AUTOREGRESSIVE FORESTS FOR MULTIVARIATE TIME-SERIES MODELING

by

Kerem Sinan Tuncel

B.S., Industrial Engineering, Bilkent University, 2014

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2017

AUTOREGRESSIVE FORESTS FOR MULTIVARIATE TIME-SERIES MODELING

APPROVED BY:

Assist. Prof. Mustafa Gökçe Baydoğan (Thesis Supervisor)	.....
Prof. Necati Aras	.....
Assist. Prof. Melih Kandemir	.....

DATE OF APPROVAL: 21.06.2017

ACKNOWLEDGEMENTS

I would like to begin with thanking Dr. Mustafa Gökçe Baydoğan. I feel very lucky to have the opportunity to work with him in my graduate studies. I also thank him for his patient approach and continuous guidance and support throughout this process. I learned a lot from him.

I am grateful to Prof. Necati Aras and Assist. Prof. Melih Kandemir for their participation in my thesis jury.

I have been working as a research assistant in Dr. Baydogan's project titled 'Enhanced Ensemble Mechanisms for Time-Series Data Mining' within the scope of Bogazici University Scientific Research Projects (BAP) program. I would like to thank the BAP program for providing me assistantship throughout my studies.

Many thanks goes to my family; my mother, father and brother. I am very lucky to have such a family that can offer experienced guidance about research as well as emotional support. I owe a lot to their endless love and support. Their valuable advice always kept me in the right track.

Lastly, thanks to all my friends who are always able to lift my spirits when I feel downhearted. Especially, my beloved girlfriend Ece, thank you for always being there for me with your sympathetic and loving attitude. It gives great confidence to have such a supportive group of people around, always keen to help when it is needed.

ABSTRACT

AUTOREGRESSIVE FORESTS FOR MULTIVARIATE TIME-SERIES MODELING

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 amd 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.

ÖZET

ÇOKDEĞİŞKENLİ ZAMAN SERİLERİNİN MODELLEMESİNDE ÇOKDEĞİŞKENLİ ORMANLAR

İçerdiği verinin kompleks yapısı sebebiyle Çokdeğişkenli Zaman Serileri'nin analizine verilen önem geçtiğimiz on yıl boyunca önemli raddelere yükselmiştir. Farklı uyguluma alanlarında gözlemlenen değişken sayısı ve zaman serilerinin uzunluğundaki artış sebebiyle verimli temsili gösterimler bulmak gereklidir. Örneğin Saklı Markov Modelleri (SMM) ve Özbağlanımlı (AR) modeller gibi model bazlı yaklaşımlar temsili gösterimi modellerle sağlayıp bu problemi çözmek için model parametrelerini kullanmaktadırlar. SMM ve AR modellerinin zaman serisi gösterimi bulmada başarılı olduğu uygulamaların varlığı bilinse de, SMM modelleri veriler arası bağımsızlık varsayımıyla yola çıkmakta ve AR modelleri de değişkenler arasındaki ilişkinin doğrusal olduğunu varsaymaktadır. Gerçeteki çoğu sistem doğrusal olmadığından, geleneksel yöntemler Çokdeğişkenli Zaman Serileri'ne gösterim bulmakta başarısızdır. Bu problemi çözmek için, değişkenler arası doğrusal olmayan bağıntıları da modelleyebilecek özbağlanımlı bir ağaç bazlı topluluk yaklaşım önerilmektedir. Çokdeğişkenli özbağlanımlı orman, yani mv-ARF, güçlü ve etkili temsili gösterimler bulabilecek, büyük veri boyutlarından etkilenmeyen, parametrik olmayan, özbağımlı bir yaklaşımdır. Bu yaklaşımın temelinde, öğrenilen modellerden elde edilen tahmin hatası bazlı bir temsili gösterim bulunmaktadır. Bu yönüyle mv-ARF, zaman serisi sınıflandırma problemlerinde sıkça kullanılan "çekirdek" yöntemlerine benzemektedir. Bu tezde, mv-ARF yöntemi literatürde yaygınca kullanılan birtakım Çokdeğişkenli Zaman Serisi sınıflandırma problemleri üzerinde uygulanmış, verimli ve literatürdeki ölçütlerle rekabet edebilecek sonuçlar verdiği kanıtlanmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$a_{i,j}$	Transition probability from state i to state j
$b_i(X(t))$	Emission probability of observation value $X(t)$ from state i
C	Number of class labels in a dataset
d	Order of first differences
D	Depth of a decision tree
$G(x)$	Transition function
h_j	j^{th} tree within RF
J	Number of trees within an ensemble
k	Number of neighbor instances in k-NN algorithm
M	Number of attributes in a MTS
N	Number of time-series within the training set
p	Autoregressive lag order
r_k	Boundary for regime k
q	Moving averages order
T	Length of MTS
V	Matrix representation of training set of size N with time-series of length T and M attributes
$x_m^n(t)$	Observation value of m^{th} attribute of time-series n at time t
X^n	Notation for the n^{th} time-series within a set of MTS
$\hat{X^n}$	Notation for the estimate for the n^{th} time-series.
z_t	Threshold variable or transition variable
α	Tuning parameter in AR Kernel Method
β	Autoregressive coefficients
δ_c	Multi-response tree-based ensemble learner trained on time-series with class label c
$\mathbf{\Delta}$	Set of multi-response ensemble learners constituent of mv-ARF algorithm
γ	Ratio parameter

ϵ	Covariance Matrix
Θ	Feature space acquired with AR components
μ	Mean or Mean Vector
ϕ^n	Error representation for the n^{th} time-series
$\mathbf{\Phi}$	Error-representation matrix
π_i	Initial state probability for hidden state i

LIST OF ACRONYMS/ABBREVIATIONS

AR	Autoregressive Model
ARCH	Autoregressive Conditional Heteroskedasticity
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
AUSLAN	Australian Sign Language
BOP	Bag of Patterns
CART	Classification and Regression Trees
CBDTW	Correlation-based Dynamic Time Warping
CDM	Compression Based Dissimilarity Measure
CMU MOCAP	Carneige Mellon University Motion Capture Lab
CNN	Convolutional Neural Networks
DFT	Discrete Fourier Transform
DTW	Dynamic Time Warping
DWT	Discrete Wavelet Transform
ECG	Electrocardiogram
EEG	Electroencephalogram
EM	Expectation Minimization
Eros	Extended Frobenius Norm
EMG	Electromyography
GMM	Gaussian Mixture Models
HMM	Hidden Markov Model
KNN	K Nearest Neighbor Algorithm
LCSS	Longest Common Subsequence
LIBRAS	Brazilian Sign Language
LPS	Learned Pattern Similarity
MA	Moving Averages Model
MC-DCNN	Multi Channel Deep Convolutional Neural Networks
MRT	Multivariate Regression Tree

MSE	Mean Squared Error
MTS	Multivariate Time-Series
mv-ARF	Multivariate Autoregressive Forests
OOB	Out of Bag Observations
PAA	Piecewise Aggregate Approximation
PCA	Principal Component Analysis
PDF	Probability Distribution Function
PIP	Perceptually Important Points
RF	Random Forest
SAX	Symbolic Aggregate Approximation
SETAR	Self-exciting Autoregressive Model
SOM	Self Organizing Maps
STAR	Smooth Transition Autoregessive Model
SVD	Singular Value Decomposition
TAR	Threshold Autoregressive Model
URT	Univariate Regression Trees
UTS	Univariate Time-Series
VAR	Vector Autoregressive Model

1. INTRODUCTION

Time series are a type of temporal data that measures the dynamic behavior of an entity with respect to time. An ECG reading, daily temperature measurement or stock prices which measures the daily prices of stocks of a corporation are some examples for time-series data. The research in time-series analysis and time-series data mining has increased significantly due to the vast areas of practical applications of time-series data. Some well known and popular examples for application areas of time-series are; finance, medicine and various industrial fields.

Time-series data mining can be broadly separated into two problems. First problem is time-series representation. Time-series data are complex and high dimensional due to its temporal and continuous nature, thus conventional analysis techniques are hard to apply on such datasets. In order to make robust analysis on time-series and to make conventional approaches applicable for time-series data, finding efficient representations or appropriate measures for time-series is necessary. The second problem is mining of the time-series, which is extracting usable information from time-series based on the time-series representation. Most common knowledge extraction tasks within literature are; pattern discovery, classification, rule discovery and summarization [3]. More detailed information about the tasks are provided in later parts. Moreover, as the research on time-series evolved and more data became available, focus on Multivariate Time-Series (MTS) gained significant importance. In this work, we propose a novel modeling strategy which is able to provide efficient representations for MTS.

Multivariate Time-Series analysis emerged as a popular new research direction in time-series research due to the enhancements on hardware, communication and signal transfer technologies [4]. MTS are a complex extension for time-series which consists of multiple variables that are measured together. Motion capture is a good example for MTS data. For example, a sensor reading measuring the xyz coordinates of the hand while making a hand gesture is a MTS. Altough MTS is considered to be equivalent with multiple, independent time-series [4], MTS analysis gained considerable focus

because multiple variables adds a complexity to the modeling task. The reason for this additional complexity is due to interactions and relationships inbetween separate variables. Thus, MTS are not simply multiple, independent UTS. This extension calls for different type of analysis for MTS, or at least adaptations of UTS methods to work in MTS framework.

Another reason for the immense interest on MTS is the vast area of application of multivariate and temporal data. MTS datasets became widespread and there are various applications of MTS modelling in fields such as; medicine [5–8], image and video processing [9,10] finance, engineering and industrial fields [11]. For example, in motion recognition, electromyography (EMG) signals from multiple muscles are recorded for four seperate upper-arm movements and this MTS dataset is used to classify these four seperate upper-arm movements [7]. Also, a public sector use is shown [12], where a novel MTS clustering technique is discussed for clustering of crime locations, such as states and districts, with similar crime trends. Another example from hydrology is provided [13], in which MTS modelling of water resources is considered in water management and planning. Furthermore, as a tourism forecasting application, the use of multivariate time-series analysis of tourist inflow data over univariate methods, to forecast tourism demand is discussed [14].

Many of the existing studies in MTS analysis can be described within two categories; feature based methods and distance (similarity) based approaches. Feature based approaches 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. This way, a conventional distance matrix is generated for the MTS dataset. Thus, any

learner that utilizes similarity information ,such as K-Nearest Neighbor (K-NN), can be trained on the MTS dataset. Figure 1.1 illustrates the basic ideologies of distance based and feature based methods.

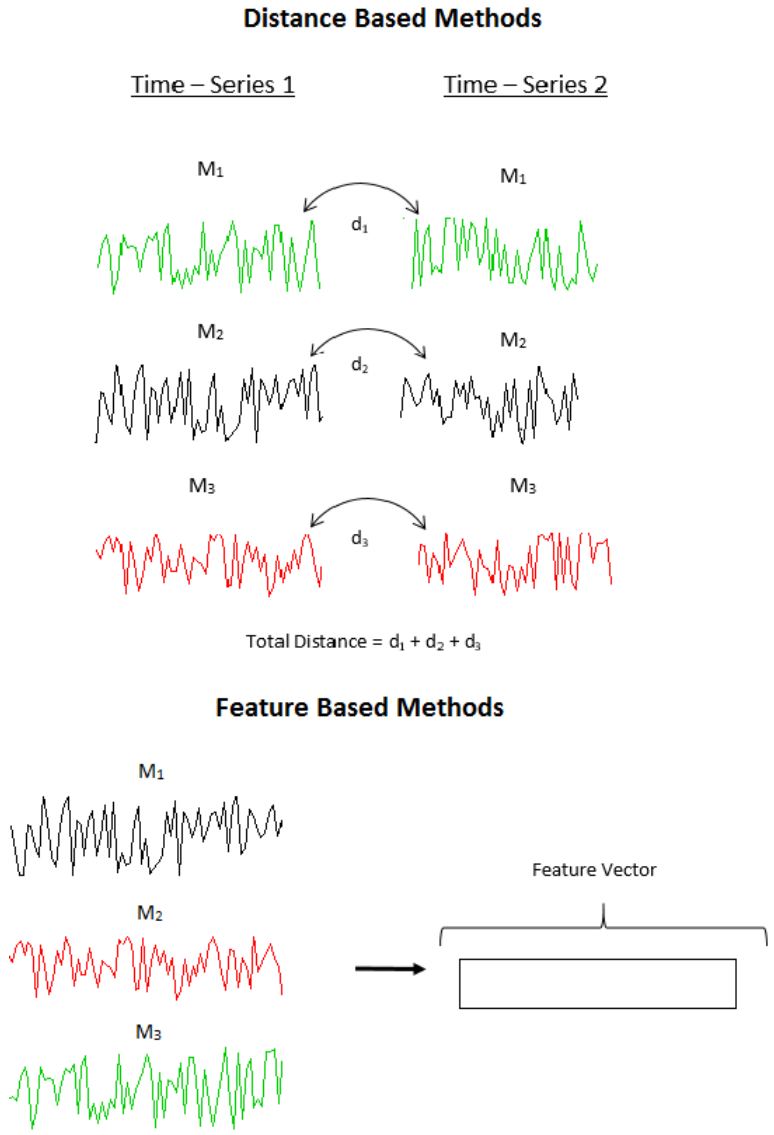

Figure 1.1. Distance based methods vs. Feature based methods

Furthermore, model based approaches can be placed at the intersection of feature based and distance based approaches. 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 [7]. 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). Benefiting from the ideas in model based approaches, this paper provides an autoregressive modeling strategy for MTS. As a common approach, we consider classification as an application to evaluate our modeling approach. 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 [15]. Consequently, Symbolic Aggregate Approximation (SAX) [16] is another representation technique designed for univariate time series (UTS). SAX methodology discretize the time-series by learning symbols from aggregate values within intervals. Similarly, [17] proposes another symbolic representation for MTS which generates a codebook from the terminal nodes of decision trees which considers the features simultaneously. Moreover, for longer time-series and higher complexity data, higher order of transformations such as wavelets and Fourier transforms are discussed [18]. Furthermore, using HMM to represent MTS is also considered [19]. An HMM-based methodology with a PCA representation for motion recognition is provided by [19].

Distance based methods mainly consists of modifications of univariate time-series (UTS) similarity approaches to work in MTS framework. For example, [20] utilizes Dy-

namic Time Warping (DTW) in human action recognition and [21] discusses the use of DTW in MTS pattern matching. Among the distance measures in the literature, Dynamic Time Warping (DTW) [22] has became 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. [23] focuses on improving time efficiency of DTW whereas [24] focuses on improvement of DTW accuracy. Also, there is research being conducted to find more robust distance measures than DTW. For example, [25] proposes 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, [26] proposes an autoregressive kernel-based distance measure for MTS data. 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. Another AR modeling strategy [27] aims at learning MTS representation based on local autopatterns The provided representation is presented with a similarity measure that is called learned pattern similarity (LPS). LPS methodology provides an unsupervised representation and similarity measure between time-series based on a tree-based ensemble. Provided methodology is similar to a bag-of-words approach since the representation is learned from segments of time-series Furthermore, the methodology is different from other feature based approaches since it do not contain a feature extraction step due to the embedded learning of the representation.

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 strategy with the capabilities of tree-based ensembles which provides a flexible and simple generative approach with only a few parameters. Moreover, combining AR components with the multi-response learning scheme utilized in mv-ARF provides comprehensive knowledge for modeling the interactions between features. The classifier we introduce exploits the modeling strengths of mv-ARF approach. 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. Provided approach aims to capture the dynamics of MTS by using a representation based on a vector autoregressive model. Proposed methodology is denoted as "*M*ulti*V*ariate *A*uto*R*egressive *F*orests", mv-ARF, which trains tree-based ensemble learners with autoregressive (AR) components in a multitask setting. Therefore, mv-ARF is a multitask vector autoregression model which aims to model a vector of targets based on a matrix of past observations. Figure 1.2 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, which leads 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.

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 Figure 1.2. 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.

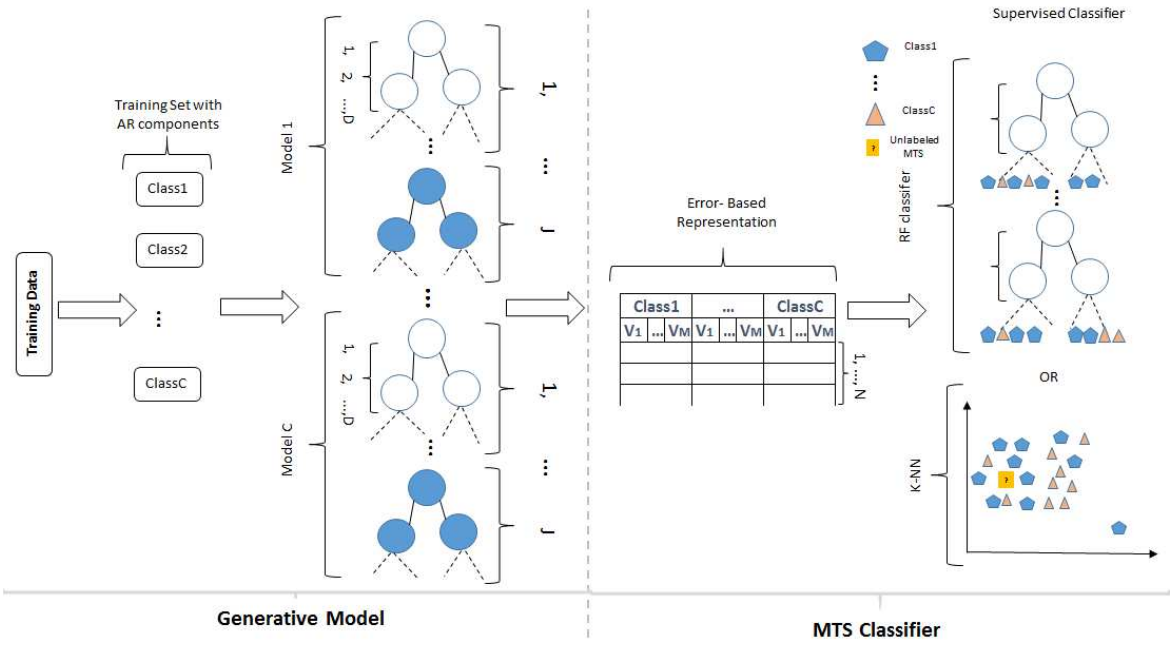

Figure 1.2. Summary of mv-ARF algorithm. For each class, a tree-based ensemble model is trained on lagged observations. After models are learned for each class, each multivariate time series is represented by their goodness of fit to the models.

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. Therefore, multivariate models provide more robust solutions for MTS. 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.

2. LITERATURE REVIEW

This work is concerned with MTS modeling. Since MTS is an extension for time-series, research on MTS is a sub-branch of a wider research area that is time-series data mining. For that reason, the general applications and data mining tasks of MTS are very similar to time-series research. In order to have a good understanding about the ongoing research on MTS, it is necessary to know the research directions of time-series data mining. Therefore, this chapter is devoted to a literature survey of time-series representation and time-series data mining tasks as well as MTS modelling.

