



AFRL-AFOSR-UK-TR-2020-0030

Modeling Unevenly Spaced Multivariate Time Series with Mixed Variable Types

Mustafa Baydogan
BOGAZICI UNIVERSITESI
GUNEY KAMPUS BEBEK BESIKTAS
Istanbul, 34342
TR

06/23/2020
Final Report

DISTRIBUTION A: Distribution approved for public release.

Air Force Research Laboratory
Air Force Office of Scientific Research
European Office of Aerospace Research and Development
Unit 4515 Box 14, APO AE 09421

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Executive Services, Directorate (0704-0188). Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.					
1. REPORT DATE (DD-MM-YYYY) 23-06-2020		2. REPORT TYPE Final		3. DATES COVERED (From - To) 01 Feb 2017 to 31 Jan 2020	
4. TITLE AND SUBTITLE Modeling Unevenly Spaced Multivariate Time Series with Mixed Variable Types				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER FA9550-17-1-0138	
				5c. PROGRAM ELEMENT NUMBER 61102F	
6. AUTHOR(S) Mustafa Baydogan				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) BOGAZICI UNIVERSITESI GUNAY KAMPUS BEBEK BESIKTAS Istanbul, 34342 TR				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) EOARD Unit 4515 APO AE 09421-4515				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/AFOSR IOE	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-AFOSR-UK-TR-2020-0030	
12. DISTRIBUTION/AVAILABILITY STATEMENT A DISTRIBUTION UNLIMITED: PB Public Release					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT This research into modeling unevenly spaced multivariate time series with mixed variable types was fruitful and produced 8 submitted/published reports. The most significant work in this research is the unified, flexible and interpretable time series representation framework, Rand-TS, proposed by Gorgulu and Baydogan (2020). This approach handles all the challenges introduced in this research. Rand-TS models density characteristics of the time series data of any type as a mixture of time-varying Gaussian distributions using random decision trees and it embeds density information of each time series into a sparse vector. A simple similarity measure based on this efficient representation is also shown to be successful in classification tasks. Rand-TS is a novel framework for both univariate and multivariate time series (MTS) representation, it can work with time series of various length and missing observations, furthermore, it allows using additional customized features. The experimental results show that Rand-TS provides competitive performance with state-of-the-art similarity-based methods in addition to its flexibility and interpretability. This work was submitted to Data Mining and Knowledge Discovery in April, 2020.					
15. SUBJECT TERMS Multivariate, Time Series Analysis, Statistics					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT SAR	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON FRIEND, MARK
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (Include area code) 011-44-1895-616292

“Modeling Unevenly Spaced Multivariate Time Series with Mixed Variable Types”

Apr 30 2020

Name of Principal Investigators (PI and Co-PIs): Mustafa Gokce Baydogan

- e-mail address : mustafa.baydogan@boun.edu.tr
- Institution : Department of Industrial Eng., Bogazici University, Istanbul, Turkey
- Mailing Address : Bogazici Universitesi Endustri Muh. Bolumu 34342 Istanbul / Turkiye
- Phone : +90-212-359 4869
- Fax : +90-212-265 1800

Period of Performance: Feb 01 2017 – Jan 31 2020

Abstract:

In the last decade, the increasing use of temporal data, especially time series data, has initiated a great deal of research and development attempts. Many data sources in different fields, such as in medicine, education and finance, naturally generate time series data (e.g. electrocardiogram (ECG), weekly sales totals, and prices of stocks) and modeling of this data has become important to understand the system generating the time series. Modeling time series data involves various tasks such as forecasting, discovery of underlying driving factors or variables, or understanding the dynamics of the series in terms of its variance and other characteristics. A fundamental problem in time series analysis approaches is how to represent/model the time series data. However modeling the time series data is difficult when:

1. There are hundreds (to thousands) or more variables with thousands (to millions) of observations.
2. They are of different type (e.g. numerical, ordinal and nominal).
3. For multivariate data, the observations are sampled in nonuniform time points and/or the sampling frequency for each variable may not be the same.
4. The relations between the variables are nonlinear. Interaction effects are to be detected, and redundant variables are expected to be linked.
5. There is an uncertainty component to be modeled. Outliers and missing values are common.
6. The temporal and/or spatial effects, and transient effects are important.

Many data sources in different fields, such as in medicine, education and finance, naturally generate time series data with aforementioned characteristics. Hence, modeling the data under these challenges is problematic. The research focus is on the development of algorithms for modeling of the time series under aforementioned critical challenges. During the project, novel tree-based ensemble learning strategies are developed to handle a number of statistical learning tasks for time series directly. Developed strategies provide a research direction for approaches that break from the linear dependency models to potentially foster other promising nonlinear approaches. Moreover, proposed research based on the tree-based learning is flexible in handling any type of sequential data from different applications. Our experiments on sequential data from diverse domains have shown the success of the proposals. In time series data mining domain, there are few studies proposing a generalized view to alternative statistical learning tasks such as forecasting, similarity search, classification and clustering. The most important contribution of this research is the introduction of the generalized framework that can handle multiple important time series data mining tasks under a unified framework. Moreover, all the developed approaches are shared with the research community not only for reproducibility purposes but also foster potential studies based on our research.

Introduction:

Many data sources in different fields, such as in medicine, education and finance, naturally generate time series data. In medical domain, functional magnetic resonance imaging (fMRI) data is an example of time series data [1]. This imaging modality generates time series of 3-dimensional images of the brain. Each 3-d pixel called voxel of the fMRI image is basically time series data. In the field of seismology, seismograms are used to identify seismic events [2]. 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 connectivity under different experimental conditions [3].

The size of a multivariate time series data set can become very large quickly. For example, the medical history of a patient store a time-stamped record of what has happened to the patient since birth. Recently, Electronic Health Record (EHR) systems that allow patient to access his/her important information have become very important. Hundreds (to thousands) of variables from different sources are collected by EHR systems over a lifetime of a patient. Given the average life expectancy of 76.7 years for men and 82.6 years for women at birth for 2009 across the 27 member states of the European Union [4] and their frequency of visits to the hospitals, the number of observations can get very large. There are many variables to monitor in an EHR such as test results, vital signs, medications, etc. and summarization of this information is critical for efficient data analysis. A potential output of this modeling effort is the discovery of clues to current disease states.

Most of the existing time series analysis approaches assume that the observations are numerical. However, different types of data exist in many of the real life applications. For example, Balkan et al. (2015) [5] described an adaptive advertising application where displays were used to show advertisements in a supermarket. The displays can collect anonymous information about the audience with the help of a camera and video analytics software. The aim was to understand the audience dynamics and propose a new ad for the next time period. The researchers received raw data in transactional format. So for each store and display they had the time-stamped information of the person captured by the system on date, start time, viewing time, gender, and age. This transactional database has numeric data which is defined by scale and order. On the other hand, the age of the customers is of ordinal type (e.g. child, young adult, and etc.). Ordinal data has order, but it is without scale. Also, nominal data which has no order or scale (e.g. gender) exists in this database. In order to determine the next ad to show, one may want to forecast the characteristics of the people looking at the display in the next time period. In other words, if male children are expected to be more, an ad from toy category would be more appropriate. Forecasting with mixed variable types requires approaches to handle the numeric, ordinal and nominal variables.

Traditional time series analysis approaches assume that observations are sampled at the same frequencies for each variable. However, sampling rate can be different in an experiment where the data is collected from multiple sources. For example, to quantify the cognitive load, multiple sensor (EEG, eye tracker, pressure, skin conductance) collected under different experimental conditions in a learning environment (Stacey et al., 2013). Each sensor has its own sampling frequency and some sensors (e.g. eye tracker) generates missing values when there is a problem with data acquisition (e.g. eyes are out of camera focus). Modeling the time series data from multiple sources is problematic as the number of observations are not the same. Segmenting the data into equal time intervals and feature extraction are proposed to obtain a rectangular structure however this adds an important parameter to the approaches. How to extract features is a research topic itself. This challenges the modeling of the multivariate series.

The time series data collected from real life applications usually exhibits nonlinear relations between the variables. For example, human brain is a complex nonlinear system showing complicated emergent properties [7]. Therefore, working with fMRI data requires the consideration of nonlinear time series dependencies [8] to discover actual connectivity relations of the brain. An absence of such relationship between certain regions of the brain may be considered to be an abnormality caused by a brain disease. Hence, nonlinear modeling of fMRI time series is required to identify the potential problems of the brain [9]. Moreover, redundancy is another important problem in modeling of the

time series data. If one (or more) independent variable is highly correlated with another independent variable, then determining the separate influence of each variable may be problematic. fMRI data is a good example in explaining this phenomenon since voxels that are temporally close to each other is expected to be highly correlated. Therefore, there is a need for linking the redundant variables and their effects in the models.

High-dimensionality and the multiple data types are not the only problem challenging the modeling of the time series but inherent noise and non-stationarity also challenge the traditional approaches. The noise is commonly described by the unavailability of complete information from past behavior. This unknown information might lead to the risk of model over-fitting which degrades the performance of the model or difficulty in obtaining stable models. With an ever increasing amount of information available for modeling, it is common for the data to be contaminated. The temporal and/or spatial effects, and transient effects are important and these effects are considered in the context of the non-stationarity which is about the changes in distribution of the time series observations over time. Obviously, different effects may be important at different time periods. Modern stationary models need to get more and more complex (e.g. markov-switching GARCH models, models with long memory, etc.) to handle the non-stationarity in a stationary way. Handling these models may not be easy since they require mathematical, statistical and computational foundations. This problem becomes more evident as the dimensions gets larger.

Time series modeling is a well-studied problem, especially in economics [9]. Researchers develop algorithms to handle the aforementioned challenges. However, each study focuses on a particular problem. In modeling the time series with numerical observations (regular intervals), the most common methods are vector autoregressive models (VAR models) and dynamic factor models (DFMs) [10]. These strategies have problems when the number of dimensions and observations get larger. To handle the problem of high-dimensionality, penalized regression approaches such as Graphical Lasso [11] are proposed to model the covariance structure in the data. These approaches have been very successful in modeling the time series with numerical observations however the time between the observations should be the same in order to employ these strategies. Most of the approaches assume a linear dependence within the time series observations in one dimension (autocorrelation). For the multivariate data, the dependence between the multiple variables (e.g. covariance) is also assumed to be linear by many methods however most of the real-world time series data exhibits a nonlinear behavior. Also, the definition of covariance, autocorrelation or linearity is not clear when the observations are of different type. The continuous variable behaves nonlinearly conditional on a categorical time series in many contexts. Traditional regression strategies are modified to handle the problems in modeling [12, 13] however the parameter estimation is problematic for these approaches when the number of variables is large. Also, these approaches assume that observations are sampled at the same frequencies. Finally, most of the studies in the literature have problems with the outliers or missing values and handling these problems without breaking the relations between the variables is a challenging research topic.

To summarize, existing research focuses on particular problem and makes assumptions about the remaining challenges. There has been no research on a unified framework to handle the challenges and its implications in modeling the data of this type. Hence, there is a tremendous opportunity to contribute analytical methods and frameworks to model the complex time series data.

The research focus is on the development of approaches that allow for modeling of the time series data under these critical challenges. This is achieved by building robust time series representations that can handle such complex time series data and characterizing the representations using statistical and machine learning approaches. Recent advances in the field of data mining allow for robust approaches to handle different types of non-time series data. Adaption of these approaches for time series data is considered to resolve aforementioned problems. The performance of the proposed approaches are evaluated on time series data from different domains such as education, medicine and marketing. The aim of this evaluation is to demonstrate the robustness of the proposed mechanisms in applications from various domains.

The outlined research is highly interdisciplinary. Especially, the outputs of the proposed mechanisms

are of utmost importance for experts from various domains. Real-world data is dirty, inherently nonlinear and multivariate. Consequently, the problem of fusing multiple sources of information about common entities is frequent for many interdisciplinary applications in commercial and government organizations. Thus, proposed research will have broader impacts on cross disciplines such as engineering, education and medicine. Also, each task is characterized by different requirements depending on the discipline. For example, a biologist is interested in clustering the genes that are functionally similar where an industrial engineer is interested in identifying the anomalous behavior in the manufacturing environment. Hence, development of the proposed modeling framework will also help practitioners from different areas with multiple learning objectives.

Brief Summary of the Experiments, Results and Discussion

Some part of the proposed research has been described in refereed journal articles and the others are either under review or revision. This section summarizes the essence of the developed strategies and refer reader to relevant articles in the list of publications for the details. They are also provided as attachment to this report. The summary of the developed strategies can be summarized as follows in chronological order:

In order to deal with the challenges 2, 4, 5 and 6 mentioned in the abstract, an autoregressive tree-based ensemble approach that can model the nonlinear behavior embedded in the time series with the help of tree-based learning. Multivariate autoregressive forest, namely mv-ARF, is a nonparametric vector autoregression approach which provides an easy and efficient representation that scales well with large datasets. An error-based representation based on the learned models is the basis of the proposed approach. This is very similar to time series kernels used for multivariate time series classification problems. We test mv-ARF on MTS classification problems and show that mv-ARF provides fast and competitive results on benchmark datasets from several domains. Furthermore, mv-ARF provides a research direction for vector autoregressive models that breaks from the linear dependency models to potentially foster other promising nonlinear approaches. The details of the proposed approach are available in Tuncel and Baydogan (2018). Please note that this work is published in Pattern Recognition which is one of the top journals in the area of machine learning.

The challenges with different data types (challenge 2 in the abstract) are addressed by a novel data adaptive representation approach for supervised learning. Biological sequences such as DNA and protein are a class of categorical sequences. In machine learning, categorical sequences are generally mapped to a lower dimensional representation for learning tasks to avoid problems with high dimensionality. The proposed method, namely SW-RF (sliding window-random forest), is a feature-based approach requiring two main steps to learn a representation for categorical sequences. In the first step, each sequence is represented by overlapping subsequences of constant length. Then a tree-based learner on this representation is trained to obtain a bag-of-words like representation which is the frequency of subsequences on the terminal nodes of the tree for each sequence. After representation learning, any classifier can be trained on the learned representation. A lasso logistic regression is trained on the learned representation to facilitate the identification of important patterns for the classification task. Our experiments show that proposed approach provides significantly better results in terms of accuracy on both synthetic data and DNA promoter sequence data. Moreover, a common problem for microarray datasets, namely missing values, is handled efficiently by the tree learners in SW-RF. Although the focus of this paper is on biological sequences, SW-RF is flexible in handling any categorical sequence data from different applications. The details of the proposed approach are available in Cakin et al. (2018). This work is a joint work with the research group of Prof. Jing Li from Arizona State University and Prof. Na Zou from Texas A&M University and it is published in the Journal of Healthcare Informatics Research.

To handle the categorical series for clustering tasks (challenges 2 and 5 in the abstract), a novel model-based approach is introduced by nTreeClus. Proposed approach is at the intersection of the concepts behind existing approaches, including Decision Tree Learning, k -mers, and autoregressive models for categorical time series, culminating with a novel numerical representation of the categorical sequences. This new representation is used to perform clustering considering the inherent patterns in categorical time series. Empirical evaluation using synthetic and real datasets, protein

sequences and categorical time series, declares that nTreeClus is competitive or superior to most of the state-of-the-art methods. Jahanshahi and Baydogan (2019) provides the details about the approach. Please note that this work was submitted to Data Mining and Knowledge Discovery in September 2019 and still under review. Data Mining and Knowledge Discovery is considered A-journals in the field of data mining in the U.S.

To address the third challenge in the abstract, a framework to learn the regions of the time series that have potentially representative patterns is proposed in Gorgulu et al. (2019). These representative patterns, so called shapelets, are defined to be the time series subsequences that discriminates the class variable. Ensemble-Based Fast Shapelet Approximation (EFSA) discovers the regions of the time series that have potentially representative patterns related to the classification tasks. These representative patterns, so called shapelets, are defined to be the time series subsequences that discriminates the class variable. A novel piecewise approximation scheme is developed to identify shapelets in EFSA. Experimental results show that EFSA provides faster, more accurate and interpretable results than competitive methods. Gorgulu and Baydogan (2019) provides the details about the approach. Please note that this work was submitted to Data Mining and Knowledge Discovery in 2019 and received a weak reject decision in December, 2019. Revised paper is to be submitted to Pattern Recognition by June of 2020.

Machine learning applications dealing with unstructured data (like described by the challenges 4 and 6) can deal with learning by a transformation to a new structure. Multiple instance learning can deal with the problems in such cases. In other words, time series data can be represented with multiple instance representation. Yuksekgonul et al. (2019) proposes a simple model that provides permutation invariant maximally predictive prototype generator from a given dataset, which leads to interpretability of the solution and concrete insights to the nature and the solution of a problem. Our aim is to find out discriminative patterns in the unstructured data and simultaneously learn a linear classifier for multiple instance learning (MIL). Our experiments on classical MIL benchmark datasets demonstrate that proposed framework is an accurate and efficient classifier compared to the existing approaches. Extensions and details of the proposed approach is available in Sivrikaya et al. (2020) which is submitted to Pattern Recognition Letters.

As forecasting is problematic under the challenges 1, 2, 4 and 6 mentioned in the abstract, a novel time series forecasting approach Tree-Based Moving Average (TBMA) is developed by Sel and Baydogan (2020). With the help of using the tree-based ensemble strategy, the proposed model deals with a large number of predictors without sacrificing accuracy. Our comprehensive experiments show that TBMA provides competitive results compared to benchmark models in point and probabilistic forecasting of the electricity load of Turkey. The details are available in the manuscript submitted to "Applied Energy". Moreover an R package implementing the proposed mechanisms is developed in order to make the research outcomes publicly available on <https://cran.r-project.org/web/packages/tbma/index.html>.

I think most significant work in this research is the unified, flexible and interpretable time series representation framework, Rand-TS, proposed by Gorgulu and Baydogan (2020). This approach handles all the challenges introduced in this research. Rand-TS models density characteristics of the time series data of any type as a mixture of time-varying Gaussian distributions using random decision trees and it embeds density information of each time series into a sparse vector. A simple similarity measure based on this efficient representation is also shown to be successful in classification tasks. Rand-TS is a novel framework for both univariate and multivariate time series (MTS) representation, it can work with time series of various length and missing observations, furthermore, it allows using additional customized features. The experimental results show that Rand-TS provides competitive performance with state-of-the-art similarity-based methods in addition to its flexibility and interpretability. This work is submitted to Data Mining and Knowledge Discovery in April, 2020.

The fundamental innovation in this project is modeling the time series data under the aforementioned critical challenges. Until this research, there was no unified framework to handle all the mentioned problems together. The modeling used, namely tree-based ensembles for time series modeling, is novel relative to related work in fields such as medicine, finance and multimedia. The ensemble framework combining both serial and parallel ensembles of decision trees is proposed to learn time

series models. The ensembles can handle a number of learning tasks directly (e.g., supervised learning, clustering). With all these in perspective, novel ensemble learning strategies for the modeling the complex time series data. Tree-based ensembles are robust to the few parameters involved in the algorithms [14]. Consequently, they can be applied in the vast majority of applications with minimal performance degradation. This contrasts with alternative time series modeling approaches where a large number of parameter settings can be critical. The developed strategies are developed using scripting languages R and Python and shared online to facilitate the future research. For all the reasons aforementioned, the proposed research makes original and innovative contributions to the literature.

References:

- [1] L Harrison, W.D Penny, and K Friston. "Multivariate autoregressive modeling of fMRI time series". In: *NeuroImage* 19.4 (2003), pp. 1477 –1491.
- [2] Yoshihide Kakizawa, Robert H. Shumway, and Masanobu Taniguchi. "Discrimination and Clustering for Multivariate Time Series". In: *Journal of the American Statistical Association* 93.441 (1998), pp. 328–340.
- [3] Stacey Schink Joseph, Mustafa Gokce Baydogan, and Robert Atkinson. *Measuring Cognitive Load: A Comparison of Self-report and Physiological Methods*. Tech. rep. Arizona State University, 2013.
- [4] Mortality and life expectancy statistics.
http://epp.eurostat.ec.europa.eu/statistics_explained/index.php/Mortality_and_life_expectancy_statistics. accessed: September 10, 2013.
- [5] Sule Balkan, Mustafa Baydogan, and Sanjay Addicam. "Adaptive Advertisement Recommender Systems for Digital Signage". In: *2015 Americas Conference on Information Systems*. 2015.
- [6] Zbigniew J. Kowalik, Andrzej Wrobel, and Andrzej Rydz. "Why does the human brain need to be a nonlinear system?" In: *Behavioral and Brain Sciences* 19 (02 June 1996), pp. 302–303.
- [7] J.D. Ramsey et al. "Six problems for causal inference from fMRI". in: *NeuroImage* 49.2 (2010), pp. 1545–1558.
- [8] Yang Yu et al. "Functional connectivity-based signatures of schizophrenia revealed by multiclass pattern analysis of resting-state fMRI from schizophrenic patients and their healthy siblings". In: *BioMedical Engineering OnLine* 12.1 (2013), pp. 1–13.
- [9] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. "Multivariate Time Series Analysis". In: *Time Series Analysis*. John Wiley and Sons, Inc., 2008, pp. 551–595.
- [10] James Douglas Hamilton. *Time series analysis*. Vol. 2. 1994.
- [11] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Sparse inverse covariance estimation with the graphical lasso". In: *Biostatistics* 9.3 (2008), pp. 432–441.
- [12] Bent Jørgensen et al. "State-Space Models for Multivariate Longitudinal Data of Mixed Types". English. In: *The Canadian Journal of Statistics* 24.3 (1996), pp. 385–402.
- [13] Ginger M. Davis and Katherine B. Ensor. "Multivariate Time-Series Analysis With Categorical and Continuous Variables in an Lstr Model". In: *Journal of Time Series Analysis* 28.6 (2007), pp. 867–885.
- [14] Leo Breiman. "Random forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

List of Publications and Significant Collaborations that resulted from your AOARD supported project:

a) Papers published in peer-reviewed journals,

- Tuncel, K.S. ve Baydogan, M.G., 2018, "Autoregressive forests for multivariate time series modeling," *Pattern Recognition*, 73, 202-215 USA
- Cakin, H., Gorgulu, B., Baydogan, M.G., Zou, N., Li, J., 2018, "A Data Adaptive Biological Sequence Representation for Supervised Learning", *Journal of Healthcare Informatics Research*, vol. 2, no.4, pp. 448-471.

b) Papers published in peer-reviewed conference proceedings,

- Yuksekgonul, M., Sivrikaya E.S., Baydogan, M.G., 2019, "Learning Maximally Predictive Prototypes in Multiple Instance Learning", accepted Sets & Partitions Workshop at NIPS'19, International Conference on Neural Information Processing Systems, available online: <https://arxiv.org/abs/1910.00965>

c) Conference presentations without papers,

- Tuncel, K.S. ve Baydogan, M.G., "Autoregressive forests for multivariate time series modeling," 2017, INFORMS Annual Meeting, Houston, TX, USA
- Gorgulu, B., Baydogan, M.G., Yucel, G., "Ensemble-Based Fast Shapelet Approximation", 2018, INFORMS Annual Meeting, Phoenix, AZ, USA
- Gorgulu, B., Baydogan, M.G., "Time-Series Classification Using Randomized Trees", presented in 2019 INFORMS Annual Meeting, Seattle, WA, USA
- Jahanshahi, H., Baydogan, M.G., "nTreeClus: A model-based partitioning clustering of sequential data", presented in 2019 INFORMS Annual Meeting, Seattle, WA, USA
- Yuksekgonul, M., Sivrikaya E.S., Baydogan, M.G., 2019, "Learning Maximally Predictive Prototypes in Multiple Instance Learning", presented in Sets & Partitions Workshop at NIPS'19, Vancouver, Canada

d) Manuscripts submitted but not yet published:

- Jahanshahi, H., Baydogan, M.G., "nTreeClus: A model-based partitioning clustering of sequential data", submitted to *Data Mining and Knowledge Discovery* on 18th of September, 2019
- Sivrikaya E.S., Yuksekgonul, M., Baydogan, M.G., 2020, "Learning Maximally Predictive Prototypes in Multiple Instance Learning" submitted to *Pattern Recognition Letters*, on 14th of April, 2020
- Sel, B., Baydogan, M.G., "A tree-based moving average approach for electricity load forecasting", submitted to *Applied Energy* on 2nd of April, 2020
- Gorgulu, B., Baydogan, M.G., "Time-Series Classification Using Randomized Trees", submitted to *Data Mining and Knowledge Discovery* on 29th of April, 2020

e) Publications in preparation:

- Gorgulu, B., Baydogan, M.G., Yucel, G., "Ensemble-Based Fast Shapelet Approximation", submitted to *Data Mining and Knowledge Discovery* on 10th of September, 2019. Rejected - Under revision.

f) Interactions with industry or with Air Force Research Laboratory scientists

- I have met Jeremy Jordan (Jordan, Jeremy D Maj USAF AETC AFIT/ENC Jeremy.Jordan@afit.edu) to discuss the status and future plans for the project on July 31st, 2017 in Baltimore.
- I have made collaborations with the research group of Prof. Jing Li from Arizona State University and Prof. Na Zou from Texas A&M University. We have published a paper out of this collaborations
- The work related to electricity load forecasting (submitted to *Applied Energy*) is developed with a Turkey based start-up company during 2019, Algopoly. Algopoly develops machine learning algorithms in the area of energy analytics.



Autoregressive forests for multivariate time series modeling



Kerem Sinan Tuncel*, Mustafa Gokce Baydogan

Department of Industrial Engineering, Boğaziçi University, Bebek/İstanbul 34342, Turkey

ARTICLE INFO

Article history:

Received 13 March 2017

Revised 4 August 2017

Accepted 15 August 2017

Available online 24 August 2017

MSC:

00-01

99-00

Keywords:

Multivariate time series

Vector autoregression

Time series representation

Ensemble learning

Classification

ABSTRACT

Multivariate Time Series (MTS) modeling has received significant attention in the last decade because of the complex nature of the data. Efficient representations are required to deal with the high dimensionality due to the increase in the number of variables and duration of the time series in different applications. For example, model-based approaches such as Hidden Markov Models (HMM) or autoregressive (AR) models focus on finding a model to represent the series with the model parameters to handle this problem. Both HMM and AR models are known to be very successful in the representation of the time series however most of the HMM approaches assume independence and traditional AR models consider linear dependence between the variables of MTS. As most of the real systems exhibit nonlinear relations, traditional approaches fail to represent the time series. To handle these problems, we propose an autoregressive tree-based ensemble approach that can model the nonlinear behavior embedded in the time series with the help of tree-based learning. Multivariate autoregressive forest, namely mv-ARF, is a nonparametric vector autoregression approach which provides an easy and efficient representation that scales well with large datasets. An error-based representation based on the learned models is the basis of the proposed approach. This is very similar to time series kernels used for multivariate time series classification problems. We test mv-ARF on MTS classification problems and show that mv-ARF provides fast and competitive results on benchmark datasets from several domains. Furthermore, mv-ARF provides a research direction for vector autoregressive models that breaks from the linear dependency models to potentially foster other promising nonlinear approaches.

© 2017 Elsevier Ltd. All rights reserved.

1. Introduction

Research on the efficient representation of the high-dimensional time series is ever-increasing due to the ubiquitous nature of the multivariate and temporal data. Many researchers in different fields such as statistics, machine learning and signal processing focus on modeling multivariate time series (MTS) data. For example, in motion recognition, electromyography (EMG) signals from multiple muscles are recorded for four separate upper-arm movements and the aim is to classify upper-arm movements [17]. Also, a public sector use is shown, where a novel MTS clustering technique is discussed for clustering of crime locations, such as states and districts, with similar crime trends [9]. Another example from hydrology is provided by Raman and Sunilkumar [22], in which MTS modeling of water resources is considered in water management and planning.

Most of the existing studies focus on obtaining a rectangular representation for MTS data. In other words, each MTS is mapped

to a feature vector to be used as an input to machine learning approaches. If one can learn a model characterizing the dynamics of the system generating the MTS data, model parameters can be used as a feature vector. For example, transition and emission probabilities of a Hidden Markov Model (HMM) trained on MTS can be used to represent it. On the other hand, some of the time series data mining approaches make use of the similarity information between the MTS. Traditional similarity-based approaches consider the distance between each univariate series of MTS and aggregates this information to obtain the similarity between MTS. Model-based approaches (i.e. HMM) either compare the model parameters for MTS (i.e. feature vector) or the likelihood of observing the MTS given a model. Suppose we are interested in the classification of an MTS as in the arm-movement example [17]. Each training time series is represented by an HMM and the parameters of an HMM trained on the test series can be compared to the ones for training series. The other option is to compute the likelihood of observing the test series using the HMMs of the training series. Similar to nearest-neighbor classification, based on the likelihood over all HMMs (i.e. one HMM for each training series), one can determine the training series that is similar to test series (i.e. the one with the largest likelihood).

* Corresponding author.

E-mail addresses: keremsinantuncel@gmail.com, kerem.tuncel@boun.edu.tr (K.S. Tuncel).

Benefiting from the ideas in model-based approaches, this paper provides an autoregressive (AR) modeling strategy for MTS. As a common approach, we consider classification as an application to evaluate our modeling strategy. We assume that the time series can be modeled as an autoregressive process and time series from different classes has a distinct model. Using these models, each time series can be represented by their goodness of fit to the models of each class. There are numerous algorithms proposed to solve the classification problem, the methods mostly fall into two main categories; feature-based methods and distance-based methods. In this sense, model-based approaches can go into both categories depending on how the models are utilized.

Due to the complexity of MTS data, there is exhaustive research on finding representations and extracting meaningful patterns from MTS data sets. The most popular and simple approach is using rectangularization approaches. For example, a two dimensional Singular Value Decomposition (SVD) approach is proposed in [28]. Consequently, Symbolic Aggregate Approximation (SAX) [20] is another representation technique designed for univariate time series (UTS). Similarly, Baydogan and Runger [4] propose a symbolic representation for MTS which generates a codebook from the terminal nodes of decision trees. Moreover, for longer time series, alternative transformations such as wavelets and Fourier transforms are discussed [8]. HMMs to represent MTS is also considered by several studies in the literature. For example, an HMM-based methodology with a principal component analysis (PCA) representation for motion recognition is provided by Bashir et al. [2].

Distance-based methods mainly consists of modifications of univariate time series (UTS) similarity approaches to work in MTS framework. Among the distance measures in the literature, Dynamic Time Warping (DTW) [6] has become the benchmark due to its high accuracy in many of the commonly used datasets. Subsequently, many researchers focus on improvements on various aspects of DTW methodology such as efficiency and accuracy. Also, there is research being conducted to find more robust distance measures than DTW. For example, Yang and Shahabi [29] propose a PCA-based distance measure for MTS called *Eros* (Extended Frobenius Norm). High dimensionality is another problem to be addressed in MTS datasets. For that purpose, Cuturi and Doucet [12] propose an autoregressive kernel-based distance measure for MTS data. Another AR modeling strategy aims at learning MTS representation based on local autopatterns [5], which also introduces a similarity measure called learned pattern similarity (LPS). Distance-based approaches mostly work by taking each attribute of an MTS as an independent UTS. However, this methodology may sometimes cause some important data loss since MTS are not only defined by separate attributes but also by relationships between them.

The difference of mv-ARF from the works in literature is that it provides a framework for modeling the dynamic behavior of MTS. The strategy incorporates the strengths of AR modeling with the capabilities of tree-based ensembles. Use of tree-based learners enables a flexible and simple generative approach with only a few parameters. Moreover, combining AR components with the multi-response learning scheme provides comprehensive information to model the interactions between features. In that sense, the representation learned by mv-ARF is comparable to the AR-based strategies such as LPS and AR Kernel in addition to HMM-based strategies due to its generative and AR approach.

The primary contribution of this work is the introduction of a new autoregressive modeling strategy. Proposed approach aims to capture the dynamics of MTS by using a representation based on a vector autoregressive model. Proposed methodology is denoted as “MultiVariate AutoRegressive Forests”, mv-ARF, which trains tree-based ensemble learners with autoregressive components in a multitask setting. Fig. 1 illustrates a summary of the mv-ARF

methodology. There are some key features of mv-ARF model; the multitask learning approach utilized in the model provides a way to consider all the variables at the same time. The representation is acquired from a supervised learner which uses AR components as input. Therefore, the only form of feature extraction is to generate the AR components with a predefined lag value leading to a very simple feature extraction scheme with only lag parameter. A simple representation is used, which consists of time index and the observation values of each attribute as columns. Most AR models either aim to capture non-linear relations or non-stationary behavior, mv-ARF is a robust AR model which is able to capture both. Tree-based ensembles utilized by the model are able to capture non-linear relations between AR components and incorporation of time order enables mv-ARF to capture non-stationary behavior. Furthermore, the lag (or order) of autoregression serves as an upper-bound, that is, the model is able to detect and use the lag value that gives the best accuracy, within the range of lag values defined by the upper-bound. Lastly, the computational complexity of mv-ARF is almost linear to both problem and algorithm parameters and each step of mv-ARF is embarrassingly parallel.

The capabilities and the effectiveness of the mv-ARF model is tested within MTS classification framework. For a classification task with C class labels, the model ends up with C explanatory models. A novel classifier is introduced which utilizes a representation using the prediction errors acquired from the constituent models as illustrated in Fig. 1. The representation and the tree-based approach allows mv-ARF to efficiently deal with datasets that contain time series with differing lengths as well as different data types (i.e. categorical, ordinal). The final error-based representation can be acquired by calculating the predictions and the prediction errors for each time series from each model. Effectiveness of the proposed model is demonstrated by experiments on a large amount of commonly used datasets in MTS classification literature.

Another important motivation for this work, is to underline the importance of utilizing multivariate learning methods instead of modifying univariate models, in MTS analysis. Multivariate models are able to capture the interactions between attributes, which may be of importance. In order to show the superiority of multivariate approaches, the univariate approach for mv-ARF framework is also considered, that is, univariate forests are built and the final representation is acquired by combining separate predictions, instead of using multivariate learning methods. The results of mv-ARF is compared with the univariate approach, along with several other benchmark approaches used in MTS classification.

The rest of the paper is organized as follows; Section 2 introduces some relevant works in the literature and Section 3 provides background. Section 4 explains the methodology of the method in detail. Section 5 provides the experiments from a list of datasets [10,19,21,24]. Section 6 gives the concluding remarks.

2. Literature review

There are two modeling strategies that are common in the literature. Feature-based strategies aims to find a rectangular representation for MTS by extracting descriptive features from the time series. On the other hand, distance-based strategies aim to provide similarity measures that works well with MTS. Moreover, model-based strategies can be used for both extracting features and to define a similarity depending on how the model is utilized. As mv-ARF is a model-based framework, the main focus is on model-based approaches.

As one of the many feature-based approaches, a recent successful approach by Grabocka et al. [16] extends the shapelet discovery techniques for MTS classification. Shapelet discovery is a methodology to discover subsequences (which are called shapelets) to represent a time series. Generally, shapelets with high

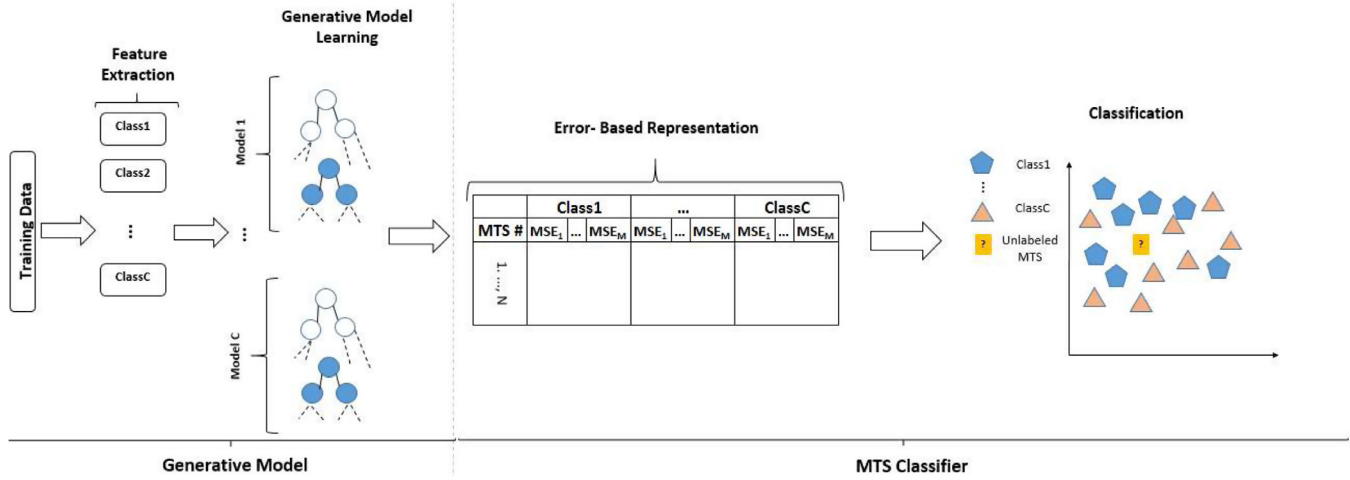


Fig. 1. Summary of mv-ARF algorithm. For each class, a tree-based ensemble model is trained on lagged observations. The target is the observation vector at time t where all observations from $t - p$ to $t - 1$ are used as predictors. This modeling strategy is similar to vector autoregression as we consider multiple targets simultaneously with the proposed based learner. After models are learned for each class, each multivariate time series is represented by their goodness of fit to the models. Mean square error (MSE) for each attribute of MTS is obtained using the models and the feature vector of MTS contains the error rates from each model.

predictive accuracy are chosen which results in high performance costs. A novel principle to learn optimal shapelets through classification loss functions is proposed and this approach is shown to be a computationally efficient way to classify MTS. Later, Karlsson et al. [18] introduce alternative ways to identify shapelets. These approaches exploit the use of distances of subsequences to the time series as features. Criticizing the fact that determination of which features to generate is not a trivial task, Baydogan and Runger [4] propose a symbolic representation for MTS. The idea is to identify the observations of MTS that defines the class with the help of classification trees. Proposed approach implicitly make use of subsequence information which makes it similar to the shapelet-based approaches.

A novel similarity measure, Correlation-based Dynamic Time Warping (CBDTW), utilizing PCA and dynamic time warping is proposed in [1]. Two common PCA models are used as homogeneity measures for segmentation, which is dependent on covariance between variables. Subsequently, the similarity (or dissimilarity) between segments are computed from the PCA similarity factor. Furthermore, DTW with PCA local similarity is utilized to allow for the time shifts between segments.

A model-based application is discussed in [2]. The application uses HMM models for classification of motion trajectories. The trajectory data is segmented into atomic units of action and a PCA representation is acquired for all the subtrajectories and for all dimensions. HMM models are trained on the subtrajectories. The state size of the HMM is selected with a spectral clustering algorithm. The state distributions are Gaussian Mixture Models (GMMs). After the HMM models are built for each class, the classification is made by calculating log-likelihoods of unlabeled trajectories from each HMM model.

Recently, there are few successful AR model-based approaches shown to provide competitive results on MTS classification problems. A pioneering work in this sense is the AR Kernel methodology [12]. AR Kernel aims to provide an alternative distance measure for DTW. The proposed kernel builds on the “covariance kernel” introduced in [23]. Cuturi and Doucet [12] use VAR model define the parameter space. An alternative kernel which uses Gram matrices is also considered. This allows the kernel to efficiently handle structured data sets. Reported results demonstrate the efficiency of the kernel-based approach for data sets with large number of attributes. Furthermore, the reported accuracy results are competitive with DTW distance. Similar ideas are benefited by

Baydogan and Runger [5] which learns representation based on local autopatterns acquired with AR components. This representation is presented with a similarity measure called learned pattern similarity (LPS). LPS methodology provides a similarity measure based on a tree-based ensemble. The ensembles are trained to learn an AR model between the subsequences of the time series. With the use of bag-of-words approach, a new representation is obtained using the structure of learned model.

For the MTS classification problem, mv-ARF provides a model-based approach to express the MTS in a simpler and manageable manner. Similar to feature-based strategies, a rectangularization using the AR components is obtained in mv-ARF. But unlike feature-based strategies, the classification is not applied on the rectangular representation. As a generative model, mv-ARF is utilized to obtain a measure to represent the goodness of fit of an MTS to the model. In that sense, mv-ARF is similar to HMM models using likelihood information as similarity for MTS classification. Furthermore, some model-based learners (such as HMMs) not only require extensive parameter optimization processes but they are also sensitive to the setting of the parameters, whereas mv-ARF has very few parameters to consider. On the other hand, autoregressive nature of mv-ARF model can be problematic in cases where the autoregressive assumption simply does not apply.

3. Background

Primary output of this work is a novel modeling framework for MTS data, which is called mv-ARF algorithm. Proposed algorithm provides an efficient, model-based approach to model MTS by making use of tree-based learners and AR models. Moreover, as a model-based approach with many applications in MTS modeling, HMMs are also used in the experiments for comparison. Hence, a brief background on tree-based learning, autoregressive models and Hidden Markov Models is necessary in order to have a comprehensive understanding of mv-ARF model and the experiments.

3.1. Autoregressive models

Autoregressive modeling is a modeling strategy that is commonly used in time series analysis. The inception of AR modeling is the basic $AR(p)$ model shown in Eq. (1). $AR(p)$ model, where p denotes the lag value, is a simple regression model in which, the

predictor variables are past observations.

$$X(t) = \beta_0 + \sum_{i=1}^{i=p} \beta_i X(t-i) + \xi(t) \quad (1)$$

Since the capabilities of AR(p) model is limited, many other models are developed to enhance the use of AR models. For example, Autoregressive Conditional Heteroskedasticity (ARCH) model [14] and Threshold AR (TAR) models discussed by Tong [25] are proposed to include non-stationary and non-linear behavior.

Autoregressive models assume that, the current state of a temporal process has a linear or non-linear relation with the past states. Even though this assumption may not hold for every dataset, the frequent use of AR models suggests that it is a robust assumption. AR assumption is central for mv-ARF methodology, which utilizes AR features in the prediction stage. Thus, mv-ARF method can also be classified as an AR model which incorporates several other features.

3.2. Tree-based learning

Decision trees are established learners with convincing accuracy, that have a broad range of applications. Some of the univariate decision trees have multivariate counterparts which are able to handle multiple response variables. CART trees proposed by Friedman et al. [15] have a multivariate extension named Multivariate Regression Trees (MRT) [13], which is utilized by mv-ARF.

Univariate Regression Trees (URT) aim to explain the variation of a singular response variable with explanatory variables which may be numeric and/or categorical [15]. The tree structure is built with top down induction, which include splitting of the dataset into data with similar response values. The measure of impurity for regression trees is the total sum of squares (of the response variable) over the node mean. MRT's are natural extensions from URT's and are obtained by changing the univariate response variable with a multivariate response and by replacing the impurity measure to sum of squared distances which is the sum of distances from the node centroid.

Tree ensembles are used in order to enhance tree learners. A random forest (RF) [7] model is a parallel ensemble of N decision trees, $\{h_n, n = 1, 2, \dots, N\}$. RF is based on two methods; bagging and sampling. Each h_n is built from a bootstrap sample where each split is made with a random sample of features. At each bootstrap sample, a subset of observations are left out. For each tree, the observations that are left out of the sample are called out-of-bag (OOB) observations. OOB error can be calculated for each observation by averaging the prediction error for trees when the observation is out of bag. The OOB score has proven to be a robust estimate of the models performance. Moreover, random sampling reduces the variance of the model as well as the computational complexity of the algorithm. Therefore, RF algorithm scales well with large number of features. For a RF with N trees, the prediction of response variable Y can be calculated as;

$$\hat{Y} = \frac{\sum_{h_n} \hat{y}_n}{N}$$

where $\hat{y}_n$ is the prediction from tree h_n . Also, RF algorithm is good with ordinal and nominal data and are able to capture non-linear interactions between features. Tree-based ensembles built with MRT's are utilized in mv-ARF.

3.3. Hidden Markov Models

Hidden Markov Models are well known generative models for modeling time series that are most commonly used in speech recognition. Given a discrete or continuous sequence X , HMM models the sequence as a set of hidden states with Markovian property.

Each HMM consists of a set of latent states, a set of emission probabilities for each state and a set of transition probabilities between states. With this modeling scheme, HMM calculates the probability of X being generated by the model which is the sum of the probabilities underlying each possible state-sequence path that can generate X . Given $X = X_1, X_2, \dots, X_T$ and where $a_{i,j}$ and $b(X_t)$ are the transition and emission probabilities respectively, the mathematical formulation for the probability of the sequence at time $t \in T$, X_t , being generated from the HMM model λ is as follows;

$$P(X_t | \lambda) = \sum_s \prod_t a_{s_{t-1}, s} b_{s_t}(X_t)$$

Moreover, HMM's are highly parametric models that require the optimization of many parameters. Also, a formal method for the selection of the number of hidden states is not provided. The number of states are mostly selected with intuition or with trial and error.

HMM's can be used for classification of time series. HMM's are similar to mv-ARF since they are both generative approaches. The HMM classification approach provided in this paper acquires the likelihood scores of separate HMM models that are built for each class label. It is essentially the same with mv-ARF and the only difference is that prediction error-based representation is used in mv-ARF instead of likelihood scores.

4. Methodology

A single MTS, which is a time series with M attributes, length T and coming from a set of size N is denoted by, X^n . The m th attribute of series n is denoted by x_m^n and the observation at time t is denoted by $x_m^n(t)$. A MTS, X^n is represented as a $T \times M$ matrix;

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

A column m of time series n is represented as;

$$x_m^n = [x_m^n(1), x_m^n(2), \dots, x_m^n(t), \dots, x_m^n(T)]'$$

There are N MTS in the training set, each of which is associated with a class label; $c^n \in \{1, 2, \dots, C\}$ for $n = 1, 2, \dots, N$. Given a set of unlabeled test set, the problem is to associate each MTS in the test set with one of the pre-defined class labels. Although proposed approach can handle the time series of different length, the definitions assume that each time series is of length T for the sake of simplicity.

4.1. Feature extraction and model generation

The initial representation used in mv-ARF introduces time order to the model. A single MTS, X^n , is represented as a $T \times M$ matrix in which, each row represents an instance. A training set of N MTS will result in $\Psi_{NT \times M}$;

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

Autoregressive columns with the desired lag (denoted by p) and the time index are inserted as columns to this representation which yields the feature space; $\Theta_{NT \times (Mp+1)}$. A single row from Θ representing an observation looks like;

$$[t, x_1^n(t), x_1^n(t-1), \dots, x_1^n(t-p), \dots, x_M^n(t), x_M^n(t-1), \dots, x_M^n(t-p)]$$

Autoregressive columns carries information about past values. Thus, each observation is assumed to have a relation to the past observations of all attributes of the MTS. Also, time order of MTS is incorporated which provides a way to capture the non-stationary dynamics. First p rows for any lag value p is not available and thus assumed to be missing.

As a common approach, an MTS classification scheme provided by mv-ARF will be utilized to test the capabilities of the proposed approach. For MTS classification problem, mv-ARF aims to describe the behavior of each class and to capture inter-class differences. Classification approach is based on the assumption that, each MTS with identical class labels display similar dynamic behavior and similar interactions are observed between attributes. Therefore, a model for each class that is able to capture the generalized information of the class, is provided. Furthermore, a multivariate tree-based ensemble with AR components searches for the intrinsic dynamic behavior that is generalizable for all the time series with the same class label. For this purpose, multitask tree-based ensembles are trained on MTS associated with each class. Thus, a total of C models are learned from MTS associated with C class labels.

Each learner is built with a different lag order, which makes the model capable of capturing differences between the classes with respect to autoregressive behavior. The trained models capture the generalized information for each class. The trees of the ensembles are MRT's with lag values and time index as the predictor variables and the attribute values as the response vector. The functional relationship achieved is both multivariate and autoregressive;

$$(x_1^n(t), x_2^n(t), \dots, x_M^n(t)) = f(t, x_1^n(t-1), \dots, x_1^n(t-p), \dots, x_M^n(t-1), \dots, x_M^n(t-p)) \quad (2)$$

Modeling the autoregressive behavior of MTS in a multivariate manner makes the model capable of capturing the interaction between the current state vector and the past observations of each feature. This provides a comprehensive foundation for modeling the dynamic behavior underlying MTS. Trees are built in a breadth-first manner and with some parameter restrictions which will be discussed in further parts. Also, high dimensionality due to $(Mp+1)$ columns when M or p is too large, is dealt by the tree-based ensemble since it scales well with large number of features.

4.2. Classification

Multivariate, tree-based ensembles with AR components are the primary output of mv-ARF. After the models are learned, a prediction error-based representation is acquired by using observation value predictions and OOB predictions. For any given model, predictions are made for each observation in each MTS that have a different class label than the given model and OOB predictions are used for MTS that have the same class label with the given model. This approach is utilized in order to avoid any possible overfitting issues. For a MTS with M attributes and length of T , a $T \times M$ matrix is obtained from each model, either by prediction or using the OOB values. The simple mean squared error (MSE) calculations, from Eq. (3), are made for predictions from each model.

$$\frac{1}{T} \sum_{t=1}^{t=T} (x^{\text{observed}}(t) - x^{\text{predicted}}(t))^2 \quad (3)$$

In the error-based representation, the final output of the mv-ARF framework, each MTS is represented as a row and MSE values of each attribute, acquired from each model are the features of the MTS. In a dataset containing N MTS from C classes with M attributes, we map the dataset to the feature space $\Phi_{N \times MC}$. A sample row from Φ looks like;

$$[MSE_1^1, MSE_2^1, \dots, MSE_M^1, \dots, MSE_1^C, MSE_2^C, \dots, MSE_M^C]$$

Table 1
Sample error-based representation of MTS.

Class	Error ^{Attribute1} _{Class1}	Error ^{Attribute2} _{Class1}	Error ^{Attribute1} _{Class2}	Error ^{Attribute2} _{Class2}
1	0.048	0.056	0.081	0.083
1	0.156	0.067	0.113	0.168
2	0.314	0.098	0.188	0.047
2	0.107	0.201	0.119	0.144
⋮	⋮	⋮	⋮	⋮

A sample error-based representation of a MTS classification problem with two attributes and two class labels.

