Approximation (SAX) [12] basically represents the time series based on the mean level of the intervals extracted from the time series. An NN classifier based on this representation searches for similarity on the mean feature of the intervals. We consider the most accurate NN classifiers based on the raw data in this thesis.

NN classifiers with appropriate distance measures are known to provide strong and robust solutions [21, 22] although their space and time requirements may be problematic for

some application. NN classifiers are easy to understand and do not require the setting of many parameters, but they typically do not provide insight into time series features important to the classifier. Why a particular instance is assigned to a certain class is not clear.

Feature-based classifiers work on the features of the time series to reduce the dimensionality. They are interpretable and generally faster than instance-based classifiers depending on the feature extraction method and classification algorithm. The feature extraction step should handle the temporal information relevant to classification and a classifier that can take the temporal relations into account is required. Two types of features are generated in these approaches, global and local features. Global features are extracted from each time series and provide a compact representation of the time series (such as the mean of all observed values) but they are usually insufficient to represent time series information useful to classifiers. On the other hand, local features are extracted from segments of the time series and require such segments to be determined. Since the set of local features may vary in cardinality and lack a meaningful ordering, many classification algorithms requiring feature vectors of fixed dimension have problems in handling the local feature set.

In this thesis, we explore the problems related to time series classification. We propose time series representations that overcome some limitations of existing approaches for classifying the time series. In particular, we consider the following questions in details:

- Long time series with time warped patterns, relatively short features of interest, and moderate noise, are difficult to identify. What are the benefits of the feature-based approaches in such cases? Are there methods that can handle time warping with all the benefits of a feature-based approach?

- Why is a time series assigned to a certain class? Are there patterns specific to certain classes? Which patterns are relevant to the classification task?

- There might be more than one time series relevant to the classification task and multiple series challenge the similarity-based approaches. Scalability of the approaches become important as the number of time series increases. Also, both features within the time series and interactions between the time series can be important to models. Are there computationally efficient strategies to learn both relations simultaneously for time series classification?

1. A Bag-of-Features framework to classify time series

A framework based on the bag-of-features (BoF) representation is proposed to benefit from the speed and other advantages of feature-based methods to handle the problems for which NN classifiers with DTW distance are challenged. A BoF representation characterizes complex objects by feature vectors of sub-objects. We propose interval selection and local feature extraction strategies to explore time series representation that can handle translation and dilations based on the BoF idea.

To capture local information, random subsequences are extracted from each time series and further divided into intervals. The subsequences vary randomly in length and location. The number of intervals that partition a subsequence are fixed so that the interval length varies with the subsequence length. Several features (such as the mean, standard deviation, etc.) are extracted from each interval and these features comprise a row in a new data matrix X (one row for each subsequence). Because the subsequences selected vary in length and location, a particular column in X consists of features from different time locations computed over different length intervals. Consequently, the similarity between time series

can be captured independent of certain non-linear variations in the time dimension. This representation captures information in a manner similar to DTW, but from a very different construction. After representing the features of the subsequences in data matrix X, a classifier is trained assuming that each subsequence has the label of the time series from which it is extracted. Classification results on the subsequences are summarized to obtain the new representation for the time series. This data structure along with a tree-based ensemble allows for relevant features to be used by the classifier while irrelevant one tend to be ignored.

Our local feature generation scheme allows for a novel representation that captures information in a manner similar to DTW, we then label the subsequences and use a supervised approach to summarize the local information unlike the existing studies. Our supervised approach allows for desirable properties for time series classification problem. It provides fast and efficient time series representation for classification even with very basic features such as slope, mean and variance from the subsequences. Global features (e.g autocorrelation of the time series) can also be extracted from the time series and combined with other features. Finally time series may be classified via any supervised learner. We denote the new algorithm as BoF framework to classify Time Series (TSBF).

In Chapter 3, we will address time series classification problem based on bag-of-features representation. We show how TSBF handles the temporal data and demonstrate its efficiency and accuracy by comparing to alternative time series classifiers on a full set of benchmark data sets.

2. Supervised time series pattern discovery through local importance

In Chapter 4, we consider a framework for finding important patterns of time series for classification. We focus on finding the segments of the time series that have potential to distinguish the classes. These segments are referred as the regions of interest. Regions of interests are very important to understand the temporal relations. Moreover, they help to reduce the effort in searching for the time segments useful to a classifier. After finding the region of interests for each time series, we generate sequences from these regions. These sequences are referred as patterns. We generate multiple patterns from the time series and find the best matching segments of the time series to these patterns. Then each time series is represented by the distances of the patterns to the best matching segments of the time series. Another classifier is then trained on this representation. A feature selection algorithm on the new feature set allows for finding the patterns that are critical in classification.

A feature-based algorithm is used to reduce the effort to prune the search space of the regions of interest in our algorithm. [23] also discusses the necessity of pruning the search space to find the regions relevant to classification and proposes a distance-based method. Feature-based approaches allow for some desirable properties such as handling the interactions and fast computation. Interaction between the features in this context is the relationship of the patterns over multiple intervals that may define a class as discussed by [23].

In Chapter 4, we will describe how the interpretability is achieved through the pattern discovery process. We illustrate the compactness of the new representation which reduces the time and space required for classification.

3. Multivariate time series classification with learned discretization

Chapter 5 proposes a time series representation for classification of the multivariate time series (MTS). In the multivariate scenario, there are multiple variables, each in a time series, related to the classification task. This problem has been studied in different fields such as statistics, signal processing and control theory [24]. The most common approach is to obtain a rectangular representation of MTS by transforming the set of multivariate input sequences to a fixed number of columns using different rectangularization approaches [25]. For example, singular value decomposition (SVD) is used by [26–28]. Principal component analysis (PCA) is used for both feature selection and transformation by [29]. Any supervised learner can be trained on the transformed data for classification. Most of these approaches assume that the variables are numerical; however, certain variables of the series can be nominal or missing.

Another strategy is to modify the similarity-based approaches which are used for univariate time series. However, MTS are not only described by the variables, but also by relationships between the variables [30]. This potentially valuable information is lost if only the similarity between the individual variables are taken into consideration [28]. Moreover, as in telecommunication application [25], observations can be nominal (i.e., call type) for which similarity computation is not well-defined.

We follow a different approach and propose a symbolic representation of MTS that is then integrated to produce a new type of MTS classifier. Rather than select intervals from the time series and extract features, the observations in the time series are recursively partitioned into terminal nodes of trees. This leads to a new symbolic representation that is learned based on the class labels. Furthermore, all time series, along with their relation-

ships, are considered simultaneously as the nodes are constructed. Ensembles repeat the process to strengthen the algorithm. This unique representation is then summarized in a high-dimensional codebook. However, another ensemble handles the high dimensionality to generate an effective classifier. there is only one sequence of symbols regardless of the number of variables in a MTS

The relationships between the variables, nominal and missing values are handled efficiently with tree-based learning. There is only one sequence of symbols regardless of the number of variables in a MTS which makes our method computationally efficient when compared to similarity-based methods. Our approach can handle MTS examples with different length and it does not require a special rectangularization mechanism since the final representation is simply obtained by the frequency of the symbols over the time series.

Chapter 5 introduces a novel representation for multivariate time series. We show how the new representation leads to a locality sensitive, scalable and accurate time series classifier.

4. Contributions

This dissertation proposes time series representations and methods for classification The approaches combine new representations that handle translations and dilations of patterns with bag-of-features strategies and tree-based ensemble learning. This provides flexibility in handling time-warped patterns in a computationally efficient way. The ensemble learners provide a classification framework that can handle high-dimensional feature spaces, multiple classes and interaction between features. The proposed representations are useful for classification and interpretation of the time series data of varying complexity.

The first contribution handles the problem of time warping with a feature-based approach. An interval selection and local feature extraction strategy is proposed to learn a bag-of-features representation. This is distinctly different from common similarity-based time warping. This allows for additional features (such as pattern location) to be easily integrated into the models. The learners have the capability to account for the temporal information through the recursive partitioning method.

The second contribution focuses on the comprehensibility of the models. A new representation is integrated with local feature importance measures from tree-based ensembles, to diagnose and interpret time intervals that are important to the model.

Multivariate time series (MTS) are especially challenging because the input consists of a collection of time series and both features within time series and interactions between time series can be important to models. Another contribution uses a different representation to produce computationally efficient strategies that learn a symbolic representation for MTS. Relationships between the multiple time series, nominal and missing values are handled with tree-based learners.

Applications such as speech recognition, medical diagnosis and gesture recognition are used to illustrate the methods. Experimental results show that the time series representations and methods provide better results than competitive methods on a comprehensive collection of benchmark datasets. Moreover, the proposed approaches naturally provide solutions to similarity analysis, predictive pattern discovery and feature selection.

5. Organization of this dissertation

The rest of this dissertation is organized as follows. Chapter 3 introduces the bag-of-features framework to classify the time series. Chapter 4 proposes a supervised algorithm to

discover predictive patterns from the time series. Chapter 5 develops a symbolic representation for multivariate time series classification. Chapter 6 concludes and discusses several directions for future study.

CHAPTER 2

BACKGROUND

1. Notation

We focus on both univariate and multivariate time series classification problems in this dissertation. We first define the key terms used for univariate time series classification.

Definition 1. A univariate time series, $x^n = (x_1^n, x_2^n, \ldots, x_T^n)$ is an ordered set of T values. We assume time series are measured at equally-spaced time points indexed by t. Each time series is associated with a class label y^n, for $n = 1, 2, \ldots, N$ and $y^n \in \{0, 1, 2, ..., C-1\}$. Two time series of each class from a two-class time series classification problem are illustrated in Figure 2 ($T = 400, C = 2, y^n \in \{0, 1\}$). Time series from class zero are defined by three peaks, whereas two peaks define class one, regardless of locations.

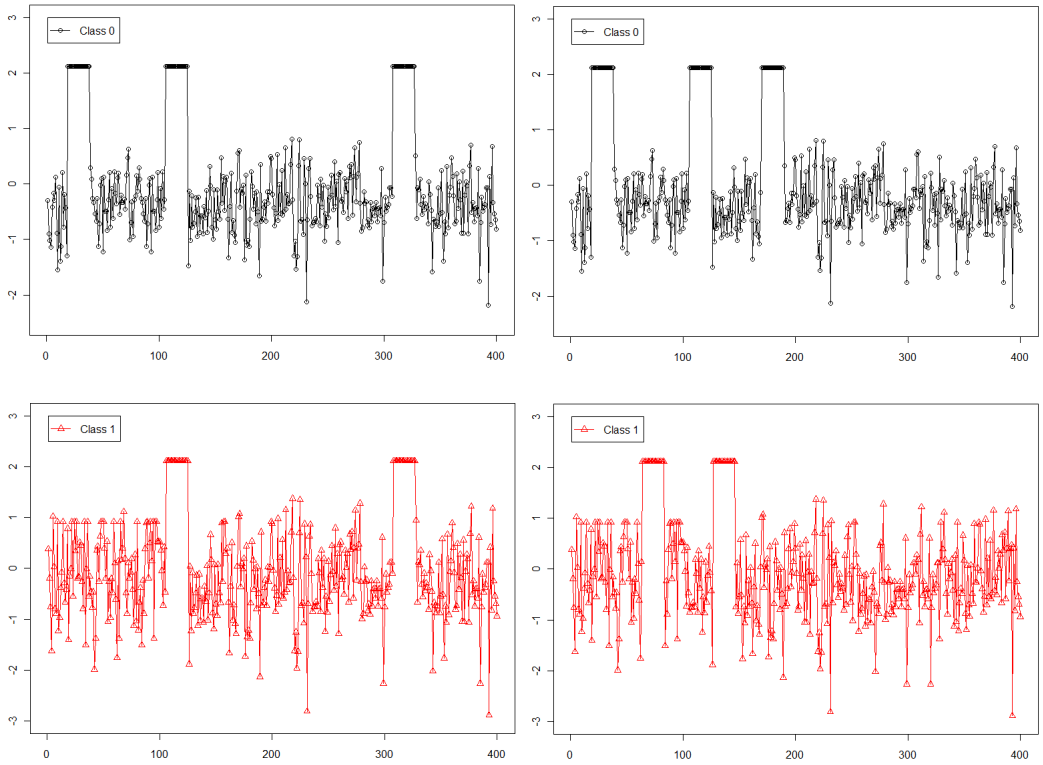


Figure 2. Two time series from each class are shown ($T = 400, C = 2, y^n \in \{0, 1\}$). The number of peaks defines each class. The location of the peaks is not important.

Definition 2. An interval of the time series x^n, $I_p(x^n)$, is a sampling of length $w < T$ of contiguous positions from x^n starting at position p. Thus, $I_p(x^n) = (x_p^n, \ldots, x_{p+w-1}^n)$ for $1 \leq p \leq T - w + 1$. Two intervals of the first time series are given in Figure 3.

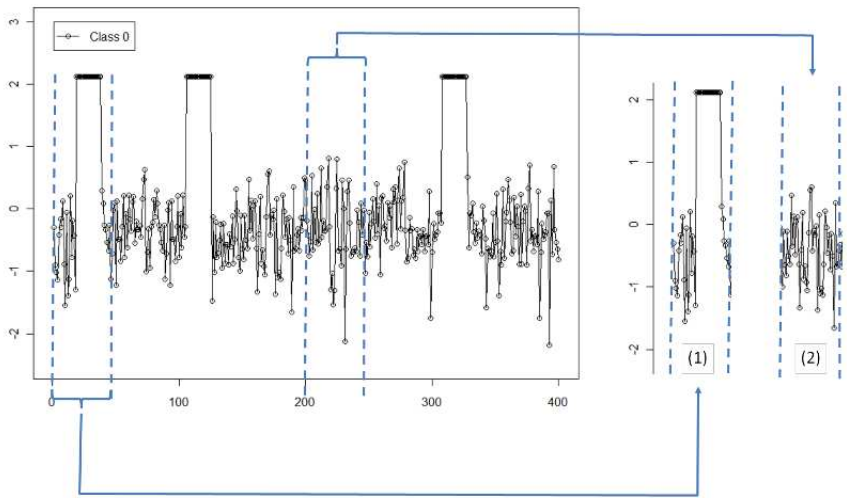


Figure 3. Two intervals (right) ($w = 50$) extracted from the time series (left). First interval starts at $t = 1$ ($p = 1$) and second interval starts at $t = 200$, ($p = 200$).

Definition 3. A subsequence of the time series, $x^n(s)$, is a time series segment consisting of d contiguous intervals. Figure 4 illustrates a subsequence of time series composed of $d = 5$ intervals each of length $w = 20$.

Definition 4. A sliding step of size $r < w$ is used to generate overlapping intervals from x^n. Let $I_p(x^n)$ be the interval of length w which starts at position p. A representative set of intervals of length w can be extracted by sliding $r < w$ positions from p across x^n. The set of the representative intervals of length w across x^n is then $\{I_1(x^n), I_{1+r}(x^n), \ldots, I_{1+T-w}(x^n)\}$. Setting $r = 1$ generates all possible intervals of length w. All possible intervals of length $w = 40$ segmented from the time series using a sliding step of $r = 20$ are illustrated in Figure 5.

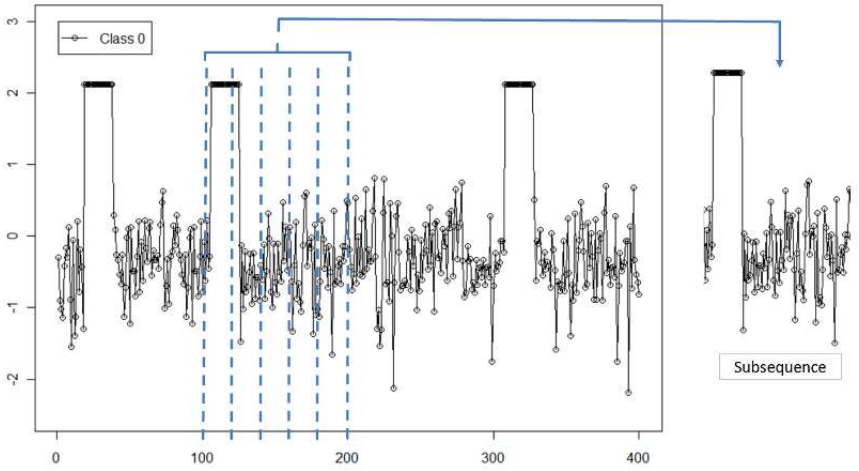


Figure 4. A subsequence starting at time $t = 200$ (right) consists of $d = 5$ intervals of length $w = 20$ time units .

Definition 5. A pattern of time series x^n, $\Psi(x_n)$, is described by the combination of certain intervals of x^n. A pattern formed by combining three intervals is schematized in Figure 6.

The notation for multivariate time series is slightly different than that for univariate time series.

Definition 6. A multivariate time series, (MTS), X^n, consists of M univariate time series each of which has T observations where $x_m^n(t)$ denotes the observation at time t from variable m of MTS n. Formally, MTS X^n is represented by $T \times M$ matrix as:

$$X^n = [x_1^n, x_2^n, \ldots, x_m^n, \ldots, x_M^n]$$

where

$$x_m^n = [x_m^n(1), x_m^n(2), \ldots, x_m^n(T)]^{'}$$

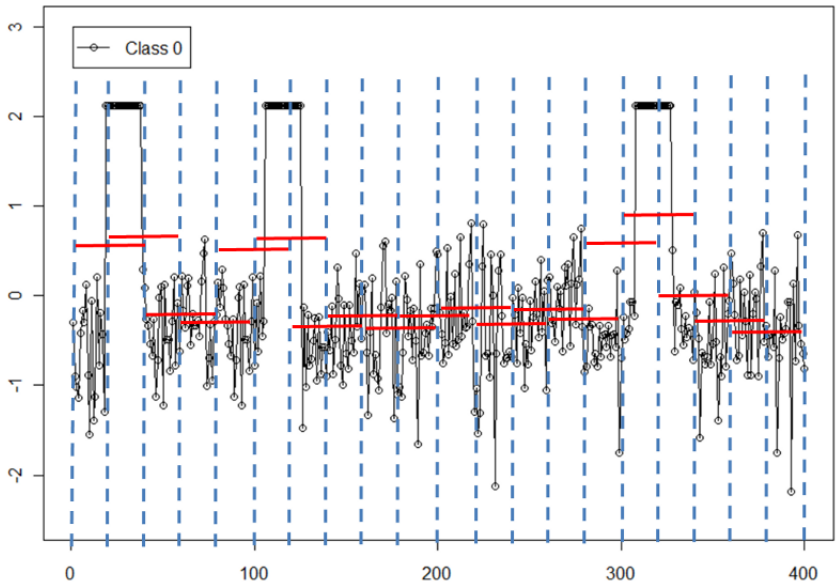


Figure 5. Shown are 19 intervals of length $w = 40$ segmented from the time series using a sliding step of $r = 20$. Mean level of the data points over each interval is represented by the red line.

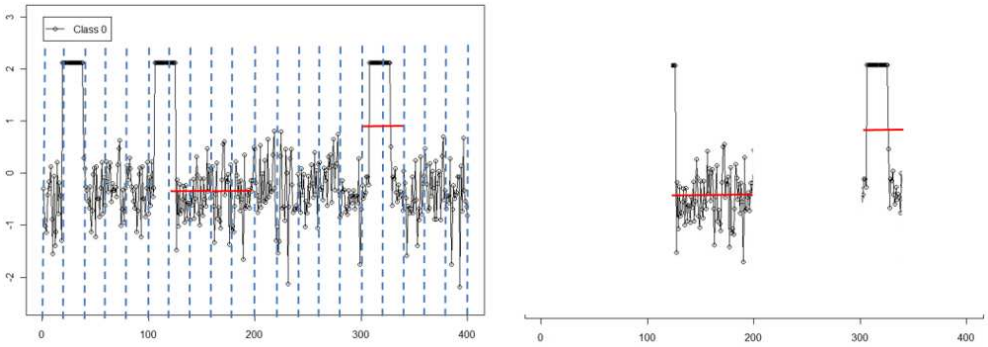


Figure 6. A pattern of time series x^n composed of 3 discontiguous intervals. It is a combination of $I_{120}(x^n)$, $I_{160}(x^n)$ and $I_{300}(x^n)$ where $w = 40$.

There are N training MTS, each of which is associated with a class label y^n, for $n = 1, 2, \ldots, N$ and $y^n \in \{0, 1, 2, ..., C - 1\}$. Univariate time series is a special case of MTS where M is equal to one.

2. Dynamic time warping

Dynamic time warping (DTW) is a well-known method for measuring similarity between two given (time-dependent) sequences (e.g. time series) which may vary in time or speed. This similarity is measured by finding the optimal alignment between two given time series under certain restrictions. An example alignment is provided in Figure 7 from [2]. Intuitively, the sequences "warped" non-linearly in the time dimension to measure similarity independent of certain non-linear variations in the time dimension.

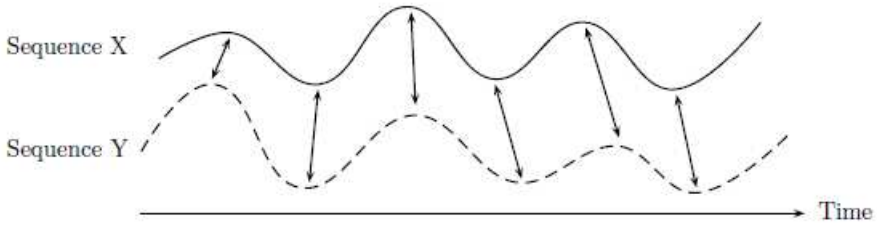


Figure 7. Time alignment of two time-dependent sequences [2]. Aligned points are indicated by the arrows

Formally, the objective of DTW is to compare two univariate time series $x^1 = (x_1^1, x_2^1, \ldots, x_N^1)$ of length N and $x^2 = (x_1^2, x_2^2, \ldots, x_M^2)$ of length M. To compare two series, one needs a local cost measure, sometimes also referred to as local distance measure [2], which is the evaluation of the local cost measure for each pair of elements of the time series x^1 and x^2. Then a cost matrix is computed as:

$$c(x_n^1, x_m^2) = \left\| x_n^1 - x_m^2 \right\|$$

Visual representation of the cost matrix $c \in R^{N \times M}$ is illustrated in Figure 8 for two time series [2].

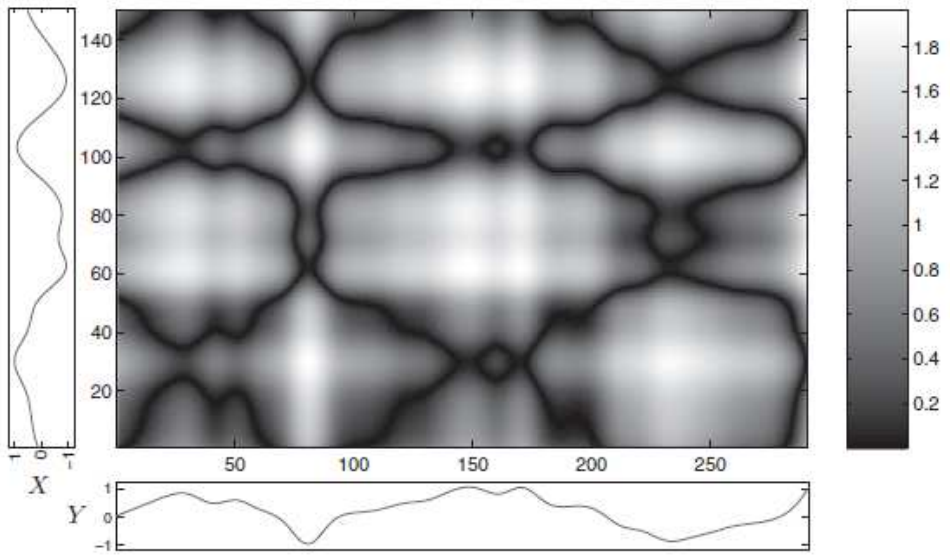


Figure 8. Cost matrix of two time series using the Manhattan distance (absolute value of the difference) as local cost measure. Regions of low cost are indicated by dark colors and regions of high cost are indicated by light colors

The goal is to find an optimal path p minimizing the overall cost. To determine an optimal path p, one could test every possible warping path between the series which is not computationally efficient. Therefore, a dynamic programming approach is proposed to solve this problem. Let $D(n, m)$ be an $N \times M$ matrix, which is also referred to as the accumulated cost matrix, then:

$$D(n, m) = min\{D(n-1, m-1), D(n-1, m), D(n, m-1)\} + c(x_n^1, x_m^2)$$

where

$$D(n, 1) = \sum_{k=1}^{n} c(x_k^1, x_1^2) \text{ and } D(1, m) = \sum_{k=1}^{m} c(x_1^1, x_k^2)$$

To recover the optimal path, tracing back from the upper right corner of D (denoted as $p_l = (N, M)$) is required:

$$p_{l-1} = \text{arg min}\{D(n-1, m-1), D(n-1, m), D(n, m-1)\}$$

.

The optimal path is illustrated in Figure 9(a) (white line) for the time series of Figure 8 [2]. Here, p^* covers only cells of c that exhibit low costs. The resulting accumulated cost matrix D is also provided in Figure 9(b) [2].

Various modifications have been proposed to better control the possible routes of the warping paths [2]. We refer to [2] for further details of the modifications of DTW.

3. Bag-of-Features approach

Bag-of-features (BoF) approach characterizes complex objects by feature vectors of sub-objects. BoF representations are popular, mostly in computer vision as content based image retrieval [31–33], natural scene classification [34] and object detection and recognition [35–39] because of their simplicity and good performance [40]. A BoF is also referred to as bag of words [41] (in which occurrences of each word are counted to summarize the text contents in document), bag of instances in the multiple instance learning (MIL) literature [42, 43] and bag of frames in audio and speech recognition [44, 45].

The basic idea is illustrated in Figure 10 for images [3]. In the traditional approach to bag-of-words representation, the local image regions are first sampled using an appropriate method (e.g., random, interest point detector [46]) and characterized by features computed from the pixels in the region (e.g., distribution of the pixel values). Each region generates a vector of features, and there can be many rows generated from each image. A visual

dictionary (or codebook) is then learned using the collection of rows from all images (e.g., clustering to assign discrete labels to regions). The resulting distribution of the regions is quantized through the codebook (e.g., a histogram of the cluster assignments for the sampled regions of each instance) as the summary of the image.

Similar to the terms used in computer vision problems, time series segments may contain rich local information about the time series. A BoF representation allows one to integrate local information from segments of the time series in an efficient way. Moreover, assumptions on the cardinality of the local feature set and patterns in the same time interval can be relaxed by this framework.

In areas such as image classification, codebooks can be constructed in supervised and unsupervised manners using the local feature set. Unsupervised construction often uses clustering algorithms such as k-means [47–49] or agglomerative clustering [50, 51]. The collection of rows from all images are clustered, and a cluster ID is assigned to each region as illustrated in 10. An alternative approach to clustering is to generate a codebook based on the histogram of the raw features [44]. However, these representations are highly dependent on the histogram generation procedure. Also, similarity-based approaches are proposed in the MIL literature. Instead of labeling instances, the similarity of the instances within a bag and between bags are used to construct the codebook [52].

As opposed to the unsupervised case, the class labels are used to guide the learning of the codebook in supervised approaches. The class labels for the sample regions are unknown but the class for of the image is known in these studies. [33, 42, 53] made use of similarity information for feature transformation under certain assumptions in MIL. [54,55] classified regions with decision trees and then predicted labels for the image. The class

label defined for each region is the class of the corresponding image in these studies. [56] extended this idea with randomly-created clustering trees whose leaves define a partitioning or grouping.

4. Random forest

A RF is an ensemble of J decision trees, $\{g_j, j = 1, 2, \ldots, J\}$. Each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the single tree. These are called out-of-bag (OOB) samples.

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 instances in the leaf node that x is assigned to by the rules that define 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

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

where $I(\cdot)$ is an indicator function that equals one if its argument is true and zero otherwise.

In the tree growing steps of RF, the best split are determined based on only a random sample of features. In this study, features are also referred as variables and both terms are used interchangeably. The Gini measure of impurity is used to determine the feature selected to make the nodal split in the tree construction process. Often, the number of features evaluated for split decision 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 instances). Therefore, for a large number of features a RF can be as computationally efficient as a single decision tree.

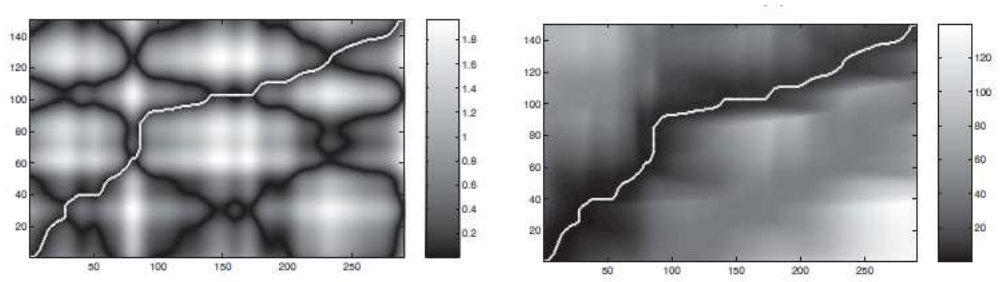
RF provides a variable importance measure called Gini Variable Importance (GVI) which is the sum of the Gini impurity decrease for a particular variable over all trees. Let N_j^ρ be the number of observations at node ρ of the j^{th} tree, and $N_j^\rho(L)$ and $N_j^\rho(R)$ be the number of observations of the left and right child nodes after splitting, and let $d_j^\rho(k)$ be the decrease in impurity produced by variable k at the ρ^{th} node of the j^{th} tree.

The decrease in impurity is $d_j^\rho(k) = G_j^\rho - (\frac{N_j^\rho(L)}{N_j^\rho} G_j^\rho(L) + \frac{N_j^\rho(R)}{N_j^\rho} G_j^\rho(R))$ where $G_j^\rho(L)$ and $G_j^\rho(R)$ are the Gini indices of the left and right node respectively and G_j^ρ is the Gini index of the parent node. The Gini Variable importance of variable k is defined as

$$GVI(k) = \frac{1}{J} \sum_{j=1}^{J} (\sum_{\rho \in S_j} d_j^\rho(k) I_j^\rho(k))$$

where $I_j^\rho(k)$ is an indicator variable for whether variable k was used to split node ρ of tree j and S_j is the set of split nodes of the tree j.

To summarize, an instance is labeled through a majority voting approach using the tree results for which it is OOB. The estimates computed from OOB predictions are easily obtained and have been shown to be good estimates of generalization error [57]. Variable importance is important to find out the features relevant to the classification task. Moreover, RF has several advantages when compared to other classifiers. High dimensional feature spaces, multiple classes, and missing values are handled. Nonlinear models and interactions between features are allowed. It is scale invariant and robust to outliers, and computations are reasonable even for large datasets.



(a) (b)

Figure 9. (a) Cost matrix c as in Figure 8 and (b) accumulated cost matrix D with optimal warping path p^* (white line).

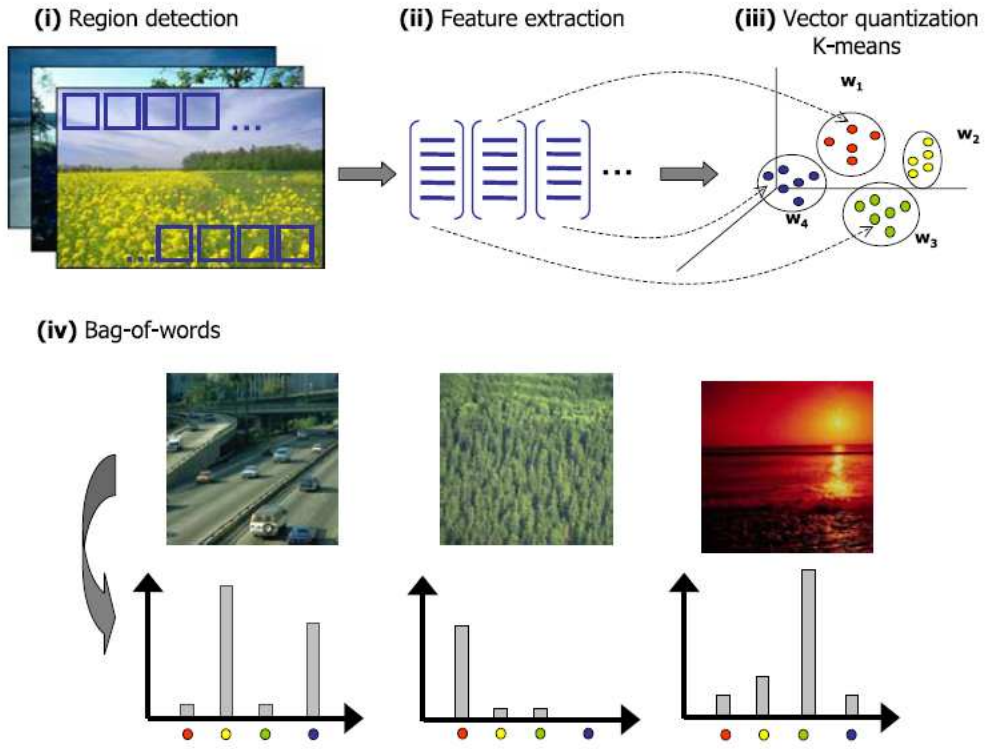


Figure 10. Four steps to compute the bag-of-words representation for images [3]. (i–iii) obtain the visual dictionary (or codebook) by vector quantizing the feature vectors of sampled regions using an appropriate method (k-means clustering in this example), and (iv) compute the image histograms – bag-of-words – for images according the obtained codebook. (ii) shows three columns representing the images and each row is a feature vector of a sampled region from the corresponding image. The sampled regions are then labeled by unsupervised learning at the subsequent step.

CHAPTER 3

A BAG-of-FEATURES FRAMEWORK TO CLASSIFY TIME SERIES

1. Abstract

Time series classification is an important task with many challenging applications. Nearest-neighbor classifiers with Dynamic Time Warping (DTW) distance is a strong solution in this context, but its performance degrades with long time series, relatively short features of interest, and moderate noise. On the other hand, feature-based approaches have been proposed as both classifiers and to provide insight into the series, but these approaches have problems handling translations and dilations in local patterns, which can be important for classification. Considering the shortcomings of both approaches, we present a framework to classify time series based on a bag-of-features representation (TSBF). Local information is captured from multiple subsequences selected from random locations and of random lengths and partitioned into shorter intervals. Consequently, features computed from these subsequences measure properties at different locations and dilations when viewed from the original time series. This provides a feature-based approach that can handle warping, although in a substantially different manner from DTW. We further partition subsequences into intervals to detect patterns represented by a series of measurements over shorter time segments. Local features are aggregated into a compact codebook through class probability estimates from a supervised learner. Additional information (such as subsequence locations) are easily integrated through a fast, efficient learner that handles mixed data types, different units, etc., and relevant global features can easily supplement the codebook in our framework. We compare our classifier to well-known nearests from the UCR time series database.

Key words: supervised learning, feature extraction, codebook

2. Introduction

Classification of time series is an important task with many challenging applications such as signature verification, speech recognition or financial analysis. The algorithms proposed for time series classification can be divided into instance-based and feature-based methods. Instance-based classifiers predict a test instance based on its similarity to the training instances. For time series, one-nearest-neighbor (NN) classifiers with Euclidean (NNEuclidean) or a dynamic time warping distance (NNDTW) have been widely, and successfully used [15–19]. Although Euclidean distance is time and space efficient, it is often weak in terms of prediction accuracy [17]. DTW [20] allows a measure of the similarity independent of certain non-linear variations in the time dimension, and is considered as a strong solution for time series problems [58]. Despite the fact that finding DTW distance without any modification on the algorithm is known to be computationally demanding for many applications [22], fast lower bounding function is used by [59] to prune the time series that cannot be the best match. Significant improvement is achieved in terms of computation time when the bounding scheme is used together with indexing, but a 1-NN classifier using DTW is still less tractable for real-time classification of time series [23]. Also, a DTW solution typically does not provide insight into time series features important to the classifier. For example, [60] proposed a decision tree approach which splits instances based on DTW distance between a pair of time sequences. It is faster compared to NNDTW in terms of testing, but the information provided is limited because of the feature representation. On the other hand, [61, 62] proposed an approach to find subsequences of the time series which are thought to be maximally representative of a class. These subsequences are called shapelets and algorithms based on shapelets facilitate interpretability. Because the

information provided by time series shapelets is limited to their presence or absence and the computation time required for generating them is significant, [23] proposed a more expressive shapelet representation by combining multiple shapelets in logic expressions that can be faster and more accurate. Another approach that makes use of the similarity of the series based on the subsequences is to use kernel-based classifiers. These approaches find a kernel function based on the similarity between the time series in local regions. [63] similarly generated subsequences from the time series and defined spatial similarity kernels based on the subsequences (distance-based approach), with classification from a support vector machine (SVM).

Feature-based approaches work on the feature vectors extracted from a set of instances. They are generally faster than instance-based classifiers depending on the feature extraction method and classification algorithm. [64] used knots from a piecewise-linear approximation of the time series to detect patterns and classify the series. [65] proposed an automated approach for feature extraction using a genetic algorithm. Then the extracted features were taken as inputs to a SVM [66]. [67] proposed a multi-layer perceptron neural network fed by statistical features such as means and standard deviations calculated from the time series. [68] used intervals of time series to extract features on which a SVM was trained.

Two types of features are generated in feature-based approaches, global and local features. Global features are a compact representation of the instances (such as the mean value). On the other hand, local features are extracted from segments of the time series and require such segments to be determined. Standard classification algorithms can be built on global features easily, but they may omit important local characteristics. Local features can supplement global information with useful patterns, but the set of local features may

vary in cardinality and lack a meaningful ordering. These are basic problems for many classification algorithms requiring feature vectors of fixed dimension.

Methods based on features of intervals (such as [69, 70]) assume that patterns exist in the same time interval over the instances, but a pattern that defines a certain class may exist anywhere in time, as well as be dilated in time. DTW attempts to compensate for possible time translations/dilations between features, but with long time series, relatively short features of interest, and moderate noise, the capability for DTW is degraded.