2.1. Time-Series Representation

The principal reason for time-series representation is to reduce the dimension of the data and to find a robust approximation of the time-series. Most traditional approach for representing time-series data is to project the time-series into a domain with lower dimensionality, paired with an indexing scheme [4]. Representation methods can be analyzed within two main categories; representations within the time domain and representations in the transformation domain [4].

The simplest way within the time domain is sampling [28] the time-series data in which time-series is approximated using a subset of sampled data points. The ratio between the total number of data points and sampled data points is considered as a parameter for sampling. Altough this method is simple, it may cause distortion in time-series shape. Hence, a more advanced method, Piecewise Aggregate Approximation (PAA) is proposed [29]. In PAA, mean of each sample is used to represent the data points within the sample. Furthermore, searching for important points for time-series representation is also discussed. These points are called perceptually important points (PIP). PIP identification is initially discussed in [30]. The idea behind PIP identification methods is to approximate the time-series using major extrema and to discard smaller fluctuations. [31]. Illustrations of sampling, PAA and PIP are provided

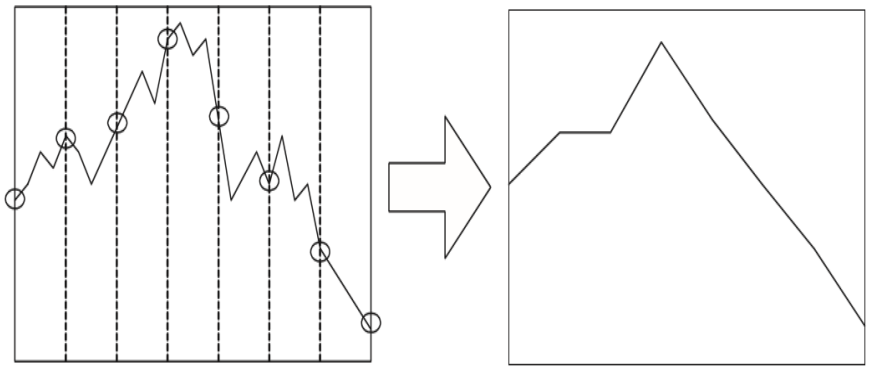

Figure 2.1. Sampling example; time-series on the right is the sampled version of the time-series on the left.

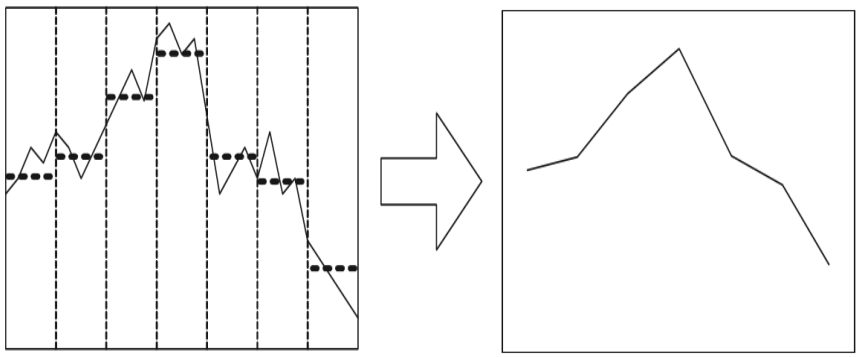

Figure 2.2. PAA example; horizontal lines show the segment means.

in Figure 2.1, Figure 2.2, Figure 2.3 respectively.[1] .

Symbolic representations are also used for reducing dimentionality. Symbolic representations aims at discretizing the time-series and associating each segment with a corresponding symbol. [32,33]. Most prominent method among symbol based representations is Symbolic Aggregate Approximation (SAX) method [16] that uses segments acquired with PAA to generate what is called a codebook or alphabet. As an alternative to using mean values; subsequence clustering, volatility, gradient alphabets and key-sequences are proposed to generate the codebook. [4, 34–36].

[1]These figures are acquired from [4]

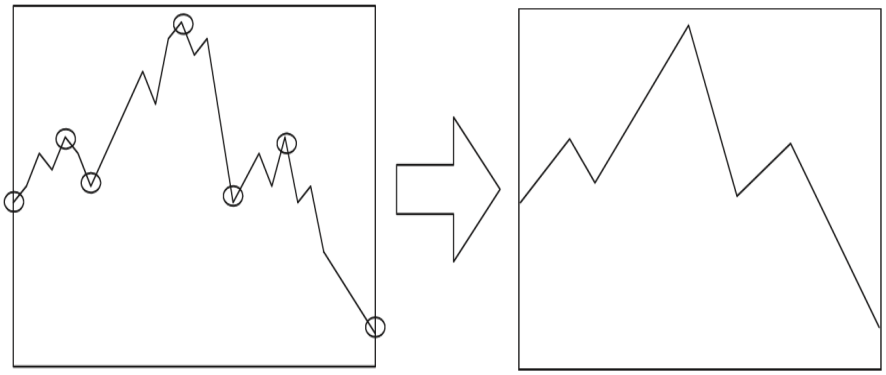

Figure 2.3. PIP example; time-series on the right is the approximation acquired with seven PIP's circled on the left.

As for the methods that represent the time-series in the transformation domain, one of the prominent transformation methods is to use Discrete Fourier Transforms (DFT) [37]. DFT explains the time-series using trigonometric forms as periodicity components. Discrete wavelet transform (DWT) [38] is also used as an alternative to DFT [39]. Another similar transformation that is used is the Haar Transform [40, 41] which defines a series of arithmetic operations on a time-series [42]. Furthermore, Principal Component Analysis (PCA) is also another transformation that is commonly used on time-series. PCA is applied to extract more important features and to reduce the dimensionality of the time-series. Another method is to use Hidden Markov Models (HMM) [43] which aims to describe the behavior of a time-series by extracting state-sequence information. HMM models transform the time-series into a state space model similar to a regular Markov Chain representation, only incorporating hidden states that define the state-sequence relationships. Moreover, time-series representations are generally associated with an appropriate indexing scheme. Thus, a representation method is efficient only if an indexing scheme can be applied to the reduced dataset.

2.1.1. Time-Series Similarity

Finding robust measures for time-series similarity is also a very important task related to time-series representation. For that reason, many representation methods

also provide similarity measures for the transformed time-series [4]. Time-series similarity research falls into two main categories [37]. First is whole sequence matching, in which the entire length of time-series are considered and a similarity function (or a distance measure) to measure the similarity (or distance, dissimilarity) between two time-series is provided. In sub-sequence matching, for a given smaller sequence A and larger sequence B, the task is to identify the sub-sequences within B that are similar to A [4].

Most well-known distance measure to asses the similarity of two time-series is to calculate the Euclidean distance with the transformed time-series metrics. For example the Euclidean distance between PIP's or DFT coefficients [37] of two time-series. Many distance measures can be found in the literature but the most prominent and commonly used among those is the famous Dynamic Time Warping (DTW) metric [22]. DTW aims to align two sequences regardless of their length or frequency. Since it is the most prominent similarity measure for time-series, detailed explanation of DTW methodology is provided in later parts. As a similar approach to DTW measure, Longest Common Subsequence (LCSS) distance is proposed [44]. Like DTW, LCSS also allows the time dimension to stretch but some subsequences are allowed to be unmatched. The advantage of LCSS over DTW is the consideration of outliers.

Furthermore, threshold based distance methods are discussed [4,45]. These methods use threholds for comparison instead of exact values. Moreover, parameter based distance measure is discussed which is based on Kolmogorov complexity theory and adopts compression based dissimilarity measure (CDM^3) [46]. Subsequently, Bag of Patterns (BOP) is proposed, which is a histogram based similarity measure that considers certain patterns within the time-series and compares them using CDM.

The other direction of similarity research is subsequence matching. Within subsequence matching context, the smaller sequence is called the query sequence and the goal is to find matches between the query sequence and subsequences of the longer sequence. In this context, a generalization of DFT approach is proposed [47] and many of the common approaches build upon this generalization. For example, Dual Match and General

Match algorithms are proposed [48,49] that are DFT-based pattern matching methods utilizing sliding windows. Furthermore, subsequence detection using SAX methodology [50] and pattern matching using linear segments [51] also include DFT. All the methods that utilize DFT are based on lower-bounding of the Euclidean distance [4]. Other than DFT approaches, an indexing method called S^2-Tree is proposed [52], that utilizes string searching methods for different time-series representation approaches. Also, a hierarchical similarity search for subsequence detection in a transformation domain is proposed.

2.2. Time-Series Data Mining Tasks

Time-series representation can sometimes be the only goal all by itself. But most of the time, it is associated with a data mining task which is mostly the ultimate goal in time-series data mining. Mining tasks aim to extract underlying information or knowledge from the time-series. Mining tasks can be applied on both time-series domain and representation domain. There are four most common data mining tasks; pattern discovery, classification, rule discovery and summarization [4]. Among those mining tasks, pattern discovery is the most common and clustering is the most common method for it.

Pattern discovery is a task to find interesting patterns within a time-series. These patterns can be continuously appearing and surprising patterns [3,53]. Pattern discovery tasks also include and sometimes called; motif discovery, anomaly detection and finding discords [4]. In pattern discovery literature, most common applied method is using clustering based on time-series distances. Two parameters; the number of cluster centers and the initial cluster centers are of importance for clustering operations. The initial cluster centers can be chosen arbitrarily or by some operations for initialization. Similarly, the number of cluster centers can be set to a predefined value or can be selected during the clustering process. The problem of data storage is a major problem in time-series clustering since time-series size increases with respect to time, which decreases the speed of pattern discovery process. A way to compress long time-series is required not for only effectiveness reasons but for keeping the amount of

data in an acceptable level for pattern discovery [4]. In order to achieve this, a PIP based clustering technique is proposed [3]. Moreover, the self-organizing map (SOM) is proposed [54], which is a special neural clustering method with immense clustering capabilities. Furthermore, for clustering of time-series with irregular spacing within time axis, Fuzzy C-means algorithm (FCM) is adopted [55]. Similarly, Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA) models are also considered for clustering of time-series as mixtures of partition around medoids and Expectation Minimization (EM) methods [56, 57]. Also, HMMs are a common approach for time-series clustering [58]. For example, a hybrid model using DTW for initialization of HMM model is proposed [59].

Another common area of research within pattern discovery framework is identifying periodic subsequences. Many methods to identify common periodic patterns are proposed [60, 61]. Clustering methods for frequent pattern discovery is also discussed. But in this case, clustering methods search for common grouping tendencies. Because of that, they are not as general as other methods for periodic pattern discovery [3]. Also, definition of a match is also discussed. Many of the algorithms are based upon discretization of time series through sliding windows methodology. Therefore, most of the matches found are the windows next to the actual match thus they are redundant. Subsequently the concept of a trivial match is defined [62]. Furthermore, for a given time-series, the problem of enumerating the subsequences with the highest frequency is defined as significant motif discovery and the K-motif discovery algorithm is proposed [62] that is a methodology to define the K most frequent subsequences within a time-series. Many other methods for frequent subsequence discovery utilizing K-motif discovery is proposed, such as the motif-based clustering methodology [63], which aims to apply a regular clustering algorithm on the motifs discovered by K-motif discovery method.

Another family of methods within pattern discovery framework is anomaly detection methods. Adoption of SAX methodology for improved detection of surprising patterns is discussed [64]. Similarly algorithms that use SAX representation for anomaly detection with respect to shape and mixing SAX with suffix trees and Markov Mod-

els for finding surprising patterns are proposed [65, 66]. Furthermore, non-parametric approaches and a bitmap approach for anomaly detection is also discussed [67, 68]. Lastly, all of the discussed methods are based on single periodicities, in other words there exist only single frequency for data income and there is also research focused on pattern discovery across multiple periodicities, which are also referred as multiple granularities within literature [4, 69, 70].

Another very common task in time-series data-mining is classification, which is a highly traditional task in data mining literature. Due to its temporal nature, time-series data requires specialized methods for the classification task. For that purpose, classification of time-series with respect to local patterns is proposed [71]. Similarly, representations based on wavelet transforms are utilized as parameters for classification [72]. Additionally, using metafeatures [73] which are recurring subsequences within time-series to generate classifiers is a widely recognized methodology which will be discussed in more detail in further sections. Other family of classification methods are based on tailoring existing classification methods for time-series classification framework. An example for this is the utilization of Gaussian Mixture Models (GMM) on signal data classification [74]. Similarly, ensembles which have interval and DTW based decision trees as base classifiers are proposed for time-series classification [75]. Lastly, another notable study within this area is the study proposes utilizing numerousity reduction using nearest neighbor and DTW based classifiers for time-series classification [76].

Rule discovery is a data mining task that is rather ambiguous when compared to the other tasks. In essence, rule discovery is prediction of the future states of a temporal sequence from the current pattern instead of the current state, which is the traditional approach in time-series prediction [77]. Most notable approach on rule discovery is the Association Rule Mining algorithm [78]. But, the model is based on symbols observed in financial data, therefore the algorithm is adopted to time-series data. The idea is to discretize the time-series using symbolic representations and applying rule mining on the disretized time-series. Examples include methods such as, using subsequence clustering for symbol generation [79], using n-dimensional inter-

transaction rules [80] and applying genetic programming on the discretized series [81]. Furthermore, decision trees are the other common approach in rule discovery [4]. A similar approach to the Association Rule Mining is to discover important rules by using decision trees with symbols acquired from subsequence clustering. Lastly, an approach to use classification trees on time-series represented as sequences of events is proposed within this framework [82].

Final data mining task within this discussion is summarization. The goal of summarization is to produce efficient and compact summary of time-series data for analysis. Mainly, clustering techniques are adopted for identifying important features and recurrent patterns throughout the time-series. Initially, a system which is a combination of knowledge discovery and word processing is proposed [83]. The system is based on wavelet transforms and it is tested on weather data. Similarly, neural networks are used to extract grammatical compounds to explain time-series with multiple variables [84]. Subsequently, the SumTime project emerged [85]. SumTime is able to produce textual summaries of time-series with actual words and sentences by mapping the most important and recurring patterns and features to words and phrases. Other research can also be found on linking and demonstrating the relations between time-series and language models [86, 87]. Furthermore, another research within summarization framework includes utilization of multi-scale wavelet analysis [88] and fuzzy quantifiers [89] to generate linguistic summarizations.

2.3. Multivariate Time-Series Modelling

MTS modelling emerged as a new research direction that aims to handle temporal and multivariate data. The primary research on MTS is evolved around finding representations to capture the complex structure of the data. There are two modeling strategies that are common within literature. First are feature based strategies which aims to find a rectangular representation for MTS by extracting descriptive features learned by various models and their parameters. Second is distance based strategies which aim to provide similarity metrics that works well with MTS. Moreover, model based strategies can be used for both extracting features and to define a similarity ma-

trix depending on the usage of the model. The proposed model, mv-ARF, is a vector autoregressive approach benefitting from the ideas of model based strategies. Since mv-ARF is tested within MTS classification framework, the literature survey made on MTS modelling primarily consists of MTS classification models.

There has been extensive discussion about effective and interpretable feature extraction techniques within MTS classification framework. For this purpose, expanding attribute-value learners to domains with recurring structure is discussed [73]. Meta features are introduced as the user-defined recurring substructures within the dataset. Recurring extrema can be given as example to a meta feature. Synthetic attributes, that are dependent on the presence (or absence) of the meta features, are built with a novel clustering method. Metafeatures methodology aims to find a comprehensive representation of the data since user defined meta features also incorporates intuition. Also, the representation allows the temporal nature of the data to be handled by basic attribute value learners, which makes it similar to mv-ARF algorithm.

It has been underlined that, many strong machine learning tools, however complex, are not able to achieve competitive accuracy results with the K Nearest Neighbor (K-NN) learner using DTW distances in MTS classification [90]. DTW proved to be a robust approach for temporal datasets. In order to combine the strength of DTW with strong machine learning tools, a feature based approach that uses DTW distances as features is proposed [90]. The proposed features consist of DTW distances among separate time-series within the dataset. A strong supervised learner (such as SVM learners) can be applied to this kind of representation. Even tough not being the first to discuss DTW features [91], positive results are reported [90].

Moreover, DTW methodology can become computationally expensive with high dimensionality [22,23]. Due to this fact, the necessity of searching for feature extraction methods with higher accuracy results is discussed [92]. In order to achieve competitive accuracy, a method based on utilizing Multi Channel Deep Convolutional Neural Networks (MC-DCNN) is proposed [92]. Deep learning techniques are known to be used in image classification. Particularly, proposed methodology uses deep convolutional

neural networks for identifying foods from images [92]. The aim is to extend the deep learning scheme to temporal datasets by learning features from each channel, which works with a single dimension of MTS. A multi-layer perceptron is trained by MC-DNN on the features learned in the initial stages, that are used for classification in further parts. Furthermore, in order to acquire higher accuracy, unsupervised initialization is applied, to pre train the data for the Convolutional Neural Networks (CNN). The reported results shows that, MC-DCNN is computationally much more efficient than 1-NN DTW classifier and it is also competitive in accuracy [92].

Extension of shapelet discovery techniques for MTS classification is discussed [93]. Shapelet discovery is a methodology to discover subsequences (which are referred to as shapelets) to represent a time series. Generally, shapelets with high predictive accuracy are chosen, thus comparison of candidate shapelets include predicting accuracy which results in high performance costs. Instead of comparing predictive accuracies, a new principle that iteratively learns optimal shapelets through classification loss functions is proposed [94]. Furthermore, a fast shapelet based classification technique with a MTS extension is also proposed [93], in which the optimal shapelets are found through scaling up candidate shapelets combined with a supervised clustering of the shapelets. The extension to MTS is achieved by simply picking candidate shapelets from random sampled features. It has been documented that this application provides a computationally efficient way to classify MTS. For this methodology, extensive experimentation in UTS has been documented but the methodology is tested with a few MTS datasets.

A general approach in MTS methods is to compare and analyze separate features as independent UTS. There is evidence stating that, this approach causes inaccuracy in datasets containing correlated features [95]. A novel similarity measure which is based on segmentation, PCA and dynamic time warping called; Correlation-based Dynamic Time Warping (CBDTW) is proposed [95]. Two common PCA models are used as homogeneity measures in the segmentation part, which is depending 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 recompense the time shifts between segments.

Another PCA based methodology is discussed [19], in which motion trajectories are classified using HMM model based on PCA representations for separate motions. The trajectory data is segmented into atomic units of action. The segmentation points are detected by computing 2-D curvature of the trajectories. PCA representation is acquired from all the subtrajectories and from all dimensions. HMM models are trained on the subtrajectories that are now similar to speech data, which is known to be successfully modeled by HMM. The state size of the HMM is selected with a spectral clustering algorithm. The state PDF's are represented as Gaussian Mixture (GM) models. Each Gaussian component is a multivariate normal distribution with the same dimension as the the number of principal components representing the subtrajectories. After the HMM models are built for each class, the classification is made by calculating log-likelihoods of unlabeled trajectories from each HMM model.