Each row representing a MTS is concatenated to acquire the final matrix representation. A sample illustration of the error-based representation can be found in Table 1. Any supervised learner can be applied to this representation. Moreover, Algorithm 1 provides

Algorithm 1 mv-ARF algorithm: $mv-ARF(X, p, D, J)$, where X is a MTS dataset with N series from C classes and M features. p is the lag value, J is the number of MRTs with depth D in the ensemble.

Learning training representation

- 1: Obtain predictor matrix Θ of size $NT \times (Mp+1)$ and response matrix B of size $NT \times M$ from X
 - 2: Define the set of ensemble models Δ
 - 3: **for** $c \in C$ **do**
 - 4: Obtain the predictor matrix $\theta_c \in \Theta$ and response matrix $b_c \in B$
 - 5: Train the ensemble model δ_c with J trees of depth D on θ_c to predict b_c
 - 6: Set $\delta_c \in \Delta$
 - 7: **for** each time series i **do**
 - 8: Obtain OOB prediction $\hat{b}^i$ from δ_c
 - 9: Calculate ϕ_c^i as defined in Eq 3;
 - 10: **end for**
 - 11: **end for**
 - 12: Populate OOB error matrix Φ of size $N \times (MC)$ with ϕ_c^i vector
- Learning test representation
- 13: Given a single test MTS, X^i , obtain the predictor matrix
 - 14: Obtain the test prediction $\hat{b}^i$ and ϕ_c^i as defined in Eq. (3) for all $\delta_c \in \Delta$
 - 15: Populate test error matrix Φ of size $1 \times (MC)$ with ϕ_c^i vector

the pseudo-code for the algorithm which illustrates the algorithm steps to acquire the error-based representation for both training and testing.

4.3. Algorithmic complexity

Training complexity for mv-ARF is the complexity associated with learning tree-based ensembles. Complexity of a single ensemble is $O(JNMD)$, where M is the number of variables considered at each split, N is the number of observations, J is the number of trees and D is the depth of each tree. Since these operations are repeated for each class label and if we assume there are C classes in the training set each built with lag order p , the overall training complexity becomes; $O(CJD(Mp)n)$. This is the algorithmic complexity for the operations mv-ARF algorithm performs to produce the error-based representation for the training data as an output.

Moreover, testing complexity is determined by acquiring the representation and classification of a MTS. The representation is acquired by operating through C ensembles which is $O(CJDn)$ where n denotes the number of testing observations. Note that, both training and testing complexity of mv-ARF algorithm are linear with respect to its parameters. Note that, the complexity associated with mv-ARF is linear with respect to all parameters as-

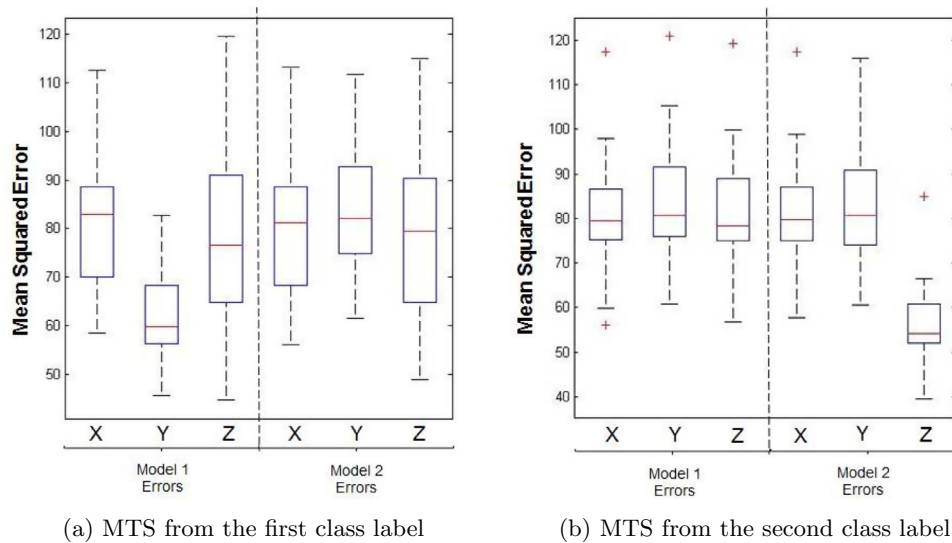


Fig. 2. Boxplots of prediction errors calculated for each feature of the synthetic dataset. Each feature is calculated twice since there are two models for $C = 2$. Fig. 2a shows the boxplots of the errors for each attribute from the first model (trained with MTS of the first class) and Fig. 2b provides the same information from the second model (trained with MTS of the second class).

sociated with the model building process, thus mv-ARF scales remarkably well with both ensembles with large number of trees and large datasets with long time series and large number of attributes.

4.4. Multivariate vs. univariate modeling strategies

To demonstrate the effectiveness of multivariate models over univariate ones (extensions of univariate models to MTS), univariate version of mv-ARF is also considered in experiments. In the univariate version, separate models are built for each attribute and each class. For the classification problem, MTS are represented by univariate models that are learned from each attribute of MTS resulting from each class label. The tree-based ensembles are constructed with regression trees [15] instead of MRT's. This way, the error-based representation is acquired by simply combining MSE's coming from separate univariate models for each feature instead of a multi-response model that predicts all the features simultaneously.

A synthetic dataset is generated with the following parameters; $M = 3$, $T = 100$, $C = 2$ and $N = 100$. Each feature is generated with standard normal distribution. For both class labels, one of the features mimics the first feature with a pre-defined delay. In other words, in each class, a feature has a lagged relationship with the first feature. This feature (the lagged feature) is the second feature for MTS of Class 1 and the third feature for MTS of Class 2. Moreover, the order of delay is different for classes. Thus, a dataset with a simple multivariate AR relationship is obtained.

mv-ARF model is trained on the synthetic data for analysis. Fig. 2 illustrates the prediction errors for constituent features of synthetic MTS. The three features are named as X, Y and Z. Two figures are for the MTS of different class labels. Each boxplot represents prediction errors of a feature. Since $M = 3$ and $C = 2$, two ensemble learners are built in mv-ARF (referred to as Model 1 and Model 2). Therefore, a total of six boxplots representing errors calculated for three features from the two forests are provided. The boxplots represent the columns of the error-based representation. The figure shows that the errors of feature Y calculated from the model trained with MTS of the first class (Model 1) and the errors of feature Z calculated from Model 2 are considerably smaller than the rest. Note that, Y and Z are the lagged features for the two classes. This shows the lagged feature in each class is successfully

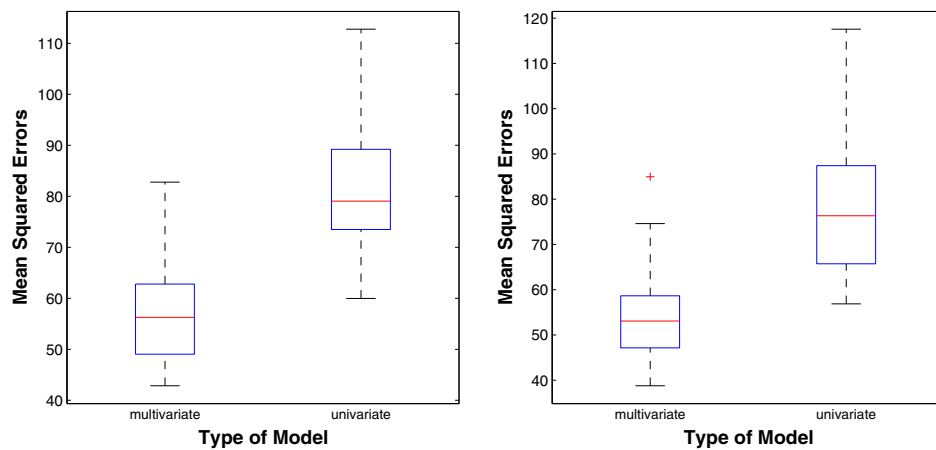
identified by mv-ARF. A similar experiment is designed to compare multivariate and univariate modeling strategies. Both the multivariate and the univariate versions of mv-ARF are trained on the synthetic dataset and MSE values for each time series are acquired from both versions. Fig. 3 shows the mean squared errors calculated for the relevant feature of each class by both the multivariate and univariate versions of mv-ARF. Only the feature with a lagged relationship (Y for Class 1 and Z for Class 2) is considered for each class since all other features are random noise (which is demonstrated in Fig. 2). The figure shows that the MSE's acquired from the multivariate version are significantly smaller than the univariate version for each class label. This indicates that the univariate modeling scheme is not able to model the autoregressive relationship between features.

5. Experiments and results

The effectiveness of mv-ARF approach is tested and accuracy results are reported for nineteen MTS datasets from varying fields such as; motion recognition, medicine and speech recognition. The datasets, which are commonly used in MTS classification research, are taken from Lichman and co-workers [10,19,21,24]. Properties of the datasets are illustrated in Table 2. Problem characteristics differ significantly in terms of time series lengths, number of attributes and number of classes.

Parameter settings of mv-ARF algorithm are straightforward. Apart from the tree parameters, the only parameter to be considered is the autoregressive lag order p . The lag order selection is through OOB accuracy comparison among a set of candidate p values. For each class, alternative lag values are considered and the lag value with the highest OOB score is selected as the p value for that class. This way, each class label is associated with a different lag order and class differences with respect to autoregressive behavior is also captured. Empirically, the algorithm works very fast. But if small time increments are of importance, search for the maximum OOB value can be reconsidered as a search for the first peak. Our empirical analysis shows that accuracy behaves like a single-peak array with increasing lag value, thus the selected lag order is always that peak.

As for the forest parameters, the number of MRT's in each model J and maximum depth of each tree D is considered. Our empirical analysis shows that, similar to any forest application, the



(a) MTS from Class 1 (the second feature)

(b) MTS from Class 2 (the third feature)

Fig. 3. Box plot of MSE's calculated by multivariate and the univariate versions for the lagged feature for each time series in the synthetic data.**Table 2**

Properties of MTS datasets; lengths, dataset sizes, number of attributes and number of classes for each dataset.

	# of Classes	# of Attributes	Length	Training Dataset Size	Test Dataset Size	Source
Arabic Digits	10	13	4–93	6600	2200	[19]
AUSLAN	95	22	45–136	1140	1425	[19]
Character Trajectories	20	3	109–205	300	2558	[19]
CMU MOCAP	2	62	127–580	29	29	[10]
ECG	2	2	39–152	100	100	[21]
Japanese Vowels	9	12	7–29	270	370	[19]
Kick vs. Punch	2	62	274–841	16	10	[10]
LIBRAS	15	2	45	360	585	[19]
PenDigits	10	2	8	300	10,692	[19]
Robot Failure						[19]
LP1	4	6	15	38	50	
LP2	5	6	15	17	30	
LP3	4	6	15	17	30	
LP4	3	6	15	42	75	
LP5	5	6	15	64	100	
Uwave	8	3	315	896	3582	[19]
Wafer	2	6	104–198	298	896	[21]
Walk vs. Run	2	62	128–1918	28	16	[10]
Digits Shape	4	2	30–98	24	16	[24]
Shapes	3	2	52–98	18	12	[24]

Source column indicates the reference of the corresponding dataset. The accuracy results are also provided in further parts.

model is robust to the number of trees after a sufficient number of trees. The change in OOB score with increasing J for each class within the *Uwave* data set is shown in Fig. 4. The figure shows that the increase in OOB score becomes negligible after sufficient J for each class. mv-ARF is also robust to D if considered individually. Since mv-ARF classification includes partitioning of the dataset with respect to its class labels, there is a risk of overfitting. D is considered as a parameter to control overfitting. The trees are built with no pruning and with random sampling.

In the experiments, the number of MRTs J is set to 100 and D is set to 10. The AR order p is selected from $\beta = [1, 2, 3, 4, 6, 8, 10, 12, 15, 17, 20]$. Also, a standardization is applied to each feature with zero mean and unit deviation, in order to avoid complications due to scale differences. Furthermore, a simple Random Forest [7] classifier with 200 trees is used for class assignment. But the final labeling can also be achieved by any other supervised classifier. An alternative scenario where the labeling is done by a k -NN classifier is also considered. The number of neighbors (k) is selected through cross-validation in this second scenario. For comparison, classification results are also obtained for

the univariate setting where the parameter setting and the labeling strategies are identical with the multivariate version.

Results from other approaches are also evaluated for comparison and discussion. MTS classifiers with AR components are selected for evaluation. MTS classification results from LPS methodology [5] are considered for comparison due to its AR approach. Similarly, Cuturi and Doucet [12] provide classification results for a subset of the datasets used in mv-ARF experimentation, acquired from an AR kernel. Furthermore, since it has become a benchmark for time series classification, DTW with 1-NN classifier results are also provided for comparison. The standardization that is used for mv-ARF is also used for DTW and the DTW distance between two MTS is taken to be sum of the distances of between corresponding features. Only DTW with full warping window is considered. Moreover, since mv-ARF is a generative approach, a methodology similar to mv-ARF with HMM models are also designed for comparison. Similar to mv-ARF, HMM learners are built for each class label and a representation is acquired where the log-likelihood scores of each class label acts as features for each MTS. The models built for each class are HMM models with Gaussian emissions and the num-

Table 3
Classification accuracy results for MTS datasets.

	$mv - ARF$	$mv - ARF_{knn}$	$mv - ARF^{uni}$	$mv - ARF_{knn}^{uni}$	LPS	DTW	ARkernel	HMM	HMM_{knn}
Arabic Digits	0.048	0.084	0.069	0.191	0.029	0.092	0.012	0.023	0.037
AUSLAN	0.066	0.172	0.067	0.240	0.246	0.238	0.082	0.439	0.610
Character Trajectories	0.072	0.187	0.095	0.105	0.035	0.033	0.100	0.086	0.089
CMU MOCAP	0.000	0.034	0.004	0.000	0.000	0.0069	0.000	0.009	0.020
ECG	0.215	0.236	0.257	0.223	0.180	0.150	0.180	0.156	0.142
Japanese Vowels	0.041	0.050	0.051	0.074	0.049	0.351	0.016	0.049	0.100
Kick vs. Punch	0.024	0.124	0.010	0.300	0.100	0.100	0.000	0.352	0.300
LIBRAS	0.055	0.251	0.065	0.076	0.097	0.200	0.073	0.083	0.073
PenDigits	0.077	0.121	0.091	0.099	0.092	0.075	0.048	0.089	0.161
Robot Failure									
LP1	0.176	0.308	0.144	0.112	0.138	0.280	0.140	0.174	0.176
LP2	0.374	0.392	0.281	0.316	0.296	0.467	0.366	0.406	0.398
LP3	0.230	0.360	0.202	0.261	0.280	0.500	0.433	0.404	0.260
LP4	0.094	0.150	0.093	0.142	0.090	0.106	0.040	0.035	0.162
LP5	0.317	0.378	0.304	0.367	0.308	0.480	0.530	0.324	0.422
Uwave	0.048	0.091	0.073	0.101	0.020	0.071	0.096	0.050	0.054
Wafer	0.069	0.097	0.070	0.099	0.038	0.040	0.032	0.064	0.075
Walk vs. Run	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Digits Shape	0.000	0.000	0.000	0.000	0.000	0.064	0.000	0.062	0.062
Shapes	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

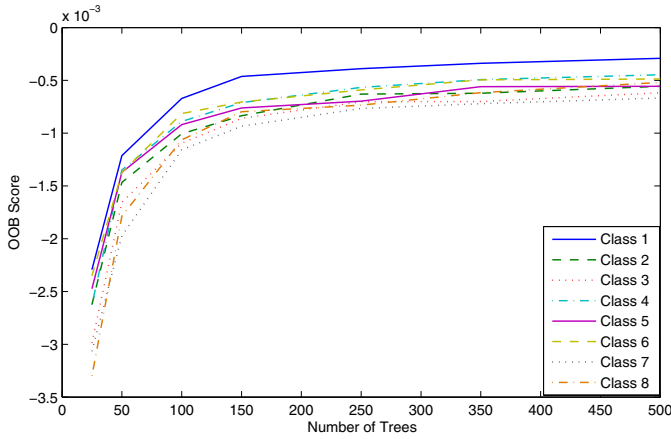


Fig. 4. OOB scores of each class with increasing J values.

ber of hidden states for each class is chosen to be the value with the highest probability score for that class within a set of alternate values which is similar to the selection of p in $mv-ARF$. Classification results for both RF and k-NN trained on HMM log-likelihood scores are reported.

5.1. Classification accuracy

Applications of both univariate and multivariate versions of $mv-ARF$ in addition to the HMM classifiers are implemented in Python [27] software. Source code for $mv-ARF$ and $mv-ARF^{uni}$ can be found in [26]. Furthermore, experiments with Autoregressive Kernels are conducted using author's source code [11], with lag order set to 5 and alpha parameter set to 0.5. LPS application is provided in R software which can be found in [3]. Moreover, the results of DTW are acquired from Baydogan and Runger [4].

Test data accuracy results for seven methods are illustrated in Table 3. The column $mv-ARF$ and $mv-ARF_{knn}$ denotes the results for the RF classification scenario and k-NN classification scenario of $mv-ARF$ respectively. Similarly, the univariate version with the RF and k-NN classifier scenarios are denoted by $mv-ARF^{uni}$ and $mv-ARF_{knn}^{uni}$ respectively. Results reported for all $mv-ARF$ related methods are the mean accuracy results for 25 replications with the described parameter settings. The results for the AR kernel, DTW and LPS are reported under column *ARkernel*, *DTW* and *LPS* respec-

tively. Results of LPS are the mean accuracy results of ten replications. Moreover, column titled *HMM* provides classification accuracy for HMM models combined with RF classifiers and HMM_{knn} denotes the scenario with the k-NN learner. The set of alternate values for number of states is; $v = [2, \dots, 10]$. The results for both HMM-based methods are the mean accuracy results over 25 replications.

$mv-ARF$ performs better or as good for entire datasets when compared to other methods. Either $mv-ARF$ or Cuturi and Doucet [12] outperform others in most datasets. As expected, $mv-ARF$ works well with motion recognition data, such as *AUSLAN* or *LIBRAS*, since autoregressive assumption is an intuitively strong assumption for motion recognition data. AR Kernel method slightly outperforms others in sensor reading data such as *Wafer* or *Arabic Digits* where $mv-ARF$ outperforms [12] in motion recognition data. Even though AR kernel method scales well with a high number of attributes with respect to other distance-based methods, its empirical time complexity is still much larger when compared to LPS and $mv-ARF$. Moreover, HMM classification methods also provide competitive results with $mv-ARF$ but HMM methods suffer from high number of parameters to be set. The high number of optimized parameters causes high complexity but more importantly, HMM models require a large training set in order to optimize a large number of parameters. This causes poor accuracy in datasets such as *AUSLAN*; where there are many classes which leads to a low number of training MTS for each class, or *Kick vs. Punch*; where the training set is simply small. Furthermore, standardization of each attribute causes problems for DTW in *Japanese Vowels* dataset. Since each attribute value represents a coefficient from linear prediction analysis, the variance within an attribute is too small which makes it harder for DTW to differentiate attributes.

The effectiveness of multivariate methods over univariate methods can be shown by the comparison between $mv-ARF$ and $mv-ARF_{knn}^{uni}$. Multivariate method outperforms $mv-ARF_{knn}^{uni}$ in all datasets apart from a couple of datasets where they perform equally well. In many datasets there are slight differences between accuracies but there are examples such as *Arabic Digits* dataset, where there exists a considerable difference between classification accuracies. The differences in accuracy can be addressed to the neglected interactions between separate attributes in the univariate version. Therefore, the difference would increase in more complicated data that exercises more complex dependency structure between attributes. Furthermore, the empirical complexity of the multivari-

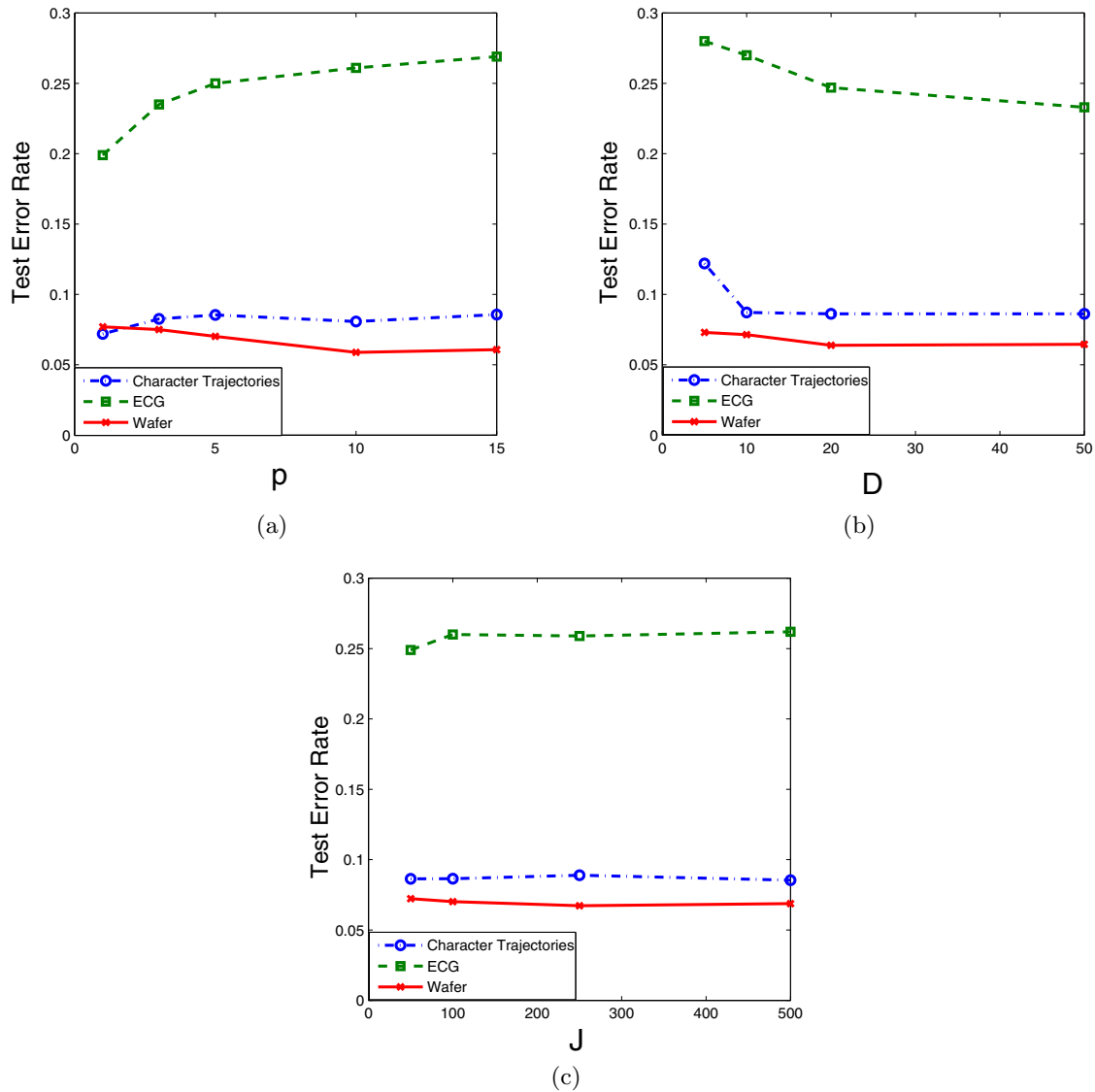


Fig. 5. Test-error rates for selected values of J , D and p acquired from three datasets. (10 replications).

ate method is much better than the univariate method due to the multi-response predictors. The difference with respect to time is more evident in large datasets such as *AUSLAN*, in which the mean training time for mv-ARF is ≈ 330 s whereas training mv-ARF_{uni} takes ≈ 1150 s. Thus, multivariate methodology provides a better approach with respect to both classification accuracy and empirical complexity.

5.2. Sensitivity analysis

Model parameters are considered for the sensitivity analysis of mv-ARF. Three datasets (*Character Trajectories*, *ECG*, and *Wafer*) are selected and classification accuracies are shown for selected parameters (J , D , p) in order to illustrate the robustness of the proposed approach to its parameters. Selected datasets provide sufficiently large training and test MTS for a consistent experimentation. For the consistency of these experiments, lag order is assumed to be the same for all class labels. Lag order, depth and number of trees considered in these experiments are; $p \in \{1, 3, 5, 7, 10\}$, $D \in \{5, 10, 20, 50\}$, $J \in \{50, 100, 250, 500\}$ respectively.

Average test data classification error rates are illustrated in Fig. 5. For the experimentation with p ; D and J are set to 10

and 100 respectively. The results of the experiments are shown in Fig. 5(a). Results show that mv-ARF is robust to the lag order unless it is too high or too low. For very low p values, the model suffers from insufficient information which may be lying within relations with further past values. On the other hand, each additional lag order results in a missing observation. Increasing p too much results in loss of valuable information especially for short MTS. Additionally, each lag order introduces extra variables to the tree-based learner. Too many redundant predictors increase both computational complexity and test-error rate. Note that, reaching definitive conclusions on the autoregressive behavior would be misleading and each set of MTS may have its own unique autoregressive property that is nonobservable. That is the primary reason behind the parameter selection scheme for p that is discussed in earlier sections.

Fig. 5(c) shows the average test-error rates for different values of J where $p = 3$, $D = 10$. Increasing J provides better accuracy as well as better stability in results (i.e. variance over runs are small). But after a certain level, improvements achieved by increasing J becomes negligible. Moreover, Fig. 5(b) shows the test-error sensitivity to depth parameter D . D can be dropped by growing full trees with vigorous splits. Figures show that the

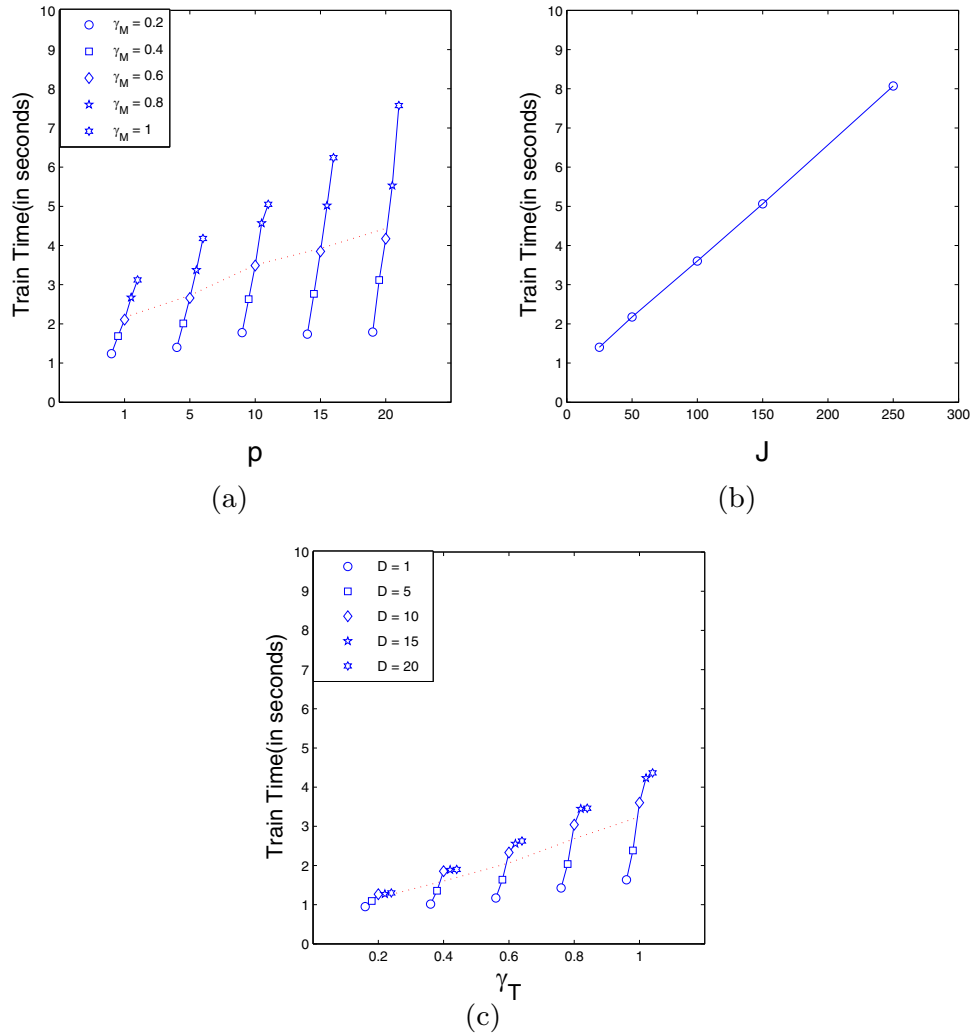


Fig. 6. Training times for selected values of J , D , p , γ_M and γ_T acquired from *Kick vs. Punch* dataset. (10 replications).

model is robust to D when it is selected sufficiently high. But overfitting in the prediction stage potentially causes decrease in the test accuracy, especially for large MTS with many class labels where the training data is divided into considerably smaller sets. So, even though mv-ARF is shown to be robust to this parameter, it is set as precaution against potential harm in test-accuracy caused by overfitting. On the other hand, too low D value would inflict the capabilities of the learners which can be observed in the figure.

5.3. Complexity analysis

Our experiments use an MS operating system with 12GB RAM and a processor with four physical cores (i7-4720HQ, 3.60 GHz). Even though the CPU is able to run on eight parallel threads, the experiments are conducted using a single thread.

For the empirical analysis, *Kick vs. Punch* dataset is used since it has sufficient number of attributes and length for each time series. The effect of the algorithm parameters (p , J , and D) as well as dataset parameters M and T on the empirical complexity is tested in the experiments. Both testing time and model training time are analyzed. Training time is simply the elapsed time to generate the error-based representation and testing time is the elapsed time to label a test MTS. A proportion of $\gamma \in [0.2, 0.4, 0.6, 0.8, 1]$ is randomly selected in each run for both M (γ_M) and T (γ_T). For the

sake of consistency, p is assumed to be constant for each class. Furthermore, for the algorithm variables, the considered values are; $p \in [1, 5, 10, 15, 20]$, $J \in [25, 50, 100, 150, 250]$ and $D \in [1, 5, 10, 15, 20]$. Empirical experiments for training and test complexity are presented in Figs. 6 and 7 respectively.

Training time results for γ_M and p (where $J = 100$, $D = 10$) is given in Fig. 6(a). A linear increase with M and p is consistent with the algorithmic complexity discussed in earlier sections. For a large dataset such as *Kick vs. Punch*, where lag order is always smaller than the total number of attributes, changing p has smaller impact on training time with respect to M . Empirical analysis for testing analysis with respect to p and M is provided in Fig. 7(a). The behavior is consistent with the algorithmic complexity discussed in earlier sections. RF algorithm is used for labeling which has a linear relationship with respect to M and a small linear increase with increasing M can be observed in the illustration.

Furthermore, Fig. 6(b) and (c) illustrates the empirical analysis for training time with respect to parameters J , D and T respectively. Fig. 6(c) illustrates a linear increase in training time with respect to both D and T that is expected from a forest-based learning algorithm. Furthermore, a linear increase with respect to J is also consistent with the algorithmic complexity of mv-ARF. As for the testing complexity, the empirical analysis for parameters J and D , T are illustrated in Fig. 7(b) and (c) respectively. A linear increase with respect to J , D and T is consistent with the algorithmic complex-

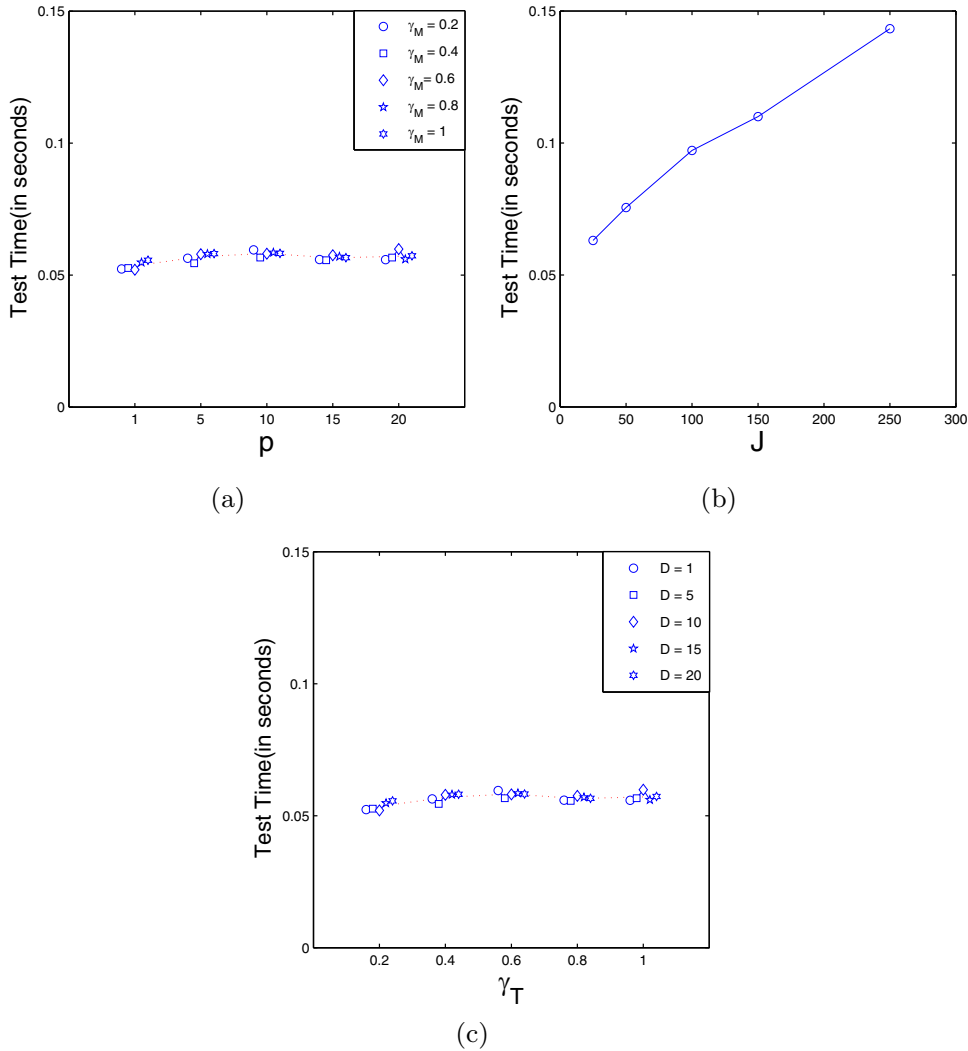


Fig. 7. Testing times for selected values of J , D , p , γ_M and γ_T acquired from *Kick vs. Punch* dataset. (10 replications).

ity. Note that, J has a linear impact for both training and testing time whereas the sensitivity results show that the impact of J on testing accuracy becomes almost negligible for very large J . These results lead to the conclusion that, it would be better to limit J to a smaller value for efficiency concerns. For a single MTS, the testing complexity is $O(J)$ since $J \geq T$, D . Thus, T and D has negligible impact on testing time compared to J . Moreover, linear growth rates for each parameter shows that mv-ARF is efficient with its parameter setting and also it scales well with large datasets that contains long MTS with high number of attributes. Also, training mv-ARF algorithm is very fast in practice. Distance-based methods don't scale as well with increasing T and M . Thus, for larger datasets, mv-ARF is considerably faster in practice than the distance-based methods discussed in earlier sections.

A similar analysis is made for AR kernel method since the discussion on complexity is limited in [12]. Two parameters (α and p) are considered in the complexity experiments of AR Kernel method since Cuturi and Doucet [12] consider only the lag order and α as parameters that needs to be fine-tuned in the model. The considered values for the model parameters are; $\alpha = [0.1, 0.3, 0.5, 0.7, 0.9]$ and $p = [1, 5, 10, 15, 20]$. The dataset used for the analysis is the same with mv-ARF as well as the ratio parameter. Fig. 8 illustrates the results of our experiments. The empirical complexity for training is the elapsed time for calculating the distance matrix. Fig. 8(a, b) illustrates the results of the empirical

complexity for training. The results show that there is a non-linear increase in training time with respect to the lag order. Similarly, there is large increase with respect to the length parameter which may suggest that the model may not scale well with datasets with long MTS which requires a larger lag order. On the other hand, the results show that the α parameter do not have a significant impact on complexity. Moreover, the algorithm scales well with increasing M , which is sensible since the primary motivation of the work in [12] is to provide a distance measure which can efficiently handle structured data. Fig. 8(c, d) illustrates the results of empirical test complexity. Test complexity is the elapsed time to label a single test MTS. The results show that the testing time is empirically small and none of the parameters have significant impact on testing time which is essentially the complexity of labeling from Support Vector Machine with one-vs-rest approach.

mv-ARF algorithm mostly consists of operations that can be parallelized. Also, if further enhancements in computational time is required, implementation of mv-ARF allows parallel use of multiple CPUs. Fig. 9 shows the changes in training time of mv-ARF with increasing number of cores used in parallel for two large datasets. Even though there is an unmistakable decrease in training time, it is not strictly linear. In theory, a steeper decrease in duration is expected. But due to the inter-process communication overhead within the operating system, a forest cannot be trained in exactly half of the regular duration by doubling the number of cores. Still,

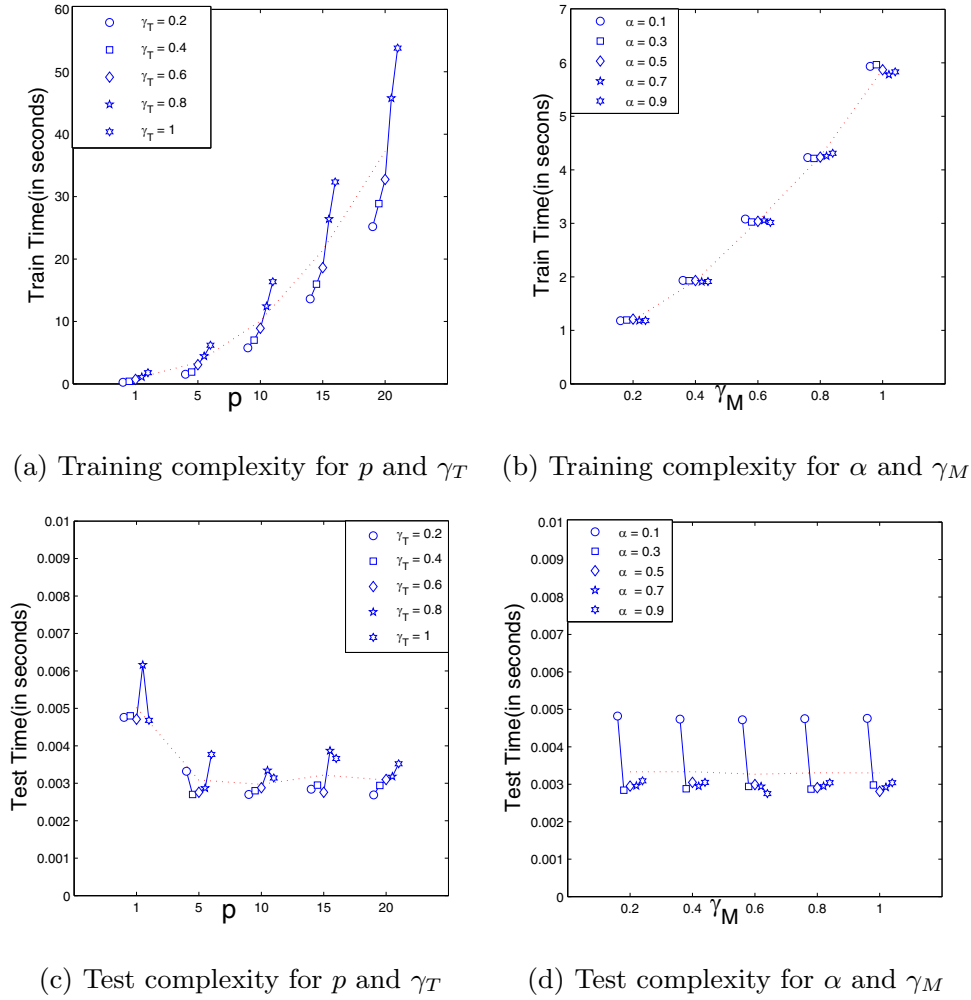


Fig. 8. Empirical complexity tests for AR kernel method with selected values of α , p , γ_M and γ_T acquired from Kick vs. Punch dataset.

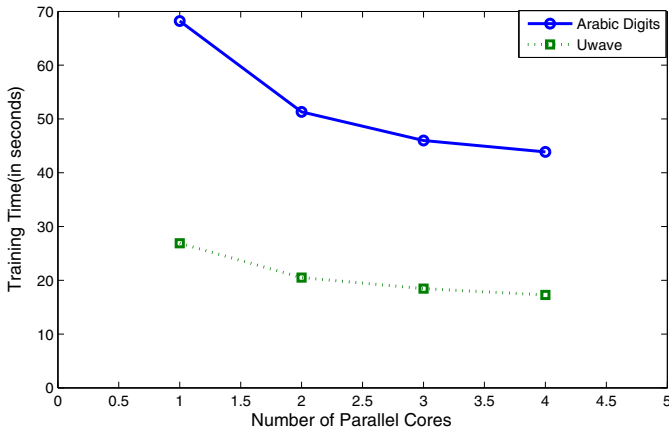


Fig. 9. Training times for Arabic Digits and Uwave datasets with parallelized operations. (15 replications).

significant improvements can be observed with paralleled operations, especially for large datasets.

6. Conclusions

In this work, a novel methodology for MTS modeling called mv-ARF is proposed. Modeling strategy of mv-ARF aims to explain

the behavior of MTS using autoregressive, multivariate and supervised learners. Proposed method is a generative approach in nature but its capabilities are tested within MTS classification framework. Autoregressive lag columns are generated for each feature which is the only feature extraction required for mv-ARF as opposed to exhaustive and time-consuming feature extraction steps which is common for feature-based approaches in literature. For MTS classification, different dynamic behaviors associated with class labels are explained by using tree-based ensemble learners with multivariate response. Multivariate tree-based ensembles are built for each class label and an error-based representation is achieved using prediction errors coming from each model. By utilizing tree-based ensembles with AR components, the model is able to capture any linear and non-linear relationships between past and current values. Furthermore, each attribute is handled simultaneously, thus the interactions and relations between separate variables and their past values are included within the model. Error-based representation includes the MSE's for each time series coming from each model. Any supervised classifier can be applied to this error-based representation. This way, mv-ARF modeling strategy provides a very simple, efficient and robust classifier for MTS classification.

In addition to its simplicity and practicality, experimental results show that mv-ARF is robust against parameters, and is able to provide efficient and satisfactory results that are competitive with its counterparts. Moreover, a univariate version of mv-ARF is also designed for comparison and it has been shown that the multivari-

ate version yields considerably better results. Experimental results leads to the conclusion that; in MTS analysis, utilizing multivariate approaches gives a better understanding of the dynamics underlying MTS than using univariate approaches because MTS are not simply defined by their attributes.

Acknowledgments

This research is supported by [Bogazici University Research Fund](#) (Grant number: [BAP 14A03SUP6](#)) and [Air Force Office of Scientific Research Grant FA9550-17-1-0138](#).

References

- [1] Z. Bankó, J. Abonyi, Correlation based dynamic time warping of multivariate time series, *Expert Syst. Appl.* 39 (17) (2012) 12814–12823.
- [2] F. Bashir, W. Qu, A. Khokhar, D. Schonfeld, Hmm-based motion recognition system using segmented pca, in: *IEEE International Conference on Image Processing 2005*, 3, IEEE, 2005, pp. III–1288.
- [3] M.G. Baydogan, *LPS*, 2016.
- [4] M.G. Baydogan, G. Runger, Learning a symbolic representation for multivariate time series classification, *Data Min. Knowl. Discov.* 29 (2) (2015) 400–422.
- [5] M.G. Baydogan, G. Runger, Time series representation and similarity based on local autopatterns, *Data Min. Knowl. Discov.* 30 (2) (2016) 476–509.
- [6] D.J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, in: *KDD Workshop*, 10, Seattle, WA, 1994, pp. 359–370.
- [7] L. Breiman, Random forests, *Mach. Learn.* 45 (1) (2001) 5–32.
- [8] K.-P. Chan, A.W.-C. Fu, Efficient time series matching by wavelets, in: *Data Engineering, 1999. Proceedings., 15th International Conference on*, IEEE, 1999, pp. 126–133.
- [9] B. Chandra, M. Gupta, M. Gupta, A multivariate time series clustering approach for crime trends prediction, in: *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*, IEEE, 2008, pp. 892–896.
- [10] CMU, *Cmu graphics lab motion capture database*, (2012).
- [11] M. Cuturi, Autoregressive kernels, 2016.
- [12] M. Cuturi, A. Doucet, Autoregressive kernels for time series, *arXiv preprint arXiv:1101.0673* (2011).
- [13] G. De'Ath, Multivariate regression trees: a new technique for modeling species–environment relationships, *Ecology* 83 (4) (2002) 1105–1117.
- [14] R.F. Engle, Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation, *Econometrica* (1982) 987–1007.
- [15] L. Friedman, L. Olshen, C.J. Stone, et al., Classification and regression trees, *Wadsworth Int. Group* 93 (99) (1984) 101.
- [16] J. Grabocka, M. Wistuba, L. Schmidt-Thieme, Fast classification of univariate and multivariate time series through shapelet discovery, *Knowl. Inf. Syst.* (2015) 1–26.
- [17] X. Hu, V. Nenov, Multivariate ar modeling of electromyography for the classification of upper arm movements, *Clin. Neurophysiol.* (2004).
- [18] I. Karlsson, P. Papapetrou, H. Boström, Generalized random shapelet forests, *Data Min. Knowl. Discov.* 30 (5) (2016) 1053–1085, doi:[10.1007/s10618-016-0473-y](#).
- [19] M. Lichman, *UCI machine learning repository*, 2013.
- [20] J. Lin, E. Keogh, S. Lonardi, B. Chiu, A symbolic representation of time series, with implications for streaming algorithms, in: *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, ACM, 2003, pp. 2–11.
- [21] R. Olszewski, <http://www.cs.cmu.edu/~bobski/>, 2012.
- [22] H. Raman, N. Sunilkumar, Multivariate modelling of water resources time series using artificial neural networks, *Hydrol. Sci. J.* 40 (2) (1995) 145–163.
- [23] M. Seeger, Covariance kernels from Bayesian generative models, *Adv. Neural Inf. Process. Syst.* 2 (2002) 905–912.
- [24] Y.C. Sübakan, B. Kurt, A.T. Cemgil, B. Sankur, Probabilistic sequence clustering with spectral learning, *Digit. Signal. Process.* 29 (2014) 1–19.
- [25] H. Tong, *Non-Linear Time Series: A Dynamical System Approach*, Oxford University Press, 1990.
- [26] K. Tuncel, *mv-ARF source code*, 2017.
- [27] G. Van Rossum, F.L. Drake Jr, *Python reference manual*, Centrum voor Wiskunde en Informatica Amsterdam, 1995.
- [28] X. Weng, J. Shen, Classification of multivariate time series using two-dimensional singular value decomposition, *Knowl. Based Syst.* 21 (7) (2008) 535–539.
- [29] K. Yang, C. Shahabi, A pca-based similarity measure for multivariate time series, in: *Proceedings of the 2nd ACM International Workshop on Multimedia Databases*, ACM, 2004, pp. 65–74.

Kerem Tuncel has received his M.Sc. from the Department of Industrial Engineering in Bogazici University in 2017. He is interested in statistical learning, time-series data mining and multi-modal learning. He received his B.Sc. degree from Industrial Engineering Department within Bilkent University.

Mustafa Gökçe Baydoğan is an assistant professor in Department of Industrial Engineering at Boğaziçi University, Istanbul, Turkey. Before joining Boğaziçi University, he worked as a postdoctoral research assistant in the Security and Defense Systems Initiative at Arizona State University (ASU) between 2012–2013. He received his Ph.D. degree in Industrial Engineering from ASU in 2012. His B.S. and M.S. degrees are in Industrial Engineering both from Department of Industrial Engineering at Middle East Technical University, Ankara, Turkey in 2006 and 2008 respectively. His current research interests focus on statistical learning, with applications in temporal data mining (time series and images) and data mining for massive, multivariate data sets. Details about him and his work can be reached through: <http://www.mustafabaydogan.com>.

A Data Adaptive Biological Sequence Representation for Supervised Learning

Hande Cakin, Berk Gorgulu, Mustafa Gokce Baydogan, Na Zou & Jing Li

Journal of Healthcare Informatics Research

ISSN 2509-4971

Volume 2

Number 4

J Healthc Inform Res (2018) 2:448-471

DOI 10.1007/s41666-018-0038-5



 **Springer**

Your article is protected by copyright and all rights are held exclusively by Springer Nature Switzerland AG. This e-offprint is for personal use only and shall not be self-archived in electronic repositories. If you wish to self-archive your article, please use the accepted manuscript version for posting on your own website. You may further deposit the accepted manuscript version in any repository, provided it is only made publicly available 12 months after official publication or later and provided acknowledgement is given to the original source of publication and a link is inserted to the published article on Springer's website. The link must be accompanied by the following text: "The final publication is available at link.springer.com".