Our work is based on the bag-of-features (BoF) approach in which complex objects are characterized by feature vectors of sub-objects. BoF representations are popular, mostly in computer vision as content based image retrieval [31–33], natural scene classification [34] and object detection and recognition [35–39] because of their simplicity and good performance [40]. A BoF is also referred to as bag of words [41] (in which occurrences of each word are counted to summarize the text contents in document), bag of instances in the multiple instance learning (MIL) literature [42, 43] and bag of frames in audio and speech recognition [44, 45].

The basic idea is that local image descriptors are sampled using an appropriate method (e.g., random, interest point detector [46]) and characterized by their feature vectors (e.g., distribution of the pixel values). A visual dictionary (or codebook) is then learned using the vectors of visual descriptors (e.g., clustering to assign discrete labels to descriptors). The resulting distribution of descriptors is quantized through the codebook (e.g., a histogram of the cluster assignments for the sampled descriptors of each instance) as the summary of the image. Similar to the terms used in computer vision problems, time series segments may contain rich local information about the time series. A BoF representation allows one to

integrate local information from segments of the time series in an efficient way. Moreover, assumptions on the cardinality of the local feature set and patterns in the same time interval can be relaxed by this framework. Three implementation issues in this framework are local feature extraction, codebook generation and classification from the codebook.

Studies on BoF representations for time series data are limited with few studies in audio and speech recognition literature [44, 45, 71–73]. Time series similarity based on a bag-of-words representation was considered by [74]. Also, time series were discretized by symbolic aggregate approximation (SAX) and time series were represented as words using the symbols generated by this approach ([12]). Similarity of the time series were then computed using the histogram of the occurrences of words. This is similar to the codebook generation from patches used in computer vision problems. In [72], the speech signals were represented as images through preprocessing (simulation, strobe detection, temporal integration) and patches were segmented from the images. Using vector quantization, segments were represented by sparse codes and they were aggregated through histograms to generate features at the bag level. [45] used a clustering approach to summarize the local information to a bag level.

Histogram-based approaches for image classification problems do not take the spatial location of the local patches into account in codebook generation. Analogously, BoF models in time series ignore the temporal ordering inherent in the signal and, therefore, may not identify a specific content or pattern [75]. Also, [74] commented that most of the existing work on time series similarity focuses on distance-based similarity. They claimed that such approaches can work well for short time series, but may degrade for long time series. They

argued that it is more appropriate to measure similarity from higher-level structures (e.g., bag of words) in long time series, rather than point to point, local comparisons.

Consequently, we consider a different direction in this work. We use a feature-based approach, but extract multiple subsequences from each time series, and these subsequences are selected from random locations and of random lengths. Therefore, features computed from these subsequences (e.g., mean, standard deviation) measure properties at different locations and dilations when viewed from the original time series. We form a matrix of these features, but the value in row i and row j of the same column may be calculated from subsequences that differ in location and/or length. These features are input to a tree-based (recursive partitioning) ensemble that enables subsequences relevant to the class to be partitioned from others. In this manner, we provide a feature-based approach that can handle warping, although in a substantially different manner from DTW. Furthermore, BoF methods disregard location information. Instead, we further partition subsequences into intervals to detect patterns represented by a series of measurements over shorter time segments. Subsequences are labeled and a supervised learner is used to construct a compact codebook from simple class probability distributions. Our supervised approach provides fast, efficient time series representation for classification, even with very basic features such as slopes, means and variances from the subsequences. Additional information (such as subsequence locations) are easily integrated through a learner that handles mixed data types, different units, etc., and relevant global features can easily supplement the codebook in our framework. Finally, time series may be classified via any supervised learner. We demonstrate TSBF is efficient and accurate by comparing to alternative time series classifiers on a full set of benchmark datasets.

The remainder of this paper is organized as follows. Section 3 provides background. We summarize the problem and describe the TSBF framework in Section 4. Section 5 demonstrates the effectiveness and efficiency of TSBF by testing on a full set of benchmark datasets from UCR time series database [76]. We discuss TSBF's behavior for certain datasets, explain how TSBF works on an example and compare it to Logical-Shapelets in Section 6. Conclusions are drawn in Section 7.

3. Background

Noncontiguous patterns in time is another problem which affects the performance of DTW. An example from the OSULeaf dataset (from [76]) is illustrated in Figure 11. The aim is to classify the leaves based on their shapes. The boundary of a leaf image is represented as time series using the angles between consecutive pixel points. Because orientations of the leaf pictures are different, shifts and noncontiguous patterns are observed in the time series representations. Consequently, it is important to allow for features useful to the classifier to occur at different times in different time series instances. On the other hand, the images can be aligned to avoid the discontinuity, but this is a different problem considered in the context of rotation invariance [77] which is not considered here.

BoF representations are based on local feature extraction which samples a representative set of subsequences from the time series, and an efficient and effective representation of the time series is required [40]. A piecewise-linear approximation is the most commonly used preprocessing step for the discretization of the data in mining time series [78]. Time series approximation is an active research topic and a comprehensive literature review of time series segmentation approaches is provided by [8, 78].

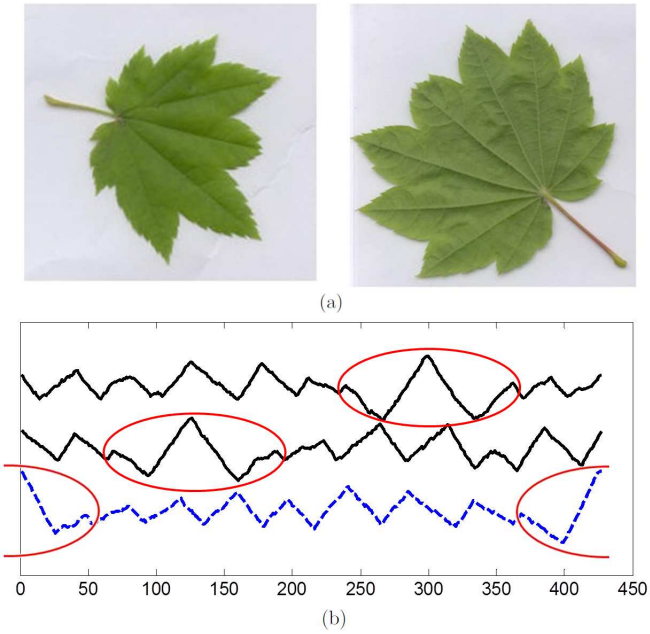


Figure 11. (a) Two sample instances from the same class from the OSUleaf dataset. The orientations of the leaves are different and this shifts patterns in the time series. (b) Time series representations of three leaves from same class. Patterns highlighted are contiguous, but the similar pattern appears in the beginning and end for the bottom series.

In areas such as image classification, codebooks can be constructed in supervised and unsupervised manners using the local feature set. Unsupervised construction does not make use of the class information from the bag. A histogram of the features from the patches can be used as the codebook ([44]) in some unsupervised approaches, however these representations are highly dependent on the histogram generation procedure. Therefore, clustering algorithms such as k-means [47–49] or agglomerative clustering [50, 51] over large sets of training patches are proposed to better represent the local feature set for image classification. [72] followed a similar approach after changing the representation of the audio data to images. In contrast to our method, regions are selected with an organized approach (not random), mean features are the focus (without location), patches are clustered with k-means

and a codebook (unsupervised) is generated based on the distribution of the cluster assignments. Also, similarity-based approaches are proposed in the MIL literature. Instead of labeling instances, the similarity of the instances within a bag and between bags are used to construct the codebook [52].

As opposed to the unsupervised case, the class labels are used to guide the learning of the codebook in supervised approaches. [33, 42, 53] made use of similarity information for feature transformation under certain assumptions in MIL. [54, 55] classified descriptors with decision trees and then predicted labels for the bag class. [56] extended this idea with randomly-created clustering trees whose leaves define a partitioning or grouping.

A random forest (RF) classifier [57] is used here to both generate class probability estimates for codebooks and to classify time series. 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 instances). Therefore, for a large number of features 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

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

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 [57]. We use OOB class probability estimates in this work.

Although other classifiers can be used in our framework, RF provides a number of desirable properties for the time series problem. High-dimensional feature spaces, multiple classes, and missing values are handled. Nonlinear models and interactions between features are allowed. Class probability estimates based on OOB instances are provided. It is scale invariant and robust to outliers, and computations are reasonable even for large datasets. Furthermore, the recursive partitioning in RF allows one feature to be used to separate some sets of time series and (potentially) a different feature to separate others. Because class-relevant patterns might appear at different time locations between series, this capability to split on different features is important to achieve the time warping through a feature-based approach (after it is combined with the random subsequence selection).

4. Time Series Classification with a Bag of Features

A univariate time series, $x^n = (x_1^n, x_2^n, \ldots, x_T^n)$ is an ordered set of T values. We assume time series are measured at equally-spaced time points. We consider univariate time series for simplicity although our method can be extended to multivariate time series in a straight-forward manner. Each time series is associated with a class label y^n, for $n = 1, 2, \ldots, N$ and $y^n \in \{0, 1, 2, ..., C-1\}$. Given a set of unlabeled time series, the task of time series classification is to map each time series to one of the predefined classes.

Note that we first 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. The basic elements of our framework for time series classification are illustrated in Figure 12. We use a supervised method to generate the temporal dictionary or codebook. Our implementation is actually simpler than this generic description. We use the class probability estimates from a supervised learner to generate the codebook in our approach. Details are provided in the following sections.

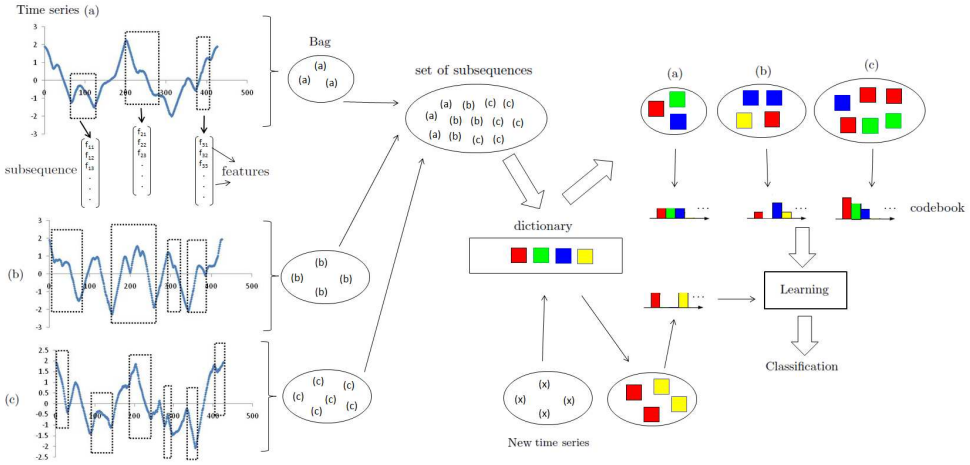


Figure 12. Generic description of the time series classification with a bag-of-features (TSBF) algorithm. Subsequences are sampled from each time series and features are extracted from the subsequences (left). Subsequence features are summarized with temporal words that are used to form a temporal dictionary or codebook. The distribution over the codebook can be described with histograms, and a supervised learner is trained on the histograms.

4.1. *Subsequences and feature extraction*

Time series classification approaches that are based on global properties of a time series can potentially be improved with local patterns that may define the class. Therefore, we represent each time series with feature vectors derived from subsequences. However, to

capture patterns along the time series, each subsequence s is represented by the features of smaller segments called intervals. A fixed-length window for segmentation has the potential to omit patterns because they may appear with different lengths and be split across the time points [8]. Thus, we generate subsequences of random length l_s and segment them using the same number of intervals to preserve the same number of features d for each subsequence. This results in intervals of random length $w_s = \frac{l_s}{d}$ which provides some desirable properties. This allows for generation of splits based on the features of different length intervals in tree-based models. Therefore, the relationships of patterns with different lengths can be better captured.

We set a lower bound on the subsequence length $l_(min)$ as a proportion $z(0 < z \leq 1)$ of the length of the time series. Thus, $l_s \geq l_{min} = z \times T$. We also set a minimum interval length w_{min} so that extracted features are meaningful (that is, we avoid a slope computed from an interval with one point). Given z and w_{min} the number of intervals to represent the subsequence is determined as $d = \left\lfloor \frac{z \times T}{w_{min}} \right\rfloor$. Note that although z and w_{min} are fixed, the actual length of an interval w_s can vary with the random samples. Consider the number of subsequences generated to represent a time series. There are $r = \left\lfloor \frac{T}{w_{min}} \right\rfloor$ possible intervals in a time series if the time series is represented using the minimum interval length. For any subsequence with d intervals $r - d$ intervals are not covered by this subsequence. We generate $r - d$ subsequences. With this setting, for every interval the expected number of subsequences that cover it is at least one.

Given the subsequence and interval calculation, we extract features from each interval and combine them to represent the subsequence. Interval features $f_k(t_1, t_2), k = 1, 2, \ldots, K$ for $(0 < t_1 \leq t_2 \leq T)$ are calculated from the data between t_1 and t_2. Linear

regression models are fit on the intervals to extract features. For each interval, the slope of the fitted regression line, mean of the values, and variance of the values are extracted. These features are important for classification because they provide information about the shape, level and the distribution of the values. A feature vector for a subsequence concatenates the features from all d intervals in the subsequence. In addition, the mean and variance of all the values in the subsequence, together with the start and end time points are also included in the feature vector. Start and end points introduce the location information which might be important for classification. That is, here $L = 4$ subsequence-level features are added. The set of subsequences S^n for time series n is built by generating $r - d$ subsequences randomly. As an illustration for $T = 100, z = 0.5, w_{min} = 10$, three subsequences are shown in Figure 13. Subsequence s of time series x^n is denoted as $x^n(s)$.

The local feature extraction algorithm for the time series is given in Algorithm 1 and illustrated in Figure 13. Here $f_k(t_1, t_2)$ is denoted as $f_{ik}(s)$ for simplicity, where i is the interval and k is the feature index. The mean and variance of the values in the subsequence s are given as $mean_s$ and var_s, respectively. The start and end points are represented as st_s and e_s.

Algorithm 1 Local feature extraction from the subsequences of time series x^n

Set subsequence count as $s = 1$
repeat
  Generate a subsequence length $l_s \in [z \times T, T]$ and a starting point for subsequence $p \in \{1, 2, ..., T - l_s + 1\}$
  Set the feature set of subsequence s as $S^n = \emptyset$
  for $i = 1$ to d **do**
    Add features from interval i of subsequence $x^n(s)$ to S^n
  end for
  Add subsequence features to S^n for subsequence $x^n(s)$
  Set the class label of subsequence $x^n(s)$ as y^n, $s = s + 1$
until $s > r - d$

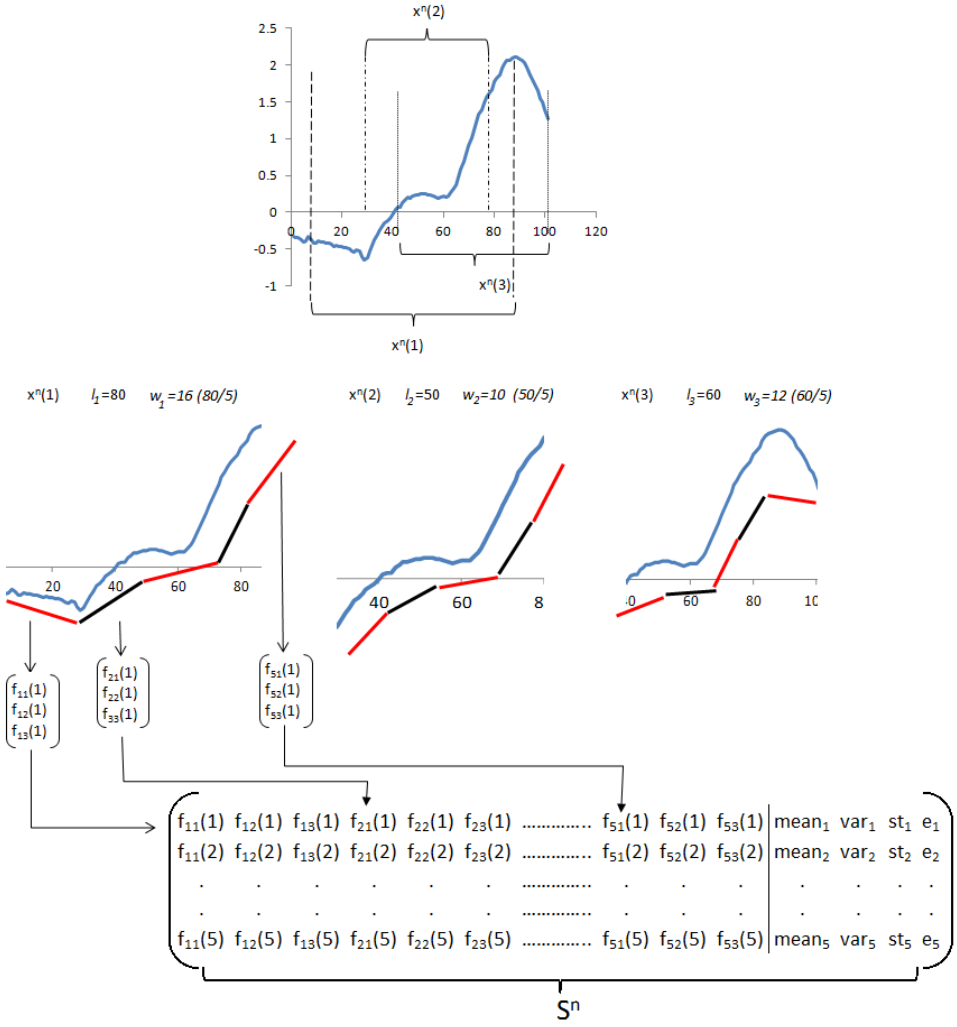


Figure 13. Interval and subsequence generation and representation. Subsequences of random length are sampled from the time series (top). Each subsequence is partitioned into intervals of length $w_s = \frac{l_s}{d}$ where d is determined by w_{min} and z as $d = \lfloor \frac{0.5*100}{10} \rfloor = 5$ (middle). A subsequence is represented by features computed from the intervals (bottom). Each instance in the feature matrix represents a subsequence. The number of subsequences generated for the time series is $\lfloor \frac{100}{10} \rfloor - 5 = 5$

4.2. *Codebook and learning*

After the local feature extraction for each time series, a new dataset is generated where each subsequence from each time series becomes an instance. The class label defined for

each instance is the class of the corresponding time series. We train a supervised learner on the new dataset and extract histograms from the classification results (such as error rates or class probability estimates) to construct a codebook.

In our approach, we use a classifier that generates a class probability estimate for each instance (subsequence). The estimate provides information on the strength of an assignment. Let $p_c^n(s)$ denote the class probability estimate for class c from subsequence s of series x^n. For each time series x^n and each class c, the distribution of $p_c^n(s)$ over s is summarized with a histogram with b bins (denoted by a vector h_c^n). The vectors are concatenated over each class c to form the codebook, h^n, for time series x^n. Because the sum of class probability estimates for a subsequence is equal to one, the features for one class can be dropped in the codebook. We use equally-spaced bins in our approach so that $(C-1) \times b$ features are in the codebook. We aim to capture the details of the similarity between subsequences with the histograms of class probability estimates. The relative frequencies of the predicted classes for the subsequences supplements the codebook. That is, if we generate 10 subsequences for a single time series in a two-class problem and seven subsequences are assigned to one class, the relative frequency of this class is $7/10 = 0.7$. The information provided by the relative frequencies is less detailed, but a meaningful summary, of the class probability estimates.

A codebook is an effective way to simplify the information in the subsequences in terms of speed and good discrimination [40, 56]. Using the predictions of a classifier trained on the subsequence information, we produce the codebook as the summary of the local information. We use a RF to generate the class probability estimates, although another

learner that provides class probability estimates can be used in the framework. We denote the RF applied to the subsequence dataset as $RFsub$.

Moreover, global features such as autocorrelation are easily introduced to obtain a better representation of a time series. In addition to representation of the time series as codebooks, we can add any global feature that has the potential to improve the classification results. After adding the global features, any supervised learner can be used to classify the time series. The main algorithm is summarized in Algorithm 2.

Algorithm 2 TSBF (Time Series Classification Based on a Bag-of-Features Representation)

for all time series x^n **do**
  Standardize the time series
  Generate the features for subsequences S^n
end for
Build a classifier on $\bigcup_{n=1}^{N} S^n$
for all time series x^n **do**
  Construct the codebook using classification results
  Generate global features
end for
Classify the time series using the codebook and the global features

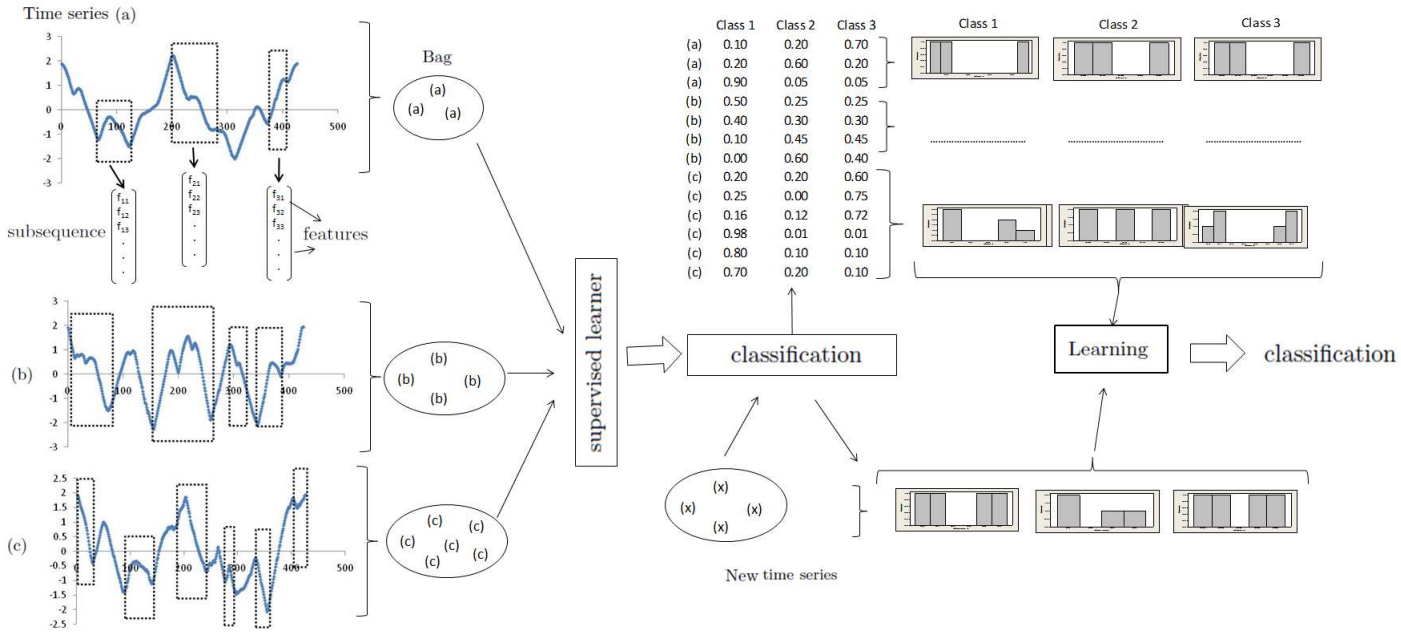


Figure 14. More specific description of the time series classification with a bag-of-features (TSBF) algorithm. Subsequences are sampled from the time series and features are extracted from the subsequences (left). Each subsequence is labeled with the class of the time series, and a learner generates class probability estimates. Histograms of the class probability estimates are generated (and concatenated) to summarize the subsequence information. Global features are added. A final classifier is then trained on the new representation to assign each time series.

Given the codebook and the global features, a RF is applied to classify the dataset of time series. This RF is denoted as $RFts$. RF is competitive with the widely used learners on high-dimensional problems [79]. Fast evaluation is another important requirement for most of the time series classification tasks and RF is fast in terms of both training and evaluation. Moreover, it is inherently multiclass; therefore, building several binary classifiers (as for one-versus-one training in a support vector machine) is not required. Figure 14 illustrates some of the steps of our approach. The TSBF is slightly simpler than the generic algorithm in Figure 12 because a supervised learner generates class probability estimates that are directly used as inputs to histograms. We consider the bins used to form the histograms of the class probability estimates and the frequency of the predicted classes as the temporal dictionary (codebook) in our specific approach.

4.3. *Illustrative Examples*

We discuss how the BoF approach handles patterns with two simple examples. The first example illustrates location invariance. Consider a two-class problem in which series from class zero are defined by three peaks, whereas two peaks define class one, regardless of locations. Two time series from each class are illustrated in Figure 15. These series are standardized. Methods built on interval features which assume that patterns exist in the same time interval over the series can have problems, as shown in our small analysis below.

The location of the peaks for the series in Figure 15 are $[20, 40]$, $[60, 80]$, $[105, 125]$,$[130, 150]$, $[165, 185]$ and $[310, 330]$. Suppose that subsequences of length 80 are generated and characterized with the mean over two intervals (each of length 40) for illustration purposes. The subsequences are generated in a sliding manner with an overlap of one interval (and this yields 9 subsequences per series). Denote the mean features of inter-

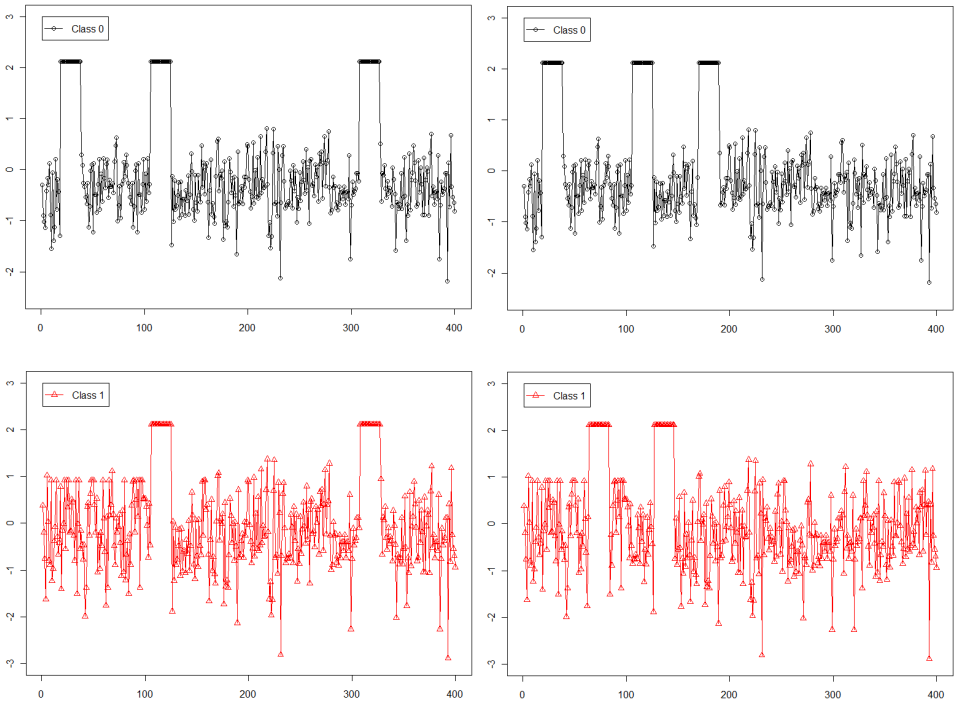


Figure 15. Two time series from each class are shown. The number of peaks defines each class. The location of the peaks is not important.

val 1 and 2 as f_1 and f_2, respectively, The two series of each class are illustrated in Figure 15. Figure 16(a) provides the histogram of the class probability estimates from $RFsub$ for two time series of different classes. As given in Figure 16(a) , a supervised learner can separate the two classes based on the subsequence distributions. Numerical results are shown later in this section. The subsequence location features are not important for this particular example.

A different type of example defines the classes by the locations of the peaks (Figure 17). If the peak is in the first half of the time series, the series is class zero, and otherwise it is class one. Suppose we generate the subsequences in the same way as we did for the first example with an interval length of 5. We use a smaller interval length here to demonstrate

the effects of subsequence lengths in the following experiment. The distributions of the subsequences from each time series are the similar if only mean interval features are used. However, subsequence location features (i.e., the start and end point features) capture the location information for the subsequences. The distribution of the subsequences is illustrated on the three-dimensional feature space in Figure 16(b).The location feature provided in the figure is the average of the start and end point of the subsequence (i.e. midpoint of the subsequence).

To further describe the characteristics of our approach, we discard the location features so that the interval means are not sufficient features when the class is defined by peak location. However, such a situation can be handled with a simple approach. That is, generate longer subsequences. Consider the case where a subsequence is the time series itself. Then classifiers (such as trees) based on features (such as means) from fixed intervals can easily identify these classes. Our approach could also handle this by generating multiple random subsequences based on a minimum subsequence length setting. Also, the smaller the minimum subsequence length, the more subsequences are generated as described in Section 4.1. This increases the likelihood of having longer subsequences. We present an example analysis below. However, our basic algorithm considered in the experiments section includes location features.

In order to illustrate the BoF approach we generate synthetic data for each of these examples. The first and second examples are referred to as the number of peaks and the peak location examples, respectively. The length of each time series is set to 400 and 200 time units, respectively. Following the class definitions, we generate a small dataset with 10 time series from each class for training. Then, 200 time series per class are generated for

testing. Table 1 summarizes the average test error rates over 10 replications of two versions of TSBF which are the regular and TSBF without the subsequence location features. We also include a row for a RF classifier (with 500 trees) based on the mean features computed from non-overlapping intervals of five time units each. This RF might be considered a simple, baseline approach to classify the time series. We consider four levels of minimum subsequence length settings, $z \in \{0.1, 0.25, 0.5, 0.75\}$, bin size of 10 and 500 trees in both RFs ($RFsub$ and $RFts$). We also set an upper limit on the subsequence length (denoted as a proportion u) in these synthetic examples to illustrate the role of location features in the results for TSBF, but the default TSBF algorithm does not use such an upper limit ($u = 1$). Also, the number of sequences per times series followed the $r - d$ formula described in Section 4.1.

			Test Error Rate	
Method	Lower Bound Factor	Upper Bound Factor	Example with Peak Location	Example with Number of Peaks
TSBF	0.1	1	0.005	0.001
TSBF	0.25	1	0.011	0
TSBF	0.5	1	0.012	0
TSBF	0.75	1	0.01	0.001
TSBF w/o location	0.1	0.25	0.441	0
TSBF w/o location	0.1	0.5	0.244	0.002
TSBF w/o location	0.1	0.75	0.056	0.014
TSBF w/o location	0.25	0.5	0.289	0
TSBF w/o location	0.25	0.75	0.098	0
TSBF w/o location	0.5	0.75	0.074	0
RF			0.11	0.16

TABLE 1. Average test error rates over 10 replications for TSBF, TSBF without the subsequence location features (TSBF w/o location) and baseline RF classifier (last row) applied to two synthetic datasets. The TSBF performance is always relatively good for the number of peaks example because it is amenable to the bag of features approach. Error rates for the peak location example are lower when longer subsequences are allowed to be generated without the location features.

The results from these simple examples illustrate properties of the BoF approach. As expected, for the number of peaks example all the TSBF algorithms perform well, better

than an RF based on fixed intervals. This shows the strength of the BoF approach and the limitations of features from fixed intervals in such examples.

For the peak location example, the error rates for TSBF are substantially smaller than the error rate of RF classifier. This illustrates the problem a generic classifier can have with location invariance. For this experiment only, to study the effects, we constrained the maximum subsequence length. The results with $z = 0.1, u = 0.25$ show much poorer performance with the constrained shorter intervals when the location features are not used. For intermediate constraints (such as $z = 0.25, u = 0.5$) the performance results are relatively moderate. However, even with a small $z = 0.1$, the results improve substantially when we relax the maximum constraint to $u = 0.75$. Without the location features, longer subsequences are required to capture certain characteristics.

Another mini experiment is designed to compare our supervised BoF approach to an unsupervised one with a codebook derived from K-means clustering. In the unsupervised approach, the Euclidean distance between the subsequences generated by our BoF approach is computed. Then K-means clustering with different k settings is used to label subsequences. We use the histogram of the cluster assignments to generate the codebook. To avoid a normalization step because of the differences in scales of the other features (i.e. mean, variance and slope have different scales), only the mean features of the intervals and subsequences are used. In a similar manner, the location features are also discarded. Consequently the results here should not be compared with those in Table 1. The time series are standardized in the algorithm. We train an RF on the unsupervised codebook for classification. Table 2 summarizes the average test error rates over 10 replications of TSBF and BoF approach with an unsupervised codebook.

		Test Error Rate	
Method	Lower Bound Factor	Example with Peak Location	Example with Number of Peaks
TSBF (w/o locations, means only)	0.1	0.021	0.026
K-means ($k = 10$)	0.1	0.011	0.427
K-means ($k = 25$)	0.1	0.012	0.411
K-means ($k = 50$)	0.1	0.006	0.356
TSBF (w/o locations, means only)	0.25	0.026	0.006
K-means ($k = 10$)	0.25	0.011	0.458
K-means ($k = 25$)	0.25	0.013	0.452
K-means ($k = 50$)	0.25	0.021	0.398
TSBF (w/o locations, means only)	0.5	0.035	0.007
K-means ($k = 10$)	0.5	0.013	0.497
K-means ($k = 25$)	0.5	0.005	0.444
K-means ($k = 50$)	0.5	0.014	0.430
TSBF (w/o locations, means only)	0.75	0.022	0.040
K-means ($k = 10$)	0.75	0.034	0.490
K-means ($k = 25$)	0.75	0.017	0.466
K-means ($k = 50$)	0.75	0.012	0.465

TABLE 2. Average test error rates over 10 replications for TSBF, and RF classifiers trained on an unsupervised codebook generated by K-means clustering applied to two synthetic datasets. TSBF uses mean features only (location features along with slopes, variances, etc., are omitted).

The TSBF performance is substantially better for the number of peaks example because Euclidean distances are not descriptive in this case. On the other hand, our supervised approach uses only the relevant features from the supervised learning and generates the class probability estimates (codebook) accordingly. There are not dramatic differences in error rates with k.

Error rates for the peak location example are comparable. The time series of this dataset are not noisy as illustrated in Figure 17 and our local feature-extraction scheme generates subsequences that can characterize the time series well. Consequently, the classes can be well clustered with Euclidean distance. Still, TSBF even without location features (or other features such as slopes and variances) is competitive in performance.

5. Experiments and Results

We tested TSBF on a full set of time series data from [76]. The dataset characteristics are given in Table 3. This is a good testbed with diverse characteristics such as length of the series, number of classes, etc., which enables a comprehensive evaluation.

	Number of classes	Training instances	Test instances	Time series length
50words	50	450	455	270
Adiac	37	390	391	176
Beef	5	30	30	470
CBF	3	30	900	128
Coffee	2	28	28	286
ECG200	2	100	100	96
FaceAll	14	560	1,690	131
FaceFour	4	24	88	350
Fish	7	175	175	463
GunPoint	2	50	150	150
Lightning2	2	60	61	637
Lightning7	7	70	73	319
OliveOil	4	30	30	570
OSULeaf	6	200	242	427
SwedishLeaf	15	500	625	128
Syntheticcontrol	6	300	300	60
Trace	4	100	100	275
TwoPatterns	4	1,000	4,000	128
Wafer	2	1,000	6,164	152
Yoga	2	300	3000	426

TABLE 3. Characteristics of the time series: number of classes, number of training instances, number of testing instances, and lengths of time series. The performance analysis of the algorithms on this diverse set of data provides a wide-ranging comparison.

Our algorithm does not require the setting of many parameters and it is robust to the settings. A RF is insensitive to both the number of trees and the number of candidate attributes scored to potential split a node [57]. For example, Figure 18 illustrates how the OOB error rate changes as the number of trees increases for $RFsub$ and $RFts$. Error rates provided are the average OOB error rate based only on the training data over all 20 datasets. This error rate is a good estimate of generalization error [57]. This parameter is not determined using the accuracy on the test data. As the number of trees increases, the

error rates improve, but the marginal gain is comparably small after 400 trees. Therefore, the number of trees is set to 500 for both forests. Although we set the level as 500 trees for all datasets, this parameter may be adjusted for each dataset based on the OOB error rate of RF. If fewer trees are enough for certain datasets, this can reduce the computation time.

The number of features evaluated at each node of the tree is set to the default which equals the approximate square root of the number of features. Therefore, the number of features generated for $RFsub$ is $K = 3$ features per interval plus four features for the subsequence ($= 3 \times d + 4$). For $RFts$ the number of features is $(C - 1) \times b$ features for class probability estimates plus $(C-1)$ features for class frequencies ($= (C-1) \times (b+1)$).

The codebook is determined from three parameters. We simply set the minimum interval length w_{min} as five time units in order to have meaningful features (such as slopes). This setting can be adjusted based on the dataset characteristics in favor of our algorithm (as discussed in Section 6), but we did not modify it because the random subsequence generation scheme allows for larger interval lengths to occur. The number of bins b is set to 10 in our experiments. This parameter is expected to have a small effect on performance, if it is set large enough, because of the embedded features selection in RFs. We illustrate some effects of this parameter in Section 6. We tested our algorithm for different minimum subsequence length settings z and compare the differences in the results. We replicate TSBF 10 times with different seeds. Classification accuracy and solution characteristics are discussed in the following sections.