Furthermore, use of a multivariate HMM model to identify sea regimes from multivariate marine time-series is discussed [96]. An unsupervised learning scheme is provided, which is aimed to handle missing data, temporal autocorrelation and skewness of features by combining the use of HMM with clustering. Two clusters, Torodial and skew elliptical, are defined by mixture of von Mises and mixture of linear models respectively. The clustering is achieved by using HMM models. Furthermore, an EM algorithm is used to handle missing values and other sources of incomplete information. Commonly, a GM model, in which the multiple responses are represented as products of univariate responses, is used in multvariate HMM, which can be restrictive. Similarly, the non-normal mixtures commonly assume conditional independence. But the aim in this work is to provide a partial relaxation to this assumption in order to incorporate non-normal features and variable dependencies to HMM framework.

A framework using stacking ensemble [97] is discussed [98]. The method aims to capture the relative importance of univariate classifiers on MTS classifying framework. Different univariate classifiers are trained on each feature in addition to a multivariate classifier, which is trained on the entire MTS. The results from the classifiers are then combined by stacking. Stacking can combine results from different models. Thus, the common use is to train different models at the base level. However, apart from the the

multivariate model, the $M + 1$ base-level models ,trained for a M-dimensional MTS, are all identical. Furthermore, it has been shown that this framework can be used to enhance the accuracy of the initial, singular multivariate classifier.

Moreover, as a distance based example, a kernel based methodology called VAR kernel is discussed [26] in order to provide an alternative distance measure for DTW. VAR kernel is also robust for datasets with high number of attributes. The proposed kernel builds on the "covariance kernel" methodology [99]. Given two time-series, the defined kernel is the integral of products of probability distributions with a prior function over a parameter space defined by VAR method. The prior function is the matrix-normal inverse-Wishart prior, which allows a useful closed form expression for the kernel. Conventionally, the VAR model is used to find parameters for feature extraction methods. But in this kernel, VAR model is used to define the parameter space. Furthermore a second kernel is also proposed [26]. The only difference of the second kernel is the utilization of Gram matrices, which allow the kernel to efficiently handle structured datasets. The results reported for VAR kernel implies that the computational complexity of the proposed kernel is manageable for datasets with large number of attributes. Furthermore, accuracy results which are competitive with DTW distance are reported.

A novel modeling strategy which aims to learn representation based on local autopatterns acquired with AR components is proposed in [27]. This representation is presented with a similarity measure called learned pattern similarity (LPS). LPS methodology provides an unsupervised representation and a similarity measure between time-series based on a tree based ensemble. The tree based ensemble is utilized for its flexibility in handling categorical or ordinal features. Provided methodology is a bag of words approach since the representation is learned from segments of time-series similar to a codebook and then the similarity measure is introduced which is based on this representation. Furthermore, the methodology is different from other feature based approaches since there is not a feature extraction step due to embedded learning of the representation. Lastly, the paper discusses a straightforward MTS extension of LPS methodology which is included in our analysis due to its AR modeling strategy.

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 utilized in mv-ARF. But unlike feature based strategies, the primary output of mv-ARF is a generalizable generative model for each class label, whereas feature based methods contribute by providing a rectangular representation. Moreover, mv-ARF provides a similarity measure from prediction error (goodness of fit) rates of its constituent generative model for classification. In that sense, mv-ARF is similar to HMM models using likelihood as similarity for MTS classification. Extracting AR components is the only form of feature extraction included in the model, which makes mv-ARF a simple and easy to interpret methodology. Also, some model based classifiers include extensive parameter optimization processes whereas mv-ARF has very few parameters to consider beforehand. On the other hand, autoregressive nature of mv-ARF model can be problematic in cases where the autoregressive assumption simply doesn't apply.

3. BACKGROUND

Primary output of this work is a novel modeling algorithm for MTS data, which is called mv-ARF algorithm. Proposed algorithm provides an efficient way to model MTS by making use of tree-based learners and autoregressive models. Furthermore, DTW with 1-NN classifier is included in the experiments since DTW has become a benchmark approach for time-series classification problems. Subsequently, as a generative approach with many applications within MTS classification framework, HMM models are also used in the experiments for comparison. Therefore, in order to have a comprehensive understanding of mv-ARF model and the experiments, a little background on tree-based learning, autoregressive models, Dynamic Time Warping and Hidden Markov Models is necessary.

3.1. Dynamic Time Warping

DTW is a method to align two time-dependent sequences regardless of their speeds and lengths. Unlike Euclidean distance, the optimal alignment is found by non-linear warping between data points (Figure 3.1[2]). DTW is an algorithm initially developed for speech recognition to discriminate and evaluate speech templates [101]. In data-mining framework, the output of DTW algorithm is being used as a similarity measure to measure the similarity/dissimilarity between two time-series with uneven lengths. The optimal alignment for time-series X and Y;

$$X = [x_1, x_2, \ldots, x_m] \text{ and } Y = [y_1, y_2, \ldots, y_N] \text{ for } M, N \in \aleph;$$

is found by defining the distance (or similarity, cost) matrix D, where d_{ij} is the distance between the corresponding points in X and Y. D is a MxN matrix and usually Euclidean distance is used to obtain D. A warping path p is a sequence $p = [p_1, p_2, \ldots, p_L]; L \in \aleph$ and $p_i = (n_i, m_i) \in [1, M]x[1, N]$, satisfying the following conditions;

[2]This figure is acquired from [100]

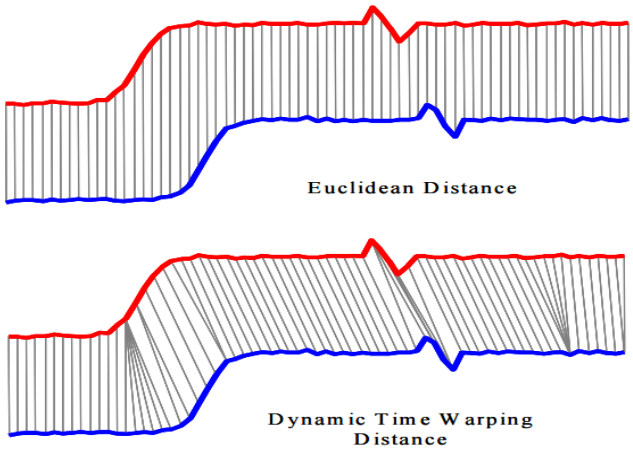

Figure 3.1. Two time-series are similar in shape, but not aligned in time axis. Figure shows how DTW and Euclidean Distances differ with respect to alignment. DTW will capture the similarities in the shape but diagonal warping of Euclidean distance will perform poorly in this example.

(i) $p_1 = (1, 1)$ and $p_L = (M, N)$

(ii) $m_1 \leq m_2 \ldots \leq m_L$ and $n_1 \leq n_2 \ldots \leq n_L$

(iii) $p_{i+1} - p_i = [(1, 0), (0, 1), (1, 1)]\ \forall\ i \in [1, L-1]$

These are the classical conditions applied in DTW algorithm, (i) is the boundary condition which ensures the beginning and the ending of the path is the same with the time-series, (ii) ensures monotonic behavior and (iii) ensures that each element of both time-series are aligned with another point. There are optional parameters and conditions sometimes included to define the warping path. Most common are warping window; $|n_i - m_i| \leq w\ \forall\ i \in [1, L]$, where w is called the warping window controlling deviance from the diagonal and $\frac{n_i}{m_i} \leq r\ \forall\ i \in [1 : L]$ where r is called the slope parameter controlling the slope of the path.

There can be many paths satisfying the above conditions. Each warping path corresponds to an alignment between X and Y, the optimal alignment is the path with

the minimum total cost. Cost of each path is defined as;

$$c_p = \sum_{i=1}^{L} d(x_{m_i} - y_{n_i}) \ \forall \ p$$

DTW distance is the square-root of the total cost of the best alignment. So, DTW similarity between X and Y would be $\sqrt{c_p^*}$. Lastly, a search through the grid for the optimal path results in an exponential complexity, but DTW algorithm applies a Dynamic Programming methodology by calculating the optimal DTW distances matrix and applying backward iterations on DTW distances, thus achieving $O(MN)$ complexity for the algorithm.

DTW distance measure is successfully applied in many time-series applications and has became a benchmark with respect to accuracy in time-series analysis. The DTW distance between two MTS is defined to be the aggregate distance between corresponding features of the MTS. DTW has proven to be successful in MTS classification as well. For that reason, classification results for 1-NN classifier using DTW distances are also acquired and used for comparison.

3.2. Autoregressive Models

Autoregressive models is the common name for a family of statistical models that is widely used in time-series forecasting. AR models are widely used because of their flexibility in modeling many stationary processes [102]. The entire family of AR models are derived from the basic $AR(p)$ model shown in Equation 3.1. $AR(p)$ model (where p denotes the lag value) is a simple regression model in which, the predictor variables are past observations (hence the name autoregressive).

$$X(t) = \beta_0 + \sum_{i=1}^{i=p} \beta_i X(t-i) + \xi(t) \tag{3.1}$$

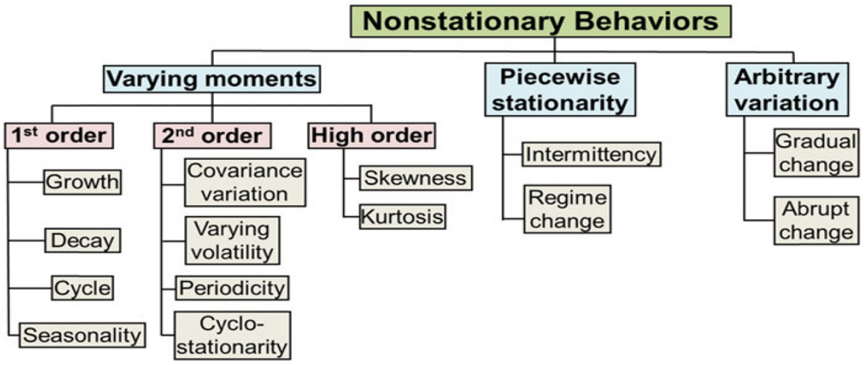

Figure 3.2. A list of notable non-stationary behavior in time-series

Most of the statistical models designed for temporal data analysis assumes stationary and linear dynamic behavior. However, most time-series data observed in real life follows non-linear and non-stationary behavior. Therefore, development of time-series forecasting models mostly evolves around incorporating non-stationary and non-linear behavior within the model framework. In that respect, autoregressive models are not different. Note that, in this context, linearity in behavior means the linear or non-linear interactions between states of a stochastic process with respect to time order. Whereas, non-stationarity implies inconstant behavior of dynamic process moments. The first moment of a probabilistic process is its expectation and second moment is its variance. Thus, in simple terms, first order non-stationary behavior follows unstable mean with respect to time order and second order non-stationary behavior follows unstable standard deviation throughout the dynamic process. Subsequently, higher order non-stationarity implies unstable behavior in higher moments of the dynamic process. A concise summary of non-stationary behavior within time-series framework is illustrated in Figure 3.2[3] .

With its original description in Equation 3.1, capabilities of $AR(p)$ model is only limited to time-series with stationary and linear behavior, which is not a realistic assumption for many real life time-series data. Therefore, many improvements derived from the original $AR(p)$ model is discussed in order to incorporate non-linear or non-

[3]The figure can be found in [102]

stationary behavior into the model. Autoregressive Moving Averages (ARMA) model is introduced which combines the power of AR model with Moving Averages (MA) model [103]. The description of $ARMA(p, q)$ model can be found in Equation 3.2. Note that, p denotes lag order and q denotes moving averages order.

$$X(t) = \beta_0 + \alpha_0 + \sum_{i=1}^{i=p} \beta_i X(t-i) + \sum_{i=1}^{i=q} \alpha_i \xi(t-i) + \xi(t) \tag{3.2}$$

Note that, even though it works well with linear time-series, ARMA model fails when trend or seasonality is introduced [102]. Subsequently, the idea of integrating differences within ARMA model to deal with non-stationarity is proposed, which is called Autoregressive Integrated Moving Averages (ARIMA) model [103]. The idea of differencing is quite simple but also quite effective in practice. ARIMA model is defined also by the order of of first differences (d), thus $ARIMA(p, d, q)$ model handles non-stationarity by introducing first differences into a time series as follows;

$$Y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) \ldots - (Y_{t-(d-1)} - Y_{t-d}) \tag{3.3}$$

where Y_t is a $ARMA(p, q)$ process. As a demonstration of the idea of first differences, the illustration of a first difference of order $d = 1$ applied on a random walk process with a trend is provided in Figure 3.3.

Eventough ARIMA is a clever way to handle non-stationary behavior, the parameter setting scheme for the difference parameter is tedious [102]. Also, the differencing scheme is only able to handle first-order non-stationary behavior. As a methodology to handle second-order non-stationarity within time-series, Autoregressive Conditional Heteroskedasticity (ARCH) methodology is introduced [104], which is initially designed for financial time-series. ARCH model is a statistical method to model the variance of a time-series as a function of its past values. Because of its inception within finance

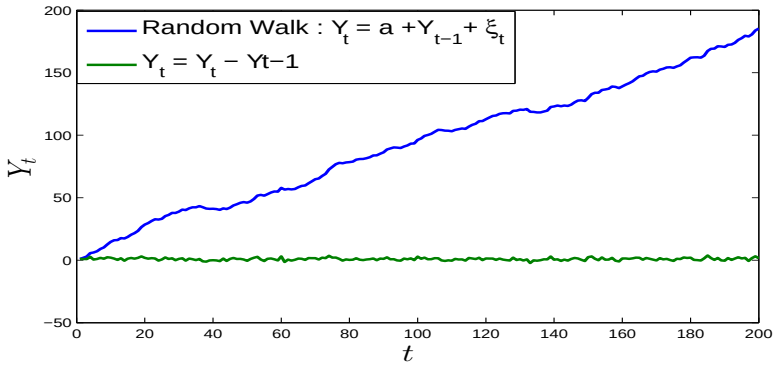

Figure 3.3. Non-stationarity removed from a random walk process with a shift by using first differences

domain, the most common use of ARCH models is in situations with volatile variation where short periods of decrease of increase in variance is present within the time-series. But ARCH model is a robust model to handle second-order non-stationarity, thus it is able to handle gradual changes in variation as well. Given a time-series $X(t)$ with variance $Y(t)$, the $ARCH(p)$ model can be written as;

$$Y(t) = \sigma_t^2 = \beta_0 + \beta_1 y_{t-1}^2 + \beta_2 y_{t-2}^2 + \ldots + \beta_p y_{t-p}^2 \tag{3.4}$$

Note that, $[\beta_0, \beta_1, \ldots, \beta_p]$ have non-negativity constraints in order to avoid negative variances. Also, the formal definition of the variance may differ with respect to the application of the model. The most common definitions for $Y(t)$ are; $\frac{X(t)-X(t-l)}{X(t-l)}$ and $log(\frac{X(t)}{X(t-l)})$ since it is most commonly used in financial time-series.

Another important aspect is to include non-linear behavior within the AR framework. For that purpose, Threshold Autoregressive (TAR) models are introduced [105] which incorporate piecewise linearity into AR framework. In TAR model, threshold variable (denoted by z_t) is defined and the domain of the threshold variable is split into non-trivial regimes each of which provides a different AR(p) model. Therefore, a piecewise linear model is defined for time-series where the approximation changes with

respect to the threshold variable. Formal definition for the TAR model is as follows;

$$X(t) = [1, X(t-1), \ldots, X(t-p)]^{(j)}[\beta_0, \beta_1, \ldots, \beta_p]^{(j)} + \sigma^{(j)}\xi(t) \text{ if; } r_{j-1} \leq z_t \leq rj \quad (3.5)$$

Note that, $0 \leq r_1 < r_2 < \ldots < r_k < \infty$ defines the boundaries for the regimes for non-negative threshold variable. Also, in TAR model the threshold variable is an exogenous variable which may be limiting for the model. In order to increase the flexibility of the TAR model. Self-exciting TAR (SETAR) models are proposed [106]. SETAR model includes an autoregressive threshold variable $X(t-d)$ instead of an exogenous variable, hence the name self-exciting. The positive integer d is called the delay parameter. This parameter selection provides flexibility in the choice of model parameters [102]. Both TAR and SETAR models are defined by two parameters; (p, k) where p is the autoregressive lag order and k is the number of regimes. The regimes defined in TAR and SETAR models both have abrupt changes in regime boundaries. Abrupt regime changes are unrealistic for many time-series applications even if correct regimes are learned. Subsequently, Smooth-transition AR (STAR) model is proposed which eliminates abrupt changes between regimes by introducing a transition variable and transition functions. The formal definition for a two regime STAR model can be found in Equation 3.6 where $0 \leq G(x) \leq 1$ is the transition function and z_t is the transition parameter. This way, as the transition function (which is tuned by the transition parameter) increases, the regime changes, but not abruptly.

$$\begin{aligned} X(t) = {} & [1, X(t-1), \ldots, X(t-p)]^{(1)}[\beta_0, \beta_1, \ldots, \beta_p]^{(1)}(1 - G(z_t)) \\ & + [1, X(t-1), \ldots, X(t-p)]^{(2)}[\beta_0, \beta_1, \ldots, \beta_p]^{(2)}(G(z_t)) \end{aligned} \quad (3.6)$$

In short, Autoregressive models assume that, the current state (and the future states) of a temporal process has a linear or non-linear relation with the past states. Even tough 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. mv-ARF method can also be classified as an AR model which incorporates several other features. The

non-linear decision boundaries that are provided by the tree based structure is able to capture non-linear interactions between past and present states. Also, time index is also incorporated as a key predictor in mv-ARF which makes the model capable of handling non-stationary behavior as well. Thus mv-ARF is in essence, a comprehensive AR model.

3.3. Tree Based Learning

Decision trees are established learners with convincing accuracy, that have a broad range of applications. The most notable and commonly used decision trees are Classification and Regression (CART) trees [107]. Some of the univariate decision trees have multivariate counterparts which are able to handle multiple response variables. CART trees have a multivariate extension named Multivariate Regression Trees (MRT) [108], which is utilized by mv-ARF.