A Data Adaptive Biological Sequence Representation for Supervised Learning

Hande Cakin, et al. *[full author details at the end of the article]*

Received: 15 September 2017 / Revised: 1 October 2018 / Accepted: 2 October 2018 /

Published online: 26 October 2018

© Springer Nature Switzerland AG 2018

Abstract

Proper expression of the genes plays a vital role in the function of an organism. Recent advancements in DNA microarray technology allow for monitoring the expression level of thousands of genes. One of the important tasks in this context is to understand the underlying mechanisms of gene regulation. Recently, researchers have focused on identifying local DNA elements, or motifs to infer the relation between the expression and the nucleotide sequence of the gene. This study proposes a novel data adaptive representation approach for supervised learning to predict the response associated with the biological sequences. Biological sequences such as DNA and protein are a class of categorical sequences. In machine learning, categorical sequences are generally mapped to a lower dimensional representation for learning tasks to avoid problems with high dimensionality. The proposed method, namely SW-RF (sliding window-random forest), is a feature-based approach requiring two main steps to learn a representation for categorical sequences. In the first step, each sequence is represented by overlapping subsequences of constant length. Then a tree-based learner on this representation is trained to obtain a bag-of-words like representation which is the frequency of subsequences on the terminal nodes of the tree for each sequence. After representation learning, any classifier can be trained on the learned representation. A lasso logistic regression is trained on the learned representation to facilitate the identification of important patterns for the classification task. Our experiments show that proposed approach provides significantly better results in terms of accuracy on both synthetic data and DNA promoter sequence data. Moreover, a common problem for microarray datasets, namely missing values, is handled efficiently by the tree learners in SW-RF. Although the focus of this paper is on biological sequences, SW-RF is flexible in handling any categorical sequence data from different applications.

Keywords Gene expression · Biological sequences · Time series · Categorical · Classification · Representation learning

This article is part of the Topical Collection on *Special Issue on Data Mining in Healthcare Informatics*

1 Introduction

Machine learning on large biological sequence databases has received considerable interest over the past decade with the increase in the sequential data due to the advancements in the microarray technology. An important challenge for the analysis of these sequential databases is coming from the high dimensionality of the sequences.

Much of the research has focused on finding a high-level representation by transforming the original data to another domain to reduce the dimension and capture certain relevant information. On the other hand, efficient analysis of the sequential data is not only a concern of bioinformaticians [4]. Similar problems exist in different domains such as online marketing where the sequence of pages or the path viewed by users are analyzed to understand the purchasing behavior [18] and telecommunication where the task is to predict a future event (i.e., the failure of a piece of equipment) based on past events (i.e., the alarm messages) [22].

With the recent advancements in whole-genome approaches to measure the gene expression, researchers have focused on learning the structural and dynamical properties of gene regulatory networks (GRN) [4]. A GRN consists of regulators which interact with each other and with other substances to govern the gene expression levels of mRNA and proteins. The regulators can be DNA, RNA, protein, and combination of these. The human genome has approximately 20000 protein-coding genes and these genes are expressed to generate proteins at specific levels in specific cells [14]. The creation of body structures (i.e., cells or tissues) is achieved by this way. Important factors driving this process are transcription factors (TFs). Regulation of genes are achieved by specialized proteins known as transcription factors which “bind” to the short sequences on the DNA and these binding sites are called TF binding sites (TFBSs). Describing the gene expression as a function of regulatory inputs specified by interactions between proteins and DNA is important to understand the underlying mechanisms by which genes are regulated [4]. Generally, pattern recognition algorithms are used to identify the sets of DNA sequence elements, or motifs, that are predictive of gene’s expression pattern. Genetic variations in the DNA sequence may indicate a person’s risk of particular disease or reaction to certain medications [17]. For example, in cancer genomics, a certain amount of specific genes have been identified in malignant tumors. DNA sequence could be used to detect variations or mutations in the established panel of genes for immediate treatment decisions and potential response to chemotherapy drugs. The second example is to use whole-genome sequence for predicting a future risk of specific disease even though without known clinical significance from found variants. A healthy newborn could be screened for a potential risk of genetic disease onset in early childhood. It is important to understand DNA sequence so that decision makers or healthcare professionals could determine when and how to use sequencing.

Biological sequences such as DNA is a class of sequential data. Sequential data can be described as the collection of events that take place in an order and events can be represented with categories (i.e., nucleotides), numbers (i.e. temperature), or a combination of both (i.e., network traffic direction and payload size) [24]. Sequential data-related problems have an important difference compared to regular data mining

problems since the information is embedded in the order of the sequence. In other words, there is no explicit feature regarding the data point. In traditional learning tasks, each data point is represented with a feature vector describing the characteristics. However, sequential data is represented with the consecutive observations and the length of the sequences has potential to be very large depending on the sequential process. Therefore, an additional step is required to represent given sequences in a more effective manner.

The methods to represent categorical sequences can be categorized under three main groups, namely, similarity-based approaches, feature-based approaches, and model-based approaches. Similarity-based approaches are one of the most popular techniques in which alignment-based kernels are introduced to solve the classification problem with support vector machines (SVM) [11]. Depending on the area of application, similarity-based methods can be successful; however, they lack in interpretability. Additionally, computational performance deteriorates substantially with increasing data size as they require to store complete training data and calculate similarity between each sequence.

Besides from similarity-based approaches, there are model-based approaches. These approaches aim to discover underlying model of the observed sequences and use this model (i.e., logic) for classification of the new instances. Under the assumption of sequences having a parametric distribution, Markovian models are introduced as probabilistic model-based methods. These methods process using conditional probability of the occurrences under Markovian Property. Variations of Markov models are used in the field of categorical sequence representation. Hidden Markov Models (HMM) are the most famous approach to represent sequential data in this category [6]. These approaches learn a model for the sequence and use model parameters (i.e., learned transition and emission probabilities) as an input to learning algorithms. Model-based methods are generalizable and applied successfully on various domains and problems. However, training HMMs might be problematic since there are several parameters to tune and learn (i.e., number of hidden states and state transition probabilities) and the performance is affected significantly with the choice of initialization parameters.

Computational inefficiency and non-interpretable nature of similarity-based and model-based approaches have made researchers focus on feature-based approaches. The most popular approach in this category is the k -mers method. After all possible combinations of the observations are considered to generate words of length k , frequency of these words are used to represent the sequence. For instance, 3-mers representation of a DNA sequence is a feature vector of length $4^3 = 64$ as there are 4 different nucleotides (i.e., A, T, G, C). It is a representation which has many applications in various domains such as speech recognition [9, 25], text categorization, and protein and genomic sequences. Furthermore, these methods perform significantly better than similarity-based approaches computationally since they do not require the storage of complete training data clustering. On the other hand, obtaining a representation with larger k values can be problematic for the cases where the number of categories is large. For example, there are protein sequences composed of 20 different letters [1]. Obtaining long words is almost impossible with k -mers representation for such datasets.

There are some variants of k -mers introduced to decrease the computational complexity and memory requirements of k -mers. KRAKEN [23] is a method that depends only the mapping of k -mers representations to the lowest common ancestors (LCA), namely, the deepest node that has two descendant nodes. It aims to decrease the complexity of k -mers without losing any performance. In order to speed up further and to decrease the memory requirements of the KRAKEN, [16] suggests CLARK. CLARK defines k -mers representation as k -spectrum which is a representation vector that has length of $4k$ and counts frequencies of all possible k -mers. This representation can be taught as a protector in order to decrease the effect of noise by eliminating the rare k -mers.

Although some variants of k -mers are introduced to speed up the process, computational complexity and memory requirements still are major problems in DNA sequencing. Considering this fact, data adaptive representations are proposed to reduce the dimensionality. k -mers is an unsupervised representation learning approach focusing on the generation of a frequency matrix [15, 16, 23]. However, a supervised representation learning that makes use of the class information of the sequences has potential to generate a scalable feature set [3]. These approaches make use of the response information for each sequence to obtain a compact representation.

This study proposes a novel supervised representation learning method, namely, sliding window-random forest (SW-RF) for categorical sequential data representation and classification. Proposed method can be utilized to find representative DNA sequence elements that are predictive of gene's expression. Moreover, it is capable of working with any biological sequence and all varieties of categorical sequences.

Essentially, SW-RF is a combination of sliding windows and a tree-based ensemble learner. Therefore, it inherits the advantages of the tree learners. SW-RF has the computational efficiency advantage of tree learners and can handle large dimensional data successfully. Moreover, it does not require to process each sequence separately since tree classifiers evaluate the data as a whole. Tree learner used in this study is the well-known classification and regression trees algorithm proposed by [8]. We evaluate the performance of the proposed representation method on several categorical sequence datasets. Firstly, we perform experiments on synthetic datasets to uncover the characteristics and evaluate the effectiveness of the approach. Then, a DNA promoter sequence dataset is used to illustrate the interpretability and accuracy of the approach on a real-world dataset. Moreover, we conduct further experiments on sequences with missing values and sequences with varying lengths to elaborate on the robustness of the approach in adverse conditions. To illustrate the generalizability of the proposed method, more experiments on protein sequences and numerical time series are also considered.

The remainder of this paper is organized as follows. Section 2 describes the framework for learning the representation. In Section 3, we illustrate the basics of the approach on a simple example. Sections 4 and 5 demonstrate the effectiveness and efficiency of our proposed approach by testing on synthetic and real-world datasets. Section 7 provides conclusions and potential future works.

2 Sliding Window - Random Forest Method

SW-RF method for categorical sequence representation and classification proceeds in four steps. The first step is to represent the given sequence with a sliding window and to assign class label of the corresponding sequence to all sub-windows (i.e., sub-sequences). The second step is to construct a decision tree classifier based on step one representation. The third step is generating a feature vector/matrix by using the frequency of each pattern that occurs on the terminal nodes of the decision tree/random forest. Finally, the last step is training any supervised learning algorithm on this representative vector.

Consider an observation sequence set X which is an $N \times T$ matrix where N is number of sequences and T is the length of each sequence. Then, $x_{nt} \in \{1, 2, 3, \dots, M\}$ represents observation at t^{th} position in sequence n . x_{nt} is a categorical variable with M levels and y_n is the corresponding response variable. Parameter M is referred to as alphabet size. For illustration purposes, the sequences are assumed to be of the same length, T , although our approach can handle the sequences of different lengths. Observed sequence X and response Y are defined as follows:

$$X = \begin{bmatrix} x_{11} & \dots & x_{1T} \\ \vdots & \ddots & \vdots \\ x_{N1} & \dots & x_{NT} \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

In the first step, the sliding window with pre-determined window size W is shifted horizontally one position at a time over each row of X to generate all subsequences of length W . All subsequences are merged to create a new feature matrix named as RF matrix. Note that for each categorical sequence, $T - W + 1$ subsequences are created and added to the RF . Point by point sliding scheme considers all possible subsequences of a length W which guarantees that the new representation does not miss any important patterns. For illustration, the representation SW_1 is obtained by applying sliding window over first sequence in X (i.e., $x_{11} \dots x_{1T}$) and RF_1 matrix is obtained by assigning the response variable y_1 to all subsequences from first sequence and provided below. The same procedure is repeated for each categorical sequence and matrices that are obtained are combined vertically to create RF . Therefore, the combined matrix RF has $N \times (T - W + 1)$ rows and $(W + 1)$ columns:

$$SW_1 = \begin{bmatrix} s_1 & s_{\dots} & s_W \\ x_{11} & \dots & x_{1W} \\ x_{12} & \dots & x_{1(W+1)} \\ \vdots & \ddots & \vdots \\ x_{1(T-W+1)} & \dots & x_{1T} \end{bmatrix} \quad RF_1 = \begin{bmatrix} s_1 & s_{\dots} & s_W & y \\ x_{11} & \dots & x_{1W} & y_1 \\ x_{12} & \dots & x_{1(W+1)} & y_1 \\ \vdots & \ddots & \vdots & \vdots \\ x_{1(T-W+1)} & \dots & x_{1T} & y_1 \end{bmatrix}$$

$$SW_n = \begin{bmatrix} s_1 & s_{\dots} & s_W \\ x_{n1} & \dots & x_{nW} \\ x_{n2} & \dots & x_{n(W+1)} \\ \vdots & \ddots & \vdots \\ x_{n(T-W+1)} & \dots & x_{nT} \end{bmatrix} \quad RF_n = \begin{bmatrix} s_1 & s_{\dots} & s_W & y \\ x_{n1} & \dots & x_{nW} & y_n \\ x_{n2} & \dots & x_{n(W+1)} & y_n \\ \vdots & \ddots & \vdots & \vdots \\ x_{n(T-W+1)} & \dots & x_{nT} & y_n \end{bmatrix}$$

$$SW = \begin{bmatrix} s_1 & s_{\dots} & s_W \\ x_{11} & \dots & x_{1W} \\ x_{12} & \dots & x_{1(W+1)} \\ \vdots & \ddots & \vdots \\ x_{1(T-W+1)} & \dots & x_{1T} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ x_{N1} & \dots & x_{NW} \\ x_{N2} & \dots & x_{N(W+1)} \\ \vdots & \ddots & \vdots \\ x_{N(T-W+1)} & \dots & x_{NT} \end{bmatrix} \quad RF = \begin{bmatrix} s_1 & s_{\dots} & s_W & y \\ x_{11} & \dots & x_{1W} & y_1 \\ x_{12} & \dots & x_{1(W+1)} & y_1 \\ \vdots & \ddots & \vdots & \vdots \\ x_{1(T-W+1)} & \dots & x_{1T} & y_1 \\ \vdots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \vdots & \vdots \\ x_{N1} & \dots & x_{NW} & y_N \\ x_{N2} & \dots & x_{N(W+1)} & y_N \\ \vdots & \ddots & \vdots & \vdots \\ x_{N(T-W+1)} & \dots & x_{NT} & y_N \end{bmatrix}$$

The second step is to construct a tree-based ensemble on the combined representation in Step 1 (RF). The base learner used in this ensemble is classification trees proposed by [8]. Classification trees recursively partition the input space by making binary splits until a termination condition is met. As a result of each split, observations are divided into two groups (nodes) according to a splitting rule. Splitting rule is determined with the aim of gathering similar observations together. In mathematical sense, splits are selected to minimize the total impurity in the terminal (i.e., leaf) nodes. Classification trees use Gini index as impurity measure and selects the split that maximizes the information gain in corresponding nodes. The motivation of using an ensemble, namely, random forests, is to avoid from greedy nature of a single tree [7].

The ensemble is trained on the given RF matrix that has the response vector y as the label and the column vectors $s_1, \dots, s_W$ representing the symbols as predictors. We are interested in quantifying how the subsequences generated from categorical sequences (RF) are distributed on the terminal nodes. This is very similar to a bag-of-words (BoW) approach traditionally used in computer vision problems. By applying a sliding window ($T - W + 1$) subsequences are created for each categorical sequence. When a classification tree is trained on such data, subsequences of each sequence reside in a certain terminal node. The subsequences in the same terminal nodes share similar characteristics because of the recursive partitioning scheme of the decision trees and a sequence can be represented by the frequency of its subsequences at each terminal node. Then we obtain the final representation where rows correspond to sequences and columns represent the terminal nodes of the decision tree. As mentioned, we fit J trees on a random subsample of features and instances as in random forests [7] to avoid the potential problems with overfitting. The trees are trained in a breadth-first order and the final representation is the concatenation of the frequency vectors. We benefit from the desirable characteristics of the tree-based learning (i.e., computational efficiency, missing value handling by surrogate splits and etc.) at this stage.

Last step consists of an application of a supervised classification method to the proposed representation. Any supervised learning algorithm can be utilized on the

proposed representation. However, extracted feature space becomes wide and sparse with the increasing number of terminal nodes. In order to facilitate interpretability with the proposed sparse representation, we train a lasso logistic regression model as the final classifier. The lasso regression model has a shrinkage effect, which would impose sparsity on feature selection in the model. It forces some coefficients to be zeros and others non-zeros. These predictors with non-zero coefficients are considered as having relatively higher impact on the response variable.

3 Illustration

All steps of SW-RF are described using a sample dataset depicted in Table 1. There are $N = 200$ sequences where $T_n = 20$. The alphabet size, M , is equal to 5. There is a binary class variable y_n corresponding to each sequence. The sequences that has class variable $y_n = 1$ are manipulated by adding a class indicator sequence {a,b,c} to randomly selected positions in order to check the ability of the proposed model whether to capture indicator sequence or not.

SW-RF begins with constructing a sliding window representation for a given W . For illustration purposes, W is chosen as 3. Then RF_1 matrix, i.e., matrix that contains all subsequences of length 3 of categorical sequence 1 is generated as:

$$RF_1 = \begin{bmatrix} s_1 & s_2 & s_3 & y \\ e & a & d & 0 \\ a & d & d & 0 \\ d & d & b & 0 \\ d & b & a & 0 \\ \vdots & \ddots & \vdots & \vdots \\ c & c & b & 0 \end{bmatrix}$$

Note that y values of all rows in RF_1 are equal to 0 which is the label of the 1st categorical sequence in the dataset, because, if a categorical sequence belongs to a certain class, all of its subsequences should belong to the same class. RF matrix is constructed by merging subsequence matrices of all categorical sequences. In other words, row-wise stacking all RF_n for n in $1, \dots, N$. Therefore, when W is equal to 3, sliding window representation matrix RF has $(20 - 3 + 1) \times 200 = 3600$ rows and $3 + 1 = 4$ columns. Table 2 represents RF matrix with corresponding categorical sequence ids (SeqId) and subsequence ids (SubseqId).

For the sake of easy interpretation, only one tree (i.e., $J = 1$) is trained on the given RF matrix that has classification labels y as response vector and s_1, s_2, s_3 , as predictors. The resulting tree is schematized in Fig. 1. Our aim is to find a mapping for each sequence by using terminal nodes of this decision tree which are displayed as rectangles in the Fig. 1. There are 12 terminal nodes in the following tree and each of them has a unique ID as it can be seen in the Fig. 1. Unique terminal node IDs are {5, 7, 8, 9, 11, 15, 16, 17, 20, 21, 22, 23}.

Application of sliding window constructs $(20 - 3 + 1) = 18$ subsequences for each categorical sequence. Each of these subsequences are given as an observation

Table 1 First 25 categorical sequences of the sample dataset, $N = 200$, $T_n = 20$, $M = 5$, $y_n \in \{0, 1\}$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	y
1	e	a	d	d	b	a	c	e	e	b	b	c	e	d	a	a	b	c	c	b	0
2	b	b	b	c	d	c	c	a	e	c	c	a	c	a	b	c	e	b	a	e	1
3	a	b	a	a	b	c	d	d	c	d	b	c	a	b	b	c	e	d	c	a	1
4	b	b	e	a	b	a	b	a	a	a	d	b	b	b	b	a	b	c	a	b	1
5	c	e	e	d	e	c	e	d	b	d	b	a	a	b	c	b	a	c	c	d	1
6	d	b	a	d	e	d	c	d	c	d	c	c	e	c	b	e	d	c	b	a	0
7	d	d	c	d	c	b	c	a	d	a	c	b	c	d	b	a	c	a	d	c	0
8	b	b	e	d	e	b	d	c	c	b	c	b	b	e	a	a	d	d	a	d	0
9	e	b	c	c	e	c	a	d	b	b	c	e	d	a	d	a	c	d	d	a	0
10	e	e	b	e	e	b	a	c	c	b	b	a	d	a	a	e	d	e	c	a	0
11	d	a	e	e	d	c	a	c	b	a	d	c	d	d	a	a	a	e	e	b	0
12	d	e	c	a	e	c	c	a	a	b	c	b	a	a	d	c	c	d	e	b	1
13	d	c	a	b	e	d	d	c	b	d	b	a	a	b	e	b	b	b	e	e	0
14	e	a	c	e	b	e	d	c	c	a	c	e	a	b	c	b	c	d	b	b	1
15	e	d	d	c	b	e	d	c	e	a	b	d	e	c	a	e	c	b	d	e	0
16	d	c	c	d	e	b	d	d	e	c	d	d	e	b	e	a	b	e	c	b	0
17	a	e	e	d	e	d	d	d	d	d	c	c	a	d	d	b	d	e	d	a	0
18	e	a	b	a	d	a	d	d	e	d	e	d	b	d	a	c	e	a	d	e	0
19	e	a	b	d	d	e	d	e	b	a	a	b	e	e	b	a	d	e	d	b	0
20	a	c	d	d	e	a	a	a	d	c	e	a	c	d	b	d	a	b	a	b	0
21	b	a	d	d	b	a	c	d	c	d	c	b	b	a	b	e	c	b	b	e	0
22	e	c	e	a	e	e	a	e	d	a	d	a	d	a	d	c	a	e	e	a	0
23	b	a	c	e	c	e	c	d	a	b	c	d	e	b	b	e	e	b	e	b	1
24	c	a	b	b	c	c	e	e	a	e	d	b	b	a	b	c	e	a	b	c	1
25	d	a	e	d	c	c	c	e	a	c	a	d	a	e	b	e	c	b	b	b	0

to a decision tree. Based on the decision tree in Fig. 1, each subsequence end up in a single terminal node. Therefore, each subsequence can be represented as a binary vector showing the presence of each subsequence in each node. Sample binary representation is shown in Table 3 with corresponding categorical sequence ids and subsequence ids.

SW-RF requires a representation for categorical sequences. A frequency vector is therefore constructed for each categorical sequence as the frequency of the subsequences residing at each terminal node. This procedure is realized simply by summing the binary vectors over their sequence ids (SeqId). Resulting representation contains a single frequency matrix for each categorical sequence. Table 4 demonstrates the final representation for the sample data and this representation. Each column in the final representation forms a feature vector and represented as φ_q where q represents the terminal node Ids and p is the number of terminal nodes which is equal to 12 for this example.

Table 2 *RF* representation with categorical sequence and subsequence ids for the sample dataset

SubseqId	SeqId	s1	s2	s3	y
1	1	e	a	d	0
2	1	a	d	d	0
3	1	d	d	b	0
4	1	d	b	a	0
5	1	b	a	c	0
6	1	a	c	e	0
7	1	c	e	e	0
8	1	e	e	b	0
.	.	.	.	.	.
.	.	.	.	.	.
1796	19	c	a	e	0
1797	19	a	e	e	0
1798	19	e	e	b	0
1799	19	e	b	c	0
1800	19	b	c	b	0
1801	20	b	a	d	0
1802	20	a	d	e	0
1803	20	d	e	e	0
1804	20	e	e	d	0
1805	20	e	d	d	0
.	.	.	.	.	.
.	.	.	.	.	.
3593	200	c	b	c	1
3594	200	b	c	a	1
3595	200	c	a	d	1
3596	200	a	d	a	1
3597	200	d	a	b	1
3598	200	a	b	d	1
3599	200	b	d	b	1
3600	200	d	b	a	1

Note that the row sums of Table 4 equals to 18 because 18 subsequences are created for each sequence when sliding window is applied. For instance, sequence 1 has no subsequence that fit the rule set of terminal node 7. However, it has 3 subsequences that fit the rules set of terminal node 16. Table 5 provides a visualization for the rule sets of terminal node 7 and 16 in detail. s_1 , s_2 , and s_3 are shown in the rule sets represent the first, second, and third columns (observation) in the sliding window representation which is provided in Table 2.

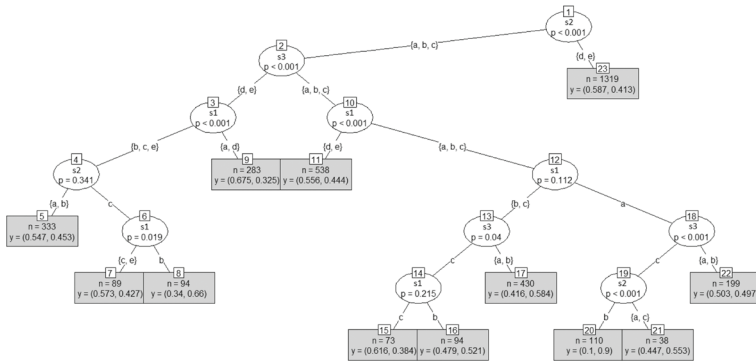


Fig. 1 A sample decision tree trained on representation RF

After obtaining the final representation (Table 4) for each categorical sequence, a lasso regression model is trained to predict the binary class variable. The regularization parameter λ is selected internally using 10-fold cross-validation and λ providing the best likelihood is selected for model training. We consider area under the ROC curve (AUC) as an evaluation metric in our calculations. AUC is a well-known and widely used metric for comparison of binary classification algorithms. AUC is known to be a better measure than accuracy empirically in terms of consistency and discriminancy as discussed by [13].

With the proposed representation, each feature refers to a certain rule that describes a pattern, features with non-zero coefficients are interesting in the sense that they imply important motifs related to the class. Fitted model is as follows:

$$y \sim 0.6109(\varphi_{20}) - 0.0379(\varphi_{23})$$

The model indicates that terminal node 20 is significant in order to predict binary class variable with positive sign and terminal node 23 is significant with a negative sign. When we look at the rule set of node 20 and node 23 in detail as in Table 6,

Table 3 Binary representations of the subsequences with respect to their presence on the terminal nodes

SubseqID	SeqID	Terminal nodes											
		5	7	8	9	11	15	16	17	20	21	22	23
1	1	1	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	1
3	1	0	0	0	0	0	0	0	0	0	0	0	1
4	1	0	0	0	0	1	0	0	0	0	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.
3599	200	0	0	0	0	0	0	0	0	0	0	0	1
3600	200	0	0	0	0	1	0	0	0	0	0	0	0

Table 4 Final representation table for the sample dataset

SeqID	Terminal nodes												Class
	5	7	8	9	11	15	16	17	20	21	22	23	
1	1	0	1	1	3	0	3	1	1	0	1	6	0
2	2	0	2	0	3	1	1	4	1	0	1	3	1
3	0	0	2	1	2	0	1	3	1	0	3	5	1
4	1	0	0	1	2	0	0	8	1	0	3	2	1
5	0	2	0	1	1	0	1	3	1	1	1	7	1
6	2	1	0	2	4	0	0	1	0	0	0	8	0
7	2	0	1	1	3	2	1	1	0	0	2	5	0
8	3	0	0	2	2	1	0	3	0	0	0	7	0
9	1	1	1	2	4	0	2	0	0	0	0	7	0
10	2	0	0	1	3	0	1	3	0	1	0	7	0
11	1	0	0	3	2	1	0	1	0	0	2	8	0
12	1	1	0	1	3	0	0	5	1	0	1	5	1
13	2	0	0	2	4	0	0	3	0	0	1	6	0
14	1	0	1	2	4	2	0	2	1	0	0	5	1
15	3	0	0	2	4	0	0	0	0	0	0	9	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.

we can see that node 20 has a rule set that are similar to our manipulative indicator sequence {a, b, c} and node 23 has the vice-versa. Moreover, Table 4 points that categorical sequences from class 1 are tend to include subsequences represented on terminal node 20; however, categorical sequences from class 0 are tend to include subsequences represented on terminal node 23. Average cross-validation AUC for the best lambda parameter is close to 1 as illustrated in Fig. 2. This also confirms the reliability of the model in terms of the accuracy.

4 Experimental Setup

SW-RF method is tested on both a synthetic and a real DNA sequence promoter dataset. We compare SW-RF to the state-of-the-art methods. Additionally, the performance of SW-RF is evaluated on modified synthetic datasets with varying lengths and missing values to illustrate the robustness of the approach in adverse conditions. SW-RF is implemented on R programming language and experimentation is performed on Windows 10 × 64 system with Intel Core i5-3337U CPU @ 1.60 GHz × 4 and 6 GB RAM. Although the CPU can handle four threads in parallel, only a single thread is used in the experiments.

Table 5 Rule set of terminal node 7 and terminal node 16

Terminal Nodes 7

- 1) $s2 == \{a, b, c\}$
- 2) $s3 == \{d, e\}$
- 3) $s1 == \{b, c, e\}$
- 4) $s2 == \{c\}$
- 6) $s1 == \{c, e\}$
- 7)* weights = 89

Terminal Node 16

- 1) $s2 == \{a, b, c\}$
- 2) $s3 == \{a, b, c\}$
- 10) $s1 == \{a, b, c\}$
- 12) $s1 == \{b, c\}$
- 13) $s3 == \{c\}$
- 14) $s1 == \{b\}$
- 16)* weights = 94

4.1 Synthetic Dataset

A synthetic dataset with varying problem characteristics is generated to characterize the behavior of the proposed approach in terms of accuracy and computational efficiency. Two HMM models are used to generate categorical sequences from different classes. In order to observe the performance of algorithms based on changing parameters, it is preferred to generate data sets with varying number of sequences, length of sequence and alphabet size. All combinations of these parameter summarized in the Table 7 was applied with five replications. Therefore, in total 120 datasets are generated with respective parameters.

Table 6 Rule set of terminal node 20 and terminal node 23

Terminal Node 20

- 1) $s2 == \{a, b, c\}$
- 2) $s3 == \{a, b, c\}$
- 10) $s1 == \{a, b, c\}$
- 12) $s1 == \{a\}$
- 18) $s3 == \{c\}$
- 19) $s2 == \{b\}$
- 20)* weights = 110

Terminal Node 23

- 1) $s2 == \{d, e\}$
- 23)* weights = 1319

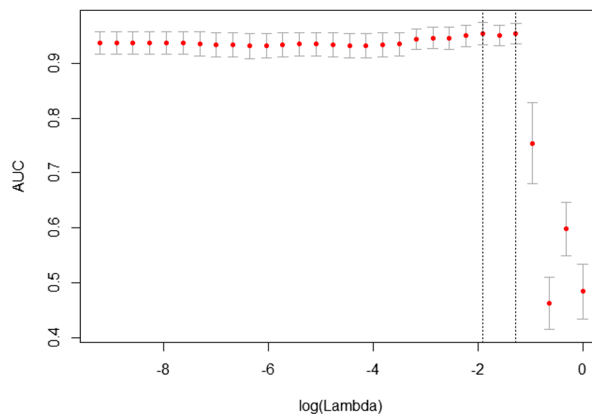


Fig. 2 Plot of AUC with varying lambda values. The dashed line on the left represent the lambda value providing the maximum average AUC whereas the one on the left is the lambda value which gives the most regularized model such that performance is within one standard error of the maximum average AUC

4.2 DNA Sequence Promoter Dataset

A DNA promoter sequence data from eight human cell lines, which are H1-hESC, GM12878, HMEC, HSMM, HUVEC, K562, NHEK, and NHLF. The karyotype of K562 is cancer and others have normal karyotype. Table 8 provides the details about the characteristics of the cells. Naturally, there are 23 chromosome for each cell, in which 22 are autosomes and 23rd pair determines the gender. The letters of each chromosome of each cell line are from the DNA alphabet A, C, G, and T. Each chromosome have different number of genes, but each genes of each chromosome have the same length which is equal to 2100.

Moreover, the response variable for each gene of each chromosome is included in the dataset. The response variable is the gene expression that refers to complex processes in DNA is converted into a functional substance. It is a binary variable, if the genetic information stored in a gene is used to produce a substance then gene expression becomes “1”, otherwise it is “0”. To sum up, each row in data set corresponds to one gene and for each gene sequence there is a corresponding gene index binary variable. Using DNA Sequence Promoter Dataset it is aimed to classify a specific gene whether it has gene expression of “1” or “0”.

Table 7 Parameter settings that used to generate synthetic dataset

Parameter	Values
The number of sequences (N)	{50, 100, 250}
The length of sequence n (T_n)	{50, 100, 200, 500}
Alphabet size (M)	{5, 10}

Table 8 Cell lines used as the source of experimental material

Cell	Tier	Lineage	Tissue	Karyotype	Sex
K562	1	Mesoderm	Blood	Cancer	F
GM12878	1	Mesoderm	Blood	Normal	F
NHEK	3	Ectoderm	Skin	Normal	U
NHLF	3	Endoderm	Lung	Normal	U
HUVEC	2	Mesoderm	Blood vessel	Normal	U
HSMM	3	Mesoderm	Muscle	Normal	U
H1-hESC	1	Inner cell mass	Embryonic stem cell	Normal	M

5 Algorithm Evaluation

We compare SW-RF with k -mers algorithm and HMM models with different parameter settings. For each method, a related representation is obtained by applying given methods. All representation methods have method-related parameters such as number of hidden states for HMM, value of k for k -mers and the window size W for SW-RF. Parameters of the each method are chosen by 10-fold cross-validation from the parameter values provided in Table 9. After the representations are learned a lasso logistic regression is trained where λ values are selected by 10-fold cross-validation. Unless otherwise stated, the number of trees J is set to 100. Random forests are known to provide robust results if the number of trees are set large enough [7] and we have identified that predictive performance stabilizes around 60 trees based on our preliminary experiments.

State-of-the-art methods are compared in terms of their theoretical complexity as well as classification performance (AUC) and computational times. Time complexity of sliding window algorithm is $O(T_n W)$ where the length of the sequence is T_n and W is the determined window size. Time complexity of k -mers representation is $O(V)$ where V stands for number of distinct k -mers, namely, equals to M^k where M be the alphabet size and k stands for length of subsequences [5]. Space complexity of k -mers representation is $O(Vk)$ and it also increase exponentially with k . Time complexity of HMM representation which is used with Baum-Welch inference algorithm has a time complexity of $O(H^2 T_n) * (\# \text{ of iterations})$ where H is the number of hidden states and T_n is the length of the sequence.

Table 9 Set of parameters considered in 10-fold cross-validation

Method	Parameter	Values
HMM	Number of hidden states (H)	2, 4, 6
k -mers	Length of words (k)	3, 4, 5
SW-RF	Window size (W)	3, 4, 5
SW-RF	Maximum number of terminal node (R)	36, 64, 128

Table 10 AUC values on synthetic data set with different length of sequence, number of instances and alphabet size against competitor approaches

Length of sequence	Number of instances	Alphabet size = 5			Alphabet size = 10		
		HMM	<i>k</i> -mers	SW-RF	HMM	<i>k</i> -mers	SW-RF
		Average of AUC	Average of AUC	Average of AUC	Average of AUC	Average of AUC	Average of AUC
50	50	0.542	0.553	0.647	0.497	0.510	0.558
50	100	0.630	0.610	0.747	0.500	0.547	0.719
50	250	0.676	0.660	0.766	0.503	0.594	0.778
100	50	0.614	0.612	0.848	0.507	0.548	0.694
100	100	0.683	0.671	0.835	0.499	0.575	0.817
100	250	0.727	0.768	0.854	0.496	0.665	0.874
200	50	0.618	0.718	0.932	0.495	0.608	0.903
200	100	0.741	0.820	0.940	0.510	0.687	0.932
200	250	0.803	0.875	0.946	0.502	0.770	0.952
500	50	0.663	0.854	0.991	0.495	0.742	0.991
500	100	0.835	0.934	0.992	0.506	0.820	0.995
500	250	0.865	0.977	0.994	0.499	0.891	0.997
Grand average		0.700	0.754	0.874	0.501	0.663	0.851

5.1 Comparative Performance on Synthetic Dataset

According to reported AUC results on Table 10, proposed method provides significantly better results than HMM and *k*-mers for each parameter combination of the synthetic data set. Computational time analysis has been done for two metrics, namely, preparation time of competitor approaches HMM, *k*-mers and proposed SW-RF representations, and training and testing time of classification process on each representation. Each algorithm is implemented in R Software and performed on Windows 10 × 64 system with Intel Core i5-3337U CPU @ 1.60 GHz × 4 and 6 GB RAM.

Computational time required for SW-RF and competitor approaches are provided in Table 11. Time required for HMM does not change significantly with changing problem parameters. As the alphabet size increases, computational time for *k*-mers increases significantly with the increase in the representation size. Computational requirements for SW-RF are not affected significantly with the changes in problem parameters. Computational performance of the SW-RF representation is also examined under different parameter settings to make the characteristics of the proposed algorithm clear. Interested parameter settings are window size for the sliding window, maximum number of terminal nodes and number of trees for random forest. Figure 3 illustrates the results of sensitivity analyses. Computation times increases with the increase in the window size from 10 to 25. Maximum number of terminal nodes cause a sharp increase in training and testing time, while a smooth increase in representation preparation time. Increasing number of decision trees causes an

Table 11 Computational performances of HMM, k-mers and SW-RF on synthetic data set with changing sequence length and number of instances

Length of sequence	Number of instances	Alphabet size = 10						Alphabet size = 10					
		HMM			k-mers			SW-RF			HMM		
		Preparation time	Train and test time	HMM test time	Preparation time	Train and test time	k-mers test time	Preparation time	Train and test time	SW-RF test time	Preparation time	Train and test time	SW-RF Preparation time
50	50	0.048	0.695	0.987	0.550	0.278	0.487	0.047	0.918	1.031	0.966	0.402	0.463
50	100	0.053	0.860	1.886	1.273	0.532	1.714	0.055	2.733	1.979	2.810	0.633	1.110
50	250	0.070	0.609	4.648	4.156	1.172	8.497	0.068	0.678	4.929	11.665	1.289	5.309
100	50	0.093	0.880	1.814	0.560	0.446	0.555	0.092	0.884	1.893	1.436	0.565	0.547
100	100	0.096	1.073	3.61	1.247	0.866	1.803	0.099	3.385	3.771	3.914	0.974	1.374
100	250	0.121	0.671	9.006	6.960	1.953	15.476	0.123	0.717	9.271	14.625	2.083	5.908
200	50	0.174	0.953	3.528	0.573	0.803	0.64	0.180	1.221	3.619	1.972	0.899	0.625
200	100	0.185	2.132	7.014	1.160	1.467	1.636	0.187	3.141	7.164	5.070	1.593	1.572
200	250	0.221	0.822	17.460	4.956	3.66	10.274	0.226	0.742	18.109	16.900	3.829	5.665
500	50	0.426	0.847	8.568	0.573	1.843	0.742	0.4379	1.102	8.895	2.780	1.925	0.717
500	100	0.457	1.684	17.287	1.133	3.470	1.616	0.460	3.520	17.669	6.411	3.544	1.471
500	250	0.539	1.229	43.218	3.320	9.612	5.203	0.551	0.852	44.773	19.546	9.673	4.385
Grand average		0.207	1.038	9.919	2.205	2.175	4.053	0.210	1.658	10.258	7.341	2.284	2.429

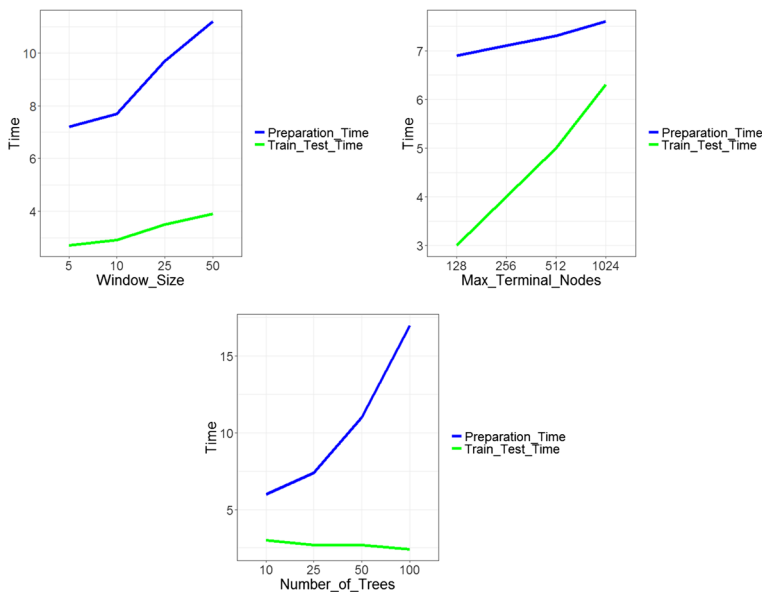


Fig. 3 Empirical complexity of SW-RF representation with respect to window size, max number of terminal nodes and number of decision tree respectively

increase in representation preparation time. The empirical behaviors are consistent with the theoretical complexity of SW-RF.

5.2 Performance on Data with Varying Lengths

The previous section indicates that SW-RF performs well on the dataset consists of sequences with constant length. SW-RF is also capable of handling sequences with variable lengths without losing its predictive accuracy. SW-RF generates a sliding window representation of the sequences in the datasets and this representation is given to a random forest classifier as an input. Therefore, each sequence is transformed to a collection of subsequences with equal length and provided to the random forest classifier. This allows SW-RF to handle sequences with varying lengths. Table 6 provides the performance of SW-RF on the synthetic data that is generated in different lengths. Length of the sequences in each dataset is selected uniformly from the interval provided. Also, noise is introduced to the data to make it harder for SW-RF. From Table 12, it can be observed that SW-RF works with high accuracy on the data with varying lengths without significant performance loss.

5.3 Performance on Data with Missing Values

SW-RF utilizes tree-based methods to obtain new representation. Decision trees can handle missing values using surrogate splits without any imputation [8]. Therefore SW-RF is also capable of working with missing data. For illustration, new datasets are created by 5, 10, 25, and 50 percent of the data from Table 13 with length of

Table 12 AUC values on synthetic data with varying lengths

Length of sequence	Number of instances	Alphabet size	
		10	20
		AUC	AUC
10–100	200	0.97	0.97
20–200	200	0.92	0.97
50–500	200	0.92	0.92
100–1000	200	0.75	0.84
Average performance		0.89	0.92

sequence 10–100, alphabet size 10, and number of instances 200. Table 13 provides the performance of SW-RF on these datasets. These results clearly illustrates that SW-RF can perform with missing values. Naturally, performance in the dataset decreases as missing ratio increases. However, SW-RF still performs well with missing values.

5.4 Comparative Performance on DNA Sequence Promoter Dataset

Proposed SW-RF representation, competitor HMM and k -mers representation methods are trained on DNA promoter sequence data set for each cell separately. Moreover, for each cell 10-fold cross-validation is applied for five replications. Therefore, presented results are averages over individual results of separate replications. We perform cross-validation to determine best parameter settings using the parameter levels in Table 9. Then, obtained parameter settings are used on the test part of the dataset. AUC values of the both performance are recorded in order to see whether there is a significant difference between AUC value of the cross-validation step and AUC value of the test step. The aim of such comparison is to show that SW-RF does not suffer from overfitting. When cross-validation AUC values are compared to test AUC values, there is not a significant difference that signs an overfitting. Consequently, results in following table indicates that our proposed SW-RF representation method outperforms HMM and k -mers representations for each cell. While HMM representation provides worst results in terms of AUC, k -mers representation provides moderate results that are close to results of HMM representation. In order to evaluate whether the difference between mean of the representation methods is significant or not, t test is conducted at the 0.05 significance level. Results also prove that

Table 13 AUC values on synthetic data with changing missing proportion

Missing ratio	AUC
0	0.970
0.05	0.902
0.1	0.886
0.25	0.782
0.5	0.686

Table 14 AUC values of HMM, k -mers and SW-RF on DNA sequence promoter data set

Cell	HMM		k -mers		SW-RF	
	Average of cvAUC	Average of testAUC	Average of cvAUC	Average of testAUC	Average of cvAUC	Average of testAUC
GM12878	0.565	0.567	0.612	0.612	0.770	0.766
H1hESC	0.603	0.603	0.620	0.621	0.770	0.766
HMEC	0.615	0.616	0.614	0.614	0.740	0.736
HSMM	0.602	0.603	0.603	0.603	0.720	0.717
HUVEC	0.601	0.602	0.624	0.624	0.777	0.774
K562	0.578	0.580	0.636	0.636	0.800	0.798
NHEK	0.595	0.597	0.634	0.634	0.782	0.779
NHLF	0.588	0.588	0.606	0.606	0.739	0.736
Grand average	0.593	0.594	0.619	0.619	0.765	0.758

mean difference between SW-RF and HMM representation, and SW-RF and k -mers representation is statistically significant (Table 14).

It is also possible to identify subsequences that have non-zero coefficients as illustrated in Section 3. For terminal nodes with non-zero coefficients, the distribution of the nucleotides is examined. Since, it is believed that detecting such sequences that have an effect on gene expression level may provide benefits with regards to biological researches. A similar study is conducted on the same data set previously by Zou [26]. A transfer learning approach on k -mers is proposed on the same DNA promoter sequence dataset to predict the same response [26]. These biological sequences are so called motifs and can be represented with a position-specific scoring matrix (PSSM) [19]. Figure 4 provides a motif logo example for the cell line GM12878 for $W = 5$. PSSM represents the distribution of the nucleotides for the corresponding terminal node of the tree. These results are consistent with the discussion that GC-rich (guanine-cytosine rich) DNA elements are important transcriptional regulatory elements in the promoter, enhancer, and locus control regions of many eukaryotic genes from several species [10, 21].

6 Evaluation of SW-RF on different applications

In the previous sections, success of our method is demonstrated on synthetic datasets as well as a real DNA sequence data. However, application domain of SW-RF method lays well beyond these applications. Proposed method is suitable for any kind of categorical sequence datasets and it can even be used to classify numerical sequences with some pre-processing. In addition, applications of SW-RF in biological sequences are not limited to only DNA sequence data, it can be applied to any biological sequence such as proteins, DNA, RNA, etc. In order to demonstrate the generalizability, SW-RF is applied to both protein classification and numerical time series classification tasks.

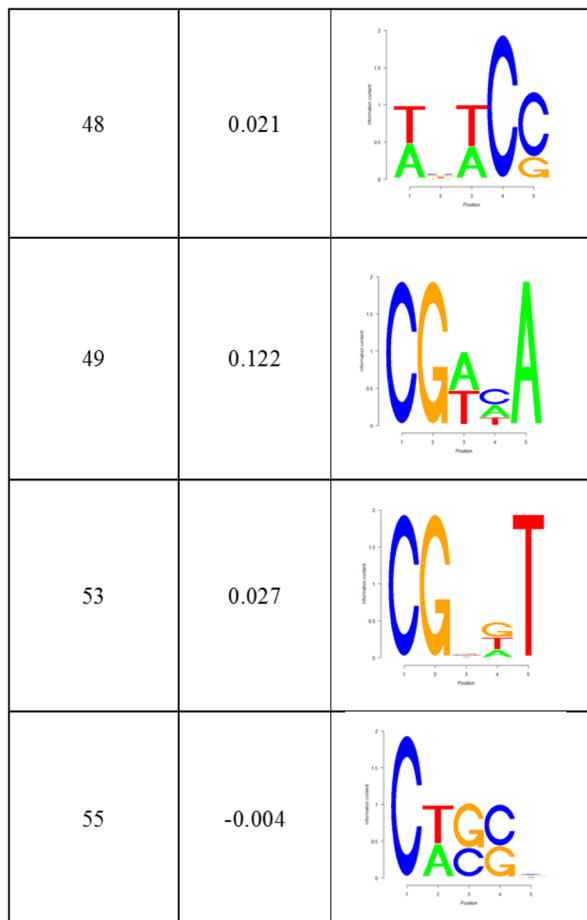


Fig. 4 Example for a motif logo of cell line GM12878, $W = 5$. First column refers to terminal node id. second column is the corresponding coefficient

6.1 Protein classification

Classification of the different protein families allows understanding the functional roles and structures of the proteins. In this experiment, SW-RF is applied to a protein dataset to determine the family of each protein sequence. The protein dataset is a sample of 2112 sequences taken from www.uniprot.org, where a sequence is from 20 alphabets (amino acids). Each sequence has one of two functions: “Might take part in the signal recognition particle (SRP) pathway” and “Binds to DNA and alters its conformation”. The functions are treated as the sequence labels. The experiments with 5-fold cross-validation yield that SW-RF with various window sizes w yields the AUC of 1. For demonstration purposes SW-RF with $w = 5$ is applied on the protein dataset and motif logo examples of important terminals are demonstrated in Fig. 5. For example, “YYDD” subsequence is a signature subsequence for “Might take part

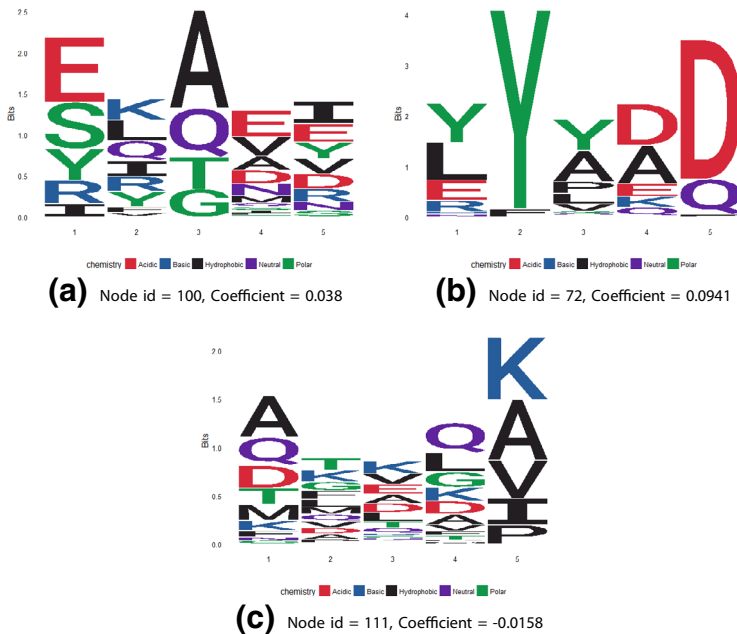


Fig. 5 Example for a motif logo of Protein data, $W = 5$

in the signal recognition particle (SRP) pathway” function. “Y” refers to amino acid tyrosine which is known to occur in proteins that are part of signal transduction processes [20].

6.2 Numerical time series classification

SW-RF method is essentially suggested for categorical sequences; however, it is possible to apply proposed method to any numerical time series with additional preprocessing. As an illustration, SW-RF is applied to 3 ECG datasets (ECG200, ECGFiveDays, TwoLeadECG) from [2]. ECG200 dataset contains two class of time series which are normal heartbeat and myocardial infarction. The aim is to distinguish heartbeats with myocardial infarction from normal healthy heartbeat. ECGFiveDays dataset contains ECG recordings of 67 years of male recorded in two different dates with the aim of distinguishing two types of recordings. TwoLeadECG is an ECG dataset contains two types of signals. The task is to classify signal 0 and signal 1.