5.1. *Classification accuracy*

TSBF with the given settings is compared to nearest neighbors (NN) classifiers with DTW. Two versions of DTW are considered: NNDTWBestWin (also referred to as NNBest-

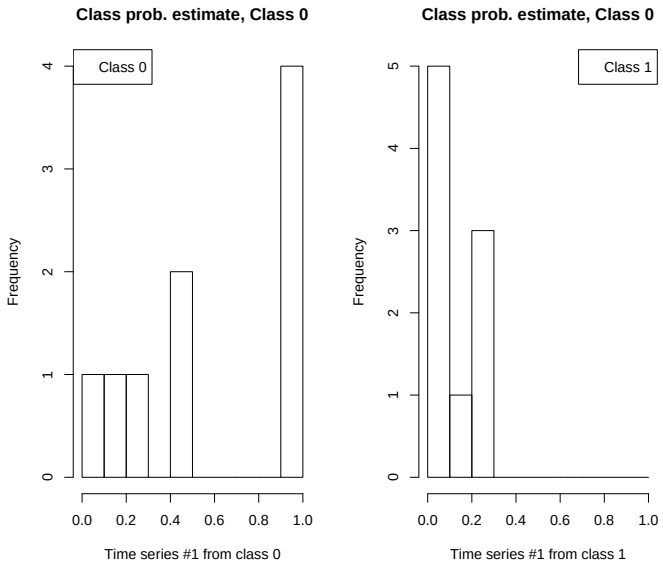
Parameter	Levels
number of trees	500
number of features in each split	$\sqrt{numberoffeatures}$
z	$\{0.1, 0.25, 0.5, 0.75\}$
w_{min}	5
b	10

TABLE 4. Parameter settings of TSBF

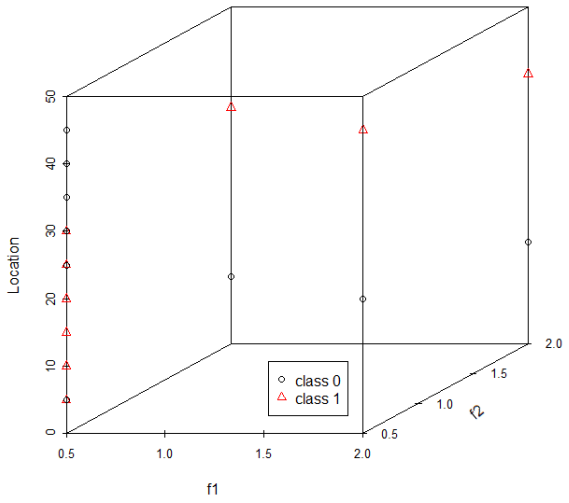
DTW) [17] searches for the best warping window, based on the training data, then uses the learned window on the test data, while NNDTWNoWin has no warping window. Note that DTW is a strong solution for time series problems in a variety of domains [58], although it is limited in real-time applications because of computational requirements [23]. The results for NN classifiers are obtained from [76]. Table 5 summarizes the average, maximum and minimum error rates from 10 replications of our algorithm on the test data. Features generated for the test data are based on the same subsequence locations generated for training.

We also compare our results with Logical-Shapelets [23] which significantly outperforms the original shapelet representation proposed by [61]. Because this comparison is not based on all datasets due to the computational requirements of Logical-Shapelets, we compare TSBF to Logical-Shapelets in Section 6.2.

We use the same approach proposed by [21] to compare results. Scatter plots are used to conduct pairwise comparisons of error rates. Each axis represents the approach under consideration and each dot represents the error rate for a particular dataset. The line $x = y$ is drawn to represent the region where both methods perform about the same. A dot above the line indicates that approach on the X axis has better accuracy than the one on Y axis. If a dot is further from the line, the margin of accuracy improvement is greater. A method can be regarded as superior to other if there are more dots on one side of the line.



(a) Example with the number of peaks. The two classes can be separated based on the subsequence distributions.



(b) Example with peak location. The two classes can be separated with the addition of location features.

Figure 16. Distributions of the subsequences in the feature spaces of interval means for two examples.

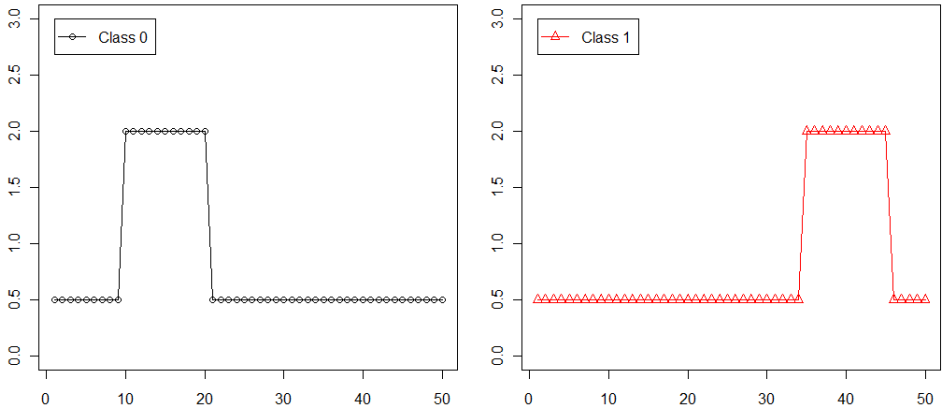


Figure 17. One time series from each class is shown. A peak in the first or second half of the time series defines class zero or one, respectively.

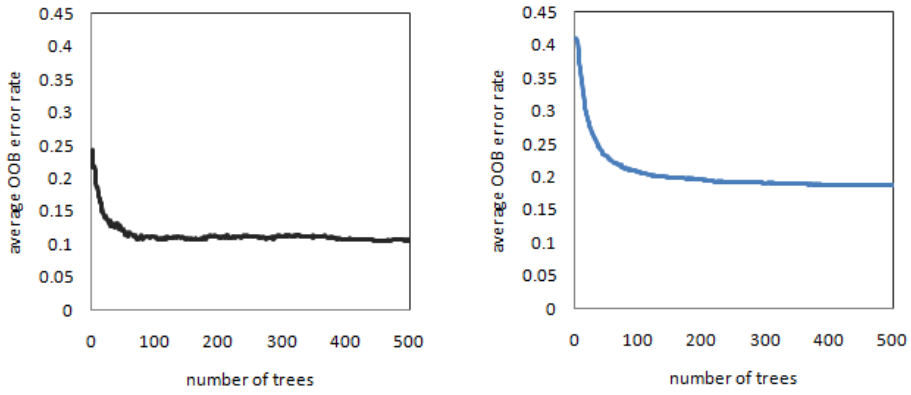


Figure 18. The average OOB error rates on the training data of RFts (left) and RFsub (right) over all datasets. The plots indicate that the results are insensitive to the number of tree when the number is sufficiently large (500 in our case).

	TSBF ($z = 0.1$)			TSBF ($z = 0.25$)			TSBF ($z = 0.5$)			TSBF ($z = 0.75$)			NNDTW	
	average	max	min	average	max	min	average	max	min	average	max	min	BestWin	NoWin
50Words	0.200	0.213	0.185	0.191	0.204	0.180	0.199	0.215	0.182	0.202	0.215	0.185	0.242	0.310
Adiac	0.416	0.448	0.394	0.286	0.304	0.271	0.237	0.258	0.215	0.233	0.251	0.217	0.391	0.396
Beef	0.333	0.433	0.233	0.350	0.433	0.267	0.307	0.367	0.200	0.223	0.300	0.133	0.467	0.500
CBF	0.001	0.003	0.000	0.005	0.008	0.002	0.008	0.011	0.006	0.016	0.023	0.010	0.004	0.003
Coffee	0.054	0.071	0.036	0.004	0.036	0.000	0.000	0.000	0.000	0.004	0.036	0.000	0.179	0.179
ECG	0.183	0.230	0.140	0.138	0.160	0.120	0.155	0.190	0.130	0.145	0.190	0.120	0.120	0.230
Face (all)	0.282	0.300	0.265	0.217	0.241	0.199	0.234	0.249	0.205	0.246	0.256	0.229	0.192	0.192
Face (four)	0.045	0.068	0.034	0.038	0.045	0.034	0.035	0.045	0.023	0.026	0.045	0.011	0.114	0.170
Fish	0.095	0.114	0.080	0.071	0.091	0.034	0.076	0.114	0.063	0.073	0.086	0.046	0.160	0.167
Gun-Point	0.017	0.033	0.013	0.011	0.027	0.000	0.011	0.020	0.000	0.007	0.013	0.000	0.087	0.093
Lighting-2	0.256	0.279	0.230	0.249	0.279	0.213	0.225	0.230	0.213	0.218	0.230	0.197	0.131	0.131
Lighting-7	0.262	0.288	0.219	0.307	0.329	0.260	0.290	0.301	0.274	0.271	0.301	0.219	0.288	0.274
OliveOil	0.120	0.167	0.100	0.113	0.167	0.067	0.130	0.167	0.100	0.137	0.167	0.100	0.167	0.133
OSU Leaf	0.261	0.277	0.231	0.233	0.256	0.202	0.279	0.314	0.244	0.330	0.360	0.298	0.384	0.409
Swedish Leaf	0.173	0.195	0.152	0.089	0.101	0.080	0.067	0.074	0.062	0.075	0.088	0.053	0.157	0.210
Synthetic Control	0.064	0.100	0.037	0.019	0.037	0.007	0.008	0.013	0.003	0.011	0.020	0.007	0.017	0.007
Trace	0.013	0.020	0.000	0.020	0.020	0.020	0.020	0.020	0.020	0.020	0.030	0.010	0.010	0.000
Two Patterns	0.003	0.007	0.001	0.001	0.003	0.000	0.001	0.003	0.000	0.007	0.013	0.003	0.002	0.000
Wafer	0.008	0.010	0.007	0.004	0.005	0.004	0.004	0.006	0.002	0.004	0.006	0.003	0.005	0.020
Yoga	0.162	0.187	0.151	0.160	0.172	0.150	0.163	0.180	0.147	0.146	0.157	0.135	0.155	0.164

TABLE 5. Error rates of TSBF for four different settings of z based on average, maximum and minimum of 10 replications, nearest-neighbor classifiers with dynamic time warping distance, where NNDTWBestWin searches the best warping window based on the training data, NNDTWNoWin has no warping window.

Figure 19 summarizes the performance of TSBF against NNDTWNoWin for different levels of z. It can be observed that TSBF performs better than NNDTWNoWin on most of the datasets for all z levels.

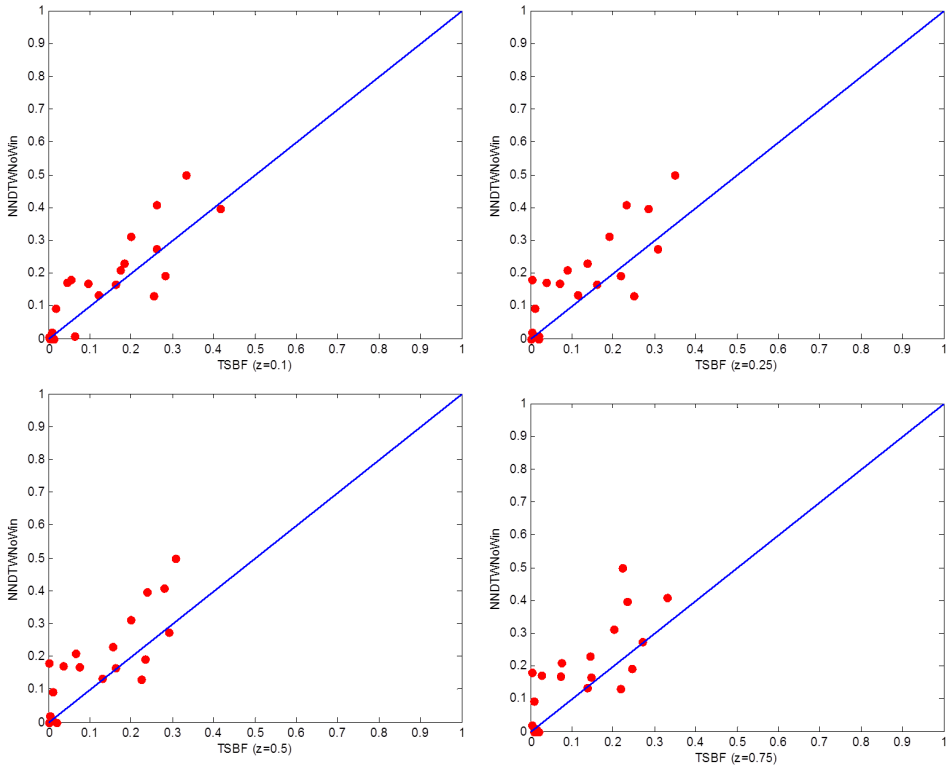


Figure 19. TSBF outperforms NNDTWNoWin for most of the datasets in all z levels

Performance of NNDTWBestWin against TSBF with different z settings is illustrated in Figure 20. The performance of TSBF is still better than DTW with the best window setting. The error rates of TSBF on OSU Leaf dataset is much smaller. The explanation relies on the connection of the time series classification to the image classification problem introduced in Section 2. Content- based image retrieval algorithms [31–33] are based the BoF idea to handle the invariances in terms of rotation and location. Consequently, TSBF can handle the rotational invariance for this particular dataset.

We also illustrated the performance of TSBF ($z = 0.5$) over the replications to illustrate the random behavior of the algorithm in Figure 21. The ranges of error rates are reasonable for most of the datasets. However certain datasets such as Beef and ECG have larger ranges compared to others. This is mainly due to the few number of test instances for these datasets (30 and 100, respectively). Thus, a single misclassification increases the error rate substantially and this results in higher variability.

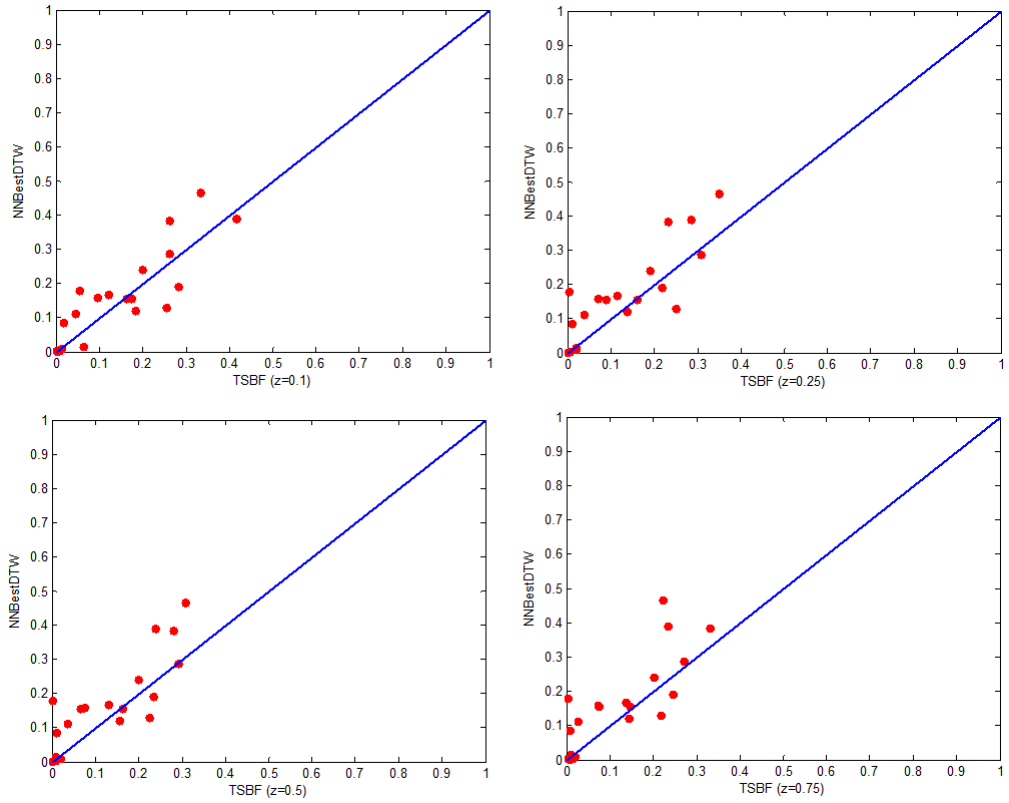


Figure 20. TSBF outperforms NNDTWBestWin for most of the datasets in all z levels

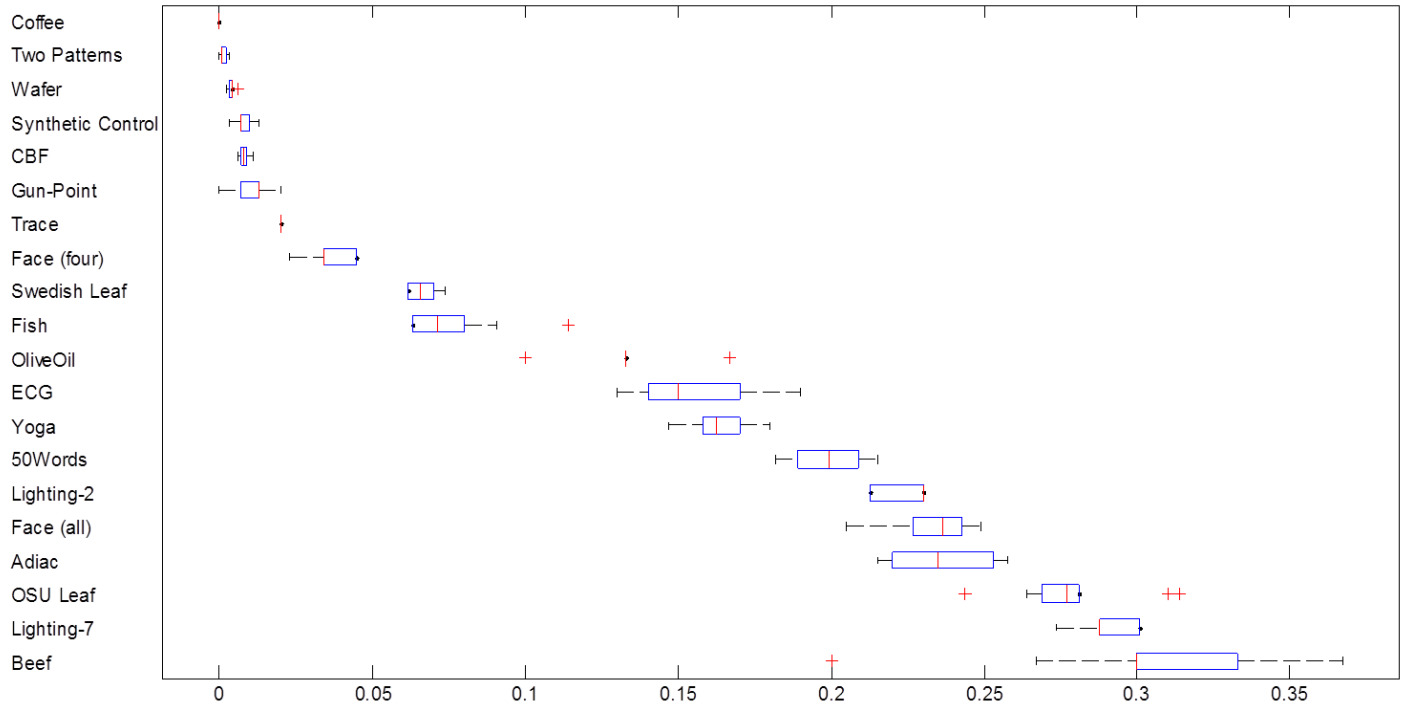


Figure 21. Boxplot of the replication results for TSBF ($z = 0.5$). Datasets are sorted based on their average error rate. Beef and ECG have larger error rate ranges compared to others. This is mainly due to the number of test instances for these datasets which are 30 and 100 respectively.

5.2. *Computational complexity*

TSBF is implemented in both C and R Software and our experiments use a Windows 7 system with 8 GB RAM, dual core CPU (i7-3620M 2.7 GHz). We use R only for building the RFs and implemented the algorithms for subsequence and codebook generation in C, because R is computationally inefficient in execution of the loops. Moreover, although the CPU can handle four threads in parallel, only a single thread is used.

The overall computational complexity of our algorithm is mainly due to $RFsub$. The time complexity of building a single tree in $RFsub$ is $O(\sqrt{\nu}\eta \log \eta)$ where $\nu = K \times d + L$ is the number of features extracted from each subsequence and η is the number of training instances for $RFsub$. The size of the training data for $RFsub$ depends on the total number of time series for training and the number of subsequences generated for each time series $|S^n|$. We generate $r - d$ subsequences for each time series (where $d = \left\lfloor \frac{z \times T}{w_{min}} \right\rfloor$). The smaller z, the more subsequences are generated, but with fewer features for each.

Computation times for training TSBF are provided in Table 6 and illustrated in Figure 22 for all z settings to show how training time changes with different parameter settings. The difference in computation times is due to the number of subsequences and features generated in each setting as provided in the complexity analysis. TSBF (z=0.25) and TSBF (z=0.5) take longer compared to other settings.

We also provide the testing time which is the time required for classifying one object (feature generation and classification through RFs) and it is not affected significantly by different parameter settings as illustrated. Our approach takes less than a second to classify single time series after the models are built. It is very fast and convenient for real time classification of time series data.

	TSBF (z=0.1)		TSBF (z=0.25)		TSBF (z=0.5)		TSBF (z=0.75)	
	Train T. (s)	Test T. (s)	Train T. (s)	Test T. (s)	Train T. (s)	Test T. (s)	Train T. (s)	Test T. (s)
50Words	48.36	0.0083	69.91	0.0077	71.42	0.0062	42.62	0.0037
Adiac	22.68	0.0046	28.09	0.0043	27.95	0.0037	15.13	0.0025
Beef	3.90	0.0129	6.53	0.0143	7.34	0.0120	4.41	0.0092
CBF	0.25	0.0011	0.37	0.0011	0.36	0.0011	0.23	0.0006
Coffee	1.07	0.0045	1.79	0.0051	1.74	0.0065	1.17	0.0038
ECG	0.87	0.0012	0.96	0.0008	0.94	0.0007	0.52	0.0008
Face (all)	12.64	0.0022	17.15	0.0019	17.68	0.0016	11.57	0.0010
Face (four)	1.50	0.0068	2.44	0.0072	2.61	0.0069	1.65	0.0044
Fish	34.25	0.0141	59.79	0.0153	65.46	0.0147	36.24	0.0116
Gun-Point	0.74	0.0015	0.83	0.0016	0.79	0.0022	0.47	0.0010
Lighting-2	12.04	0.0181	20.70	0.0223	25.25	0.0250	15.46	0.0138
Lighting-7	4.34	0.0067	6.74	0.0056	7.37	0.0062	4.56	0.0038
OliveOil	7.19	0.0214	10.76	0.0196	11.43	0.0177	6.71	0.0121
OSU Leaf	32.12	0.0125	59.86	0.0135	71.65	0.0128	38.66	0.0087
Swedish Leaf	11.00	0.0026	14.23	0.0020	14.41	0.0015	9.04	0.0009
Synthetic Control	1.53	0.0006	1.63	0.0006	1.42	0.0004	0.77	0.0002
Trace	4.19	0.0047	7.12	0.0046	7.04	0.0042	3.63	0.0042
Two Patterns	16.93	0.0022	26.90	0.0019	31.75	0.0016	20.03	0.0009
Wafer	23.90	0.0020	36.67	0.0020	42.22	0.0017	24.35	0.0010
Yoga	49.55	0.0116	94.31	0.0128	120.18	0.0122	62.83	0.0080

TABLE 6. Computation times of TSBF for different parameter settings. The differences in computation times are due to the number of subsequences and features generated in each setting. The time required to test a single time series is also given (and this includes the time required for feature generation).

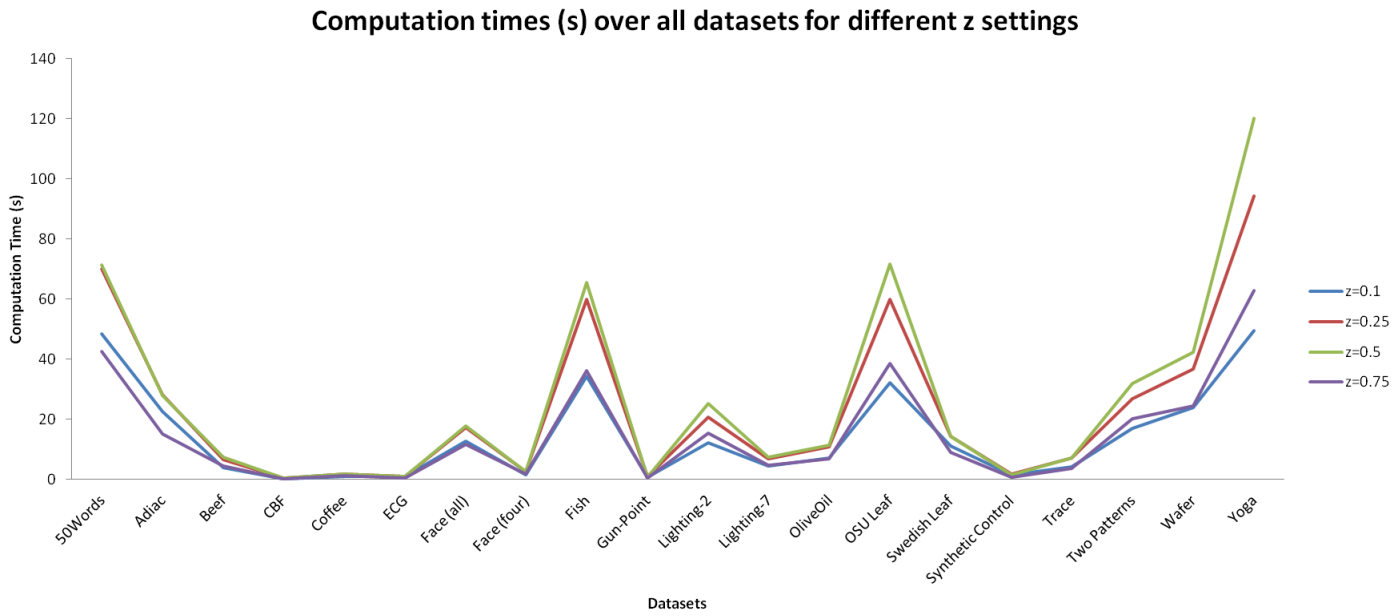


Figure 22. Computation time of TSBF over all datasets for all z settings

6. Discussion

6.1. *What OOB error rates provide*

Although our parameters are constant over all datasets, one could use OOB error to tune the parameters for each dataset. This could potentially improve the classification results further.

For example, with $z = 0.1$ TSBF performs reasonably well, but the accuracy is slightly worse compared to other z settings, especially for some datasets (i.e., Adiac, Swedish Leaf). With w_{min} fixed, a smaller value for z reduces d (because $d = \left\lfloor \frac{z \times T}{w_{min}} \right\rfloor$) and subsequences are represented by fewer features. Because we generate random length subsequences ($l_s \in [z \times T, T]$), longer subsequences are characterized by longer intervals where the level of detail provided by the features is reduced. This can affect the performance for certain datasets for which the level of detail is important.

On the other hand, an upper bound on the random subsequence length can help to improve the accuracy (i.e., generate subsequences of length $l_s \in [z \times T, u \times T]$ where u is the bounding factor). Because d is fixed (based on w_{min} and z), a shorter subsequence produces shorter intervals. The performance is provided for $u = 0.25$ in Table 7 on the datasets for which TSBF ($z = 0.1$) performs worse than the others. We generate the same number of subsequences as in original case. We again report the average error rate over 10 replications. Both OOB error rates on the training data and error rates on the test data are shown in the table. The results are improved when an upper bound of $0.25 \times T$ is introduced on the maximum subsequence length.

The OOB error rates are based on the training data only. Consequently, after an analysis of OOB error rates for $RFts$ for certain settings, the setting providing the best error rate can

be determined. This idea is similar to searching for best window of DTW on the training data. Table 7 illustrates that OOB error rates are consistent with the error rates on the test data (which is consistent with our claims). For the results reported in Table 5 we did not search for best level of any parameters. Better accuracy can potentially be achieved through analysis of OOB error rates with respect to different parameter settings (z, w_{min}, etc.), and this can be conducted for each dataset.

	$u = 0.25$		$u = 1$ (original)	
	Test error	OOB error	Test error	OOB error
Adiac	0.358	0.378	0.416	0.423
Swedish Leaf	0.136	0.126	0.173	0.180

TABLE 7. Test and OOB error rates for different settings of maximum subsequence length. Originally we do not have an upper bound on the subsequence length. An upper bound of $u \times T$ is introduced. The test and OOB error rates improve when $u = 0.25$ (in a similar manner) for the datasets here.

The number of bins b is set to 10 in our experiments. This parameter is expected to have a small effect on performance, if it is set large enough, because of the embedded features selection in RFs. For example, we again illustrate the use of OOB error estimates to find an appropriate level for this parameter. More bins provides more detailed information about the class probability estimates and results in a larger codebook. Figure 23 provides OOB error rates of TSBF ($z = 0.5$) with $b = 10$ and with $b = 50$. The results with $b = 10$ are slightly better than with $b = 50$. We do not provide the OOB error rates for other z settings, but the behavior is similar to TSBF ($z = 0.5$) for different settings of b.

6.2. *Shapelets and TSBF*

Shapelets are defined as the time series subsequences which are highly likely to represent a class [61]. Also, [23] extended the approach to multiple shapelets with certain rules

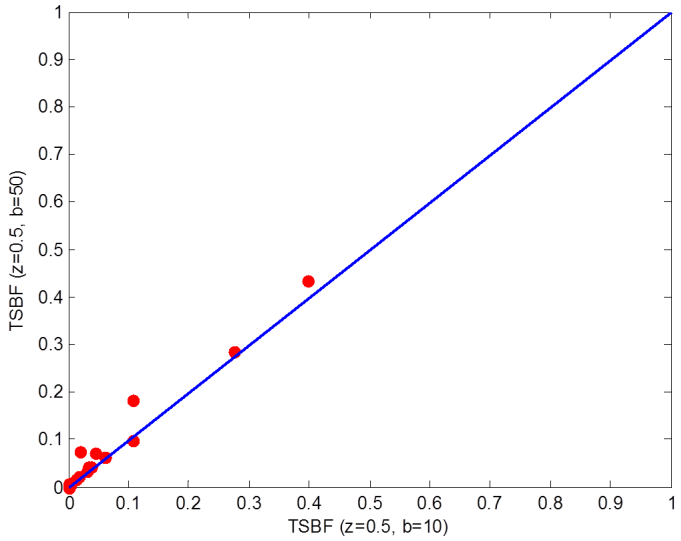


Figure 23. OOB error rates of TSBF ($z = 0.5$) with $b = 10$ and with $b = 50$. The results with $b = 10$ are slightly better than with $b = 50$.

to represent classes. Although shapelet methods are distance-based methods and ours is a feature-based one, both exploit local patterns related to the classes. Logical-Shapelets tries to find subsequences that express class relations based on caching and reuse of computations, and pruning of the search space [23]. Our algorithm generates subsequences from the time series and evaluates them based on the information gain using a supervised classifier on the features. Instead of trying to find rules from $RFsub$, we make use of the summarized version of this information which is the class assignments of the subsequences. The prediction results are used to determine efficient representations for the time series through the BoF idea.

We compare the performance of TSBF to Logical-Shapelets for certain datasets. In order to be fair in terms of comparison, we set the parameters of the logical shapelet algorithm so that it searches for all possible shapelets. However, because of the computational requirements of this algorithm, we could not achieve this for certain datasets. There-

fore, we perform this comparison based on a subset of the datasets: Beef, CBF, Coffee, ECG and Trace. Moreover, we tested our algorithm on three additional datasets discussed in [23]. These datasets are Cricket, Sony AIBO Robot and Passgraphs. Three parameters of Logical-Shapelets are the maximum and minimum length of the shapelet and the step size. We set the maximum to the series length, the minimum as two,and the step size to one. Furthermore, we do not tune the parameters of TSBF for the new datasets; we use the same settings as previously. We also do not compare the algorithms in terms of computation time because the comparison depends to a large extent on parameter settings. The results are provided in Table 8.

	TSBF					NNDTW	
	$z = 0.1$	$z = 0.25$	$z = 0.5$	$z = 0.75$	Logical-Shapelets	BestWin	NoWin
Beef	0.333	0.350	0.307	0.223	0.600	0.467	0.500
CBF	0.001	0.005	0.008	0.016	0.336	0.004	0.003
Coffee	0.054	0.003	0.000	0.004	0.071	0.179	0.179
ECG	0.183	0.138	0.155	0.145	0.140	0.120	0.230
Trace	0.013	0.020	0.020	0.02	0.530	0.010	0.000
Sony A.R.	0.250	0.178	0.135	0.175	0.041	0.305	0.275
Cricket	0.020	0.026	0.041	0.040	0.010	0.051	0.010
Passgraphs	0.301	0.322	0.293	0.253	0.298	0.260	0.282

TABLE 8. Error rates of Logical-Shapelets and TSBF on 8 datasets. TSBF has better or comparable performance on the datasets except for Sony AIBO Robot.

TSBF has better or comparable performance on the datasets except for Sony AIBO Robot (and TSBF is still better than NNDTWBestWin and NNDTWNoWin on this dataset). Recall that the parameters of Logical-Shapelets are set so that it searches over the entire space which increases the computational time significantly. Potentially equivalent accuracy can be obtained with alternative settings on the parameters, but our objective here is to assess the accuracy. Also, we do not provide the time for testing because both algorithms are very fast in classification. Shapelets facilitate some interpretability. Still, the information

provided by RFs such as variable importance, proximity, etc., can be used to improve the interpretability of TSBF [57].

7. Conclusions

A framework is presented to learn a bag of features representation for time series classification. Subsequences extracted from random locations and of random lengths provides a method to handle the time warping of patterns in a feature-based approach. Furthermore, the partition into intervals allows one to detect patterns represented by a series of measurements over shorter time segments. The supervised codebook allows one to integrate additional information (such as subsequence locations) through a fast, efficient learner that handles mixed data types, different units, etc. TSBF provides a comprehensive representation that handles both global and local features. The flexible bag of features representation allows for the use of any supervised learner for classification. Our experimental results shows that TSBF gives better results than competitive methods on the benchmark datasets from UCR time series database [76]. Although our focus in this study is on the classification of the time series, the bag of features approach can be adjusted to other applications such as similarity analysis, clustering, and so forth.

CHAPTER 4

SUPERVISED TIME SERIES PATTERN DISCOVERY THROUGH LOCAL IMPORTANCE

1. Abstract

Similarity search and classification on time series databases has received great interest over the past decade. Nearest neighbor (NN) classifiers with an appropriate distance measure are widely used to solve this problem. Dynamic Time Warping (DTW) distance provides accurate results but its performance degrades with long time series, relatively short features of interest, and moderate noise. The space and computational requirements are problems of NN classifiers for the applications in which the resources are limited. In many time series classification problems, the question is basically about the reason why a time series is assigned to a certain class. NN classifiers lack the aspect of interpretability since they are based on the similarity of the whole time series although temporal relations within the time series are important.

In this work, we present an exploratory approach that finds the regions of the time series that have potentially representative patterns to be used for classification based on a local importance measure. We address the limitations of nearest neighbor classifiers through sampling the patterns from these regions. The distances of time series segments to the selected patterns from the interesting regions are used as features to a random forest classifier. We compare our classifier to well-known nearest-neighbor classifiers, with dynamic time warping distance measures. Experimental results show that our algorithm provides comparable and interpretable results than competitive methods on the benchmark data sets from the UCR time series database.

Key words: supervised learning, time series, classification

2. Introduction

Time series data mining is an important task with many challenging applications including finance, science, medicine and multimedia. Effective and efficient data mining methods are required for the knowledge extraction from time series databases since analysis and modeling of time series data can be time consuming due to its high dimension. Classification is the primary goal in many of the applications. For example, a cardiologist might be interested in analysis of ECG signals from different patients in order to see whether a particular type of patients has a different temporal pattern in their heart signals than a control group [6]. Seismologist aim at discriminating the nature of the seismic waves to classify events such as earthquakes, mining explosions or nuclear explosions [7].

The algorithms proposed for time series classification can be divided into instance-based and feature-based methods. Instance-based classifiers predict a test instance based on its similarity to the training instances. For time series, one-nearest-neighbor (NN) classifiers with Euclidean (NNEuclidean) or a dynamic time warping distance (NNDTW) have been widely, and successfully used [15–19]. DTW [20] is a method that allows a measure of the similarity independent of certain non-linear variations in the time dimension, and is considered as a strong solution for time series problems [58].

Feature-based approaches work on the feature vectors extracted from a set of instances. [64] used knots from a piecewise linear approximation of the time series to detect patterns and classify the time series. [65] proposed an automated approach for feature extraction using a genetic algorithm, then the extracted features were taken as inputs to a support vector machine (SVM) [66]. [68] used intervals of time series to extract features on which a SVM was trained. [80] proposes an efficient multivariate decision tree approach which

selects the interval features by fitting Fused-Lasso logistic regression models [81] at each tree node.

NN classifiers with appropriate distance measures are known to be accurate and robust methods [21, 22] although their space and time requirements may be problem depending on the application. NN classifiers are easy to understand and do not require setting of many parameters, but they typically do not provide insight into time series features important to the classifer. On the other hand, feature-based approaches are interpretable and generally faster than instance-based classifiers depending on the feature extraction method and classification algorithm. Two types of features are generated in these approaches, global and local features. Global features are a compact representation of the instances (such as the mean value) and not sufficiently expressive for classification. Therefore, local features are extracted from segments of the time series to obtain a detailed representation. However, the set of local features may vary in cardinality and lack a meaningful ordering. These are basic problems for many classification algorithms requiring feature vectors of fixed dimension.

Methods based on features of intervals (segments) (such as [69, 70]) assume that patterns exist in the same time interval over the instances, but a pattern that defines a certain class may exist anywhere in time. We illustrate this problem on a synthetic dataset illustrated in Figure 24. There are time series from two classes in this dataset and the location of the peak determines the class label. Class 0 has the peak in at a random location between time 0 and 100 where class 1 has the peak between 100-200. An interval-feature based classifier can determine the peaks and classify well in the training data, but it may fail for a test instance which has a peak at a different location (i.e. a location where training time series has no

peaks) when a feature vector of fixed dimension is used. They cannot handle the invariance in terms of location.