Univariate Regression Trees (URT) aim to explain the variation of a singular response variable with explanotary variables which may be numeric and/or categorical [107]. The tree structure is build with top down induction, which include piecewise partitioning of the dataset into groups with similar values of the response variable [108]. Therefore, the decision boundary provided by a decision tree is piecewise linear. An example for decision tree is provided in Figure 3.4, which shows the boundary acquired to classify whether a patient is healthy or not by using two variables [109]. Generally, a split divides a node into two, but multi-way splits are also possible. The split that maximizes the homogeneity (or minimizes the impurity) is considered for each node. The measure of impurity for regression trees is the total sum of squares (of the response variable) over the node mean. A split aims to minimize the total sum of squares inside the two nodes formed by the split. The terminal, unsplited nodes are called leaves. Predicted value for a data point is simply the mean response variable value of the terminal node the new data point falls in. URT building algorithms are greedy. First, a large tree is built and then it's pruned back to the desired size. The size of the tree can be selected with cross-validation. MRT's are natural extensions from URT's and are obtained by changing the univariate response variable with a multivariate response

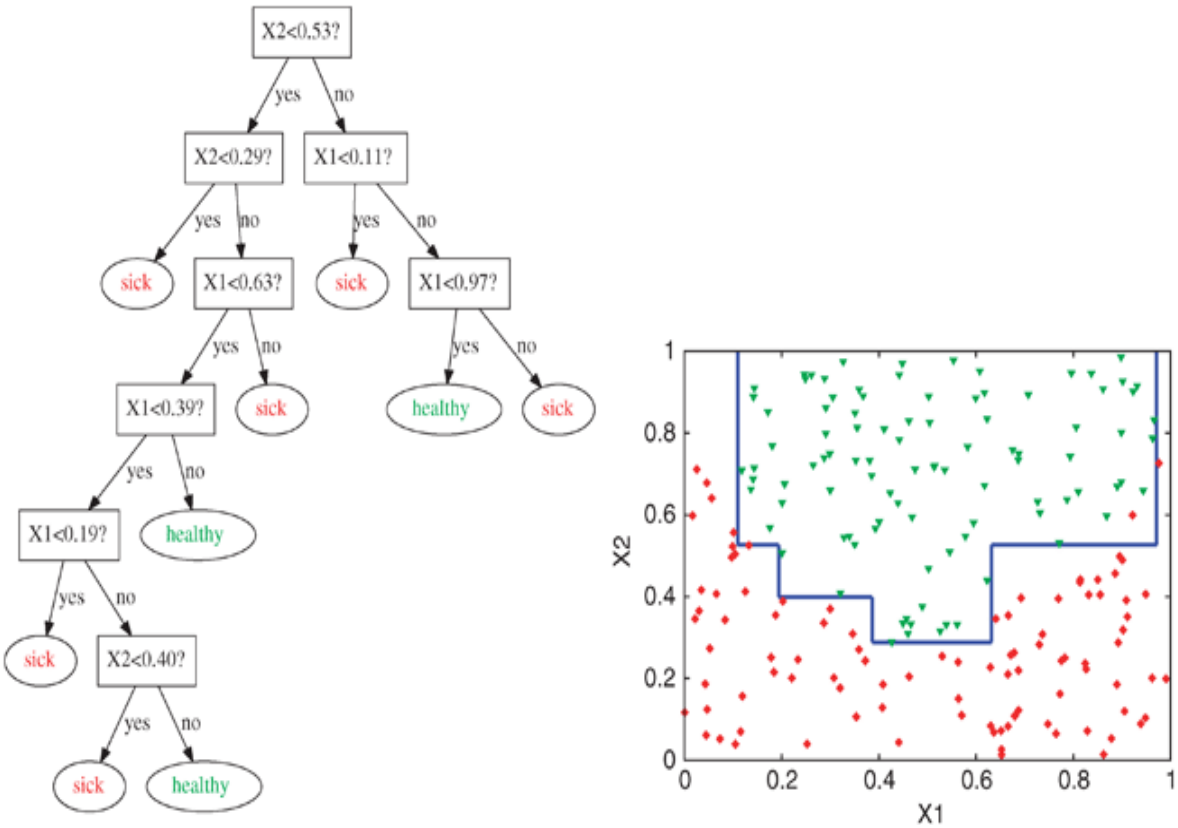

Figure 3.4. A sample classification tree (on the left) and the decision boundary defined by the tree (on the right).

and by redifining the impurity [108]. The extension of sum of squares in URT is defined as sum of squared distances [108], which is geometricallly the sum of distances of the response variables from the node centroid.

3.3.1. Random Forest Ensemble

The top down induction of decision trees, which is the most common strategy for learning decision trees, is an example of a greedy algorithm [110]. This kind of greedy approach with exhaustive splits have a potential to overfit with large datasets. Tree ensembles are used in order to enhance tree learners. An ensemble method is a combination of multiple models in order to achieve higher accuracy than the singular learners. An ensemble can be built in parallel where each learner is built independent of each other. Or in sequence, where each learner builds upon the output of the previous

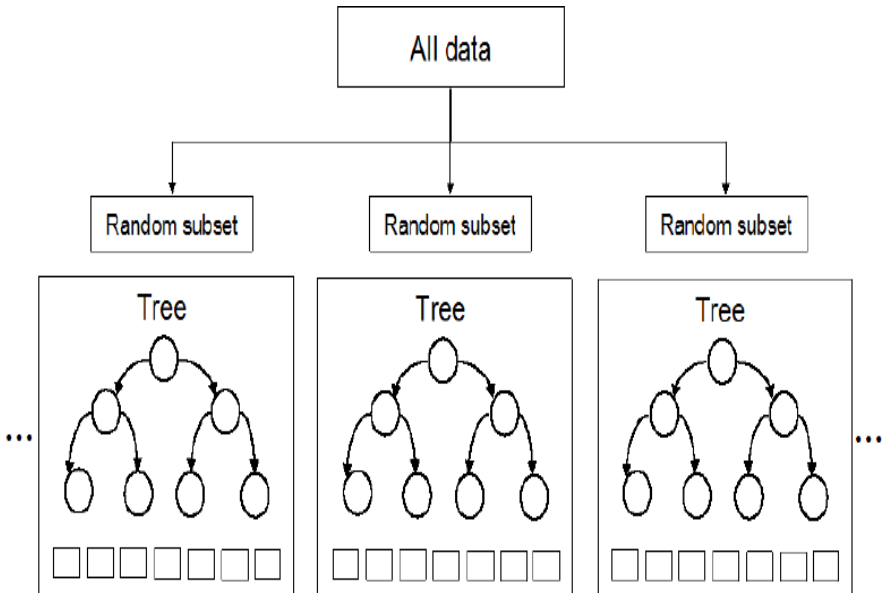

Figure 3.5. A simple illustration of the RF algorithm

one.

A random forest (RF) [111] model is a parallel ensemble of J decision trees, $\{h_j, j = 1, 2, \ldots, J\}$. In essence, RF algorithm combines simpler trees acquired from smaller datasets with random splits. RF is based on two methods; bagging and sampling. A simple illustration of RF algorithm can be found in Figure 3.5. Each h_j is built from a bootstrap sample. A bootstrap sample refers to random sampling of the dataset with replacement. As a result of bootstrapping, each observation is left out of approximately one thirds of the samples [111] and therefore is not included in one-thirds of the trees. For each tree, the observations that are left out of the sample are called out-of-bag (OOB) observations. The primary performance estimator of a RF model is the OOB error. OOB error in a random forest predictor is calculated as the sum of mean prediction error from each tree which an observation is out of bag, for each observation. The OOB error has proven to be a robust estimator for RF performance [111] and it is also utilized in parameter selection and representation stages of mv-ARF algorithm.

Furthermore, at each split of each tree, a random sub-sample of the features are considered. The customary size of the sampled features is, $\sqrt{M}$ for M features. Random sampling reduces the variance of the model (which is a problem for greedy decision trees) and also reduces 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;

$$Y = \frac{\sum_{h_j} \hat{y}_j}{J} \tag{3.7}$$

where $\hat{y}_j$ is the prediction from tree h_j. Also, RF algorithm is good with ordinal and nominal data and are able to capture non-linear interactions between features, which makes the classifier a desirable choice for time series data mining [17]. Tree based ensembles build with MRT's are utilized in mv-ARF.

3.4. Hidden Markov Models

Hidden Markov Models are well known generative models for modeling time-series that are most commonly used in speech recognition. HMM's are statistical models that are essentially Markov models where the states are latent. Given a discrete sequence X, 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. HMM's consists of the following; a set of states S, a set of output symbols (i.e the symbols that are present in the sequence X) Y, a set of transition probabilities which are the probabilities of going from one hidden state to the other and a set of emission probabilities for each state, which consists of probabilities of generating each output symbol from Y. The list of HMM parameters is provided in Table 3.4.

Given $X = X_1, X_2, \ldots, X_T$ and where $a_{i,j}$ and $b_j(X_t)$ are the transition and emission probabilities respectively, the mathematical formulation for the probability of

Table 3.1. Summary of HMM parameters.

Parameter	Explanation
$a_{i,j}$	Transition probability; probability of going from state i to state j.
$b_j(X_t)$	Emission probability; probability of instance X_t being generated from state j.
S	Set of latent states.
π_i	Initial state probability for hidden state i.
X	Input sequence.
Y	Set of output symbols.

the sequence at time $t \in T$, X_t, being generated from the HMM model λ is as follows;

$$P(X_t|\lambda) = \sum_{S} \prod_{t} a_{s_{t-1},s} b_{s_t}(X_t) \tag{3.8}$$

Complexity associated for calculating the probability of each path is large. Therefore, in most applications, probability for the most probable path is calculated using the Viterbi Algorithm which substantially decreases the computational complexity. Furthermore, discrete response variables are not a realistic assumption. Emission models are used to model the emission probabilities for each state in a continuous manner. For that purpose, a parametric emission model (a probability distribution) is fitted for each state. For multivariate datasets, the most common emission models are Multivariate Gaussian Model (Multivariate Normal) and Gaussian Mixture Models (GMM). GMM's are multiple Gaussian Models with different parameters and the emission distribution for each state is calculated as a linear combination of the separate Gaussian Models. 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. In cases where the states of a time-series is obvious or domain-related, intuitive determination of the hidden-states is not a problem. But for many settings this is not the case and the absence of a formal methodology to define the number of states becomes problematic. On the other hand, the emission probabilities and transition probabilities can be set with EM algorithms such as the Baum-Welch Algorithm which

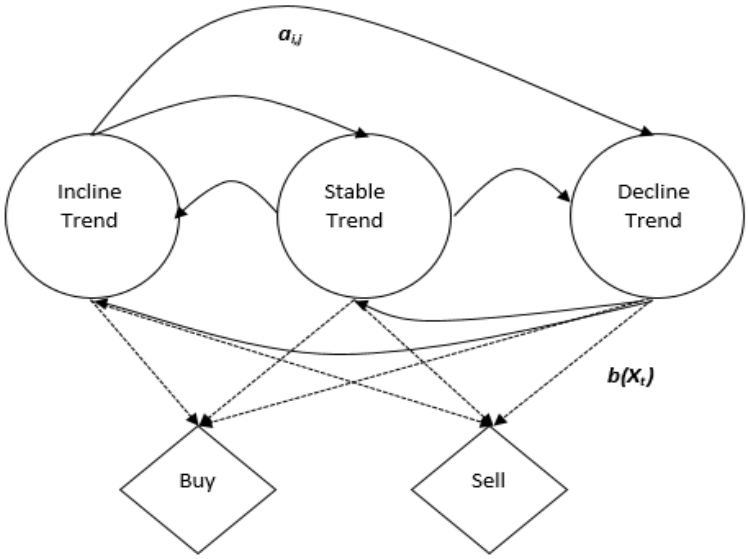

Figure 3.6. A HMM example built from trading sequence of a financial instrument.

is the most commonly used method in the applications. A sample illustration of a HMM is provided in Figure 3.6.

Moreover, HMM's can be used for classification of time-series. HMM's are similar to mv-ARF since they are both generative approaches. Thus, HMM classification is also considered for experimentation. A simple classification scheme where models are built for each class label and each unlabeled time-series is assigned to the class label with the model that yields the highest probability is provided in literature [112]. This approach is similar to the classification scheme of mv-ARF which uses supervised learners trained on prediction errors coming from models associated with each class label instead of using maximum probability. The HMM classification approach provided in this paper is the same with mv-ARF. Only difference is that probability score coming from the model representing each class is used for representation instead of the prediction error-based representation used in mv-ARF classification.

4. METHODOLOGY

The proposed method, which is denoted as mv-ARF, is actually a generative method which aims to provide a model based strategy to explain dynamic behavior of MTS. Note that, the primary output of mv-ARF methodology are the tree based ensemble learners which yields the error-based representation. Thus, mv-ARF can be used as a representative model or a forecasting model as well. But the capabilities of the model is tested within MTS classification framework. Therefore, for the sake of consistency, explanation of the model will be within the context of MTS classification.

4.1. Problem Definition

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 x M matrix;

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

A column m of time-series n is represented as;

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

There are N MTS in the training set, each of which is associated with a class label; $c^n \in \{1, 2, \ldots, C\}$ for $n = 1, 2, \ldots, N$. 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.

4.2. 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 x M matrix in which, each row represents an

instance. A training set of N MTS will result in $V_{NT \text{ x } M}$;

$$V_{NT \text{ x } M} = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_M^1 \\ x_1^2 & x_2^2 & \dots & x_M^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^n & x_2^n & \dots & x_M^n \\ \vdots & \vdots & \vdots & \vdots \\ x_1^N & x_2^N & \dots & x_M^N \end{bmatrix}$$

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; $\Theta_{NT \text{ x } (Mp+1)}$. 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. With this representation, each observation is defined also by several previous values of all the constituent attributes of the MTS. mv-ARF is based on the assumption that the present value of a time-series is effected by the past observations, which is the basic AR assumption. But, as a multi-response model current values are predicted simultaneously and relationships between current values of each attribute with past values of each attribute are also considered. A tree learner is able to capture the non-linear interactions between the current values and the past values if there is any. Also, time order of MTS is also 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. Missing values are handled by the tree learners. Lengths may differ across MTS but, lengths of all attributes are assumed to be the same within a single MTS.

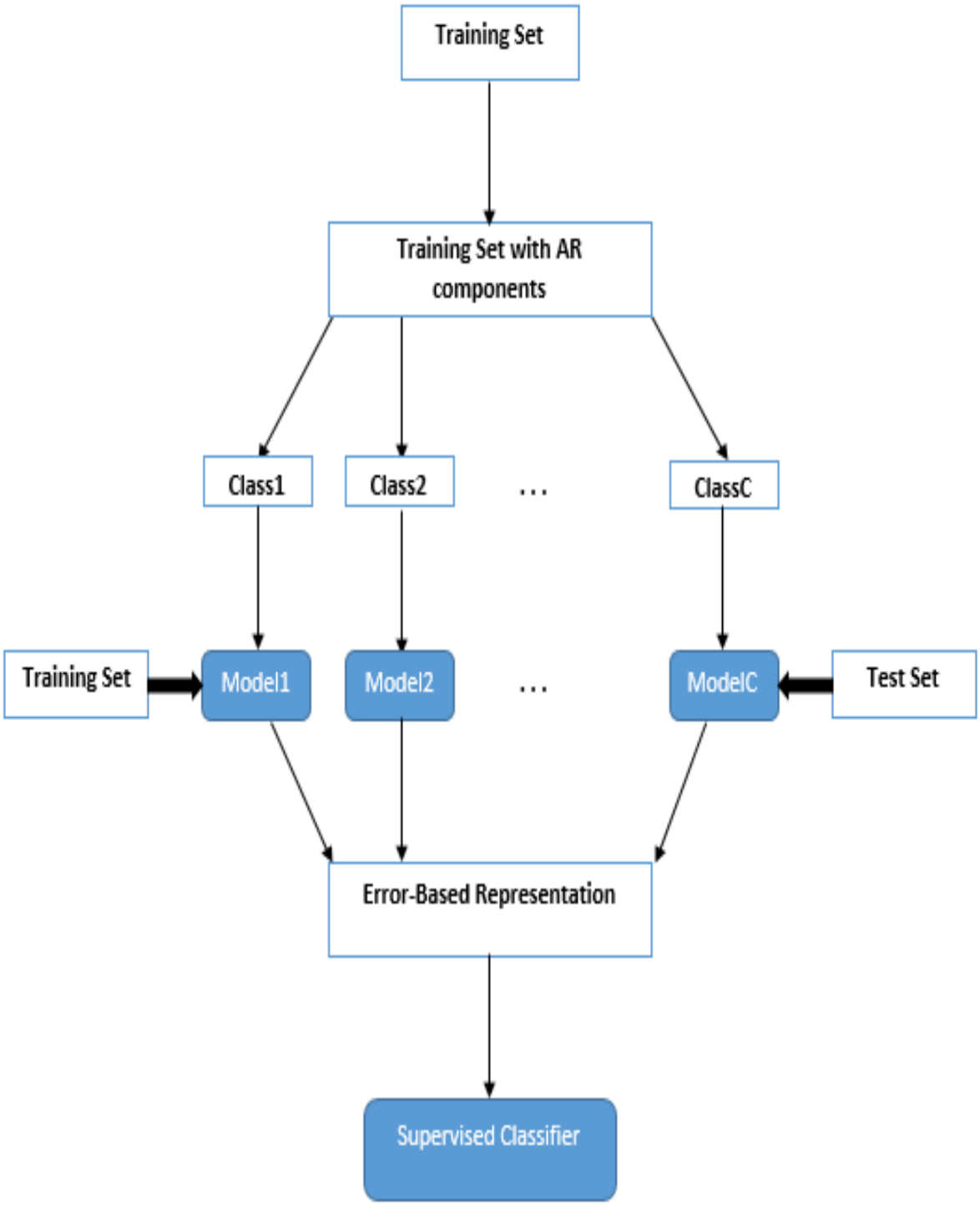

Figure 4.1. Flowchart summarizing mv-ARF classification methodology

mv-ARF model is a generative model which aims to capture the dynamics of the MTS rather than finding hard or soft decision boundaries. Therefore, mv-ARF is model based strategy for modeling of MTS. In this work, a novel MTS classifier using outputs of mv-ARF is proposed and the effectiveness of mv-ARF methodology is tested within MTS classification framework. Thus, for the sake of consistency, we will elaborate on the MTS classification scheme provided by mv-ARF. Table 4.2 provides an example for the autoregressive representation for MTS that is used in mv-ARF.

Table 4.1. MTS with two attributes. Two time-series represented in the table has the same class label. Four additional columns are generated for $p = 2$. The data is sampled from *ECG* dataset [2].

Series Index	Time Index	Attribute1	$Att1_{lag1}$	$Att1_{lag2}$	Attribute2	$Att2_{lag1}$	$Att2_{lag2}$
1	1	-134	-	-	60	-	-
1	2	-262	-134	-	-16	60	-
1	3	-158	-262	-134	-14	-16	60
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	51	-4	-2	14	40	14	12
2	1	175	-	-	-68	-	-
2	2	302	175	-	-138	-68	-
2	3	358	302	175	-11	-138	68
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2	63	86	72	80	30	26	10
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Classification approach is based on the assumption that, each MTS with identical class labels display similar dynamic behavior and similar interactions are observed inbetween 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 learners are trained on MTS associated with each class label. Thus, a total of C ensemble models are learned from MTS associated with C class

labels.

Each ensemble is built with a separate lag order, which means different number of autoregressive components are used in the model learning process, which makes the model capable of capturing differences between classes with respect to autoregressive behavior. The models built capture the generalized information within classes. The trees within the ensembles are MRT's with the attribute values as the response variables and lag values and time index as the predictor variables. The functional relationship achieved is both multivariate and autoregressive;

$$(x_1^n(t), x_2^n(t), \ldots, x_M^n(t)) = f(t, x_1^n(t-1), x_1^n(t-2), \ldots, x_1^n(t-p), \ldots, x_M^n(t-1), x_M^n(t-2), \ldots, x_M^n(t-p)) \tag{4.1}$$

Therefore, the time order of the data as well as the autoregressive behavior is incorporated in mv-ARF. 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 which provides a comprehensive foundation for modeling the dynamic behavior underlying MTS. Trees within the models are built in a breadth-first manner and with some parameter restrictions which will be discussed in further parts. Also, the high cardinality, which may be associated with $(Mp+1)$ columns when M or p is too large, is dealt by the tree based ensembles since it scales well with high dimensionality.