Pre-processing requires discretization of the continuous time series which is realized by applying symbolic aggregate approximation (SAX) [12]. For a given alphabet size and dimensionality parameter SAX divides the series into intervals and represent them with their means. Alphabet size determines the number of distinct mean levels to be taken into account and dimensionality parameter decides number of observation to be considered for discretization. All ECG datasets are discretized by applying SAX with alphabet size α and dimensionality parameter β . Figure 6 contains an illustration

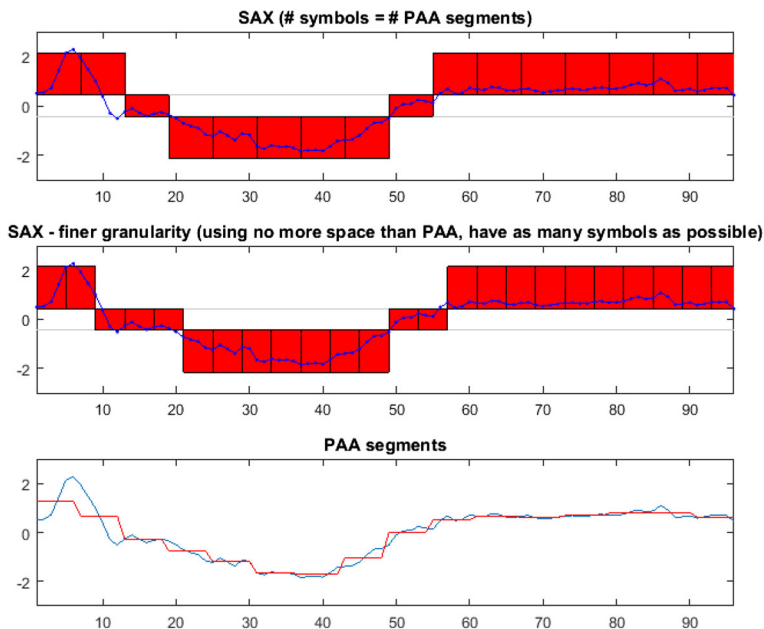


Fig. 6 Sample SAX representation for an alphabet size of four and desired dimensionality of 16. The time series length is 128. Horizontal grids differentiate the mean levels implied by each symbol and red boxes visualizes symbol assignments

of the discretization process on a sample time series from ECG200 dataset. These discretized time series are expressed by a sequence of categorical variables and forms categorical time series. SW-RF is then applied to the categorical time series for classification. Table 15 provides comparison of the classification accuracies of SW-RF against Euclidean distance, dynamic time warping (DTW), and derivative dynamic time warping (DDTW) followed by 1-nearest neighbor classifier. DTW is considered as one of the most well-known benchmarks in time series classification. DTW measures the distance between two time series with varying temporal patterns and shifts by calculating the optimal match using dynamic programming. DDTW is suggested as a variant of DTW which calculates the distance between two time series as a weighted combination of the DTW distance between the time series and the DTW distance between the first order differences. Although there is a loss of information due to discretization SW-RF yields a classification accuracy that is comparable to the state of the art approaches.

7 Conclusion and Future Work

This study proposes a novel data adaptive representation approach for categorical sequential data. The proposed SW-RF method offers a two-step processing of categorical sequences to obtain a meaningful representation. Applying tree-based

Table 15 Comparison of SW-RF with benchmark methods on 3 ECG datasets

	SW-RF	Euclidean 1NN	DTW_R1 1NN	DTW_Rn 1NN	DDTW_R1 1NN	DDTW_Rn 1NN
ECG200	0.830	0.880	0.770	0.880	0.830	<i>0.890</i>
ECGFiveDays	<i>0.810</i>	0.797	0.768	0.797	0.687	0.696
TwoLeadECG	0.923	0.747	<i>0.905</i>	0.868	0.995	0.982

Accuracy result for best performing approach is provided in italic

ensemble to a simple sliding window representation of the sequential data and summarizing sequence as a frequency distribution of the sequences at each terminal node provide some advantageous properties to SW-RF compared to state-of-the-art representation methods. It gives relatively better classification accuracy while number of observation symbols are increasing. Furthermore, the SW-RF representation can learn longer words based on the settings of maximum number of terminal nodes on decision trees. Our experiments show that proposed approach provides significantly better results in terms of accuracy on both synthetic data and DNA promoter sequence data. We have shown that important motifs related to gene expression can be identified when the representation is used for training lasso logistic regression. Success of SW-RF is also illustrated on alternative applications such as protein sequence and numerical time series classification. Although not considered in this study, proposed approach can be extended to regression problems in a straightforward manner.

Funding Information This research is supported by Air Force Office of Scientific Research Grant FA9550-17-1-0138.

Compliance with Ethical Standards

Conflict of Interest The authors declare that they have no conflict of interest.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Bacardit J, Stout M, Hirst JD, Valencia A, Smith RE, Krasnogor N (2009) Automated alphabet reduction for protein datasets. *BMC Bioinf* 10(1):6
2. Bagnall A, Lines J, Bostrom A, Large J, Keogh E (2017) The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Min Knowl Disc* 31(3):606–660
3. Baydogan MG, Runger G (2015) Learning a symbolic representation for multivariate time series classification. *Data Min Knowl Disc* 29(2):400–422
4. Beer MA, Tavazoie S (2004) Predicting gene expression from sequence. *Cell* 117(2):185–198
5. Benoit G, Peterlongo P, Mariadassou M, Drezen E, Schbath S, Lavenier D, Lemaître C (2016) Multiple comparative metagenomics using multiset k-mer counting. *Peer J Computer Science* 2:e94
6. Blasiak S, Rangwala H (2011) A hidden markov model variant for sequence classification. In: *IJCAI proceedings-international joint conference on artificial intelligence*, vol 22, p 1192
7. Breiman L (2001) Random forests. *Mach Learn* 45(1):5–32
8. Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees. CRC press

9. Brown PF, deSouza PV, Mercer RL, Pietra VJD, Lai JC (1992) Computational linguistics. arXiv:1608.03533 18(4):467–479
10. Hapgood JP, Riedemann J, Scherer SD (2001) Regulation of gene expression by gc-rich dna cis-elements. *Cell Biol Int* 25(1):17–31
11. Kuksa P, Pavlovic V (2009) Efficient alignment-free dna barcode analytics. *BMC Bioinforma* 10(14):S9
12. Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing SAX: a novel symbolic representation of time series. *Data Min Knowl Disc* 15:107–144
13. Ling CX, Huang J, Zhang H (2003) Auc: a better measure than accuracy in comparing learning algorithms. In: *Conference of the canadian society for computational studies of intelligence*, Springer, pp 329–341
14. MacNeil LT, Walhout AJ (2011) Gene regulatory networks and the role of robustness and stochasticity in the control of gene expression. *Genome Res* 21(5):645–657
15. Meher PK, Sahu TK, Rao A (2016) Identification of species based on dna barcode using k-mer feature vector and random forest classifier. *Gene* 592(2):316–324
16. Ounit R, Wanamaker S, Close TJ, Lonardi S (2015) Clark: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics* 16(1):236
17. Phillips KA, Trosman JR, Kelley RK, Pletcher MJ, Douglas MP, Weldon CB (2014) Genomic sequencing: assessing the health care system, policy, and big-data implications. *Health Aff* 33(7):1246–1253
18. Richter C, Luboschik M, Röhlmg M, Schumann H (2015) Sequencing of categorical time series. In: *2015 IEEE conference on visual analytics science and technology (VAST)*, IEEE, pp 213–214
19. Stormo GD, Schneider TD, Gold L, Ehrenfeucht A (1982) Use of the 'perceptron' algorithm to distinguish translational initiation sites in e. coli. *Nucleic Acids Res* 10(9):2997–3011
20. Ullrich A, Schlessinger J (1990) Signal transduction by receptors with tyrosine kinase activity. *Cell* 61(2):203–212
21. Vinogradov AE (2003) Dna helix: the importance of being gc-rich. *Nucleic Acids Res* 31(7):1838–1844
22. Weiss GM, Hirsh H (1998) Learning to predict rare events in categorical time-series data. In: *Proceedings of the AAAI/ICML workshop on time-series analysis*, Madison, Wisconsin
23. Wood DE, Salzberg SL (2014) Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol* 15(3):R46
24. Xing Z, Pei J, Keogh E (2010) A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter* 12(1):40–48
25. Zissman MA, Singer E (1994) Automatic language identification of telephone speech messages using phoneme recognition and n-gram modeling. In: *IEEE international conference on acoustics, speech and signal processing (ICASSP-94)*, vol 1, pp 305–308
26. Zou N (2015) A probabilistic framework of transfer learning: Theory and application. Arizona State University

Affiliations

Hande Cakin¹ · Berk Gorgulu¹ · Mustafa Gokce Baydogan¹  · Na Zou² · Jing Li³

✉ Mustafa Gokce Baydogan
mustafa.baydogan@boun.edu.tr

¹ Department of Industrial Engineering, Boğaziçi University, İstanbul, Turkey

² Texas A&M University, College Station, TX, USA

³ Arizona State University, Tempe, AZ, USA

nTreeClus: A model-based partitioning clustering of sequential data

Hadi Jahanshahi · Mustafa Gokce Baydogan.

Received: date / Accepted: date

Abstract The overwhelming presence of categorical/sequential data in diverse domains emphasizes the importance of sequence mining. The challenging nature of sequences proves the need for a continuing research to find a more accurate and faster approach providing a better understanding of their (dis)similarities. This paper proposes a new model-based approach for clustering sequence data, namely nTreeClus. Proposed approach is at the intersection of the conceptions behind existing approaches, including Decision Tree Learning, k -mers, and autoregressive models for categorical time series, culminating with a novel numerical representation of the categorical sequences. This new representation is used to perform clustering considering the inherent patterns in categorical time series. Furthermore, the only parameter of the method, the window size (n), is examined and the robustness of the method to its parameter has been shown. Under different simulated scenarios and using various internal and external cluster validation indices, the performance of the method is analyzed and reported. Finally, empirical evaluation using synthetic and real datasets, protein sequences and categorical time series, declares that nTreeClus is competitive or superior to most of the state-of-the-art methods.

Keywords Sequence Mining · categorical time series · model-based clustering

Mustafa Gokce Baydogan
Department of Industrial Engineering: Boazii University, 34342, Bebek, Istanbul, Turkey
E-mail: mustafa.baydogan@boun.edu.tr

Hadi Jahanshahi
E-mail: Hadi.Jahanshahi@boun.edu.tr

1 Introduction

Sequences are among the most common types of data found in diverse domains such as bioinformatics, marketing, social science, text mining, web mining, security, health-care, or business. For example, short repeated DNA sequences are one of the most interesting features of prokaryotic and eukaryotic genomes [1] since they encode the structure and mechanism associated with neurological disorders. Learning the structure of these repeated patterns is also applicable to the studies of genetic variation, gene tagging and linkage mapping [2]. Moreover, sequences can also capture some behavioral characteristics of human through their purchase history in a shop or their click-stream sequences on a website [3–5]. Analyzing the past transaction logs or historical data of an organization plays a crucial role for decision makers [6]. Assessing process flows in a hospital (as a sequence of past events) [7] or evaluating the past transaction data of the orders a company received [8] are some other typical examples of the ubiquity of sequences to be mined.

Numerous studies have been carried out in the field of sequence data mining, although clustering categorical sequences gained the least attention in the literature noting that clustering sequences is a major challenge to be addressed in the aforementioned domains. The task demands either accurately computing (dis)similarity between sequences or finding the underlying model generating sequences. Therefore, from the problem-solving approach, we categorize them into two folds: *Proximity-Based approaches*, including *Similarity-Based* and *Feature-Based*; and *Model-Based approaches* [9]. The main aim of Proximity-Based methods is to devise a similarity or distance measure between sequences, applied either on raw-data in *Similarity-Based* or on feature-extracted data in *Feature-Based*. On the other hand, in *Model-Based* methods, a new representation of the given data has been introduced through a model which best fits the data. Hence, to cluster categorical sequences, we need to define a (dis)similarity measure or latent model appropriately depicting the behavior of sequences.

There has been a significant change in the approaches towards solving these types of problems during past decade. Previously, Similarity-Based approaches, especially alignment-based ones, was the most popular method trying to align sequences and find the best or longest common pattern which exists in a sequence. They can be categorized as global, local, or glocal alignments based on dynamic programming [10–14], pairwise alignment which can be used for only two sequences at a time [15–17], multi-sequence alignment [18–23], and structural alignment which is applicable to the sequences whose structures are known [24, 25]. Accordingly, most of these methods are used to handle bioinformatics sequences and in most cases cannot be generalized to other domains. Additionally, most of the more accurate ones have many limiting factors; namely, the number of sequences to be handled, computation time and memory, the position of the pattern in a sequence, and the length of sequences which should not be varying significantly.

Similarity-Based approaches considering distance among sequences are apt to disclose the (dis)similarity between given sequences. Most of the methods in this category use cost of dissimilarity to count mismatches; nevertheless, the definition of mismatch in each method differs. Hamming distance, applicable to the sequence with the same length, counts the number of elements to be substituted in order to change one string to another. The Levenshtein distance, or edit distance, is the number of insertion, deletion, and replacement which is required to turn one string into another [26]. The drawback of this distance method is widely discussed; it is not a good choice as far as computation complexity is concerned, the median string problem is not tractable in edit distance method [27], and finally, it fails in some important cases such as automated language classification tasks [28].

Feature-Based approaches perform information abstraction by capturing merely important information of the data. k -mers (n -grams), as the most common method in this category, are more omnipresent than alignment methods, including their usage in word prediction (speech recognition) [29,30], text categorization and classification [31], detection of new malicious code [32], similarity and distance of strings [33], authorship identification [34], protein and genome sequences [35], file and data type classification [36], and information retrieval [37]. Nevertheless, there are some arguments about its performance such as finding the optimal k for k -mer [7]. Position Weight Matrix (PWM) is a method to find a pattern, especially a motif, in a sequence [38]. The method uses a matrix representation of the sequence based on the appearance of each element in a specific window. In order to have a better understanding of the sequence, this method needs pre-alignment for the sequences with different length to find the optimal window. Hence, again the aforementioned problem of finding the best window arises.

The last one is the Model-Based approach in which a model is trained on the sequence to find the logic of how the letters are ordered. Markovian models are more general methods which are founded on the assumption that sequences have parametric distribution. The method is heavily dependent on the conditional probability of occurrence of each element in the sequence while satisfying the so-called “Markov Property”. There are many types of Markovian Models, the most famous ones of which include first-order Markov Chain, Higher order Markov Chain, Interpolated Markov Motif Models (IMM), Multivariate Markov Chain Model, and Hidden Markov Model (HMM) and its Mixture version (MHMM). It is deserved to be mentioned that Model-Based methods, such as MHMM, simultaneously cluster sequences while finding the best-fitted model. Unlike PWM that has no memory, Markov Chain Models consider previous elements of a sequence by introducing states [3]. Despite being utilized in diverse domains [39–42], Hidden Markov Models are hard to be tuned and in most cases are computationally intensive, thus, affecting its generality and scalability [7]. Autoregressive models for categorical sequences (AR models) have been introduced in the past decade. These methods investigate the correlative structure of sequences. Discrete Autoregressive Model (DAR) [43] and One-channel Autoregressive Model (AR) [44] were presented

to cluster DNA sequences. Further researches on Autoregressive Models for sequential data are mostly applicable to bioinformatics and lack generality. They also require setting model order and window length. Finding the optimal parameters takes time and may lead to higher complexity of the model [45]. AR models also require numerical mapping while categorical data might not have a logical order and mapping them may lead to an irrational deduction.

Here we provide a new Model-Based clustering method, which is called nTreeClus, to capture the sequence features while ignoring the aforementioned restrictions. Moreover, the method offers a novel representation of categorical sequential data which can be used as a feed for (dis)similarity measures to cluster the sequential datasets. nTreeClus combines the notions behind k -mers, autoregressive (AR) models, decision trees (DT), and similarity measures to generate a new comprehensive method to enrich the understanding of the sequences. nTreeClus tries to segment a given sequence to shorter sequences with the length of n while tracking down the relation among all elements of the sequence using autoregressive DT. The new numeric representation is the upshot of counting the number of repetition of each segmented subsequence in terminal nodes (leaf nodes) of Decision Trees. Based on the nTreeClus representation of the data, available (dis)similarity methods, such as Cosine and Manhattan, can be used. The trials reveal the robustness and accuracy of the method.

The remainder of the paper is organized as follows. Section 2 summarizes a brief notion of the method and its competitive approaches. In section 3, the preliminaries are introduced and nTreeClus as a method to cluster sequential data is presented, including the novel approach for representing categorical data and similarity measurements. Furthermore, computational complexity of the method briefly has been discussed. In section 4, the sensitivity of the method to the parameter n and the ability of the method to predict the number of clusters have been discussed. Moreover, results on the performance of the approach for different scenarios have been reported and the clustering ability of nTreeClus on the real data has been examined. Section 5 provides conclusions.

2 Background

nTreeClus is a synthesis of the ideas from Decision Tree learners, k -mers, and autoregressive models for categorical time series. In this section, a brief elucidation of the concept behind these methods and models, in addition to their virtues and shortcomings, has been discussed.

Decision Tree learners are interpretable models with satisfactory accuracy used in a vast variety of domains. Univariate Decision Trees, such as CART [46] and C4.5 [47], regard only one attribute (feature) at each decision node leading to axis-aligned splits. Unlike the greedy nature of univariate trees, tree ensembles, such as Random Forest (RF), are less likely to overfit on data since they are simply a collection of DTs whose results are less biased and less likely to overfit due to the random subset of features they pick. RF classifier

is an ensemble of k decision trees, $h(x, \Theta_k)$, where the Θ_k is a random vector which is independent of past vectors but with the same distribution, each casting a unit vote for the most popular class at input x [48]. For choosing the number of features to be used for tree k , often $\sqrt{\nu}$ is considered where ν is the number of total available features. The random selection of features leads to the reduction of variance of the classifier and decreases computation complexity of DT from $O(\nu \mathcal{L} \log \mathcal{L})$ to $O(\sqrt{\nu} \mathcal{L} \log \mathcal{L})$ where $\mathcal{L}$ is the number of training instances (in-bag). Random Forests are simple to interpret and implement, robust to outliers, strong in handling non-linearity, fast in real cases, and friendly in parallel training to save more time, although there are ways to improve its suboptimal issues and greed nature of each subtree [49, 50].

Autoregressive Models (AR models) are widely employed in data mining domains especially time series, indicating the output variable depends on its own previous values and because of its stochastic nature, it takes advantage of the statistics of the data [51]. Not until recently have data scientists been engaged in applying AR models to categorical sequences. AR spectral analysis tools capture certain features of DNA sequences of coding and noncoding regions [44]. It claims that considering the type of the numerical mapping rule which is used, the Autoregressive Feature-Based searching techniques have a high accuracy in identifying the patterns and their locations in DNA sequences. Thereafter, many scholars verify that AR models for categorical time series perform well in analyzing the structure of sequences and finding patterns in them [45, 52–56]. However, AR models are new and mostly implemented in bioinformatics, thus lacking generality. They also have two parameters, model order and window length, to be set, leading a higher complexity of them. Finally, they use numerical mapping of categorical data which leads to an irrational deduction.

2.1 Related Work

In this section, we briefly introduce the more common methods from each of three available approaches, including Jaro-Winkler and Levenshtein from Similarity-Based approaches, k -mers from Feature-Based approaches, and finally Mixture of Hidden Markov Models from Model-Based approaches. These methods are considered as alternatives for pattern recognition in sequences.

Jaro metric [57], a metric widely used in the record-linkage community, has a variation due to Winkler [58], counting the common characters between strings even if their positions are slightly different. This common edit-distance method places more emphasis on the strings that match from the beginning (common prefix); therefore, it is unstable from one set to another [59]. Levenshtein distance (LD) [60], edit distance, is a promising measurement to compare two or more strings by various operations, including the insertion, deletion, and reversal (substitution) of symbols. It is defined as the minimum cost to transform one string into another through the aforementioned modifications. The more edit operations are needed for transformation, the more

distant two strings are. LD's computation complexity increases as the size of two strings is long.

n -gram method, also called k -mer in bioinformatics, is a Feature-Based method which is used as another competitive approach since the logic behind window size (n in n -gram and k in k -mer) is closely comparable to n in nTreeClus. Mixture Hidden Markov models (MHMM), also called mixed Markov latent class models [61] or mixture latent Markov Model (MLMM) [62], contains five types of variables: response variables, time-constant explanatory variables, time-varying explanatory variables, time-constant discrete latent variables, and time-varying discrete latent variables. Here, for the simplicity of the model, we only consider time-constant covariates. Having a set of Hidden Markov Models $\mathcal{M} = \{\mathcal{M}^1, \dots, \mathcal{M}^K\}$, where $\mathcal{M}^i = \{\pi^i, A^i, B_1^i, \dots, B_C^i\}$ for submodels $i = 1, \dots, I$. For each subject Y_i , $P(\mathcal{M}^i) = \omega_i$ is the prior probability that it follows the submodel $\mathcal{M}^i$. The log-likelihood of the parameters of the Mixture Hidden Markov Model can be formulated as [63]

$$\begin{aligned} \log L &= \sum_{i=1}^N \log P(Y_i | \mathcal{M}) \\ &= \sum_{i=1}^N \log \left[\sum_{k=1}^N P(\mathcal{M}^k) \sum_{all z} P(Y_i | z, \mathcal{M}^k) P(z | \mathcal{M}^k) \right] \end{aligned} \quad (1)$$

This method assumes that there exists a latent state resulting in the observed sequence. In all the experiments, we select the number of latent states (m) based on Bayesian information criterion (BIC), $BIC = -2l(\theta) + p \log m$ for m in $[1, 5]$. Minimizing the BIC which corresponds to maximizing the posterior model probability is desirable [64].

3 nTreeClus: A new methodology for clustering categorical sequential data

The categorical sequential dataset, $X_{\mathcal{L}}^{\mathcal{M}}$, is comprised of $\mathcal{L}$ sequences of length $\mathcal{M}$ over an alphabet set of $\mathcal{A}$ where x_l^m is the m th element of sequence l . For the purposes of simplification, the sequences are assumed to be of the same length, $\mathcal{M}$, while the approach can handle sequences of different length. When the length of sequences is different, $\mathcal{M}$ is equal to $\max\{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_l\}$ and the shorter sequences will be treated as sequences with some missing values. In other words, nTreeClus can handle missing values no matter how they are distributed in the dataset. Therefore, x_i^j can either be selected of the alphabet set $\mathcal{A} = \{s_1, s_2, \dots, s_a\}$ or remain empty in the case that $\mathcal{M}_i < j$. X_i , the i th sequence of the dataset, can be represented by

$$X_i = x_i^1 x_i^2 \dots x_i^{m-1} x_i^m \quad (2)$$

where it forms the i th row of the categorical dataset, $X_{\mathcal{L}}^{\mathcal{M}}$; thus, $X_{\mathcal{L}}^{\mathcal{M}}$ is illustrated by an $\mathcal{L} \times \mathcal{M}$ matrix as

$$X_{\mathcal{L}}^{\mathcal{M}} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^{m-1} & x_1^m \\ x_2^1 & x_2^2 & \cdots & x_2^{m-1} & x_2^m \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ x_{l-1}^1 & x_{l-1}^2 & \cdots & x_{l-1}^{m-1} & x_{l-1}^m \\ x_l^1 & x_l^2 & \cdots & x_l^{m-1} & x_l^m \end{bmatrix} \quad (3)$$

3.1 Representation Learning

nTreeClus is founded based on two existing approaches, from both supervised and unsupervised learning. The primary aim of the algorithm is to find specific patterns in Categorical Sequence Data. The premise on which it is founded is extensible to any problem in which a pattern should be determined. In order to find a pattern in data, nTreeClus utilizes basic principles in available and commonly used approaches. In sequences, in which the definite number of alphabets is used to specify a distinctive characteristic of the instances, a pattern recognizer can be used to cluster them based on including or excluding particular pattern.

3.1.1 Phase I: segmented matrix representation

In the first step, nTreeClus tries to segment each sequence using a predefined length (n) in order to have a length-smoothed data instead of having sequences of different length. This step provides the algorithm an environment in which employing single Decision Tree or Decision Tree Ensembles is attainable. It also has the same logic as k -mers with the window size of n while trying to extract all possible substrings of length n that are contained in a given string. In the segmentation part, as shown in Figure 1, based on the predefined length n we divide each string into some substrings which have the length of $n + 1$. The last column is used as a class label while the first n elements play the role of features which result in the class label (in an autoregressive manner). In the given sequence in Figure 1, n is equal to 5. “a b c a a” are the five elements leading to the output “a” which is the next alphabet of the string. Based on both the length of each string and the predetermined “ n ”, many or few new substrings are expected. In Algorithm 1, the loops to create the segmented matrix out of a given sequential Dataset have been demonstrated.

The segmentation part will be done to all rows of matrix $X_{\mathcal{L}}^{\mathcal{M}}$ and the segmented matrix will be generated. Figure 1 demonstrates the segmentation algorithm only for X_1 , the first row of the supposed $X_{\mathcal{L}}^{\mathcal{M}}$. Assuming that all rows of $X_{\mathcal{L}}^{\mathcal{M}}$ have the same length $\mathcal{M} = 11$ and the total number of sequences is $\mathcal{L} = 100$, then for the window size of $n = 5$, we will have the segmented matrix (S) with 6 columns and $6 \times 100 = 600$ rows. Generally speaking, the segmented output is represented by a $((\mathcal{M} - n)\mathcal{L}) \times (n + 1)$ matrix.

	1*	a	b	c	a	a	a
	2*	b	c	a	a	a	b
X_1 : abcaaabcbaa	3*	c	a	a	a	b	c
	4*	a	a	a	b	c	b
	5*	a	a	b	c	b	a
	6*	a	b	c	b	a	a

Fig. 1 A sample of segmented sequence (the right column indicates labels while the first 6 columns are features. Since both labels and features come from the same data, it mimics autoregressive behavior. The left column is location of each substring. This column is optional and can be added whenever the location of substring is of importance.)

Algorithm 1 Matrix Segmentation

Data: sequential/categorical dataset (D) of size $\mathcal{L} \times \mathcal{M}$, the segmentation length (n) (with default value of $\sqrt{\mathcal{M}}$).

Result: Segmented matrix (length-smoothed sequences)

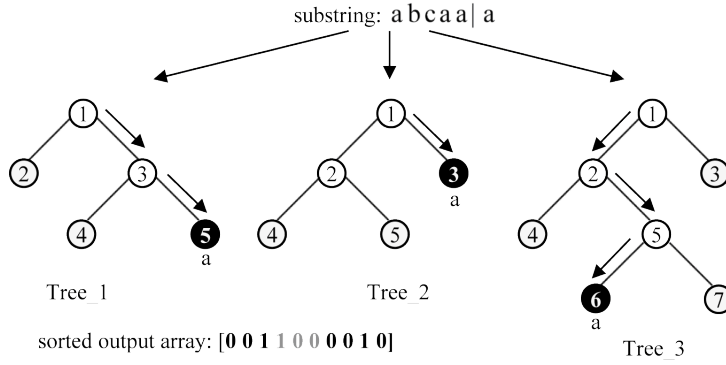
```

1 initialization;
  Segmented Matrix =  $\emptyset$ 
2 for  $i \in \{1, \dots, \mathcal{L}\}$  do
3   for  $j \in \{1, \dots, (\mathcal{M}_i - n)\}$  do
4     Temporary Row  $\leftarrow$  Dataset [ $i, j:j+n$ ]
5     Temporary Row ['y']  $\leftarrow i$ 
      if the position of element is important then
        Temporary Row ['position']  $\leftarrow j$ 
      end if
    Append [Temporary Row] to Segmented Matrix
6   end
7 end

```

3.1.2 Phase II: providing the novel nTreeClus representation

The segmented matrix is an input for the next phase of the algorithm in which the existing classification method, Decision Tree (DT), will be used. The CART Decision Tree, utilized by nTreeClus, is a binary partitioning process with the ability of handling both continuous and discrete attributes while the splitting part is supervised by Gini index so that the reduction in tree impurity becomes maximized [65]. nTreeClus utilizes either Decision Tree or Tree Ensembles to have the least error at the end. Nonetheless, in this stage, the set of terminal nodes (leaf nodes) to which a specific substring belongs generates the most valuable input for Clustering. In general, after implementing Tree Ensembles, each substring goes to a specific terminal node of each generated tree while nTreeClus traces the exact position of the substring that is to say, each rule of DT describes a pattern in the sequence. Assuming that the total number of trees in Decision Tree Ensembles is 3, the first substring in Figure 1, “abcaa”, is assigned to terminal nodes 5, 3, and 6 of tree 1, 2, and 3 respectively (Figure 2). The same case for the rest of substrings is held.

**Fig. 2** Introducing an instance to Decision Tree Ensembles

What nTreeClus is eager to record is a large, sparse matrix of terminal nodes to which each instance belongs. Therefore, the number of its columns is equal to the number of unique terminal nodes, here 10, and the number of its rows is equal to the number of strings (instances) given by the original matrix. Assume for the given sample in Figure 1, six substrings end in the terminal nodes as shown in Table 1.

Table 1: Terminal nodes for a sample instance.

substring's id	tree 1	tree 2	tree 3
1 (abcaa)	5	3	6
2 (bcaaa)	2	5	6
3 (caaab)	5	5	6
4 (aaabc)	5	5	7
5 (aabcb)	2	3	7
6 (abcba)	4	3	6

The Table 1 can easily be shown as a binary matrix (η) whose columns are terminal nodes and whose rows are substrings. The equivalent matrix for the Table 1 will be as

$$\eta_{(abcaabcbba)} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad (4)$$

Table 1 holds the number of repetition for each instance which is defined by its substrings. It can be represented as Table 2 whose column names indicate the trees' ids and their terminal nodes; for instance, "2-4" specifies terminal node 4 of tree 2. The numerical values inside the Table 2 demonstrate how

many times an instance is ended in a specific tree of Decision Tree Ensembles. “0” under column “2-4” declares none of the substrings of instance id 1 is ended in terminal node 4 of tree number 2 and so forth. (Table 2 has the number of rows equal to the total number of instances, one of which is illustrated herein.)

Table 2: Decision Tree Ensembles’ output matrix

instance id	1-2	1-4	1-5	2-3	2-4	2-5	3-3	3-4	3-6	3-7
1	2	1	3	3	0	3	0	0	4	2

that is to say the matrix 4 can be aggregated and generate the equivalent matrix form of Table 2 as

$$\eta_{aggregated} = (2\ 1\ 3\ 3\ 0\ 3\ 0\ 0\ 4\ 2) \quad (5)$$

Algorithm 2 nTreeClus Representation

Data: Segmented Matrix of size $((\mathcal{L} \times (\mathcal{M} - n)) \times (n + 2))$, number of trees (t) with default value of 10.

Result: nTreeClus representation of dataset D (nTreeClus Representation)

```

1 initialization;
  xtrain  $\leftarrow$  Segmented Matrix  $[:,1:n]$ 
  ytrain  $\leftarrow$  Segmented Matrix  $[:,n+1]$ 
2 Train RandomForest (X=xtrain, Y=ytrain, ntree =  $t$ )
3 terminalRF  $\leftarrow$  trace the path of each row in Segmented Matrix to the terminal node for all  $t$ 
  trees and store it.
4 nTreeClus Representation = An empty DataFrame whose number of columns is equal to
  terminalRF's one and whose number of rows is equal to Dataset's one.
5 for  $i \in \{1, \dots, \mathcal{L}\}$  do
6   | nTreeClus Representation  $[i, :] = \sum_{j=1}^{(\mathcal{L} \times (\mathcal{M} - n))} \text{terminalRF}[j, :] \text{ iff Segmented Matrix } [j, 'y'] == i$ 
7 end

```

3.2 Similarity Measure

Using Algorithm 2, we can find a new representation for each sequence. Here, for the categorical sequence (abcaaabcbaa), we propose the numeric representation (2133030042) which can record the logic behind this set of alphabets using autoregressive DT ensembles. This numeric representation could be a proper feed to find the distance between each pair of strings. There are numerous ways to find (dis)similarity between these sequences, from which we have implemented Cosine and Manhattan distance to find out which one is the optimal method to be used. Having the distance between each sequence, we employ Hierarchical grouping using Ward’s method to cluster data [66]. Ward’s minimum variance method, outperform other linkage method except when the outliers exist. [?]

Given two sets of numeric vectors of the same length (l), here nTreeClus representations, Cosine dissimilarity (θ) can be shown as

$$\cos(\theta) = 1 - \frac{a.b}{\|a\|_2\|b\|_2} = 1 - \frac{\sum_{i=1}^l a_i \times b_i}{\sqrt{\sum_{i=1}^l (a_i)^2} \times \sqrt{\sum_{i=1}^l (b_i)^2}} \quad (6)$$

Where $\|*\|_2$ indicates 2-norm of its argument $*$, and $a.b$ is the dot product of a and b . Cosine similarity is within the range of 0 and +1 while +1 indicates close similarity and 0 declares strong dissimilarity. On the other hand, Manhattan distance places more emphasis on differences between elements of two different vectors. Manhattan distance, or the Minkowski distance with 1-norm, can be shown as

$$D_{man}(a, b) = \|a - b\|_p = \left(\sum_{i=1}^l (|a_i - b_i|)^p \right)^{\frac{1}{p}}; p = 1 \quad (7)$$

The complexity of nTreeClus is comprised of the complexity of two phases, which is $\mathcal{L} \times (\mathcal{M} - n)$ for segmentation part and the complexity of Random Forest $O(\sqrt{(\mathcal{M} - n)\mathcal{L}} \log \mathcal{L})$; therefore, it is highly related to the length and the number of sequences. Since Decision Tree ensembles are fast in real cases, the method have a satisfactory running time. However, the computational complexity of k -mer is $O(\mathcal{L}^2\mathcal{M})$ and quadratically increases if the number of sequences increases [?, 67]. The same case exists for Jaro-Winkler where the time complexity is quadratic $O(\mathcal{M}_p + \mathcal{M}^2)$ [68]. Furthermore, in Levenshtein distance, the time complexity is $O(\mathcal{M}^2)$ [?] and similarly it has a quadratic form. Mixture of Hidden Markov Models have the quadratic relation to the number of hidden states while both the length and the number of sequences have direct impact on its complexity. The

4 Experiments and Examinations

In order to have a fair comparison between nTreeClus and other methods, different scenarios imitating the properties of categorical/nominal sequential data with varying levels of intricacy have been generated. The algorithm has been applied to the generated data with different amounts of n (window size). A comparison with alternative approaches which are defined as *Similarity-Based*, *Feature-Based*, and *Model-Based Approach* has been conducted to make deductions about the competitiveness of nTreeClus. Furthermore, Python implementation of nTreeClus is provided at [69] promoting a full reproducibility of the method for further researches.

4.1 Performance metric

Clustering as an unsupervised learning approach strives to determine the innate pattern in a set of unlabeled data while considering intra-cluster quality and inter-cluster separation [70]. Two types of cluster validity, namely internal and external criteria [71], have been introduced. External validation is the one which is based on pre-specified structure or previous knowledge of data (knowing the true labels); whereas, Internal Validation evaluates the clustering result using features inherited in the dataset when no external criteria are available. In most real cases, using External Validation criteria is not possible; nevertheless, when synthetic data is generated with the known objective functions, associating cluster identifiers with the generated dataset is viable [72].

In this paper, External Validation indices are used whenever we are conversant with the structure of data and Internal Validation indices are implemented whenever we desire to determine the correct number of clusters.

4.1.1 Internal cluster validation indices

Calinski-Harabasz Index (CH) is referred to as two top performers for determining the number of clusters [73], representing a global criterion. Calinski and Harabasz [74] suggest “the best sum of squares split” of the dendrite is where not only is the within-group (cluster) sum of squares (WGSS) minimum, but also the between-group sum of square (BGSS) becomes maximum, which corresponds to the maximum value of $CH(k)$, which is defined as

$$CH(k) = \frac{BGSS}{k-1} \bigg/ \frac{WGSS}{n-k} \quad (8)$$

where n is the number of clustered samples and k is the number of clusters. The average silhouette width (ASW) [75] has been computed whenever there is no need to establish a comparison with the true parameter. Given a partition ξ_S , the ASW is given by

$$ASW = \frac{1}{N} \sum_{i=1}^N (s(i)) \quad (9)$$

where $s(i)$, silhouette width, is obtained by

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (10)$$

with $a(i)$ being the average dissimilarity of object i to all other objects of A, the cluster to which it has been assigned; and if $d(i, C)$ be defined as the average dissimilarity of object i to all objects of C, any cluster which is different from A, then

$$b(i) = \underset{C \neq A}{\text{minimum}} d(i, C) \quad (11)$$

$s(i)$ takes values in $[-1, +1]$; when $s(i)$ is close to the value $+1$, object i is nearer to its own cluster than the other closest cluster and therefore is taken as “well-classified”. On the other hand, when it is close to -1 , the opposite relationship applies and i is taken as “misclassified”. Finally, being about zero means it is unclear whether it should be assigned to its own cluster or the neighboring cluster.

As the last internal validity index, the Dunn Index [76] measures the ratio between the smallest intercluster and the greatest intra-cluster distance. Let $\delta(C_q, C_r)$ be the intercluster distance metric between clusters C_q and C_r , then for the number of clusters k , the Dunn Index is defined as:

$$DI_k = \min_{1 \leq q \leq k} \left\{ \min_{\substack{1 \leq r \leq k \\ r \neq q}} \delta(C_q, C_r) \right\} / \max_{1 \leq p \leq k} \Delta(C_p) \quad (12)$$

where $\Delta(C_p)$ is the diameter of cluster C_p or the maximum intra-cluster distance. The larger the value of DI_k is, the more exact the clustering solution will be.

4.1.2 External cluster validation indices

When a priori information of the dataset is available, external cluster validation indices can be used. A recent study [77], considering most of the papers in the field of categorical time series, suggests that Clustering Accuracy/Purity, Rand Index (RI)/Adjusted Rand Index (ARI), and F-measure are the most common External Validation measures. We incorporate those methods in the simulation part wherever the true labels are known.

Given a set of n elements $S = \{X_1, \dots, X_n\}$, and two clusterings of them, RI is the proportion of pairs of the two clusters which either belong to the same cluster or different clusters. RI [78] ranges from 0 when two clusterings have no similarity to 1 when the two clustering are identical. The Adjusted Rand Index (ARI) [79] is an index-corrected-for-chance version of RI which is bounded above by 1 meaning perfect agreement and takes on the value 0 when partitions are selected randomly. Cluster Purity, as one of the frequently used external measures [80], determines to which extent clusters contain a single class. Given $P_j = \frac{1}{n_j} \max_i (n_j^i)$ as the purity of cluster j with class label i , the overall Purity is defined as

$$Purity = \sum_{j=1}^k \frac{n_j}{n} P_j \quad (13)$$

which is as a weighted sum of the individual cluster’s purities. Here, k is the number of clusters, n_j the size of cluster j , and n the total number of objects. High values of the Purity is desirable. Finally, F-measure which combines the precision and recall concepts can be used as an external cluster criterion which takes the values within $[0,1]$, for which the higher value indicates a better

clustering performance. F-measure is a useful method to evaluate clustering structure [81].

4.1.3 Comparing with Nearest Neighbor classifier

The K-NN classifier is based on the premise that classification of unknown instances can be done by associating the unknown object to the known using the given dis(similarity) function. The idea is to apply one nearest neighbor wherever the ground truth of a data set is available to measure the efficacy of the newly introduced method using leaving-one-out [82–84]. Therefore, We employed 1-NN to assess the effectiveness of nTreeClus and compare it with the other available methods.

4.2 Patterns to be identified

Sequential pattern discovery has been available in a vast variety of domains. Such an omnipresence leads to many suggested approaches processing data in a hope of finding a better understanding of the nature of different data types. In a given sequence, manifold forms of patterns may be available, some of which are illustrated in Figure 3. In Figure 3a, the first important type of the patterns, a shifted pattern, has depicted; Figure 3b is a modification of the previous type while a gap is seen between the pattern; Figure 3c is a closed lopped version of the first type of the pattern; finally Figure 3d represent a rare type of the sequence in which the same structure and order can be seen while different alphabet has been used. Because of scarcity of the last case, in the following trials, the first three patterns have been examined and a clear contrast between all the available methods has been drawn.

....AABCD.....	(a)	AACABBBB	(c)
.....AABCD...		ACABBBBA	
.AA...BCD.....	(b)	...BBABAB...	(d)
....AA...BCD..		...CCDCDC...	

Fig. 3 (a): repeated pattern in different positions; (b): repeated gapped pattern; (c): shifted pattern in a closed loop; (d): same pattern with different alphabets.

4.3 Tuning parameter n

The parameter n in nTreeClus is similar to the parameter k in k -mers. Using a large n has the advantage of covering a general relation among the elements; by contrast, the small n is superior wherever the pattern is short and the volume of noise in data is high. Apparently, if the size of the pattern is known and equal to τ , the window size (n) should be set as τ to capture whole substrings with the equal length; nonetheless, in most real cases, the size of the pattern

is unknown. In order to analyze the sensitivity of the model to the parameter n , a contrast has been drawn between nTreeClus and k -mers (n -grams).

A simulation including $N = 40$ batches of sequences has been conducted. Each batch is formed from $\mathcal{L} = 180$ sequences with different characteristics. They may take $a = \{7, 20\}$ different categorical values in $\mathcal{A} = \{A, B, C, \dots, S, T\}$ with the length of $\mathcal{M} = 40$. Eventually, a fixed random pattern with the length of $\mathcal{L}_P = \{8, 15\}$ chosen from the same set $\mathcal{A}$ has been inserted in a random position of half of the sequences. In order to eliminate any biased randomness, the simulation has been replicated 10 times.

Figure 4 clarifies the result of the experiment on 4 different datasets with the aforementioned properties. In Figure 4 (a) and (b), the length of the pattern ($\mathcal{M}_P$) is 7 and only the number of alphabets which is selected from set $\mathcal{A}$ is different. As the number of alphabets rises (going from (a) to (b)), so does the Average Silhouette Width (ASW) in that the noise in the sequences increases and recognizing the pattern turns less challenging. Skipping the shift in ASW, the general form of both graphs are similar; as long as the n in nTreeClus is less or equal to the length of pattern (gray area), the ASW for nTreeClus is stable and afterward, it converges to a less but acceptable ASW. k -mers, by contrast, experiences no stable region, no convergence, and even no specific correlation with the pattern length. In Figure 4 (c) and (d), the length of the pattern ($\mathcal{M}_P$) is larger and, therefore, it is easier to be recognized. Again nTreeClus possesses the safe gray region of accuracy (for $3 \leq n \leq \mathcal{L}_P$), becomes stable, and then asymptotically approaches a reasonable value for ASW. Conversely, k -mer lacks the safe region and is not applicable to the generated dataset for the large values of k .

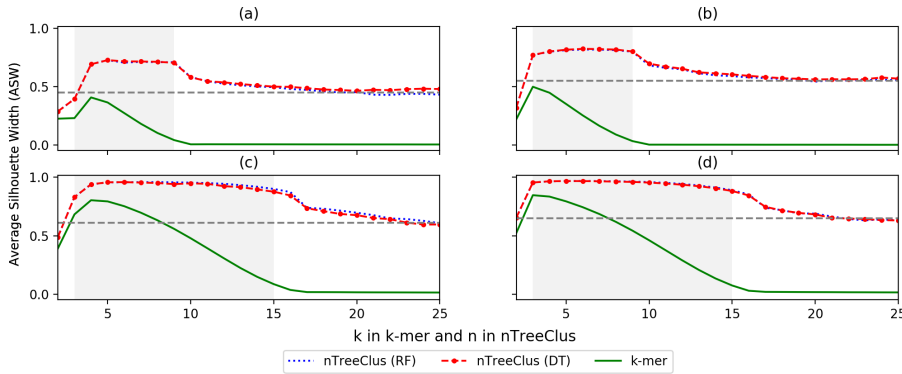


Fig. 4 The output of 10 replications for different characteristics of data: (a): $a = 7$ & $\mathcal{L}_P = 8$; (b): $a = 20$ & $\mathcal{L}_P = 8$; (c): $a = 7$ & $\mathcal{L}_P = 15$; (d): $a = 20$ & $\mathcal{L}_P = 15$; three methods have been compared based on ASW for different values of n .

Figure 4 also illustrates that for the randomly simulated datasets, nTreeClus consistently outperforms its rival k -mers. In this simulation, using cosine dis-

similarity, nTreeClus is robust to the classification methods since both Random Forest (RF) and Decision Tree (DT) lead to almost identical consequences.

As the simulation puts forward, if the ground truth is available, the best n is the value close to $\mathcal{M}_P$. For the cases where no knowledge about data is at hand, the second simulation has been carried out and a default value for parameter n is examined. Three different values for n are inspected, namely square root, one-half, and one-tenth of the length of sequences, $\mathcal{M}$.

Second simulation includes 90 batches of sequences with the structure of $\mathcal{L} = \{180, 450, 720\}$, $a = 7$ different categorical values in set of $\mathcal{A} = \{A, B, C, \dots, G\}$, $\mathcal{L}_P = \{7, 15\}$, and finally $C = \{2, 5, 8\}$ number of clusters. Unlike the first simulation, the number of clusters varies in order to cover different circumstances that may occur in real cases. After five replications, 40500 sequences with the specified characteristics have been simulated. Figure 5 shows the average ranks for all different settings of n . The Friedman test shows a significant difference between the 3 settings at the both 0.05 and 0.1 level exists. Proceeding with the Nemenyi test, Critical Difference (CD) is computed, showing the performance of $n = \sqrt{\mathcal{M}}$ is significantly better than $n = 0.1\mathcal{M}$ and $n = 0.5\mathcal{M}$ at different α levels. Therefore, in the following experiments, n always has been set to the square root of $\mathcal{M}$.

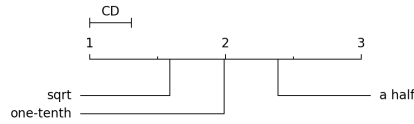


Fig. 5 The average rank for three different setting of n . The critical difference at 0.05 and 0.1 levels are 0.349 and 0.306, respectively. The performance of $n = \sqrt{\mathcal{M}}$ is significantly better at different α levels.

4.4 Simulations

As it is described in section 4.2, in this part, we try to mimic different scenarios which may occur as a pattern in a sequence data. The simulated datasets are either with the normal shifted pattern inside or with a more complicated shifted-gapped pattern. The length of the sequences and the number of alphabets differ case by case. The more detailed description of the scenarios is presented in the following sections.

4.4.1 Simulation 1 - pattern recognition

The first simulation is comprised of $N=1,080$ batches of sequences, each of which is formed from $\mathcal{L} = \{40, 120, 200\}$ sequences with different characteristics. They may take $a = \{4, 6, 10, 20\}$ different categorical values in $\mathcal{A} = \{A, B, C, \dots, S, T\}$. The length of the sequences ($\mathcal{M}$) oscillates between 20 to 90 while it is fixed for each batch. Eventually, a random pattern with the length

of $\mathcal{M}_P = \{6, 10, 15\}$ has been inserted in each half of the sequences of each batch, thus generating two clusters ($C = 2$). In order to eliminate any biased randomness, the simulation has been replicated 10 times. The ultimate output of these replications consists of $\mathcal{N}_S=129,600$ sequences.

Table 3 demonstrates the results of the six methods for the first scenario. It indicates nTreeClus, whether uses Decision Tree or Random Forest, leads to substantially higher indices than those of the rest of algorithms. The closest approach to nTreeClus is k -mers for $k = \sqrt{\mathcal{M}_i}$ which has the Average Silhouette Width (Internal Validation) only half as much as that of nTreeClus. Another somewhat surprising result is that the accuracy of nTreeClus is not affected by choosing either Decision Tree or Decision Tree Ensembles, indicating its robustness to the method selection. In MHMM, ranked third based on External indices, the number of latent states (m) is chosen based on Bayesian Information Criterion (BIC), $BIC = -2l(\theta) + p \log m$ for m in $[1, 5]$. The same setting for the following simulations has been done. Assuming no foreknowledge is available, the prior probability is selected randomly; hence, its accuracy changes constantly.

Table 3 Clustering goodness measures in Simulation 1; here, k in k -mer and n in nTreeClus is floating and equal to the square root of the length of sequences. The starred methods are the ones in which the position of substrings is used in matrix segmentation.

Methods	External Validation				Internal Validation	1NN
	Purity	RI	ARI	F-meas	ASW	
nTreeClus (DT)	0.988	0.980	0.959	0.987	0.655	0.986
nTreeClus (RF)	0.992	0.984	0.969	0.990	0.651	0.989
nTreeClus (DT)*	0.973	0.949	0.898	0.962	0.529	0.978
nTreeClus (RF)*	0.984	0.970	0.940	0.979	0.546	0.986
Levenshtein	0.892	0.830	0.660	0.868	0.343	0.932
Jaro-Winkler	0.654	0.497	-0.006	0.508	0.007	0.234
k -mers						
$k = 1$	0.793	0.688	0.377	0.762	0.385	0.740
$k = 2$	0.910	0.861	0.721	0.901	0.420	0.894
$k = 3$	0.967	0.945	0.891	0.963	0.442	0.958
$k = \sqrt{\mathcal{M}_i}$	0.971	0.936	0.873	0.939	0.346	0.937
MHMM	0.960	0.849	0.700	0.816	-	-

4.4.2 Simulation 2 - gapped pattern recognition

Unlike the rigid and simple nature of the first simulation, the second one consists of sequence patterns with gaps where the relative distance between positions is dynamic. Here, patterns x and y with the length of $\mathcal{L}_{(P_x, P_y)} = \{(2,3), (4,6), (6,9)\}$ respectively have been inserted in each half of the data while the gap in between randomly changes. For $x=\{A,C\}$ and $y=\{A,B,E\}$, it is expected to have them in sequences with different gap in between; for instance, **..ACAABDBAABE..**, **..ACABE..**, or **..ACBEABE..**. The other variables, including $\mathcal{L}$, a , $\mathcal{A}$, $\mathcal{M}$, and $\mathcal{N}_S$, are the same as the first simulation. The second simulation with dynamic distance and shorter pattern's length

aims to examine the performance of each method in identifying intricate patterns in a given categorical sequence.

Table 4 indicates the general performance of the six methods has reduced as the structure of the data becomes more complex. Once again, it shows that nTreeClus results in considerably higher indices than those of the rest of algorithms. Here 3-mer and MHMM are the closest opponent of the current method. On account of the greater complexity of this new dataset, the ASW index for all the methods decreases while the external indices still produce a satisfactory result. Again the worst performance corresponds to the Jaro-Winkler distance in that using edit distance is more rational whenever the length of the sequence is short; namely words and names [85].

Table 4 Clustering goodness measures in Simulation 2; here, k in k -mer and n in nTreeClus is floating and equal to the square root of the length of sequences. The starred methods are the ones in which the position of substrings is used in matrix segmentation.

