

INTERPRETABLE EARLY CLASSIFICATION OF MULTIVARIATE TIME SERIES

A Dissertation
Submitted to
the Temple University Graduate Board

In Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy

By
Mohamed F. Ghalwash
January 2014

Examining Committee Members :

Dr. Zoran Obradovic, Advisory Chair, Computer and Information Science
Dr. Slobodan Vucetic, Computer and Information Science
Dr. Alexander Yates, Computer and Information Science
Dr. Carmen Sapienza, External Member, Pathology and Laboratory
Medicine

ABSTRACT

Recent advances in technology have led to an explosion in data collection over time rather than in a single snapshot. For example, microarray technology allows us to measure gene expression levels in different conditions over time. Such temporal data grants the opportunity for data miners to develop algorithms to address domain-related problems, e.g. a time series of several different classes can be created, by observing various patient attributes over time and the task is to classify unseen patient based on his temporal observations. In time-sensitive applications such as medical applications, some certain aspects have to be considered besides providing accurate classification.

The first aspect is providing early classification. Accurate and timely diagnosis is essential for allowing physicians to design appropriate therapeutic strategies at early stages of diseases, when therapies are usually the most effective and the least costly. We propose a probabilistic hybrid method that allows for early, accurate, and patient-specific classification of multivariate time series that, by training on a full time series, offer classification at a very early time point during the diagnosis phase, while staying competitive in terms of accuracy with other models that use full time series both in training and testing. The method has attained very promising results and outperformed the baseline models on a dataset of response to drug therapy in Multiple Sclerosis patients and on a sepsis therapy dataset.

Although attaining accurate classification is the primary goal of data mining task, in medical applications it is important to attain decisions that are not only accurate and obtained early, but can also be easily interpreted which is the second aspect of medical applications. Physicians tend to prefer interpretable methods

rather than black-box methods. For that purpose, we propose interpretable methods for early classification by extracting interpretable patterns from the raw time series to help physicians in providing early diagnosis and to gain insights into and be convinced about the classification results. The proposed methods have been shown to be more accurate and provided classifications earlier than three alternative state-of-the-art methods when evaluated on human viral infection datasets and a larger myocardial infarction dataset.

The third aspect has to be considered for medical applications is the need for predictions to be accompanied by a measure which allows physicians to judge about the uncertainty or belief in the prediction. Knowing the uncertainty associated with a given prediction is especially important in clinical diagnosis where data mining methods assist clinical experts in making decisions and optimizing therapy. We propose an effective method to provide uncertainty estimate for the proposed interpretable early classification methods. The method was evaluated on four challenging medical applications by characterizing decrease in uncertainty of prediction. We showed that our proposed method meets the requirements of uncertainty estimates (the proposed uncertainty measure takes values in the range $[0,1]$ and propagates over time).

To the best of our knowledge, this PhD thesis will have a great impact on the link between data mining community and medical domain experts and would give physicians sufficient confidence to put the proposed methods into real practice.

ACKNOWLEDGEMENTS

“In the Name of Allah Most Graceful and Merciful”

I humbly acknowledge the blessings of Almighty Allah who has enabled me to complete the PhD thesis. May Allah prays on Mohamed “peace be upon him” the prophet and the messenger of Allah.

All my profound gratitude goes to Dr. Zoran Obradovic not only for suggesting the problem of the research but also for his supervision and encouragement during my studies. I appreciate his constant contributions of advice, time and funding to finish my PhD. I believe that this dissertation could not have been finished without the help and support from him.

I would like to thank Dr. Slobodan Vucetic and Dr. Alexander Yates for serving on my dissertation committee and for providing me with many useful comments and valuable suggestions during my preliminary exams which helped me a lot in the presentation of all thesis-related publications. Also, thanks to Dr. Carmen Sapienza for providing me a time slot in his busy schedule to serve as an external examiner.