4.3. 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. In any model that is built, 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 x M matrix is obtained from each model, either by prediction or using the OOB values. The simple mean squared errors (MSE) calculations, from Equation 4.2, are made for predictions from each model.

$$\frac{1}{T}\sum_{t=1}^{t=T}(x_m^{observed}(t) - x_m^{predicted}(t))^2 \tag{4.2}$$

In the error based representation,which is 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 with M attributes, coming from C class labels, we map the dataset to the feature space $\mathbf{\Phi}_{N\times MC}$. A sample row from $\phi^n \in \mathbf{\Phi}$ looks like;

$$[MSE_1^1, MSE_2^1, \ldots, MSE_M^1, \ldots, MSE_1^C, MSE_2^C, \ldots, MSE_M^C] \tag{4.3}$$

Each row reprsenting a MTS is concatenated to acquire the final matrix representation. The class labels are also attached to the MTS. The final representation of the MTS is a Nx$(MC+1)$ matrix. A sample representation can be found in Table 4.2.

Table 4.2. A sample error based representation of a MTS classification problem with two attributes and two class labels.

Class	$\text{Error}_{\text{Class1}}^{\text{Attribute1}}$	$\text{Error}_{\text{Class1}}^{\text{Attribute2}}$	$\text{Error}_{\text{Class2}}^{\text{Attribute1}}$	$\text{Error}_{\text{Class2}}^{\text{Attribute2}}$
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
⋮	⋮	⋮	⋮	⋮

This way, mv-ARF provides a rectangular approximation for the MTS data by using the prediction errors acquired from multi-response tree-based ensemble learners and any supervised learner can be applied to this representation. The pseudo-code of the overall mv-ARF algorithm for generating the error-based representation is provided in Figure 4.3.

Form the $NT \times M$ matrix $\mathbf{V}$ as defined in Table. 4.2;
Compute the $NT \times (Mp + 1)$ feature space $\mathbf{\Theta}$;
Define the set of ensemble models $\mathbf{\Delta}$;
for $c \in C$ **do**
  Train the ensemble model δ_c on dataset θ_c, $\forall \delta_c \in \mathbf{\Delta}$ and $\forall \theta_c \subset \mathbf{\Theta}$;
end for
Define the $N \times MC$ error matrix $\mathbf{\Phi}$;
for $X^n \in \mathbf{\Theta}$ **do**
  for $\delta_c \in \mathbf{\Delta}$ **do**
    $\hat{X^n} \leftarrow$ prediction for X^n from δ_c ;
    if $class(X^n) \in c$ **then**
      $\hat{X^n} \leftarrow$ OOB prediction for X^n from δ_c ;
    end if
    Calculate ϕ_c^n as defined in Eq 4.2;
  end for
  Form vector $\phi^n \leftarrow [\phi_1^n, \ldots, \phi_c^n, \ldots, \phi_C^n]$
end for
Concatenate ϕ^n to form $\mathbf{\Phi}$, $\forall n \in N$;

Figure 4.2. Pseudo-code for mv-ARF algorithm.

4.4. Multivariate vs. Univariate Modeling Strategies

In order to demonstrate the effectiveness of multivariate models over univaraiate models (extensions of univariate models to MTS), univariate version of mv-ARF is also considered in experiments. The logic behind the univariate scheme is the same with mv-ARF. The difference is; in the univariate version, separate models are build for each class and for each attribute. The tree-based ensembles are learned from regression trees [107]. So, for a dataset with M attributes and C class labels, a total of MC models are built instead of the C models in the multivariate setting. This way, the error-based representation is acquired by simply combining error rates coming from separate univariate models for each feature instead of a single multi-response model that predicts all the features simultaneously. Everything else is the same with regular mv-ARF.

A synthetic dataset that is a noise dataset is generated with 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 within Class 1 and the third feature for MTS within Class 2. Moreover, the order of delay is also different for classes. Thus, a dataset with a simple multivariate AR relationship is provided. More detailed explanation of this dataset will be provided in later parts.

mv-ARF model is trained on the synthetic data for testing purposes. Figure 4.3 and Figure 4.4 illustrates the prediction errors for constituent features of synthetic MTS. Two figures are for the MTS coming from 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. Thus there are a total of six boxplots representing errors calculated for each feature from the two forests learned with mv-ARF. The figure shows that the errors of the second feature calculated from the model trained with MTS from the first class and the error of the third feature calculated from the model for the second class are considerably smaller than the rest. These are the lagged features for

the two classes which shows that mv-ARF successfully identified the lagged feature in each class.

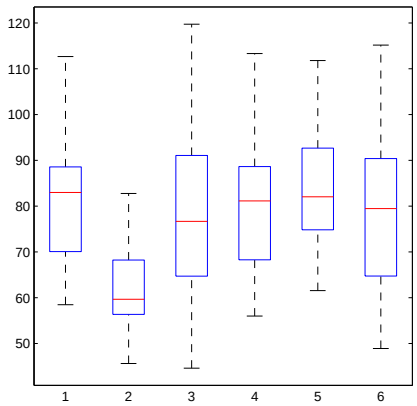

Figure 4.3. Boxplots of prediction errors calculated for each feature of the synthetic dataset for the first class.

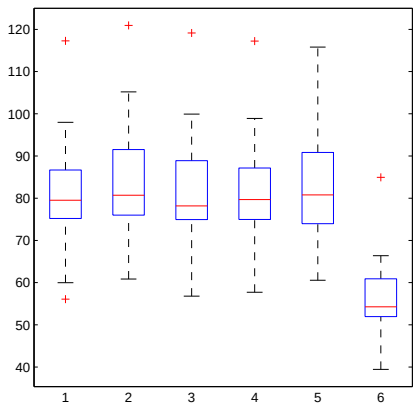

Figure 4.4. Prediction errors for the second class of the synthetic dataset.

Furthermore, a toy 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. Figure 4.5 and Figure 4.6 shows the mean squared errors calculated for the lagged feature of each class by both the multivariate and univariate versions of mv-ARF. Only the feature with a lagged relationship (second feature for Class 1 and third

feature for Class 2) is considered for each class since all other features are random noise (which is demonstrated in Figure 4.3 and Figure 4.4). Predictions for both training and test MTS are included in the figure. The figure shows that the MSE's acquired from the multivariate version are significantly smaller than the univariate version for each class label. This is due to the fact that the lagged feature is no different than the other noise features for univariate version which indicates that the univariate modeling scheme is not able to model this simple linear relation between features.

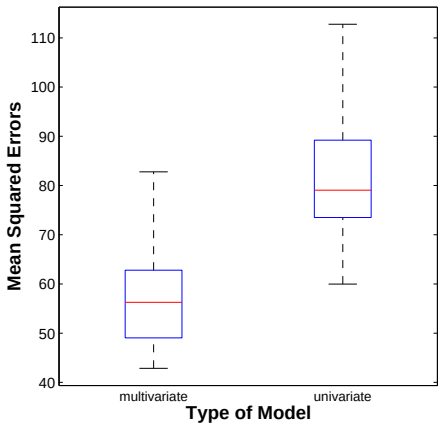

Figure 4.5. Box plot of MSE's calculated by multivariate and the univariate versions for the lagged feature for each time-series in Class1 (the second feature)

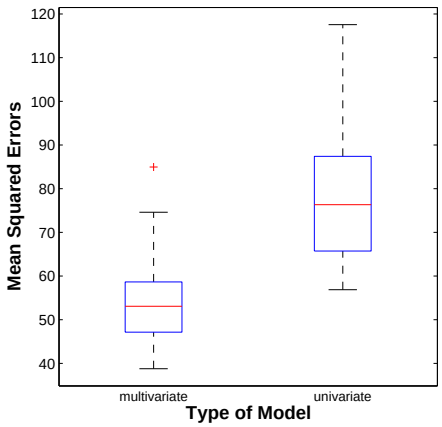

Figure 4.6. Box plot of MSE's calculated by multivariate and the univariate versions for the lagged feature for each time-series in Class2 (the third feature)

4.5. Algorithmic Complexity

The complexity associated with learning mv-ARF algorithm is mainly defined by learning the tree based ensembles for each class label. 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 (i.e number of time-series in the training set), J is the number of trees and D is the depth of each tree. An unpruned tree would have $O(log(n))$ complexity but mv-ARF limits the depth parameter for each tree. The operations that define the rate of the mv-ARF algorithm are; constructing the forests and acquiring predictions from each model. Thus, there is a time associated with prediction for each forest algorithm, which is $O(JDk)$ where k is the number of observations to predict (i.e test observations). But for complexity notation the determining process is always the model building step. Discussed operations are completed for each class label. Thus, number of class labels is also included in the algorithmic complexity of mv-ARF algorithm. So, given a set of N training MTS with M attributes, from C class labels where n time-series are associated with each class, the algorithmic complexity of mv-ARF algorithm of order p, forest size J built from MRT's of depth D is of order; $O(CJD(Mp)n)$. This is the algorithmic complexity for the operations mv-ARF algorithm performs to produce the error-based representation as an output. Note that, the complexity associated with mv-ARF is linear with respect to all parameters associated with the model buildng process, thus mv-ARF scales remarkably well with both ensembles with large trees and large datasets with long time-series and high number of attributes.

Moreover, testing complexity is determined by acquiring the representation and classification of a MTS. The representation is acquired by operating through C ensemble learners which is $O(CJDn)$ where n denotes the number of testing observations. Subsequently the classification complexity is dependent on the supervised classifier of choice. Any well known supervised classifier can be applied to the representation and most of the popular choices would also result in a linear complexity. Both training and testing complexity of mv-ARF algorithm are linear with respect to its parameters. Therefore, mv-ARF is scales well with large data and it is a good choice for handling

high number of attributes and long time-series.

Also, an overview of mv-ARF algorithm is provided as a flowchart in Figure 4.1. The flowchart underlines the fact that all the constituent operations of mv-ARF are parallel operations. Empirically, mv-ARF algorithm is an exceptionally fast algorithm and this parallel operations leaves room for further improvements with respect to computational complexity if additional improvements in empirical time is of importance.

5. EXPERIMENTS AND RESULTS

Eventough mv-ARF is a generative approach, its capabilities are tested within MTS classification framework. Note that the primary output of mv-ARF algorithm is the model based representation of each class label which aim to explain the dynamic behavior of the time-series constituent of that class. The classification scheme described in this work is tailored to utilize the strengths of this explanatory approach and to convert it to a model based MTS classifier. Furthermore, a high classification accuracy do not implicitly mean that the classifier provides a robust approach for modeling the MTS but that the classifier is able to differentiate MTS coming from different classes. On the other hand, a robust and comprehensive modeling approach can also identify inter-class differences. Thus, that kind of approach can also provide a robust classifier as well. This statement is also supported by the experimentation of this work. The effectiveness of the method is tested and accuracy results are reported for 19 MTS datasets from varying domians such as; motion recognition, medicine and speech recognition. The datasets, which are commonly used in MTS classification research are taken from; [2], [113], [114], [115]. The datasets have different characteristics such as different time-series lengths, different number of attributes and different number of classes. Properties of the datasets are illustrated in Table 5.1. The explanations and details of each dataset is also provided.

5.1. Datasets

In addition to real life datasets, two synthetic datasets are also designed to underline some key points of emphasis. Descriptions of each dataset is provided in order to introduce the scope of application of mv-ARF in terms of domains.

	# of Classes	# of Atrributes	Length	Training Dataset Size	Test Dataset Size	Source
Arabic Digits	10	13	4-93	6600	2200	[2]
AUSLAN	95	22	45-136	1140	1425	[2]
Character Trajectories	20	3	109-205	300	2558	[2]
CMU MOCAP	2	62	127-580	29	29	[113]
ECG	2	2	39-152	100	100	[115]
Japanese Vowels	9	12	7-29	270	370	[2]
KickvsPunch	2	62	274-841	16	10	[113]
LIBRAS	15	2	45	360	585	[2]
PenDigits	10	2	8	300	10692	[2]
Robot Failure						[2]
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	[1]
Wafer	2	6	104-198	298	896	[115]
WalkvsRun	2	62	128-1918	28	16	[113]
Digits Shape	4	2	30-98	24	16	[116]
Shapes	3	2	52-98	18	12	[116]

Table 5.1. Properties of MTS datasets; lengths, dataset sizes, number of attributes and number of classes for each dataset

5.1.1. Sythetic Datasets

5.1.1.1. Noise Data with Autoregressive Relationships. This dataset consists of MTS formed by random numbers generated from standard normal distribution. Dataset contains two classes with equal number of time-series for both test and training sets.

In summary, the set includes $N = 100, M = 10,$ and $T = 100$. The class differences are established with the simplest AR relationship inbetween different variables. One of the features of each MTS that belong to the first class are associated with a lag order of $p = 3$ whereas a (different) feature of each MTS belonging to the second class are associated with a lag order of $p = 5$. The mathematical notation for such a relationship is as follows;

$$\begin{aligned} X_m^n(t) = X_v^n(t-p) \text{ for; } & m, v \in \{1, 2, \ldots, M\} \\ & n \in \{1, 2, \ldots, N\} \\ & t \in \{1, 2, \ldots, T\} \\ & p > t \end{aligned} \tag{5.1}$$

In this setting, values for p, m and v are different for two classes which are the properties differentiating the classes. The remaining observations are random noise from standard normal distribution.

Reason for such a design is to further underline the strength of multi-response modeling against univariate modeling. Also, for datasets that are collected from real settings, the nature of the dynamic behavior of the time-series is always somewhat ambiguous, it is not possible to precisely know the relationships underlying the dynamics. Therefore, it is a non-trivial task to understand why or why not a model succeed or fail for such datasets with ambiguous nature. But the underlying dynamic relationship for this dataset is known and success or failure is determined by identifying the difference in a simple multivariate and autoregressive dynamic relationship. Therefore the accuracy results for this dataset provides valuable insight about the ability of an approach to model such dynamic behavior.

5.1.1.2. Multivariate Normal Distribution. Another synthetic dataset is constituent of time-series coming from separate multivariate normal distributions. Multivariate Normal Distribution is selected since it is a well known distribution for multivariate data with certain correlations. Thus, data generated from Multivariate Normal Dis-

tribution provides a nice dataset to demonstrate that mv-ARF is able to effectively capture the relationships with correlations inbetween features. The probability density function of multivariate normal distribution is provided in Equation 5.2. Two classes of MTS are generated from two multivariate normal distributions with different covariance matrices and mean vectors. Both covariance matrices and mean vectors are generated randomly. Furthermore, each MTS is of $M = 3$ amd $T = 350$. Each class label consists of 25 test and 25 training time-series which adds up to a total of 100 time-series.

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{5.2}$$

5.1.2. Real Datasets

5.1.2.1. Spoken Arabic Digits Dataset. The dataset was first introduced by a study on speech recognition [117]. The dataset consists of time-series of mel-frequency spectrum coefficients of spoken Arabic digits. It contains a total of 8800 time-series acquired from 44 male and 44 female native Arabic speakers between the ages 18 and 40. A total of 10 digits are recorded 10 times from each speaker which adds up to a total of 8800 time-series of varying lengths. The set is divided into 6600 training series and 2200 test time-series.

5.1.2.2. Australian Sign Language Dataset. The dataset was introduced in a PhD. thesis on classification of multivariate temporal data [112]. The data is collected from a volunteer native Australian sign language (AUSLAN) user. It is collected using Fifth Dimension Technologies gloves and magnetic point trackers for each hand [2] which generates a total of 11 features for each hand including xyz position of the fingers and finger bend measures. A total of 2565 time-series representing 27 examples of 95 signs of the language are included. A sample image illustrating a set of signs can be found in Figure 5.1[4] . The dataset is divided into 1140 training and 1425 test time-series.

[4]The image is acquired from [2]

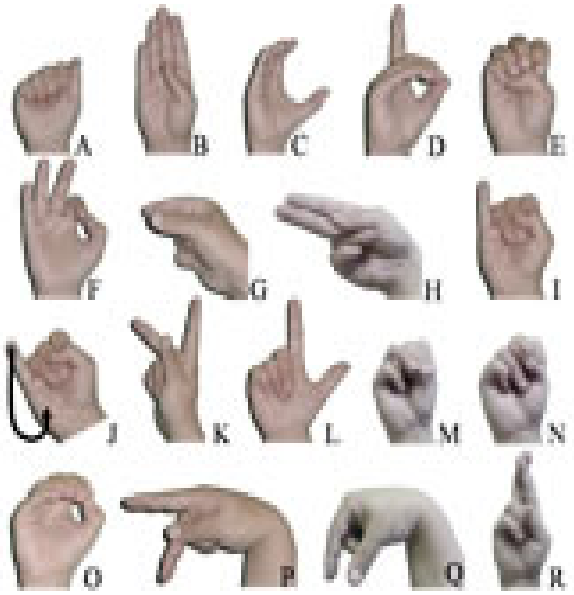

Figure 5.1. Example of some of the classes from AUSLAN dataset

5.1.2.3. Character Trajectories Dataset. The dataset was introduced for a study of primitive extraction using HMM learners [118]. The data is the collection of pen tip trajectories of handwritten characters. Each trajectory is a time-series with three dimensional pen tip velocity with differing lengths according to the length of the character that is being written. All time-series are sampled from the same writer for the purpose of primitive extraction and only characters with single pen-down segments are considered. In total, 2858 samples are collected from 20 characters. For each character 15 training samples are acquired which makes a total of 300 training time-series and 2558 test time-series.

5.1.2.4. CMU Motion Capture Database. These datasets are acquired from CMU's free to use motion capture database [113]. The database contains time-series consisting of position information of 62 joints while performing various physical activities. The length of the time-series changes with respect to the length of the motion being captured. While there are time-series acquired from a total of 144 subjects, the motions are not classified by only subject but they are classified by the motion type as well. A total of three datasets are used from CMU's database:

- Subject 16: The entire motions from subject 16 is used since it is the one of the very few subjects to have sufficient number of examples. The task is to differentiate whether the subject is walking or running with 29 training and 29 test time-series.
- Walking vs Running: Task of differentiating walking and running motions acquired from various subjects with a total of 28 training time-series and 10 test time-series.
- Kicking vs Punching Task of differentiating kicking and punching motions acquired from various subjects with a total of 16 training time-series and 10 test time-series.

5.1.2.5. ECG Dataset. ECG dataset consists of time-series that are combination of measurements from two electrodes during heartbeats. Length of each time-series depends on the length of the heartbeat. Each heartbeat is labelled as normal or abnormal. The task is to differentiate abnormal heartbeats which are indicators of a cardiac disorder called superventricular premature heartbeat.

5.1.2.6. Japanese Vowels Dataset. This dataset is collected for testing a MTS classification method described in [119]. The dataset contains time-series data acquired from utterances of two Japanese vowels from nine male speakers. Each utterance is a discrete time-series with 12 linear predictive coding coefficients. The lengths of time-series varies with respect to the lengths of utterances. A total of 640 time-series are recorded where there are 270 training time-series for 30 utterances from each speaker. Two methods are tested on this dataset [119], with prediction accuracy results of 0.068 and 0.069 are reported.