Methods	External Validation				Internal Validation	
	Purity	RI	ARI	F-meas	ASW	1NN
nTreeClus (DT)	0.939	0.885	0.771	0.904	0.473	0.909
nTreeClus (RF)	0.943	0.899	0.798	0.916	0.466	0.920
nTreeClus (DT)*	0.923	0.853	0.707	0.879	0.387	0.903
nTreeClus (RF)*	0.935	0.881	0.763	0.902	0.378	0.919
Levenshtein	0.869	0.796	0.592	0.839	0.319	0.868
Jaro-Winkler	0.655	0.496	-0.006	0.507	0.007	0.241
k -mers						
$k = 1$	0.788	0.677	0.355	0.748	0.374	0.728
$k = 2$	0.888	0.825	0.650	0.870	0.368	0.864
$k = 3$	0.927	0.886	0.773	0.917	0.335	0.907
$k = \sqrt{M_i}$	0.902	0.774	0.549	0.777	0.195	0.808
MHMM	0.943	0.783	0.571	0.732	-	-

4.4.3 Simulation 3 - length sensitivity analysis

In most real cases, the length of sequences to be compared with each other is varying. In such a case, the method which is robust to the length of sequences can yield the best result. Hence, the third simulation is comprised of $N=360$ batches of sequences, each of which is formed from $\mathcal{L} = \{40, 120, 200\}$ sequences with different characteristics. They may take $a = \{5, 7, 11, 16\}$ different categorical values in $\mathcal{A} = \{A, B, C, \dots, O, P\}$. Here, the length of the sequences ($\mathcal{M}$) oscillates between 80 to 120 and is not fixed even for each batch. Eventually, a random pattern with the length of $\mathcal{M}_P = \{6, 10, 20\}$ chosen from set $\mathcal{A}$ has been inserted in each half of the sequences of each batch, thus generating two clusters ($C = 2$). To eliminate any biased randomness, the simulation has been repeated 10 times.

Because of ever-changing nature of sequences, parameter n in nTreeClus has been set to 10 which is equal to the square root of average length 100 ($n = \sqrt{\text{average}(\mathcal{M})} = 10$).

Table 5 Clustering goodness measures in Simulation 3; here, $n = 10$ in nTreeClus is equal to the square root of the average length of sequences. The starred methods are the ones in which the position of substrings is used in matrix segmentation.

Methods	External Validation				Internal Validation	
	Purity	RI	ARI	F-meas	ASW	1NN
nTreeClus (DT)	0.971	0.951	0.901	0.969	0.367	0.966
nTreeClus (RF)	0.983	0.968	0.936	0.981	0.360	0.981
nTreeClus (DT)*	0.964	0.940	0.880	0.962	0.334	0.959
nTreeClus (RF)*	0.978	0.960	0.921	0.975	0.322	0.975
Levenshtein	0.815	0.748	0.496	0.787	0.146	0.898
Jaro-Winkler	0.648	0.496	-0.008	0.503	0.007	0.276
k -mers						
$k = 1$	0.689	0.565	0.129	0.636	0.228	0.706
$k = 2$	0.898	0.830	0.660	0.888	0.201	0.884
$k = 3$	0.969	0.946	0.892	0.967	0.228	0.961
$k = \sqrt{M}$	0.988	0.977	0.953	0.988	0.087	0.983
MHMM	0.944	0.781	0.566	0.741	-	-

Table 5 clearly shows that, based on External Validation indices, nTreeClus and k -mer exceed the others in performance. Notwithstanding the good performance of k -mer in External Validation, it is nTreeClus that outperforms all other methods regarding Average Silhouette Width index. Once more, the worst performance corresponds to Jaro-Winkler and Levenshtein method.

4.4.4 Simulation 4 - analyzing sensitivity to pattern location

In this examination, all the properties of the generated sequences, including N , $\mathcal{L}$, a , $\mathcal{A}$, C , $\mathcal{M}$, and $\mathcal{M}_P$, are the same as simulation 3. The only difference is characteristics of the pattern in sequences. Here, we have inserted the same pattern in each batch while the position of the pattern in half the sequences is exactly the same. For example, we introduced a pattern in the fifth position of half of data and the same pattern in the twelfth position of the other half of data. Therefore, we defined two clusters with exactly the same pattern but in different locations.

Table 6 Clustering goodness measures in Simulation 4; here, $n = 10$ in nTreeClus is equal to the square root of the average length of sequences. The starred methods are the ones in which the position of substrings is used in matrix segmentation.

Methods	External Validation				Internal Validation	
	Purity	RI	ARI	F-meas	ASW	1NN
nTreeClus (DT)	0.740	0.526	0.054	0.538	0.024	0.523
nTreeClus (RF)	0.779	0.525	0.053	0.522	0.014	0.534
nTreeClus (DT)*	0.962	0.935	0.870	0.961	0.198	0.952
nTreeClus (RF)*	0.988	0.979	0.957	0.987	0.142	0.983
Levenshtein	0.994	0.989	0.979	0.994	0.441	0.989
Jaro-Winkler	0.658	0.495	-0.008	0.501	0.006	0.350
k -mers						
$k = 1$	0.652	0.503	0.006	0.535	0.184	0.609
$k = 2$	0.670	0.513	0.026	0.560	0.055	0.672
$k = 3$	0.740	0.600	0.200	0.679	0.020	0.707
$k = \sqrt{M}$	0.951	0.902	0.804	0.940	0.013	0.900
MHMM	0.883	0.497	0.002	0.370	-	-

Table 6 indicates that most of the available methods fail in finding the location of exactly the same pattern in a dataset; nonetheless, Levenshtein (edit distance) and a modification of nTreeClus can capture such a behavior in a data easily. In location-modification of nTreeClus, the position of each substring is recorded, therefore, Decision Tree ensembles treat it as a feature while generating the autoregressive tree. It leads to have better understanding of the nature of pattern. On the other hand, Levenshtein distance records all the edits needed to transform a substring to another. In this case, the location of the pattern is crucial to have less edits and it can be captured by counting the number of deletion, insertion, and substitution required to turn one string to another. Consequently, It is suggested that, in nTreeClus, the position factor of substrings should be considered wherever the location of the pattern is of importance.

4.5 Estimating the number of clusters

In this experiment, the number of clusters has been estimated to find out how consistent the nTreeClus representation of the sequential data is with the internal indices. Average Silhouette Width, Dunn Index, and Calinski and Harabasz have been considered as the functions examining inter- and intra-clusters sum of squares. The data to measure the efficacy of the method in estimating the number of clusters consists of $N=216$ batches of sequences; based on definition, for them $\mathcal{L} = \{30, 100\}$, $a = \{4, 7, 20\}$, $\mathcal{M} = \{50, 100\}$, and $\mathcal{M}_P = \{7, 18\}$. This time, the number of clusters (C) also has changed and has been taken from the values of 3, 6, and 10. The trials have been repeated 3 times which provide us with the set of 72 batches with different characteristics.

All the three methods have been utilized to estimate the number of clusters in each batch. Distribution of the estimated number of clusters using internal

cluster validation indices is shown in Table 7. The surprising result is that ASW, Dunn, and CH show a robustness under different scenarios. ASW with the accuracy of 93.5 percent has the highest performance while CH and Dunn index comes second and third with 77.7 and 72.2 percent respectively. The last two methods underestimate the number of clusters whenever there is a high level of noise. The experiment indicates that no matter what the characteristic of the data is, nTreeClus has a high intra-cluster and low inter-cluster similarity which enable it to cluster the sequential data with robust, acceptable internal evaluation score.

Table 7 Distribution of the estimated number of clusters ($\hat{C}$) using internal cluster validation indices for 24 different scenarios, each replicated 3 times

Index	Number of clusters, $\hat{C}$												median
	2	3	4	5	6	7	8	9	10	11-15	16-20		
Sim. 1 ($C = 3$)													
ASW		69									3	3	
Dunn	10	59	3									3	
CH	12	60										3	
Sim. 2 ($C = 6$)													
ASW					66					1	5	6	
Dunn	17				51	4						6	
CH	15	1			56							6	
Sim. 3 ($C = 10$)													
ASW							1	1	67	1	2	10	
Dunn	23								46	3		10	
CH	17	1		1				1	52			10	

Figure 6 shows how the performance of the methods changes for different number of clusters (C). The graph clearly indicates that Random Forest modification of nTreeClus, is robust for different settings of C while the clustering goodness measure of all other methods decreases as the number of clusters increases.

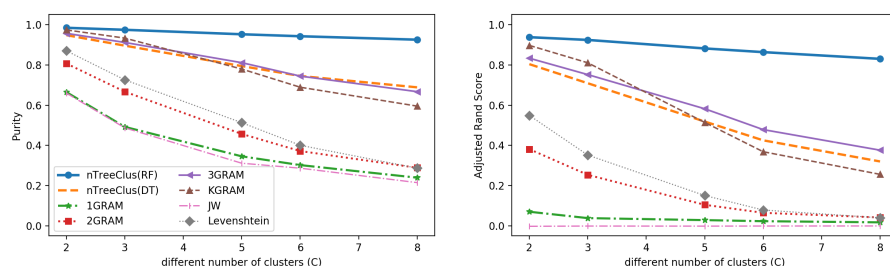


Fig. 6 Analyzing robustness / ruggedness of the methods under different amounts of C

4.6 Real data application

4.6.1 Austrian Wage Mobility Data

The Austrian Wage Mobility dataset, first used by Pamminer and Fruhwirth-Schnatter [86], reports the wage category in successive years for the young men entering labour market from 1975 to 1980. It consists of $\mathcal{L} = 9402$ of such workers whose gross monthly wages of successive years (ranging from $\mathcal{M} = 2$ to $\mathcal{M} = 32$ years) have been examined. The gross monthly wages have been divided to 6 categories, from 0 to 5, in which zero corresponds to unemployment and categories one to five correspond to the quintiles of the income distribution [87].

Pamminer and Fruhwirth-Schnatter use Dirichlet multinomial clustering and Markov Chain clustering, suggesting that the number of cluster is between four and six. We have tried nTreeClus on the data and for the Internal Validations, ASW and CH, it recommends 5 and 6 clusters. Considering the weighted average score of both validation methods, the number of clusters ($\hat{C}$) is more likely to be 5 (Figure 7).

Since the same dataset has been examined by Garca-Magarios and Vilar [88] with special attention to $\hat{C} = 4$ and $\hat{C} = 5$, and by Pamminer and Fruhwirth-Schnatter [86] with special attention to $\hat{C} = 4$, nTreeClus has been applied for the both cases to contrast it with the result of the previous studies.

Parameter n in nTreeClus has been set to 1, since the shortest sequence in the given dataset has the length of 2 years. In this case, where n is too small, nTreeClus cannot capture the relation between the categorical time series for the upcoming years. If n increases, inevitably short sequences should be removed from the dataset and in order to keep the consistency between nTreeClus and the other two methods, we decided not to augment the value of n . In spite of the compulsion to choose the least possible value for n , the output seems easily interpretable. Previously, the dataset has been categorized as *unemployed*, *climbers*, *low-wage*, and *flexible*, while nTreeClus suggests a new behavior in the dataset (shown in the first row of Figure 8), in which flexible (*temporary workers*), climbers to the level 4 of wage (*middle-level employees*), top-level employees (*top managers*), and finally *the low-paid workers* can be seen. Making inferences clearer, the first mixed behavior can be considered as the ordinary workers who have no opportunity to take middle or high rank in the organization and after a while they become unemployed; the second group are those who are the middle-level employees with the chance of attaining to a better paying position but not the highest; the third cluster mostly contains top-level employees who move mainly between the fourth and fifth income level; and finally the last cluster can be considered as the ones who work and work without any improvement in their income, keeping them as a low-wage employees during their whole service to the organization. Comparing to Fig. 6 in Pamminer and Fruhwirth-Schnatter [86] and to Fig. 11 in Garca-Magarios and Vilar [88], nTreeClus includes the *low-wage* and *climbers* while defines new easily interpretable groups of *top managers* and *temporary workers*.

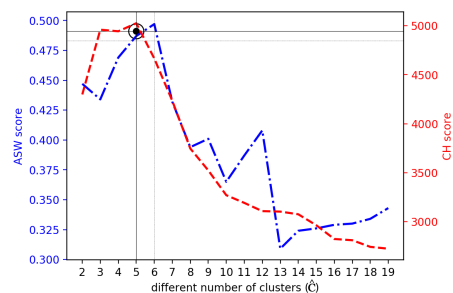


Fig. 7 Finding the best number of clusters ($\hat{C}$) using Average Silhouette Width and Calinski-Harabasz Index

For $\hat{C} = 5$ (Figure 9), which is mainly discussed by Garca-Magarios and Vilar [88] and is considered as the optimal number of clusters by nTreeClus, flexible (*temporary workers*) cluster has been split up to two groups; first, the ones who can maintain their position in the organization without quitting the job and even with a limited prospect for promotions; second, the ones who resign from the organization after a short period and remain unemployed. The transition diagram in the second row of Figure 9 clearly indicates how each cluster moves among different wage levels. For cluster 1 (the most left Figure), there is a chance of promotion from level 1 to level 3, while for the cluster 2, the probability of quitting the job is more than any other states. For cluster 3, going from state three to four, for cluster 4, going from state four to five, and for the last cluster, remaining as a low-paid worker (keeping in a constant level 1) have the highest likelihood.

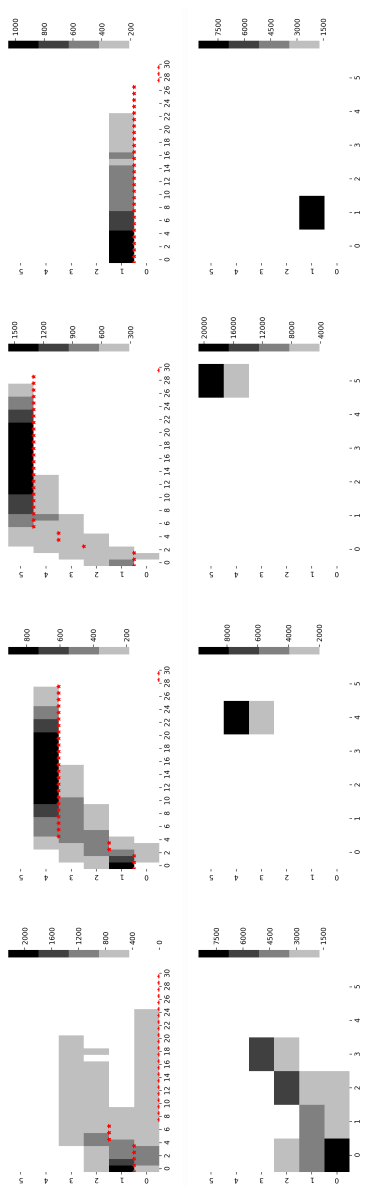


Fig. 8 Heat-map of the wage clusters for $\hat{C} = 4$ (first row) and the transition between different wage groups for $\hat{C} = 4$ (second row)

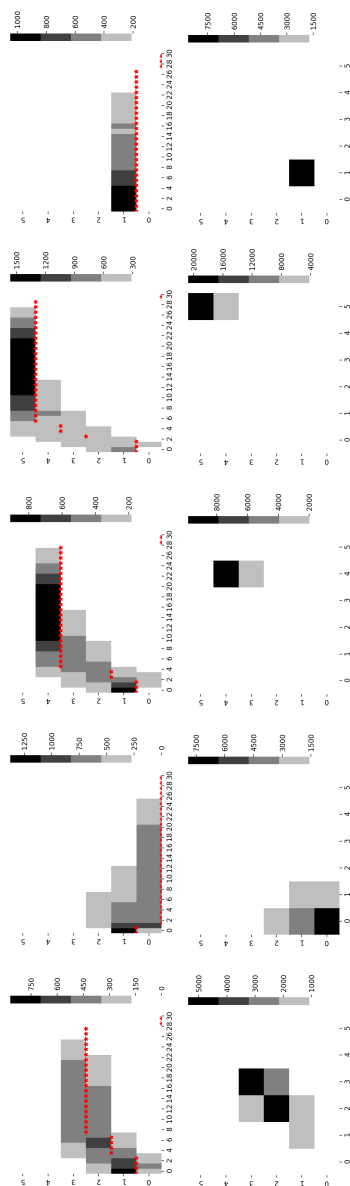


Fig. 9 Heat-map of the wage clusters for $\hat{C} = 5$ (first row) and the transition between different wage groups for $\hat{C} = 5$ (second row)

4.6.2 Protein Data

As the second real dataset, a protein dataset has been taken from [7], containing $\mathcal{L} = 2112$ sequences with the length $\mathcal{M}$ between 80 and 127. The protein dataset has $a = 20$ alphabets (amino acids). Each sequence has one of two tasks, viz. “Might get involved in a Signal Recognition Particle (SRP) pathway” or “Bind to DNA and changes its conformation”. Sequences are labeled based on these two functions and have a reasonably balanced distribution (979 of the first task and 1133 of the second task).

Table 8 Clustering goodness measures in Protein Dataset; here, k in k -mer and n in nTreeClus is equal to the square root of the average length of sequences (11).

Methods	External Validation				Internal Validation
	Purity	RI	ARI	F-meas	ASW
nTreeClus (DT)	1.000	1.000	1.000	1.000	0.815
nTreeClus (RF)	1.000	1.000	1.000	1.000	0.832
Levenshtein	1.000	1.000	1.000	1.000	0.950
Jaro-Winkler	0.750	0.500	-0.001	0.480	-0.003
k -mers	0.949	0.903	0.805	0.948	0.444
MHMM	1.000	1.000	1.000	1.000	-

The result shown in Table 8 demonstrates that nTreeClus has the pure result based on External Validation criteria, while for Internal Validation, it is Levenshtein that has the highest Average Silhouette Width. In previous datasets, Levenshtein obtained a low score indicating although there is a probability to get a proper result through the method, this result is highly correlated with the type of the pattern in the sequence. Again, nTreeClus is among the best methods based on the quality of clustering (Internal or External Validation indices).

5 Concluding remarks and future work

In this paper, we argued that sequence mining is a paramount need to be addressed by data scientists and that the state-of-the-art algorithms face major hurdles including universal applicability, computational complexity, sensitivity to the position of a pattern and the length of sequences, and vulnerability to parameter setting. We further showed empirically that based on high external cluster validation indices and low internal cluster validation indices, those methods overfit on sequences in the case that we are inconvensant with the structure of the dataset.

As a step towards mitigating these potential problems, we introduced a new framework for clustering categorical time series. nTreeClus exploits the strength of the current common methods, including Decision Tree Ensembles, k -mers, and autoregressive models for categorical sequences, and introduces a comprehensive approach to answer the aforementioned problems of earlier

works. The study proposes a numeric representation for categorical sequences established upon the autoregressive behavior of the data. At the first phase, sequences are segmented to length-smoothed substrings based on the predefined window size. This segmented matrix is a feed for autoregressive Decision Tree ensembles where each rule of DT describes a pattern in the sequence. In the second phase, nTreeClus representation of the sequences is introduced where the number of observation in each terminal node is counted. It indicates the behavior of sequences on the foundation of the correlative structure of the DT's rules. Using the common similarity methods, the new representation can be clustered.

Our experimental results show that nTreeClus is robust towards parameter setting and provides computationally efficient and superior results on benchmark real and synthetic datasets with different characteristics. Nevertheless, There are many directions in which the method may be extended. The performance of the method for classification should be explored. Furthermore, nTreeClus should be extended to empirically and theoretically investigate categorical sequences which are continuous series of elements and where the time between two of their elements is of importance. This type of dataset can be found in a clickstream mining domain, although discretizing the time and repeating the element is an option to use the current version of nTreeClus for them.

References

1. M. Karaca, M. Bilgen, A.N. Onus, A.G. Ince, S.Y. Elmasulu, Exact tandem repeats analyzer (e-tra): A new program for dna sequence mining, *Journal of Genetics* **84**(1), 49 (2005)
2. R.V. Kantety, M. La Rota, D.E. Matthews, M.E. Sorrells, Data mining for simple sequence repeats in expressed sequence tags from barley, maize, rice, sorghum and wheat, *Plant Molecular Biology* **48**(5), 501 (2002)
3. G. Dong, J. Pei, *Sequence data mining*, vol. 33 (Springer Science & Business Media, 2007)
4. M.N. Garofalakis, R. Rastogi, K. Shim, in *VLDB*, vol. 99 (1999), vol. 99, pp. 7–10
5. A.L. Montgomery, S. Li, sKannan Srinivasan, J.C. Liechty, Modeling online browsing and path analysis using clickstream data, *Marketing Science* **23**(4), 579 (2004)
6. F. Masegla, M. Teisseire, P. Poncelet, in *Encyclopedia of Data Warehousing and Mining* (IGI Global, 2005), pp. 1028–1032
7. C. Ranjan, S. Ebrahimi, K. Paynabar, Sequence graph transform (sgt): A feature extraction function for sequence data mining, arXiv preprint arXiv:1608.03533 (2016)
8. M.S. Chen, J.S. Park, P.S. Yu, Efficient data mining for path traversal patterns, *IEEE Transactions on Knowledge and Data Engineering* **10**(2), 209 (1998)
9. M. Bicego, V. Murino, M.A.T. Figueiredo, Similarity-based clustering of sequences using hidden markov models, *Machine Learning and Data Mining in Pattern Recognition* pp. 86–95 (2003)
10. S.B. Needleman, C.D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, *Journal of Molecular Biology* **48**(3), 443 (1970)
11. T. Smith, M. Waterman, Identification of common molecular subsequences, *Journal of Molecular Biology* **147**(1), 195 (1981)
12. H. Bunke, *Hybrid Methods in Pattern Recognition* (Springer Berlin Heidelberg, Berlin, Heidelberg, 1987), pp. 367–382

13. M. Brudno, S. Malde, A. Poliakov, C.B. Do, O. Couronne, I. Dubchak, S. Batzoglou, Glocal alignment: finding rearrangements during alignment, *Bioinformatics* **19**(suppl₁), i54(2003)
14. N. Bray, I. Dubchak, L. Pachter, Avid: A global alignment program, *Genome research* **13**(1), 97 (2003)
15. E.L. Sonnhammer, R. Durbin, A dot-matrix program with dynamic threshold control suited for genomic dna and protein sequence analysis, *Gene* **167**(1), GC1 (1995)
16. S.F. Altschul, T.L. Madden, A.A. Schffer, J. Zhang, Z. Zhang, W. Miller, D.J. Lipman, Gapped blast and psi-blast: a new generation of protein database search programs, *Nucleic Acids Research* **25**(17), 3389 (1997)
17. (Academic Press, 1990), pp. 63 – 98
18. R.C. Edgar, Muscle: multiple sequence alignment with high accuracy and high throughput, *Nucleic Acids Research* **32**(5), 1792 (2004)
19. J.D. Thompson, D.G. Higgins, T.J. Gibson, Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, *Nucleic Acids Research* **22**(22), 4673 (1994)
20. C. Notredame, D.G. Higgins, J. Heringa, T-coffee: a novel method for fast and accurate multiple sequence alignment¹edited by j. thornton, *Journal of Molecular Biology* **302**(1), 205 (2000)
21. K. Katoh, K. Misawa, K.i. Kuma, T. Miyata, Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform, *Nucleic acids research* **30**(14), 3059 (2002)
22. C.B. Do, M.S. Mahabhashyam, M. Brudno, S. Batzoglou, Probcons: Probabilistic consistency-based multiple sequence alignment, *Genome research* **15**(2), 330 (2005)
23. R.C. Edgar, S. Batzoglou, Multiple sequence alignment, *Current Opinion in Structural Biology* **16**(3), 368 (2006). *Nucleic acids/Sequences and topology*
24. L. Holm, C. Sander, Dali: a network tool for protein structure comparison, *Trends in Biochemical Sciences* **20**(11), 478 (1995)
25. L. Holm, P.i. Rosenstrij, 1/2m, Dali server: conservation mapping in 3d, *Nucleic acids research* **38**(suppl.2), W545 (2010)
26. S. Burkhardt, J. Kärkkäinen, in *Annual Symposium on Combinatorial Pattern Matching* (Springer, 2002), pp. 225–234
27. L.P. Dinu, A. Sgarro, A low-complexity distance for dna strings, *Fundamenta Informaticae* **73**(3), 361 (2006)
28. S.J. Greenhill, Levenshtein distances fail to identify language relationships accurately, *Comput. Linguist.* **37**(4), 689 (2011)
29. P.F. Brown, P.V. deSouza, R.L. Mercer, V.J.D. Pietra, J.C. Lai, Class-based n-gram models of natural language, *Comput. Linguist.* **18**(4), 467 (1992)
30. M.A. Zissman, E. Singer, in *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, vol. i (1994), vol. i, pp. 1/305–1/308 vol.1
31. W.B. Cavnar, J.M. Trenkle, in *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval* (Citeseer, 1994)
32. T. Abou-Assaleh, N. Cercone, V. Keselj, R. Sweidan, in *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.*, vol. 2 (2004), vol. 2, pp. 41–42 vol.2
33. G. Kondrak, in *String processing and information retrieval* (Springer, 2005), pp. 115–126
34. J. Houvardas, E. Stamatatos, *N-Gram Feature Selection for Authorship Identification* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006), pp. 77–86
35. A. Tomovi, P. Janii, V. Keelj, n-gram-based classification and unsupervised hierarchical clustering of genome sequences, *Computer Methods and Programs in Biomedicine* **81**(2), 137 (2006)
36. N.L. Beebe, L.A. Maddox, L. Liu, M. Sun, Sceadan: Using concatenated n-gram vectors for improved file and data type classification, *IEEE Transactions on Information Forensics and Security* **8**(9), 1519 (2013)
37. E. Millar, D. Shen, J. Liu, C. Nicholas, Performance and scalability of a large-scale n-gram based information retrieval system, *Journal of Digital Information* **1**(5) (2006)
38. G.D. Stormo, T.D. Schneider, L. Gold, A. Ehrenfeucht, Use of the perceptron algorithm to distinguish translational initiation sites in e. coli, *Nucleic acids research* **10**(9), 2997 (1982)

39. L.R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, *Proceedings of the IEEE* **77**(2), 257 (1989)
40. W. Zucchini, I.L. MacDonald, R. Langrock, *Hidden Markov models for time series: an introduction using R* (CRC press, 2017)
41. C. Annachhatre, T.H. Austin, M. Stamp, Hidden markov models for malware classification, *Journal of Computer Virology and Hacking Techniques* **11**(2), 59 (2015)
42. S.R. Eddy, Profile hidden markov models., *Bioinformatics* **14**(9), 755 (1998)
43. M. Dehnert, W. Helm, M.T. Htt, A discrete autoregressive process as a model for short-range correlations in dna sequences, *Physica A: Statistical Mechanics and its Applications* **327**(3), 535 (2003)
44. N. Chakravarthy, A. Spanias, L.D. Iasemidis, K. Tsakalis, Autoregressive modeling and feature analysis of dna sequences, *EURASIP J. Appl. Signal Process.* **2004**, 13 (2004)
45. M. Akhtar, E. Ambikairajah, J. Epps, Comprehensive autoregressive modeling for classification of genomic sequences, 6th International Conference on Information, Communications Signal Processing pp. 1–5 (2007)
46. L. Breiman, J. Friedman, R. Olshen, C. Stone, *Classification and regression trees* (cart) wadsworth, Pacific Grove, CA (1984)
47. J.R. Quinlan, C4. 5: Programming for machine learning, *Morgan Kauffmann* **38**, 48 (1993)
48. L. Breiman, Random forests, *Machine Learning* **45**(1), 5 (2001)
49. S. Ren, X. Cao, Y. Wei, J. Sun, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 723–730
50. G. Biau, E. Scornet, A random forest guided tour, *TEST* **25**(2), 197 (2016)
51. V. Jandhyala, E. Michielssen, R. Mittra, Fdtd signal extrapolation using the forward-backward autoregressive (ar) model, *IEEE Microwave and Guided Wave Letters* **4**(6), 163 (1994)
52. H. Zhou, H. Yan, in *2006 IEEE International Conference on Systems, Man and Cybernetics*, vol. 2 (2006), vol. 2, pp. 1286–1290
53. G. Rosen, Comparison of autoregressive measures for dna sequence similarity, *IEEE International Workshop on Genomic Signal Processing and Statistics* pp. 1–4 (2007)
54. K. Blinowska, B. Trzaskowski, M. Kaminski, R. Kus, Multivariate autoregressive model for a study of phylogenetic diversity, *Gene* **435**(1), 104 (2009)
55. M.K. Choong, D. Levy, H. Yan. Clustering of dna microarray temporal data based on the autoregressive model (2008)
56. N.Y. Song, H. Yan, Short exon detection in dna sequences based on multifeature spectral analysis, *EURASIP J. Adv. Signal Process* **2011**, 2:1 (2011)
57. M.A. Jaro, Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida, *Journal of the American Statistical Association* **84**(406), 414 (1989)
58. W.E. Winkler, The state of record linkage and current research problems. Tech. rep., Statistical Research Division, U.S. Census Bureau (1999)
59. G. Stoilos, G. Stamou, S. Kollias, in *The Semantic Web – ISWC 2005*, ed. by Y. Gil, E. Motta, V.R. Benjamins, M.A. Musen (Springer Berlin Heidelberg, Berlin, Heidelberg, 2005), pp. 624–637
60. V.I. Levenshtein, in *Soviet physics doklady*, vol. 10 (1966), vol. 10, pp. 707–710
61. F. van de Pol, R. Langeheine, Mixed markov latent class models, *Sociological Methodology* **20**, 213 (1990)
62. J.K. Vermunt, B. Tran, J. Magidson, Latent class models in longitudinal research, *Handbook of longitudinal research: Design, measurement, and analysis* pp. 373–385 (2008)
63. S. Helske, J. Helske, Mixture hidden markov models for sequence data: the seqhmm package in r (2017)
64. W. Ernst, H.E. van den, R. JanWillem, all models are wrong...: an introduction to model uncertainty, *Statistica Neerlandica* **66**(3), 217
65. X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.H. Zhou, M. Steinbach, D.J. Hand, D. Steinberg, Top 10 algorithms in data mining, *Knowledge and Information Systems* **14**(1), 1 (2008)
66. J.H.W. Jr., Hierarchical grouping to optimize an objective function, *Journal of the American Statistical Association* **58**(301), 236 (1963)
67. Y. Li, et al., Mspkmercounter: a fast and memory efficient approach for k-mer counting, *arXiv preprint arXiv:1505.06550* (2015)

68. B. Leonardo, S. Hansun, Text documents plagiarism detection using rabin-karp and jaro-winkler distance algorithms, *Indonesian Journal of Electrical Engineering and Computer Science* **5**(2), 462 (2017)
69. H. Jahanshahi, M.G. Baydogan. ntreeclus codes. <https://github.com/HadiJahanshahi/nTreeClus>. DOI 10.5281/zenodo.1295516
70. E. Rendón, I. Abundez, A. Arizmendi, E.M. Quiroz, Internal versus external cluster validation indexes, *International Journal of computers and communications* **5**(1), 27 (2011)
71. M. Halkidi, Y. Batistakis, M. Vazirgiannis, Cluster validity methods: part i, *ACM Sigmod Record* **31**(2), 40 (2002)
72. C.C. Aggarwal, *Data mining: the textbook* (Springer, 2015)
73. *Hierarchical Clustering* (Wiley-Blackwell), chap. 4, pp. 71–110
74. T. Caliski, J. Harabasz, A dendrite method for cluster analysis, *Communications in Statistics* **3**(1), 1 (1974)
75. P.J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, *Journal of Computational and Applied Mathematics* **20**, 53 (1987)
76. J.C. Dunn, Well-separated clusters and optimal fuzzy partitions, *Journal of Cybernetics* **4**(1), 95 (1974)
77. N. Sowmiya, B. Valarmathi, A review of categorical data clustering methodologies based on recent studies, *The IIOAB Journal* **8**, 362
78. W.M. Rand, Objective criteria for the evaluation of clustering methods, *Journal of the American Statistical Association* **66**(336), 846 (1971)
79. L. Hubert, P. Arabie, Comparing partitions, *Journal of Classification* **2**(1), 193 (1985)
80. C.C. Aggarwal, C.K. Reddy, *Data Clustering: Algorithms and Applications*, 1st edn. (Chapman & Hall/CRC, 2013)
81. L. Rokach, O. Maimon, *Clustering Methods* (Springer US, Boston, MA, 2005), p. 330
82. X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, E. Keogh, Experimental comparison of representation methods and distance measures for time series data, *Data Mining and Knowledge Discovery* **26**(2), 275 (2013)
83. M.G. Baydogan, G. Runger, Time series representation and similarity based on local autpatterns, *Data Mining and Knowledge Discovery* **30**(2), 476 (2016)
84. E. Keogh, S. Kasetty, On the need for time series data mining benchmarks: A survey and empirical demonstration, *Data Mining and Knowledge Discovery* **7**(4), 349 (2003)
85. P. Christen, in *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on* (IEEE, 2006), pp. 290–294
86. C. Pammlinger, S. Frhwirth-Schnatter, Model-based clustering of categorical time series, *Bayesian Analysis* **5**(2), 345 (2010)
87. C. Pammlinger, *R Package bayesMCclust: Mixtures-of-Experts Markov Chain Clustering and Dirichlet Multinomial Clustering*. R Foundation for Statistical Computing (2012). URL <https://www.rdocumentation.org/packages/bayesMCclust/versions/1.0>
88. M. García-Magariños, J.A. Vilar, A framework for dissimilarity-based partitioning clustering of categorical time series, *Data Mining and Knowledge Discovery* **29**(2), 466 (2015)

Learning Maximally Predictive Prototypes in Multiple Instance Learning

Ozgur Emre Sivrikaya

Department of Industrial Engineering
Bogazici University, Istanbul, Turkey
ozgur.sivrikaya@boun.edu.tr

Mert Yuksekgonul

Department of Industrial Engineering &
Department of Computer Engineering
Bogazici University, Istanbul, Turkey
mert.yuksekgonul@boun.edu.tr

Mustafa Gokce Baydogan

Department of Industrial Engineering
Bogazici University, Istanbul, Turkey
mustafa.baydogan@boun.edu.tr

Abstract

In this work, we propose a simple model that provides permutation invariant maximally predictive prototype generator from a given dataset, which leads to interpretability of the solution and concrete insights to the nature and the solution of a problem. Our aim is to find out prototypes in the feature space to map the collection of instances (i.e. bags) to a distance feature space and simultaneously learn a linear classifier for multiple instance learning (MIL). Our experiments on classical MIL benchmark datasets demonstrate that proposed framework is an accurate and efficient classifier compared to the existing approaches.

1 Introduction

Classification problems can be divided into two with respect to the labeling characteristics of the data, single instance (SI) and multiple instance (MI) problems. In single instance learning (SIL) problems, each instance is individually labeled. A common example for SIL problems is detecting spam e-mails. Each e-mail can be labeled as spam or not. Each e-mail is individually labeled in this example. However, multiple instance learning (MIL) concentrates on bags of instances, not individually labeled instance data. Detecting whether an object is on an image or not can be given as an example to MIL problems. There may be several objects in an image. However, if one is focusing on finding a certain object, i.e. an elephant, other objects in the image might be misleading. Therefore, dividing the image into several patches and solving the problem in MI domain is a good approach. Two different labeling characteristics of classification problems can be seen in Table 1.

A traditional MIL problem consist of some bags and each bag has a number of instances. A simple example of MIL representations can be found in Table 2. In this example, there are two bags B_1 and B_2 , and each instance has two features. Number of instances that each bag has may vary in MIL problems. In this study, instances are represented with X . B_1 has two instances X_1 and X_2 ; B_2 has three instances X_3 , X_4 and X_5 . Previous studies generally use this representation of bags and instances.

Unlike many common problems in machine learning, multiple instance learning problems do not have a fixed input size. Mainly there are two different approaches to address this issue, first one is instance level approaches and the second one is embedding approaches. The instance level approaches try

Instance	Label
X_1	1
X_2	0
X_3	1
X_4	1
X_5	0

(a) SIL Problem

Bag	Instance	Label
B_1	X_1	1
	X_2	
B_2	X_3	0
	X_4	
	X_5	

(b) MIL Problem

Table 1: Labeling in MIL and SI Problems

R^2			
Bag	Label	Feature 1	Feature 2
B_1	1	$X_{1,1}$	$X_{1,2}$
		$X_{2,1}$	$X_{2,2}$
B_2	0	$X_{3,1}$	$X_{3,2}$
		$X_{4,1}$	$X_{4,2}$
		$X_{5,1}$	$X_{5,2}$

Table 2: Representation of a simple MIL example.

to reward a probability per each instance that exists in a bag, then apply some pooling function to probabilities to obtain the final bag probability. In the second type of approaches, an arbitrary function, often a neural network, is used to come up with an embedding for each instance, then again some pooling function is applied to aggregate information from each embedding which is fed to a classifier. In this work, we propose an embedding approach with some modifications. The idea is to find some representative prototypes in the feature space so that bags are linearly separable when they are represented as their distances to the prototypes.

Details of the study will be given in the following chapters under the following structure. Chapter 2 gives an overview of the previous studies of the field. Chapter 3 explains learning algorithm and the solution method of the problem. Updates for the initial learning algorithm also in scope of Chapter 3. Finally, results of the solution approach on various data sets and our conclusions can be found in Chapters 4 and 5.

2 Literature Review

MIL problems have different categorizations based on instance or bag characteristics. We refer to Alpaydin et al. [2015] which divide MIL problems into three categories. These categories are determined with the instance characteristics. If all instances carry bag’s information, i.e. a bag of pictures of similar animal images, instances can be labeled with bag’s label. In second category, one or more instances determines bag’s label, i.e. MIL problems with standard assumption. Standard MIL assumption tells that if a bag has at least one positive instance, then bag’s label is positive Dietterich et al. [2001]. According to Alpaydin et al. [2015], in third category, all instances carry some information about bag’s label. This case occurs when an instance carries only a portion of bag’s information and each instance contributes bag’s label. For example, there is a set of images that belong to the patches of a cat. In that case to label the bag as cat, all information that comes from all individual images is needed to be used. Even certain patches may carry strong information on the image’s label, more than one patch are needed for classification in general.

Even the data set is in MIL domain, problems can be solved using non-MIL techniques. Especially, if each instance keeps all information of a bag label, instance labels may be used directly. Classification can be made using these labels. For example, consider different images of people and main objective is finding the image of a specific person. This case can be formulated as an MIL problem by grouping several images and labelling the bag as positive if a bag includes that person’s image. Each image, i.e. instance in MIL domain, is a pose from a different perspective and lighting. Therefore, it plays an important role in bag’s label. However, individual images can also be labelled and the problem can be reformulated as a non-MIL problem. In other words, a prediction model can be learnt using instances and working in MIL domain is not required.

Standard MIL assumption tells that if a bag has at least one positive instance, then bag's label is positive. Numerous research can be found using standard assumption of MIL Dietterich et al. [2001] and its variations and extensions such as standard assumption of MIL problem under the assumption that a certain amount of positive instances should be in a bag to label a bag positive Weidmann et al. [2003], Li et al. [2013], Wang and Zucker [2000]. The following example can be given for this category: There is a set of pictures from a concert and a learning method is needed to identify the lead vocal. In this case, it is known that the lead vocal is in one of the images. However, exact image is not known. This is a regular MIL problem in which sets of images are bags and images are instances. The aim is to find the instance that makes the bag positive, in other words image that belongs to lead vocal.

In such problems, bag level decision is made with the use of instance level information Zhang et al. [2006]. This approach named as collective approach Xu [2003]. The predictions of instance level logistic regression can be used to estimate the bag level probability Xu and Frank [2004]. Their main assumption is each individual instance has same effect on bag's label which is independent from each other. Therefore, their approach is taking the average of individual instances and maximization of likelihood at the bag level. Collective approach is extended in Raykar et al. [2008] with feature selection and their learning algorithm selects the subset of the most important features in instance level that contributes the bag level classification.

Under standard MIL assumption, one approach is Diverse Density method Maron and Lozano-Pérez [1998], Zhang and A. Goldman [2001] which focuses on the finding a point t whose neighbourhood has the highest positive bags' instances' intensity. This intensity is calculated using the proportion of the instances of positive bags to the negative bags. Another approach focuses on the distances between positive bags and negative bags. These approaches do not make assumptions about instances and concentrates only on distances between bags. One example of this approach is the representation of each bag with the minimum and maximum values of its instances Tax et al. [2011], Gartner et al. [2002].

In Li et al. [2013], instances are not labeled as positive or negative. Instead instances can belong to a concept or non-concept and they can be the member of a positive or negative bag. Their assumption is modelling the positive bags by a mixture of concept and non-concept distributions. Negative bags are determined with the non-concept distributions and positive bags are determined with both concept and non-concept. A positive bag may not have any concept in this case and they indicates that they may add a constraint that adds a minimum concept threshold to the positive bags. To determine whether an instance belongs to a concept, they use traditional classification approaches. Wang and Zucker [2000] adopt the lazy learning strategy, Bayesian-KNN and Citation-KNN in MIL problems. In lazy learning classes of test examples can be found using classes of training examples. However, this technique deals with instances. Therefore, it cannot be used directly in MIL problems. To solve this problem, Hausdorff distance and k-NN approaches in MIL problems are used.

Two different alternatives for bag level classification is summarized in Alpaydin et al. [2015]. One alternative is bag-space approach Amores [2013]. In this approach, distances between bags and instances using some distance function is defined. After calculating these distances, they are used to calculate bag to bag distances. Bag to bag distances are used in classification algorithms such as k-nearest neighbour. This approach is also applied for recommending web index pages based on user behaviour Zhou et al. [2005]. Another bag level classification method is defining directly a kernel over bags. This kernel calculates the difference or similarity between the bags. Each bag is represented by a graph and used graph kernel with support vector machines in Zhou et al. [2009]. The other alternative is using a single vector that is generated from all instances in a bag with some calculation and using this vector in classification.

All research summarized before propose instance level or bag level approaches to MIL problems. Instance level approaches have dimensionality problem. Bag level approaches overcome the problem of dimensionality but have a disadvantage of losing information that can be gathered from instances.

These disadvantages lead to another solution under MIL domain, embedding approaches. Dissimilarity is one of the most popular one of embedding approaches Pekalska and Duin [2005]. It stands for the representation of an object by describing it relative to a set of reference objects. This definition enables instance level information to be kept in a single vector of dissimilarities for each bag. Thus, possibility of using low dimensional instance level information and complete bag level information together make dissimilarity a popular solution method in MIL domain.

Each bag is represented as a dissimilarity vector to other bags in Cheplygina et al. [2015]. Therefore, instances can be represented as features. Several bag to bag dissimilarity definition method is introduced:

1. **Point set** This method defines each bag as a set and maximum mismatch between bags is calculated as the maximum of minimum distances between vectors.
2. **Distribution of instances** This method approximates each bag by a normal distribution with parameters μ and σ and defining a distance formula using these parameters
3. **Attributed graph** In this method instances are the nodes and relationships between the instances are the edges.

Instance level and bag level methods is combined in Cheplygina et al. [2013] by showing the connection between the two methods and representing by combining classifiers and summarize their representations (bag to bag, bag to instance). Bag to bag dissimilarity measures based on pairwise dissimilarities between instances is introduced in Tax et al. [2011]. Bag to bag dissimilarities are calculated by different distance measures including mahalanobis and earth mover's is used. Procedure starts with the estimation the probability densities of instances for each bag and then after obtaining a single bag distribution these distribution vectors are used. They also adopt instance-based methods using overall minimum, mean minimum and hausdorff distances. After defining dissimilarities, they make k-nearest neighbour classifier to obtain the results. A new bag to prototype dissimilarity is introduced in Cheplygina et al. [2016] which combines two standard dissimilarity approaches, first one is determined by the number of training bags and the second one is determined by the number of instances in the set. Bags are labeled with more than two levels in Briggs et al. [2012]. They use random forest to determine the class of an instance which is the fraction of trees in the forest that vote for label y given input x and it is only used in segmentation. They also summarize the whole concept of data generation like determining features of each segment (i.e. turning their data into MIL data). A deterministic annealing algorithm for SVM formulations of MIL problems is presented in Gehler and Chapelle [2007].

Application of neural network based approaches to MIL problems has been drawing attention from several different domains in the recent years Ilse et al. [2018], Wang et al. [2018]. Well known problems have also been reformulated for this particular purpose, such as common computer vision tasks like image classification Wu et al. [2015], weakly supervised object detection Tang et al. [2018], Wan et al. [2019]; sequence predictions Dennis et al. [2018], sentiment analysis Angelidis and Lapata [2017] and sound event detection McFee et al. [2018].

One of the main challenges in MIL solution approaches is providing results without sacrificing interpretability Ilse et al. [2018]. Only a few approaches summarized are capable of generate interpretable results Ilse et al. [2018], Zhang et al. [2006]. One other challenge is developing a robust approach that works in several MIL problems. Certain methods summarized in this chapter outperforms in specific problem cases. Main objective of this study is to provide a basic and interpretable approach to MIL problems which is detailed in Chapter 3.

3 Joint Learning of Prototypes and Classification Boundary in Multiple Instance Learning

3.1 Definitions

Let X_{ij} be a r -dimensional feature vector of instance j . Feature r of instance j is referred to as $X_{ij,r}$. A bag i is a group of instances which is defined as $\chi = \{B_i : i = 1, \dots, M\}$. Each bag is also defined with its label y_i and there are K_i number of instances in bag i . The number of instances is not fixed for each bag. In MIL, we are searching a classifier to identify the bag's label. Each instance has L number of features. L is fixed for all instances in bag i but changes between bags.

The model has two main objectives: First one is learning the feature vectors or prototypes that is maximally predictive of the bag class after finding an embedding in the distance space. A simple illustration of the idea is presented in Figure 1. Suppose there are two positive and two negative bags each of which have two instances in two-dimensional feature space shown in Figure 1a. Our aim is to identify prototypes such that the bags are linearly separable when each bag is represented by its minimum distance to each prototype. Figure 1b represents the bags in the new feature space. In other

words, proposed model is optimized over both the linear classifier parameters and the prototypes. An overview of architecture can be seen in Figure 2. Depending on the application, our proposal is flexible in generating average and maximum type of features which are famous in multiple instance learning domain Cheplygina et al. [2015].

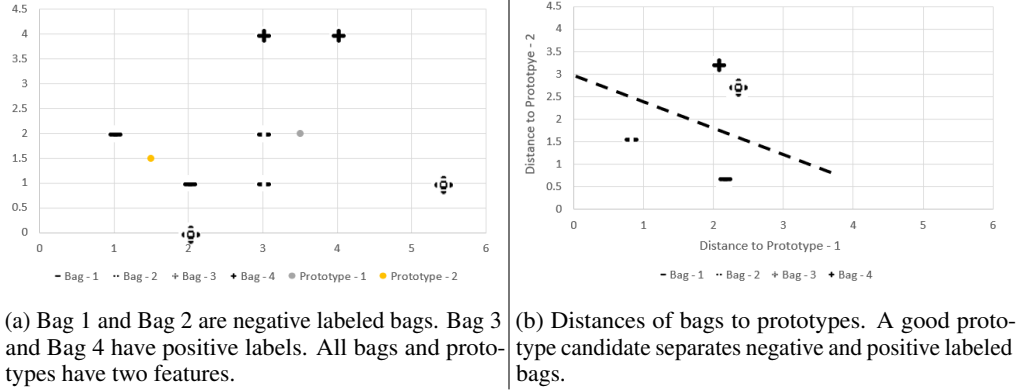


Figure 1: Representation of Bags and Prototypes.

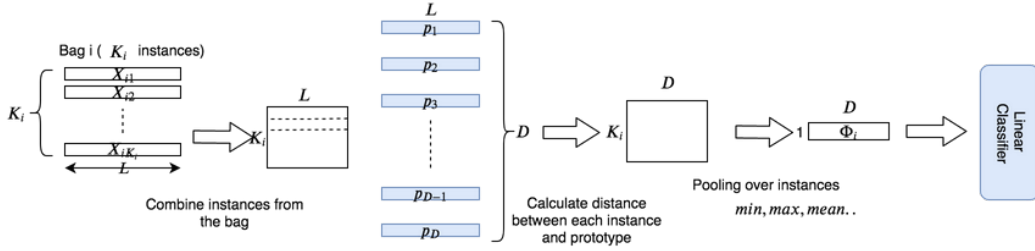


Figure 2: Overview of the architecture. Blue color shows the variables, which the model will be optimized over. L : Number of features in an instance, constant; D : Number of prototypes, hyperparameter; K_i : Number of instances bag i , varies between different bags

As illustrated in Figure 2, this method focuses on the dissimilarities between the bags and the prototypes. Main aim is to search for D prototypes which is generally smaller than the number of bags, M . Let P_d is a vector with length L . Feature r of prototype d is defined as $P_{d,r}$. A bag B_i is represented based on its dissimilarity to each prototype. This way, a bag can be summarized with D features (i.e. distance of bag to each prototype). However, since a bag is composed of multiple instances and there is a need to define the distance between a prototype and instances. Euclidean distance of prototype d to instance X_{ij} , $Dist(X_{ij}, d)$, can be defined as in Equation 1:

$$Dist(X_{ij}, d) = \sum_{r=1}^L (X_{ij,r} - P_{d,r})^2 \quad (1)$$

3.2 Distance Feature Extraction

Just as in instance-level approaches in MIL problems, proposed model also needs to pool information that is extracted from instances in a given bag with potentially variable number of instances. To be more specific, for a given bag after the distance from each instance to each prototype is calculated, the model needs to aggregate the information before being fed into a linear classifier. These pooling operations should be differentiable to be optimized with a gradient based approach. Most basic and widely used pooling operators having these characteristics are min, mean and max operators Cheplygina et al. [2015]. These are also intuitively informative in our case, since we have distance metrics as features, such that these should provide information about defining characteristics of an instance, assuming the existence of prototypes which are described above.