Special thanks go to Uros Midic who supported me at the early phase of my PhD and for his kindly advice to have solid experimental results in my publications. Also, special thanks to Vladan Radosavljevic who I enjoyed working with him of many of my dissertation-related papers. Also, special thanks go to Dusan Ramljak and Alexey Uversky for their support during some phases of my PhD and to Mladen Nikolic and Nemanja Djuric for being a mock reviewer for some related publications.

I am gratefully thankful to all my colleagues at Computer and Information

Science Department¹: Debasish Das, Joseph Jupin, Qiang Lou, George Mathew, Uros Midic, Mladen Nikolic, Athanasia Polychronopoulou, Vladan Radosavljevic, Dusan Ramljak, Kosta Ristovski, Shoumik Roychoudhury, Jelena Slivka, Alexey Uversky, and Ping Zhang for their support, valuable discussion in our weekly meetings, enjoyable time outside the PhD environment, and for accepting me to lead the “Just for fun” group which intended to have some valuable discussion.

It is also my pleasure to thank Dr. Soheir M.Khamis and Dr. Hazem Baheeg Ain Shams University, Egypt; for first preparing me to have solid background and encouraging me to travel to USA to obtain the doctoral degree.

I am also grateful to my family for their patience, understanding and encouragement. All my profound gratitude goes to the soul of late of my father for helping me since my early childhood to be more productive and self-dependent, and to my mother who supported me all the time by her supplications. And now, last but not least, special thanks to my wife and kids for being patience in the foreign country during my entire research.

Finally, This work was funded, in part, by DARPA grant [DARPAN66001-11-1-4183] negotiated by SSC Pacific grant and by the Egyptian Ministry of Higher Education.

¹Names are sorted alphabetically based on the last name

TABLE OF CONTENTS

	PAGE
ABSTRACT	iii
ACKNOWLEDGEMENTS	v
LIST OF FIGURES	vii
LIST OF TABLES	ix
1 INTRODUCTION	1
1.1 Motivation	3
1.2 Related Work	6
1.3 Datasets	8
1.3.1 Drug Response Dataset	8
1.3.2 Viral Challenge Datasets	10
1.3.3 Myocardial Infarction Dataset	12
1.3.4 Sepsis Dataset	13
1.4 Terminology	16
2 EARLY CLASSIFICATION	20
2.1 Early classification of univariate time series	20
2.2 Early classification of multivariate time series	22
2.2.1 Hybrid Model for Early Classification	25
2.2.2 Experiments	32
2.2.3 Conclusion	40
3 INTERPRETABLE EARLY CLASSIFICATION	42
3.1 Interpretable early classification of univariate time series	42
3.2 Interpretable early classification of multivariate time series	52
3.2.1 Multivariate Shapelet Detection	52
3.2.2 Extraction of Interpretable Multivariate Patterns for Early Discrimination	70
4 UNCERTAINTY ESTIMATE FOR SHAPELET-BASED EARLY CLASSIFICA- TION MODELS	98
4.1 Uncertainty Estimation	99
4.1.1 Confidence Estimate	100
4.1.2 Uncertainty for Early Classification	105
4.2 Experiments	106
4.2.1 Modified EDSC versus Original EDSC	106

4.2.2	Case Study	107
4.2.3	Uncertainty Propagation	109
4.2.4	Uncertainty versus evaluation measures	110
4.3	Related Work	112
4.4	Conclusion	115
5	APPLICATION - SEPSIS THERAPY OPTIMIZATION	116
5.1	Analysis of Onset and Duration of Blood Purification Therapy . . .	119
5.1.1	Therapy Onset	119
5.1.2	Therapy Duration	120
5.2	Treatment with IPED Therapy Onset	121
5.3	Hybrid Treatment Strategy	123
5.4	Conclusion	123
	BIBLIOGRAPHY	125