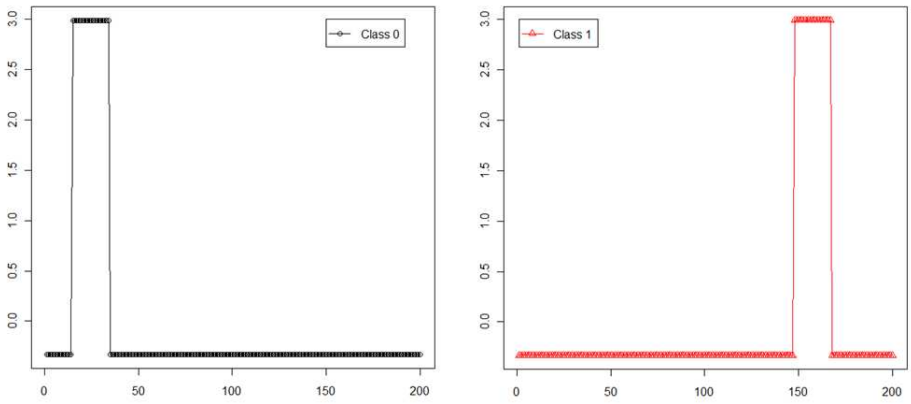


Figure 24. Two sample time series from different classes. If the peak is in the first half of the time series, the series is labeled as class zero and it is labeled as class one otherwise.

Although DTW attempts to compensate for possible time translations and dilations between features, the capability for DTW is degraded with long time series, relatively short features of interest, and moderate noise. Moreover understanding what exactly relates to the class is not trivial task. On the other hand, feature based approaches can be interpretable but they have certain problems with location invariance. Consequently, an important research task is to identify the regions (segments) of time series useful to the classifier that can occur at different times in different time series instance and make classification based on these segments. [82] proposes a bag-of-features approach (TSBF) to handle the possible time translations and dilations between the features. Although the classification performance of TSBF is good, further analysis of the prediction models are required to identify the important features (regions) for classification.

Our work is based on finding the segments of the time series that have potential to describe a class. These segments are referred as the region of interest. We make use of the

structure of a supervised feature-based learner to identify the region of interest in our study. Region of interests are very important to understand the temporal relations. Moreover they help to reduce the effort in searching for the time segments related to the classification task. [23] also discusses the necessity of pruning the space of the potential segments and proposes a distance based method. Feature-based approaches allow for some desirable properties such as handling the interactions and fast computation. Interaction between the features in this context is the relationship of the patterns over multiple intervals that may define a class as discussed by [23].

Considering the strength of the feature based approaches, we train a classification algorithm on an interval feature representation to find the regions of the time series that are informative. We segment the time series using overlapping intervals to reduce the probability of missing a pattern and generate features on the intervals. We build a classifier on the interval representation and compute a local importance measure for each interval of each time series. Local importance [57] is a measure which is related to the effect of a feature for predicting an outcome of interest. In time series context, local importance of a certain interval feature for a particular time series provides information about the relevance of the pattern observed on the corresponding interval to the classification task. Once the local importances are identified, the similarity between the time series can be sought over the important patterns instead of the whole time series. Since only relevant segments of the time series are considered for the classification, the result will be less affected by the noise and the computation and storage requirements can be reduced significantly with the shorter representation. More significantly, this type of representation will handle the translations and dilations inherent in the time series.

Local importance computation is the key step in our approach. This information is obtained from a fast feature-based learner which allows for finding regions of the time series relevant to classification. The patterns in the intervals with features having high local importance values constitute the region of interest. After finding the region of interests for each time series, we generate sequences from these regions. These sequences are referred as patterns in our study. We generate multiple patterns from the time series and find the best matching subsequences of the time series to these patterns based on a distance measure. A new feature set based on the distances of the patterns to the best matching subsequences of the time series is used to build another classifier for final classification. A feature selection algorithm on the new feature set allows for finding the patterns that are critical in classification. In addition to their interpretability, patterns are compact compared to the whole time series which reduces the time and space required for classification [23].

Focusing on the smaller segments of the time series for classification is an active research area. Recently, many of the work has focused on the extraction of interpretable patterns for classification of large time series databases [23,61,83,84]. Criticizing the disadvantages of NN classifiers in terms of computational requirements and interpretability, [60] proposed a method that searchs for the best subsequence in an exhaustive way for decision-tree induction. However, at each split they computed and used the DTW distance of the entire time series to the subsequences instead of computing the distance of the subsequence to the related region of the time series. [61] also proposed an approach to find subsequences of the time series which are thought to be maximally representative of a class and compared the subsequences to the relevant regions of the time series unlike [60]. These subsequences are called shapelets and algorithms based on the shapelets facilitate interpretability. Since

the information provided by time series shapelets is limited to their presence or absence and computation time required for generating them is significant, [23] proposes a more expressive shapelet representation by combining multiple shapelets in logic expressions such that complex concepts can be described. It is faster compared to shapelets and has better accuracy since it can combine multiple shapelets for classification of the time series. [84] proposes a similar approach for early classification of time series. The goal is to find the time segments that achieves a certain level of classification accuracy as early as possible. These approaches search for the predictive regions of the time series through efficient representations and search techniques.

The closest works in terms of overall approach are [23, 60, 61]. These studies aim at finding the representative subsequences of the time series to be used in decision trees. However [60] exhaustively search for the subsequences, [23,61] proposes pruning techniques for finding the shapelets where we propose an efficient feature-based method to discover the region of interest. After finding the subsequences, [60] compute the distances to the whole time series using DTW but we find the distances based on the best matching subsequences from the time series as in [23, 61]. Our shapelet representation scheme allows for handling the interaction that might be important to classification. [23] also discusses the necessity of accounting for the interaction and an approach that combines the shapelets through logic expressions are proposed in their study. Our approach uses a random forest classifier in which interpretability is achieved through the generation of an importance measure using the structure of the classifier.

In this paper, we propose a supervised Time Series Pattern Discovery algorithm (TSPD). A large number of local features are extracted from intervals. Subsequently, a local

importance measure is generated for each interval of the time series using a random forest classifier. After regions of interests are identified for each time series using the local importance values, potential shapelets are generated. Each time series is then represented by their distances to the potential shapelets and a new feature matrix of distances is used for classification. We demonstrate TS-PD is efficient, accurate and interpretable on a full set of benchmark data sets [85].

The remainder of this paper is organized as follows. Section 3 provides background. We summarize the problem and describe the TS-PD framework in Section 4. Section 5 demonstrates the effectiveness and efficiency of TS-PD by testing on a full set of benchmark datasets from UCR time series database [85]. We discuss TS-PD's behaviour for certain datasets, explain how TS-PD works on an example in Section 6. Conclusions are drawn in Section 7.

3. Background

3.1. *Random Forest*

A random forest (RF) classifier [57] is used here to both generate the regions of interest and classify time series. A RF is an ensemble of J decision trees, $\{g_j, j = 1, 2, \ldots, J\}$. Each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the single tree. These are called out-of-bag (OOB) samples.

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 estimated 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

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

where $I(\cdot)$ is an indicator function that equals one if its argument is true and zero otherwise. The predicted class is $\hat{y}(x) = \mathrm{argmax}_c\, p^c(x)$.

To summarize, an instance is labeled through a majority voting approach using the tree results for which it is OOB. The estimates computed from OOB predictions are easily obtained and have been shown to be good estimates of generalization error [57].

In the tree growing steps of RF, the best split are determined based on only a random sample of features. In this study, features are also referred as variables and both terms are used interchangeably. 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 instances). Therefore, for a large number of features a RF can be as computationally efficient as a single decision tree.

The Gini measure of impurity is used to determine the variable selected to make the nodal split in the tree construction process. This allows for a variable importance measure called Gini Variable Importance (GVI) which is the sum of the Gini impurity decrease for a particular variable over all trees. Let N_j^ρ be the number of observations at node ρ of the j^{th} tree, and $N_j^\rho(L)$ and $N_j^\rho(R)$ be the number of observations of the left and right child nodes after splitting, and let $d_j^\rho(k)$ be the decrease in impurity produced by variable k at the ρ^{th} node of the j^{th} tree.

The decrease in impurity is $d_j^\rho(k) = G_j^\rho - (\frac{N_j^\rho(L)}{N_j^\rho} G_j^\rho(L) + \frac{N_j^\rho(R)}{N_j^\rho} G_j^\rho(R))$ where $G_j^\rho(L)$ and $G_j^\rho(R)$ are the Gini indices of the left and right node respectively and G_j^ρ is the Gini index of the parent node. The Gini Variable importance of variable k is defined as

$$GVI(k) = \frac{1}{J} \sum_{j=1}^{J} (\sum_{\rho \in S_j} d_j^\rho(k) I_j^\rho(k))$$

where $I_j^\rho(k)$ is an indicator variable for whether variable k was used to split node ρ of tree j and S_j is the set of split nodes of the tree j.

Variable importance is important to find out the features relevant to the classification task. We use a RF classifier for time series classification in our study. Our time series representation scheme allows for finding the important patterns efficiently using the variable importance. Moreover, RF has several advantages when compared to other classifiers. High dimensional feature spaces, multiple classes, and missing values are handled. Nonlinear models and interactions between features are allowed. It is scale invariant and robust to outliers, and computations are reasonable even for large datasets.

3.2. *Local importance measure from random forests*

A random forest classifier is not directly interpretable since it is a combination of multiple unpruned trees build on the random subspaces of the features. However there are important measures that can be derived from the forest structure such as feature importance discussed in Section 3.1. Other than the Gini variable importance, an accuracy based feature importance is also discussed by [57]. To compute this feature importance, local importance of a feature is computed for each instance based on the changes in accuracy of the classifier

when the features are perturbed. This information is then aggregated to obtain the accuracy based feature importance in [57].

RF local importance for feature k of instance n, $LI_k(n)$, is defined as follows. For each tree g_j of the forest, consider the associated OOB sample represented by $OOB(g_j)$ (instances not included in the bootstrap sample used to train g_j) and let the proportion of votes for the correct class be v_n for instance n based on the trees in which instance n is OOB. Now, randomly permute the values of the feature k in $OOB(g_j)$ to get a perturbed sample denoted by $\tilde{OOB_k}(g_j)$ and prediction based on the perturbed sample provides a new proportion of votes $\tilde{v_n^k}$ for instance n. Local importance for feature k of instance n is then equal to $LI_k(n) = v_n - \tilde{v_n^k}$.

If the number of votes for the correct class decreases with the perturbed OOB data for particular feature of an instance, we can say that feature plays an important role in the classification of the instance in consideration. Conversely, if the number of votes does not change or increases, the feature is not found to be informative.

A global feature importance is computed by aggregating the local importances over all instances by [57]. However, local importance is a better description of the patterns of the time series that can be related to the classification. A global feature importance is not descriptive enough because of the translations and dilations in the time series. Features generated over different regions of the time series may be important to classification for different time series and a global feature importance loses the detailed information about the translations and dilations. Although the local importance information is provided by [86] as a visualization tool, their focus is on generating a global feature importance by aggregating the local importance information. On the other hand, analysis of the local importance is

required for the time series classification problems because of the temporal ordering of the features.

3.3. *Tree models with interval features*

In tree-based models developed for time series such as [70], features (such as mean, deviation, etc.) are extracted for the intervals segmented from the time series. The intervals and features are selected in a way that the splitting criterion is maximized when the collection of time series is partitioned into child nodes. A typical example of a rule for interval split in a node for a time series is $variance(I\,[t_1, t_2]) \leq threshold$, where the notation indicates that a series for which the variance over the interval $[t_1, t_2]$ is less than or equal to a threshold is assigned to the left child, and assigned to the right child otherwise.

Segmentation of the time series and feature extraction requires sampling representative set of intervals from the time series. The focus should be on the segments and features that are the most informative for classification. Piecewise linear approximation is the most commonly used preprocessing step for the discretization of the data in mining time series databases [78]. Time series approximation is an active research topic and a comprehensive literature review of time series segmentation approaches is provided by [8, 78]. How segmentation affects the performance of the tree-based models is discussed in the following paragraph on a simple example.

We illustrate the approach for building a tree-based model on CBF dataset from [85] given in Figure 25 to discuss the strengths and weaknesses of tree based models built on interval features. We segment the time series using a fixed size intervals and generate the mean, variance and slope features for each interval. Here, slope is computed by fitting

a regression line using the data points of the interval. Although fixed-length interval for segmentation is prone to omit certain patterns [8], it is easy to understand.

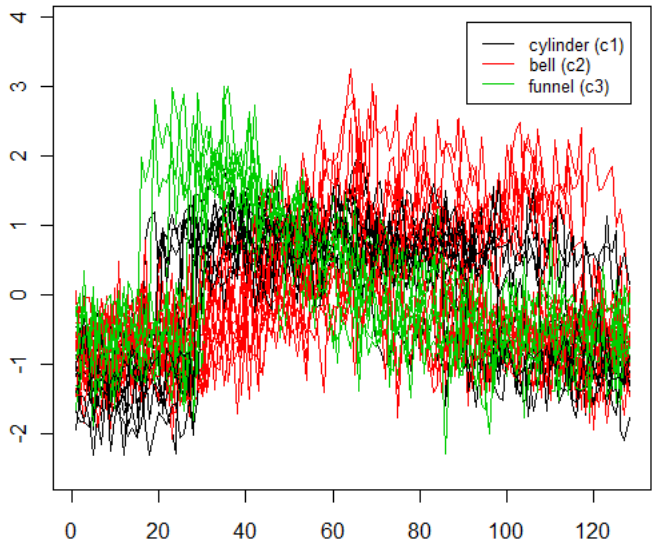


Figure 25. 30 training instances from CBF dataset. Cylinder, Bell and Funnel are labeled as 'c1,c2 and c3' and represented by 'black, red and green' lines respectively.

The length of the CBF dataset is 128 and we start the analysis with intervals of size 5 time units. This makes $\lfloor \frac{128}{5} \rfloor = 25$ intervals plus the last interval with three observations $(128 - 125)$. Thus, $26 \times 3 = 108$ features are extracted for each time series. We also generate second set of features by setting the interval size to 10 time units in order to illustrate the effect of the choice of interval length parameter. Two decision trees built using C4.5 [4] are illustrated in Figure 26. Each node in the tree represents a specific feature and interval. For example, the first node of (a), $interval(8)\,[36, 40]\,\_mean$, is the mean of the data points between 36 and 40 where 8 represents the interval id.

As discussed earlier, comprehensibility of a classifier is highly important in this domain. Trees provide set of rules that leads to a classification as given in Figure 26. The important

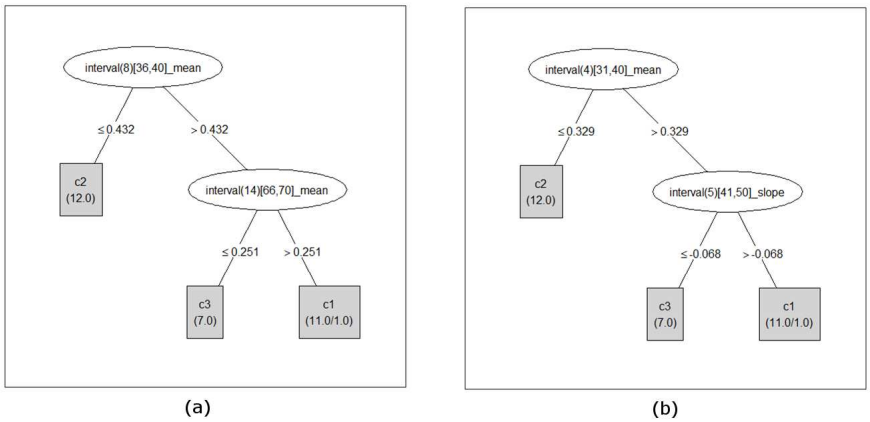


Figure 26. Decision trees built using C4.5 [4] on the interval features. Left (a) is the tree built on the intervals of 5 time units. Right (b) is the tree built on the intervals of length 10. Each node represents a specific feature and interval. For example, $interval(8)[36, 40]\_mean$ is the mean of the data points between 36 and 40 where 8 represents the interval id.

time intervals found to be important are not the same for both trees if the splits are considered. This simple example illustrates that there might exist multiple regions that are related to classification and there is a possibility of missing certain regions because of the feature generation scheme. Moreover as discussed by [84], extracted features do not stay in the same data space of the input data therefore it may not be easy to understand the information provided by the features. In other words, the information provided by the raw data points is lost by the transformation to the feature space. This is one of the motivations of the time series classification studies based on the distances instead of features [23, 60, 61]. A similarity based approach based on the raw values of subsequences are claimed to be more intuitive since there is no transformation of the data in feature extraction.

Another problem discussed in Section 2 is the location invariance. Trees built on interval features generate the rules based on fixed locations of the training data however patterns defining a class may shift on a test instance. A test instance from the cyclinder class may

have longer tails and have a small mean on the intervals between 31 and 40, this instance will be classified as bell (c2) by both trees. Although trees on interval features provide interpretable results, they may fail in classification because of the location invariance inherited in the time series. Please note that the example tries to illustrate potential problems of the feature based approaches in terms of comprehensibility and it does not consider different sampling strategies or alternative feature definitions.

3.4. *Shapelets*

Shapelets are defined as the time series subsequences which are highly likely to represent a class. Instance based approaches require comparison to the entire dataset which is a problem in terms of space and computational requirements in resource limited systems such as sensor nodes, cell phones, mobile robots, smart toys, etc [23]. Shapelets are shorter and finding the distance to the shapelet is faster compared to computing the distance to the whole time series. Classification algorithms based on shapelets are also interpretable.

The main idea is that there exist local patterns related to the classes in a time series classification problem. Extracting the relevant part of the time series is important since NN neighbor classifiers account for the entire time series and they are prone to misclassification because of the curse of dimensionality. An example from [61] from Gun-Point dataset is provided in Figure 27. The aim is to classify a motion as ”Gun” or ”NoGun” through time series generated by mapping the motions as in Figure 27. The shapelet found for ”NoGun” class is represented by red line which desribes a certain phenomenon called overshoot [61].

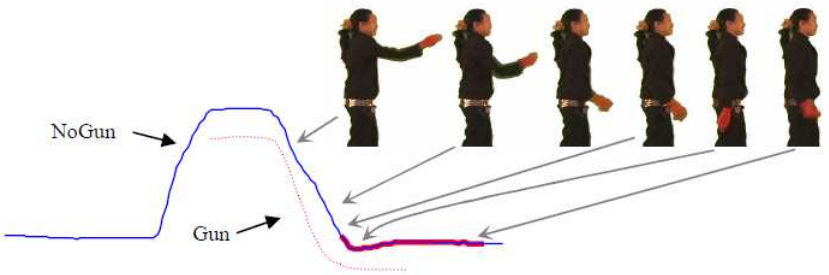


Figure 27. Illustration of the classes for Gun-Point dataset. A ”dip” is observed for NoGun class since the actor put her hand down by her side, and inertia carries her hand a little too far and then she tries to correct for it (a phenomenon known as overshoot). On the other hand, actor returns her hand to her side carefully when she has the gun and no dip is seen [61]. Shapelet discovered for ”NoGun” class is given by red line.

4. Supervised Time Series Pattern Discovery through Local Importance

We propose a method to discover the regions of the time series that has potential to have information about the classes. This discovery relies on the results of a random forest classifier built on interval features. A local importance measure is computed for the interval features of each time series. Consequently, potential patterns from the time series are sampled based on the local importance of the intervals. We then find the best matching subsequences of each time series to each pattern using a distance measure and generate a new feature set based on the distances between best matching subsequences and the patterns. A RF classifier is built on the new feature set to find the labels of the time series. Then RF is used to find the patterns that are important for classification.

4.1. *Region of Interest Selection based on Local Importance*

We represent each time series with feature vectors derived from intervals to capture patterns along the time series. A fixed-length interval for segmentation has the potential to omit patterns because they may appear with different lengths and be split across the time

points [8]. We slide the windows to extract overlapping subsequences from the time series to reduce the opportunity to miss patterns. Before going into details of the algorithm, we define notations used in the paper.

Definition 1. A univariate time series, $x^n = (x_1^n, x_2^n, \ldots, x_T^n)$ is an ordered set of T values. We assume time series are measured at equally-spaced time points indexed by t. Each time series is associated with a class label y^n, for $n = 1, 2, \ldots, N$ and $y^n \in \{0, 1, 2, ..., C-1\}$.

Definition 2. An interval of the time series x^n, $I_p(x^n)$, is a sampling of length $w < T$ of contiguous positions from x^n starting at position p. Thus, $I_p(x^n) = (x_p^n, \ldots, x_{p+w-1}^n)$ for $1 \leq p \leq T - w + 1$

Definition 3. A sliding step of size $d < w$ is used to segment overlapping intervals from x^n. Let $I_p(x^n)$ be the interval of length w which starts at position p. A representative set of intervals of length w can be extracted by sliding $d < w$ positions from p across x^n. The set of the representative intervals of length w across x^n is then $\{I_1(x^n), I_{1+d}(x^n), \ldots, I_{1+T-w}(x^n)\}$. Setting $d = 1$ generates all possible intervals of length w. We aim at generating overlapping intervals to avoid missing a pattern, therefore $d < w$ is preferred.

Given the time series x^n of length T and an interval length w, intervals are segmented using a sliding step of d across T. This segmentation is illustrated in Figure 28 for one of the instances in CBF dataset ($M = 128, w = 20, d = 10$). Linear regression models are fit on the intervals to extract features. The following features are extracted for each interval: slope of the fitted regression line, mean of the values, variance of the values. More features

can be extracted to include detailed information about the intervals but three features which gives information about the level and shape of the interval is used in this study.

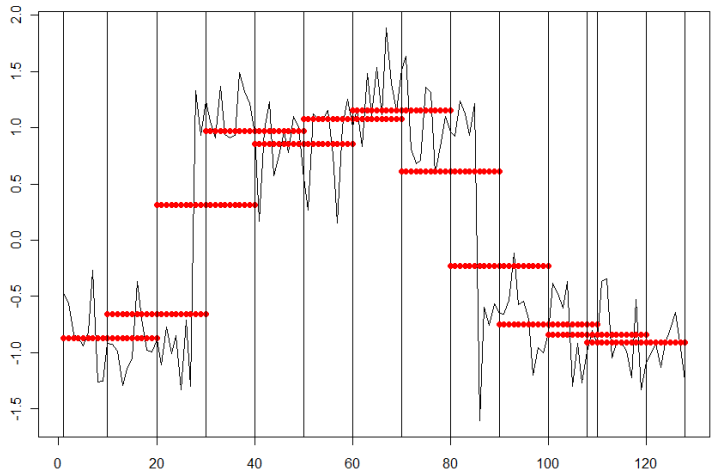


Figure 28. Illustration of feature generation on the intervals of one time series from CBF dataset. The parameters are set as $w = 20, d = 10$. 12 intervals and their means are given.

A random forest classifier, $RFint$, is trained on the interval feature representation and local importance of each interval is computed during the training of $RFint$ as described in Section 3.2. The algorithm for computing the local importance of each interval is provided in Algorithm 3. Since each interval may be described by multiple features (i.e. slope, mean, etc.), we set the interval importance as the maximum of the local importances of the features. Intuitively, if there is at least one relatively important feature observed for the interval, the importance of the interval is set based on the most important interval feature. In order to visualize the local importance, we normalize the local importance values so that importance of the intervals of a time series sums up to one. The intervals with high importance values have potential to contain the patterns related to class.

Local importance values are computed for all instances and instead of starting the search for the patterns from an arbitrary time point as in other shapelet studies [23, 60, 61, 84],

Algorithm 3 Local importance computation

for all time series x^n **do**
  Standardize the time series
  Generate the interval features
end for
Build a random forest on the interval feature set $(RFint)$
for all time series x^n **do**
  for all interval i **do**
    Let local importance of interval i be $\max_{f \in F(i)} (LI_f(x^n))$ where $F(i)$ is the set of features of interval i
  end for
  Normalize local importance over all intervals i of time series x^n
end for

we use the region of the time series that are found to be important. Unlike the existing work which finds the important regions based on the similarity, we generate features on the intervals and use a supervised learner to find out those regions. Approximating the information by feature extraction from intervals and using a supervised learner that allows for the interactions provides fast discovery of the important regions of the time series.

We illustrate the idea of local importance using time series of each class from CBF in Figure 29 ($M = 128, w = 10, d = 10$). We do not use overlapping intervals (i.e. $d = w$) in this particular example in order to simplify the representation. In a random forest, 500 trees are built and selected number of features at each split is square root of the number of features unless otherwise stated for the illustrations. Three intervals with the highest local importance values are represented on the right for each time series. The local importance information matches with class definitions and these local regions can separate these time series which supports our idea of focusing on the local regions.

Interval length setting is a smoothing parameter in our algorithm because of the features considered for each interval. The level of detail decreases as the interval length increases

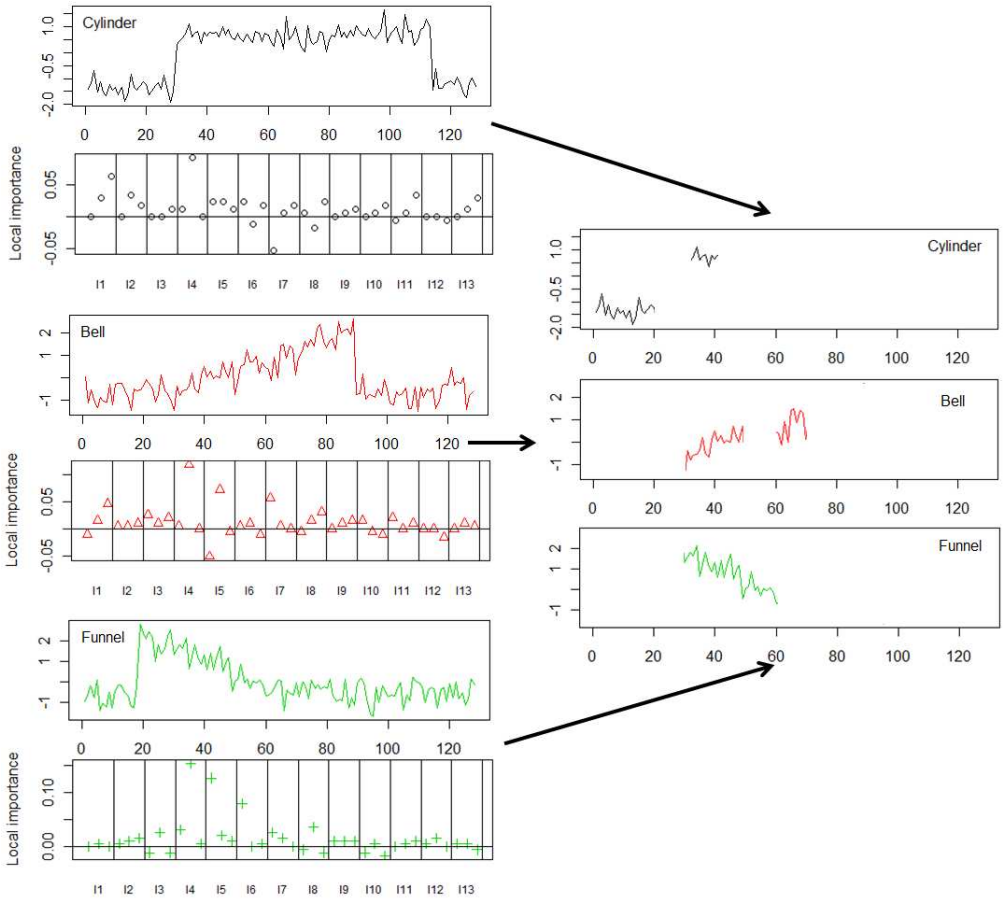


Figure 29. Three time series from CBF dataset and corresponding local importance plot. The intervals are labeled as I(interval id). For each interval, there are three measures representing the local importance of slope, mean and variance features in the order from left to right. Three intervals with the highest local importance values are represented on the right.

(i.e. slope becomes meaningless) therefore smaller interval lengths should be preferred for an application where features of interest are short. Assuming that there is no information provided about the application, this parameter is set based on the analysis of OOB error rates from $RFint$. The interval lengths providing smaller OOB error rates should be preferred for the analysis. If the interval length is set too small for the case where feature of interest is long, multiple short intervals will be found to be important.

The normalized importance values against the intervals is illustrated in Figure 30 for each class on CBF dataset ($w = 10$, $d = 5$). The regions with high importance values have potential to contain the patterns related to class. The time frame between 20 and 80 where the mean differences are found to be important for most of the training instances. The regions in the beginning and end are also found to be important for certain instances of class cylinder which matches with the class definitions.

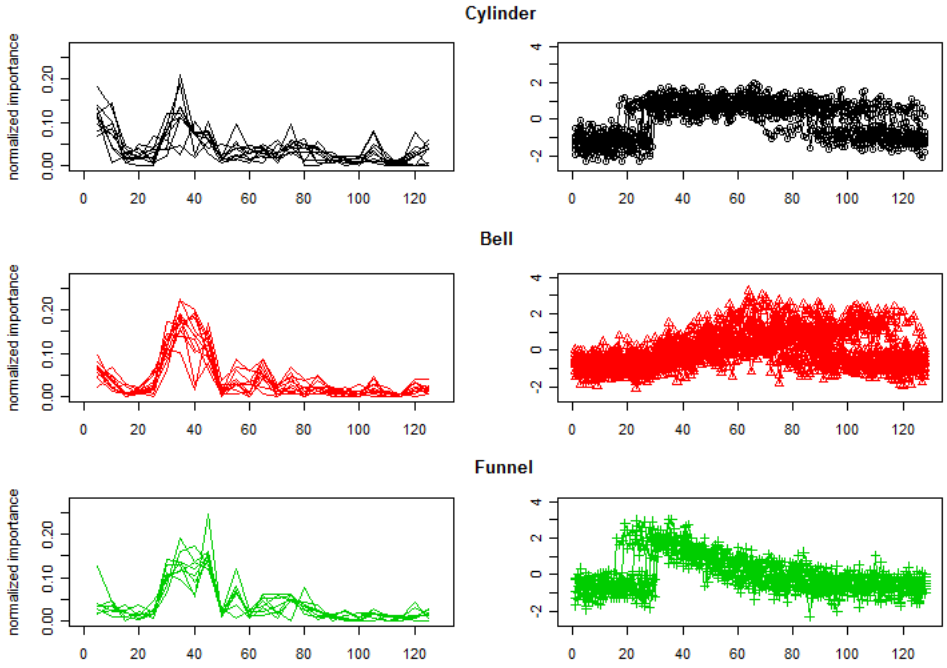


Figure 30. Normalized local importance information on CBF dataset (left) and time series of each class (right). The regions with high importance are informative when time series are compared and the class definitions are considered. The parameters are set as $w = 10, d = 5$.

4.2. *Pattern Discovery and Classification*

After finding the regions of interest, important intervals are used to search for similarity between the time series. Intervals are used as reference patterns and distance between the time series and reference patterns are computed. The distance between each pattern and the

time series is later used as a feature in the learning algorithm. Before going into details of the pattern discovery approach, we provide the definition of the terms used in this section.

Definition 4. A pattern of time series x^n, $\Psi^l(x_n)$, is obtained by combining the most l important intervals of x^n. A pattern set $S(x^n)$ consists of the patterns from the time series x^n.

Patterns are generated starting from the interval that has the largest local importance ($\Psi^1(x_n)$). These patterns are referred as *level 1 patterns*. For each time series, we add these patterns to our pattern set $S(x^n)$. In the second pass, first two important intervals constitutes the pattern which is included in set $S(x^n)$ as *level 2 patterns*. Following the same manner, we generate all patterns up to level L and add them to the pattern set $S(x^n)$. Thus, $S(x^n) = \{\Psi^1(x_n), \Psi^2(x_n), \ldots, \Psi^l(x_n)\}$ where $\Psi^1(x_n) = \{I^1(x^n)\}$, $\Psi^2(x_n) = \{I^1(x^n), I^2(x^n)\}, \ldots, \Psi^l(x_n) = \{I^1(x^n), I^2(x^n), \ldots, I^l(x^n)\}$ where $I^l(x^n)$ is the l^{th} important interval of time series x^n, the interval notation is changed here to represent the importance of an interval. We keep the temporal relation between the intervals while generating the patterns. In other words, pattern may contain discontiguous intervals as illustrated in Figure 31 if the most important intervals of the time series are not contiguous. This way, we keep the information provided by the temporal relations between the intervals which might be important to classification.

Pattern set $S(x^n)$ contains the patterns generated by combining certain number of intervals up to l. Let S be the set of all possible patterns from all time series, (i.e. $S = \bigcup_{n=1}^{N} S(x^n)$) and suppose we enumarate the patterns in set S as Ψ_i where i is the pattern index.

Definition 5. Best Matching Subsequence (BMS) of time series x_n, is the subsequence that has the minimum distance to the pattern Ψ_i. The minimum distance is referred to as **Best Matching Distance (BMD)** given by $D(x_n, \Psi_i)$. $D(x_n, \Psi_i)$ is the minimum of the distances computed by sliding Ψ_i over the time series x_n as schematized in Figure 31. The subsequence providing the minimum distance is called BMS of x_n to Ψ_i. The distance measure considered in this study is the Euclidean distance although other distance measures (i.e. Manhattan) can be used. If a pattern contains discontiguous intervals, we only consider the relevant matching sections of the subsequences for distance computation as illustrated in Figure 31

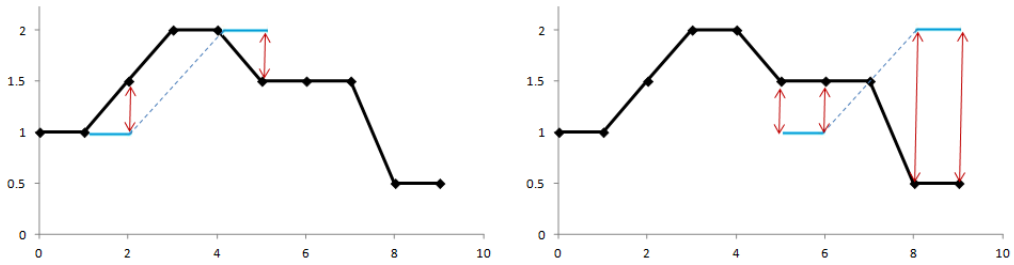


Figure 31. Illustration of distance computation over the time series for a generated patterns (represented by blue). This pattern includes two separated intervals. Dashed lines stand for the regions that are not included in the pattern. The distance is computed by sliding the pattern over the time series.

After constructing the pattern set S, we compute $D(x_n, \Psi_i)$ for all time series and pattern pairs. A feature vector for x^n is then obtained by combining the BMD of x^n to all patterns in set S. A new feature matrix is created using BMD of the time series to the patterns and build a random forest, $RFpattern$, on the new feature set. Our pattern discovery and classification algorithm is summarized in Algorithm 4.

The discontinguous intervals in the patterns allow for handling the interaction between the patterns of the time series. [23] similarly combines shapelets through logical expres-

Algorithm 4 Supervised Time Series Pattern Discovery through Local Importance (TS-PD)

Compute local importance using Algorithm 3 for all time series x^n, set $S = \emptyset$
for all time series x^n **do**
  Generate pattern set, $S(x^n)$, for all levels up to L, set $S = S \cup S(x^n)$
end for
Generate a new feature set using the BMD ($D(x_n, \Psi_i)$) of the time series x^n to each pattern $\Psi_i \in S$ and build a random forest ($RFpattern$) on this representation to obtain final classification.

sions. Although our pattern generation scheme and similarity computation is different compared to [23], the idea of multiple segments' being informative is similar.

One can claim that there is a redundancy in the pattern set of a time series since the same intervals are shared by the patterns (i.e. the most important interval is seen for patterns of all levels). However BMS may be different and higher level patterns may be better in terms of descriptiveness. Although this will increase the computation time required for distance computation, the increase is not significant since higher level patterns already has the intervals of the lower level patterns and distance computation can be done in an incremental manner. In other words, once the distances for the highest level patterns are computed, we also obtain the distances for the lower level patterns.

Similar or same patterns may be generated by our pattern generation scheme since we do not consider the similarity of the patterns in set S. This may result in highly correlated features but RF is robust to correlated features. Although more features add computational complexity to $RFpattern$, the search for the similarities between the patterns is eliminated with the random feature sampling mechanism in RFs. An efficient pruning algorithm to reduce the number of patterns in the set S may improve TS-PD but our algorithm is not severely affected by the correlated features in terms of accuracy since a RF classifier has the embedded feature selection [57].

4.3. *Feature selection and summary using TS-PD*

TS-PD is based on the trees built on the random subspaces of the distances to the patterns. It does not provide a result that is directly interpretable since there are multiple trees and structure of trees are based on the distances to multiple patterns. However, we can use Gini importance from $RFpattern$ to sort the patterns in terms of predictive power. This allows for finding the important patterns for each class which brings the interpretability. However since a large number of patterns could be generated, and some of them might be redundant or ignorable, a further feature selection procedure may be useful.

Feature subset selection methods such as CFS [87], FCBF [88], ACE [89] can be used to select the relevant patterns in our study, and a discussion about these methods is provided in [90]. However, this will add complexity to the algorithm. Therefore we use the variable importance measure of $RFpattern$ for finding the informative patterns. This measure is computed online and the only drawback is that it does not generate a compact set of patterns. We order the patterns based on their information value which may include some redundant patterns. One can select the ones that are thought to be relevant and use the distance values to generate rules on the patterns.

The variable importance computed for $RFpattern$ on CBF dataset is illustrated in Figure 32. The average of the decreases in Gini impurity is provided for the first 20 important distance features. Variable importance can be used to find out the patterns that are important to classification. A compact representation may not be possible since some of the patterns share the same information as discussed. The patterns found to be important for classification is schematized in Figure 33. First 12 important patterns are represented based on their importance values. The first two important pattern from bell class represents the increasing

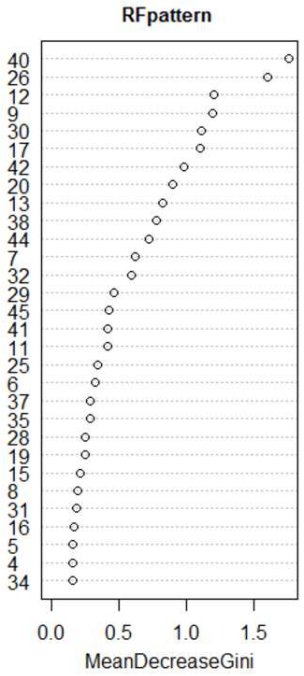


Figure 32. Variable importance of $RFpattern$ based on Gini measure on CBF dataset($w = 10, d = 5, L = 3$). The plot does not illustrate the importance of all features, only first 20 important features are provided. y axis represents the id of the patterns.

time segment. In this particular example the most important pattern is found to be of *level 3 pattern* and it consists of separated intervals. Third pattern (*level 3*) represents the decreasing behavior of the funnel class. 11^{th} pattern (*level 2*) is from the cylinder class capturing the straight segments of the cylinder shape.