Formally, calculation of average distance between bag i and prototype d , $Dist(i, d)$, can be found in Equation 2. Minimum and maximum distances can also be calculated very similar to below

formulation.

$$Dist(i, d) = \frac{1}{K_i} \sum_{j \in B_i} \sum_{k=1}^{K_i} Dist(X_{ikj}, d) \quad (2)$$

The aim is to learn a logistic regression classifier to predict the binary positive or negative labels of bags, $\hat{Y}$. In logistic regression, $\hat{Y}$ is calculated as in Equation 3. β_0 is the intercept, β_d represent the weight in linear classifier that corresponds to d_{th} prototype. Φ_{id} is the d^{th} element of the output vector for bag i . Corresponds to the minimum distance of bag i to prototype d after layer normalization.

$$\hat{Y}_i = \beta_0 + \sum_{d=1}^D \beta_d \Phi_{id} \quad (3)$$

Logistic regression minimizes a cross-entropy loss function defined in Equation 4. σ is the well-known sigmoid function defined in Equation 5.

$\mathcal{L}_{ce}(y_i, \hat{y}_i)$: Cross-entropy loss where y_i is the label and $\hat{y}_i$ is the prediction

$$\mathcal{L}_{ce}(Y, \hat{Y}) = -Y \ln(\sigma(\hat{Y})) - (1 - Y) \ln(1 - \sigma(\hat{Y})) \quad (4)$$

$$\sigma(\hat{Y}) = \frac{1}{1 + e^{-\hat{Y}}} \quad (5)$$

We can decompose the objective for each bag i and solve the nonlinear optimization problem using gradient descent. Decomposed cross-entropy loss will be denoted as $\mathcal{L}_{ce}(Y_i, \hat{Y}_i)$.

A detailed formulation of the described method can be found in Section 3.3.

3.3 Model Formulation

A formulation for the discussed method is given in this section. The formulation for *min* features is demonstrated for simplicity, which could easily be generalized to any combination of *max*, *min* and *mean*. Also sigmoid is used for binary classification, but the same framework could easily be extended to multinomial case utilizing a softmax function.

λ_w is the regularization parameter for linear classifier weights and λ_p is the regularization parameter for prototypes in the formulation.

$$\min_{P, \beta} \sum_{i=1}^N \mathcal{L}_{ce}(y_i, \hat{y}_i) + \lambda_w \|\beta\|_1 + \lambda_p \sum_{d=1}^D \|P_d\|_2 \quad (6)$$

$$\Phi_{id} = \min_{X_{ik} \in X_i} Dist(X_{ik}, P_d) \quad (7)$$

$$\hat{y}_i = \sigma(\beta_0 + \sum_{d=1}^D \beta_d \Phi_{id}) \quad (8)$$

The objective function given in Equation 6 is optimized over the prototypes, P , and the linear classifier weights, β . Different learning rates are used for prototypes and the classifier parameters, namely α_w for weights and α_p for prototypes.

3.4 Feature Normalization

The distance features are prone to scale issues. This can cause problems with both gradient updates and the learning of linear classifier parameters. To overcome this, we adapt a similar approach to Ba et al. [2016], Ioffe and Szegedy [2015]. In other words, for each bag, we normalize the aggregated distance vector. Note that the information related to "shape" of the distance features will be preserved under the normalization operation that is only recentering and rescaling, which is what we aim to achieve. Namely, each row in the transformed distance space is scaled to zero mean and unit variance as illustrated in Equations 9a, 9b. Layer normalization is important because it stabilizes the issues that could occur during optimization due to the scale of distance features and it reduces the sensibility

of linear classifier to the scale of distances but it keeps the relative distance information. As a side benefit, we observe that as argued in Ba et al. [2016], it speeds up the convergence.

$$\mu_i = \frac{1}{D} \sum_{d=1}^D \Phi_{id}, \sigma_i = \sqrt{\frac{1}{D} (\Phi_i - \mu_i)^2} \quad (9a)$$

$$\Phi_i = \frac{\Phi_i - \mu_i}{\sigma_i} \quad (9b)$$

In our setting, for a given training task we choose a fixed number of prototypes, D , of a fixed size, L , to be learned, we initialize these prototypes randomly. We combine each instance of length L in a given bag, which yields K_i (number of instances in bag i) vectors representing bag i . K_i is not constant between different bags, since each bag potentially has different number of instances. At each training step, we calculate the distance from each instance to the prototypes to extract distance features. Given these features, the model learns a classifier to predict the bag class.

4 Experimental Results

MIL literature have common data sets that are generally used to report algorithm performances. There are 68 data sets that varies from molecular activity prediction to image annotation. Solution approach is tested on all of these data sets. Details of these datasets can be found in Table 3. We repeat a stratified 10-fold cross-validation five times. Randomly generated cross validation indices are taken from Kucukasci and Baydogan [2018]. For all experiments, prototypes are generated randomly and logistic regression is used as default classifier. Important parameters are the number of prototypes to learn and learning rate for weights and prototypes. The model was implemented in PyTorchPaske et al. [2017]. Parameters were tuned on a new randomly generated fold. Namely, the number of epochs is set to 100 with a minibatch size of 1. Adam optimizer from Kingma and Ba [2015] was used, with the tuned learning rate of prototypes and learning rate for weight updates of the linear classifier.

We applied stepwise learning rate decay, namely in every 40 epochs, we decreased the learning rate to the half. L2 regularization was applied to prototypes and L1 regularization was applied to the classifier parameters. The number of prototypes is selected among 4, 8 and 12 (i.e. $D = 8, 16, 24$). Layer normalization is applied as in Ba et al. [2016] to the distance features. Hyperparameter tuning is completed using GPyOpt authors [2016].

Experiments were run on Windows Server 2016 operating system. System has 10.0 GB installed memory (RAM) and 2.30 GHz Intel Xeon(R) Processor.

Table 3: MIL datasets.

Name	Instances	Min	Max	Features	Bags	+ bags	- bags
Musk 1 Dietterich et al. [2001] $\oplus$	476	2	40	166	92	47	45
Musk 2 Dietterich et al. [2001] $\oplus$	6598	1	1044	166	102	39	63
Mutagenesis 1 Srinivasan et al. [1995] $\oplus$	10486	28	88	7	188	125	63
Mutagenesis 2 Srinivasan et al. [1995] $\oplus$	2132	26	86	7	42	13	29
Protein Tao et al. [2004] $\oplus$	26611	35	189	8	193	25	168
Elephant Andrews et al. [2003] $\ominus$	1391	2	13	230	200	100	100
Fox Andrews et al. [2003] $\ominus$	1302	1	13	230	200	100	100
Tiger Andrews et al. [2003] $\ominus$	1220	2	13	230	200	100	100
Corel, African Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Antique Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Battleships Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Beach Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Buses Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Cars Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Desserts Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Dinosaurs Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Dogs Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Elephants Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Fashion Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Flowers Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Food Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Historical Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Horses Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Lizards Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Mountains Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Skiing Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900
Corel, Sunset Chen et al. [2006] $\ominus$	7947	2	13	9	2000	100	1900

Continuation of Table 3							
Name	Instances	Min	Max	Features	Bags	+ bags	- bags
Corel, Waterfalls Chen et al. [2006] ⊖	7947	2	13	9	2000	100	1900
UCSB Breast Cancer Kandemir et al. [2014] ⊖	2002	21	40	708	58	26	32
Newsgroups 1, alt.atheism Zhou et al. [2009] ⊗	5443	22	76	200	100	50	50
N.g. 2, comp.graphics Zhou et al. [2009] ⊗	3094	12	58	200	100	50	50
N.g. 3, comp.os.ms-windows.misc Zhou et al. [2009] ⊗	5175	25	82	200	100	50	50
N.g. 4, comp.sys.ibm.pc.hardware Zhou et al. [2009] ⊗	4827	19	74	200	100	50	50
N.g. 5, comp.sys.mac.hardware Zhou et al. [2009] ⊗	4473	17	71	200	100	50	50
N.g. 6, comp.windows.x Zhou et al. [2009] ⊗	3110	12	54	200	100	50	50
N.g. 7, misc.forsale Zhou et al. [2009] ⊗	5306	29	84	200	100	50	50
N.g. 8, rec.autos Zhou et al. [2009] ⊗	3458	15	39	200	100	50	50
N.g. 9, rec.motorcycles Zhou et al. [2009] ⊗	4730	22	73	200	100	50	50
N.g. 10, rec.sport.baseball Zhou et al. [2009] ⊗	3358	15	58	200	100	50	50
N.g. 11, rec.sport.hockey Zhou et al. [2009] ⊗	1982	8	38	200	100	50	50
N.g. 12, sci.crypt Zhou et al. [2009] ⊗	4284	20	71	200	100	50	50
N.g. 13, sci.electronics Zhou et al. [2009] ⊗	3192	12	58	200	100	50	50
N.g. 14, sci.med Zhou et al. [2009] ⊗	3045	11	54	200	100	50	50
N.g. 15, sci.space Zhou et al. [2009] ⊗	3655	16	59	200	100	50	50
N.g. 16, soc.religion.christian Zhou et al. [2009] ⊗	4677	21	71	200	100	50	50
N.g. 17, talk.politics.guns Zhou et al. [2009] ⊗	3558	13	59	200	100	50	50
N.g. 18, talk.politics.mideast Zhou et al. [2009] ⊗	3376	15	55	200	100	50	50
N.g. 19, talk.politics.misc Zhou et al. [2009] ⊗	4788	21	75	200	100	50	50
N.g. 20, talk.religion.misc Zhou et al. [2009] ⊗	4606	25	79	200	100	50	50
Web recommendation 1 Zhou et al. [2005] ⊗	2212	4	131	5863	75	17	58
Web recommendation 2 Zhou et al. [2005] ⊗	2212	5	200	6519	75	18	57
Web recommendation 3 Zhou et al. [2005] ⊗	2212	5	200	6306	75	14	61
Web recommendation 4 Zhou et al. [2005] ⊗	2291	4	200	6059	75	55	20
Web recommendation 5 Zhou et al. [2005] ⊗	2546	5	200	6407	75	61	14
Web recommendation 6 Zhou et al. [2005] ⊗	2462	4	200	6417	75	59	16
Web recommendation 7 Zhou et al. [2005] ⊗	2400	4	200	6450	75	39	36
Web recommendation 8 Zhou et al. [2005] ⊗	2183	4	200	5999	75	35	40
Web recommendation 9 Zhou et al. [2005] ⊗	2321	5	200	6279	75	37	38
Birds, Brown creeper Briggs et al. [2012] ⊗	10232	2	43	38	548	197	351
Birds, Chestnut-backed chickadee Briggs et al. [2012] ⊗	10232	2	43	38	548	117	431
Birds, Dark-eyed junco Briggs et al. [2012] ⊗	10232	2	43	38	548	20	528
Birds, Hammonds flycatcher Briggs et al. [2012] ⊗	10232	2	43	38	548	103	445
Birds, Hermit thrush Briggs et al. [2012] ⊗	10232	2	43	38	548	15	533
Birds, Hermit warbler Briggs et al. [2012] ⊗	10232	2	43	38	548	63	485
Birds, Olive-sided flycatcher Briggs et al. [2012] ⊗	10232	2	43	38	548	90	458
Birds, Pacific slope flycatcher Briggs et al. [2012] ⊗	10232	2	43	38	548	165	383
Birds, Red-breasted nuthatch Briggs et al. [2012] ⊗	10232	2	43	38	548	82	466
Birds, Swainsons thrush Briggs et al. [2012] ⊗	10232	2	43	38	548	79	469
Birds, Varied thrush Briggs et al. [2012] ⊗	10232	2	43	38	548	89	459
Birds, Western tanager Briggs et al. [2012] ⊗	10232	2	43	38	548	46	502
Birds, Winter Wren Briggs et al. [2012] ⊗	10232	2	43	38	548	109	439
⊕ molecular activity prediction, ⊖ image annotation, ⊗ text classification, ⊙ audio recording classification							
End of Table							

4.1 Classification Accuracy

Solution approach in this study outperforms or at least does as good as all other well-known methods in terms of classification accuracy. Besides this approach has much less parameters compared to a neural network, namely only prototypes and few parameters for linear classifier. Area Under The Curve (AUC) is our primary performance measure to compare the approach with other well-known approaches. Receiver Operating Characteristic (ROC) Curve is a probability graph. It is plotted with True Positive Rate (TPR) against False Positive Rate (FPR). AUC measures the area under ROC curve. AUC is used to show how successful the model makes classification, especially if there is high imbalance between the numbers of positive and negative bags. Comparison of PL and other well-known approaches can be found in Table 4.

Table 4: Results of the algorithms.

Avg of AUC*	APR versiv [1997]	Uni- et al.	CCE and Zhang	Citation - Wang and Zucker [2000]	KNN Cheplygina et al. [2015]	D max min	D mean min	D min min	MILES Chen et al. [2006]	miFV Wei et al. [2017]	PL
Musk1	0.7720	0.8816	0.8712	0.9200	0.9446	0.9312	0.8960	0.8738	0.9086		
Musk2	0.8064	0.7812	0.8501	0.9560	0.9761	0.9563	0.8227	0.7860	0.9415		
Mutagenesis1	0.5007	0.8328	0.8268	0.8196	0.8506	0.7655	0.9096	0.9091	0.7674		
Mutagenesis2	0.4583	0.7267	0.6875	0.5900	0.6467	0.3450	0.8817	0.8667	0.8733		
Protein	0.5089	0.6434	0.5516	0.5611	0.5226	0.8756	0.8723	0.8732	0.8674		
Elephant	0.7280	0.8535	0.8824	0.8762	0.9360	0.9148	0.8858	0.8818	0.9363		
Fox	0.5850	0.6225	0.5689	0.4574	0.6124	0.7039	0.6304	0.6546	0.6718		
Tiger	0.5830	0.8323	0.7521	0.7610	0.8528	0.8498	0.8264	0.8597	0.8916		
CorelAfrican	0.5795	0.8398	0.8615	0.9386	0.9673	0.9664	0.8363	0.8512	0.9020		
CorelAntique	0.5945	0.7216	0.7466	0.8769	0.9223	0.9103	0.7694	0.8006	0.8941		
CorelBattleships	0.5689	0.9123	0.8661	0.9723	0.9806	0.9656	0.8763	0.8893	0.9541		
CorelBeach	0.5956	0.9694	0.9262	0.9719	0.9831	0.9902	0.9766	0.9804	0.9792		

Continuation of Table 4										
Average of AUC*	APR versiv [1997]	Uni- et al. [2007]	CCE Zhou and [2007]	Citation - KNN Zhang Wang and [2000]	D max min Cheplygina et al. [2015]	D mean min Cheplygina et al. [2015]	D min min Cheplygina et al. [2015]	MILES Chen et al. [2006]	miFV Wei et al. [2017]	PL
CorelBuses	0.5908	0.9630	0.8475	0.9622	0.9731	0.9805	0.9439	0.9444	0.9595	
CorelCars	0.6076	0.8779	0.8353	0.9299	0.9476	0.9164	0.8560	0.8641	0.9054	
CorelDesserts	0.5708	0.9220	0.8655	0.9498	0.9740	0.9658	0.9151	0.9433	0.9347	
CorelDinosaurs	0.5508	0.8973	0.8020	0.9484	0.9828	0.9819	0.8823	0.9161	0.9271	
CorelDogs	0.5697	0.8050	0.7515	0.8936	0.9185	0.9107	0.8381	0.8562	0.8440	
CorelElephants	0.6431	0.8931	0.8696	0.9661	0.9825	0.9696	0.8583	0.8841	0.9231	
CorelFashion	0.8681	0.9501	0.9094	0.9833	0.9898	0.9912	0.9044	0.9280	0.9377	
CorelFlowers	0.6358	0.8438	0.8332	0.9318	0.9470	0.9526	0.8562	0.8897	0.8849	
CorelFood	0.8596	0.9848	0.9337	0.9928	0.9977	0.9957	0.9794	0.9838	0.9826	
CorelHistorical	0.7731	0.9777	0.9416	0.9844	0.9982	0.9938	0.9845	0.9865	0.9285	
CorelHorses	0.6131	0.8091	0.7455	0.8607	0.9196	0.9172	0.7982	0.8154	0.8587	
CorelLizards	0.5531	0.9380	0.8973	0.9604	0.9802	0.9711	0.9399	0.9340	0.9531	
CorelMountains	0.9812	0.9875	0.9790	0.9976	0.9996	0.9990	0.9880	0.9926	0.9629	
CorelSkiing	0.5085	0.8747	0.7700	0.9563	0.9604	0.9528	0.8670	0.8729	0.9226	
CorelSunset	0.4951	0.7279	0.7064	0.8033	0.8369	0.7512	0.7039	0.7507	0.7661	
CorelWaterfalls	0.5947	0.8360	0.8646	0.9620	0.9751	0.9662	0.8388	0.8957	0.8909	
UCSBBreastCancer	0.5692	0.6439	0.7063	0.7253	0.8306	0.7908	0.8228	0.8478	0.9081	
Newsgroups1	0.5000	0.7920	0.8032	0.9048	0.9408	0.5000	0.6672	0.6536	0.6564	
Newsgroups2	0.5080	0.6600	0.6380	0.8952	0.8976	0.5544	0.7180	0.6136	0.6736	
Newsgroups3	0.5000	0.6220	0.5844	0.8184	0.8104	0.5000	0.6504	0.6600	0.6584	
Newsgroups4	0.5080	0.6784	0.6364	0.8216	0.8568	0.4792	0.7040	0.6632	0.6252	
Newsgroups5	0.4880	0.6092	0.5852	0.8528	0.8520	0.5592	0.6492	0.6376	0.6280	
Newsgroups6	0.5000	0.7416	0.7324	0.8652	0.8904	0.5720	0.6756	0.6504	0.5852	
Newsgroups7	0.4960	0.6476	0.6328	0.7520	0.7896	0.5468	0.6452	0.6540	0.6396	
Newsgroups8	0.4980	0.6920	0.5704	0.8404	0.8704	0.4600	0.7132	0.6232	0.5936	
Newsgroups9	0.5000	0.8064	0.8176	0.3480	0.3260	0.5000	0.6988	0.6160	0.7124	
Newsgroups10	0.5000	0.7472	0.8212	0.9184	0.9144	0.4764	0.6244	0.6516	0.6948	
Newsgroups11	0.4920	0.7936	0.7140	0.9680	0.9576	0.4600	0.6672	0.6344	0.6208	
Newsgroups12	0.5100	0.7556	0.8064	0.8680	0.8396	0.4664	0.6340	0.6416	0.6236	
Newsgroups13	0.5000	0.5284	0.6212	0.9320	0.9464	0.5000	0.4684	0.6324	0.5680	
Newsgroups14	0.5000	0.7768	0.7800	0.9216	0.9416	0.4648	0.6656	0.6408	0.6492	
Newsgroups15	0.5000	0.7900	0.7628	0.8744	0.9052	0.5172	0.6208	0.6440	0.5756	
Newsgroups16	0.5000	0.7872	0.7728	0.8792	0.8984	0.5088	0.6404	0.6268	0.6796	
Newsgroups17	0.4940	0.7660	0.6920	0.8184	0.8736	0.5312	0.6800	0.6204	0.5136	
Newsgroups18	0.5000	0.8248	0.8448	0.8328	0.8736	0.4632	0.6880	0.6656	0.7000	
Newsgroups19	0.5020	0.8312	0.7656	0.7848	0.8020	0.5608	0.6420	0.6380	0.6676	
Newsgroups20	0.4980	0.7696	0.6464	0.8392	0.8336	0.4432	0.6252	0.6424	0.6172	
Web1	0.5470	0.7883	0.6135	0.4107	0.6337	0.7883	0.8263	0.8337	0.8285	
Web2	0.5223	0.4733	0.4450	0.5013	0.4743	0.5213	0.6997	0.6933	0.6908	
Web3	0.6000	0.6032	0.6491	0.5010	0.7083	0.6081	0.7721	0.7668	0.7058	
Web4	0.5750	0.8340	0.7070	0.5480	0.7990	0.6287	0.8165	0.8465	0.7935	
Web5	0.5398	0.6116	0.5064	0.5066	0.7112	0.7941	0.7313	0.7755	0.6782	
Web6	0.5810	0.6212	0.5200	0.4942	0.5247	0.5430	0.7222	0.7268	0.7412	
Web7	0.5858	0.5921	0.6748	0.5998	0.6904	0.6700	0.6896	0.6879	0.7258	
Web8	0.5517	0.5750	0.4981	0.5785	0.4090	0.3663	0.6521	0.6496	0.6883	
Web9	0.5950	0.6308	0.5990	0.4967	0.7354	0.6867	0.7298	0.7413	0.7502	
BrownCreeper	0.5920	0.9445	0.8825	0.7292	0.8991	0.9272	0.9890	0.9924	0.9260	
Chestnut-backedChickadee	0.5197	0.8267	0.8021	0.8305	0.8530	0.7488	0.8982	0.9099	0.8436	
Dark-eyedJunco	0.6829	0.6674	0.5935	0.6952	0.8559	0.8701	0.9380	0.9474	0.7864	
HammondsFlycatcher	0.5340	0.9922	0.8843	0.7178	0.9439	0.8826	0.9992	0.9989	0.9960	
HermitThrush	0.6681	0.4879	0.5351	0.5696	0.5776	0.8924	0.8077	0.8363	0.7496	
HermitWarbler	0.5934	0.8178	0.6973	0.7346	0.7810	0.9261	0.9812	0.9789	0.8797	
Olive-sidedFlycatcher	0.6441	0.8781	0.7938	0.8525	0.8964	0.9236	0.9693	0.9626	0.9121	
PacificslopeFlycatcher	0.5306	0.8363	0.6992	0.7228	0.7536	0.7691	0.9572	0.9578	0.8307	
Red-breastedNuthatch	0.5829	0.8719	0.7708	0.8033	0.8758	0.8770	0.9789	0.9819	0.8875	
SwainsonsThrush	0.5188	0.8260	0.6867	0.7818	0.7670	0.8841	0.9682	0.9810	0.7709	
VariedThrush	0.5844	0.8686	0.7816	0.7509	0.8397	0.9402	0.9997	0.9998	0.9488	
WesternTanager	0.6957	0.8780	0.7551	0.4749	0.8493	0.8239	0.9877	0.9824	0.8965	
WinterWren	0.5984	0.9654	0.9068	0.9441	0.9315	0.8537	0.9925	0.9902	0.9275	
* 10 fold cross-validation is repeated 5 times.										
End of Table										

We applied the procedure in Friedman [1940] to compare the results of different approaches. Friedman Test is a non-parametric test and concentrates on the average ranks. In our test, Null Hypothesis indicates that average ranks of all different approaches are same. However, when we calculate the test statistic, p-value is less than α at significance level 5%. We reject Null Hypothesis and conclude that average ranks of all different approaches are not same. Therefore, we continue with Nemenyi Test Nemenyi [1963]. In Nemenyi Test, we calculate a critical difference (CD). If average rank of two approaches is greater than Nemenyi CD value, we can conclude that performances of two approach is significantly different than each other. Results of Nemenyi Test can be found in Figure 3. $D_{meanmin}$ is significantly better than the approach presented in this paper in terms of average rank. However, there is no significant difference between our approach and miFV, MILES, D_{maxmin} , D_{minmin} and CCE. Prototype Learning is significantly outperforms Citation KNN and APR.

In order to analyze run time of the algorithm, average run time of each replicate is reported in Table 5. As a result of all experiments, we can conclude that average run times are highly correlated with number of features and number of bags as expected.

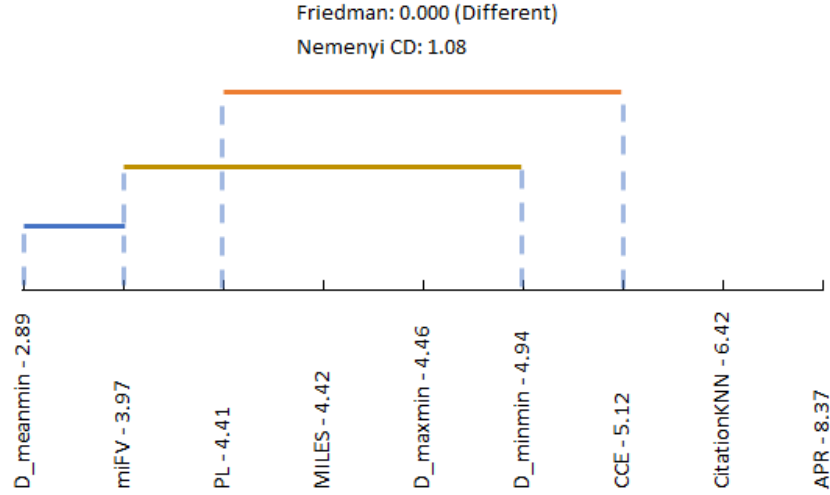


Figure 3: Mean AUC Performances.

Data Set	Run Time	Data Set	Run Time
BrownCreeper	1248	Newsgroups5	728
Chestnut-backedChickadee	1057	Newsgroups6	1139
CorelAfrican	3500	Newsgroups7	1272
CorelAntique	3026	Newsgroups8	1766
CorelBattlehips	3220	Newsgroups9	660
CorelBeach	3199	Newsgroups10	618
CorelBuses	3178	Newsgroups11	1150
CorelCars	3314	Newsgroups12	643
CorelDesserts	3191	Newsgroups13	601
CorelDinosaurs	3239	Newsgroups14	600
CorelDogs	4191	Newsgroups15	599
CorelElephants	3429	Newsgroups16	617
CorelFashion	3643	Newsgroups17	653
CorelFlowers	3581	Newsgroups18	737
CorelFood	3468	Newsgroups19	724
CorelHistorical	3273	Newsgroups20	656
CorelHorses	3238	Olive-sidedFlycatcher	1214
CorelLizards	3296	PacificslopeFlycatcher	1177
CorelMountains	3182	Protein	1442
CorelSkiing	3436	Red-breastedNuthatch	1300
CorelSunset	4086	SwainsonsThrush	1106
CorelWaterfalls	3486	Tiger	778
Dark-eyedJunco	1097	UCSBBreastCancer	1279
Elephant	1210	VariedThrush	1109
Fox	1518	Web1	1219
HammondsFlycatcher	1086	Web2	1890
HermitThrush	1206	Web3	2384
HermitWarbler	1288	Web4	2789
Musk1	715	Web5	3264
Musk2	713	Web6	3027
Mutagenesis1	884	Web7	3363
Newsgroups1	788	Web8	3336
Newsgroups2	760	Web9	3754
Newsgroups3	722	WesternTanager	1089
Newsgroups4	625	WinterWren	1095

Table 5: Average run time of the algorithm (in seconds).

4.2 Interpretability

Interpretability of the solution is one of the main aspects of prototype learning. To demonstrate this interpretability, here we apply our approach to the MNIST MIL problem which was introduced in Ilse et al. [2018]. MNIST database is an image database. The database has 60.000 training and 10.000 test images of handwritten digits. A subset of images from MNIST database can be seen in Figure 5.



Figure 4: MNIST image examples LeCun et al. [1998].

In this case, each instance is an image and each bag consists of images. The task is finding whether a target number exists in images in a bag. To keep things simple, we chose the number of prototypes to be 2. Examples of prototypes from 2 different runs can be seen in Figure 5a and 5b. In this application, we only used *min* as the aggregator for better interpretation. For instance, looking at Figure 5a, we see that second prototype looks a lot like a 9, and the classifier found a negative coefficient for minimum distance to this prototype. This indicates, if the minimum distance to this prototype is larger the output probability will suffer. Moreover, since the first prototype has a positive coefficient, if the minimum distance of the bag to this prototype is larger, the output probability will be higher. Same analysis can be done for Figure 5b.

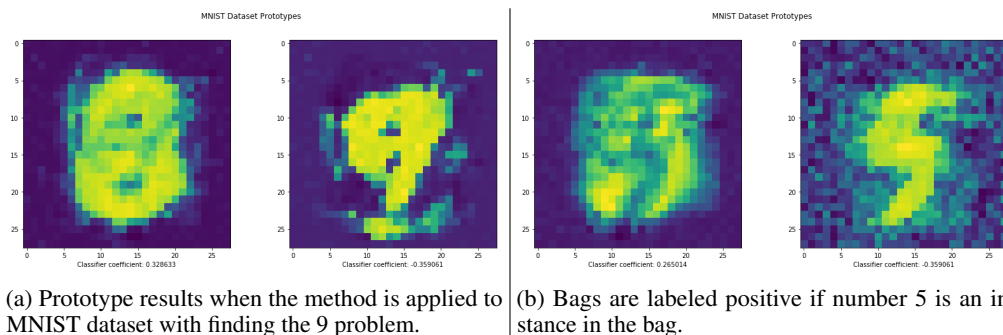


Figure 5: MNIST Prototypes.

5 Conclusion

This work presents a prototype learning framework for MIL problems. This simple architecture can be applied to data from all kinds of domains and offers interpretability of the solutions. Although the method is applied to the classical MIL datasets, its modification to different problems is an interesting research direction. Our aim is to present the simplicity of the approach and hence stuck to the logistic regression classifiers and Euclidean distance. We compared our results with well-known approaches in the field. Our approach is not the best performing approach among these state-of-art approaches. However, considering the flexibility of the architecture, one could incorporate more complex classifiers, different distance metrics and different aggregation procedures to obtain more powerful models. We applied our approach to a well-known MNIST problem in order to show

interpretability power of our approach. As a result of this example we can learn perfectly interpretable prototypes.

Acknowledgments: This research is supported by Air Force Office of Scientific Research Grant FA9550-17-1-0138.

References

- Ethem Alpaydin, Veronika Cheplygina, Marco Loog, and David Tax. Single- vs. multiple-instance classification. *Pattern Recognition*, 48, 04 2015.
- Thomas Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 03 2001.
- Nils Weidmann, Eibe Frank, and Bernhard Pfahringer. A two-level learning method for generalized multi-instance problems. *Machine Learning: ECML*, pages 468–479, 2003.
- Yan Li, David M.J. Tax, Robert P.W. Duin, and Marco Loog. Multiple-instance learning as a classifier combining problem. *Pattern Recognition*, 46(3):865 – 874, 2013.
- Jun Wang and Jean-daniel Zucker. Solving the multiple-instance problem: A lazy learning approach. *Proc. 17th International Con. on Machine Learning*, pages 1119–1126, 01 2000.
- Cha Zhang, John C. Platt, and Paul A. Viola. Multiple instance boosting for object detection. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1417–1424. MIT Press, 2006. URL <http://papers.nips.cc/paper/2926-multiple-instance-boosting-for-object-detection.pdf>.
- X. Xu. Statistical learning in multiple instance problems (m.sc. thesis), university of waikato. *University of Waikato*, 2003.
- X. Xu and E. Frank. Logistic regression and boosting for labeled bags of instances. *Proceedings of the Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 272–281, 2004.
- V.C. Raykar, B. Krishnapuram, J. Bi, M. Dundar, and R.B. Rao. Bayesian multiple instance learning: Automatic feature selection and inductive transfer. *Proceedings of the 25th International Conference on Machine Learning, ACM, New York, NY*, pages 808–815, 2008.
- Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. pages 570–576. MIT Press, 1998.
- Qi Zhang and Sally A. Goldman. Em-dd: An improved multiple-instance learning technique (preliminary version). 08 2001.
- David Tax, Marco Loog, Robert Duin, Veronika Cheplygina, and Wan-Jui Lee. Bag dissimilarities for multiple instance learning. 7005:222–234, 09 2011.
- T Gartner, PA Flach, A Kowalczyk, and AJ Smola. *Multi-Instance Kernels*, pages 179 – 186. Morgan Kaufmann, 7 2002.
- J. Amores. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence 201*, pages 81–105, 2013.
- Z. Zhou, K. Jiang, and M. Li. Multi-instance learning based web mining. *Applied Intelligence*, 22(2): 135–147, 2005.
- Z.H. Zhou, Y.Y. Sun, and Y.F. Li. Multi-instance learning by treating instances as non-iid samples. *Proceedings of International Conference on Machine Learning (ICML)*, pages 1249–1256, 01 2009.
- E. Pekalska and R. P. W. Duin. The dissimilarity representation for pattern recognition: Foundations and applications. *Singapore: World Scientific*, 2005.

- Veronika Cheplygina, David M.J. Tax, and Marco Loog. Multiple instance learning with bag dissimilarities. *Pattern Recognition*, 48(1):264 – 275, 2015.
- Veronika Cheplygina, David Tax, and Marco Loog. Combining instance information to classify bags. *Multiple Classifier Systems, submitted*, 05 2013.
- V. Cheplygina, D. M. J. Tax, and M. Loog. Dissimilarity-based ensembles for multiple instance learning. *IEEE Transactions on Neural Networks and Learning Systems*, 27(6):1379–1391, June 2016.
- Forrest Briggs, Balaji Lakshminarayanan, Lawrence Neal, Xiaoli Z. Fern, Raviv Raich, Sarah J. K. Hadley, Adam S. Hadley, and Matthew G. Betts. Acoustic classification of multiple simultaneous bird species: A multi-instance multilabel approach. *The Journal of the Acoustical Society of America*, 131:4640–50, 2012.
- Peter Gehler and Olivier Chapelle. Deterministic annealing for multiple-instance learning. 2007.
- Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. 80:2127–2136, 10–15 Jul 2018. URL <http://proceedings.mlr.press/v80/ilse18a.html>.
- Xinggong Wang, Yongluan Yan, Peng Tang, Xiang Bai, and Wenyu Liu. Revisiting multiple instance neural networks. *Pattern Recogn.*, 74(C):15–24, February 2018. ISSN 0031-3203. doi: 10.1016/j.patcog.2017.08.026. URL <https://doi.org/10.1016/j.patcog.2017.08.026>.
- J. Wu, Yinan Yu, Chang Huang, and Kai Yu. Deep multiple instance learning for image classification and auto-annotation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3460–3469, June 2015. doi: 10.1109/CVPR.2015.7298968.
- Peng Tang, Xinggong Wang, Song Bai, Wei Shen, Xiang Bai, Wenyu Liu, and Alan L. Yuille. PCL: proposal cluster learning for weakly supervised object detection. *CoRR*, abs/1807.03342, 2018. URL <http://arxiv.org/abs/1807.03342>.
- Fang Wan, Chang Liu, Wei Ke, Xiangyang Ji, Jianbin Jiao, and Qixiang Ye. C-MIL: continuation multiple instance learning for weakly supervised object detection. *CoRR*, abs/1904.05647, 2019. URL <http://arxiv.org/abs/1904.05647>.
- Don Kurian Dennis, Chirag Pabbaraju, Harsha Vardhan Simhadri, and Prateek Jain. Multiple instance learning for efficient sequential data classification on resource-constrained devices. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems, NIPS’18*, pages 10976–10987, USA, 2018. Curran Associates Inc. URL <http://dl.acm.org/citation.cfm?id=3327546.3327753>.
- Stefanos Angelidis and Mirella Lapata. Multiple instance learning networks for fine-grained sentiment analysis. *CoRR*, abs/1711.09645, 2017. URL <http://arxiv.org/abs/1711.09645>.
- Brian McFee, Justin Salamon, and Juan Pablo Bello. Adaptive pooling operators for weakly labeled sound event detection. *CoRR*, abs/1804.10070, 2018. URL <http://arxiv.org/abs/1804.10070>.
- Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL <http://arxiv.org/abs/1607.06450>.
- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on Machine Learning - Volume 37, ICML’15*, pages 448–456. JMLR.org, 2015. URL <http://dl.acm.org/citation.cfm?id=3045118.3045167>.
- Emel S. Kucukasci and Mustafa G. Baydogan. Multiple instance learning repository, 2018. URL <http://www.multipleinstancelearning.com/>.
- Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL <http://arxiv.org/abs/1412.6980>.
- The GPyOpt authors. GPyOpt: A bayesian optimization framework in python. <http://github.com/SheffieldML/GPyOpt>, 2016.
- A. Srinivasan, S. Muggleton, and R. King. Comparing the use of background knowledge by inductive logic programming systems. *Proceedings of the 5th International Workshop of Inductive Logic Programming, Department of Computer Science, Katholieke Universiteit Leuven*, pages 199–230, 1995.
- Q. Tao, S. Scott, N. Vinodchandran, and T. T. Osugi. Svm-based generalized multiple-instance learning via approximate box counting. *Proceedings of the twenty-first international conference on Machine Learning, ACM*, page 101, 2004.
- S. Andrews, T. Hofmann, and I. Tsochantaridis. Support vector machines for multiple-instance learning. *Advances in Neural Information Processing Systems 15*, pages 561–568, 2003.
- Yixin Chen, Jinbo Bi, and James Z. Wang. Miles: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):1931–1947, 2006.
- M. Kandemir, C. Zhang, and F. A. Hambrecht. Empowering multiple instance hispathology cancer diagnosis by cell graphs. In: *Medical Image Computing and Computer-Assisted Intervention - MICCAI*, pages 228–235, 2014.
- O. S. Universiv, C. Science, and S. S. Francisco. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89:31–71, 1997.
- Zhi-Hua Zhou and Min-Ling Zhang. Solving multi-instance problems with classifier ensemble based on constructive clustering. *Knowledge and Information Systems*, 11(2):155–170, 2007.
- Xiu-Shen Wei, Jianxin Wu, and Zhi-Hua Zhou. Scalable algorithms for multi-instance learning. *IEEE transactions on neural networks and learning systems*, 28(4):975–987, 2017.
- Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940. ISSN 00034851. URL <http://www.jstor.org/stable/2235971>.
- P. Nemenyi. Distribution-free multiple comparisons. 1963.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998. URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665>.

A tree-based moving average approach for electricity load forecasting

Burakhan Sel^{a,b} (Corresponding Author, burakhan.sel@boun.edu.tr)

Mustafa Gökçe Baydoğan^a (mustafa.baydogan@boun.edu.tr)

^a*Industrial Engineering Department, Faculty of Engineering, Bogazici University,
34342, Bebek, Istanbul, Turkey*

^b*Industrial Engineering Department, Faculty of Management, Istanbul Technical
University, 34367, Macka, Istanbul, Turkey*

A tree-based moving average approach for electricity load forecasting

Abstract

Short-term electricity load forecasts enable proper planning and eliminate the costs related to the overprediction or underprediction of the electricity load. The electricity load is affected by factors such as weather, special days, and time-related factors. Therefore, a large number of predictors is needed considering the variety of weather-related predictors (e.g., temperature, wind, and humidity), heterogeneity of the weather in the consumption regions, and a high number of special days. For example, our data, which is the Turkish hourly electricity load data, consists of 115 predictors, although it does not involve extra predictors that can be important such as humidity and wind. In such cases, load forecasting can be problematic in terms of robustness to redundant predictors and capturing correlations between predictors and consumption values. To resolve these problems, we propose a novel tree-based ensemble model, which is called tree-based moving average, for time series forecasting problems. With the help of using the tree-based ensemble approach, the proposed model deals with a large number of predictors without sacrificing accuracy. As a distinctive feature, the suggested model overcomes the disadvantage of the tree-based ensemble models in capturing autocorrelations in time series data with the integration of the statistical time series approach into the proposed model. Our comprehensive experiments show that the proposed approach provides competitive results compared to benchmark models in point and probabilistic forecasting of the electricity load of Turkey.

Keywords: Forecasting; regression trees; ensemble models; moving average; short-term load forecasting

1. Introduction

Short-term electricity load forecasting is essential for power systems in terms of proper planning and effective operation. Considering the operation sizes of the electricity retailers providing electricity to large industrial areas, such as New York City, even slight improvements in the accuracy of the forecasts result in significant benefits by eliminating the costs related to overprediction or underprediction. In that sense, the accuracy in forecasting is worth to improve especially in the electricity sector.

The short-term electricity load forecasting, which is the term used for up to seven-days-ahead forecasts, usually concerns probabilistic or point forecasts of the hourly electricity consumption. As a characteristic, the hourly electricity consumption behavior follows multiple seasonal patterns, such as daily and weekly cycles. In the literature, one-day and one-week lagged response values are used to keep track of the daily and weekly cycles of the consumption behavior by Qiu et al. (2017) and Lahouar & Slama (2015). Another characteristic is that the consumption behavior is affected by exogenous factors such as weather conditions and special days (i.e., calendar effects) Taylor & Buizza (2003); Lahouar & Slama (2015). Therefore, a large number of predictors is needed in the short-term electricity load forecasting problems considering the variety of weather-related predictors, heterogeneity of the weather in the consumption regions, and a high number of special days.

The statistical time series models such as moving average (MA), autoregressive integrated moving average (ARIMA), and seasonal autoregressive integrated moving average (SARIMA) are used in the electricity consumption forecasting problems to model the autocorrelations in time series data (Chavez et al., 1999; Chen et al., 1995; Barakat et al., 1990). The MA model is one of the computationally simple statistical models, and it assumes a linear relationship between recent observations. The MA model's capability to capture overall relationships between time series observations is degraded when significant observations are beyond the scope of the MA model. The ARIMA and

SARIMA models overcome the disadvantage of the MA model by modeling the trend, noise, or seasonality components of time series data. However, since the mentioned statistical models are univariate models and assume linear relationships between time series observations, the statistical models may not solely be effective for the electricity load forecasting problems that include predictors having nonlinear correlations with the response variable.

The machine learning models are used in the electricity consumption forecasting problems to capture the nonlinear dynamics of the hourly electricity demand and to consider predictors of the electricity consumption (Kuster et al., 2017). The artificial neural network-based models and support vector regression machine-based models are widely used machine learning models in the energy consumption forecasting problems (Wei et al., 2019). The artificial neural network-based models and support vector regression machine-based models have high learning capacity but have limitations in model parameter handling (Ghoddusi et al., 2019). Therefore, the artificial neural network-based models and support vector regression machine-based models are hard to generalize and specific to problems that they are used. Considering the variety of the short-term electricity load problems due to the consumption area, such as a countries or a residential buildings, the low adaptability to other time series forecasting problems indicates a disadvantage for the neural network-based models and support vector regression machine-based models.

The tree-based ensemble models are favorable machine learning models in terms of the adaptability of the tree-based ensemble models to time series forecasting problems. Tree-based ensemble models such as random forest (Breiman, 2001) and extra trees (ET) (Geurts et al., 2006) allow the use of large number of predictors and capture the complex correlations between the predictors and response variable (Cutler et al., 2008). The tree-based ensemble models tend to have low bias and low variance as advantages of utilizing decision trees and aggregating the outputs of multiple decision trees. However, the tree-based ensemble models lack in considering the autocorrelation in time series data by assuming that the observations are independent. Therefore, the tree-based en-

semble models are needed to be adapted to time series forecasting problem. Although the adaptation can be achieved by emphasizing the autocorrelation in time series data with some input variables (i.e., predictors) such as a variable indicating the order of observations, there are still rooms to improve the accuracy of the tree-based ensemble models for forecasting problems by not only using proper inputs but also improving the algorithms.

This study proposes a new tree-based ensemble model that is called tree-based moving average (TBMA) for time series forecasting problems, including the short-term electricity load forecasting problems. The TBMA model is proposed considering the lack of forecasting models in the literature that capture autocorrelations in time series data, handle a large number of predictors, and are generalized for time series forecasting problems. The suggested approach utilizes both of the tree-based ensemble and MA approaches to adapt a tree-based ensemble model to time series forecasting problems. With the help of using the MA approach, the proposed tree-based ensemble model not only captures the correlations between the predictors and the response variable but also captures autocorrelations between the response values. The suggested model can handle a large number of numeric and categorical predictors without sacrificing accuracy as an advantage of using the tree-based ensemble approach. With the features of handling autocorrelations and predictors, the proposed model is suitable for the time series forecasting problems such as the short-term electricity load forecasting problems.

The TBMA model is applied to the hourly electricity consumption data of Turkey with 115 predictors. The data from 2016 and 2017 is used to train the model. The consumption values in 2018 are predicted from one-day-ahead to seven-days-ahead as point and probabilistic forecasts. The predictors are temperature values, previous consumption values, and special day information. The temperature values correspond to the hourly temperature values of the seven cities that consume the biggest part of the electricity in Turkey. The previous consumption value indicates lagged consumption observations. Special day information, such as Ramadan holiday or Republic Day, is also used as predictors

because the consumption behaviors differ according to the specialty of days. The performance of the TBMA model is compared with the ARIMA model with seasonal components, the ET model, and a naive model that predicts a recent consumption value with the last week's consumption value in the corresponding day and hour. The comparisons of point forecasts are made in terms of mean absolute percentage error (MAPE) and mean percentage error (MPE) performance measures. The comparisons of probabilistic forecasts are made in terms of the pinball loss performance measure.

The contribution of this study to the literature is twofold. First, it proposes a novel model that is suitable for hourly electricity forecasting problems and can be generalized for time series forecasting problems with predictors. Second, this study presents the application of the TBMA to a real-life case with hourly electricity consumption data of Turkey provided by Energy Exchange Istanbul on the transparency platform¹.

The rest of the study is organized as follows. In the next section, background information is provided. The methodology of the TBMA model is described in Section 3. The experiment of the TBMA approach in the electricity consumption forecasting problem is explained in Section 4. Section 5 presents the results of the experiment, along with the discussion. The main conclusions are given in Section 6.

2. Background

This section introduces the principles of the MA, regression trees, and tree-based ensemble approaches, which are used in the proposed model.

2.1. Moving average

The MA approach is used in time series forecasting with various versions. The simple equally weighted version of the MA approach, which is called simple moving average (SMA), predicts the next observation with the average of the

¹<https://seffaflik.epias.com.tr/transparency/>, Online accessed: March 2020

last m observations. The formula of the SMA model is given in Equation 1, where m corresponds to the order of the SMA model, $\hat{y}_{t+1}$ is the predicted value for time $t+1$, and y_t is the response value in time t .

$$\hat{y}_{t+1} = \frac{1}{m} \sum_{i=t-m+1}^t y_i \quad (1)$$

The order parameter m affects the sensitivity of the predictions to the changes in the most recent observation. The smaller the m value, the higher the sensitivity to changes in the recent data. When m is equal to 1, the model becomes naive and simply predicts an observation with the response value of the previous observation.

2.2. Regression trees

Decision trees are the machine learning models that are used for regression and classification problems. One of the versions of decision trees is known as Classification and Regression Trees (Breiman, 2017). The procedure of the regression trees is divided into two steps:

1. Divide the train space R into rectangular regions $R_1, R_2, \dots, R_n$ that cover the the entire train space
2. Predict the response values of the observations in the same region (y_i where $i \in R_k$ and $k \in 1, 2, \dots, n$) with their mean or mode.

The aim of the division into regions is to obtain a partition with a low impurity in the final regions. The total impurity of a regression tree is usually measured with the residual sum of square, $RSS = \sum_{k=1}^n \sum_{i \in R_k} (y_i - \hat{y}_{Rk})^2$, where the predicted value $\hat{y}_{Rk}$ for observations in the region R_k is equal to the mean response value of the observations in the region R_k . Since covering all the possible partitions to find the one with the lowest impurity is not computationally efficient (Friedman et al., 2001), the recursive binary splitting method is used to achieve a partition with a low impurity. The recursive binary splitting method is known as a greedy approach because it splits a region with the most impurity reducing cut-point and does not consider the cut-points that

lead to a better impurity reduction in the further steps. The partition of a train space with the recursive binary splitting method is illustrated with an arbitrary dataset in Figure 1.

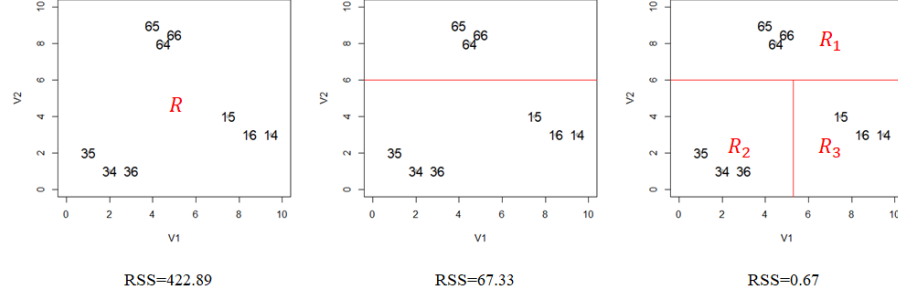


Figure 1: The partition of the train space R into rectangular regions R_1, R_2, R_3 by the recursive binary splitting method. The plotted values are the response values of the corresponding observations. The observations have two input variables that are V_1 and V_2 . The minimum number of observations in a region is allowed to be 3. The predictions are made with the mean of the observations in a region. RSS stands for the residual sum of square.

In the field of electricity consumption forecasting, practitioners usually relate a recent consumption behavior to previous consumption behaviors. In that sense, regression trees are useful to detect similarities among the observations. The observations residing in the same terminal node (i.e., region) can be considered as similar to each other, and further analyses for forecasting can be made according to the similarity information.

2.3. Tree-based ensemble models

The tree-based ensemble models such as the bagging (Breiman, 1996), random forest, and ET models combine multiple decision trees to achieve better performance than their base learners (i.e., decision trees) in terms of bias and variance. The decision tree models provide forecasts with low bias and high variance due to the decision trees' tendency to overfit the train data. The tree-based ensemble models reduce the correlation between decision trees and aggregate the results of decision trees to achieve low bias and low variance. The

bagging model fits multiple decision trees to the bootstrapped samples that are drawn with replacement and aggregates the forecasts of the trees to obtain a final forecast. The random forest model follows similar procedures with the bagging model except for the random subspace technique (Tin Kam Ho, 1998) that is randomly selecting a set of input variables as candidate input variables to be used for splitting. The ET model, which trains decision trees to the samples drawn without replacement, alters the splitting procedure by adding one more random process that is choosing the cut points of the candidate input variables randomly. With the mentioned strategies of the tree-based ensemble models, such as the random subspace technique, diversity of decision trees is achieved to obtain low bias and low variance.

3. Tree-based moving average approach

The idea of the TBMA approach is inspired by electricity consumption forecasting problems. The practitioners in this field usually relate a test observation to old observations by using a set of rules based on features such as temperature ranges, day of week, and hour. The TBMA model relates a recent observation to old observations multiple times by using diversified decision trees and an additional set of rules based on grouping parameters of the TBMA model. With the use of decision trees in the TBMA model, the similarity information between observations is identified in a supervised way in big data.

The TBMA model calculates multiple candidate forecasts by using the similarity information from multiple decision trees and a set of rules based on the grouping parameters. The proposed model aggregates the candidate forecasts by their median value to obtain a point forecast and uses the distribution of the candidate forecasts to obtain probabilistic forecasts.