5.1.2.7. Brazilian Sign Language Dataset. This dataset is generated for testing of the method proposed in [120]. The dataset consists of examples of 15 signs from the official Brazilian sign language (LIBRAS) where the hand movement of each sign is represented as a two dimensional time-series. A video pre-processing has been applied for the dataset where 45 frames of each video is selected. Thus, the length of each

time-series is 45. A total of 24 examples for each of the signs are used as training time-series and a total of 585 time-series are used for testing.

5.1.2.8. Pen Based Recognition of Handwritten Digits Dataset. The dataset was provided for a study of handwritten digit recognition [121]. The digit database is collected using a tablet that records x and y and pressure information from samples written by 44 writers. There are a total of 10 classes for each digit and only the coordinate values are included for classification. The dataset is represented as constant length feature vectors in order to work with the proposed classifier [121]. Therefore the points are resampled and a constant length of 8 is used for each time-series.

5.1.2.9. Robot Execution Failures Dataset. The dataset consists of force torque measurements of a robot after a failure is detected [2]. Each failure is represented by three dimensional force and torque measurements. Measured sequences are of length 15. Each failure is represented as an integer valued time-series. Five different failures are introduced within this dataset:

- LP1: A total of 4 types of failures in approach to grasp position.
- LP2: A total of 5 types of failures in transfer of a part.
- LP3: A total of 4 different positions of part after transfer failure.
- LP4: A total of 3 types of failure in approach to ungrasp position.
- LP5: A total of 5 types failures in motion with part.

5.1.2.10. Gesture Recognition. The dataset is acquired from uWave Gesture Library which consists of over 4000 samples of gestures acquired from eight users [1]. The dataset is generated using *xyz* coordinate readings from a single, three axis accelerometer. A total of eight gestures are sampled which forms the classes for the classification task. The illustration of the gestures can be found in Figure 5.2. The data is preprocessed and each time-series is standardized with a resampled length of 315. In our experiments, 112 training time-series from each class and 3582 test time-series are used.

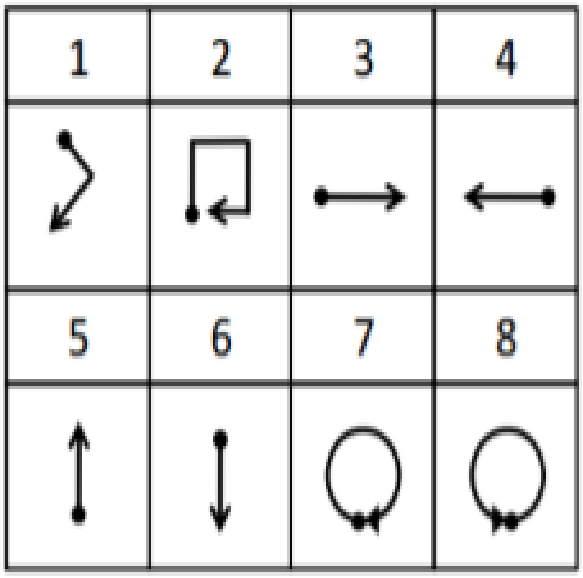

Figure 5.2. Gestures from Uwave dataset [1].

5.1.2.11. Wafer Dataset. The Wafer dataset includes time-series that are combination of sequential measurements from vacuum-chamber sensors while a silicon wafer is in etching process during manufacturing of semiconductors. There are a total of six sensors. Each wafer is labeled normal or abnormal. The abnormal wafers may indicate a range of problems commonly encountered during semiconductor manufacturing [115].

5.1.2.12. Shape Clustering Datasets. The datasets are generated for a HMM based clustering study [116]. The time-series are xy coordinates recorded by manual mouse clickings. Two datasets are used from this work:

- Shapes: Time-series dataset of 2-D coordinate data which consists of three shapes with 18 training time-series and 12 test time-series.
- Digits Shape: Time-series dataset of 2-D coordinate data which consists of three different digit shape with 24 training time-series and 16 test time-series.

5.2. Parameter Setting

Parameter setting process of mv-ARF algorithm is straightforward. Apart from the forest parameters, the only parameter to be considered is the autoregressive lag order p. The lag order selection is through OOB accuracy comparison among different candidate p values. For each class, a tree based ensemble is built with alternative lag values and OOB scores from different lag values are acquired. Then 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, so this parameter selection doesn't add much empirical complexity. But if small time increases 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. Empirical analysis of J on OOB scores for *Uwave* dataset are illustrated in Figure 5.3. The provided OOB score is;

$$(1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{5.3}$$

which is similar to the R^2 notation provided in Equation 5.4. It can be seen that the increase in the OOB score becomes negligible after sufficient number of trees for all classes. Thus, classification accuracy is not affected by J after a sufficient number of MRT's. mv-ARF is also robust to D if considered individually. Reason for setting D specifically is to avoid overfitting. The algorithm divides the training set and builds comprehensive learners with respect to class labels. This partitioning of the training set increases the chances of a possible overfitting problem, especially for small sets and datasets with high number of class labels. D is considered as a parameter to control overfitting. The trees are built with no pruning, bootstrap subsets and with random

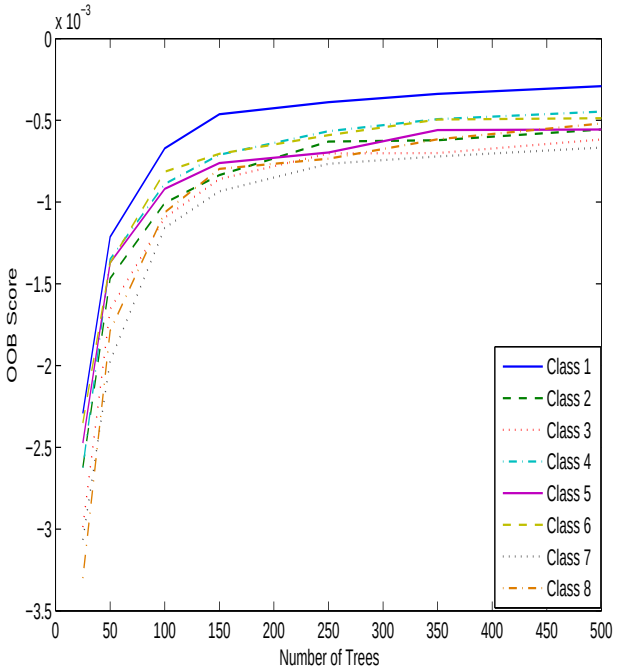

Figure 5.3. OOB scores of each class with increasing J values. (15 replications)

sampled features where the total number of sampled features is the square root of M.

$$R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2} \tag{5.4}$$

In the experiments, the resuts are reported with the AR order p is selected from $[1, 2, 3, 4, 6, 8, 10, 12, 15, 17, 20]$, the number of MRT's J is set to 100 and D is set to 10. Also, a standardization is applied to each feature with zero mean and unit deviation, in order to avoid any complications and bias in multi-response MSE calculations due to scale differences. Furthermore, after the error-based representation described in Section 3 is achieved, the final class assignment is made with a simple Random Forest [111] algorithm with 200 trees. RF algorithm is used to capture the non-linear relations inbetween features and classes that are represented as errors rates but, the final labeling can also be achieved by any other supervised classifier. An alternative scenario where a k-NN classifier is built on the error-based representation instead of a RF classifier is also considered for analysis. The number of neighbors (k) is selected with cross-validation.

For comparison, classification results for each dataset is acquired from the univariate setting as well. For the univariate version, p, J and D is selected with the same methodology as the multivariate version. Two scenarios with RF classifier and k-NN classifier is also considered for the univariate version of mv-ARF. Same as the multivariate version, RF algorithm with 200 trees is used for classification after the error-based representation is acquired. It is hard to provide concrete proof for the superiority of multivariate modeling strategies against univariate strategies in MTS modeling because it is not sensible to compare two approaches if the underlying modeling principals are different. The results acquired from mv-ARF with two different modeling strategies provide a unique opportunity to observe the difference between univariate modeling against multivariate modeling since we are able to provide results for both multivariate and univariate modeling strategies with the same underlying principals.

Results from other approaches are evaluated with their parameter settings for comparison and discussion. MTS classifiers with AR components are selected for evaluation. [27] reports MTS classification results from LPS methodology, which utilizes a strategy to learn representation based on local autopatterns acquired with AR components. Similarly, [26] provides classification results acquired from an AR kernel. Experiments with Autoregressive Kernels are conducted with lag order set to 5 and alpha parameter set to 0.5. Furthermore, since it has become a benchmark for time-series classification, DTW with 1-NN classifier results are also provided for comparison. The standardizaation 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 distance of the 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 number of hidden states for each class is selected with the same approach that is discussed for the selection of lag order. Classification results are reported for both scenarios where a RF classifier and a k-NN classifier is learned on the log-likelihood features.

5.3. Classification Accuracy

Applications of both univariate and multivariate versions of mv-ARF in addition to the HMM classifiers are implemented in Python [122] software. The implementations are built using 'ExtraTreesRegressor', 'RandomForestClassifier', 'KNeighborsClassifier' and 'hmm' packages from the skicit-learn library [123]. Source code for multivariate and univariate versions of mv-ARF can be found in [124]. Furthermore, experiments with Autoregressive Kernels are conducted by using the source code in MATLAB [125] format, which is provided in [126]. LPS application is provided in R software [127] which can be found in [128]. Moreover, the results of DTW are acquired from [17].

The names $mv-ARF$ 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 classifier scenario is denoted by mv-ARF^{uni} and the k-NN classifier scenario for the univariate version is denoted by $mv-ARF^{uni}_{knn}$. Results reported for all mv-ARF related methods are the mean accuracy results for 25 replications with the described parameter settings. Also, test and training MTS with lengths (T) smaller than the lag value (p) are left out and are not used in model learning and are not considered for classification. Results for DTW with 1-NN classifier and LPS methodology are reported under DTW and LPS respectively. The results for LPS are the mean accuracy results for ten replications. Furthermore, classification results of Autoregressive Kernels method are denoted as $ARkernel$. Moreover, HMM denotes the scenario where HMM models are combined with RF classifiers and HMM_{knn} denotes the scenario with the k-NN learner. The results for both HMM based methods are the mean accuracy results over 25 replications.

Classification results from the synthetic datasets can be found in Table 5.2. Note that, only HMM and mv-ARF based classification is applied on synthetic data since synthetic data is produced in order to demonstrate the modeling capabilities and the considered models generative applications in which the classification is based on output from a modeling scheme. The results show that, each model, regardless of the classification scenario, is able to classify MTS coming from Multivariate Normal distribution

with two different parameter sets with perfect accuracy. Neverheless, it is misleading to suggest that each approach is able to model the multivariate relationships based on the classification accuracy. A simple linear relationship is introduced by the covariances within Multivariate Normal Distribution whereas both HMM and mv-ARF methodologies aim to capture non-linear interactions inbetween constituent features. Thus, the simple linear interactions can be captured even by the univariate approach as exogenous coefficients. Furthermore, the classification scheme within each approach is designed to differentiate inter-class differences based on error-based representation or likelihood scores and the accuracy does not offer any insight about the dynamic behavior of the data. Moreover, a good modeling of the data can provide a robust classification scheme but vice-versa may not always be true.

On the other hand, for the noisy dataset there are extremely large differences in accuracy. Both classification scenarios which utilizes mv-ARF performed much better than both scenarios with HMM and $mv-ARF^{uni}$. This is due to the fact that both HMM and the univariate version are not able to capture a simple multivariate autoregressive relationship. HMM is not an AR model and each state of the HMM is modeled by a Gaussian Distribution which is not able to capture the relationship with a past observation. The univariate version of mv-ARF however is able to handle relationship with past observations but it is not able to differentiate the lagged relationship within features generated from identical distributions. Another important point is the parameter setting and computational durations between $mv-ARF$ and $mv-ARF^{uni}$. The multivariate version empirically requires less than half the time to train and less than quarter of the time to set the parameters when compared to the univariate version. Both of the synthetic datasets are small datasets; the multivariate normal dataset is of $T = 350$ and $M = 3$ whereas the noisy dataset is of $T = 100$ and $M = 10$. There is not a significant difference in the multivariate normal dataset since the size of the dataset is dominated by T rather than M. As the number of features increases the difference becomes more significant since the univariate version includes multiple operations for each feature. This difference in efficiency is also valid for real life datasets and the differences in computational times are much more significant for large datasets.

Table 5.2. Classification results of synthetic datasets from mv-ARF and HMM based methods.

	Multivariate Normal Distribution			Noise with Lag		
	Parameter Setting Duration (in secs)	Computational Duration (in secs)	Classification Error	Parameter Setting Duration (in secs)	Computational Duration (in secs)	Classification Error
$mv-ARF$	0.895	1.830	0.000	0.797	1.209	0.039
$mv-ARF_{knn}$	0.835	1.669	0.000	0.795	1.128	0.104
$mv-ARF^{uni}$	3.042	1.403	0.000	4.624	2.717	0.413
$mv-ARF^{uni}_{knn}$	3.165	1.499	0.000	4.797	2.903	0.476
HMM	5.986	1.688	0.000	2.191	0.459	0.597
HMM_{knn}	5.209	1.687	0.000	2.362	0.458	0.600

As for the real life datasets, test data accuracy results for seven methods are illustrated in Table 5.3. mv-ARF performs better or as good for entire datasets when compared to other methods. Either mv-ARF or AR Kernel outperforms 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 logical assumption for motion recognition data. Autoregressive Kernel method slightly outperforms others in sensor reading data such as *Wafer* or *Arabic Digits* where mv-ARF outperforms AR Kernel in motion recognition data. Eventough 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. Thus, when standardization is applied it makes it harder for DTW to differentiate the attributes. No such problem arises for mv-ARF since each attribute is considered

simultaneously.

Table 5.3. Classification Accuracy Results for MTS datasets

	$mv-ARF$	$mv-ARF_{knn}$	$mv-ARF^{uni}$	$mv-ARF^{uni}_{knn}$	LPS	DTW	$ARkernel$	HMM	HMM_{knn}
Arabic Digits	0.039	0.080	0.096	0.199	0.029	0.092	0.012*	0.023	0.037
AUSLAN	0.077*	0.383	0.079	0.389	0.246	0.238	0.082	0.439	0.610
Character Trajectories	0.070	0.194	0.089	0.116	0.035	0.033*	0.100	0.086	0.089
CMU MOCAP	0.000*	0.068	0.034	0.068	0.000*	0.0069	0.000*	0.009	0.020
ECG	0.220	0.246	0.322	0.284	0.180	0.150	0.180	0.156	0.142*
Japanese Vowels	0.045	0.174	0.058	0.224	0.049	0.351	0.016*	0.049	0.100
KickvsPunch	0.012	0.300	0.012	0.300	0.100	0.100	0.000*	0.352	0.300
LIBRAS	0.055*	0.273	0.065	0.066	0.097	0.200	0.073	0.083	0.073
PenDigits	0.074	0.123	0.095	0.099	0.099	0.075	0.048*	0.089	0.161
LP1	0.240	0.220	0.154	0.136	0.138	0.280	0.140*	0.174	0.176
LP2	0.305*	0.426	0.314	0.453	0.296	0.467	0.366	0.278	0.406
LP3	0.208	0.365	0.234	0.193*	0.280	0.500	0.433	0.404	0.260
LP4	0.085	0.236	0.096	0.229	0.690	0.106	0.040	0.035*	0.162
LP5	0.292*	0.438	0.299	0.306	0.020	0.480	0.530	0.324	0.422
Uwave	0.056	0.078	0.058	0.089	0.020*	0.071	0.096	0.050	0.054
Wafer	0.061	0.088	0.076	0.108	0.038	0.040	0.032*	0.064	0.075
WalkvsRun	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*	0.0625	0.0625
Shapes	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*

The effectiveness of multivariate methods over univariate methods can be shown by the comparison between mv-ARF and mv-ARF$_{uni}$. Multivariate method outperforms mv-ARF$_{uni}$ 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 since the only difference between the methods is that mv-ARF treats attributes simultaneously where mv-ARF$_{uni}$ treats them separately and interactions inbetween attributes are neglected in latter. Therefore, the difference would increase in more complex data that have more dependency between attributes. Furthermore, the empirical complexity of the multivariate 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 $\approx$ 330 seconds whereas training mv-ARF$_{uni}$ takes $\approx$ 1150 seconds. Thus, multivariate methodology provides a better approach with respect to both classification accuracy and empirical complexity.

5.4. Sensitivity Analysis

Three parameters are considered in the parameter setting of mv-ARF. The autoregressive lag order (p) and forest parameters (J and D). Lag order is the only parameter to be dealt with, because mv-ARF is robust to the forest parameters when considered individually. The algorithm is robust when J and D are selected sufficiently high. 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 mv-ARF to its model 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.

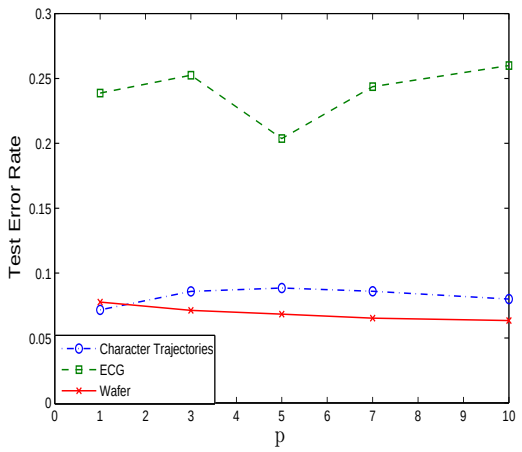

Figure 5.4. Sensitivity results for different values of p (10 replications)

Average test data classification error rates are illustrated in Figure 5.4,5.5, 5.6. For the experimentation with p; D and J are set to 10 and 100 respectively. The results of the experiments are shown in Figure 5.4. 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 row, increasing p too much results in loss of valuable information especially for short MTS. For example,

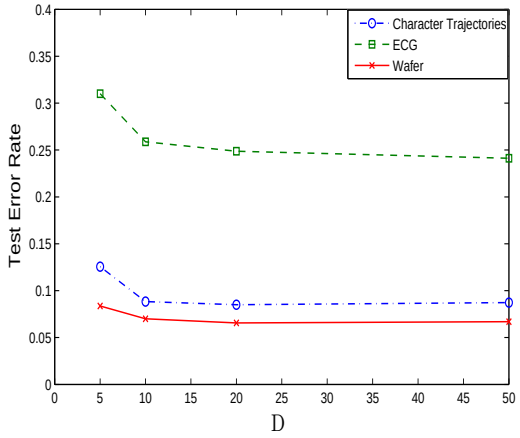

Figure 5.5. Sensitivity results for different values of D (10 replications)

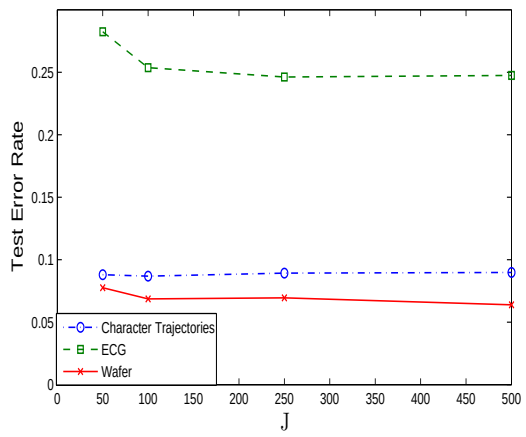

Figure 5.6. Sensitivity results for different values of J (10 replications)

time-series within *Pen Digits* dataset are of length = 8. For even $p = 4$, this means loss of half the observations for each time-series. Additionally, each lag order introduces extra variables to the tree-based ensemble. 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 nonobservant. That is the primary reason behind the parameter selection scheme for p that is discussed in earlier sections.