The interpretability of TS-PD is achieved through the importance values as given in Figure 33. We provide an ordered list of patterns based on their importance instead of generating a set of patterns used for classification unlike the existing work [23, 60, 61, 84]. All patterns are associated with an importance value which can be interpreted as how well they describe a certain concept.

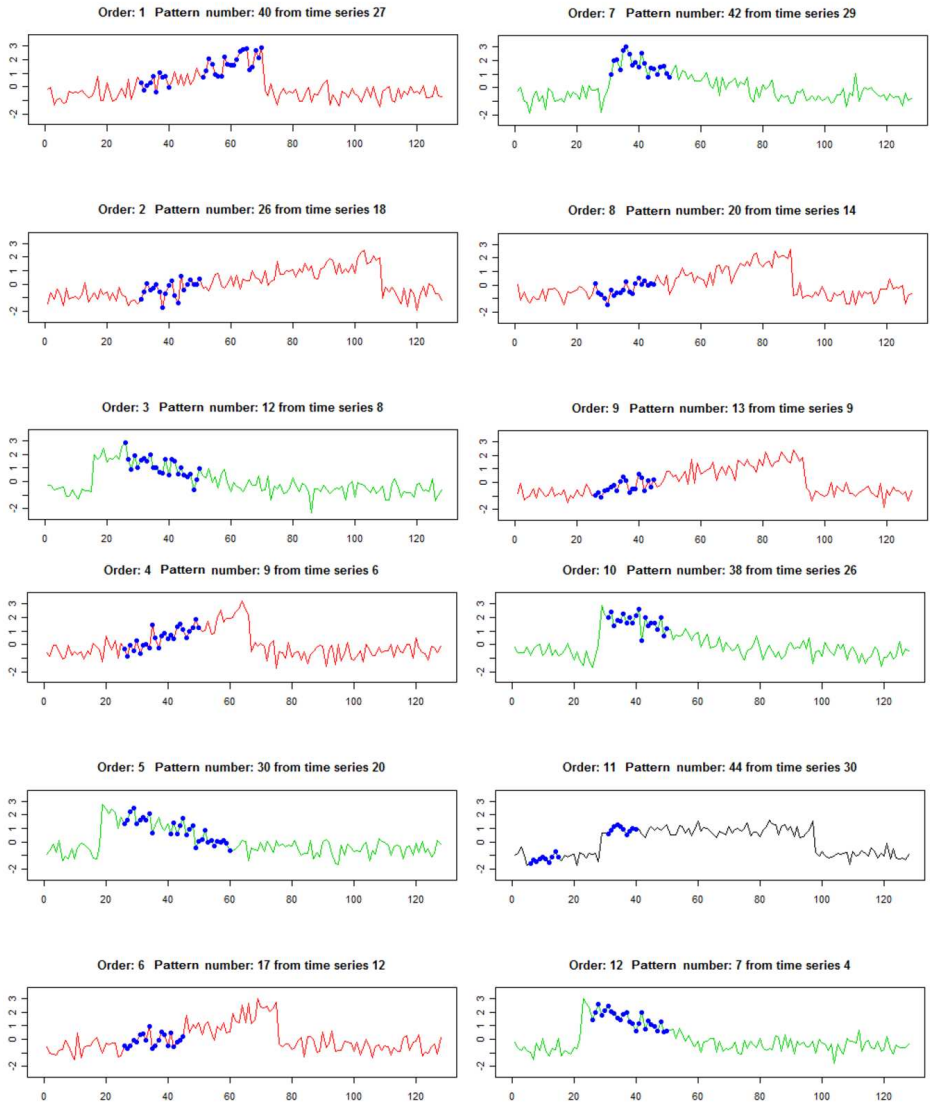


Figure 33. First 12 important patterns of TS-PD for CBF dataset ($w = 10, d = 5, L = 3$) represented by blue dots, the order, id of the pattern and the corresponding time series is provided in the titles of the plots.

4.4. *Parameters of TS-PD*

Although we propose TS-PD as a time series classifier, our algorithm is more of an exploratory tool for time series classification. The parameters should be set based on the preliminary analysis of the time series. We summarize the parameters of our algorithm in

Table 9 categorized with respect to their type and discuss how the parameters should be set in this section.

Random forest	**Feature generation**	**Pattern**
Number of trees	Interval length w	Maximum level L
Number of features in each split	Sliding step d	

TABLE 9. Parameters of TS-PD

The number of features evaluated at each split and the number of trees are the parameters of both RF. The number of features evaluated at each node of the tree is set to default [57] which is equal to the square root of the number of features. As stated by [57], RF is insensitive to the number of features selected to split each node. The number of trees is determined based on the OOB error rates over trees. Figure 34 illustrates how the OOB error rate changes as the number of trees increases for $RFint$ and $RFpattern$ on CBF dataset with the following settings ($w = 10, d = 5, L = 3$). The plots indicate that the results are insensitive for number of trees greater than 400 trees.

Interval and sliding step length are feature generation parameters of $RFint$. Setting sliding step too small will result in correlated features. On the other hand, the probability of missing a pattern increases as sliding step increases. Thus, we fix the sliding step as the half of the interval length. Assuming that the model with the best accuracy provides better local importance results, OOB error rate of $RFint$ is used to set w. This parameter should be large enough so that features like slope and variance are meaningful. Experimentation with different interval lengths will lead to a reasonable setting of this parameter. The change of OOB and test error rates of $RFint$ for CBF dataset with different interval length settings is provided in Table 10 to illustrate how w is set. Setting $w = 16$ provides the minimum OOB error rate thus it is a good choice for interval length. We also provide the test error

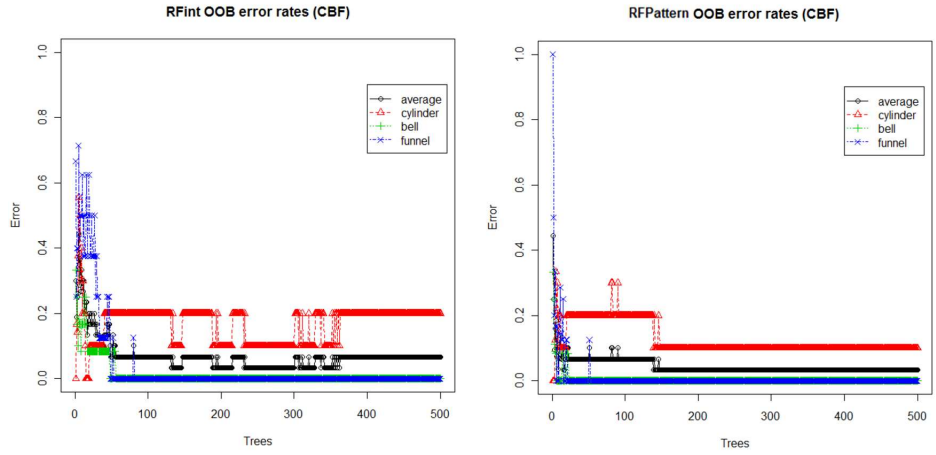


Figure 34. The OOB error rates of $RFint$ (left) and $RFpattern$ (right) of CBF dataset ($w = 10, d = 5, L = 3$). OOB error rate for each class and average of them are provided. The plots indicate that the results are insensitive to the number of trees when it is sufficiently large (500 in this case).

rate for $RFint$ to illustrate that OOB errors are good estimator of the generalization error. On the other hand, the difference of OOB error rates are not significant. If OOB error rates are around the same level for $RFint$ as in the example, setting w smaller is suggested since we are interested in finding shorter patterns so that time required for distance computation will be smaller.

The maximum pattern level setting, L, works as an upper bound on the number of intervals to be included in the pattern. This does not affect the performance of our algorithm if set large enough. However larger L levels result in more patterns to be generated which is not computationally efficient. Although the pattern level is the same for all time series in our approach, it can be set for each class using the corresponding local importance plot. Detailed analysis of the local importance plots may help reducing the testing time. The number of peaks in the local importance plots is a good estimator of the pattern level. Figure 30 illustrates the local importance of each series of different classes for CBF dataset

w	OOB Error Rate	Test Error Rate
6	0.167	0.066
8	0.100	0.044
10	0.067	0.032
12	0.067	0.024
14	0.067	0.027
16	0.018	0.000
18	0.067	0.023
20	0.033	0.028

TABLE 10. The OOB and test error rates of $RFint$ on CBF dataset for different interval settings. There are 30 training instances for this dataset therefore single misclassification increases error rate significantly. Interval length of 16 time units provides the minimum OOB error rate. On the other hand, the differences of OOB error rates are small therefore setting w smaller is suggested since we are interested in finding shorter patterns so that computation time required for distance computation is decreased. Test error rate is consistent with the OOB error rate.

($w = 10, d = 5$). Setting $L = 4$ or $L = 5$ is reasonable when the number of peaks in the local importance plot are considered for time series of each class .

After understanding the structure of the time series, the OOB error rates of $RFpattern$ for different L settings are analyzed. Larger L is expected to lead to better results up to certain level since more expressive patterns are generated by including more intervals. However the distance may become meaningless because of the curse of dimensionality if the length of the pattern gets too large. Although running $RFpattern$ for different L settings introduces complexity, computation time for testing and required space for storing the patterns can be reduced with the compact set of patterns obtained by smaller L level.

Figure 35 illustrates the progress of the error rates on CBF dataset for the setting $w = 6, d = 3$ and all L levels up to 15. OOB error rates becomes stable after $L = 8$ which is consistent with the test error rates. Test error rates are provided to illustrate the effectiveness of OOB error rates in terms of generalizability. Setting L larger may result in overfitting

since the distances of the patterns to the training data are more precise and dependent on the training data as longer patterns are generated. Overfitting problem is also discussed by [23] and the number of patterns to be generated is fixed to certain number (4 in their case) to overcome this problem. The same phenomenon is observed on the OOB error rates when L is larger than 10. Test error rates also show the same behavior. However the effect of overfitting is not severe because of the random selection of features at each split.

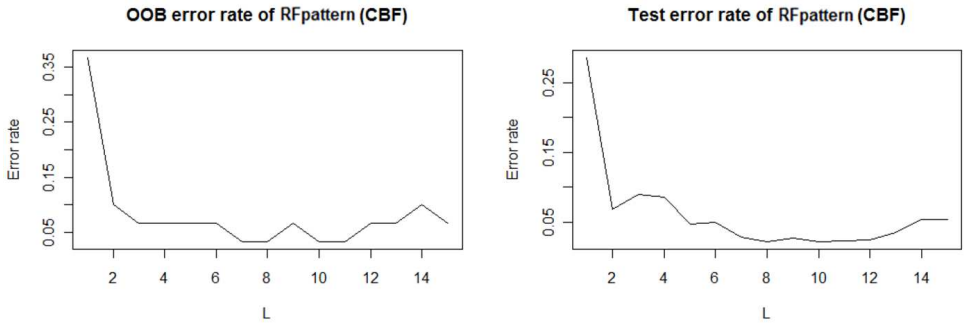


Figure 35. Progress of OOB error rates and test error rates over L settings. OOB error rates becomes stable after $L = 8$ which is consistent with the test error rates. Setting L larger may result in overfitting since the distances of the patterns to the training data are more precise and dependent on the training data. When L is larger than 10, slight increase on OOB error rates which is an indication of overfitting is observed.

5. Experiments

We test TS-PD on 43 time series data from [85]. The dataset characteristics are given in Table 11. This is a good testbed with diverse characteristics such as length of the series, number of classes etc. which enables a comprehensive evaluation.

In order to show the effectiveness of TS-PD in terms of accuracy, we test our algorithm with fixed w and L settings. Fixed parameters are considered to illustrate the robustness of TS-PD although the settings can be adjusted based on the data set characteristics in favor of our algorithm (as discussed in Section 6.1). Thus, we set $w = 6$ in order to have meaningful

	Number of classes	Training cases	Testing cases	Time series length
50words	50	450	455	270
Adiac	37	390	391	176
Beef	5	30	30	470
CBF	3	30	900	128
Coffee	2	28	28	286
ECG200	2	100	100	96
FaceAll	14	560	1,690	131
FaceFour	4	24	88	350
Fish	7	175	175	463
GunPoint	2	50	150	150
Lightning2	2	60	61	637
Lightning7	7	70	73	319
OliveOil	4	30	30	570
OSULeaf	6	200	242	427
SwedishLeaf	15	500	625	128
Syntheticcontrol	6	300	300	60
Trace	4	100	100	275
TwoPatterns	4	1,000	4,000	128
Wafer	2	1,000	6,164	152
Yoga	2	300	3000	426
ChlorineConcentration	3	467	3,840	166
CinC_ECG_torso	4	40	1,380	1,639
Cricket_X	12	390	390	300
Cricket_Y	12	390	390	300
Cricket_Z	12	390	390	300
DiatomSizeReduction	4	16	306	345
ECGFiveDays	2	23	861	136
FacesUCR	14	200	2,050	131
Haptics	5	155	308	1,092
InlineSkate	7	100	550	1,882
ItalyPowerDemand	2	67	1,029	24
MALLAT	8	55	2,345	1,024
MedicalImages	10	381	760	99
MoteStrain	2	20	1,252	84
SonyAIBORobot Surface	2	20	601	70
SonyAIBORobot SurfaceII	2	27	953	65
StarLightCurves	3	1,000	8,236	1,024
Symbols	6	25	995	398
TwoLeadECG	2	23	1,139	82
uWaveGestureLibrary_X	8	896	3,582	315
uWaveGestureLibrary_Y	8	896	3,582	315
uWaveGestureLibrary_Z	8	896	3,582	315
WordsSynonyms	25	267	638	270

TABLE 11. Characteristics of the datasets: number of classes, number of training cases, number of testing cases, and lengths of time series. The performance analysis of the algorithms on this diverse set of data provides a wide-ranging comparison.

features (such as slopes). Maximum pattern level setting is set as $L \in \{2, 4, 6, 8, 10\}$ to illustrate the progress of RFpattern's OOB and test error rates over different L settings. Although, patterns generated with $L = 10$ may be insufficient to describe certain features

for long time series, the same levels are considered on all datasets for illustration purposes. The number of trees for both forest is set to 2000.

5.1. *Computational accuracy*

TS-PD with the given settings is compared to nearest neighbors (NN) classifiers with DTW. Two versions of DTW are considered: NNDTWBestWin (also referred to as NNBestDTW) [17] searches for the best warping window, based on the training data, then uses the learned window on the test data, while NNDTWNoWin does not search for any constraints on the warping path. Note that DTW is a strong solution known for time series problems in a variety of domains [58] although it may not be suitable for certain applications because of computational and space requirements [23]. The results for NN classifiers are obtained from [85]. Tables 12 and 13 summarizes the OOB and test error rates for $RFpattern$ for all L settings. For certain L settings, TS-PD is not run (represented as '-') since the pattern is potentially longer than the time series. We also compare our results with Logical-Shapelets [23] which significantly outperforms the original shapelet representation proposed by [61]. Since this comparison is not based on all datasets because of the computational requirements of Logical-Shapelets, we compare TS-PD to Logical-Shapelets in Section 6.4.

	$RFpattern$										NNDTW	
	OOB error rate					Test error rate					BestWin	NoWin
	$L=2$	$L=4$	$L=6$	$L=8$	$L=10$	$L=2$	$L=4$	$L=6$	$L=8$	$L=10$		
50Words	0.404	0.336	0.329	0.331	0.329	0.354	0.295	0.273	0.266	0.257	0.242	0.310
Adiac	0.287	0.285	0.282	0.287	0.295	0.243	0.246	0.240	0.248	0.246	0.391	0.396
Beef	0.500	0.467	0.533	0.400	0.467	0.367	0.333	0.233	0.233	0.267	0.467	0.500
CBF	0.100	0.067	0.033	0.033	0.033	0.113	0.078	0.033	0.027	0.038	0.004	0.003
Coffee	0.000	0.000	0.000	0.000	0.000	0.000	0.036	0.036	0.036	0.036	0.179	0.179
ECG	0.150	0.170	0.160	0.120	0.130	0.220	0.220	0.200	0.180	0.190	0.120	0.230
Face (all)	0.086	0.068	0.054	0.046	0.045	0.234	0.234	0.254	0.263	0.258	0.192	0.192
Face (four)	0.167	0.083	0.042	0.042	0.000	0.295	0.091	0.114	0.102	0.045	0.114	0.170
Fish	0.200	0.194	0.206	0.200	0.206	0.154	0.189	0.166	0.166	0.166	0.160	0.167
Gun-Point	0.080	0.100	0.060	0.060	0.060	0.067	0.047	0.060	0.040	0.060	0.087	0.093
Lighting-2	0.150	0.117	0.117	0.117	0.133	0.311	0.246	0.262	0.279	0.279	0.131	0.131
Lighting-7	0.314	0.229	0.257	0.243	0.243	0.384	0.329	0.329	0.301	0.288	0.288	0.274
OliveOil	0.100	0.067	0.033	0.100	0.067	0.267	0.200	0.200	0.200	0.200	0.167	0.133
OSU Leaf	0.315	0.245	0.250	0.250	0.235	0.380	0.310	0.314	0.318	0.302	0.384	0.409
Swedish Leaf	0.098	0.096	0.098	0.092	0.098	0.096	0.086	0.090	0.098	0.101	0.157	0.210
Synthetic Control	0.030	0.017	0.020	0.023	0.023	0.033	0.023	0.017	0.010	0.017	0.017	0.007
Trace	0.000	0.000	0.000	0.040	0.020	0.010	0.000	0.000	0.020	0.020	0.010	0.000
Two Patterns	0.005	0.000	0.000	0.000	0.001	0.004	0.001	0.000	0.000	0.001	0.002	0.000
Wafer	0.020	0.011	0.005	0.006	0.006	0.020	0.010	0.006	0.005	0.005	0.005	0.020
Yoga	0.217	0.187	0.187	0.187	0.187	0.181	0.174	0.149	0.156	0.145	0.155	0.164
ChlorineConcentration	0.298	0.315	0.315	0.315	0.313	0.319	0.312	0.317	0.335	0.344	0.350	0.352
CinC_ECG_torso	0.425	0.400	0.350	0.475	0.375	0.459	0.476	0.491	0.452	0.452	0.070	0.349

TABLE 12. Error rates of TS-PD ($w = 6, 2000$ trees) for different settings of L, nearest-neighbor classifiers with dynamic time warping distance, where NNDTWBestWin searches the best warping window based on the training data, NNDTWNoWin has no warping window.

	$RFpattern$										NNDTW	
	OOB error rate					Test error rate					BestWin	NoWin
	$L=2$	$L=4$	$L=6$	$L=8$	$L=10$	$L=2$	$L=4$	$L=6$	$L=8$	$L=10$		
Cricket_X	0.295	0.269	0.244	0.236	0.241	0.318	0.297	0.274	0.256	0.272	0.236	0.223
Cricket_Y	0.369	0.331	0.313	0.305	0.274	0.385	0.338	0.315	0.287	0.251	0.197	0.208
Cricket_Z	0.338	0.308	0.279	0.256	0.269	0.321	0.262	0.233	0.244	0.218	0.180	0.208
DiatomSizeReduction	0.063	0.063	0.063	0.063	0.063	0.108	0.131	0.075	0.127	0.124	0.065	0.033
ECGFiveDays	0.217	0.174	0.130	0.130	0.087	0.233	0.224	0.256	0.289	0.252	0.203	0.232
FacesUCR	0.150	0.150	0.125	0.100	0.100	0.224	0.186	0.150	0.105	0.094	0.088	0.095
Haptics	0.458	0.432	0.387	0.394	0.394	0.532	0.558	0.513	0.516	0.519	0.588	0.623
InlineSkate	0.640	0.700	0.680	0.670	0.690	0.660	0.631	0.604	0.611	0.611	0.613	0.616
ItalyPowerDemand	0.030	0.075	-	-	-	0.048	0.049	-	-	-	0.045	0.050
MALLAT	0.036	0.036	0.018	0.018	0.036	0.065	0.043	0.038	0.030	0.026	0.086	0.066
MedicalImages	0.262	0.252	0.255	0.249	0.262	0.289	0.283	0.278	0.284	0.271	0.253	0.263
MoteStrain	0.300	0.200	0.250	0.150	0.100	0.154	0.105	0.114	0.121	0.121	0.134	0.165
SonyAIBORobot Surface	0.050	0.150	0.150	0.150	0.150	0.153	0.123	0.143	0.098	0.065	0.305	0.275
SonyAIBORobot SurfaceII	0.222	0.259	0.148	0.148	0.148	0.256	0.248	0.196	0.187	0.199	0.141	0.169
StarLightCurves	0.037	0.033	0.036	0.038	0.042	0.039	0.037	0.037	0.037	0.038	0.095	0.093
Symbols	0.240	0.120	0.040	0.040	0.040	0.169	0.150	0.134	0.067	0.104	0.062	0.050
TwoLeadECG	0.043	0.087	0.130	0.087	0.087	0.143	0.163	0.147	0.105	0.139	0.132	0.096
uWaveGestureLibrary_X	0.232	0.227	0.224	0.209	0.203	0.250	0.228	0.210	0.206	0.203	0.227	0.273
uWaveGestureLibrary_Y	0.280	0.267	0.276	0.267	0.265	0.303	0.291	0.299	0.298	0.296	0.301	0.366
uWaveGestureLibrary_Z	0.288	0.263	0.270	0.259	0.267	0.284	0.266	0.269	0.264	0.266	0.322	0.342
WordsSynonyms	0.502	0.427	0.408	0.386	0.401	0.469	0.406	0.409	0.378	0.368	0.252	0.351

TABLE 13. Error rates of TS-PD ($w = 6, 2000$ trees) for different settings of L (continued), nearest-neighbor classifiers with dynamic time warping distance, where NNDTWBestWin searches the best warping window based on the training data, NNDTWNoWin has no warping window. For certain w and L combination, TS-PD is not run (represented as '-') since the pattern is potentially longer than the time series.

We use the same idea proposed by [21] for comparison of TS-PD to other algorithms. Pairwise comparison of error rates is done using scatter plots in which each axis represents the approach under consideration and each dot represents the error rate for a particular dataset. The line $x = y$ is drawn to represent the region where both methods perform about the same. A point above the line indicates that approach on the X axis has better accuracy than the one on Y axis. If a point is further from the line, the margin of accuracy improvement is greater. A method can be regarded as superior to other if there are more points on one side of the line. Figure 36 illustrates the comparison of TS-PD with NNDTWNoWin and NNDTWBestWin. The error rates of $L = 10$ are used for comparison since we expect TS-PD to provide stable results after certain L setting based on our discussion. We use the result of the largest possible L setting for the cases that pattern length is larger than the time series length (i.e. error rate of TS-PD $L = 4$ is used for ItalyPowerDemand dataset).

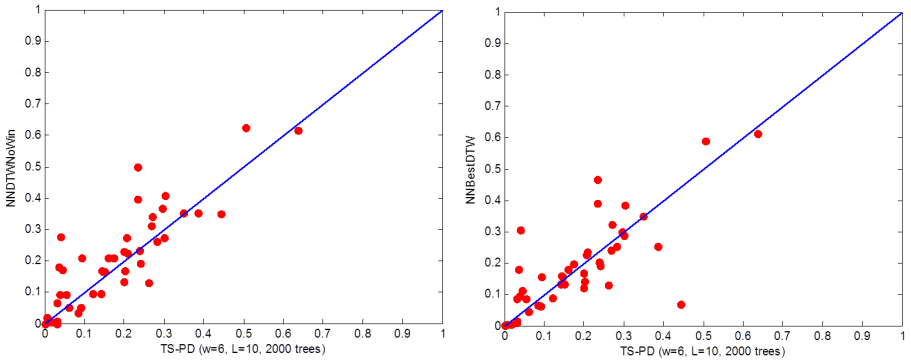


Figure 36. Scatter plot of error rates of TS-PD vs NNDTWNoWin and NNDTWBestWin. TS-PD with the given settings provides comparable results to NNDTWNoWin and NNDTWBestWin

TS-PD with the given parameters provides comparable results to NNDTWNoWin and NNDTWBestWin. For certain instances such as CinC_ECG_torso, DTW based classifiers have significantly better error rates. This is related to the problem structure and the param-

eter settings. Note that the maximum possible pattern length with the given settings is 60 time units which may create problems for certain datasets in which features of interest are long (length of the series is 1639 for CinC_ECG_torso dataset). For example, setting the parameters as $w = 50, L = 10$ for CinC_ECG_torso dataset reduces to OOB and test error rates from 0.375 and 0.452 to 0.35 and 0.343, respectively, with the same number of trees. This again confirms our discussion about setting the parameters after the analysis of the OOB error rates of $RFint$ and local importance plots.

5.2. *Computational complexity*

TS-PD is implemented in R Software and our experiments use a Windows 7 system with 8 GB RAM, dual core CPU (i7-3620M 2.7 GHz). We use R only for building the RFs and implemented the algorithms for feature generation and distance computation in C, because R is computationally inefficient in execution of the loops. Moreover, although the CPU can handle four threads in parallel, only a single thread is used. The computation times of TS-PD ($w = 6, 2000$ trees) for different settings of L are provided in Tables 14 and 15.

	Training time (secs)					Test time (secs)				
	$L=2$	$L=4$	$L=6$	$L=8$	$L=10$	$L=2$	$L=4$	$L=6$	$L=8$	$L=10$
50Words	74.63	95.35	123.26	150.63	181.48	0.0097	0.0112	0.0131	0.0151	0.0162
Adiac	49.85	73.64	98.43	112.65	137.31	0.0052	0.0067	0.0082	0.0102	0.0099
Beef	1.24	1.35	1.42	1.56	1.76	0.0090	0.0093	0.0093	0.0093	0.0070
CBF	0.59	0.64	0.78	0.74	0.86	0.0003	0.0003	0.0002	0.0004	0.0004
Coffee	0.59	0.71	0.61	0.67	0.80	0.0068	0.0036	0.0068	0.0071	0.0050
ECG	1.93	2.51	2.68	2.85	3.03	0.0022	0.0018	0.0026	0.0029	0.0023
Face (all)	73.85	95.23	117.86	135.41	158.17	0.0048	0.0066	0.0078	0.0085	0.0097
Face (four)	0.71	0.72	0.87	0.87	0.89	0.0023	0.0027	0.0022	0.0027	0.0028
Fish	13.86	18.35	22.79	28.51	34.61	0.0105	0.0120	0.0139	0.0151	0.0169
Gun-Point	0.78	0.81	0.96	1.05	1.30	0.0009	0.0015	0.0014	0.0013	0.0009
Lighting-2	2.85	3.32	3.78	4.54	5.52	0.0144	0.0133	0.0141	0.0144	0.0134
Lighting-7	2.73	3.22	3.71	4.11	4.66	0.0067	0.0078	0.0071	0.0085	0.0086
OliveOil	1.25	1.30	1.54	1.60	1.58	0.0113	0.0123	0.0090	0.0097	0.0143
OSU Leaf	16.84	22.03	27.87	36.51	45.51	0.0095	0.0098	0.0107	0.0119	0.0128
Swedish Leaf	58.45	86.26	104.45	123.53	191.52	0.0053	0.0069	0.0077	0.0088	0.0096
Synthetic Control	10.27	12.07	14.78	16.72	18.92	0.0018	0.0024	0.0026	0.0023	0.0028
Trace	3.13	3.34	4.39	5.41	6.26	0.0059	0.0086	0.0070	0.0064	0.0075
Two Patterns	165.13	217.21	294.93	358.69	483.36	0.0092	0.0115	0.0142	0.0163	0.0179
Wafer	420.35	608.73	786.82	928.63	1151.34	0.0128	0.0141	0.0189	0.0222	0.0256
Yoga	32.18	45.77	60.05	82.29	104.67	0.0077	0.0097	0.0114	0.0155	0.0172
ChlorineConcentration	71.32	121.89	157.26	206.80	300.02	0.0057	0.0076	0.0091	0.0103	0.0114
CinC_ECG_torso	6.02	5.58	6.33	6.91	8.18	0.0044	0.0062	0.0083	0.0087	0.0112

TABLE 14. Computation times of TS-PD ($w = 6, 2000$ trees) for different settings of L. Testing time is the computation time of classifying single time series. For certain d and L combination, TS-PD is not run (represented as '-') since the pattern is longer than the time series.

	Training time (secs)					Test time (secs)				
	$L=2$	$L=4$	$L=6$	$L=8$	$L=10$	$L=2$	$L=4$	$L=6$	$L=8$	$L=10$
Cricket_X	55.44	75.86	93.59	116.22	130.30	0.0114	0.0145	0.0171	0.0184	0.0196
Cricket_Y	53.78	63.62	77.53	89.56	113.50	0.0109	0.0143	0.0162	0.0177	0.0192
Cricket_Z	58.75	74.32	83.71	106.47	112.14	0.0119	0.0134	0.0148	0.0191	0.0180
DiatomSizeReduction	0.55	0.50	0.57	0.64	0.64	0.0009	0.0008	0.0009	0.0009	0.0012
ECGFiveDays	0.42	0.42	0.53	0.42	0.60	0.0002	0.0003	0.0003	0.0004	0.0004
FacesUCR	11.36	13.11	16.68	17.40	21.30	0.0018	0.0020	0.0026	0.0028	0.0030
Haptics	25.45	32.03	37.95	49.47	63.99	0.0188	0.0230	0.0248	0.0283	0.0311
InlineSkate	25.12	28.54	33.14	38.79	48.57	0.0192	0.0220	0.0229	0.0242	0.0289
ItalyPowerDemand	0.83	0.85	-	-	-	0.0001	0.0001	-	-	-
MALLAT	7.10	6.84	7.87	8.47	10.13	0.0044	0.0055	0.0065	0.0067	0.0078
MedicalImages	29.48	47.00	54.93	61.12	68.92	0.0031	0.0037	0.0044	0.0052	0.0049
MoteStrain	0.36	0.38	0.46	0.43	0.53	0.0001	0.0002	0.0002	0.0002	0.0002
SonyAIBORobot Surface	0.24	0.28	0.24	0.36	0.38	0.0001	0.0001	0.0002	0.0001	0.0001
SonyAIBORobot SurfaceII	0.35	0.42	0.49	0.42	0.56	0.0001	0.0002	0.0001	0.0002	0.0001
StarLightCurves	450.63	652.21	992.21	1496.34	2068.24	0.0577	0.0807	0.1030	0.1292	0.1529
Symbols	1.37	1.44	1.61	1.44	1.63	0.0009	0.0010	0.0010	0.0012	0.0015
TwoLeadECG	0.49	0.47	0.58	0.63	0.59	0.0001	0.0001	0.0002	0.0001	0.0002
uWaveGestureLibrary_X	278.50	408.43	484.23	642.23	762.76	0.0224	0.0284	0.0341	0.0382	0.0408
uWaveGestureLibrary_Y	283.78	439.94	508.32	786.38	917.28	0.0206	0.0259	0.0299	0.0347	0.0370
uWaveGestureLibrary_Z	290.61	412.75	527.15	670.36	757.88	0.0223	0.0292	0.0354	0.0394	0.0422
WordsSynonyms	31.10	35.95	44.88	55.28	64.39	0.0074	0.0081	0.0086	0.0099	0.0092

TABLE 15. Computation times of TS-PD ($w = 6, 2000$ trees) for different settings of L (continued). Testing time is the computation time of classifying single time series. For certain d and L combination, TS-PD is not run (represented as '-') since the pattern is longer than the time series.

There are three components of TS-PD, local importance generation, pattern discovery and distance computation, classification. We will discuss the complexity of each component instead of providing an overall computational complexity since our approach is an exploratory tool where user should find out certain settings through the analysis of OOB errors and visual tools presented.

Computational complexity of the local importance generation is mainly due to $RFint$. Time complexity of building single tree of $RFint$ is $O(\sqrt{\nu}N \log N)$ where ν is the number of features extracted from each time series and N is the number of training instances. Smaller interval and sliding step lengths result in larger number of features for $RFint$ however the increase in the complexity is comparably small since only subset of features are considered at each split. However, one may want to generate more trees when number of features is large since the probability of selecting each feature decreases. The number of trees is decided by the analysis of the OOB error rates. Same discussion holds for $RFpattern$, the classification component of TS-PD, since it is also an RF classifier.

Pattern discovery and distance computation requires sampling of the important intervals and finding the distances of these samples to each time series. The time to compute the distance of a pattern to the time series is $O(zM)$ where z is the length of the pattern and M is the length of the time series. The length of the pattern is determined by the interval length w and maximum level setting L in our algorithm. Although we generate L patterns from each time series, the complexity of distance computation does not change since the distances can be computed for all patterns in the same loop. Thus, the complexity of computing all pattern distances is $O(zM)$ where z is now the length of the level L pattern which is $z = Lw$ in the worst case. The length of the pattern can be less than $z = Lw$ because of

the overlapping intervals generated. The minimum possible length is $z = (L-1) \times d + w$ when all intervals overlap (i.e. all first L important intervals are contiguous).

The computation times of TS-PD with increasing training dataset size, N, are illustrated in Figure 37 on Two Patterns dataset ($w = 6, L = 5, 1000$ trees). This increase is mainly due to the complexity of $RFpattern$ which is $O(\sqrt{\|S\|} N \log N)$. The number of patterns in the set S increases as the number of training time series increases. This is the main reason of the practically quadratic complexity on the number of the training time series illustrated in Figure 37. The training time increases as N becomes larger due to the combined effect of the increase in the number of features and the training data. On the other hand, the increase in the time for classifying an instance is practically linear because of the increase in the number of patterns. On the other hand, there are several ways to reduce the complexity such as pruning the pattern set or downsampling the training data as discussed in Section 5.3.

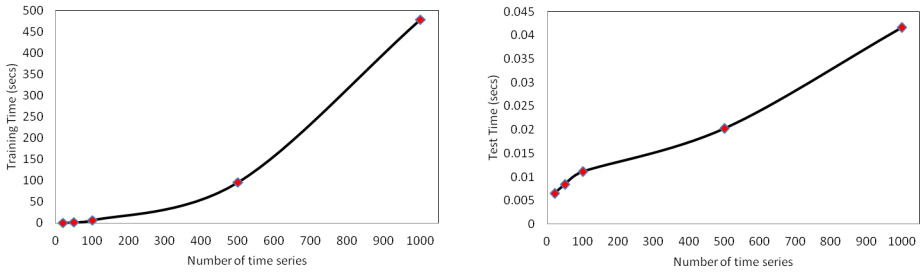


Figure 37. Training (left) and testing (right) times of TS-PD on Two Patterns dataset for increasing dataset sizes ($w = 6, L = 5, 1000$ trees).

We illustrate the behavior of the computation times with different L and w settings in Figure 38 for FacesUCR dataset. The increase in the number of patterns with larger L setting is the main reason of the increase in training time. The complexity added by introducing larger L is mainly because of the increase in the number of features for $RFpattern$. The computation time required for computation of the distances also increases but it is not

significant since the distance computation is done in a single pass for all possible patterns. Considering the training times, the practical complexity of TS-PD is approximately linear on L setting when other parameters are fixed.

Larger interval setting results in longer patterns that requires larger computation time however the training times are slightly smaller or about the same level when L is larger as in Figure 38. This is mainly due to the less number of features considered for $RFint$ since the number of features is less for larger w settings which will reduce the training time of $RFint$ significantly. Moreover, the number of features stays the same for $RFpattern$ for respective L settings.

To classify a time series, distances to each pattern is computed over the time series. The time required for testing is mainly due to this distance computation. After the distance computation, the object is classified by traversing the trees of $RFpattern$ which is very fast. Time for distance computation increases as L increases however this increase is not large because of the efficient distance computation for the different pattern levels as described earlier. Therefore time to classify an instances does not increase significantly as illustrated in in Figure 38. Computation time is almost linear to the pattern level setting.

We also consider the computation times of TS-PD for time series of different length. These datasets are ItalyPowerDemand, Synthetic Control, ECG, CBF, Trace, OliveOil, MALLAT and InlineSkate (the lengths of the time series are 24, 60, 96, 128, 275, 570, 1024 and 1882 respectively). We randomly selected 30 training instances from each dataset. The computation times are illustrated in Figure 39 ($w = 6, L = 3, 1000$ trees). Similar discussion in terms of the number of features holds for longer time series. The number of features increases for $RFint$ for longer series and the training time increases. Similarly time for

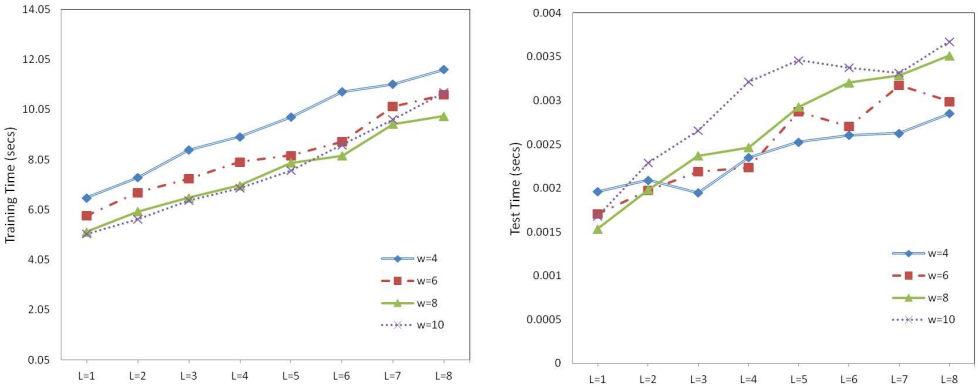


Figure 38. Training (left) and testing (right) times of TS-PD for FacesUCR dataset for different w and L settings. Emprically, the training and test times is linear with the pattern level setting.

distance computation is larger for longer time series. The computation time changes in a linear fashion with the change in the length of the time series when other parameters are fixed.