The procedure of point forecasting a test observation with the TBMA model is divided into three main steps:

1. Determine N sets of train observations that are similar to the test observation $S_1, S_2, \dots, S_N$ with the help of N decision trees

2. Apply SMA to each set to calculate candidate forecasts $C_1, C_2, \dots, C_N$
3. Obtain the point forecast $\hat{y}_{test}$ by the median of the candidate forecasts

The determination of S_k where $k \in (1, 2, \dots, N)$ is based on a decision tree and the grouping parameters of the TBMA model that are group identity (θ) and horizon (h) parameters. The observations in S_k satisfy the following conditions:

- Be a member of the sample that the decision tree is fitted to (i.e., in-bag observations)
- Fall in the same terminal node with the test observation
- Have the same θ feature with the test observation
- Be among the last h observations of the train set

The algorithm fits a decision tree to a sample from a train set. The sample observations that reside in the same terminal node with the test observation are detected for the first step of determining sets of similar observations.

The θ parameter emphasizes the input variables that have significant effects on the response value. The θ parameter is optional to use and can be set to one or more categorical input variables when differences between the observations of different θ levels are significant. For example, in hourly electricity consumption forecasting problems, the hour variable is a potential θ parameter. The hour variable as the θ parameter brings the condition that only the observations that occur in the same hour of days can be similar.

Similar observations must be close to each other in terms of the order of the observations. The h parameter provides a measure for the closeness. The use of h in the TBMA model means that observations have to be among the last h observations of the train set to be considered similar to a test observation.

The SMA part of the TBMA model obtains a candidate forecast by averaging the recent observations in a set of similar observations. The order parameter of the SMA model determines the number of recent observations to average. Since the autocorrelation between the similar observations in a set still exists after

the selection procedure of the similar observations, the TBMA model uses the mean value of the recent and similar observations as a candidate forecast.

The TBMA model finds the median of the candidate forecasts for point forecasting. Since the correlation between the decision trees in tree-based ensemble models is low due to some strategies, the random subspace technique, a wide range of candidate forecasts are obtained from the previous steps. In the presence of diverse and outlier candidate forecasts, the median value becomes a suitable point forecasting because the median is robust to outliers. The algorithm of the TBMA model for point forecasting is given in Algorithm 1.

Algorithm 1: Pseudo-code of the TBMA algorithm.

```

Draw samples from train set;
for each sample do
    Grow a decision tree;
    Identify the terminal node that each sample observation falls in;
    for each observation in the test set do
        Identify the terminal node that the test observation falls in;
        Calculate the candidate forecasts by applying SMA to the
        sample observations that satisfies following conditions:
        Fall in the same node with the test observation
        Have the same group identity with the test observation
        Be among the last  $h$  observations of the train set;
    Predict each test observation with the median of the corresponding
    candidate forecasts;

```

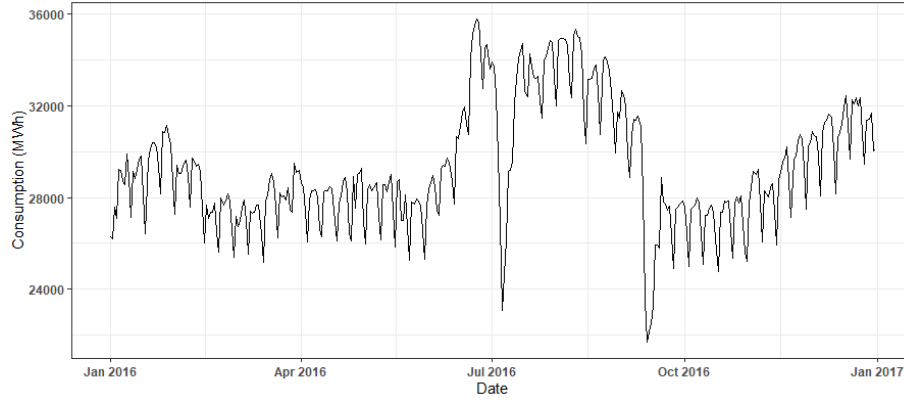
The TBMA approach provides probabilistic forecasts by using the distribution of the candidate forecasts. The probabilistic forecasts give information about the variance of future response values. In the TBMA model, the 50th percentile (i.e., median) of the candidate forecasts is the point forecast. The 75th percentile of the candidate forecasts predicts the value that the future responses will be less than with 0.75 probability.

The parameters of the TBMA model are based on the tree-based ensemble, and MA models except for the θ and h parameters. The θ and h parameters are specifically defined for the TBMA model. The tree-based ensemble part of the TBMA model requires parameters to rule the processes such as fitting decision trees, preventing them from overfitting, and determining the number of decision trees. The required parameters in the MA part of the TBMA model depend on the version of the MA model. In the SMA version of the MA model, there is only the order parameter that determines the number of recent observations to average.

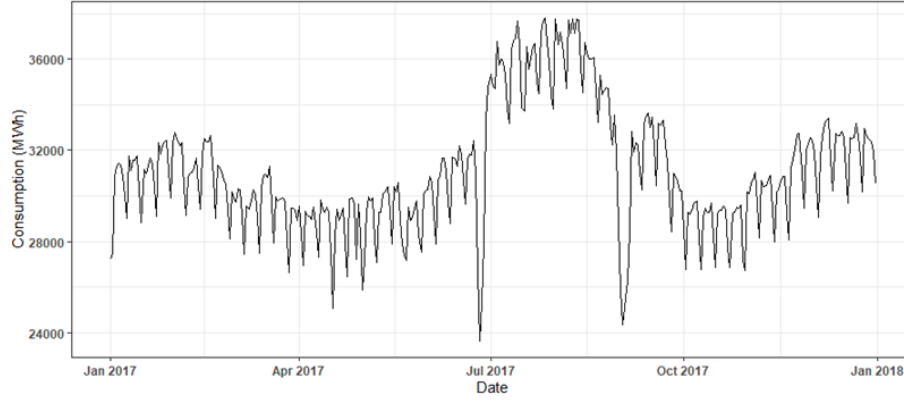
4. Computational experiments

The TBMA approach is applied to the hourly electricity consumption data of Turkey. The data consists of 26280 observations from 2016 to 2018. The consumption values in 2016 and 2017 are visualized in Figure 2. The electricity consumption data is rich in time series components such as seasonality, trend, and noise. The similar consumption behaviors in 2016 and 2017 show the seasonality component in the data. In the same days of the years, the electricity consumption values are generally higher in 2017 than in 2016, which indicates the trend component of the electricity consumption behavior. The variation of the consumption behavior in short time intervals represents the noise element in the data. Since the electricity consumption data reflects typical features of a times series dataset, the electricity consumption data is suitable for the test of the proposed model.

The sudden decreases in the consumption values in July 2016 and June 2017 correspond to the Ramadan holiday, which starts ten days earlier every year. The irregular decrease in August 2016 and August 2017 correspond to another religious holiday Festival of Sacrifice, which also starts ten days earlier every year. In the remaining days of 2016 and 2017, there is no significantly irregular consumption behavior. Therefore, in the further parts of this study, the Ramadan holiday and Festival of Sacrifice are called irregular days, and the



(a) Electricity consumption in 2016.



(b) Electricity consumption in 2017.

Figure 2: Graphs of daily averages of hourly electricity consumption values in Turkey.

remaining days are called regular days.

4.1. Point forecasts

The consumption values in 2018 are predicted from one-day-ahead to seven-day-ahead periods with the TBMA, ET, ARIMA, and naive models. The implementations are provided on Github² to promote reproducibility. The forecasts are obtained by 371 iterations to cover all of the days in 2018. At each iteration,

²<https://github.com/treebasedmovingaverage/TBMA>, Online accessed: March 2020

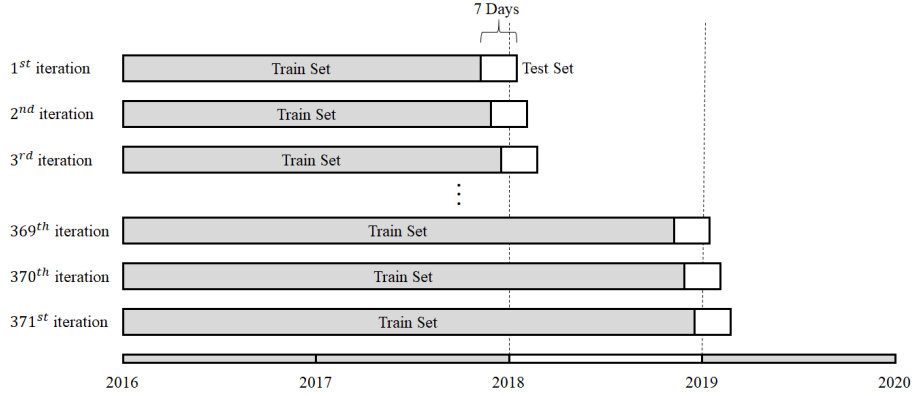


Figure 3: The procedure of splitting train and test sets, and one day enlarging at each iteration.

the consumption values in the next seven days of the train set are predicted, and the train set is enlarged one day in the next iteration. The iterations are illustrated in Figure 3.

The number of predictors is decreased from 115 to 50 with the embedded feature selection process based on a random forest model to increase accuracy, although the proposed model is robust to a large number of predictors. Both of the ET and TBMA models use the 50 predictors. The data from 2018 is saved to the test sets and not used for the feature selection process. The summary of the predictors before and after the feature selection process is shown in Table 1.

The ET and TBMA models utilize the same decision trees for a fair comparison. The same decision trees are fitted to the same samples drawn with replacement. While determining the parameters required for training decision trees, the cross-validation technique is applied to the ET model. During the cross-validation process, three validation sets are created by the time slice technique, which is choosing train and validation sets sequentially from the train set. The parameters of the ET and TBMA models are shown in Table 2, along with their selected levels. The detailed information about the parameters of the ET model is available in Geurts et al. (2006).

The data is split into 24 parts by hours of a day for the application of the

Table 1: Summary of the features before and after the feature selection process.

Feature	Type	Count before selection	Count after selection
Hour	Categorical	1	1
Day of week	Categorical	1	1
Month	Categorical	1	1
Day of month	Categorical	1	1
Special days	Binary	34	17
One day after special days	Binary	34	12
Seven days after special days	Binary	34	8
Temperature	Numerical	7	7
Last week's consumption	Numerical	1	1
Order of the observation in the data	Numerical	1	1
<i>Total</i>		115	50

ARIMA model. The ARIMA model is separately applied to each of the 24 datasets with seven days periodicity. The parameters of the ARIMA model for each of the 24 datasets are selected independently according to the parameter selection method proposed by Hyndman & Khandakar (2008).

The naive model predicts a consumption value with the last week's consumption value in the corresponding day and hour. Although the model is called naive, which is also known as persistence, the naive model in this application has two additional assumptions that electricity consumption behavior in the same days of weeks and the same hours of days are similar. These assumptions are based on the observed consumption behavior in Figure 2.

4.2. Probabilistic forecasts

The TBMA model generates probabilistic forecasts from the distribution of its candidate forecasts. As each decision tree provides a candidate forecast, the percentiles of the candidate forecasts indicate the probabilistic forecasts. The probabilistic forecasts from the 1st to 99th percentiles are obtained by the distribution of the 500 candidate forecasts of the TBMA model. A similar strategy is followed for the probabilistic forecasting with the ET model.

The ARIMA model assumes a normally distributed noise with a constant variance around its point forecasts (i.e., mean). The probabilistic forecasts of

Table 2: The parameters of the Extra Trees and Tree-based Moving Average models.

Model	Parameter name	Parameter level or value
TBMA and ET	Minimum node size	10
	Maximum depth	14
	Replacement	TRUE
	Splitrule	Extra trees splitting rule
	Random splits	5
	Number of features evaluated at each split	7
	Always split variables	Hour, Ramadan, Festival of Sacrifice
	Number of trees	500
TBMA	Order	1
	Weight	Null*
	Group identity	Hour
	Horizon	Null

Order, weight, group identity and horizon parameters are not the parameters of the ET model.

*Null means that the parameter exists but no value is given.

the ARIMA model are calculated by using the point forecasts, the variance of the noise, and the cumulative distribution function of the standard normal distribution.

Since the naive model is a point forecasting model, the probabilistic forecasts are assumed to be the same with the point forecasts of the naive model for every percentile.

5. Result and discussion

This section presents the results of the computational experiments with the TBMA, ET, ARIMA, and naive models in the electricity consumption forecasting problem. The performances of the TBMA, ET, ARIMA, and naive models in point forecasting are analyzed in terms of measures that are based on hourly absolute percentage errors and hourly percentage errors. The performances of the models in probabilistic forecasting are compared in terms of pinball loss.

5.1. Performance of point forecasts

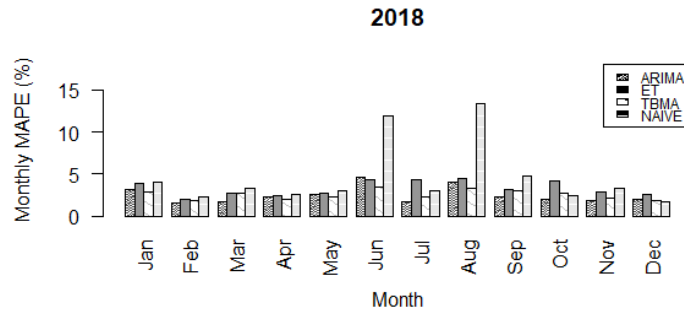
The annual MAPE values are calculated by averaging the hourly absolute percentage errors in a year (i.e., between 2018.01.01 - 2018.12.31). The annual MAPE values of the models for all forecasting horizons are given in Table 3. The TBMA model outperforms the existing models except for the one-day-ahead forecasts. The performance of the naive model is constant in all forecasting horizons because the consumption value in the previous week is independent of the forecasting horizon.

Table 3: The annual MAPE values of the models in 2018.

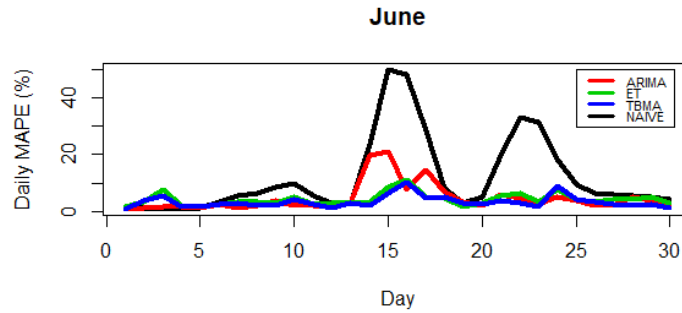
Forecast horizon	Annual MAPE*			
	TBMA	ET	ARIMA	Naive
One-day-ahead	2.547	3.314	2.468	4.658
Two-days-ahead	2.788	3.397	3.486	4.658
Three-days-ahead	2.878	3.419	4.020	4.658
Four-days-ahead	2.881	3.423	4.372	4.658
Five-days-ahead	2.888	3.438	4.637	4.658
Six-days-ahead	2.862	3.453	4.795	4.658
Seven-days-ahead	2.941	3.496	4.878	4.658

*The smallest annual MAPE in each row is written in bold face.

The monthly and daily MAPE performance measures are calculated by averaging the hourly absolute percentage errors in a month and a day. The graph of the monthly MAPE values of one-day-ahead forecasts in 2018 (Figure 4a) and the graph of the daily MAPE values of one-day-ahead forecasts in June 2018 (Figure 4b) are shown in Figure 4. Figure 4a shows that the TBMA model has the lowest monthly MAPE in June and August that include religious holidays: Ramadan holiday and Festival of Sacrifice. In June and August, the naive model has the highest monthly MAPE values because the electricity consumption behavior in religious holidays is significantly different from the electricity consumption behavior in the previous week. According to Figure 4b, the ET and



(a) Graph of the monthly MAPE values of one-day-ahead forecasts in 2018.

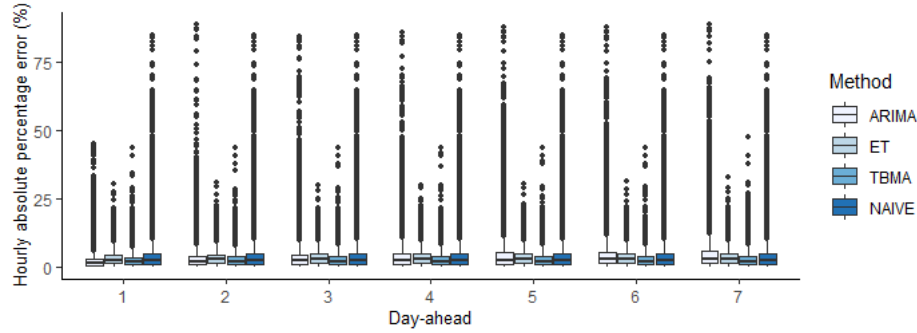


(b) Graph of the daily MAPE values of one-day-ahead forecasts in June 2018.

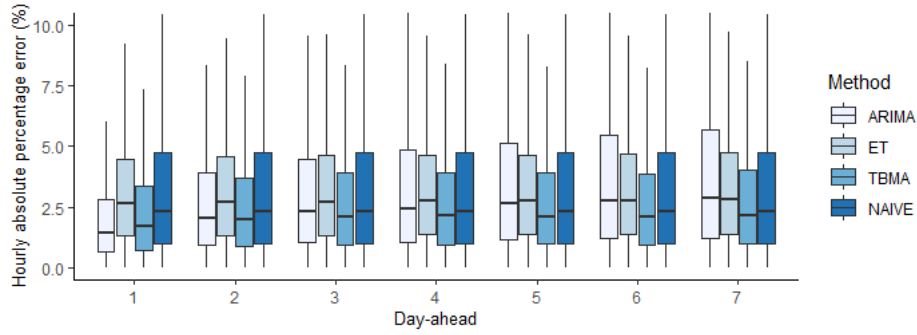
Figure 4: Graphs of monthly and daily MAPE values of one-day-ahead forecasts.

TBMA models outperform the ARIMA and naive models in the days of the Ramadan holiday (i.e., 14.06.2018-17.06.2018). The reasons are that the ARIMA model is univariate, and the naive model is incapable of capturing special day effects with the consumption value in the previous week.

The distributions of the hourly absolute percentage errors for all forecasting horizons are visualized in Figure 5 with outliers (Figure 5a) and without outliers (Figure 5b). The outliers give clues about the errors in irregular days. The distribution between the 1st and 3rd quartiles mostly show the performance of the models in regular days. According to the distributions between 1st and 3rd quartiles, the TBMA model performs better than the other models in all forecasting horizons except for the one-day-ahead forecasts. Although the ARIMA



(a) Boxplots of the hourly absolute percentage errors with outliers.



(b) Boxplots of the hourly absolute percentage errors without outliers.

Figure 5: Boxplots of the hourly absolute percentage errors.

and TBMA models outperform the ET model in early forecasting horizons in terms of the distributions between 1st and 3rd quartiles, the ET model performs better than the other models in terms of outliers. This situation of the ET model is a result of a trade-off between the capability of capturing irregular and regular consumption behaviors. The TBMA model presents a balance point for this trade-off by having a better performance than the ET model in regular days while not compensating significantly from the ability to capture irregular consumption behavior.

The distributions of the hourly percentage error differences between the TBMA and other models are shown in Figure 6. The hourly absolute percentage error of the TBMA model is subtracted from the hourly absolute percentage

error of the other models to compare the performance of the TBMA with the performance of the other models. The distribution that is mostly on the positive side shows the advantage of the TBMA model. Figure 6 shows that the distributions are mostly on the positive side except for the distribution of the difference between the ARIMA and TBMA models in one-day-ahead forecasting. The distribution of the difference between the naive and TBMA models starts on the positive side in one-day-ahead forecasts and decreases until seven-days-ahead forecasts. The reason for the decrease is that as the forecasting horizon approaches to seven days, the effects of the previous week's consumption value on the forecasts of the TBMA and naive models become closer.

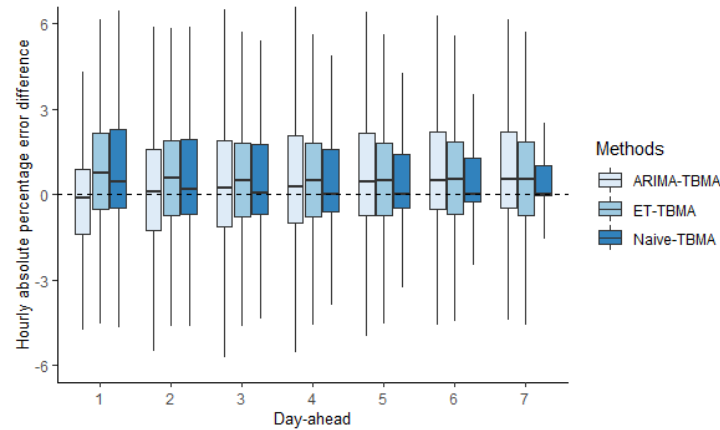


Figure 6: Boxplots of the hourly absolute percentage error differences between each of the ET, ARIMA, and naive models, and the TBMA model without outliers.

The MPE values of the models that are calculated by averaging the hourly percentage errors in a year are shown in Table 4. While the negative MPE values mean underprediction, the positive MPE values mean overprediction. Table 4 shows that the ET model has the closest MPE values to zero. While the ET model tends to overpredict, the TBMA, ARIMA, and naive models tend to underpredict. The MPE of the naive model is negative and stable for all forecasting horizons because of a weekly increasing trend in electricity consumption in general.

Table 4: The MPE values of the models in 2018.

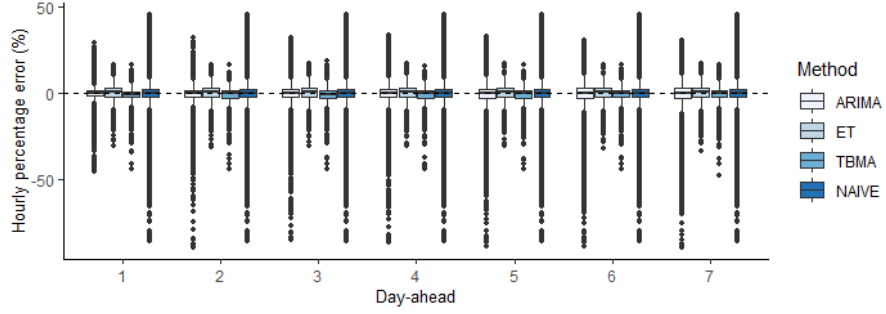
Forecast horizon	MPE*			
	TBMA	ET	ARIMA	Naive
One-day-ahead	-0.805	0.086	-0.136	-0.426
Two-days-ahead	-0.687	0.092	-0.250	-0.426
Three-days-ahead	-0.648	0.106	-0.344	-0.426
Four-days-ahead	-0.574	0.134	-0.419	-0.426
Five-days-ahead	-0.505	0.164	-0.462	-0.426
Six-days-ahead	-0.391	0.201	-0.492	-0.426
Seven-days-ahead	-0.346	0.219	-0.510	-0.426

*The MPE that is the closest to zero in each row is written in bold face.

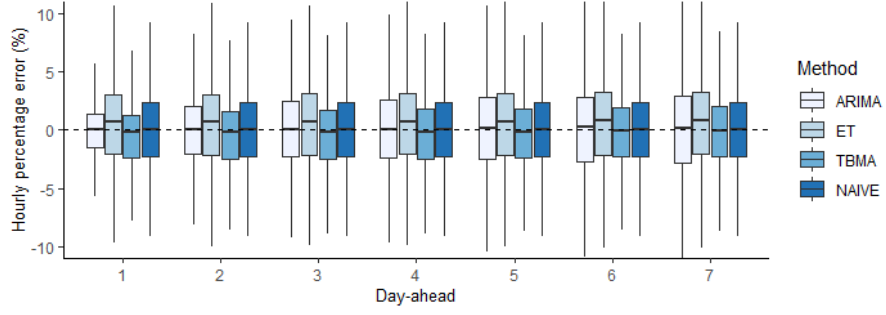
The distributions of the hourly percentage error values are visualized with outliers (Figure 7a) and without outliers (Figure 7b) in Figure 7. The ET model generally overpredicts the consumption value but does not provide high outlier hourly percentage errors. Therefore, the MPE of the ET model is low in Table 4. The ARIMA and naive models are promising models in terms of the distributions of hourly percentage errors between the 1st and 3rd quartiles, but both of the ARIMA and naive models have high outlier hourly percentage errors. The TBMA model presents a balance in terms of outlier hourly percentage errors and the distribution between 1st and 3rd quartiles.

5.2. Performance of probabilistic forecasts

The pinball loss measure is used to test the accuracy of a probabilistic forecast. The pinball loss value for the forecast of a specific day and hour with specific horizon and percentile is calculated as in Equation 2 where d is the day, h is the hour, n is the forecasting horizon, p is the percentile, $\hat{y}$ is the predicted value, and y is the actual value. The lower pinball loss measure indicates a more accurate probabilistic forecast. The mean pinball loss value (W_n) of the



(a) Boxplots of the hourly percentage errors with outliers.



(b) Boxplots of the hourly percentage errors without outliers.

Figure 7: Boxplots of the hourly percentage errors.

n -day-ahead hourly forecasts is calculated as in Equation 3.

$$L_{d,h,n,p}(\hat{y}_{d,h,n,p}, y_{d,h}) = \begin{cases} (1 - \frac{p}{100})(\hat{y}_{d,h,n,p} - y_{d,h}) & \text{if } \hat{y}_{d,h,n,p} \geq y_{d,h} \\ \frac{p}{100}(y_{d,h} - \hat{y}_{d,h,n,p}) & \text{if } \hat{y}_{d,h,n,p} < y_{d,h} \end{cases} \quad (2)$$

$$W_n = \frac{\sum_{d=1}^{365} \sum_{h=0}^{23} \sum_{p=1}^{99} L_{d,h,n,p}(\hat{y}_{d,h,n,p}, y_{d,h})}{365 * 24 * 99} \quad \text{for } n = 1, 2, \dots, 7 \quad (3)$$

The result of the mean pinball loss comparisons in Table 5 is parallel to the results of the annual MAPE comparison of the models in point forecasting. The ARIMA model outperforms the other models in one-day-ahead forecasts. The TBMA model performs better than the other models in the remaining forecasting horizons. The performance of the ET and the ARIMA models decreases

Table 5: Mean pinball loss values in 2018.

Forecast Horizon	Mean Pinball Loss*			
	TBMA	ET	ARIMA	Naive
One-day-ahead	331.719	364.996	318.435	744.131
Two-days-ahead	358.841	375.360	435.456	744.131
Three-days-ahead	368.123	378.695	506.269	744.131
Four-days-ahead	370.877	380.977	556.942	744.131
Five-days-ahead	370.870	383.243	589.496	744.131
Six-days-ahead	367.875	384.977	610.081	744.131
Seven-days-ahead	374.418	390.364	622.058	744.131

*The smallest pinball loss value in each row is written in bold face.

as the horizon of the forecasts increases. The performance of the naive stays constant.

6. Conclusion

A new tree-based ensemble model, which is called TBMA, is proposed for time series forecasting problems. The main idea of the TBMA model is to integrate the MA model to a tree-based ensemble model. The tree-based ensemble models capture complex relationships between predictors and a response variable without being affected by the number of predictors but lack in modeling time series components. The integration of the computationally simple MA model to a tree-based ensemble model helps to model time series components without complicating the computation of the tree-based ensemble model.

The TBMA model is tested in a real-life case, which is the hourly electricity consumption forecasting problem in Turkey. The hourly electricity consumption values in 2018 are predicted from one-day-ahead to seven-days-ahead. The performance of the TBMA model in point and probabilistic forecasting is compared with the ET, ARIMA, and naive models. One conclusion of the analysis of point forecasting is that the TBMA model outperforms the other models in

all forecasting horizons except for the one-day-ahead forecasting in which the ARIMA and TBMA models are competitive. Another conclusion is that the TBMA model accurately captures both of the irregular consumption behaviors in religious holidays and regular consumption behaviors in the remaining days of the year. The analysis of probabilistic forecasting indicates a parallel conclusion to the analysis of point forecasting, which is the outperformance of the TBMA model in all forecasting horizons except for the one-day-ahead forecasts.

There are several limitations to this study. First, the MA part of the TBMA model may lack in capturing the dynamics of a time series dataset because of the simple linearity assumption of the MA model. Thus, the MA part can be altered with other time series models without complicating the computation. Second, the TBMA model is only applied to the electricity consumption problem. The proposed model can be implemented in various benchmark problems to analyse the performance of the TBMA model from a broader perspective.

7. Acknowledgements

This research is supported by Air Force Office of Scientific Research Grant [FA9550-17-1-0138], and The Scientific and Technological Research Council of Turkey 2210 National Scholarships Programme.

8. Data Availability

Datasets related to this article can be found at <https://github.com/trebasedmovingaverage/TBMA> on Github.

References

- Barakat, E. H., Qayyum, M. A., Hamed, M. N., & Al Rashed, S. A. (1990). Short-term peak demand forecasting in fast developing utility with inherit dynamic load characteristics. i. application of classical time-series methods. ii. improved modelling of system dynamic load characteristics. *IEEE Transactions on Power Systems*, 5, 813–824.

- Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24, 123–140. doi:10.1023/A:1018054314350.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32. doi:10.1023/A:1010933404324.
- Breiman, L. (2017). *Classification and Regression Trees*. CRC Press. URL: <https://books.google.com.tr/books?id=MG1QDwAAQBAJ>.
- Chavez, S. G., Bernat], J. X., & Coalla], H. L. (1999). Forecasting of energy production and consumption in asturias (northern spain). *Energy*, 24, 183 – 198. URL: <http://www.sciencedirect.com/science/article/pii/S0360544298000991>. doi:[https://doi.org/10.1016/S0360-5442\(98\)00099-1](https://doi.org/10.1016/S0360-5442(98)00099-1).
- Chen, J.-F., Wang, W.-M., & Huang, C.-M. (1995). Analysis of an adaptive time-series autoregressive moving-average (arma) model for short-term load forecasting. *Electric Power Systems Research*, 34, 187 – 196. URL: <http://www.sciencedirect.com/science/article/pii/0378779695009771>. doi:[https://doi.org/10.1016/0378-7796\(95\)00977-1](https://doi.org/10.1016/0378-7796(95)00977-1).
- Cutler, A., Cutler, D., & Stevens, J. (2008). Tree-based methods. (pp. 1–19). doi:10.1007/978-0-387-69765-9.5.
- Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* volume 1. Springer series in statistics New York.
- Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63, 3–42. doi:10.1007/s10994-006-6226-1.
- Ghoddusi, H., Creamer, G. G., & Rafizadeh, N. (2019). Machine learning in energy economics and finance: A review. *Energy Economics*, 81, 709 – 727. doi:<https://doi.org/10.1016/j.eneco.2019.05.006>.
- Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *JOURNAL OF STATISTICAL SOFTWARE*, 27, 1–22.

- Kuster, C., Rezgui, Y., & Mourshed, M. (2017). Electrical load forecasting models: A critical systematic review. *Sustainable Cities and Society*, 35, 257 – 270. URL: <http://www.sciencedirect.com/science/article/pii/S2210670717305899>. doi:<https://doi.org/10.1016/j.scs.2017.08.009>.
- Lahouar, A., & Slama, J. B. H. (2015). Day-ahead load forecast using random forest and expert input selection. *Energy Conversion and Management*, 103, 1040 – 1051. doi:<https://doi.org/10.1016/j.enconman.2015.07.041>.
- Qiu, X., Ren, Y., Suganthan, P. N., & Amaratunga, G. A. (2017). Empirical mode decomposition based ensemble deep learning for load demand time series forecasting. *Applied Soft Computing*, 54, 246 – 255. doi:<https://doi.org/10.1016/j.asoc.2017.01.015>.
- Taylor, J. W., & Buizza, R. (2003). Using weather ensemble predictions in electricity demand forecasting. *International Journal of Forecasting*, 19, 57 – 70. doi:[https://doi.org/10.1016/S0169-2070\(01\)00123-6](https://doi.org/10.1016/S0169-2070(01)00123-6).
- Tin Kam Ho (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20, 832–844. doi:10.1109/34.709601.
- Wei, N., Li, C., Peng, X., Zeng, F., & Lu, X. (2019). Conventional models and artificial intelligence-based models for energy consumption forecasting: A review. *JOURNAL OF PETROLEUM SCIENCE AND ENGINEERING*, 181. doi:{10.1016/j.petro1.2019.106187}.

Randomized Trees for Time Series Representation and Similarity

Berk Görgülü, Mustafa Gökçe Baydoğan

Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, ON

Department of Industrial Engineering, Bogazici University, Istanbul, Turkey

Abstract

There is a growing interest in the mining of temporal data sets with the increase in the time series data from different domains. Most of the temporal data mining tasks require representations to capture important characteristics of the time series in an efficient way. Representation learning is challenging when time-series data is collected in various domains. For example, series can differ in distributional characteristics such as mean and variance, they can have irregularities such as varying length series, missing observations and outliers that should be addressed carefully. Moreover, when time series observations are multivariate, relation between variables should be characterized in an efficient way. This study proposes a unified, flexible and interpretable time series representation framework called Rand-TS. Rand-TS models density characteristics of each time-series as a mixture of time-varying Gaussian distributions using random decision trees and it embeds density information of each time series into a sparse vector. A simple similarity measure based on this efficient representation is also shown to be successful in classification tasks. Rand-TS is a novel framework for both univariate and multivariate time series (MTS) representation, it can work with time series of various length and missing observations, furthermore, it allows using additional customized features. We evaluate Rand-TS on 85 univariate time-series with non-varying length, 15 univariate time-series with varying length and missing data as well as 19 multivariate time series from UCR database. The results show that Rand-TS provides competitive performance with state-of-the-art similarity-based methods in addition to its flexibility and interpretability.

1. Introduction

There is a growing interest in the mining of temporal data sets due to the ubiquitous nature of the time series data. Most of the temporal data mining tasks require representations to capture important characteristics of the time series in an efficient way. Representation learning is challenging when time-series data is collected in various domains. In the literature, well-known transformations such as fourier, wavelet and piecewise polynomial models are widely

utilized to learn useful representations [11, 14, 25]. Moreover, discretization approaches such as SAX [25, 26, 38, 27] and piecewise average approximation (PAA) [5] provides simple and useful transformations. For example, SAX standardizes time series and models as a standard Gaussian process. It can detect the mean shifts to some extent. However, these transformations potentially fail to represent time series that differ in some distributional characteristics such as mean and variance especially when these characteristics are of time-varying nature. Additionally, irregularities such as varying length series, missing observations and outliers are also should be addressed carefully [41]. The common approaches to deal with vary length series is categorized as uniform scaling [21], adding low amplitude noise [8] and zero padding [33, 35] by [41]. Each technique contains advantages and disadvantages based on the dataset of interest. For a detailed analysis on the time series with varying lengths please refer to [41].

Moreover, time series are also found in the multidimensional domain. These series are called multivariate time series (MTS), and contains multiple attributes. Unlike univariate time series, processing individual attributes is not sufficient for characterizing MTS. Modeling the relations between individual attributes also carry vast importance, therefore, all attributes need to be processed together [2]. This makes the accomplished univariate transformation methods to fail in MTS domain and encourages the researchers to develop representation learning methods tailed for MTS such as [2, 30].

This paper proposes a unified, flexible time series representation framework called Rand-TS. Rand-TS models density characteristics of each time-series as a mixture of time-varying Gaussian distributions using random decision trees. Then, the density information is embedded into a sparse vector for each time series. As Rand-TS models the density, it can process time series with varying lengths and missing data without any additional effort. Moreover, MTS are implicitly modeled as from multivariate Gaussian distribution for which time-varying covariance information between the variables are taken into consideration.

The effectiveness of Rand-TS is evaluated on time series classification problems from various domains. Due to the aforementioned challenges, the state-of-the-art classifiers such as random forest [4], logistic regression and support vector machines [42] usually yield unsatisfactory performances. Moreover, existing transformation methods ([11, 14, 25, 26, 38, 27, 5]) are not extended to multivariate setting and time-series with varying length. In the literature, researchers develop similarity-based methods that aim to address one or more of these challenges. Dynamic time warping (DTW) [35] measures the similarity between given time-series by utilizing one-to-many point comparison. This comparison scheme allows DTW to handle missing points and time-series with varying length. Moreover, it can be generalized to multivariate setting by measuring the similarity separately for each dimension and averaging them. Due to success of DTW, researchers have developed variants of DTW including, Longest common subsequence (LCSS) [24] that uses one-to-many and one-to-none comparison, Weighted DTW (WDTW) [19] penalizes the higher phase difference in DTW, Derivative DTW (DDTW) [23] utilizes first-order difference

of the time series. There are also other successful distance-based methods such as edit distance with real penalty (ERP) [6], time warp edit (TWE) [29] and move-split-merge (MSM) [40].

In the classification setting, it is important to understand the distinguishing characteristics of each class and to identify why a certain time series belongs to a particular class. For example, shapelet-based methods work on raw time series to identify distinguishing subsequences for classification [17, 18, 31, 43, 44, 45, 46, 15, 34, 20]. Feature-based representations are also used for interpretability purposes [10]. However, not all representation learning approaches facilitate interpretability [37, 39, 36]. Rand-TS is a very flexible unified framework for both univariate and multivariate time-series (MTS). It can work with time series of varying lengths and missing observations, furthermore, it allows using additional customized features. In addition to its flexibility, Rand-TS is interpretable. Each feature in the transformed representation points to a subsequence which allows identification of important patterns, moreover, these patterns can be supplied as an input to the shapelet-based classification algorithms. Rand-TS has similarities to a recent work by [3]. LPS transforms given time-series database considering the autoregressive relations and provides state-of-the-art classification performance in both univariate and multivariate setting. Similar to our work, it also utilizes decision trees for learning a new representation, whereas, our framework aims to model time-varying Gaussian distribution by using simple random trees.

The remainder of this paper is structured as follows: Section 2 provides necessary backgrounds. Section 3 introduces the framework for representation learning and similarity calculations. Section 4 illustrates the suggested methods' ability to model time-varying Gaussian distribution and provides evidence on the benefit of using random splits. Section 5 provides experiments demonstrating the classification and computational performance of the suggested method. We conclude in Section 6.

2. Background

Rand-TS benefits from decision trees to learn a density-based representation. This section introduces basics of density estimation with decision trees before introducing the method.

2.1. Randomized Tree Transformation

Decision trees can be represented as a collection of sequential decision rules. Observations in each parent node are split into children nodes based on a given decision rule. These decision rules are usually constructed to myopically minimize the impurity in the children nodes until some stopping condition is satisfied.

The idea of randomly constructed trees is introduced to the literature by [13]. Unlike decision trees, randomized trees are formed by a set of randomly constructed decision rules. A random decision rule is determined by first sampling a feature and then sampling a random threshold uniformly between the

minimum and maximum values of the selected feature in the training set. Due to its random construction, a single random tree might or might not be informative regarding the provided input, however, as shown in [13], a collection of random trees can potentially describe the given input successfully.

2.2. Density Estimation with Random Trees

In this work, we propose a method with the aim of transforming given time-series into a new representation that reflects the distinct features of the time-series. Performing this task requires proper discovery and modeling of the underlying density that generates given time-series. Gaussian Mixture Models (GMMs) are the most popular and widely studied models in the density estimation literature. In addition to GMMs, [7] shows that decision trees can also successfully estimate the underlying density function. When a random tree is constructed on a time-series, each observation ends up in a terminal nodes and distribution of the observations over the all terminal nodes creates a density estimation. This random tree-based density estimation is very flexible and can successfully model **any** underlying density function that generates the given time series. In fact, according to [7], if we assume multivariate Gaussian distribution at the terminal nodes, density forest can be thought as a generalization of Gaussian mixture models (GMM). For a more detailed explanation, please refer to [7].

3. Framework

In this section, we introduce our representation learning method in the univariate setting and provide a similarity-based classification algorithm. A univariate time series $X_i = (X_i^{(1)}, \dots, X_i^{(T)})$ with length T is a set ordered observations that are assumed to be collected in equally divided intervals and a time-series database D is a collection of N univariate time series. Rand-TS utilizes an initial representation that requires time series to be in vertical format (ω) with the time information (t) present as a separate feature. The reason behind including the time information (t) is to capture time-series' temporal characteristics by considering the location of each corresponding observation. An initial representation $\Psi_{NT \times 2}$ constructed from a database D is presented as follows:

$$\Psi_{NT \times 2} = \begin{matrix} & \begin{matrix} t & \omega \end{matrix} \\ \begin{pmatrix} 1 & X_1^{(1)} \\ 2 & X_1^{(2)} \\ \vdots & \vdots \\ T & X_N^{(T)} \end{pmatrix} \end{matrix} \quad (1)$$

3.1. Single-Tree Transformation

After initial representation $\Psi_{NT \times 2}$ is obtained, a random tree is constructed on the $\Psi_{NT \times 2}$ based on randomly generated decision rules until a pre-determined

depth level d is reached. Let $\phi = (s_1, \dots, s_l)$ be a vector of length r where r is the number of terminal nodes in the constructed tree. Each observation $(i, X_i^{(n)})$ of $\Psi_{NT \times 2}$ falls into a single terminal node which allows us to represent this observation with a binary vector where $s_i = 1$ if the observation falls into the terminal node i and 0 otherwise. Then, by applying the same transformation to all observations, a new sparse representation $\Phi_{NT \times r}$ is obtained and represented as follows:

$$\Phi_{NT \times r} = \begin{pmatrix} \phi_1^{(1)} \\ \phi_2^{(1)} \\ \vdots \\ \phi_N^{(T)} \end{pmatrix} = \begin{pmatrix} 0 & \dots & 1 & \dots & 0 & \dots & 0 \\ 0 & \dots & 0 & \dots & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \dots & 0 & \dots & 0 & \dots & 0 \end{pmatrix} \quad (2)$$

Note that for each observation of each time series, $\Phi_{NT \times r}$ contains a binary vector with a single 1 representing the association of that observation to a terminal node. We can create a condensed representation of dimension $N \times r$ from this $NT \times r$ dimensional representation by element-wise summing these binary vectors over each observation for each time series. We call this new condensed representation $\bar{\Psi}_{N \times r}$. Intuitively, $\bar{\Psi}_{N \times r}$ represents nothing but the number of observations fall into each terminal node for each time series. The overall transformation steps are demonstrated in Figure 1 and $\bar{\Psi}_{N \times r}$ is represented as follows:

$$\bar{\Psi}_{N \times r} = \begin{pmatrix} \sum_{j=1}^T \phi_j^{(1)} \\ \sum_{j=1}^T \phi_j^{(2)} \\ \vdots \\ \sum_{j=1}^T \phi_j^{(N)} \end{pmatrix} \quad (3)$$

3.2. Ensemble

In the previous section, we have demonstrate the representation learning procedure on a single tree setting, however, Rand-TS utilizes an ensemble of trees to obtain final representation. Here, we extend single tree representation to multiple tree representation by a simple vector concatenation. Assume that we construct J number of random trees and let $r_1, \dots, r_J$ be the number of terminal nodes for each tree. By concatenating $\bar{\Psi}_{N \times r_1}, \bar{\Psi}_{N \times r_2}, \dots, \bar{\Psi}_{N \times r_J}$, we obtain our final representation $\Theta_{N \times (\sum_{k=1}^J r_k)}$ of dimension $(N \times (\sum_{k=1}^J r_k))$. Figure 2 visualizes an example of a final representation and, the pseudo-code for representation learning procedure is provided in Algorithm 1.

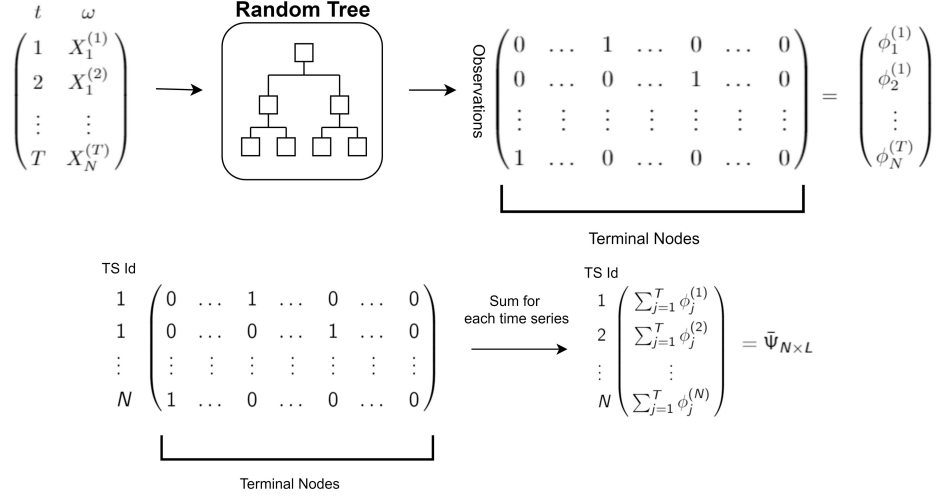


Figure 1: Visual illustration of the representation learning process.

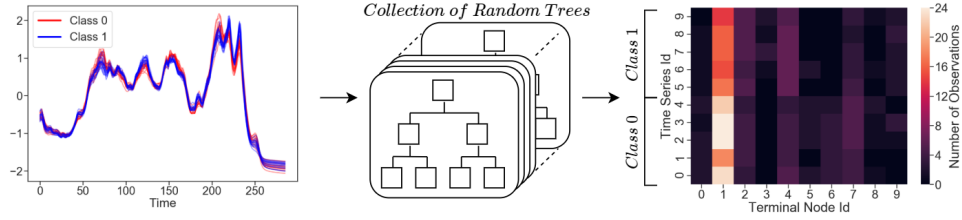


Figure 2: Representation learning procedure with a collection of random trees.

3.3. Time-Series Categorization and Algorithm

In the classification setting, time-series datasets can be categorized into three groups according to discriminatory characteristics of their patterns (*c*): level (*l*), motion (*m*) and by both level and motion (*b*).

In Section 3.1 we demonstrate our representation learning procedure for univariate time-series where we start with an initial representation $\Psi_{NT \times 2}$ containing a feature vector $\omega = (X_1^{(1)}, X_1^{(2)}, \dots, X_N^{(T)})'$ and time information (*t*). However, depending on the characteristics of the time series, our initial representation Ψ can become one of the following matrices:

Algorithm 1 Representation Learning (D, J, d, c)

- 1: Construct initial representation Ψ based on c ,
 - 2: $\Theta = \{\}$
 - 3: **for** J times **do**
 - 4: Construct a random tree on Ψ with depth d .
 - 5: Obtain terminal node representation Φ .
 - 6: Set $\Theta = \Theta \cup \Phi$.
 - 7: **Return** Θ .
-

$$\begin{array}{ccc}
 \begin{array}{cc} t & \omega \\ \left(\begin{array}{c} 1 \\ 2 \\ \vdots \\ T \end{array} \right. & \left. \begin{array}{c} X_1^{(1)} \\ X_1^{(2)} \\ \vdots \\ X_N^{(T)} \end{array} \right) \\ c = l \end{array} &
 \begin{array}{cc} t & \omega \\ \left(\begin{array}{c} 1 \\ 2 \\ \vdots \\ T-1 \end{array} \right. & \left. \begin{array}{c} X_1^{(2)} - X_1^{(1)} \\ X_1^{(3)} - X_1^{(2)} \\ \vdots \\ X_N^{(T)} - X_N^{(T-1)} \end{array} \right) \\ c = m \end{array} &
 \begin{array}{cc} t & \omega \\ \left(\begin{array}{c} 1 \\ 2 \\ \vdots \\ T-1 \end{array} \right. & \left. \begin{array}{c} X_1^{(1)} \quad X_1^{(2)} - X_1^{(1)} \\ X_1^{(2)} \quad X_1^{(3)} - X_1^{(2)} \\ \vdots \quad \vdots \\ X_N^{(T-1)} \quad X_N^{(T)} - X_N^{(T-1)} \end{array} \right) \\ c = b \end{array}
 \end{array}$$

In our framework, we treat category (c) as a hyperparameter that can take tree values (l, m, b) and tune it in our experiments. However, if there is prior knowledge about the distributional characteristics of the interested time-series database, c could also be set manually.

3.4. Similarity Based Classification

Measuring the similarity between two time series using conventional metrics have major drawbacks: inability to detect pattern shifts and extensions. For a given hyperparameter setting (c, J, d) our representation learning algorithm transforms given time-series database into a final representation such that it can detect pattern shifts and extensions (demonstrated in Section 5.5). This yields the most important benefit of Rand-TS by allowing the usage of conventional metrics to measure the similarity without sacrificing any information. Here we utilize k-NN with Manhattan distance ($l_1 - norm$) as the classification scheme and denote the dissimilarity matrix as S where $S_{ij} = \|\sum_{j=1}^T \phi_j^{(i)} - \sum_{j=1}^T \phi_j^{(j)}\|_1$ for $i, j = 1, \dots, N$.

3.5. Parameters

Rand-TS contains three parameters: number of random trees (n), depth of a random tree (d) and time-series category (c). n and d parameters very intuitive and simple to interpret since they are among the default parameters of a regular decision tree. Moreover, time-series category (c) is explained in Section 3.3.

Our representation framework is robust in terms of parameter selection. If there is no concern with the computational time, n can be selected as large as possible whereas in a regular case as long as it is selected large enough, it does

not require parameter tuning. c can only take one of the tree values (l, m, b) therefore, the selection of this parameter only takes a small effort, moreover, it can also be specified manually with prior knowledge about the dataset of interest.

Tree depth (d) is a very important parameter that needs to be tuned carefully. Having a small d causes significant information lost whereas a large d can increase dimension of the representation significantly and introduce noise to your density estimation. In order to illustrate the effect of d on the learned representation and to prove that the learned representation captures the essential characteristics of the given time series, we provide an example time series from the Gun Point dataset. We construct 200 random trees with a range of depth values $d \in \{2, 3, 4, 5\}$. Then, we obtain a terminal node representation for each observation in the time series and reconstruct the time-series (reduce the dimension into 1-D) by applying PCA. Figure 3 shows that when depth is too small, reconstructed representation fails to represent original time series. On the other hand, if depth is too large, then additional noise is introduced to the representation.