Figure 5.6 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. Therefore, limiting J to a sufficient number of trees would be better for computational complexity. Moreover, Figure 5.5 shows the test-error sensitivity to depth parameter D. D can be dropped by growing full trees with vigorous splits. Figures show that the model is robust to D when it is selected sufficiently high. But for mv-ARF, the prediction accuracies of the initial ensemble learners are essential for the overall classification success, overfitting in the prediction stage potentially causes decrease in the test accuracy. Especially for large MTS with high number of attributes and many class labels, training data is divided into considerably smaller sets with the same number of attributes and this causes a potential risk of overfitting. Thus, growing full trees within forests increases that risk even more. 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.

5.5. Complexity Analysis

mv-ARF is implemented in Python [122] software and the experiments used a MS operating system with 12 GB RAM and a processor with four physical cores (i7-4720HQ, 3.60 Ghz). Even tough the CPU is able to run on eight parallel threads, the

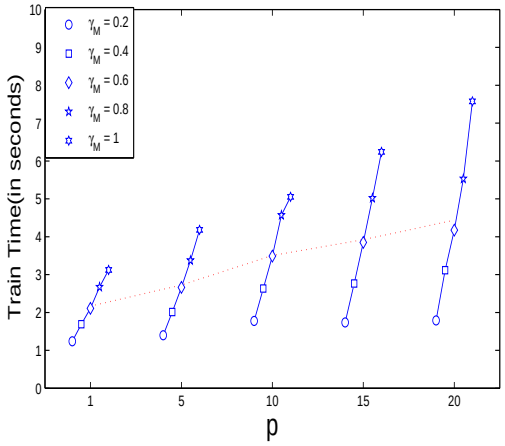

Figure 5.7. Empirical training complexity for differing p and M (10 replications)

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. Considered training time is simply the elapsed time to build the ensemble learners for each class label and to calculate the MSE values to achieve the error-based representation. The calculated 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 Figure 5.9, 5.7, 5.8 and Figure 5.12, 5.10, 5.11 respectively.

Training time results for γM and p (where $J = 100, D = 10$) is given in Figure 5.7. A linear increase with M and p is expected and 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 with

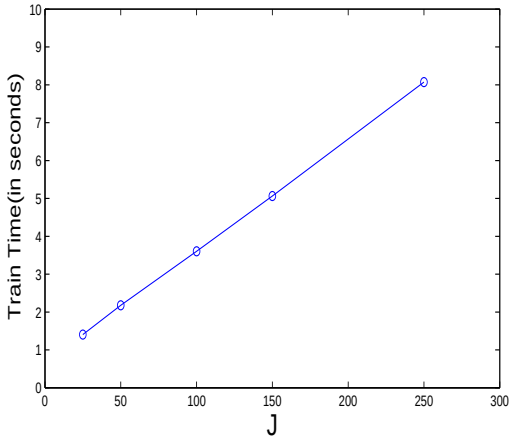

Figure 5.8. Empirical training complexity for differing J (10 replications)

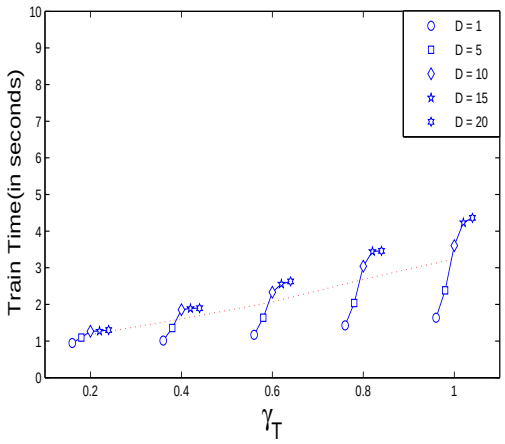

Figure 5.9. Empirical training complexity for differing D and T (10 replications)

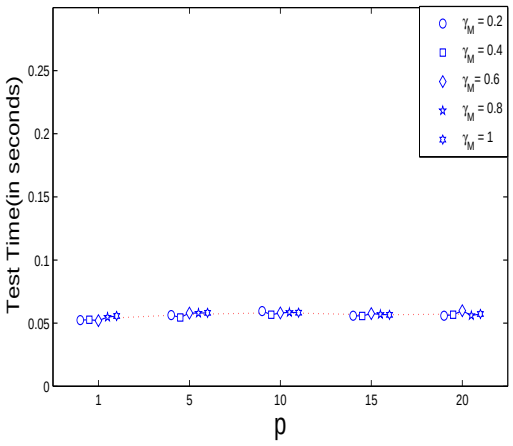

Figure 5.10. Empirical testing complexity for differing p and M (10 replications)

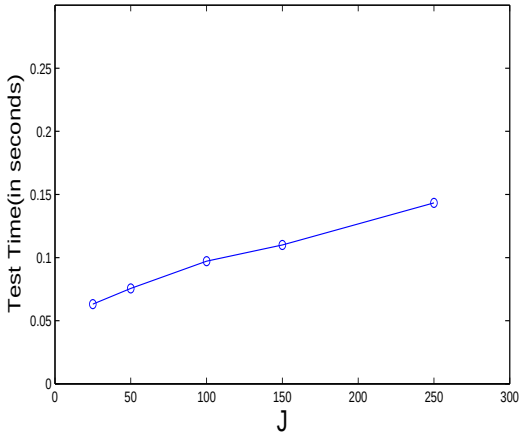

Figure 5.11. Empirical testing complexity for differing J (10 replications)

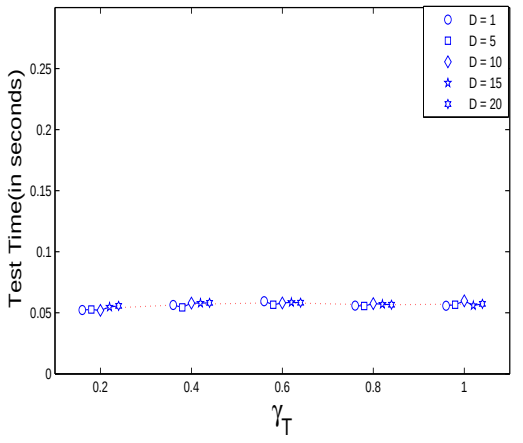

Figure 5.12. Empirical testing complexity for differing T and D (10 replications)

respect to p and M is provided in Figure 5.10. The behavior is consistent with the algorithmic complexity discussed in earlier sections. The selected classifier after the representation is RF algorithm which has a linear relationship with respect to M and a small linear increase with increasing M can be observed in the illustration.

Furthermore, Figure 5.8 and Figure 5.9 illustrates the empirical analysis for training time with respect to parameters J, D and T respectively. Figure 5.9 illustrates a linear increase in training time with respect to both D and T that is expected from a tree-based ensemble. 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 Figure 5.11 and Figure 5.12 respectively. A linear increase with respect to J shows that the model is consistent with the algorithmic complexity since a linear increase in testing time with CJ is expected with increasing J, which results in a large impact. 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. Similarly, both D and T is consistent with the algorithmic complexity. 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.

Furthermore, 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. Figure 5.13 shows the changes in training time of mv-ARF with increasing number of cores used in parallel for two large datasets. Eventough there is an unmistakable decrease in training time, it is not

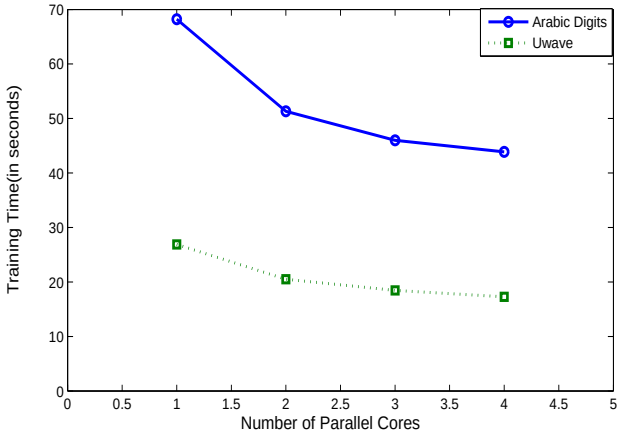

Figure 5.13. Training times for *Arabic Digits* and *Uwave* datasets with paralellized operations. (15 replications)

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

A similar complexity analysis is made for AR kernel method since the discussion on complexity is limited in [26]. Two parameters (α and p) are considered in the complexity experiments of AR Kernel method since [26] considers only the lag order and α as parameters that needs to be fine-tuned in the model. Apart from the model parameters, MTS parameters M and T are considered with ratio parameter $\gamma = [0.2, 0.4, 0.6, 0.8, 1.0]$. The considered values for the model parameters are; $\alpha = [0.1, 0.3, 0.5, 0.7, 0.9]$ and $p = [1, 5, 10, 15, 20]$. Same as the complexity analysis for mv-ARF, *Kick vs Punch* dataset is used for the analysis. The empirical complexity for training is calculated as the elapsed time for calculating the distance matrix. Figure 5.14, 5.15 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 param-

eter 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 [26] is to provide a distance measure which can efficiently handle structured data. Figure 5.16, 5.17 illustrates the results empirical test complexity. The considered 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.

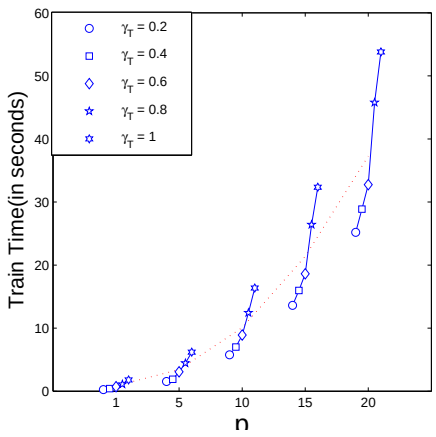

Figure 5.14. Training complexity with respect to the selected values of p and γ_T

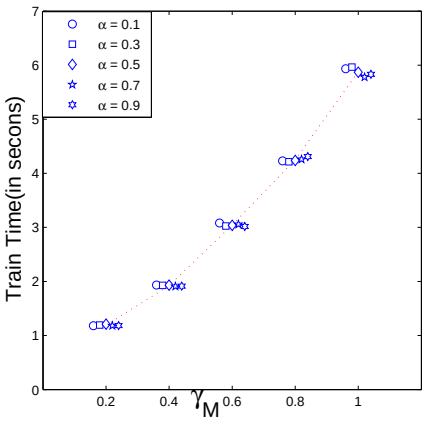

Figure 5.15. Training complexity with respect to the selected values of α and γ_M

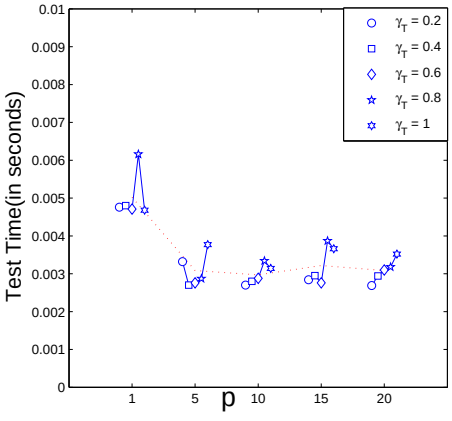

Figure 5.16. Test complexity with respect to the selected values of p and γ_T

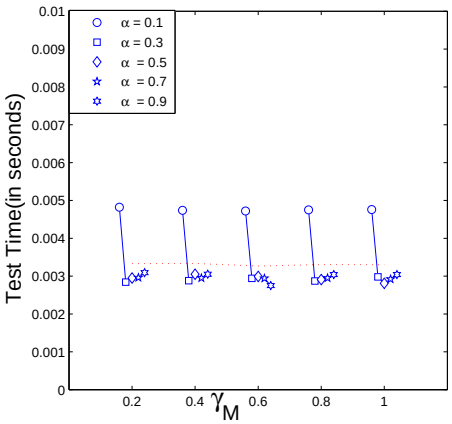

Figure 5.17. Test complexity with respect to the selected values of α and γ_M

6. CONCLUSION AND FUTURE WORK

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 model 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 ensemble 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 attribute prediction 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 multivariate version yields considerably better results in accuracy as well as computational efficiency. 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.

Lastly, there are some possible research directions which this research can lead up to. There may be some future research on enhancing the current mv-ARF algorithm. One way to do this is to find more and more datasets and test the limits of the algorithm. This approach can be possibly applied to any new algorithm. Also, there is still effort to be made on the parameter selection scheme of mv-ARF. The proposed approach is a model based approach with very few parameters to optimize (only p, J and D). But the current application of mv-ARF only provides a parameter selection scheme for the lag order. The depth parameter and number of trees are intuitively set to a value that is the same for all applied datasets. Trial and error may be an option but our experiments show that testing three parameters is computationally expensive since some of the MTS datasets used in the experiments are very large. It would be a non-trivial work to extend the parameter selection scheme to set D and J in a computationally efficient manner.

The proposed methodology is a non-linear vector autoregressive approach, which makes it a unique approach within the literature since traditional vector autoregressive approaches are limited to linear interactions. Vector autoregression is an important direction of research because it aims to extend the univariate autoregressive approaches to a multivariate learning scheme. Since autoregressive modeling has proved to be a robust modeling framework for time-series, vector autoregression methods are expected to be successful in modeling of MTS. In this thesis, mv-ARF algorithm has proven to yield competitive and computationally efficient results for MTS classification problem. Possibly this algorithm may lead the way to more research on non-linear vector autoregressive approaches.

REFERENCES

1. Liu, J., L. Zhong, J. Wickramasuriya and V. Vasudevan, "uWave: Accelerometer-based personalized gesture recognition and its applications", *Pervasive and Mobile Computing*, Vol. 5, No. 6, pp. 657–675, 2009.

2. Lichman, M., *UCI Machine Learning Repository*, 2013, `http://archive.ics.uci.edu/ml`, accesed at November 2016.

3. Fu, T.-c., F.-l. Chung, V. Ng and R. Luk, "Pattern discovery from stock time series using self-organizing maps", *Workshop Notes of KDD2001 Workshop on Temporal Data Mining*, pp. 26–29, Citeseer, 2001.

4. Fu, T.-c., "A review on time series data mining", *Engineering Applications of Artificial Intelligence*, Vol. 24, No. 1, pp. 164–181, 2011.

5. Harrison, L., W. Penny and K. Friston, "Multivariate autoregressive modeling of fMRI time series", *NeuroImage*, 2003.

6. Anderson, C. W., E. A. Stolz and S. Shamsunder, "Multivariate autoregressive models for classification of spontaneous electroencephalographic signals during mental tasks", *IEEE Transactions on Biomedical Engineering*, Vol. 45, No. 3, pp. 277–286, 1998.

7. Hu, X. and V. Nenov, "Multivariate AR modeling of electromyography for the classification of upper arm movements", *Clinical Neurophysiology*, 2004.

8. Ge, D., N. Srinivasan and S. M. Krishnan, "Cardiac arrhythmia classification using autoregressive modelling", *BioMedical Engineering Online*, 2002.

9. Zhang, Z., P. Luo, C. C. Loy and X. Tang, "Facial Landmark Detection by Deep Multi-task Learning", *LNCS 8694*.

10. Chakraborty, B., "Feature selection and classification techniques for multivariate time series", *Innovative Computing, Information and Control, 2007. ICICIC'07. Second International Conference on*, pp. 42–42, IEEE, 2007.

11. Keogh, E. and S. Kasetty, "On the need for time series data mining benchmarks: a survey and empirical demonstration", *Data Mining and knowledge discovery*, Vol. 7, No. 4, pp. 349–371, 2003.

12. Chandra, B., M. Gupta and M. Gupta, "A multivariate time series clustering approach for crime trends prediction", *SMC 2008. IEEE International Conference on Systems, Management and Cybernetics*, pp. 892–896, IEEE, 2008.

13. Raman, H. and N. Sunilkumar, "Multivariate modelling of water resources time series using artificial neural networks", *Hydrological Sciences Journal*, Vol. 40, No. 2, pp. 145–163, 1995.

14. Du Preez, J. and S. F. Witt, "Univariate versus multivariate time series forecasting: an application to international tourism demand", *International Journal of Forecasting*, Vol. 19, No. 3, pp. 435–451, 2003.

15. Weng, X. and J. Shen, "Classification of multivariate time series using two-dimensional singular value decomposition", *Knowledge-Based Systems*, Vol. 21, No. 7, pp. 535–539, 2008.

16. Lin, J., E. Keogh, S. Lonardi and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms", *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pp. 2–11, ACM, 2003.

17. Baydogan, M. G. and G. Runger, "Learning a symbolic representation for multivariate time series classification", *Data Mining and Knowledge Discovery*, Vol. 29, No. 2, pp. 400–422, 2015.

18. Chan, K.-P. and A. W.-C. Fu, "Efficient time series matching by wavelets", *Proceedings of the 15th International Conference on Data Engineering*, pp. 126–133, IEEE, 1999.

19. Bashir, F., W. Qu, A. Khokhar and D. Schonfeld, "HMM-based motion recognition system using segmented PCA", *IEEE International Conference on Image Processing 2005*, Vol. 3, pp. III–1288, IEEE, 2005.

20. Sempena, S., N. U. Maulidevi and P. R. Aryan, "Human action recognition using dynamic time warping", *International Conference on Electrical Engineering and Informatics (ICEEI)*, pp. 1–5, IEEE, 2011.

21. Li, Z.-X., F.-M. Zhang and K.-W. Li, "DTW based pattern matching method for multivariate time series", *Pattern Recognition and Artificial Intelligence*, Vol. 24, pp. 425–430, 2011.

22. Berndt, D. J. and J. Clifford, "Using Dynamic Time Warping to Find Patterns in Time Series.", *KDD workshop*, Vol. 10, pp. 359–370, Seattle, WA, 1994.

23. Keogh, E. J. and M. J. Pazzani, "Scaling up dynamic time warping to massive datasets", *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 1–11, Springer, 1999.

24. Górecki, T. and M. Łuczak, "Multivariate time series classification with parametric derivative dynamic time warping", *Expert Systems with Applications*, Vol. 42, No. 5, pp. 2305–2312, 2015.

25. Yang, K. and C. Shahabi, "A PCA-based similarity measure for multivariate time series", *Proceedings of the 2nd ACM international workshop on Multimedia databases*, pp. 65–74, ACM, 2004.