5.3. *Complexity reduction*

The complexity of TS-PD can be reduced in several ways. Similar or same patterns are not pruned in this study, thus pruning similar patterns improves the overall computation time. Moreover, a subset of instances can be selected for pattern generation based on certain

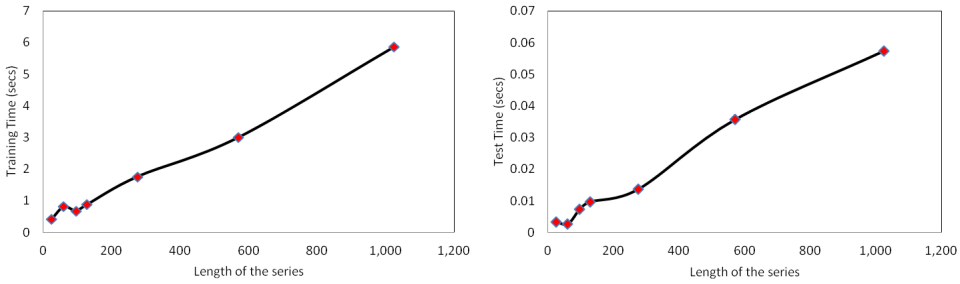


Figure 39. Training (left) and testing (right) times for series of different length ($w = 6, L = 3, 1000$ trees). There are 30 training instances for each time series.

N	Error rates		Comp. Time (secs)	
	OOB	Test	Train	Test
50	0.060	0.065	3.99	0.0041
100	0.060	0.063	8.35	0.0081
200	0.050	0.049	23.37	0.0157
400	0.050	0.048	74.59	0.0320
750	0.041	0.040	262.96	0.0610
1000	0.037	0.037	515.11	0.0814

TABLE 16. Error rates and computation times of TS-PD ($w = 6, L = 4, 1000$ trees) for different training data sizes. The training time is significantly smaller when TS-PD is trained on less number of instances. However, the change in the test error rate is not substantial. If there are certain constraints on the computation time or space availability, training on smaller datasets may be preferred.

criterion to reduce the computation time. For instance, a simple similarity computation between the training instances (i.e. finding similar instances based on Euclidean distance) and discarding the similar train instances may help to reduce the computational effort.

StarLightCurves dataset is used to illustrate how the computation time and accuracy are affected when the training data is downsampled. It is the one of the largest datasets with 1000 training and 8236 test instances of 1024 time units long. We randomly sample 50, 100, 200, 400, 750 instances while keeping the class distributions same as the original data and report the computation times and error rates of TS-PD ($w = 6, L = 4, 1000$ trees) in Table 16. The training time is significantly smaller when TS-PD is trained on less number of instances. However, the change in the test error rate is not substantial. If there is certain constraints on the computation time and space availability, training on smaller datasets may be preferred.

6. Discussion

6.1. *Illustrative example*

TS-PD is proposed as an exploratory tool for the analysis of the time series for classification purposes and parameters should be set after a detailed analysis of the certain measures such as local importance and OOB error rates of $RFint$ and $RFpattern$. Although TS-PD is robust to these parameters if they are set within a boundary as shown in Section 5.1 over all datasets, we will illustrate the steps of the analysis on FacesUCR dataset in this section. FacesUCR data consists of face images of graduate students transformed into time series. An example of this conversion is provided in Figure 40. There are 14 students and 2250 pictures are taken under different conditions such as glass/no glass or expressions. The length of the series is 131 time units and the training data consists of 200 time series where the rest is used for testing.

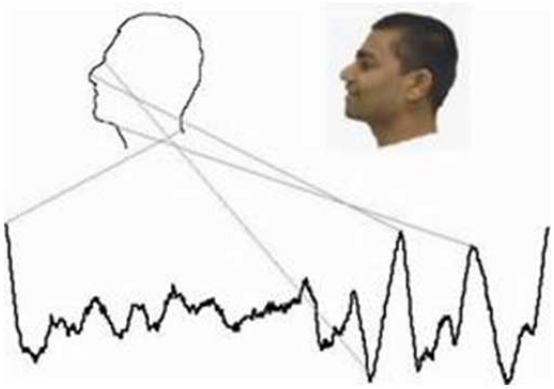


Figure 40. Illustration of the transformation of a face image to the time series.

First step in TS-PD is the local importance generation through interval feature generation and classification by $RFint$. The parameters to be set are the number of trees and interval length (assuming that the sliding window is fixed as half of the interval length).

We start with the smallest interval length that will generate meaningful patterns which is 6 and increment it by 2 up to 40 to see how the OOB error rates change. Initially we set the number of trees as 1000. OOB error rate for $w = 6$ over trees and OOB error rates of $RFint$ for different w settings are given in Figure 41. The progress of the OOB error rates shows that 1000 trees are more than enough for $w = 6$. We use the same level for other w settings assuming that the number of trees will be sufficient for a dataset with less number of features. Interval lengths between 14 and 24 can be used as the interval length setting as they provide lower error rates.

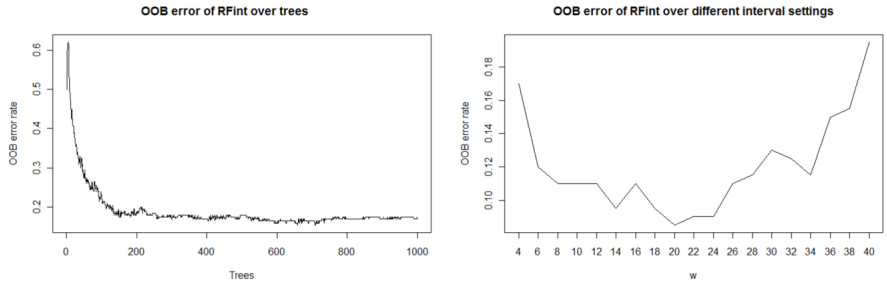


Figure 41. OOB error rates of $RFint$ over trees for $w = 6$ (left), OOB error rates of $RFint$ (number of trees=1000) for different settings of interval length (right).

Local importance plots and time series are provided for each class in Figure 42 after setting $w = 20$ considering the OOB error rates in Figure 41. The next step in TS-PD is to set the pattern level. In this particular example, setting L as 3 or 4 seems reasonable by looking at each local importance plot.

Local importance plot does not only provide insight about the pattern level, it also illustrates the difficulty of the classification problem. Consider the time series from 'class 2', most of the time series of this class does not overlap because of certain variations in the time series. Similar observation can be done considering the local importance plot. Variation of the patterns within the class can be observed from these plots.

Last step is the training of $RFpattern$ on the distance features. Number of trees is again set based on the OOB error rate over the trees. The plot of OOB error rates over trees for $RFpattern$ ($L = 4$) is given in Figure 43 for 1000 trees. OOB error rate of $RFpattern$ is 0.095 and the test error rate over 2050 time series is 0.090. The error rates of NNDTWBestWin and NNDTWNoWin are 0.088 and 0.095 respectively.

6.2. *Interpretability*

Section 4.3 discusses how interpretability is achieved by TS-PD. We will illustrate the comprehensibility of our classifier on certain examples. These examples include Gun-Point, Sony AIBO Robot and Coffee datasets.

6.2.1. *Gun-Point.* Gun-Point dataset is one of the most studied time series classification problem [21]. The aim is to classify a motion as 'Gun' or 'NoGun' through time series generated by mapping the motion of two actors. For the Gun class, the actors "have their hands by their sides, draw a gun from a hip-mounted holster, point it at a targer for approximately one second, and then return the gun to the holster and their hands to their sides" [61]. In the NoGun class, actors do the same movements as in the Gun class without a gun. Instead they use their index finger to point to a target. Therefore in NoGun class, the step of drawing the gun from holster and returning it back is skipped. The dataset characteristics are the same as provided in Table 11.

The interval length is set as $w = 20$ after the analysis of the OOB errors of $RFint$ with 1000 trees for interval lengths between 4 and 40. We set $L = 3$ considering the local importance plots. The test error rate of $RFpattern$ is 0.06 where the error rates of NNDTWBestWin and NNDTWNoWin are 0.087 and 0.093 respectively. In addition to better accuracy, TS-PD is very fast in classification when compared to nearest neighbor

classifiers. It only requires the distance computation of the patterns to the time series and tree traversal over $RFpattern$.

The first five important patterns from $RFpattern$ are schematized in Figure 44. All patterns are generated from the time series of Gun class. The regions refer to the actions, ”draw a gun from a hip-mounted holster” and ”return the gun to the holster and their hands to their sides”. The first two patterns are from the same time series and both of them are found to be important. This illustrates the redundancy issue discussed in Section 4.3. A feature selection algorithm can be used to find the compact set of patterns in that case.

6.2.2. *Sony AIBO Robot.* This dataset is created by [91] and the task is to classify the surface types using the measurements of the tri-axial accelerometer from Sony AIBO Robot [23]. Only the X-axis readings of the accelerometer is provided in [85]. Two types of surfaces, carpet and cement, are considered in this dataset. Cement floors are harder resulting in sharper changes in the accelaration [23]. The dataset characteristics are the same as provided in Table 11.

The algorithm parameters $w = 20, d = 10, L = 2$ lead to a test error rate of 0.036. Logical-shapelets [23] also achieve the same error rate where the error rates of NNDTWBestWin and NNDTWNoWin are 0.305 and 0.275 respectively. The improvement in error rate is substantial compared to NN classifiers. The important patterns provided in Figure 45 are similar to the shapelets by [23] and they refer to different shifts-of-weight in the walk cycle on the carpet floor.

6.2.3. *Coffee.* The task is to classify the coffee species in instant coffees in this dataset. A chemical analysis, called Diffuse Reflectance Infrared Fourier Transform (DRIFT), is

used to discriminate between two species of coffees as Arabica and Robusta [92]. The characteristics of the dataset is provided in Table 11.

The parameters are set as $w = 6, d = 3, L = 3$ after the analysis of the dataset characteristics. All test instances are classified correctly by TS-PD where the error rates of NNDTWBestWin and NNDTWNoWin are 0.179. The first five important patterns and the training time series are schematized in Figure 46. [92] states that certain spectral regions represent the caffeine bands. These regions correspond to the time frame between 187.7 and 247.3 as discussed by [61]. Some of the important patterns are between these regions as illustrated in Figure 46.

6.3. *Gesture recognition: an application of TS-PD to multivariate time series classification*

We illustrate the effectiveness of TS-PD only on univariate time series in Section 5. TS-PD can be extended to the multivariate time series classification (MTSC) by changing the representation. We will discuss how TS-PD can be extended to multivariate case on a gesture recognition problem proposed by [5].

A single three-axis accelerometer is used to collect data from eight users to characterize eight gesture patterns. The library, uWaveGestureLibrary, consists over 4000 samples each of which has the accelerometer readings in three dimensions (i.e. x, y and z) [5]. Individual axes are considered in Section 5 for the univariate case (the datasets are uWaveGestureLibrary_X, uWaveGestureLibrary_Y, uWaveGestureLibrary_Z). However handling this problem as a MTSC problem may provide better results by taking the interaction between the individual axes into account.

We transform the multiple time series representation to a univariate one by concatenating each axis as illustrated in Figure 47. This transformation provides desirable properties for our approach. The interaction between the time series and their correlation are two important aspects for MTSC. Our tree based local importance generation scheme handles both in efficient way. Interaction is naturally handled by RFs where correlation is not a problem as RF works on the random subsets of the features. On the other hand, the length of the concatenated series can get larger as the number of time series increases but RFs can handle large number of features with the random sampling of the features.

Local importance generation is the core component of TS-PD for classification of multivariate time series. TS-PD is modified slightly to handle multivariate case in our study. Interval features are generated for each time series and concatenated, then regions of interest for each axis are discovered. The rest of the algorithm is the same as what is done for the univariate time series with one difference. We consider the patterns within the boundary of each time series. Suppose a pattern that has one interval from each time series is generated using the importance values, first interval in the pattern have to stay in the first time series in the distance computation stage. Computing the distance of a pattern from one series to different time series does not make sense.

This dataset has 896 training and 3582 test time series. Combining the three axes results in a time series of length 945 time units. OOB and test error rates of TS-PD ($w = 20, L = 5, 1000$ trees) are 0.066 and 0.069 respectively. Considering the error rates provided in Tables 12 and 13, the error rate reduces significantly if the task is taken as a multivariate time series classification. Figure 48 provides the gesture vocabulary from [5] (bottom) and important patterns from two classes. Two series from class 7 and 8 represents the

circular movement in opposite directions. Patterns extracted for these instances represent the segments related to change in the direction during the circular movement. The change is opposite in the sign for different classes which refers the circular movement in opposite directions.

6.4. *Logical-Shapelets and TS-PD*

We compare the performance of TS-PD to Logical-Shapelets for certain datasets. In order to be fair in terms of comparison, we set the parameters of logical shapelet algorithm so that it will search for all possible shapelets. However we could not achieve this because of the computational requirements of the algorithm for certain datasets. Therefore we perform this comparison based on a subset of the datasets. These datasets are Beef, CBF, Coffee, ECG and Trace. Three parameters of Logical-Shapelets are the maximum and minimum length of the shapelet and the step size. We set the maximum length as the time series length, minumum as two and we take step size as one. This does not necessarily mean that the best accuracy is obtained on the test set with this settings since the shapelets are evaluated based on the training set. Moreover, we tested our algorithm on two additional datasets discussed in [23]. These datasets are Cricket and Passgraphs. The explanations of these datasets can be found in the original paper [23]. We do not tune the parameters of our algorithm for the new datasets, we set ($w = 6, L = 10, 2000$ trees). We also do not compare the algorithms in terms of computation time because the comparison depends to a large extent on parameter settings. The results are provided in Table 17.

TS-PD has better or comparable performance on the datasets except for ECG dataset (and TS-PD is still better than NNDTWNoWin on this data set). Recall that the parameters of Logical-Shapelets are set so that it searches over the entire space which increases

	TS-PD (OOB)	TS-PD (Test)	Logical-Shapelets	NNDTWBestWin	NNDTWNoWin
Beef	0.467	0.267	0.600	0.467	0.500
CBF	0.033	0.038	0.336	0.004	0.003
Coffee	0.000	0.036	0.071	0.179	0.179
ECG	0.130	0.190	0.140	0.120	0.230
Trace	0.023	0.017	0.530	0.010	0.000
Sony A.R.	0.150	0.065	0.036	0.305	0.275
Cricket	0.000	0.000	0.041	0.051	0.010
Passgraphs	0.261	0.260	0.298	0.260	0.282

TABLE 17. Error rates of Logical-Shapelets and TS-PD on 8 datasets. TS-PD has better or comparable performance on the datasets except ECG.

the computational time significantly. Potentially equivalent accuracy can be obtained with alternative settings on the parameters, but our objective here is to assess the accuracy. Also, we do not provide the time for testing because both algorithms are very fast in classification.

7. Conclusion

A framework is presented to analysis of time series for classification. To find the interesting regions of the time series for classification, a supervised learner is trained on the local features to generate a local importance measure. Regions of interests are important to understand the underlying relations in the time series. Once the regions of interests are identified for each time series using the local importance values, potential patterns are generated from these regions. This allows for pruning the search space without losing information about the time series in an efficient way. Each time series is then represented by their distances to the potential patterns and a new feature matrix of distances is used for classification. TS-PD is comprehensible and our experimental results show that it gives comparable results to competitive methods on the benchmark data sets from UCR time series database [85]. Although our focus in this study is on the classification of the time series, TS-PD can be easily adjusted to other applications such as similarity analysis, clustering, and so forth.

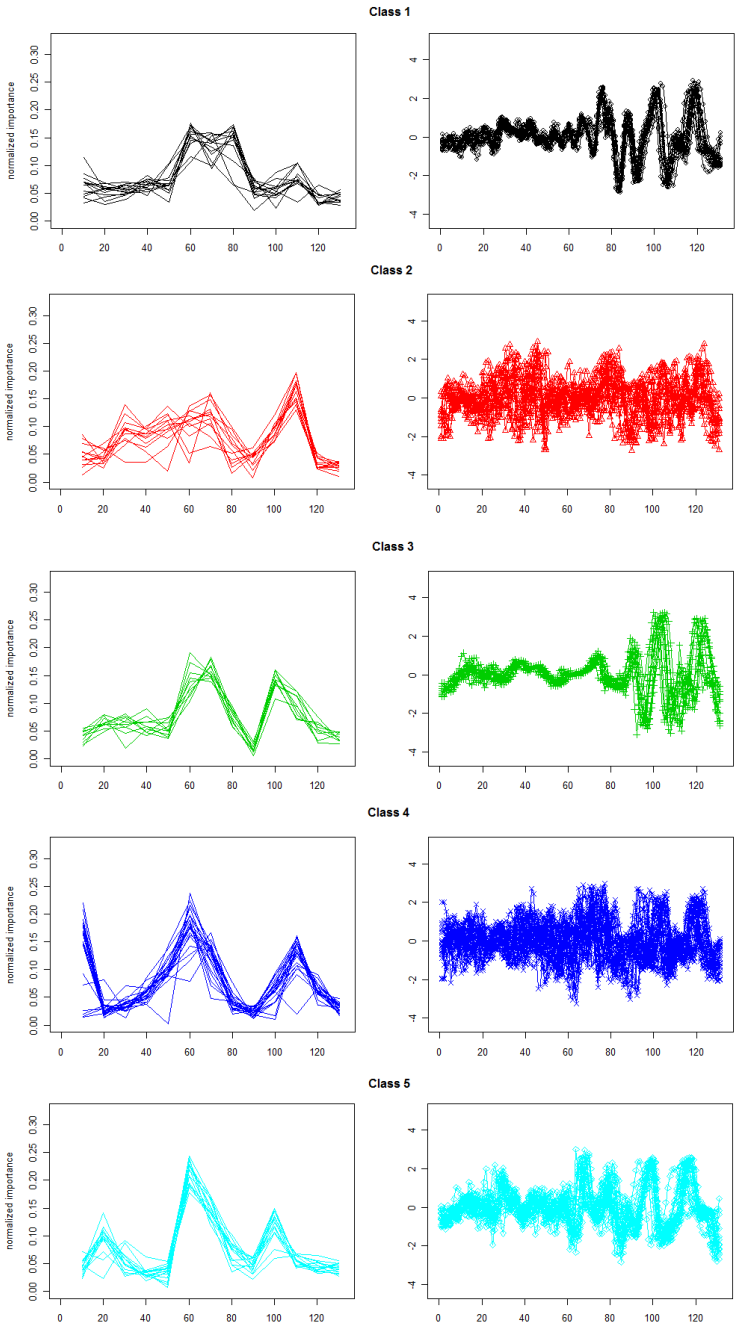


Figure 42. Normalized local importance information on FacesUCR (left) and time series of each class (right). The parameters are set as $w = 20, d = 10$.

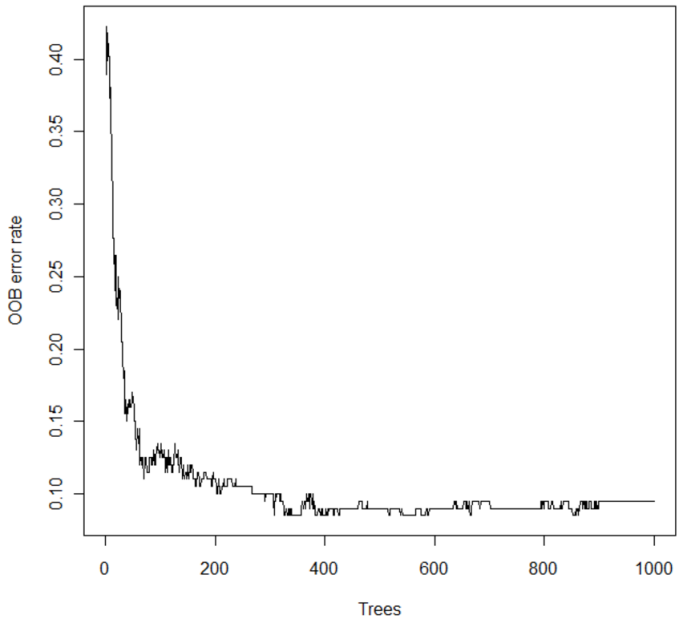


Figure 43. The OOB error rates of $RFpattern$ over trees for $w = 20$, $L = 4$

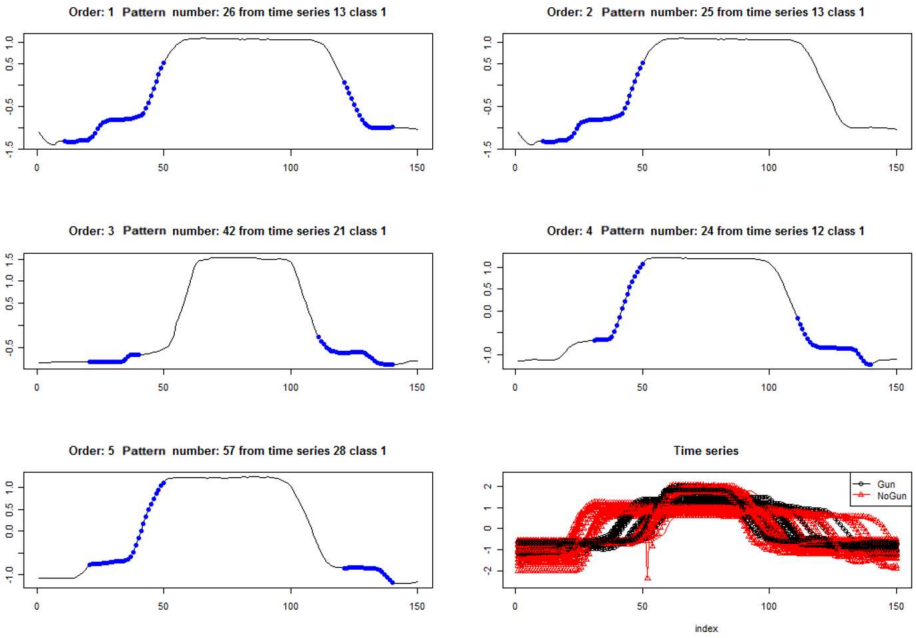


Figure 44. First five important patterns of TS-PD ($w = 20, d = 10, L = 3$) represented by blue dots (Gun-Point dataset), the order of the importance, id of the pattern and the corresponding time series is provided in the titles of the plots, the last is the plot of the training time series.

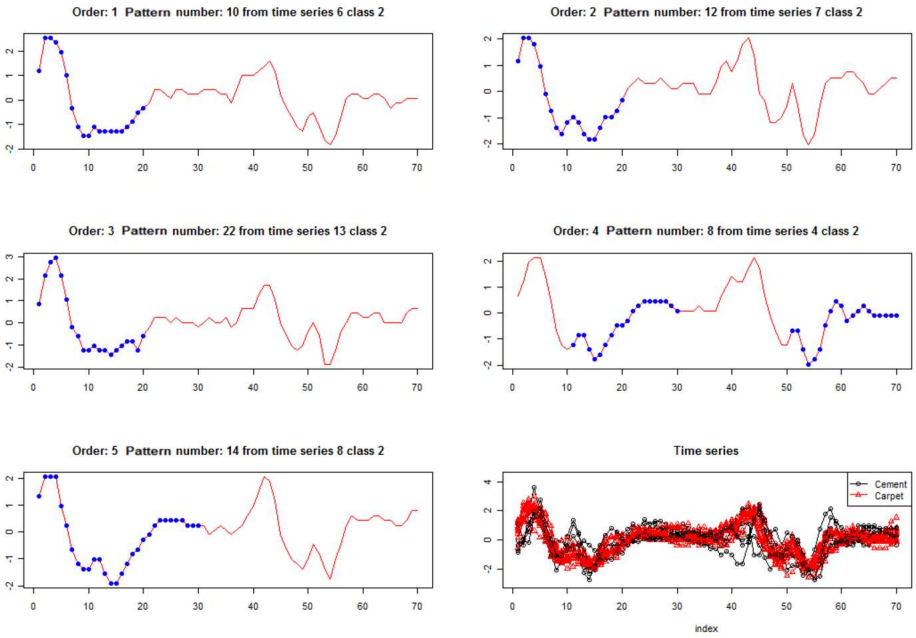


Figure 45. First five important patterns of TS-PD ($w = 20, d = 10, L = 2$) represented by blue dots (Sony AIBO Robot), the order of the importance, id of the pattern and the corresponding time series is provided in the titles of the plots, the last is the plot of the training time series.

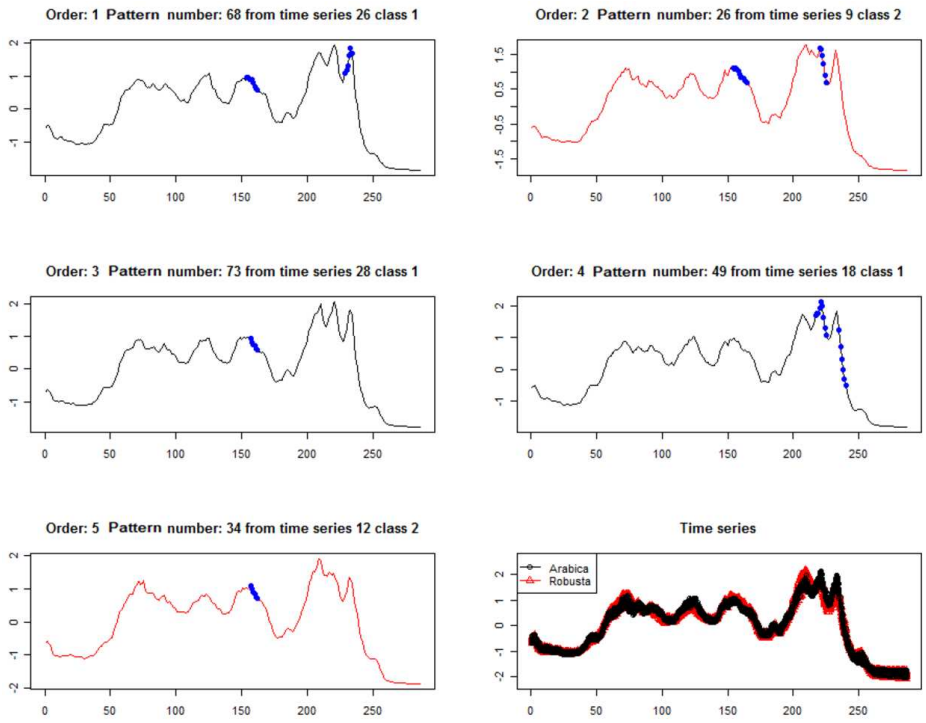


Figure 46. First five important patterns of TS-PD ($w = 6, d = 3, L = 3$) represented by blue dots (Coffee), the order of the importance, id of the pattern and the corresponding time series is provided in the titles of the plots, the last is the plot of the training time series.

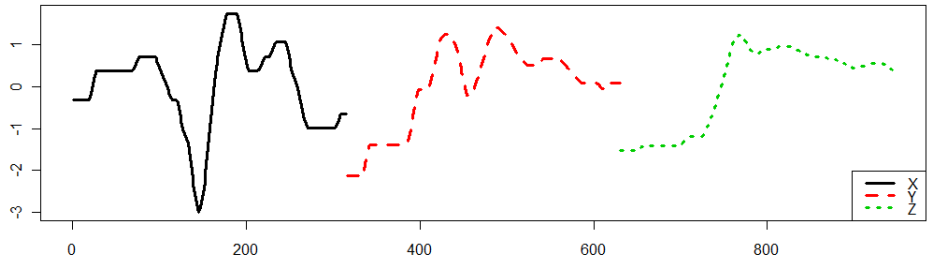


Figure 47. Univariate representation of the accelerometer data.

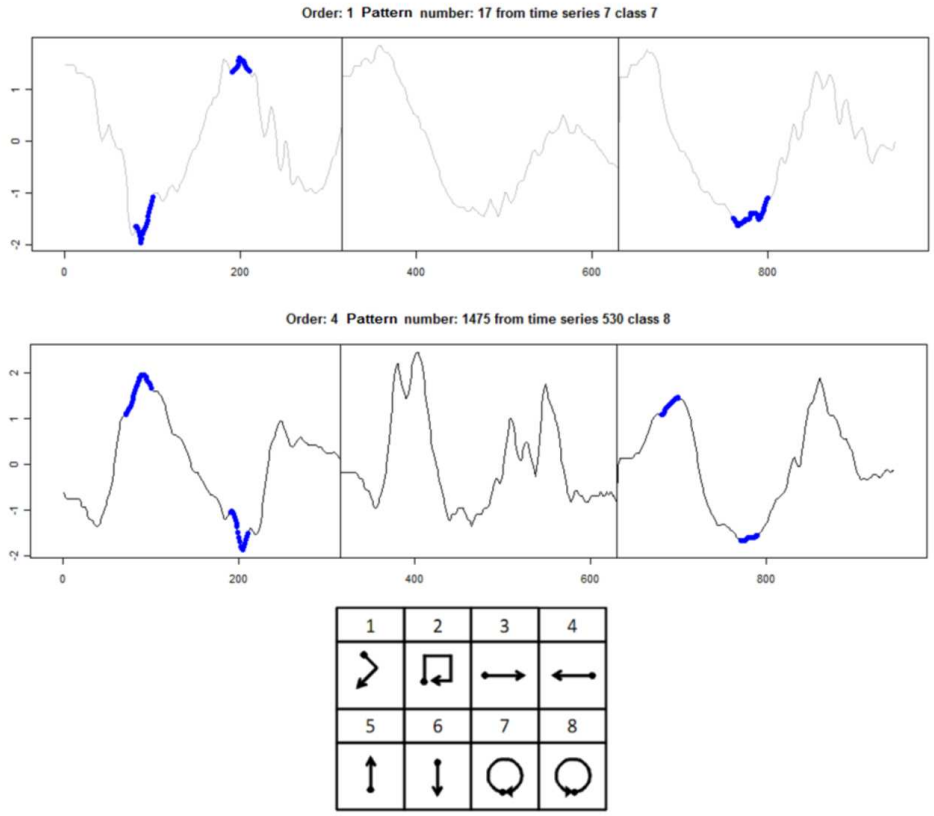


Figure 48. Gesture vocabulary from [5] (bottom). Important patterns are illustrated for two series of class 7 and 8 (top). The segments related to change in the direction during the circular movement are discovered.

CHAPTER 5

MULTIVARIATE TIME SERIES CLASSIFICATION WITH LEARNED DISCRETIZATION

1. Abstract

Multivariate time series (MTS) classification has received great interest over the past decade with the increase in the number of temporal datasets in different fields, such as medicine, finance and multimedia. Similarity based approaches such as nearest neighbor classifiers with Dynamic Time Warping (DTW) are which are successfully used for classification of univariate time series however the similarity computation is unclear for multivariate data since MTS are not only described by the variables but their relation. These approaches lose the relation among the variables of the series by breaking them into multiple univariate time series. Another strategy is to obtain a rectangular representation of MTS by transforming the set of multivariate input sequences to a fixed number of columns using different rectangularization approaches such as principal component analysis. Most of these approaches assume that the variables are numerical however certain variables of the series can be nominal or missing.

In this paper, we follow a different approach and propose a symbolic representation of MTS for classification. MTS observations are first discretized to obtain the symbolic representation. Then, the distribution of the symbols over each time series is computed and used for classification. The relation of the individual variables is taken into account with the proposed representation. Moreover, MTS with nominal and missing values are handled efficiently with tree-based learners. An ensemble learner that scales well with large number of variables and long time series is used. Our approach does not break MTS into multiple univariate series which makes it computationally efficient when compared to

other approaches. Our experiments demonstrate the effectiveness of the proposed approach in terms of accuracy and computation times in both univariate time series and MTS datasets.

Key words: supervised learning, multivariate time series, classification

2. Introduction

Similarity search and classification on time series databases has received great interest over the past decade. Multivariate time series (MTS) classification is a supervised learning problem in which the input consists of a set of training examples and associated class labels, where each example is formed by one or more time series (variables). MTS data is common in different fields, such as in medicine, finance and multimedia. Consider a patient's medical record, there are information in the medical record from multiple sources such as the test values, observations, actions and related responses. This data provides a complex characterization of the patient's status and certain relations inherent in the records may be important. Another example from multimedia applications is the motion capture studies in which position of a set of joints from humans performing a series of task is tracked by markers [93]. Learning scientists are interested in electroencephalography (EEG), which is the recording of electrical activity along the scalp to understand the perceived difficulty for a puzzle solving task in a learning environment. Moreover, in the domain of relational marketing, the behavior of customers is observed through time, and their interactions and responses are represented as MTS. An application illustrated by [25] is about a telecommunication company analyzing the customer's loyalty using the information about the transactions of each customer recorded along the time periods, described by duration, economic value and number of calls of different type (i.e. cell to cell, cell to landline etc.).

There are several approaches proposed to classify MTS. As mentioned by [24], this problem have been studied in different fields such as statistics, signal processing and control theory. We refer reader to [24] for an extensive review of these studies. The most common approach is to obtain a rectangular representation of MTS by transforming the set of multivariate input sequences to a fixed number of columns using different rectangularization approaches [25]. For example, singular value decomposition (SVD) is used by [26–28]. Principal component analysis (PCA) is used for both feature selection and transformation by [29]. Any supervised learner can be trained on the transformed data for classification. Most of these approaches assume that the variables are numerical however certain variables of the series can be nominal or missing.

Another strategy is to modify the similarity based approaches which are successfully used for univariate time series. For example, [5, 94] focus on gesture recognition based on dynamic time warping (DTW) distance. DTW [20] allows a measure of the similarity independent of certain non-linear variations in the time dimension, and is considered as a strong solution for time series problems [58]. Another approach that makes use of the similarity of the series is to use kernel-based classifiers. These approaches find a kernel function determined by pairwise similarities between the variables of MTS. [95] makes use of kernels based on the dynamic time warping for brain activity classification. [25] also proposes a temporal discrete SVM for MTS classification. Overall similarity between the time series are taken into consideration through the objective function with a term that depends on the warping distances [25].

The similarity based approaches are successful for univariate time series. However MTS are not only described by the variables but their relation [30]. Therefore the relation

among the variables are lost if only the similarity between the individual variables are taken into consideration [28]. Moreover as in telecommunication application [25], observations can be nominal (i.e. call type) for which similarity computation is not well-defined.

High dimensionality introduced by multiple variables and longer series is another important challenge for MTS classification. The number of computations required can increase substantially with the increasing number of variables for similarity based approaches. Also, approaches should scale well with the length of the time series, since the number of observations can be large depending on the application.

High-level time series representations are proposed for different data mining tasks to deal with high dimensionality introduced by longer time series [12]. These include Fourier transforms, wavelets, piecewise polynomial models, etc and [96] provides a good summary of these approaches. These representations are proposed for numerical time series. The Symbolic Aggregate approXimation (SAX) [12] and more recently indexable SAX [97] is a commonly used symbolic time series representation because of its simplicity and effectiveness in univariate time series [98]. SAX divides the time series into same length segments and each segment is represented by a symbol based on the mean value of the observations. The number of segments is called ”word size” [12]. This representation is similar to Piecewise Aggregate Approximation (PAA) [99]. However, the symbols are assigned to each segment assuming that the observed values are coming from a Gaussian distribution in SAX. Based on the alphabet size (i.e. number of possible symbols), equiprobable intervals are obtained using the Gaussian distribution assumption and the segments are represented by the symbols. Figure 49 illustrates the idea of symbolic representation on univariate time series data for which word size is 8 and alphabet size is 3.

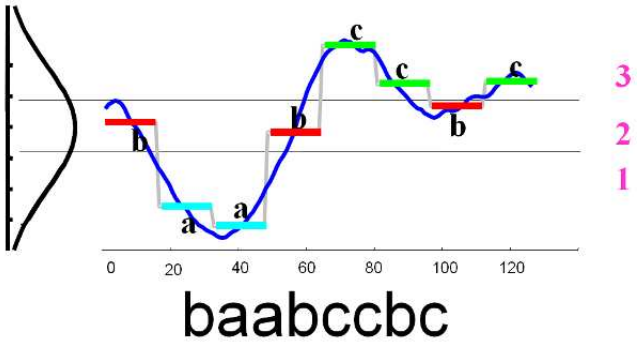


Figure 49. SAX representation with a word size of 8 and alphabet size of 3

There are many time series classifiers based on symbolic representation. [98] proposes a Bag-of-Patterns approach that makes use of SAX representation for univariate time series. For each time series, words are generated by combining symbols using a sliding window approach to capture the patterns over time. Each time series is then represented by the frequency of the words and nearest neighbor classifiers are used to classify test series. For MTS data, two alternative representations illustrated in Figure 50 are commonly considered. MTS with M variables and T observations can discretized to obtain 1D representation using vector quantization approaches similar to the representation obtained for univariate series [100]. Alternatively, each variable of MTS can be discretized and combined to obtain 2D representation of MTS. [101] presents two MTS representations based on SAX to classify physiological data. They generate multivariate words by combining the symbols of each variable at particular time and uses Bag-of-Patterns approach to classify MTS. This representation is called multivariate Bag-of-Patterns and it may capture the relationship between the time series by combining individual representations. However, the length of the words obtained by concatenating the symbols of the variables for each segment may increase substantially as the number of variables increase. This potentially affects the quality

of the information since longer words will carry less information (i.e. curse of dimensionality). Also, the represantation is not sensitive to dilations and translations of the patterns since words are combined at particular time. They also propose stacked Bags-of-Patterns that concatenates the representation of multiple univariate series into a single one. However, this representation does not take the relationship between the variables into account.