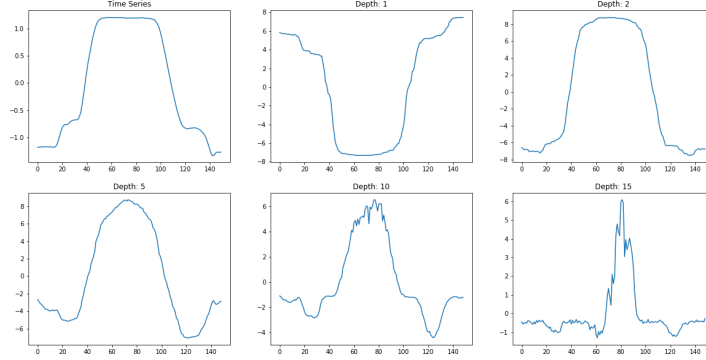


Figure 3: Illustration of the representations learned by the randomized tree transformation with various depth where the learned representations are visualized in 1-D using PCA.

3.6. Feature Selection

In general, random-tree transformation provides vast benefits in terms of flexibility and performance. However, due to the random nature of decision trees, there is no guarantee on the quality of the information provided by a single tree. Usually, this problem can be easily addressed by using many randomized trees since they are computationally very efficient to construct. However, as the number of randomized trees increases, the dimension of the final representation also increases rapidly which, as a consequence, increases the complexity of the distance calculations even though the final representation is very sparse. Moreover, features obtained from randomly constructed terminal nodes can contain meaningless information and introduce noise into the distance calculations which in return potentially reduces the classification performance.

Based on the observations above, it is possible to improve Rand-TS in terms of classification and computational performance by eliminating unnecessary/poor features. In this work, we utilize two types of feature selection methods: unsupervised and supervised. The unsupervised feature selection method eliminates features that have less variance than a given percentile of the feature variances whereas the supervised feature selection method eliminates the features that do not have an effect on the classification performance based on a supervised learner trained on the final representation.

3.7. Complexity Analysis

In this section, we conduct theoretical worst-case analysis on both training and testing complexity. The training complexity of the suggested framework is dictated by the representation learning method which is based on the construction of the randomized-trees. For a given dataset with u number of observations, constructing a randomized-tree of depth d yields a worst-case complexity of $O(ud)$. Since our initial representation contains $u = NT$ rows, constructing a single randomized tree has the time complexity of $O(NTd)$ and consequently constructing J trees has the time complexity of $O(JNTd)$.

The testing complexity depends both on the complexity of obtaining a new representation and the complexity of the classification. Theoretically, constructing a randomized-tree and transforming a given time series based on the constructed tree has the same complexity. Therefore, transforming a given test instance yields a time complexity of $O(NTd)$. In the classification phase, we are using a nearest-neighbor classifier with $l_1 - norm$ as the distance measure. Therefore, the complexity of the classification equals to nothing but N times $l_1 - norm$ calculations. The time complexity of $l_1 - norm$ calculations between two $1 - D$ vectors equal to the length of the vectors which in our case equals to the total number of terminal nodes. In the worst case, there are 2^d terminal nodes in a tree with depth d . Therefore, the time complexity of the classification is $O(JN2^d + NTd)$.

Although theoretically, the worst-case complexity of the classification phase is exponential in the tree depth, in practice, the suggested framework works very efficiently. As explained in Section 3.6, the final representation is very sparse and the dimension of the final representation can be reduced drastically without any performance loss. This dimensionality reduction translates into faster distance calculations which improves efficiency. In addition, utilization of sparse architectures allows faster computations as well as efficiency in memory storage. Moreover, the suggested methods are very convenient for parallel processing and parallelization can be applied to random-tree construction as well as distance calculations.

4. Modeling the time-series using mixture of time-varying Gaussian distributions

Symbolic aggregate approximation (SAX) [26, 28, 38] is one of the most popular and widely used methods for time-series discretization. SAX relies on the

Table 1: Characteristics of the synthetic datasets.

Type	Length	Distribution	Mean	Std	Class
Mean	100	$N(30,10)$	30	10	1
	100	$N(32,10)$	32	10	2
Variance	100	$N(30,10)$	30	10	1
	100	$N(30,12)$	30	12	2
Time Varying	100	$N(0.594t,10)$	30	10	1
Mean	100	$N(0.495t+5,10)$	30	10	2
Time Varying	100	$N(30,0.175t)$	30	10	1
Variance	100	$N(30,0.12t+3.5)$	30	10	2

argument of normalized time-series being distributed with Gaussian distribution and determines the breakpoints using quantiles corresponding to equal-sized probabilities. In this sense, SAX essentially models a time-series as a mixture of finitely many Gaussian distributions.

Rand-TS utilizes random trees to split the given time-series using both the observations and the time information. Using a similar argument as SAX, we claim that the observations in a terminal node can be thought to be coming from a gaussian distribution and a combination of all terminal nodes corresponds to a mixture of Gaussian distributions. However, unlike SAX, the suggested method also models time-varying Gaussian distribution since it utilizes the time information.

In order to demonstrate the effectiveness of Rand-TS on the modeling of time-varying Gaussian mixtures, we conducted a numerical experiment where we generate relatively difficult time-series datasets with varying characteristics and compare the classification accuracy of Rand-TS and SAX bag-of-words (SAX-BoW) where the discretization thresholds are determined from unnormalized time-series. Characteristics of different datasets are provided in Table 1.

Both methods are evaluated using a one nearest-neighbour classifier and best leave-one-out cross-validation accuracy performance is reported. For Rand-TS we set $m = l$ and $J = 500$ and select $d \in \{2, 3, 4, 5, 6, 7\}$, similarly for SAX-BoW we select alphabet size $\in \{3, 4, 5, 6, 7\}$ and window size $\in \{3, 4, 5, 6, 7, 8\}$.

Table 2 illustrates the classification accuracies of each method for the datasets with different characteristics. It can be observed from the results that Rand-TS outperforms SAX-BoW in both stationary and time-varying settings and successfully models the time-series as a mixture of time-varying Gaussian distributions.

5. Computational Experiments

In this section, we provide a set of experiments for evaluating the performance of our suggested framework in various settings. Firstly, we illustrate the

Table 2: Classification accuracy of Rand-TS and SAX-BOW on the synthetic datasets with different characteristics.

Type	Rand-TS	SAX-BoW
Mean	0.806	0.740
Variance	0.833	0.672
Time Varying Mean	0.865	0.739
Time Varying Variance	0.970	0.927

classification performance of Rand-TS on 85 univariate time series and compare them with the state-of-the-art methods in the literature. Secondly, we demonstrate the computational complexity of our method with respect to different time-series characteristics. Then, we provide an example on the performance improvement with additional features. We explain the adaptation of Rand-TS in the multivariate setting and illustrate the performance in 19 multivariate datasets. Moreover, we focus on time series with varying length/missing data and, illustrate the performance of Rand-TS on 15 datasets. Lastly, we illustrate the interpretability of Rand-TS on the Coffee dataset.

As suggested by [22], the performance of Rand-TS on both univariate and multivariate settings are evaluated based on the accuracy of a one nearest-neighbor (1-NN) classifier. The experimental setup involves hyperparameter tuning via leave-one-out (LOO) cross-validation for tree depth ($d \in \{4, 6, 8, 10\}$) and dataset category ($c \in \{l, m, b\}$) on the training set and evaluating classification performance on the test set. As explained in Section 3.5, the number of trees (J) is very robust and as long as there are enough trees, it is not required to be tuned. Therefore J is set to 100 trees in parameter tuning and 500 trees in testing.

5.1. Classification Accuracy

Table 3 illustrates the median classification accuracy (test) of Rand-TS over 20 replications on 85 univariate time-series datasets in comparison with the state-of-the-art distance-based methods. The statistical comparison between Rand-TS, Rand-TS with unsupervised feature selection (Rand-TS-U), Rand-TS with supervised feature selection (Rand-TS-S) and the 12 state-of-the-art methods is done by utilizing Friedman-Nemenyi test ([12], [32]) as suggested in [9]. Figure 4 illustrates the average ranks for all competing classifiers with critical differences. We can observe that Rand-TS-S has the best rank, Rand-TS-U has the third-best and Rand-TS has the fourth-best rank. Moreover, based on the Friedman test, there is a significant difference between the 15 classifiers at the 0.05 level. Classification accuracy of Rand-TS-S is not significantly different from LPS, MSM and LCSS, however, it is significantly different from the remaining classifiers at 0.05 level. In addition we provide a more detailed visualization of the pairwise comparison of the classification accuracies in Figures 5, 6 and 7. In these figures, each axis represents a method and each point represents a classification accuracy of a specific method. The red line is the $x=y$ line

that shows the region that both methods have the same accuracy, the triangular regions above and below the red line represents the regions that one of the classifiers outperforms the other. Moreover, Table 4 summarizes the average ranks of 7 classifiers that are not significantly different from Rand-TS-S at 0.05 significance level for different problem types. Here Rand-TS-S outperforms other classifiers in DEVICE, ECG, IMAGE, SENSOR, SIMULATED and SPECTRO categories.

Table 3: Classification accuracy of Rand-TS and 12 similarity-based classification methods on 85 univariate time-series datasets from the UCR database.

	Rand-TS			DTW			DDTW			ERP	LCSS	MSM	TWE	LPS	ED
	S-FS	U-FS	Full	R1	Rn	Weight	R1	Rn	Weight						
Adiac □	0.803	0.806	0.788	0.604	0.609	0.606	0.586	0.670	0.675	0.609	0.251	0.627	0.634	0.770	0.611
ArrowHead □	0.806	0.829	0.820	0.703	0.800	0.817	0.777	0.834	0.806	0.800	0.789	0.806	0.794	0.783	0.800
Beef ◇	0.833	0.800	0.800	0.633	0.667	0.700	0.667	0.700	0.700	0.667	0.767	0.467	0.600	0.600	0.667
BeetleFly □	0.800	0.700	0.700	0.700	0.650	0.700	0.750	0.850	0.750	0.750	0.800	0.600	0.700	0.800	0.750
BirdChicken □	0.900	0.850	0.850	0.750	0.700	0.750	0.850	0.900	0.850	0.750	0.800	0.850	0.850	1.000	0.550
CBF ♣	1.000	0.987	0.987	0.997	0.994	0.997	0.592	0.572	0.591	0.998	0.990	0.969	0.991	0.999	0.852
Car △	0.925	0.900	0.900	0.733	0.767	0.783	0.733	0.783	0.783	0.767	0.833	0.900	0.917	0.850	0.733
ChlorineConcentration △	0.712	0.693	0.695	0.648	0.650	0.649	0.683	0.698	0.690	0.660	0.720	0.628	0.648	0.608	0.650
CinCECGtorso	0.917	0.905	0.905	0.651	0.930	0.859	0.711	0.939	0.910	0.888	0.946	0.962	0.846	0.736	0.897
Coffee ◇	0.964	0.929	0.964	1.000	1.000	1.000	0.929	0.964	0.964	1.000	1.000	0.893	1.000	1.000	1.000
Computers ♥	0.726	0.720	0.712	0.700	0.624	0.700	0.668	0.676	0.608	0.540	0.644	0.576	0.632	0.680	0.576
CricketX ♠	0.700	0.700	0.700	0.754	0.779	0.790	0.631	0.631	0.641	0.764	0.731	0.764	0.749	0.697	0.577
CricketY ♠	0.753	0.751	0.754	0.744	0.756	0.759	0.559	0.562	0.582	0.731	0.787	0.772	0.754	0.767	0.567
CricketZ ♠	0.735	0.733	0.733	0.754	0.736	0.762	0.556	0.605	0.618	0.733	0.741	0.764	0.736	0.754	0.587
DiatomSizeReduction □	0.948	0.931	0.948	0.967	0.935	0.967	0.935	0.915	0.935	0.935	0.941	0.948	0.948	0.905	0.935
DistalPhalanxOutlineAgeGroup □	0.815	0.754	0.754	0.770	0.626	0.698	0.676	0.691	0.640	0.676	0.734	0.741	0.712	0.669	0.626
DistalPhalanxOutlineCorrect □	0.782	0.782	0.775	0.717	0.725	0.717	0.732	0.714	0.710	0.710	0.743	0.736	0.739	0.721	0.717
DistalPhalanxTW □	0.782	0.748	0.748	0.590	0.633	0.604	0.590	0.590	0.590	0.669	0.633	0.640	0.633	0.568	0.633
ECG200 ●	0.870	0.890	0.880	0.770	0.880	0.880	0.830	0.890	0.880	0.880	0.890	0.910	0.890	0.860	0.880
ECG5000 ●	0.938	0.924	0.925	0.924	0.925	0.925	0.920	0.921	0.922	0.932	0.930	0.932	0.927	0.917	0.925
ECGFiveDays ●	0.928	0.926	0.915	0.768	0.797	0.796	0.686	0.696	0.692	0.806	0.794	0.891	0.829	0.879	0.797
Earthquakes △	0.745	0.717	0.717	0.719	0.727	0.727	0.676	0.676	0.676	0.683	0.727	0.691	0.676	0.640	0.712
ElectricDevices ♥	0.694	0.693	0.694	0.602	0.631	0.613	0.571	0.597	0.582	0.654	0.626	0.659	0.593	0.681	0.552
FaceAll □	0.774	0.768	0.793	0.808	0.808	0.794	0.873	0.882	0.897	0.793	0.801	0.809	0.786	0.767	0.714
FaceFour □	0.955	0.966	0.920	0.830	0.898	0.875	0.625	0.739	0.716	0.864	0.966	0.943	0.852	0.943	0.784
FacesUCR □	0.918	0.918	0.915	0.905	0.908	0.920	0.843	0.875	0.868	0.921	0.955	0.971	0.920	0.926	0.769
FiftyWords □	0.778	0.776	0.775	0.690	0.765	0.771	0.692	0.763	0.769	0.712	0.798	0.813	0.793	0.818	0.631
Fish □	0.863	0.857	0.857	0.823	0.834	0.851	0.897	0.920	0.960	0.874	0.874	0.931	0.931	0.943	0.783

Continued on next page

Table 3: Classification accuracy of Rand-TS and 12 similarity-based classification methods on 85 univariate time-series datasets from the UCR database.

	Rand-TS			DTW			DDTW			ERP	LCSS	MSM	TWE	LPS	ED
	S-FS	U-FS	Full	R1	Rn	Weight	R1	Rn	Weight						
FordA $\triangle$	0.720	0.698	0.689	0.555	0.665	0.653	0.717	0.722	0.717	0.658	0.698	0.719	0.618	0.873	0.665
FordB $\triangle$	0.693	0.674	0.660	0.620	0.599	0.599	0.665	0.657	0.659	0.625	0.648	0.642	0.609	0.711	0.606
GunPoint $\spadesuit$	0.993	0.993	0.980	0.907	0.913	0.980	0.993	1.000	0.993	0.947	0.973	0.973	0.953	0.993	0.913
Ham $\diamond$	0.562	0.524	0.524	0.467	0.600	0.581	0.476	0.543	0.552	0.600	0.610	0.514	0.514	0.562	0.600
HandOutlines $\square$	0.802	0.790	0.790	0.881	0.878	0.870	0.738	0.811	0.841	0.884	0.849	0.876	0.870	0.881	0.862
Haptics $\spadesuit$	0.403	0.401	0.396	0.377	0.416	0.370	0.286	0.328	0.321	0.370	0.390	0.442	0.416	0.432	0.370
Herring $\square$	0.641	0.516	0.500	0.531	0.531	0.531	0.500	0.516	0.469	0.625	0.562	0.562	0.516	0.578	0.516
InlineSkate $\spadesuit$	0.455	0.461	0.461	0.384	0.387	0.404	0.453	0.456	0.453	0.447	0.427	0.453	0.422	0.500	0.342
InsectWingbeatSound $\triangle$	0.581	0.583	0.581	0.355	0.574	0.574	0.243	0.519	0.542	0.526	0.556	0.575	0.534	0.551	0.562
ItalyPowerDemand $\triangle$	0.945	0.935	0.942	0.950	0.955	0.950	0.914	0.973	0.941	0.955	0.957	0.944	0.948	0.923	0.955
LargeKitchenAppliances $\heartsuit$	0.704	0.705	0.705	0.795	0.795	0.795	0.768	0.757	0.768	0.619	0.584	0.669	0.752	0.717	0.493
Lightning2 $\triangle$	0.754	0.787	0.787	0.869	0.869	0.902	0.672	0.770	0.820	0.869	0.770	0.820	0.836	0.820	0.754
Lightning7 $\triangle$	0.685	0.699	0.705	0.726	0.712	0.767	0.575	0.699	0.685	0.740	0.575	0.753	0.753	0.740	0.575
Mallat $\clubsuit$	0.924	0.917	0.915	0.934	0.914	0.938	0.897	0.925	0.913	0.914	0.911	0.932	0.912	0.908	0.914
Meat $\diamond$	0.933	0.917	0.917	0.933	0.933	0.933	0.667	0.667	0.667	0.933	0.533	0.933	0.933	0.883	0.933
MedicalImages $\square$	0.749	0.749	0.746	0.737	0.747	0.737	0.663	0.663	0.682	0.676	0.707	0.741	0.711	0.746	0.684
MiddlePhalanxOutlineAgeGroup $\square$	0.778	0.748	0.744	0.500	0.519	0.519	0.558	0.578	0.539	0.481	0.610	0.494	0.519	0.487	0.519
MiddlePhalanxOutlineCorrect $\square$	0.743	0.752	0.746	0.698	0.766	0.753	0.691	0.777	0.746	0.766	0.749	0.753	0.763	0.773	0.766
MiddlePhalanxTW $\square$	0.617	0.583	0.589	0.506	0.506	0.513	0.500	0.500	0.532	0.500	0.578	0.494	0.487	0.526	0.513
MoteStrain $\triangle$	0.934	0.940	0.928	0.835	0.866	0.859	0.716	0.775	0.788	0.872	0.882	0.855	0.797	0.922	0.879
NonInvasiveFatalECGThorax1 $\bullet$	0.863	0.863	0.863	0.790	0.829	0.816	0.697	0.723	0.730	0.825	0.812	0.816	0.820	0.812	0.829
NonInvasiveFatalECGThorax2 $\bullet$	0.891	0.891	0.891	0.865	0.870	0.884	0.826	0.826	0.861	0.895	0.883	0.883	0.888	0.841	0.880
OSULeaf $\square$	0.719	0.717	0.717	0.591	0.599	0.624	0.880	0.872	0.884	0.603	0.789	0.773	0.777	0.740	0.521
OliveOil $\diamond$	0.867	0.833	0.833	0.833	0.867	0.833	0.867	0.833	0.867	0.867	0.400	0.833	0.867	0.867	0.867
PhalangesOutlinesCorrect $\square$	0.753	0.770	0.769	0.728	0.761	0.747	0.740	0.791	0.776	0.760	0.742	0.752	0.760	0.756	0.761
Phoneme $\triangle$	0.248	0.251	0.251	0.228	0.227	0.161	0.256	0.247	0.245	0.179	0.256	0.292	0.270	0.237	0.109
Plane $\triangle$	1.000	0.990	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.962
ProximalPhalanxOutlineAgeGroup $\square$	0.839	0.790	0.800	0.805	0.785	0.805	0.820	0.810	0.820	0.795	0.854	0.790	0.805	0.795	0.785
ProximalPhalanxOutlineCorrect $\square$	0.828	0.832	0.832	0.784	0.790	0.784	0.818	0.852	0.842	0.811	0.756	0.787	0.808	0.842	0.808

Continued on next page

Table 3: Classification accuracy of Rand-TS and 12 similarity-based classification methods on 85 univariate time-series datasets from the UCR database.

	Rand-TS			DTW			DDTW			ERP	LCSS	MSM	TWE	LPS	ED
	S-FS	U-FS	Full	R1	Rn	Weight	R1	Rn	Weight						
ProximalPhalanxTW $\square$	0.792	0.722	0.716	0.761	0.761	0.751	0.751	0.737	0.746	0.702	0.805	0.756	0.746	0.732	0.707
RefrigerationDevices $\heartsuit$	0.516	0.508	0.508	0.464	0.440	0.427	0.408	0.424	0.435	0.491	0.491	0.483	0.512	0.459	0.395
ScreenType $\heartsuit$	0.496	0.496	0.496	0.397	0.411	0.411	0.427	0.443	0.427	0.381	0.437	0.459	0.427	0.416	0.360
ShapeletSim $\clubsuit$	0.528	0.911	0.911	0.650	0.694	0.756	0.539	0.506	0.494	0.839	0.822	0.867	0.578	0.867	0.539
ShapesAll $\square$	0.858	0.857	0.857	0.768	0.802	0.812	0.835	0.845	0.860	0.812	0.845	0.872	0.860	0.873	0.752
SmallKitchenAppliances $\heartsuit$	0.692	0.687	0.687	0.643	0.672	0.675	0.653	0.669	0.645	0.632	0.472	0.699	0.645	0.712	0.344
SonyAIBORobotSurface1 $\triangle$	0.743	0.834	0.765	0.725	0.696	0.737	0.730	0.699	0.717	0.689	0.735	0.730	0.681	0.774	0.696
SonyAIBORobotSurface2 $\triangle$	0.814	0.834	0.834	0.831	0.859	0.831	0.858	0.861	0.858	0.823	0.811	0.871	0.853	0.872	0.859
StarlightCurves $\triangle$	0.965	0.964	0.964	0.907	0.898	0.895	0.962	0.964	0.962	0.862	0.879	0.868	0.883	0.963	0.849
Strawberry $\diamond$	0.959	0.954	0.955	0.941	0.946	0.943	0.962	0.962	0.962	0.946	0.838	0.943	0.943	0.962	0.946
SwedishLeaf $\square$	0.917	0.924	0.924	0.792	0.846	0.874	0.885	0.904	0.893	0.862	0.888	0.896	0.891	0.920	0.789
Symbols $\square$	0.951	0.951	0.950	0.950	0.938	0.950	0.971	0.969	0.974	0.921	0.955	0.949	0.960	0.963	0.899
SyntheticControl $\clubsuit$	0.983	0.982	0.977	0.993	0.983	0.993	0.567	0.567	0.567	0.977	0.953	0.973	0.987	0.980	0.880
ToeSegmentation1 $\spadesuit$	0.798	0.800	0.798	0.772	0.750	0.794	0.785	0.588	0.627	0.833	0.750	0.816	0.820	0.877	0.680
ToeSegmentation2 $\spadesuit$	0.908	0.796	0.796	0.838	0.908	0.892	0.685	0.785	0.792	0.923	0.908	0.754	0.785	0.869	0.808
Trace $\triangle$	0.980	0.930	0.930	1.000	0.990	1.000	1.000	0.990	1.000	0.950	0.970	0.930	0.990	0.980	0.760
TwoLeadECG $\bullet$	0.867	0.867	0.929	0.905	0.868	0.905	0.995	0.982	0.995	0.942	0.886	0.947	0.974	0.948	0.747
TwoPatterns $\clubsuit$	0.979	0.906	0.905	1.000	0.998	1.000	0.998	0.997	0.998	1.000	0.999	0.999	0.998	0.982	0.907
UWaveGestureLibraryAll $\spadesuit$	0.968	0.968	0.968	0.892	0.962	0.966	0.849	0.927	0.937	0.954	0.959	0.963	0.953	0.966	0.948
UWaveGestureLibraryX $\spadesuit$	0.789	0.784	0.784	0.728	0.774	0.774	0.677	0.741	0.744	0.771	0.771	0.769	0.771	0.829	0.739
UWaveGestureLibraryY $\spadesuit$	0.718	0.717	0.717	0.634	0.702	0.693	0.584	0.663	0.659	0.679	0.692	0.702	0.685	0.761	0.662
UWaveGestureLibraryZ $\spadesuit$	0.713	0.708	0.709	0.658	0.675	0.676	0.577	0.677	0.679	0.687	0.689	0.700	0.684	0.768	0.650
Wafer $\triangle$	1.000	1.000	1.000	0.980	0.996	0.997	0.978	0.997	0.997	0.995	0.990	0.997	0.996	0.997	0.995
Wine $\diamond$	0.574	0.519	0.463	0.574	0.611	0.574	0.519	0.537	0.519	0.611	0.481	0.593	0.574	0.630	0.611
WordSynonyms $\square$	0.748	0.746	0.743	0.649	0.749	0.745	0.680	0.732	0.732	0.682	0.757	0.763	0.749	0.755	0.618
Worms $\spadesuit$	0.475	0.486	0.483	0.584	0.532	0.532	0.584	0.649	0.584	0.558	0.597	0.571	0.571	0.701	0.455
WormsTwoClass $\spadesuit$	0.602	0.572	0.572	0.623	0.584	0.571	0.636	0.688	0.636	0.649	0.649	0.675	0.610	0.753	0.610
Yoga $\square$	0.865	0.863	0.862	0.837	0.843	0.853	0.820	0.829	0.840	0.847	0.860	0.865	0.872	0.869	0.830

Continued on next page

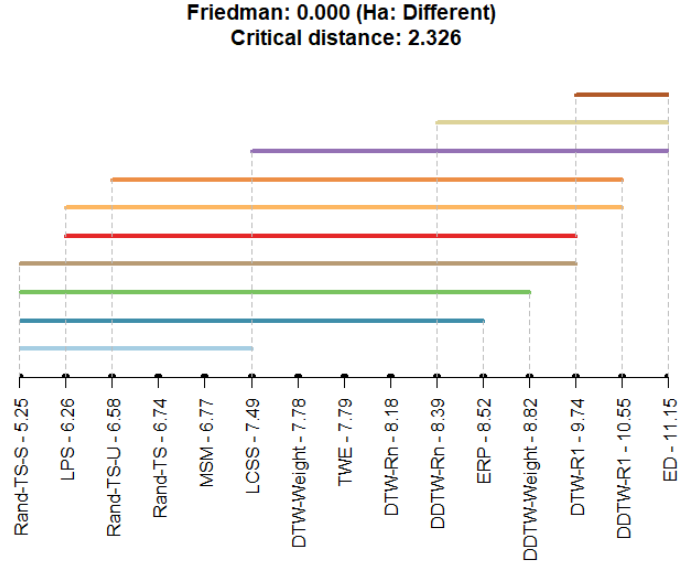


Figure 4: The average ranks for all competing classifiers on UCR univariate datasets. The critical differences at 0.05 is 2.326.

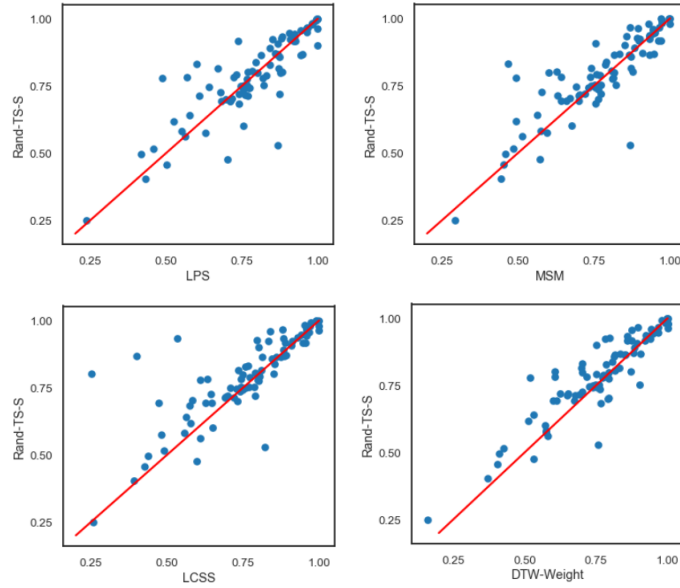


Figure 5: Classification accuracy for Rand-TS-S versus LPS, MSM, LCSS and WDTW.

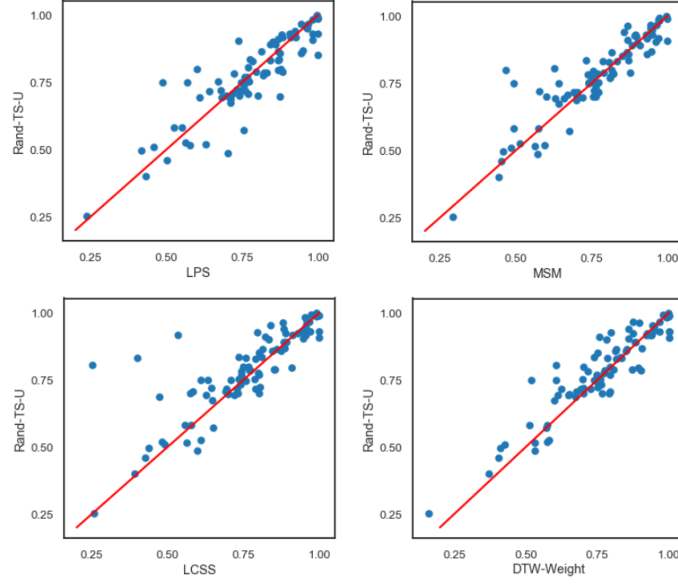


Figure 6: Classification accuracy for Rand-TS-U versus LPS, MSM, LCSS and WDTW.

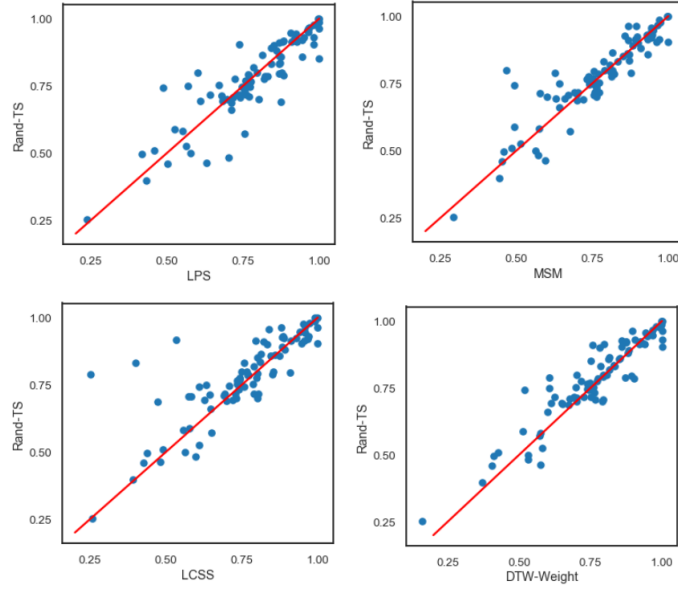


Figure 7: Classification accuracy for Rand-TS versus LPS, MSM, LCSS and WDTW.

5.2. Computational Complexity

In this section, we demonstrate the effect of the number of randomized trees J , tree depth d , time-series length T and size of the training set N on the

Table 4: Average ranks of 15 similarity-based classifiers for the 85 datasets by problem category

	Rand-TS			LCSS	MSM	LPS
	S-FS	U-FS	Full			
DEVICE	1.833	2.333	2.167	5.333	4.500	3.667
ECG	2.429	2.857	2.857	3.714	2.571	5.000
IMAGE	2.655	3.207	3.759	3.724	3.448	3.207
MOTION	3.000	3.429	3.643	4.143	3.500	2.000
SENSOR	2.400	2.867	2.933	4.333	3.600	3.000
SIMULATED	2.667	3.167	3.667	3.500	3.500	3.833
SOUND	5.000	3.000	3.000	2.000	1.000	6.000
SPECTRO	1.857	3.571	3.429	4.143	4.143	2.286

testing time using the Yoga dataset from [1]. The experimental setting involves the measurement of the testing times under following parameter settings: $\alpha_T \in \{0.2, 0.4, 0.6, 0.8, 1\}$, $\alpha_N \in \{0.2, 0.4, 0.6, 0.8, 1\}$, $d \in \{6, 7, 8, 9, 10, 11, 12\}$ and $J \in \{50, 100, 200, 300, 400, 500\}$. For each combination, average testing times are reported out of 10 replications. Rand-TS is implemented in Python 3.6 and is available in [16]. The experiments are conducted in a Windows 10 operating system with 16 GB RAM and Intel(R) Core(TM) i7-7700HQ CPU.

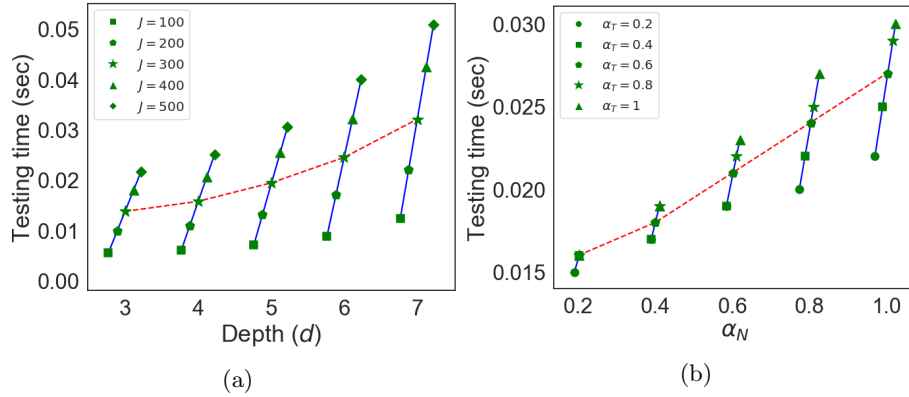


Figure 8: Test times with respect to different values of the J , N , T and d .

In theory, the worst-case complexity of the testing time is exponential in d . However, the Figure 8a shows that, in the empirical setting, as d increases, testing time does not show an exponential increase but more like a $d \log(d)$ increase. Moreover, Figure 8b shows that the testing time increases linearly with both $T\alpha_T$ and $N\alpha_N$ which is consistent with the theoretical complexity analysis. Lastly, from the testing times, it can be seen that the Rand-TS works very efficiently.

5.3. Performance Improvement with Additional Features

Rand-TS has a very flexible structure that allows incorporating custom features with the time-series observations. As illustrated in Section 3.3, ω in the initial representation can be an $NT \times 1$ vector containing the time-series observations as well as an $NT \times 2$ matrix that contains time-series observations with their first-order differences. In addition, it is also possible to extend ω matrix by adding custom features columns which potentially improve the accuracy of the classification task.

In order to illustrate this, we conduct an experiment with the Earthquake dataset [1]. The Earthquake dataset contains 461 readings seismic wave readings collected by the sensors and, the aim is to identify whether an earthquake occurred in a given seismic wave or not. In this experiment, we set the representation learning parameters $n = 500$, $d = 4$ and $m = l$ as our base setting and experimented by incorporating simple 'min' information into the initial representation. For clarity, the initial representation with 'min' is illustrated below:

$$\begin{matrix} t & \omega \\ \left(\begin{array}{ccc} 1 & X_1^{(1)} & \min\{X_1^{(1)}, X_1^{(2)}, \dots, X_1^{(T)}\} \\ 2 & X_1^{(2)} & \min\{X_1^{(1)}, X_1^{(2)}, \dots, X_1^{(T)}\} \\ \vdots & \vdots & \vdots \\ T & X_N^{(T)} & \min\{X_N^{(1)}, X_N^{(2)}, \dots, X_N^{(T)}\} \end{array} \right) \end{matrix} \quad (4)$$

The results show that, with the given base parameter setting, the method yields the classification accuracy of 74.14%. Whereas, by adding 'min' information into the initial representation, it is possible to increase the classification accuracy to 76.7%.

5.4. Multivariate Time Series

Multivariate time series (MTS) classification is a challenging task. The trivial approach in MTS classification is measuring the similarity for each attributes separately and using the weighted average of these similarities as the final similarity measure. This approach is usually unsuccessful because in MTS classification the relations between different attributes carry vast importance which can not be captured. Moreover, different attributes can have different types (e.g. numeric, categorical) and different lengths, which makes the similarity measurement even more difficult.

Similar to the univariate setting, Rand-TS transforms given MTS into a 1-D representation. Since the representation learning phase of the Rand-TS relies on the randomized-trees, it can model the relations between different attributes conveniently. Moreover, tree-based methods can easily handle numeric and categorical attributes, and thanks to our initial representation format Rand-TS is not effected by length differences as well.

Rand-TS can be adapted to the MTS setting by only updating the initial representation. Let

$$Y_i = \begin{pmatrix} Y_i^1(1) & \dots & Y_i^1(T) \\ Y_i^2(1) & \dots & Y_i^2(T) \\ \vdots & \vdots & \vdots \\ Y_i^m(1) & \dots & Y_i^m(T) \end{pmatrix}$$

be a multivariate time-series of length T with m attributes. Then, the initial representation for the MTS is provided as follows:

$$\Psi_{NT \times (m+1)} = \begin{matrix} & \begin{matrix} t & \omega \end{matrix} \\ \begin{pmatrix} 1 & Y_1^1(1) & Y_1^2(1) & \dots & Y_1^m(1) \\ 2 & Y_1^1(2) & Y_1^2(2) & \dots & Y_1^m(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ T & Y_N^1(T) & Y_N^2(T) & \dots & Y_N^m(T) \end{pmatrix} \end{matrix} \quad (5)$$

We illustrate the performance of Rand-TS on 19 MTS datasets from UCR database [1]. According to Table 5, Rand-TS provides comparable performance to the DTW in the multivariate setting.

5.5. Interpretability

Interpretability is an important quality for a classifier. It is very useful to understand the distinguishing characteristics of each class and to identify why a certain time series belongs to a particular class.

The learned representation is a matrix that represents the number of observations that fall into each terminal node for all given time-series. Therefore, each feature in the final representation corresponds to a terminal node that contains many observations satisfying a set of decision rules. This implies that if we can measure the importance of these features in the classification, we can actually identify the reasons behind the classification decision. Here we provide an example for illustrating the interpretability of the model on the Coffee dataset [1]. It can be observed from Figure 9a that the number of observations that fall into the terminal node 1 shows a significant difference for class 0 and class 1 time-series. Figure 9b visualizes the time series in the Coffee dataset with the region pointed out by the terminal node 1 is marked. It can be observed from the Figure 9b that class 1 time series show higher volatility than the class 0 time series in the marked region which causes terminal node 1 to contain more observations from the class 0 time series.

Moreover, since these terminal nodes contain a set of observations from the original time series, it is also possible to construct small interpretable subsequences called shapelets. In the literature, most of the shapelet works identify shapelets that do not have any time discontinuities, however, in a classification task, this might not be true. One advantage of Rand-TS is that it can also identify shapelets with time discontinuities as well.

Table 5: Classification accuracy of Rand-TS, DTW and ED methods on 19 multivariate time-series datasets from UCR database.

	Rand-TS			DTW		ED
	S-FS	U_FS	Full	R1	Rn	
ArticularyWordRecognition	0.993	0.993	0.993	0.980	0.987	0.970
AtrialFibrillation	0.067	0.133	0.200	0.267	0.220	0.267
BasicMotions	1.000	1.000	1.000	1.000	0.975	0.676
Cricket	0.944	0.944	0.944	0.986	1.000	0.944
ERing	0.900	0.900	0.933	0.133	0.133	0.133
Epilepsy	0.949	0.949	0.949	0.978	0.964	0.666
EthanolConcentration	0.352	0.350	0.350	0.304	0.323	0.293
FingerMovements	0.510	0.520	0.520	0.520	0.530	0.550
HandMovementDirection	0.277	0.284	0.284	0.306	0.231	0.278
Handwriting	0.309	0.307	0.307	0.316	0.286	0.200
LSST	0.470	0.470	0.470	0.575	0.551	0.456
Libras	0.672	0.667	0.667	0.894	0.870	0.833
NATOPS	0.850	0.842	0.842	0.850	0.833	0.850
PenDigits	0.856	0.851	0.847	0.939	0.977	0.973
RacketSports	0.865	0.895	0.862	0.842	0.803	0.868
SelfRegulationSCP1	0.765	0.778	0.778	0.765	0.775	0.771
SelfRegulationSCP2	0.519	0.519	0.517	0.533	0.539	0.483
StandWalkJump	0.400	0.333	0.333	0.333	0.200	0.200
UWaveGestureLibrary	0.964	0.963	0.963	0.868	0.903	0.881

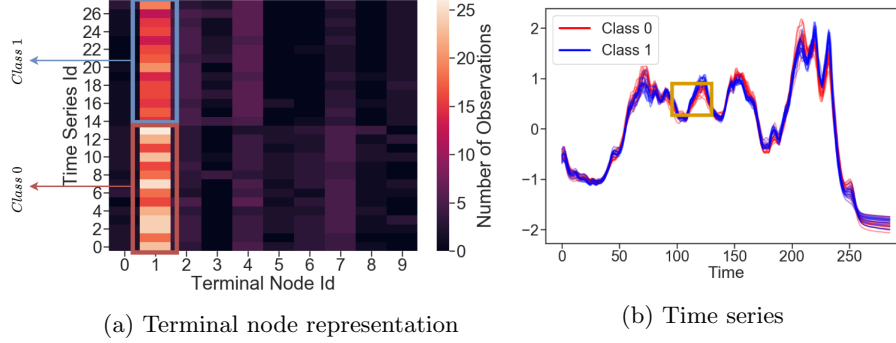


Figure 9: Visual illustration of the terminal node representation and corresponding regions in the time series.

5.6. Time-series with Varying Lengths and Missing Data

Time series data is collected in various domains and, among all these different domains, time series with varying lengths are frequently encountered. According to [41], variations in frequencies and in start/end time of the recordings are the two main reasons for having time series with varying length. In addition, time-series are the set of sequential observations that are usually recorded very frequently. Therefore, the missing values are also commonly seen in the time-series databases.

In practice, a very common way to deal with the missing data problem is by estimating them. However, estimation requires an additional process and introduces additional complexity/parameter to the algorithms. An advantage of the Rand-TS algorithm is that Rand-TS can handle missing data without any additional effort. Utilization of the randomized-trees in representation learning provides this flexibility without introducing any additional parameter or requiring any additional operation. Moreover, Rand-TS handles time series with varying lengths since it utilizes vertical time-series representation. To illustrate the performance of the Rand-TS on the time-series with missing data and varying lengths, we experimented with 15 datasets from [1].

Table 6 illustrates the classification accuracy of Rand-TS on the datasets with varying lengths and/or missing data. According to Table 6 Rand-TS is able to handle time-series with missing data and varying lengths, moreover, it yields a competitive performance.

6. Conclusions

In this work, we introduce Rand-TS, a unified, flexible and interpretable time-series representation learning framework. Rand-TS relies on the density estimation capabilities of decision trees [7] and it extends tree-based density estimation framework to the time-series domain. Rand-TS utilizes random trees to transform given time series dataset into an interpretable and informative

Table 6: Classification accuracy of Rand-TS, ED, and DTW on 15 datasets with varying length and missing data.

	Rand-TS			ED	DTW	
	S-FS	U-FS	Full		Optimal	w=100
AllGestureWiimoteX ♪	0.606	0.594	0.603	0.516	0.717	0.716
AllGestureWiimoteY ♪	0.709	0.698	0.698	0.569	0.730	0.729
AllGestureWiimoteZ ♪	0.606	0.606	0.604	0.454	0.651	0.643
DodgerLoopDay §	0.575	0.562	0.562	0.550	0.588	0.500
DodgerLoopGame §	0.616	0.580	0.562	0.884	0.928	0.877
DodgerLoopWeekend §	0.986	0.986	0.986	0.986	0.978	0.949
GestureMidAirD1 ♪	0.592	0.592	0.592	0.577	0.639	0.569
GestureMidAirD2 ♪	0.600	0.596	0.596	0.492	0.600	0.608
GestureMidAirD3 ♪	0.531	0.527	0.531	0.346	0.377	0.323
GesturePebbleZ1 ♪	0.895	0.884	0.884	0.733	0.826	0.791
GesturePebbleZ2 ♪	0.816	0.810	0.810	0.671	0.779	0.671
MelbournePedestrian §	0.967	0.966	0.966	0.848	0.848	0.791
PLAID ♪	0.711	0.711	0.711	0.560	0.660	0.660
PickupGestureWiimoteZ ♪	0.720	0.740	0.720	0.521	0.814	0.840
ShakeGestureWiimoteZ ♪	0.860	0.860	0.860	0.600	0.840	0.860
Average Rank	2.0	2.7	2.8	5.0	2.7	3.7
♪ - Varying length, § - Missing data						

representation that can be used in supervised and unsupervised learning tasks. We evaluate Rand-TS on the classification task. Our experimental results show that, despite its simplicity, Rand-TS provides competitive performance with the state-of-the-art similarity-based methods in both univariate and multivariate settings. We illustrate that Rand-TS successfully handles time-series with missing data and varying length. Moreover, we show flexibility and interpretability of our framework by conducting additional experiments.

Acknowledgments: This research is supported by Bogazici University Research Fund (Grant number: BAP 14A03SUP6) and Air Force Office of Scientific Research Grant FA9550-17-1-0138.

- [1] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, May 2017.
- [2] M. Baydogan and G. Runger. Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery*, 29(2):400–422, 2015.
- [3] M. G. Baydogan and G. Runger. Time series representation and similar-

- ity based on local autopatterns. *Data Mining and Knowledge Discovery*, 30(2):476–509, 2016.
- [4] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, MA, 1984.
 - [5] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228, June 2002.
 - [6] L. Chen, M. T. Özsü, and V. Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’05, pages 491–502, New York, NY, USA, 2005. ACM.
 - [7] A. Criminisi, J. Shotton, E. Konukoglu, et al. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends® in Computer Graphics and Vision*, 7(2–3):81–227, 2012.
 - [8] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.
 - [9] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.
 - [10] H. Deng, G. Runger, E. Tuv, and M. Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013.
 - [11] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. *Fast subsequence matching in time-series databases*, volume 23. ACM, 1994.
 - [12] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):pp. 86–92, 1940.
 - [13] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.
 - [14] A. Gionis and H. Mannila. Finding recurrent sources in sequences. In *Proceedings of the seventh annual international conference on Research in computational molecular biology*, pages 123–130. ACM, 2003.
 - [15] J. Grabocka, B. Schiling, M. Wistuba, and L. Schmidt-Thieme. Learning time-series shapelets. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–401, 2014.
 - [16] B. Görgülü and M. G. Baydoğan. Rand-TS Implementation in Python. <https://github.com/gorguluberk/Rand-TS>, 2020.

- [17] B. Hidasi and C. Gaspar-Papanek. ShiftTree: An Interpretable Model-Based Approach for Time Series Classification. In *Machine Learning and Knowledge Discovery in Databases*, volume 6912 of *Lecture Notes in Computer Science*, pages 48–64. Springer Berlin / Heidelberg, 2011.
- [18] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014.
- [19] Y.-S. Jeong, M. K. Jeong, and O. A. Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognition*, 44(9):2231 – 2240, 2011. Computer Analysis of Images and Patterns.
- [20] I. Karlsson, P. Papapetrou, and H. Boström. Generalized random shapelet forests. *Data Mining and Knowledge Discovery*, 30(5):1053–1085, 2016.
- [21] E. Keogh. Efficiently finding arbitrarily scaled patterns in massive time series databases. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 253–265. Springer, 2003.
- [22] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.
- [23] E. J. Keogh and M. J. Pazzani. Derivative dynamic time warping. In *SDM*, volume 1, pages 5–7. SIAM, 2001.
- [24] L. Latecki, V. Megalooikonomou, Q. Wang, R. Lakaemper, C. Ratanamahatana, and E. Keogh. Partial elastic matching of time series. In *Data Mining, Fifth IEEE International Conference on*, pages 701–704, 2005.
- [25] 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*, pages 2–11. ACM Press, 2003.
- [26] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15:107–144, 2007.
- [27] J. Lin, R. Khade, and Y. Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 2012.
- [28] J. Lin, R. Khade, and Y. Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, pages 1–29, 2012.

- [29] P.-F. Marteau. Time warp edit distance with stiffness adjustment for time series matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):306–318, Feb 2009.
- [30] K. Ø. Mikalsen, F. M. Bianchi, C. Soguero-Ruiz, and R. Jenssen. Time series cluster kernel for learning similarities between multivariate time series with missing data. *Pattern Recognition*, 76:569–581, 2018.
- [31] A. Mueen, E. J. Keogh, and N. Young. Logical-shapelets: an expressive primitive for time series classification. In C. Apté, J. Ghosh, and P. Smyth, editors, *KDD*, pages 1154–1162. ACM, 2011.
- [32] P. Nemenyi. *Distribution-free Multiple Comparisons*. Princeton University, 1963.
- [33] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge discovery and data mining*, KDD ’12, pages 262–270. ACM, 2012.
- [34] T. Rakthanmanon and E. Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *SIAM International Conference on Data Mining*, pages 668–676, 2013.
- [35] C. Ratanamahatana and E. Keogh. Three myths about dynamic time warping data mining. In *Proceedings of SIAM International Conference on Data Mining (SDM05)*, volume 21, pages 506–510, 2005.
- [36] P. Schäfer. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505–1530, 2015.
- [37] P. Schäfer and U. Leser. Fast and accurate time series classification with weasel. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 637–646, 2017.
- [38] J. Shieh and E. Keogh. ISAX: Indexing and mining terabyte sized time series. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.
- [39] A. Shifaz, C. Pelletier, F. Petitjean, and G. I. Webb. Ts-chief: A scalable and accurate forest algorithm for time series classification. *Data Mining and Knowledge Discovery*, pages 1–34, 2020.
- [40] A. Stefan, V. Athitsos, and G. Das. The move-split-merge metric for time series. *IEEE Trans. on Knowl. and Data Eng.*, 25(6):1425–1438, June 2013.
- [41] C. W. Tan, F. Petitjean, E. Keogh, and G. I. Webb. Time series classification for varying length series. *arXiv preprint arXiv:1910.04341*, 2019.

- [42] V. Vapnik. *Statistical learning theory*. Adaptive and learning systems for signal processing, communications, and control. Wiley, 1998.
- [43] Z. Xing, J. Pei, P. S. Yu, and K. Wang. Extracting interpretable features for early classification on time series. In *SDM*, pages 247–258, 2011.
- [44] 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)*, pages 840–847. Morgan Kaufmann, 2003.
- [45] 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*, KDD '09, pages 947–956, 2009.
- [46] L. Ye and E. Keogh. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, 22:149–182, 2011.