26. Cuturi, M. and A. Doucet, "Autoregressive kernels for time series", *arXiv preprint arXiv:1101.0673*, 2011.

27. Baydogan, M. G. and G. Runger, "Time series representation and similarity based on local autopatterns", *Data Mining and Knowledge Discovery*, Vol. 30, No. 2, pp. 476–509, 2016.

28. Åström, K. J., "On the choice of sampling rates in parametric identification of time series", *Information Sciences*, Vol. 1, No. 3, pp. 273–278, 1969.

29. Keogh, E., S. Chu and M. Pazzani, "Ensemble-index: A new approach to indexing large databases", *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 117–125, ACM, 2001.

30. Chung, F.-L., T.-C. Fu, R. Luk and V. Ng, "Flexible time series pattern matching based on perceptually important points", *Workshop on Learning from Temporal and Spatial Data in International Joint Conference on Artificial Intelligence (IJCAI'01)*, 2001.

31. Pratt, K. B. and E. Fink, "Search for patterns in compressed time series", *International Journal of Image and Graphics*, Vol. 2, No. 01, pp. 89–106, 2002.

32. Yang, Z. and G. Zhao, "Application of symbolic techniques in detecting determinism in time series", *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Vol. 20, pp. 2670–2673, 1998.

33. Ou-Yang, K., W. Jia, P. Zhou and X. Meng, "A new approach to transforming time series into symbolic sequences", *21st Annual Conference and the 1999 Annual Fall Meetring of the Biomedical Engineering Society BMES/EMBS Conference, 1999. Proceedings of the First Joint*, Vol. 2, pp. 974–vol, IEEE, 1999.

34. André-Jönsson, H. and D. Z. Badal, "Using signature files for querying time-series data", *European Symposium on Principles of Data Mining and Knowledge Discovery*, pp. 211–220, Springer, 1997.

35. Qu, Z., *Robust control of nonlinear uncertain systems*, John Wiley & Sons, Inc., 1998.

36. Megalooikonomou, V., G. Li and Q. Wang, "A dimensionality reduction technique for efficient similarity analysis of time series databases", *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pp. 160–161, ACM, 2004.

37. Agrawal, R., C. Faloutsos and A. Swami, "Efficient similarity search in sequence databases", *International Conference on Foundations of Data Organization and Algorithms*, pp. 69–84, Springer, 1993.

38. Struzik, Z. R. and A. Siebes, "Wavelet transform in similarity paradigm", *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 295–309, Springer, 1998.

39. Chan, K.-P. and A. W.-C. Fu, "Efficient time series matching by wavelets", *Data Engineering, 1999. Proceedings., 15th International Conference on*, pp. 126–133, IEEE, 1999.

40. Wang, C. and X. S. Wang, "Supporting subseries nearest neighbor search via approximation", *Proceedings of the ninth international conference on Information and knowledge management*, pp. 314–321, ACM, 2000.

41. Struzik, Z. R. and A. Siebes, "The Haar wavelet transform in the time series similarity paradigm", *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 12–22, Springer, 1999.

42. Chan, F.-P., A.-C. Fu and C. Yu, "Haar wavelets for efficient similarity search of time-series: with and without time warping", *IEEE Transactions on knowledge and data engineering*, Vol. 15, No. 3, pp. 686–705, 2003.

43. Azzouzi, M. and I. T. Nabney, "Analysing time series structure with Hidden

Markov Models", *Neural Networks for Signal Processing VIII, 1998. Proceedings of the 1998 IEEE Signal Processing Society Workshop*, pp. 402–408, IEEE, 1998.

44. Vlachos, M., G. Kollios and D. Gunopulos, "Discovering similar multidimensional trajectories", *Data Engineering, 2002. Proceedings. 18th International Conference on*, pp. 673–684, IEEE, 2002.

45. Abfalg, J., H.-P. Kriegel, P. Kroger, P. Kunath, A. Pryakhin and M. Renz, "T-Time: threshold-based data mining on time series", *2008 IEEE 24th International Conference on Data Engineering*, pp. 1620–1623, IEEE, 2008.

46. Keogh, E., S. Lonardi, C. A. Ratanamahatana, L. Wei, S.-H. Lee and J. Handley, "Compression-based data mining of sequential data", *Data Mining and Knowledge Discovery*, Vol. 14, No. 1, pp. 99–129, 2007.

47. Faloutsos, C., M. Ranganathan and Y. Manolopoulos, *Fast subsequence matching in time-series databases*, Vol. 23, ACM, 1994.

48. Moon, Y.-S., K.-Y. Whang and W.-K. Loh, "Duality-based subsequence matching in time-series databases", *Data Engineering, 2001. Proceedings. 17th International Conference on*, pp. 263–272, IEEE, 2001.

49. Moon, Y.-S., K.-Y. Whang and W.-S. Han, "General match: a subsequence matching method in time-series databases based on generalized windows", *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pp. 382–393, ACM, 2002.

50. Keogh, E., J. Lin and A. Fu, "Hot sax: Efficiently finding the most unusual time series subsequence", *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 8–pp, IEEE, 2005.

51. Morinaka, Y., M. Yoshikawa, T. Amagasa and S. Uemura, "The L-index: An indexing structure for efficient subsequence matching in time sequence databases",

Proc. 5th PacificAisa Conf. on Knowledge Discovery and Data Mining, pp. 51–60, 2001.

52. Wang, H. and C.-S. Perng, "The S 2-Tree: An Index Structure for Subsequence Matching of Spatial Objects", *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 312–323, Springer, 2001.

53. Keogh, E., S. Lonardi and B.-c. Chiu, "Finding surprising patterns in a time series database in linear time and space", *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 550–556, ACM, 2002.

54. Kohonen, T. and P. Somervuo, "Self-organizing maps of symbol strings", *Neurocomputing*, Vol. 21, No. 1, pp. 19–30, 1998.

55. Möller Levet, C. S., F. Klawonn, C. Kwang-Hyun and O. Wolkenhauer, "Fuzzy clustering of short time series and unevenly distributed sampling points", *Proceedings of the Fifth International Symposium on Intelligent Data Analysis*, pp. 330–340, 2003.

56. Kalpakis, K., D. Gada and V. Puttagunta, "Distance measures for effective clustering of ARIMA time-series", *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pp. 273–280, IEEE, 2001.

57. Xiong, Y. and D.-Y. Yeung, "Time series clustering with ARMA mixtures", *Pattern Recognition*, Vol. 37, No. 8, pp. 1675–1689, 2004.

58. Panuccio, A., M. Bicego and V. Murino, "A Hidden Markov Model-based approach to sequential data clustering", *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 734–743, Springer, 2002.

59. Oates, T., L. Firoiu and P. R. Cohen, "Clustering time series with hidden markov

models and dynamic time warping", *Proceedings of the IJCAI-99 workshop on neural, symbolic and reinforcement learning methods for sequence learning*, pp. 17–21, Citeseer, 1999.

60. Han, J., W. Gong and Y. Yin, "Mining Segment-Wise Periodic Patterns in Time-Related Databases.", *KDD*, pp. 214–218, 1998.

61. Han, J., G. Dong and Y. Yin, "Efficient mining of partial periodic patterns in time series database", *Data Engineering, 1999. Proceedings., 15th International Conference on*, pp. 106–115, IEEE, 1999.

62. Lonardi, J. and P. Patel, "Finding motifs in time series", *Proceedings of the 2nd Workshop on Temporal Data Mining*, pp. 53–68, 2002.

63. Keogh, E. and J. Lin, "Clustering of time-series subsequences is meaningless: implications for previous and future research", *Knowledge and information systems*, Vol. 8, No. 2, pp. 154–177, 2005.

64. Keogh, E., J. Lin, A. W. Fu and H. VanHerle, "Finding unusual medical time-series subsequences: Algorithms and applications", *IEEE Transactions on Information Technology in Biomedicine*, Vol. 10, No. 3, pp. 429–439, 2006.

65. Wei, L., E. Keogh and X. Xi, "SAXually explicit images: finding unusual shapes", *Sixth International Conference on Data Mining (ICDM'06)*, pp. 711–720, IEEE, 2006.

66. Lonardi, S., J. Lin, E. Keogh *et al.*, "Efficient discovery of unusual patterns in time series", *New Generation Computing*, Vol. 25, No. 1, pp. 61–93, 2006.

67. Vlachos, M., S. Y. Philip and V. Castelli, "On Periodicity Detection and Structural Periodic Similarity.", *SDM*, Vol. 5, pp. 449–460, SIAM, 2005.

68. Wei, L., N. Kumar, V. N. Lolla, E. J. Keogh, S. Lonardi and C. A. Ratanama-

hatana, "Assumption-Free Anomaly Detection in Time Series.", *SSDBM*, Vol. 5, pp. 237–242, 2005.

69. Li, Y., X. S. Wang and S. Jajodia, "Discovering temporal patterns in multiple granularities", *Temporal, Spatial, and Spatio-Temporal Data Mining*, pp. 5–19, Springer, 2001.

70. Bettini, C., X. S. Wang, S. Jajodia and J.-L. Lin, "Discovering frequent event patterns with multiple granularities in time sequences", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 2, pp. 222–237, 1998.

71. Geurts, P., "Pattern extraction for time series classification", *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 115–127, Springer, 2001.

72. Zhang, H., T. B. Ho and M. S. Lin, "A non-parametric wavelet feature extractor for time series classification", *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 595–603, Springer, 2004.

73. Kadous, M. W. and C. Sammut, "Classification of multivariate time series and structured data using constructive induction", *Machine learning*, Vol. 58, No. 2, pp. 179–216, 2005.

74. Povinelli, R. J., M. T. Johnson, A. C. Lindgren and J. Ye, "Time series classification using Gaussian mixture models of reconstructed phase spaces", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 6, pp. 779–783, 2004.

75. Rodríguez, J. J. and C. J. Alonso, "Interval and dynamic time warping-based decision trees", *Proceedings of the 2004 ACM symposium on Applied computing*, pp. 548–552, ACM, 2004.

76. Wei, L. and E. Keogh, "Semi-supervised time series classification", *Proceedings*

of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 748–753, ACM, 2006.

77. Shokoohi-Yekta, M., Y. Chen, B. Campana, B. Hu, J. Zakaria and E. Keogh, "Discovery of meaningful rules in time series", *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1085–1094, ACM, 2015.

78. Agrawal, R., T. Imieliński and A. Swami, "Mining association rules between sets of items in large databases", *Acm sigmod record*, Vol. 22, pp. 207–216, ACM, 1993.

79. Das, G., K.-I. Lin, H. Mannila, G. Renganathan and P. Smyth, "Rule Discovery from Time Series.", *KDD*, Vol. 98, pp. 16–22, 1998.

80. Lu, H., J. Han and L. Feng, "Stock movement prediction and n-dimensional inter-transaction association rules", *Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, p. 12, 1998.

81. Hetland, M. L. and P. Sætrom, "Evolutionary rule mining in time series databases", *Machine Learning*, Vol. 58, No. 2-3, pp. 107–125, 2005.

82. Cotofrei, P. and K. Stoffel, "Classification rules+ time= temporal rules", *International Conference on Computational Science*, pp. 572–581, Springer, 2002.

83. Boyd, S., "TREND: a system for generating intelligent descriptions of time series data", *IEEE International Conference on Intelligent Processing Systems (ICIPS1998)*, Citeseer, 1998.

84. Guimaraes, G. and A. Ultsch, "A method for temporal knowledge conversion", *International Symposium on Intelligent Data Analysis*, pp. 369–380, Springer, 1999.

85. Sripada, S. G., E. Reiter, J. Hunter, J. Yu and I. P. Davy, "Modelling the task of summarising time series data using KA techniques", *Applications and Innovations in Intelligent Systems IX*, pp. 183–196, Springer, 2002.

86. Ahmad, S., P. C. De Oliveira and K. Ahmad, "Summarization of Multimodal Information.", *LREC*, Citeseer, 2004.

87. Lavrenko, V., M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen and J. Allan, "Mining of concurrent text and time series", *KDD-2000 Workshop on Text Mining*, pp. 37–44, 2000.

88. Ahmad, S., T. Taskaya-Temizel and K. Ahmad, "Summarizing Time Series: Learning Patterns in 'Volatile'Series", *International Conference on Intelligent Data Engineering and Automated Learning*, pp. 523–532, Springer, 2004.

89. Kacprzyk, J., A. Wilbik and S. Zadrożny, "Linguistic summarization of time series using a fuzzy quantifier driven aggregation", *Fuzzy Sets and Systems*, Vol. 159, No. 12, pp. 1485–1499, 2008.

90. Kate, R. J., "Using dynamic time warping distances as features for improved time series classification", *Data Mining and Knowledge Discovery*, Vol. 30, No. 2, pp. 283–312, 2016.

91. Gudmundsson, S., T. P. Runarsson and S. Sigurdsson, "Support vector machines and dynamic time warping for time series", *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 2772–2776, IEEE, 2008.

92. Zheng, Y., Q. Liu, E. Chen, Y. Ge and J. L. Zhao, "Exploiting multi-channels deep convolutional neural networks for multivariate time series classification", *Frontiers of Computer Science*, Vol. 10, No. 1, pp. 96–112, 2016.

93. Grabocka, J., M. Wistuba and L. Schmidt-Thieme, "Fast classification of uni-

variate and multivariate time series through shapelet discovery", *Knowledge and Information Systems*, pp. 1–26, 2015.

94. Grabocka, J., N. Schilling, M. Wistuba and L. Schmidt-Thieme, "Learning time-series shapelets", *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 392–401, ACM, 2014.

95. Bankó, Z. and J. Abonyi, "Correlation based dynamic time warping of multivariate time series", *Expert Systems with Applications*, Vol. 39, No. 17, pp. 12814–12823, 2012.

96. Bulla, J., F. Lagona, A. Maruotti and M. Picone, "A multivariate hidden markov model for the identification of sea regimes from incomplete skewed and circular time series", *Journal of Agricultural, Biological, and Environmental Statistics*, Vol. 17, No. 4, pp. 544–567, 2012.

97. Wolpert, D. H., "Stacked generalization", *Neural networks*, Vol. 5, No. 2, pp. 241–259, 1992.

98. Prieto, O. J., C. J. Alonso-González and J. J. Rodríguez, "Stacking for multivariate time series classification", *Pattern Analysis and Applications*, Vol. 18, No. 2, pp. 297–312, 2015.

99. Seeger, M., "Covariance kernels from Bayesian generative models", *Advances in neural information processing systems*, Vol. 2, pp. 905–912, 2002.

100. Ratanamahatana, C. A. and E. Keogh, "Making time-series classification more accurate using learned constraints", SIAM, 2004.

101. Sakoe, H. and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition", *IEEE transactions on acoustics, speech, and signal processing*, Vol. 26, No. 1, pp. 43–49, 1978.

102. Cheng, C., A. Sa-Ngasoongsong, O. Beyca, T. Le, H. Yang, Z. Kong and S. T. Bukkapatnam, "Time series forecasting for nonlinear and non-stationary processes: a review and comparative study", *IIE Transactions*, Vol. 47, No. 10, pp. 1053–1071, 2015.

103. Fan, J. and Q. Yao, "Nonlinear Time Series: Nonparametric and Parametric Methods Springer-Verlag", *New York*, 2003.

104. Engle, R. F., "Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation", *Econometrica: Journal of the Econometric Society*, pp. 987–1007, 1982.

105. Tong, H., *Non-linear time series: a dynamical system approach*, Oxford University Press, 1990.

106. Tiao, G. C. and R. S. Tsay, "Some advances in non-linear and adaptive modelling in time-series", *Journal of forecasting*, Vol. 13, No. 2, pp. 109–131, 1994.

107. Friedman, L., L. Olshen, C. J. Stone *et al.*, "Classification and regression trees", *Wadsworth International Group*, Vol. 93, No. 99, p. 101, 1984.

108. De'Ath, G., "Multivariate regression trees: a new technique for modeling species–environment relationships", *Ecology*, Vol. 83, No. 4, pp. 1105–1117, 2002.

109. Geurts, P., A. Irrthum and L. Wehenkel, "Supervised learning with decision tree-based methods in computational and systems biology", *Molecular Biosystems*, Vol. 5, No. 12, pp. 1593–1605, 2009.

110. Quinlan, J. R., "Induction of decision trees", *Machine learning*, Vol. 1, No. 1, pp. 81–106, 1986.

111. Breiman, L., "Random forests", *Machine learning*, Vol. 45, No. 1, pp. 5–32, 2001.

112. Kadous, M. W., *Temporal classification: Extending the classification paradigm to*

multivariate time series, Ph.D. Thesis, The University of New South Wales, 2002.

113. CMU, *CMU Graphics Lab Motion Capture Database*, 2012, `http://mocap.cs.cmu.edu/`, accesed at November 2016.

114. Keogh, E., Q. Zhu and B. Hu, *The ucr time series classification/clustering homepage*, 2011, `http://www.cs.ucr.edu/ eamonn/time_series_data/`, accesed at November 2016.

115. Olszewski, R., *RT Olszewski Personal Webpage*, 2012, `http://www.cs.cmu.edu/ bobski/.`, accesed at November 2016.

116. Sübakan, Y. C., B. Kurt, A. T. Cemgil and B. Sankur, "Probabilistic sequence clustering with spectral learning", *Digital Signal Processing*, Vol. 29, pp. 1–19, 2014.

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

118. Williams, B. H., M. Toussaint and A. J. Storkey, "A Primitive Based Generative Model to Infer Timing Information in Unpartitioned Handwriting Data.", *IJCAI*, pp. 1119–1124, 2007.

119. Kudo, M., J. Toyama and M. Shimbo, "Multidimensional curve classification using passing-through regions", *Pattern Recognition Letters*, Vol. 20, No. 11, pp. 1103–1111, 1999.

120. Dias, D. B., R. C. Madeo, T. Rocha, H. H. Bíscaro and S. M. Peres, "Hand movement recognition for brazilian sign language: a study using distance-based neural networks", *2009 International Joint Conference on Neural Networks*, pp. 697–704, IEEE, 2009.

121. Alimoglu, F. and E. Alpaydin, "Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition", *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96)*, Citeseer, 1996.

122. Van Rossum, G. and F. L. Drake Jr, *Python reference manual*, Centrum voor Wiskunde en Informatica Amsterdam, 1995.

123. Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830, 2011.

124. Tuncel, K., *mv-ARF source code*, 2017, `http://ktuncel.myfreesites.net`, accessed at March 2017.

125. MATLAB, *version 7.10.0 (R2010a)*, The MathWorks Inc., Natick, Massachusetts, 2010.

126. Cuturi, M., *Autoregressive Kernels Source Code*, 2016, `http://www.iip.ist.i.kyoto-u.ac.jp/member/cuturi/code.html`, accessed October 24, 2016.

127. R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2008, `http://www.R-project.org`, ISBN 3-900051-07-0.

128. Baydogan, M. G., *Learned Pattern Similarity Source Code*, 2016, `http://www.mustafabaydogan.com/files/viewcategory/`, accessed at February 2017.