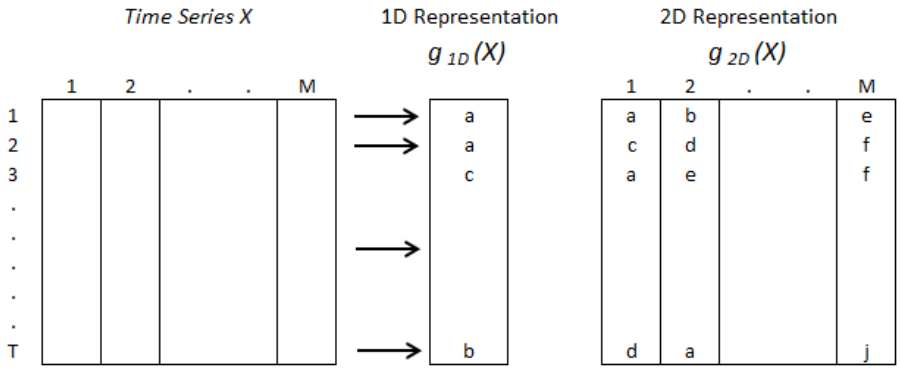


Figure 50. Alternative representations for MTS. MTS with M variables and T observations are mapped to 1D representation by the function g_{1D} (left) or 2D representation in which each variable of MTS mapped to 1D representation by g_{2D} (right). Although the length of the symbolic representation is provided to be the same as T, it can be smaller based on the mapping strategy. Similarly, 2D representation may also have fewer columns than M depending on the mapping.

[93] is another study working on finding predictive patterns for MTS based on SAX representation. This approach considers each variable of MTS separately and generates a 2D representation. Salient variables of MTS are identified first by using certain statistical performance measures efficiently computed using intelligent data structures such as the trie described in [102]. Then predictive patterns are identified for each variable. Rules for classification based on predictive patterns are then searched on the combined set of patterns from individual variables. Identification of patterns on each individual variable without taking other variables into consideration makes this approach greedy since the relation be-

tween the variables may carry the real description of a complex system [30]. Although the patterns are combined later to account for the relationships with this approach, there is a potential to miss a certain pattern that may appear unimportant when time series are considered separately in the first step.

In this paper, we propose an approach to obtain a one-dimensional symbolic representation of MTS (S-MTS) for classification. As opposed to SAX, S-MTS labels each observation instead of the segments of the time series. Observations are discretized in a supervised manner using tree learners to obtain the symbolic representation. To achieve this, each observation is considered to be an instance and the label is assumed to be the same as its time series. Observed value for each variable of MTS and the time index are the features for each instance. In other words, there is no feature extraction, the observed value is used as the feature and we fuse the local information by introducing observation time as a feature of the instance. This way, we form a matrix of these features where rows represent the observation and columns are the observed values and the time of the observation. Tree learners are then trained on this representation to partition the observation space. The terminal node of the trained tree is considered to be a symbol in S-MTS. Figure 52 illustrates the idea of discretization on three univariate time series provided in Figure 51. Partitioning obtained from each tree is used as the symbols. This representation allows S-MTS to consider all variables of MTS simultaneously. Consequently, the distribution of the symbols over each time series is computed and used for classification. The symbols are locality sensitive since the observation times are used as features as schematized in Figure 52(a).

[24] also discusses necessity of alternative representations for MTS classification. A concept called metafeature is introduced and used to represent MTS series by [24]. How-

ever metafeatures must be defined by users in this approach. One of the metafeatures discussed in the paper is based on partitioning of the feature space using Voronoi tiling. Each region of the Voronoi diagram is used as a metafeature. The partitioning is not supervised in this approach and also designing a good metafeature is not an easy task as mentioned by [24].

[103] proposes a similar approach for multidimensional curve classification. They discretize the observations space of each variable separately as in the existing MTS classification methods based on the symbolic representations. Each variable of MTS is partitioned into the rectangular regions of equal dimensions. Then the classification rules are discovered based on the common regions through which only curves of one class pass. Their proposed approach has similarities to [93] in terms of the discretization and rule generation. Since the discretization does not consider the variables simultaneously and rules are discovered based on each individual variable, there is a potential to miss the interaction of the variables. Also both approaches require modifications to handle the categorical and missing data.

There are some similarities of our approach with [56] in terms of the discretization process. [56] proposed Extremely Randomized Clustering Forests (ERC-Forests) for image classification problems. Trees are trained on the features extracted from image patches in a supervised manner and the terminal nodes are considered to be individual clusters. An image is represented as the histogram of the cluster id of the patches segmented from the images (visual codebook) and any supervised learner can be trained on the visual codebook. Unlike [56], we do not generate features. We train the trees to discretize each observation. Although [56] works on images and locality may be important, they do not consider the

location information as a feature during the tree learning. Instead, they proposed saliency maps for identifying important locations of the images.

S-MTS generates a symbolic representation for MTS classification using supervised learning. The interactions between the variables of MTS are handled efficiently with a tree based discretization approach. Moreover, MTS with nominal and missing values are handled efficiently with tree learners. An ensemble learner that scales well with large number of variables and long time series is used. Although the proposed symbolic representation is of the same length as the time series, it does not generate multiple representations for each variable of MTS. Therefore S-MTS scales well with the number of variables of MTS which makes it computationally efficient when compared to other approaches. Our approach can handle MTS examples with different length and it does not require a special rectangularization mechanism since the representation is simply obtained by the frequency of the symbols over the time series. Any learner can be trained on the features representing the frequency of each symbol over each time series in our framework. Moreover, this symbolic representation can be used by any document classification approach as used by [98]. Our experiments demonstrate the effectiveness of the proposed approach in terms of accuracy and computation times in both univariate time series and MTS datasets.

The remainder of this paper is organized as follows. Section 3 provides background and related work. We summarize the problem and describe the framework in Section 4. Section 5 demonstrates the effectiveness and efficiency of our proposed approach by testing on a full set of benchmark univariate time series datasets from UCR time series database [76] and MTS datasets from [104, 105]. Conclusions are drawn in Section 7.

3. Background

Decision tree learners are comprehensible models with satisfactory accuracy and are successfully used in many applications. Univariate trees such as CART [106] and C4.5 [4] split data based on only one variable at each node, and thus are limited to splits that are orthogonal to the variable's axis [107].

Tree ensembles are proposed to avoid from the greedy nature of univariate trees. A random forest (RF) classifier [57] 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

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

where $I(\cdot)$ is an indicator function that equals one if its argument is true and zero otherwise. The predicted class is $\hat{y}(x) = \mathrm{argmax}_c p^c(x)$. The estimates computed from OOB predictions are easily obtained and have been shown to be good estimates of generalization error [57].

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 invariant and robust to outliers, and computations are reasonable even for large data sets.

4. Approach

A multivariate time series, X^n, is an M-variable time series each of which has T observations where x_m^n is the mth variable and $x_m^n(t)$ denotes the observation at time t. Formally, MTS example X^n is represented by $T \times M$ matrix as:

$$X^n = [x_1^n, x_2^n, \ldots, x_m^n, \ldots, x_M^n]$$

where

$$x_m^n = [x_m^n(1), x_m^n(2), \ldots, x_m^n(T)]^{'}$$

There are N training MTS, each of which is associated with a class label y^n, for $n = 1, 2, \ldots, N$ and $y^n \in \{0, 1, 2, ..., C-1\}$. Given a set of unlabeled MTS, the task is to map each MTS to one of the predefined classes. Univariate time series is a special case of MTS where M is equal to one. In the following sections, the definitions assume that all variables of MTS are numerical unless stated otherwise.

4.1. *Time Series Discretization using Tree-Based Classifiers*

We propose a method to discretize MTS using an ensemble of tree learners. Instead of extracting features from each time series, each observation is considered to be an instance in our approach. This is achieved by creating a matrix of instances $D_{NT\times M}$ where

$$D_{NT\times M} = \begin{bmatrix} x_1^1\ x_2^1\ \dots\ x_M^1 \\ x_1^2\ x_2^2\ \dots\ x_M^2 \\ \vdots \\ \vdots \\ x_1^N\ x_2^N\ \dots\ x_M^N \end{bmatrix}$$

which is the concatenation of training MTS as illustrated in Table 18. We assume that the label of each instance is the same as the time series and use a supervised approach to discretize the feature space. The features are obtained by mapping $D_{NT\times M}$ to the feature space $\Phi_{NT\times(2M+1)}$. In other words, the row i of $D_{NT\times M}$ is a set of observations at certain time point t_i. Let d_{ij} be the ij^{th} entry of the matrix $D_{NT\times M}$ which is basically the observed variable j for instance i. Then the row i of $\Phi_{NT\times(2M+1)}$ is:

$$\big[t_i, d_{i1}, d_{i1} - d_{(i-1)1}, d_{i2}, d_{i2} - d_{(i-1)2}, \dots, d_{iM}, d_{iM} - d_{(i-1)M}\big]$$

The first feature is the time index. Then for each variable, we generate two features and concatenate them over the variables. The first one is the observation itself where the difference between consecutive time points is the second feature. Figure 52(a) illustrates the time and observation feature on a 2D plot for the three univariate time series in Figure 51. The difference between consecutive time points captures the information about the

trend in the time series which might be important to classification. Suppose a time series constantly increases after certain time point which will result in positive differences. A tree learner can capture this information if the increase is related with the class. This difference is not available for the first observation of MTS which is assumed to be missing in our representation.

Series	Time Index	Pressure	Temperature	Energy	Class
1	1	2.70	80.50	4.50	1
1	2	3.20	78.40	6.70	1
1	3	4.20	67.90	3.40	1
1	4	8.20	89.50	7.20	1
1	5	8.90	85.70	5.70	1
2	1	10.01	88.00	5.05	0
2	2	11.28	89.94	5.04	0
2	3	12.54	91.19	5.04	0
2	4	13.81	93.25	5.01	0
3	1	16.34	97.54	5.02	1
3	2	17.61	99.66	5.01	1
3	3	18.87	101.60	4.90	1
3	4	20.14	103.54	4.95	1
3	5	22.67	107.43	4.95	1
3	6	21.15	106.50	4.97	1

TABLE 18. Sample database with 3 MTS from 2 classes (1,0 and 1 respectively). There are three observed variables ($M = 3$): pressure, temperature and energy. The series are of length 5,4 and 6 respectively.

If observations are nominal, only the time point of the observation and the observation itself are considered to be features. For both numerical and nominal values, there may exist missing values. Missing values are handled by the tree learners in our approach. Also, the number of observations may differ across different MTS.

After obtaining the features, tree learners are trained on $\Phi_{NT\times(2M+1)}$ assuming that each instance has the same class label as its time series. This way, each instance of $\Phi_{NT\times(2M+1)}$ is mapped to a terminal node of the tree g_j. This RF is referred as $RFins$ (**R**andom **F**orest trained on the **ins**tances). Although the trees of RF without any modi-

fication are unpruned, we restrict the number of terminal nodes of each tree to R which determines the alphabet size in our approach. Second parameter is the number of trees of $RFins$ given by J_{ins}. Each tree of $RFins$ provides a symbolic representation for the time series.

4.2. *Classification*

A Bag-of-Words approach is used to classify the time series based on the symbolic representations described by each tree of $RFins$. However there is no word generation process in the proposed approach. Each symbol is simply considered to be a word and the frequency of the symbols are used to classify the time series. The frequency vector is defined by the number of symbol occurences in the representation. This vector is normalized by the number of observations.

Formally, let $H_j(X^n)$ be the $R \times 1$ frequency vector of the terminal nodes from the representation defined by tree g_j for MTS X^n. We concatenate the frequency vectors from each of the J_{ins} trees of $RFins$ (i.e. $H_j(X^n)$) to obtain the final representation of each time series. This representation is of length $R \times J_{ins}$ assuming that each tree g_j provides R symbols. Figure 53 illustrates the representation of time series based on symbol frequencies $R = 4$. Since each tree of $RFins$ is trained on a random subsample of features and instances, the final representation includes different views of the same time series.

A classifier is then trained on the symbol frequencies computed for each time series. The frequency representation can be large based on the setting of R and J_{ins}. Therefore, a scalable classifier that can handle interactions and correlations such as RF is preferred for this task. This RF is referred as $RFts$ (**R**andom **F**orest trained on the **t**ime **s**eries) for which we train J_{ts} trees.

To classify a test series, the frequency representation is obtained after generating the features and traversing the trees of $RFins$. Then traversing the trees of $RFts$ based on the frequency representation provides the classification result.

5. Experiments and Results

We test our approach on both univariate and MTS datasets. Our algorithm does not require the setting of many parameters and it is robust to the settings. The number of trees trained to obtain the symbolic representation (J_{ins}) and the alphabet size (R) are two important parameters of the algorithm. The levels considered for each parameter are provided in Table 19 for each time series type. Larger levels are introduced for MTS classification in order not to lose potential information with larger number of variables of MTS.

	Time Series	
Parameter	Univariate	Multivariate
J_{ins}	$\{25, 50, 100\}$	$\{50, 100\}$
R	$\{5, 10, 25, 50\}$	$\{50, 100, 200\}$

TABLE 19. Parameter settings of TSBF

RF is insensitive to both the number of trees and the number of candidate attributes scored to potential split a node [57]. J_{ts} can be determined based on the progress of OOB error rates over trees. The number of features evaluated at each node of the tree is set to the default which equals the approximate square root of the number of features. There are $2 \times M + 1$ features for $RFobs$ assuming that all variables are numerical. The number of features are $R \times J_{ins}$ for $RFts$.

To set the parameters of S-MTS for each dataset, the algorithm is run 10 times with different seeds for each J_{ins} and R combinations. Once the final representation from $RFins$ obtained, J_{ts} is set based on the progress of the OOB error rates from $RFts$. The mean

and the standard deviation of OOB error rates from 10 replications are used to determine J_{ts}. In our experiments, the error rates at discrete J_{ts} levels which are multiples of 50 trees are considered. We add more trees to $RFts$ if the mean OOB error rate improves at least one standard deviation from the mean OOB error rate from previous level. Figure 54 (left column) illustrates how the OOB error rate for Non-Invasive Fetal ECG Thorax1 dataset changes as J_{ts} increases. The marginal gain becomes insignificant after certain J_{ts} for all J_{ins} and R combinations. The aim of using such a criterion is to obtain the least complex model.

After setting J_{ts}, R and J_{ins} are chosen based on the size of the representation which is $R \times J_{ins}$. Starting from the smallest $R \times J_{ins}$, we search for the smallest representation providing the best OOB error rate based on the same decision criterion used for setting J_{ts} (i.e. one deviation difference). Different R and J_{ins} settings may provide the same $R \times J_{ins}$. We basically select R and J_{ins} combination providing the minimum mean OOB error rate in such cases. The aim is to obtain a compact representation of the time series. The model with selected parameters is used for the classification of the test time series.

5.1. *Univariate Time Series*

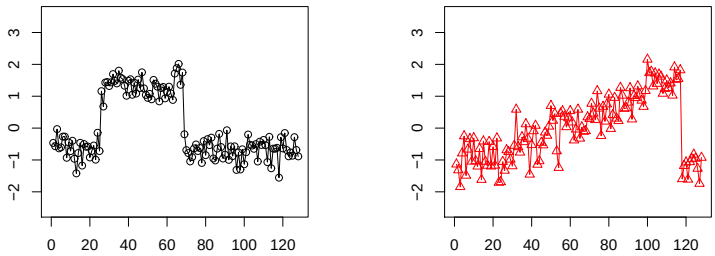
45 univariate time series from [76] are used to illustrate the effectiveness of our approach for univariate time series classification. The dataset characteristics are given in Table 20. This is a good testbed with diverse characteristics such as length of the series, number of classes etc. which enables a comprehensive evaluation.

We compare the error rates on the test data to nearest neighbors (NN) classifiers with DTW. Two versions of DTW are considered: NNDTWBestWin [17] searches for the best warping window, based on the training data, then uses the learned window on the test data,

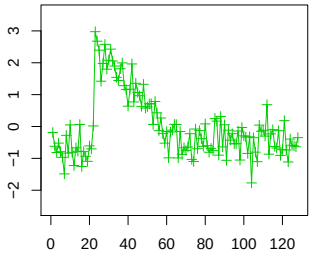
while NNDTWNoWin has no warping window. Note that DTW is a strong solution for time series problems in a variety of domains [58]. The error rates for nearest neighbor classifiers are obtained from [76]. We also compare our approach to NN classifier based on Bag-of-Patterns representation by [98]. This comparison is done on a subset of datasets since [98] reports results on 20 of the datasets. Tables 21 and 22 summarizes the results of each algorithm. Last row compares our classifier based on the number of wins/losses/ties for the algorithm on the column. Our algorithm performs better than NNDTWBestWin and NNDTWNoWin for 32 and 31 of the datasets where it has lower error rates for 12 out of 20 datasets when compared to NN classifier based on Bag-of-Patterns representation by [98]. Moreover, selected model parameters are provided in Tables 21 and 22.

	# of classes	Dataset Size Train	Test	Length
50Words	50	450	455	270
Adiac	37	390	391	176
Beef	5	30	30	470
CBF	3	30	900	128
Coffee	2	28	28	286
ECG	2	100	100	96
Face (all)	14	560	1,690	131
Face (four)	4	24	88	350
Fish	7	175	175	463
Gun-Point	2	50	150	150
Lighting-2	2	60	61	637
Lighting-7	7	70	73	319
OliveOil	4	30	30	570
OSU Leaf	6	200	242	427
Swedish Leaf	15	500	625	128
Synthetic Control	6	300	300	60
Trace	4	100	100	275
Two Patterns	4	1,000	4,000	128
Wafer	2	1,000	6,174	152
Yoga	2	300	3000	426
ChlorineConcentration	3	467	3,840	166
CinC_ECG_torso	4	40	1,380	1,639
Cricket_X	12	390	390	300
Cricket_Y	12	390	390	300
Cricket_Z	12	390	390	300
DiatomSizeReduction	4	16	306	345
ECGFiveDays	2	23	861	136
FacesUCR	14	200	2,050	131
Haptics	5	155	308	1,092
InlineSkate	7	100	550	1,882
ItalyPowerDemand	2	67	1,029	24
MALLAT	8	55	2,345	1,024
MedicalImages	10	381	760	99
MoteStrain	2	20	1,252	84
SonyAIBORobot Surface	2	20	601	70
SonyAIBORobot SurfaceII	2	27	953	65
StarLightCurves	3	1,000	8,236	1,024
Symbols	6	25	995	398
TwoLeadECG	2	23	1,139	82
uWaveGestureLibrary_X	8	896	3,582	315
uWaveGestureLibrary_Y	8	896	3,582	315
uWaveGestureLibrary_Z	8	896	3,582	315
WordsSynonyms	25	267	638	270
Non-Invasive Fetal ECG Thorax1	42	1,800	1,965	750
Non-Invasive Fetal ECG Thorax2	42	1,800	1,965	750

TABLE 20. Characteristics of the univariate time series: number of classes, number of training instances, number of testing instances, and length of time series.

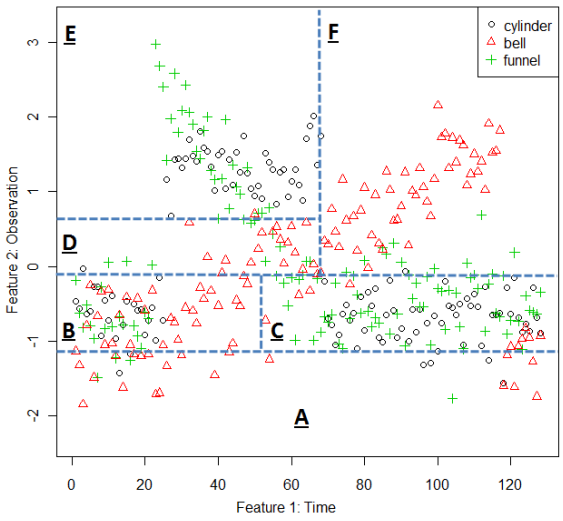


(a) Cylinder (b) Bell

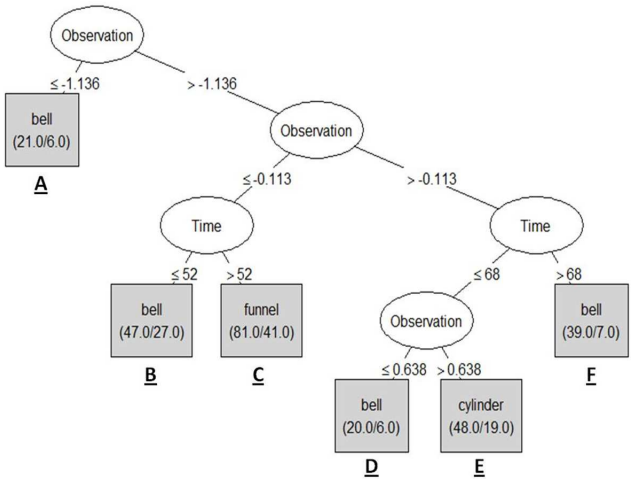


(c) Funnel

Figure 51. One time series of each class from CBF dataset.



(a) Feature Space



(b) Decision Tree

Figure 52. The feature space and the partitions (symbols) from the decision tree

	Tree 1				Tree 2				. . .	Tree J_{ins}			
Series	A_1	B_1	C_1	D_1	A_2	B_2	C_2	D_2	. . .	.	.	.	.
1	0.25	0.33	0.17	0.25	0.00	0.00	0.75	0.25	. . .	0.00	0.25	0.50	0.25
2	0.17	0.66	0.17	0.00	0.00	0.50	0.00	0.50	. . .	0.00	0.33	0.25	0.42
.	.	.	.	.	.	.	.	.	. . .	.	.	.	.
.	.	.	.	.	.	.	.	.	. . .	.	.	.	.
N	0.50	0.50	0.00	0.00	0.25	0.25	0.00	0.50	. . .	0.17	0.33	0.25	0.25

Figure 53. A visual example of the representation based on symbol frequencies. Each column denotes a symbol from a tree of $RFins$ ($R = 4$), and each row denotes a multivariate time series.

				OOB	Test			Nearest Neighbor (NN)DTW		
	J_{ins}	R	J_{ts}	Mean	Mean	Min	Max	BestWin	NoWin	BOP
50Words	100	50	400	0.315	0.286	0.270	0.299	0.242	0.310	0.466
Adiac	100	50	300	0.260	0.241	0.228	0.266	0.391	0.396	0.432
Beef	50	25	200	0.227	0.270	0.200	0.333	0.467	0.500	0.433
CBF	25	50	50	0.030	0.031	0.019	0.048	0.004	0.003	0.013
Coffee	25	10	50	0.021	0.014	0.000	0.036	0.179	0.179	0.036
ECG	25	5	400	0.120	0.168	0.140	0.200	0.120	0.230	0.150
Face (all)	100	10	400	0.124	0.185	0.165	0.198	0.192	0.192	0.219
Face (four)	100	50	200	0.142	0.130	0.114	0.182	0.114	0.170	0.023
Fish	50	50	350	0.141	0.153	0.126	0.177	0.160	0.167	0.074
Gun-Point	25	10	50	0.018	0.027	0.020	0.047	0.087	0.093	0.027
Lighting-2	25	50	50	0.153	0.267	0.213	0.295	0.131	0.131	0.164
Lighting-7	25	10	150	0.247	0.281	0.247	0.315	0.288	0.274	0.466
OliveOil	25	25	50	0.190	0.187	0.133	0.300	0.167	0.133	0.133
OSU Leaf	25	25	400	0.285	0.382	0.355	0.409	0.384	0.409	0.256
Swedish Leaf	25	50	300	0.101	0.086	0.078	0.093	0.157	0.210	0.198
Synthetic Control	100	50	500	0.016	0.022	0.007	0.040	0.017	0.007	0.037
Trace	25	10	50	0.003	0.000	0.000	0.000	0.010	0.000	0.000
Two Patterns	100	5	200	0.000	0.001	0.001	0.002	0.002	0.000	0.129
Wafer	25	5	50	0.000	0.000	0.000	0.000	0.005	0.020	0.003
Yoga	100	50	200	0.063	0.085	0.066	0.109	0.155	0.164	0.170
win/lose/tie over first 20 datasets								13/7/0	13/6/1	12/7/1

TABLE 21. Selected parameters based on OOB error rates. OOB error and test error rates. Error rates for nearest-neighbor classifiers with dynamic time warping distance and Bag-of-Patterns representation with the Euclidean distance

				OOB	Test			Nearest Neighbor (NN)DTW	
	J_{ins}	R	J_{ts}	Mean	Mean	Min	Max	BestWin	NoWin
ChlorineConcentration	50	25	450	0.278	0.326	0.316	0.335	0.350	0.352
CinC_ECG_torso	25	50	200	0.038	0.110	0.096	0.129	0.070	0.349
Cricket_X	50	50	400	0.225	0.289	0.272	0.297	0.236	0.223
Cricket_Y	25	50	400	0.258	0.289	0.274	0.300	0.197	0.208
Cricket_Z	50	50	500	0.254	0.253	0.238	0.267	0.180	0.208
DiatomSizeReduction	50	10	50	0.088	0.055	0.026	0.095	0.065	0.033
ECGFiveDays	100	10	250	0.087	0.182	0.143	0.214	0.203	0.232
FacesUCR	100	50	150	0.107	0.155	0.146	0.163	0.088	0.095
Haptics	25	25	150	0.405	0.501	0.481	0.523	0.588	0.623
InlineSkate	25	25	250	0.493	0.543	0.500	0.571	0.613	0.616
ItalyPowerDemand	50	50	50	0.037	0.037	0.028	0.052	0.045	0.050
MALLAT	100	50	450	0.038	0.052	0.045	0.059	0.086	0.066
MedicalImages	50	25	300	0.230	0.250	0.238	0.263	0.253	0.263
MoteStrain	25	10	250	0.005	0.055	0.044	0.068	0.134	0.165
SonyAIBORobot Surface	25	5	50	0.035	0.187	0.146	0.240	0.305	0.275
SonyAIBORobot SurfaceII	25	5	50	0.081	0.138	0.100	0.189	0.141	0.169
StarLightCurves	25	50	200	0.025	0.023	0.021	0.025	0.095	0.093
Symbols	25	25	50	0.100	0.056	0.030	0.101	0.062	0.050
TwoLeadECG	100	50	250	0.000	0.022	0.017	0.029	0.132	0.096
uWaveGestureLibrary_X	50	50	200	0.182	0.175	0.166	0.183	0.227	0.273
uWaveGestureLibrary_Y	50	50	150	0.225	0.252	0.246	0.257	0.301	0.366
uWaveGestureLibrary_Z	25	50	100	0.238	0.246	0.238	0.256	0.322	0.342
WordsSynonyms	25	25	200	0.369	0.407	0.390	0.426	0.252	0.351
Non-Invasive Fetal ECG Thorax1	25	50	300	0.117	0.107	0.099	0.112	0.185	0.209
Non-Invasive Fetal ECG Thorax2	50	50	200	0.084	0.077	0.074	0.083	0.129	0.135
				win/lose/tie over all 45 datasets				32/13/0	31/13/1

TABLE 22. Selected parameters based on OOB error rates (continued). OOB error and test error rates. Error rates for nearest-neighbor classifiers with dynamic time warping distance.

5.2. *Multivariate Time Series*

We test our proposed approach on datasets from different applications such as speech recognition, activity recognition, medicine and etc. 15 MTS from [76, 104, 108, 109] are used to illustrate the performance of our approach. The dataset characteristics are given in Table 23. We randomly selected train and test samples if there is no default train/test split provided for the datasets. Datasets are described in Section 6 and available in [110].

These datasets are commonly used to evaluate MTS classifiers. However, due to the high number of classes, some studies downsample certain datasets to fewer classes or instances (i.e. [28] uses instances from 25 classes of AUSLAN). Moreover, some algorithms preprocess the data for different purposes such as smoothing or obtaining an appropriate representation (i.e. [28, 30] truncates some datasets to obtain time series of same length). We compare S-MTS to the approaches using whole dataset without any preprocessing.

	# of classes	# of variables	Length	Dataset Size Train	Test	CV	Source
AUSLAN	95	22	45-136	1140	1425		
Pendigits	10	2	8	300	10692	10-fold	[104]
Japanese Vowels	9	12	7-29	270	370		
Robot Failure							
LP1	4	6	15	38	50		
LP2	5	6	15	17	30		
LP3	4	6	15	17	30	5-fold	[104]
LP4	3	6	15	42	75		
LP5	5	6	15	64	100		
ECG	2	2	39-152	100	100	10-fold	[109]
Wafer	2	6	104-198	298	896		
CMU_MOCAP_S16	2	62	127-580	29	29	10-fold	[108]
ArabicDigits	10	13	4-93	6600	2200	×	
CharacterTrajectories	20	3	109-205	300	2558	×	[104]
LIBRAS	15	2	45	180	180	×	
uWaveGestureLibrary	8	3	315	200	4278	×	[76]

TABLE 23. Characteristics of MTS: number of classes, number of variables, length of time series, number of training instances and number of testing instances. Column ”CV” provides if comparison is also done based on the cross-validation. The source of the datasets are in the last column. Test performance is also reported for all datasets.

Most of the studies working on MTS classification follow a different strategy for experimentation which makes the comparison of the approaches difficult. For instance, [25, 111] evaluates the performance using cross-validation. To have a fair comparison with the competitor algorithms, we also follow their experimentation strategy and discuss the performance of the approaches. The datasets for which CV is performed are given in Table 23. Before doing the cross validation, we combine the training and test data to obtain a single dataset. Our cross validation scheme is similar to the one recommended by [112]. We first divide the dataset into k subsets where k changes based on the number of instances. To set the parameters for each fold, each parameter combination is run on the training fold five times and the parameters are set based on the OOB error rates using the same decision criterion discussed in Section 5. Once the parameters are tuned through this process, the classification of the test instances of the fold are performed. We also run the main cross validation five times to obtain a reasonable estimate of the performance of our algorithm.

NN classifiers ($k \in \{1, 3, 5\}$) with DTW distance are considered for comparisons with S-MTS on the test data. Each time series is standardized to have a mean of zero and a standard deviation of one before distance computation. Suppose DTW distance between univariate series x_m^1 and x_m^2 is defined by $DTW(x_m^1, x_m^2)$ then the DTW distance between two MTS, X^1 and X^2, dist(X^1, X^2) is computed as:

$$\text{dist}(X^1, X^2) = \sum_{m=1}^{M} DTW(x_m^1, x_m^2)$$

Table 24 summarizes the results from our cross-validation experiments and reported error rates from other papers. S-MTS performs better when compared to the classification approaches considered by [25]. [111] reports the error rates for nearest neighbor classifiers

with DTW distance. S-MTS outperforms the similarity based approaches for two of the datasets. The best error rate for AUSLAN reported by [24] is from an ensemble of 11 different classifiers trained on the extracted metafeatures. S-MTS performs equally well for this particular dataset. S-MTS and predictive motif discovery approach from [93] provide perfect accuracy for the motion capture dataset.

			[25]		NNDTW (k=3) [111]		Tclass [24]	[93]
	S-MTS	TDVM	SVM_{DTW}	$1NN_{WD}$	NoWin	BestWin	Voting	Motif
Japanese Vowels	0.029	0.034	0.054	0.077				
Pendigits	0.013	0.037	0.066	0.055				
Robot Failure								
LP1	0.095	0.148	0.182	0.182				
LP2	0.355	0.362	0.362	0.404				
LP3	0.223	0.319	0.342	0.383				
LP4	0.056	0.145	0.128	0.137				
LP5	0.263	0.329	0.379	0.348				
ECG	0.147				0.189	0.172		
Wafer	0.011				0.091	0.066		
AUSLAN	0.025						0.021	
CMU_MOCAP_S16	0.000							0.000

TABLE 24. Cross-validation error rates for S-MTS (10 replications). Best cross-validation error rates reported by other MTS classification papers.

The error rates on the test data for S-MTS and nearest neighbor classifier with DTW distance are given in Table 25. The datasets are sorted based on the number of variables to illustrate the effectiveness of S-MTS. S-MTS provides better results for the datasets with larger number of variables where the performance is comparable for the remaining datasets. S-MTS performs equally well ArabicDigits and Japanese Vowels dataset when compared the other studies in the literature.

6. Description of MTS datasets

The description of each dataset is provided to illustrate the range of the application areas for which S-MTS can be employed.

				OOB	S-MTS			NNDTW-NoWin			
	J_{ins}	R	J_{ts}	mean	mean	min	max	k=1	k=3	k=5	Other
CMU_MOCAP_S16	50	50	50	0.000	0.003	0.000	0.034	0.069	0.138	0.172	
AUSLAN	50	200	200	0.022	0.047	0.034	0.060	0.238	0.246	0.222	
ArabicDigits	100	50	500	0.030	0.064	0.062	0.068	0.092	0.075	0.075	0.069 by [113]
Japanese Vowels	50	100	300	0.018	0.026	0.016	0.032	0.351	0.357	0.351	0.032 by [64, 114]
											0.059 by [103]
Robot Failure											
LP1	50	50	50	0.084	0.160	0.120	0.220	0.280	0.240	0.400	
LP2	100	100	300	0.094	0.227	0.167	0.267	0.467	0.567	0.567	
LP3	50	50	100	0.271	0.243	0.167	0.333	0.500	0.533	0.567	
LP4	50	50	50	0.062	0.113	0.067	0.133	0.187	0.160	0.187	
LP5	50	100	450	0.156	0.350	0.310	0.390	0.480	0.530	0.570	
Wafer	100	200	400	0.019	0.024	0.018	0.031	0.023	0.034	0.040	
CharacterTrajectories	100	50	250	0.032	0.040	0.037	0.044	0.040	0.054	0.061	
uWaveGestureLibrary	50	100	450	0.044	0.084	0.081	0.086	0.071	0.083	0.087	
LIBRAS	50	200	200	0.101	0.114	0.100	0.133	0.200	0.217	0.289	
ECG	100	50	200	0.086	0.204	0.190	0.220	0.150	0.190	0.190	
Pendigits	50	50	150	0.050	0.084	0.078	0.088	0.088	0.111	0.125	

TABLE 25. Test error rates for S-MTS (10 replications), nearest-neighbor classifiers with dynamic time warping distance. Best error rates reported by other MTS classification papers.

6.1. *Arabic speech recognition*

[113] introduces a learning method for a graphical probabilistic model for discrete speech recognition. To evaluate their approach, a dataset of size 8800 time series of 13 Frequency Cepstral Coefficients (MFCCs) are created. The experiment involved 44 males and 44 females Arabic native speakers between the ages 18 and 40 to represent ten spoken Arabic digit. Each person has 10 repetitions of each digit. In their experiments, the dataset is divided into two parts: a training set with 75% of the samples and a test set with 25% of the samples. The reported error rates on the test data are 0.0688 and 0.0690 for two approaches proposed by [113].

6.2. *Japanese Vowels*

Utterances of two Japanese vowels by nine male speakers are collected for this dataset [103]. For each utterance, 12-degree linear prediction analysis is applied to obtain a discrete-time series with 12 linear predictive coding (LPC) cepstrum coefficients. This

forms a time series whose length is in the range 7-29 and each point of a time series is of 12 variables for each utterance.

6.3. *Pen-Based recognition of handwritten digits*

[115] create a digit database using a tablet that sends x and y tablet coordinates and pressure level values of the pen at a sampling rate of 100 miliseconds. Only (x, y) coordinate information is used for digit recognition. A MTS of 8 time units with two variables is then used to classify the digits.

6.4. *ECG*

The ECG database comprises a collection of time-series data sets where each file contains the sequence of measurements recorded by one electrode during one heartbeat. Each heartbeat has an assigned classification of normal or abnormal. All abnormal heartbeats are representative of a cardiac pathology known as supraventricular premature beat.

6.5. *Robot execution failures*

This dataset contains force and torque measurements on a robot after failure detection [116]. Each failure is characterized by 15 force/torque samples collected at regular time intervals. Five datasets are introducted for different learning problems for these dataset:

- LP1: failures in approach to grasp position.
- LP2: the failures in transfer of a part.
- LP3: position of part after a transfer failure
- LP4: failures in approach to ungrasp position

- LP5: failures in motion with part

6.6. *Wafer*

The wafer database comprises a collection of time-series data sets where each file contains the sequence of measurements recorded by one vacuum-chamber sensor during the etch process applied to one silicon wafer during the manufacture of semiconductor microelectronics. Each wafer has an assigned classification of normal or abnormal. The abnormal wafers are representative of a range of problems commonly encountered during semiconductor manufacturing.

6.7. *Australian sign language (AUSLAN)*

Australian sign language (AUSLAN) is the language used by the Australian communities with hearing disabilities [24]. Two gloves with magnetic position trackers are used to collect the data. Each hand generates 11 features which consist 3 measures of orientation (roll, pitch, yaw), 3 measures of position (x,y,z) and 5 measures of finger bends. MTS is obtained by combining the features of both hand updated at 100 frames per second. 27 samples of 95 signs results a dataset of size 2565 signs.

6.8. *Brazilian sign language (LIBRAS)*

[117] introduces a dataset to recognize the movement types in LIBRAS (official Brazilian sign language). The hand movement is represented as a bidimensional curve performed by the hand in a period of time. There are 15 movement types and the centroid pixels of the hand are found, which compose the discrete version of the curve with 45 points. This way, a movement is described by a MTS of length 45 time units with 2 variables.

6.9. *Character trajectories*

The data consists of 2858 character samples. Each character sample is a 3-dimensional pen tip velocity trajectory. Multiple, labelled samples of pen tip trajectories recorded whilst writing individual characters. All samples are from the same writer, for the purposes of primitive extraction. Only characters with a single pen-down segment were considered.

6.10. *Motion recognition-CMU_MOCAP_S16*

The CMU Motion Capture database [108] provides MTS datasets that provides the position information of a sets of joints from humans performing certain tasks. We consider the data from Subject 16 since it is one of the few datasets that has sufficient examples for illustration. The task is to predict if subject is walking or running. The data has the information from 62 different joint positions recorded for a varying amount of data [93].

6.11. *Gesture recognition-uWaveGestureLibrary*

A single three-axis accelerometer is used to collect data from eight users to characterize eight gesture patterns. The library, uWaveGestureLibrary, consists over 4000 samples each of which has the accelerometer readings in three dimensions (i.e. x, y and z) [5]. Individual axes are considered in Section 5.1 for the univariate case (the datasets are uWaveGestureLibrary_X, uWaveGestureLibrary_Y, uWaveGestureLibrary_Z). However handling this problem as a MTS classification problem may provide better results by taking the interaction between the individual axes into account.

6.12. *Sensitivity Analysis*

The univariate dataset Non-Invasive Fetal ECG Thorax1 dataset is used to discuss the convergence properties of S-MTS. It provides reasonable number of training and test time series. Moreover, it has large number of classes and longer series compared to the other datasets. Although one dataset is used for the illustration, similar discussion in terms of S-MTS behavior holds for other datasets. The boxplot of OOB and test error rates are provided in Figure 54 to illustrate how S-MTS performs with each setting combination over multiple trees (confidence intervals are for 95% significance level). We add the results for $R = 100$ for the given J_{ins} settings to further investigate the behavior of S-MTS.

The error rates become more stable when time series are represented by larger number of trees and/or more symbols. The decrease in the error rates are not significant after certain number of trees for $RFts$ for most of the settings (around $J_{ts} = 300$ for most of the setting combinations). Similarly, increasing the symbol size does not improve the results significantly after $R = 50$. Similar discussion holds for the test error rates. As given in Table 22, $J_{ins} = 25$, $R = 50$, $J_{ts} = 300$ is used to classify test series based on OOB error rates. The error rates improve slightly with $R = 100$ but the marginal gain is very small where the size of representation increases significantly. This also supports our procedure for parameter selection discussed in Section 5.1.

6.13. *Computational Time Analysis*

Here we empirically evaluate the runtime of S-MTS with different settings of problem characteristics and algorithm parameters. S-MTS is implemented in both C and R Software and our experiments use a Windows 7 system with 8 GB RAM, dual core CPU (i7-3620M

2.7 GHz). Although the CPU can handle four threads in parallel, only a single thread is used.

The overall computational complexity of S-MTS is mainly due to RFs trained to obtain the symbolic representation ($RFins$) and classification ($RFts$). The time complexity of building a single tree in RF is $O(\sqrt{\nu}\eta\beta)$. 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. This makes $O(\sqrt{2M+1}(N \times T)(R-1))$ 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 $O(\sqrt{R \times J_{ins}}N(\log N))$.

StarLightCurves dataset from [76] is used to demonstrate the effect of the parameters J_{ins}, N, T and R on the computation times. For multivariate case, how S-MTS behaves with changing number of variables M, is illustrated on AUSLAN dataset from [104]. For each data set, we randomly selected $\delta \in \{0.2, 0.4, 0.6, 0.8, 1\}$ proportion of number of instances (δ_N), number of observations (δ_T) and number of variables (δ_M). The levels considered for R and J_{ins} are $R \in \{10, 20, 30, 40, 50\}$, $J_{ins} \in \{20, 40, 60, 80, 100\}$. The number of trees in $RFts$ is fixed as $J_{ts} = 500$ since the change in the computation time depends on the RF complexity which is linear with number of trees. Here 10 replications are conducted for each setting combination.

We first illustrate the computational times with changing R and J_{ins} where $\delta = 1$ for remaining parameters. Figure 55 schematizes the average train time and test time with these parameters. Time for training increases linearly with the increase in R and J_{ins} which is consistent with the complexities of $RFins$ and $RFts$. Furthermore, 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 square root of the features to evaluate at each split node. Time for testing increases with the increase in the representation size but this increase is too small since S-MTS is very fast in terms of classification. It only requires traversal of the trees from $RFins$ and $RFts$ after feature representation.

For fixed values of R and J_{ins}, computation times with changing N and T are analyzed. Figure 56 illustrates the mean computation times of S-MTS for $R = 30$ and $J_{ins} = 60$. The runtime increase with the number of training instances or longer series is consistent with $O(N \log N)$ from $RFts$ and $O(N \times T)$ from $RFins$. Testing time is not affected by the number of training series since the computation requires only the traversal of trees in the forest which is independent of the number of training series. On the other hand, it increases linearly as the length of the series increases since symbol assignment is done for each observation of the test series.

S-MTS computation times with changing the number of variables are illustrated in Figure 57 when other parameters are fixed. The runtime increase with the number of variables is consistent with $O(\sqrt{2M+1})$ from $RFins$. Testing time is not affected significantly by the number of variables.

7. Conclusion

A framework is presented to obtain a symbolic representation of MTS (S-MTS) for classification. Observations are discretized in a supervised manner using tree learners to generate the symbolic representation. Since an observation is a row of data where each column is the observed value of each variable of MTS, our supervised approach considers all variables of MTS simultaneously during the discretization process which makes it

computationally efficient when compared to other approaches. This way, the relation between the individual variables is taken into account. Moreover, MTS with nominal and missing values are handled efficiently with tree learners. Once the symbolic representation is generated for MTS, the frequency of the symbols are used to classify MTS. An ensemble learner that scales well with large number of variables and long time series makes S-MTS computationally efficient. Our experiments demonstrate the effectiveness of the proposed approach in terms of accuracy and computation times in both univariate time series and MTS datasets. Although we provide a simple classification approach where frequency of each symbol is used as a feature, potentially better accuracy can be obtained by modifying the bag-of-words approach or use of string similarity kernels on the proposed representation. Moreover, proposed representation can be used for similarity analysis, clustering, and so forth.

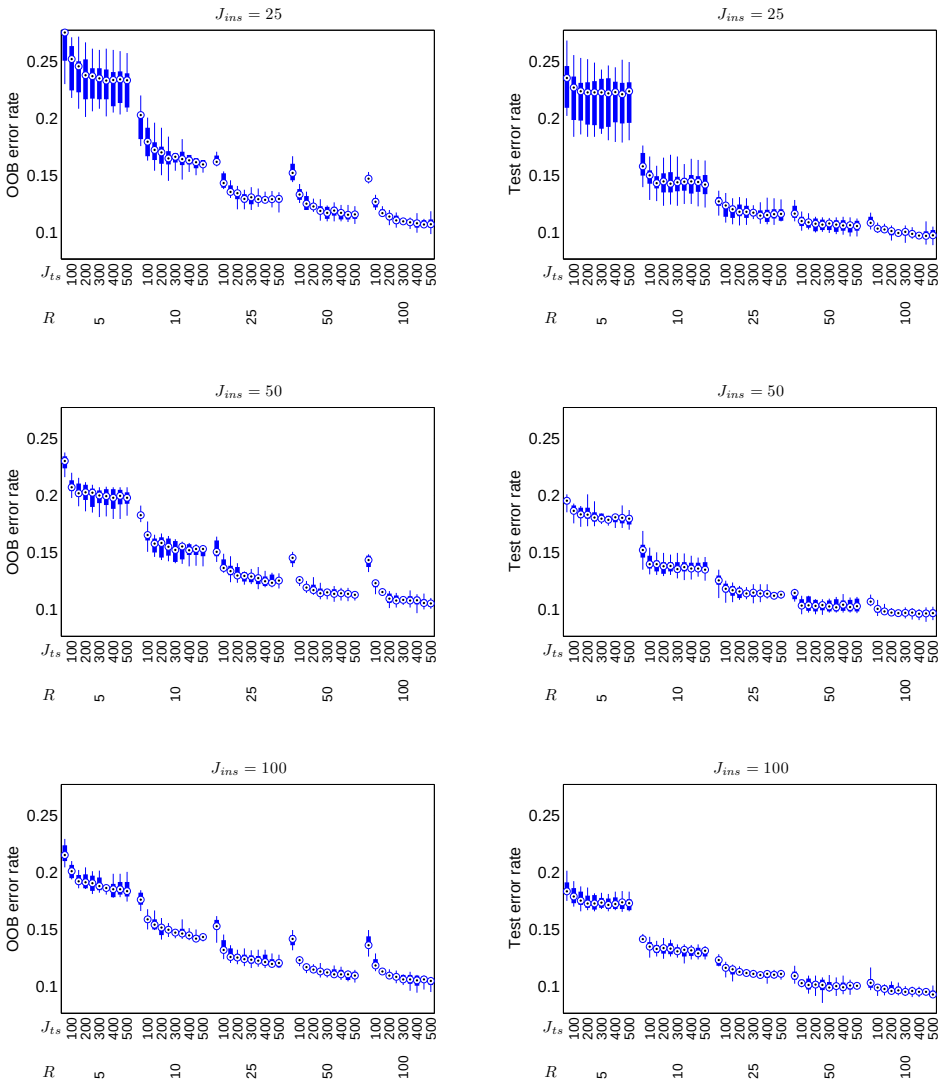


Figure 54. Boxplot of OOB error rates (left column) and test error rates (right column) for each combination setting over multiple trees for Non-Invasive Fetal ECG Thorax1 dataset (10 replications, confidence intervals for 95% significance level). The error rates become more stable when time series are represented by larger number of trees (larger J_{ins}) and/or more symbols (larger R). The decrease in the error rates are not significant after certain number of trees of $RFts$ for most of the settings (around $J_{ts} = 300$). Similarly, increasing the symbol size does not improve the results significantly after $R = 50$. Similar discussion holds for the test error rates.

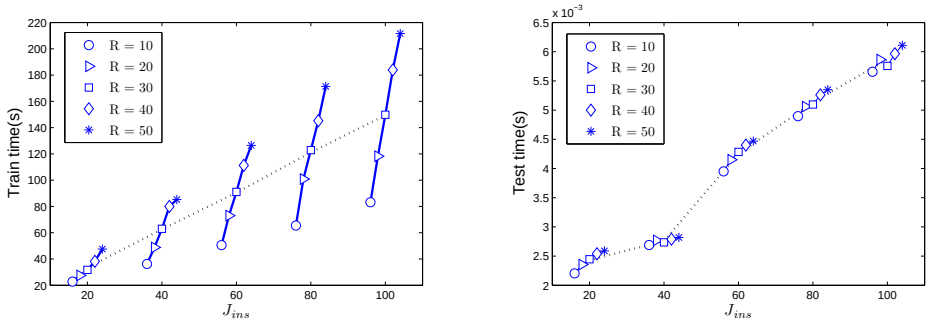


Figure 55. The mean computation times with changing R and J_{ins} ($J_{ts} = 500$ and $\delta = 1$ for remaining parameters). Time for training increases linearly with the increase in R and J_{ins} which is consistent with the complexities of $RFins$ and $RFts$. The train time increases linearly when both R and J_{ins} increase which is an advantage of the proposed approach. Time for testing increases with the increase in the representation size but the increase is too small since S-MTS is very fast in terms of classification.

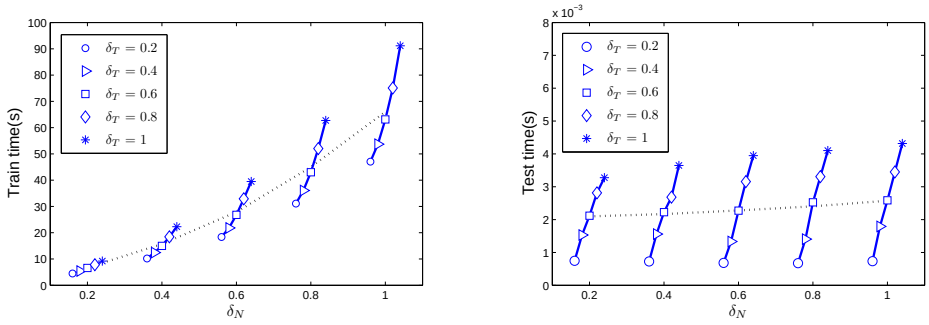


Figure 56. The mean computation times with changing the number of training instances and time series lengths ($R = 30$, $J_{ins} = 60$ and $J_{ts} = 500$). The runtime increase with the number of training instances or longer series is consistent with $O(N \log N)$ from $RFts$ and $O(N \times T)$ from $RFins$. Testing time is not affected by the number of training series but it increases linearly as the length of the series increases.

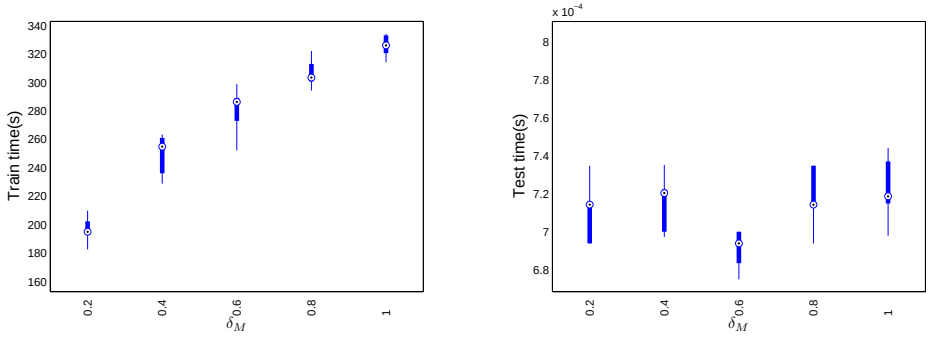


Figure 57. The boxplot of the computation times of S-MTS with changing the number of variables ($R = 50$, $J_{ins} = 50$ and $J_{ts} = 500$). The training time increase with the number of variables is consistent with $O(\sqrt{2M+1})$ from $RFins$. Testing time is not affected significantly by the number of variables.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

1. Conclusions

This dissertation proposes time series representations and methods for supervised time series analysis. The approaches combine new representations that handle translations and dilations of patterns with bag-of-features strategies and tree-based ensemble learning. This provides flexibility in handling time-warped patterns in a computationally efficient way. The ensemble learners provide a classification framework that can handle high-dimensional feature spaces, multiple classes, missing values and interaction between features. The proposed representations are useful for classification and interpretation of the time series data of varying complexity.

In our first study, a framework based on the bag-of-features (BoF) representation is proposed to benefit from the speed and other advantages of feature-based methods to handle the problems for which NN classifiers with DTW distance are challenged. We propose interval selection and local feature extraction strategies to explore time series representation that can handle translation and dilations based on the BoF idea. To capture local information, random subsequences are extracted from each time series and further divided into intervals. The subsequences vary randomly in length and location. The number of intervals that partition a subsequence are fixed so that the interval length varies with the subsequence length. Several features (such as the mean, standard deviation, etc.) are extracted from each interval and these features comprise a row in a new data matrix X (one row for each subsequence). Our local feature generation scheme allows for a novel representation that captures information in a manner similar to DTW. We then label the subsequences and use a supervised approach to summarize the local information. Our supervised approach provides desirable properties for time series classification. It provides fast and efficient time

series representation even with a collection of basic features such as slope, mean and variance from the subsequences. Global features (e.g., autocorrelation of the time series) can also be extracted from the time series and combined with the codebook.

Our second study explores a time series representation that allows for interpretability. We consider a framework for finding important patterns of time series for classification. We focus on finding the segments of the time series that have potential to distinguish the classes. These segments are referred as the regions of interest. Regions of interests are very important to understand the temporal relations. Moreover, they help to reduce the effort in searching for the time segments useful to a classifier. After finding the region of interests for each time series, we generate sequences from these regions. These sequences are referred as patterns. We generate multiple patterns from the time series and find the best matching segments of the time series to these patterns. Then each time series is represented by the distances of the patterns to the best matching segments of the time series. Another classifier is then trained on this representation. A feature selection algorithm on the new feature set allows for finding the patterns that are critical in classification.

Our third study presents a framework to learn a symbolic representation of MTS that is then integrated to produce a new type of MTS classifier. Rather than select intervals from the times series and extract features, the observations in the time series are recursively partitioned into terminal nodes of trees. This leads to a new symbolic representation that is learned based on the class labels. Furthermore, all time series, along with their relationships, are considered simultaneously as the nodes are constructed. Ensembles repeat the process to strengthen the algorithm. This unique representation is then summarized in a high-dimensional codebook. However, another ensemble handles the high dimensional-

ity to generate an effective classifier. there is only one sequence of symbols regardless of the number of variables in a MTS. Our approach can handle MTS examples with different lengths and it does not require a special rectangularization mechanism because the final representation is simply obtained from the frequency of the symbols over the time series.

Applications such as speech recognition, medical diagnosis and gesture recognition are used to illustrate the methods. Experimental results show that the time series representations and methods provide better results than competitive methods on a comprehensive collection of benchmark datasets. Although we present tools that are applicable to and effective for a wide range of important problems, there is a potential to improve and extend the proposed approaches which is further discussed in Section 2.

2. Future Work

2.1. *Local feature extraction*

Our approaches presented in Chapters 3 and 4 are feature-based and require features to be defined. The extracted features are related to the shape of the time series segments (such as the slope of the fitted regression line, the mean and variance over the segment). Proposed approaches may be improved by considering features related to application specific properties. For instance, extracting linear predictive coding (LPC) features from speech signals [118] for a speech recognition task may help to have a better classifier. Since RFs can handle large number of features in a computationally efficient manner, potentially better performance with reasonable computation times might be obtained when more features are added. Furthermore, features can be learned in a manner related to the work in Chapter 5.

2.2. *Absence of the label information*

In many real-world time series classification scenarios, acquiring a large amount of labeled training data is expensive and time-consuming. Semi-supervised learning (SSL) is the machine learning paradigm concerned with utilizing unlabeled data to try to build better classifiers [119]. In general, SSL makes use of both labeled and unlabeled data for training– typically a small amount of labeled data with a large amount of unlabeled data. It falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Therefore SSL can be thought as an attempt to reconcile classification and clustering, two contrasting modes of data analysis [120]. Our proposed approaches are flexible to make use of the unlabeled information since RF enables a proximity measure that can be used for clustering [57].

2.3. *Beyond time series*

Although we mainly focus on time series analysis, many ideas presented in this dissertation can be extended to the spatial domain such as images, trajectories etc. TSBF uses subsequences to classify the time series, as shown in Chapter 3. TSBF can be extended to classification of images by sampling patches (sampling in two dimensions) instead of sampling subsequences. A supervised learner trained on the image patches may provide better representations for images when compared to unsupervised codebooks. The pattern discovery approach presented in Chapter 4 can be used to find the interesting regions of images for classification. Also the same idea can be investigated for object detection. The time series discretization approach for MTS, described in Chapter 5, can be extended to

images and motion video analysis since both images and motion videos can be considered to be special types of multivariate time series data.

2.4. *Similarity kernels*

The similarity of time series based on subsequences might be used to obtain kernel-based classifiers [63]. Subsequences are generated and quantized into symbols. The similarity of the time series is then computed using the string representation of the subsequences. Such a similarity measure, based on the similarity of subsequences, is a distance-based approach. A support vector machine (SVM) [121] with the defined kernel is used to classify the time series. How the similarity of subsequences can be useful for classification was further discussed by [122].

Our proposed approaches can be extended to define kernels similar to the one in [63]. The similarity information from $RFsub$ in Chapter 3 and $RFts$ from Chapters 4 and 5 provide a similarity measure between subsequences and time series, respectively. Such kernels have been obtained from RF models previously [123], but not for time series problems.

REFERENCES

[1] "http://www.markcorbyn.com/work/dissertation/," accessed: June 10, 2012.

[2] M. Mller, "Chapter 4: Dynamic time warping," in *Information Retrieval for Music and Motion*. Springer Berlin Heidelberg, 2007, pp. 211–226.

[3] A. Bosch, X. Munoz, and R. Marti, "Which is the best way to organize/classify images by content?" *Image and Vision Computing*, vol. 25, no. 6, pp. 778 – 791, 2007.

[4] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[5] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uWave: Accelerometer-based personalized gesture recognition and its applications," *Pervasive Computing and Communications, IEEE International Conference on*, vol. 0, pp. 1–9, 2009.

[6] L. Khadra, A. Al-Fahoum, and S. Binajjaj, "A quantitative analysis approach for cardiac arrhythmia classification using higher order spectral techniques," *Biomedical Engineering, IEEE Transactions on*, vol. 52, no. 11, pp. 1840 –1845, nov. 2005.

[7] Y. Kakizawa, R. H. Shumway, and M. Taniguchi, "Discrimination and clustering for multivariate time series," *Journal of the American Statistical Association*, vol. 93, no. 441, pp. pp. 328–340, 1998.

[8] T.-c. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, pp. 164–181, 2011.

[9] C. A. Ratanamahatana, J. Lin, D. Gunopulos, E. Keogh, M. Vlachos, and G. Das, "Mining time series data," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Springer US, 2010, pp. 1049–1077.

[10] K. V. R. Kanth, D. Agrawal, A. E. Abbadi, and A. K. Singh, "Dimensionality reduction for similarity searching in dynamic databases." *Computer Vision and Image Understanding*, pp. 59–72, 1999.

[11] B.-K. Yi and C. Faloutsos, "Fast time sequence indexing for arbitrary lp norms," in *Proceedings of the 26th International Conference on Very Large Data Bases*, ser. VLDB '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 385–394.

[12] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, pp. 107–144, 2007.

[13] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases." Springer Verlag, 1993, pp. 69–84.

[14] K.-P. Chan and A. W. chee Fu, "Efficient time series matching by wavelets," in *In ICDE*, 1999, pp. 126–133.

[15] Y.-S. Jeong, M. K. Jeong, and O. A. Omitaomu, "Weighted dynamic time warping for time series classification," *Pattern Recognition*, vol. 44, no. 9, pp. 2231–2240, 2011.

[16] E. Keogh and S. Kasetty, "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349–371, 2003.

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

[18] K. Ueno, X. Xi, E. Keogh, and D. Lee, "Anytime classification using the nearest neighbor algorithm with applications to stream mining," in *IEEE International Conference on Data Mining (ICDM06)*. IEEE, 2007, pp. 623–632.

[19] Z. Xing, J. Pei, and P. S. Yu, "Early prediction on time series: a nearest neighbor approach," in *Proceedings of International Joint Conference on Artifical Intelligence (IJCAI09)*. Morgan Kaufmann, 2009, pp. 1297–1302.

[20] H. Sakoe, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 43–49, 1978.

[21] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures," *Proc. VLDB Endow.*, vol. 1, pp. 1542–1552, August 2008.

[22] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proceedings of International Conference on Machine Learning (ICML06)*. ACM, 2006, pp. 1033–1040.

[23] A. Mueen, E. J. Keogh, and N. Young, "Logical-shapelets: an expressive primitive for time series classification." in *KDD*, C. Apt, J. Ghosh, and P. Smyth, Eds. ACM, 2011, pp. 1154–1162.

[24] M. W. Kadous and C. Sammut, "Classification of multivariate time series and structured data using constructive induction," *Machine Learning*, vol. 58, pp. 179–216, 2005.

[25] C. Orsenigo and C. Vercellis, "Combining discrete svm and fixed cardinality warping distances for multivariate time series classification," *Pattern Recognition*, vol. 43, no. 11, pp. 3787 – 3794, 2010.

[26] C. Li, L. Khan, and B. Prabhakaran, "Real-time classification of variable length multi-attribute motions," *Knowledge and Information Systems*, vol. 10, pp. 163–183, 2006.

[27] ——, "Feature selection for classification of variable length multiattribute motions," in *Multimedia Data Mining and Knowledge Discovery*. Springer London, 2007, pp. 116–137.

[28] X. Weng and J. Shen, "Classification of multivariate time series using locality preserving projections," *Knowledge-Based Systems*, vol. 21, no. 7, pp. 581 – 587, 2008.

[29] H. Yoon, K. Yang, and C. Shahabi, "Feature subset selection and feature ranking for multivariate time series," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 9, pp. 1186 – 1198, sept. 2005.

[30] Z. Bankó and J. Abonyi, "Correlation based dynamic time warping of multivariate time series," *Expert Systems with Applications*, no. 0, pp. –, 2012.

[31] R. Rahmani and S. A. Goldman, "Missl: Multiple-instance semi-supervised learning," in *Proceedings of International Conference on Machine Learning (ICML06)*. ACM Press, 2006, pp. 705–712.

[32] C. Zhang, X. Chen, M. Chen, S.-C. Chen, and M.-L. Shyu, "A multiple instance learning approach for content based image retrieval using one-class support vector machine," in *IEEE International Conference on Multimedia and Expo (ICME05)*, 2005, pp. 1142–1145.

[33] Q. Zhang, S. A. Goldman, W. Yu, and J. E. Fritts, "Content-based image retrieval using multiple-instance learning," in *Proceedings of International Conference on Machine Learning (ICML02)*. Morgan Kaufmann, 2002, pp. 682–689.

[34] O. Maron and A. L. Ratan, "Multiple-instance learning for natural scene classification," in *Proceedings of International Conference on Machine Learning (ICML98)*. Morgan Kaufmann, 1998, pp. 341–349.

[35] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1619–1632, 2011.

[36] P. Dollár, B. Babenko, S. Belongie, P. Perona, and Z. Tu, "Multiple component learning for object detection," in *Proceedings of European Conference on Computer Vision (ECCV08)*, ser. Lecture Notes in Computer Science, vol. 5303. Springer Berlin / Heidelberg, 2008, pp. 211–224.

[37] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," vol. 2. IEEE Computer Society, 2003, p. 264.

[38] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-based object detection in images by components," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 4, pp. 349–361, 2001.

[39] V. C. Raykar, B. Krishnapuram, J. Bi, M. Dundar, and R. B. Rao, "Bayesian multiple instance learning: automatic feature selection and inductive transfer," in *Proceedings of International Conference on Machine learning (ICML08)*. ACM, 2008, pp. 808–815.

[40] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," in *Proceedings of European Conference on Computer Vision (ECCV06)*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 3954, pp. 490–503.

[41] D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," in *Proceedings of European Conference on Machine Learning (ECML98)*,

ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1998, vol. 1398, pp. 4–15.

[42] Y. Chen, J. Bi, and J. Z. Wang, "Miles: Multiple-instance learning via embedded instance selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1931–1947, 2006.

[43] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence*, vol. 89, no. 1-2, pp. 31–71, 1997.

[44] F. Briggs, R. Raich, and X. Z. Fern, "Audio classification of bird species: A statistical manifold approach," in *Proceedings of International Conference on Data Mining (ICDM09)*. IEEE Computer Society, 2009, pp. 51–60.

[45] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, "Music classification via the bag-of-features approach," *Pattern Recognition Letters*, vol. 32, no. 14, pp. 1768 – 1777, 2011.

[46] C. Harris and M. Stephens, "A combined corner and edge detector," in *The Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[47] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Proceedings of International Workshop on Statistical Learning in Computer Vision (ECCV04)*, 2004, pp. 1–22.

[48] M. Weber, M. Welling, and P. Perona, "Unsupervised learning of models for recognition," in *Proceedings of European Conference on Computer Vision (ECCV00)*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2000, vol. 1842, pp. 18–32.

[49] J. Winn, A. Criminisi, and T. Minka, "Object categorization by learned universal visual dictionary," in *Proceedings of International Conference on Computer Vision (ICCV05)*. IEEE Computer Society, 2005, pp. 1800–1807.

[50] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1475–1490, 2004.

[51] B. Leibe and B. Schiele, "Interleaving object categorization and segmentation," in *Cognitive Vision Systems*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 3948, pp. 145–161.

[52] L. Sörensen, M. Loog, D. Tax, W.-J. Lee, M. de Bruijne, and R. Duin, "Dissimilarity-based multiple instance learning," in *Structural, Syntactic, and Statistical Pattern Recognition*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2010, vol. 6218, pp. 129–138.

[53] Y. Chen and J. Z. Wang, "Image categorization by learning and reasoning with regions," *Journal of Machine Learning Research*, vol. 5, pp. 913–939, 2004.

[54] V. Lepetit, P. Lagger, and P. Fua, "Randomized trees for real-time keypoint recognition," in *Proceedings of Computer Vision and Pattern Recognition (CVPR05)*. IEEE Computer Society, 2005, pp. 775–781.

[55] R. Maree, P. Geurts, J. Piater, and L. Wehenkel, "Random subwindows for robust image classification," in *Proceedings of Computer Vision and Pattern Recognition (CVPR05)*. IEEE Computer Society, 2005, pp. 34–40.

[56] F. Moosmann, E. Nowak, and F. Jurie, "Randomized clustering forests for image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1632–1646, 2008.

[57] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[58] C. Ratanamahatana and E. Keogh, "Three myths about dynamic time warping data mining," in *Proceedings of SIAM International Conference on Data Mining (SDM05)*, vol. 21, 2005, pp. 506–510.

[59] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowl. Inf. Syst.*, vol. 7, pp. 358–386, March 2005.

[60] Y. Yamada, H. Yokoi, and K. Takabayashi, "Decision-tree induction from time-series data based on standard-example split test," in *Proceedings of International Conference on Machine Learning (ICML03)*. Morgan Kaufmann, 2003, pp. 840–847.

[61] L. Ye and E. Keogh, "Time series shapelets: a new primitive for data mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09, 2009, pp. 947–956.

[62] ——, "Time series shapelets: a novel technique that allows accurate, interpretable and fast classification," *Data Mining and Knowledge Discovery*, vol. 22, pp. 149–182, 2011.

[63] P. Kuksa and V. Pavlovic, "Spatial representation for efficient sequence classification," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, aug. 2010, pp. 3320 –3323.

[64] P. Geurts, "Pattern extraction for time series classification," in *Principles of Data Mining and Knowledge Discovery*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2001, vol. 2168, pp. 115–127.

[65] D. Eads, K. Glocer, S. Perkins, and J. Theiler, "Grammar-guided feature extraction for time series classification," in *Proceedings of Conference on Neural Information Processing Systems (NIPS05)*, 2005.

[66] M. Hearst, S. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[67] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Feature-based classification of time-series data," *International Journal of Computer Research*, vol. 10, pp. 49–61, 2001.

[68] J. Rodríguez, C. Alonso, and J. Maestro, "Support vector machines of interval-based features for time series classification," *Knowledge-Based Systems*, vol. 18, no. 4-5, pp. 171–178, 2005.

[69] J. Rodríguez, C. Alonso, and H. Boström, "Boosting interval based literals," *Intelligent Data Analysis*, vol. 5, no. 3, pp. 245–262, 2001.

[70] J. J. Rodríguez and C. J. Alonso, "Interval and dynamic time warping-based decision trees," in *Proceedings of ACM Symposium on Applied Computing (SAC04)*, 2004, pp. 548–552.

[71] J.-J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music," *The Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881–891, 2007.

[72] R. F. Lyon, M. Rehn, S. Bengio, T. C. Walters, and G. Chechik, "Sound retrieval and ranking using sparse auditory representations," *Neural Comput.*, vol. 22, pp. 2390–2416, 2010.

[73] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Semantic annotation and retrieval of music and sound effects," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 467 –476, 2008.

[74] J. Lin and Y. Li, "Finding structural similarity in time series data using bag-of-patterns representation," in *Proceedings of International Conference on Scientific and Statistical Database Management (SSDBM09)*. Springer-Verlag, 2009, pp. 461–477.

[75] M. Casey, C. Rhodes, and M. Slaney, "Analysis of minimum distances in high-dimensional musical spaces," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 5, pp. 1015–1028, 2008.

[76] E. Keogh, Q. Zhu, B. Hu, H. Y., X. Xi, L. Wei, and C. A. Ratanamahatana, "The UCR time series classification/clustering. homepage: www.cs.ucr.edu/~eamonn/time_series_data/," 2011. [Online]. Available: www.cs.ucr.edu/~eamonn/time_series_data/

[77] E. Keogh, L. Wei, X. Xi, S.-H. Lee, and M. Vlachos, "Lb-keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures," in *Proceedings of the 32nd international conference on Very large data bases*, ser. VLDB. VLDB Endowment, 2006, pp. 882–893.

[78] M. Last, A. Kandel, and H. Bunke, *Data Mining in Time Series Databases*. World Scientific, 2004.

[79] R. Caruana, N. Karampatziakis, and A. Yessenalina, "An empirical evaluation of supervised learning in high dimensions," in *Proceedings of International Conference on Machine Learning (ICML08)*. ACM, 2008, pp. 96–103.

[80] H. Deng, M. G. Baydogan, and G. C. Runger, "Sparse multivariate trees," *Technical Report*, 2011.

[81] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, "Sparsity and smoothness via the fused lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 1, pp. 91–108, 2005.

[82] M. G. Baydogan, G. C. Runger, and E. Tuv, "A bag-of-features framework to classify time series," *Technical Report*, 2012.

[83] B. Hidasi and C. Gaspar-Papanek, "ShiftTree: An Interpretable Model-Based Approach for Time Series Classification," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2011, vol. 6912, pp. 48–64.

[84] Z. Xing, J. Pei, P. S. Yu, and K. Wang, "Extracting interpretable features for early classification on time series," in *SDM*, 2011, pp. 247–258.

[85] E. Keogh, X. Xi, L. Wei, and C. Ratanamahatana, "The UCR time series classification/clustering. homepage: www.cs.ucr.edu/~eamonn/time_series_data/," 2006. [Online]. Available: www.cs.ucr.edu/~eamonn/time_series_data/

[86] L. Breiman and A. Cutler, "Random forests," http://www.stat.berkeley.edu/users/breiman/RandomForests/cc_graphics.htm, accessed: 3/14/2012.

[87] M. A. Hall, "Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning," in *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 359–366.

[88] L. Yu and H. Liu, "Feature selection for high-dimensional data : A fast correlation-based filter solution," *Machine Learning*, vol. 20, no. 2, p. 856, 2003.

[89] E. Tuv, A. Borisov, G. Runger, and K. Torkkola, "Feature selection with ensembles, artificial variables, and redundancy elimination," *J. Mach. Learn. Res.*, vol. 10, pp. 1341–1366, December 2009.

[90] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[91] D. Vail and M. Veloso, "Learning from accelerometer data on a legged robot," in *In Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004.

[92] R. Briandet, E. K. Kemsley, and R. H. Wilson, "Discrimination of arabica and robusta in instant coffee by fourier transform infrared spectroscopy and chemometrics," *Journal of Agricultural and Food Chemistry*, vol. 44, no. 1, pp. 170–174, 1996.

[93] A. McGovern, D. Rosendahl, R. Brown, and K. Droegemeier, "Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction," *Data Mining and Knowledge Discovery*, vol. 22, pp. 232–258, 2011.

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

[95] W. Chaovalitwongse and P. Pardalos, "On the time series support vector machine using dynamic time warping kernel for brain activity classification," *Cybernetics and Systems Analysis*, vol. 44, pp. 125–138, 2008.

[96] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "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*. ACM Press, 2003, pp. 2–11.

[97] J. Shieh and E. Keogh, "isax: indexing and mining terabyte sized time series," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 623–631.

[98] J. Lin, R. Khade, and Y. Li, "Rotation-invariant similarity in time series using bag-of-patterns representation," *Journal of Intelligent Information Systems*, pp. 1–29, 2012.

[99] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Trans. Database Syst.*, vol. 27, no. 2, pp. 188–228, Jun. 2002.

[100] P. P. Kuksa, "2d similarity kernels for biological sequence classification," in *ACM SIGKDD Workshop on Data Mining in Bioinformatics*, 2012.

[101] P. Ordonez, T. Armstrong, T. Oates, and J. Fackler, "Using modified multivariate bag-of-words models to classify physiological data," in *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops*, ser. ICDMW '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 534–539.

[102] E. Keogh, J. Lin, and A. Fu, "Hot sax: Efficiently finding the most unusual time series subsequence," in *Proceedings of the Fifth IEEE International Conference on Data Mining*, ser. ICDM '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 226–233.

[103] M. Kudo, J. Toyama, and M. Shimbo, "Multidimensional curve classification using passing-through regions," *Pattern Recognition Letters*, vol. 20, no. 1113, pp. 1103 – 1111, 1999.

[104] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[105] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, M. Creatura, and Del, "Collecting complex activity data sets in highly rich networked sensor environments," in *Proceedings of the Seventh International Conference on Networked Sensing Systems (INSS), Kassel, Germany*. IEEE Computer Society Press, Jun. 2010.

[106] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Wadsworth, Belmont, MA, 1984.

[107] C. Brodley and P. Utgoff, "Multivariate decision trees," *Machine Learning*, vol. 19, no. 1, pp. 45–77, 1995.

[108] "CMU graphics lab motion capture database. homepage: mocap.cs.cmu.edu," 2012.

[109] R. T. Olszewski, "http://www.cs.cmu.edu/~bobski/," accessed: June 10, 2012.

[110] M. G. Baydogan, "Multivariate time series classification. homepage: www.mustafabaydogan.com/files/viewcategory/14-multivariate-time-series-classification.html," 2012.

[111] J. Lin, S. Williamson, K. Borne, and D. DeBarr, *Pattern Recognition in Time Series*, ser. Chapman & Hall/Crc Data Mining and Knowledge Discovery. Taylor & Francis, 2012.

[112] S. Salzberg, "On comparing classifiers: Pitfalls to avoid and a recommended approach," *Data Mining and Knowledge Discovery*, vol. 1, pp. 317–328, 1997, 10.1023/A:1009752403260.

[113] N. Hammami and M. Bedda, "Improved tree model for arabic speech recognition," in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, vol. 5, july 2010, pp. 521 –526.

[114] M. Bicego, E. Pekalska, D. M. J. Tax, and R. P. W. Duin, "Component-based discriminative classification for hidden Markov models," *Pattern Recognition*, vol. 42, no. 11, pp. 2637–2648, 2009.

[115] F. Alimoglu and E. Alpaydin, "Combining multiple representations and classifiers for pen-based handwritten digit recognitio," in *4th International Conference Document Analysis and Recognition (ICDAR '97), 2-Volume Set, August 18-20, 1997, Ulm, Germany, Proceedings*, 1997, pp. 637–640.

[116] L. M. Matos, L. S. Lopes, and J. Barata, "Integration and learning in supervision of flexible assembly systems,," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 202–219, 1996.

[117] D. Dias, R. Madeo, T. Rocha, H. Biscaro, and S. Peres, "Hand movement recognition for brazilian sign language: A study using distance-based neural networks," in *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, june 2009, pp. 697 –704.

[118] L. Deng and D. O'Shaughnessy, *Speech Processing: A Dynamic and Optimization-Oriented Approach*, ser. Signal Processing and Communications. Taylor & Francis, 2003.

[119] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006. [Online]. Available: http://www.kyb.tuebingen.mpg.de/ssl-book

[120] C. Iyigun and A. Ben-Israel, "Semi-supervised probabilistic distance clustering and the uncertainty of classification." in *GfKl*, ser. Studies in Classification, Data Analysis, and Knowledge Organization. Springer, 2008, pp. 3–20.

[121] V. Vapnik, *Statistical learning theory*, ser. Adaptive and learning systems for signal processing, communications, and control. Wiley, 1998.

[122] M. G. Baydogan, G. C. Runger, and E. Tuv, "Time series classification through subsequence similarity," *Technical Report*, 2012.

[123] K. Torkkola and E. Tuv, "Ensemble learning with supervised kernels," in *ECML*, 2005, pp. 400–411.