LIST OF FIGURES

FIGURE	PAGE
1.1 Early classification framework	4
1.2 Drug response data	9
1.3 H3N2 data	11
1.4 PTB data	12
1.5 Virtual sepsis patients	19
2.1 HMM architecture	24
2.2 ECM training process	27
2.3 Step 1 of the ECM algorithm	28
2.4 ECM test phase	32
2.5 Drug Response Patients	34
2.6 Sensitivity of the user parameter confidence on the relative accuracy	35
2.7 Sensitivity of the user parameter confidence on the accuracy	36
2.8 Sensitivity of the user parameter confidence on the earliness	36
2.9 Comparison between ECM and Lin’s model at different time points	38
3.1 Multivariate shapelet example	43
3.2 Detailed multivariate shapelet example	45
3.3 Best matching distance	46
3.4 Shapelet distance threshold	46
3.5 Illustration of a 3-dimensional shapelet	53
3.6 Shapelet matches a time series	55
3.7 Earliness of shapelet	62
3.8 Illustration of the effectiveness of the MSD method on a case from H3N2 dataset	66
3.9 IPED framework	73
3.10 Distance between shapelet and time series dataset	75
3.11 Illustration of the effectiveness of the IPED method on a case from H3N2 dataset	88
3.12 The sensitivity of the model complexity \mathcal{B} on medical datasets. . . .	90
3.13 Evaluation of IPED on 3 datasets	91
4.1 Modeling confidence	100
4.2 TwoLeadECG Case Study	108
4.3 Uncertainty propagation for ECG200 dataset	109

4.4	Average uncertainty over all patients at each time point	110
4.5	The effect of the uncertainty on coverage, earliness and accuracy measures	111
5.1	Schematic diagram of dialysis-like blood purification device	117
5.2	Blood purification therapy efficacy	120
5.3	Theoretical considerations of the sepsis stages and treatment effects	121
5.4	Percentage of rescued patients	122
5.5	Hybrid treatment strategy	124

LIST OF TABLES

TABLE		PAGE
1.1	Drug reponse dataset	10
2.1	ECM Parameter optimization	34
2.2	Comparison between ECM and best model in literature on drug response data	37
2.3	Comparison between ECM and threshold-based method on sepsis data	39
3.1	Evaluation of EDSC on drug response and viral infection datasets. .	51
3.2	Evaluation of the MSD method on the viral infection and drug response datasets using all genes	67
3.3	Evaluation of MSD, ECM and EDSC methods on all datasets . . .	94
3.4	Comparison of different threshold methods	95
3.5	Comparison of different utility methods	96
3.6	Run-time analysis of MSD on the viral infection and drug response datasets	96
3.7	Evaluation of different optimizations to extract the key shapelets . .	97
4.1	ECG datasets descirption	106
4.2	Evaluation of the modified EDSC	107
4.3	Categorization of time series classification methods. Four properties are used to categorize different methods: interpretability (Inter), earliness (Ear), uncertainty estimate (Uncer), and multivariate time series (Multi)	112

CHAPTER 1

INTRODUCTION

A data set with several attributes are usually collected at one particular time point which is called static data. The static data has been used by the scientist few decades ago to explore some properties in the data such as deriving patterns that summarize the relationship in the data or predicting the value of a particular attribute (*Tan et al.*, 2006). However, the static data are assumed to be independent and identically distributed which is a strong assumption in many scenarios. Recent advances in technology have led to growth in data collection. So, the data are collected over time which maintains auto-correlation between successive points. In the last decade, time series data has gained significant interest and has become a popular and challenging source for the scientists in many applications such as financial and medical applications. In financial applications, researchers monitor stock prices to analyze different behavior of the stock market. In medical applications, physicians measure patients' clinical variables frequently to predict patients' health.

In the last decade, time series have been extensively analyzed in various fields, such as statistics, signal processing, and control theory. The focus of the research in these fields is on gaining a better understanding of the data-generating mechanism, the prediction of future values, or the optimal control of a system. From a statistical viewpoint, time series analysis is comprised of methods for analyzing time series data in order to extract meaningful statistics from the data.

As a part of time series analysis, time series forecasting is aimed to use a model, e.g. AutoRegressive Moving Average (ARMA), to predict future values based on previously observed values (*Box et al.*, 2008). The ultimate objective of the signal processing community is the characterization of the time series in such a manner as to allow for transformation of the time series, with a method like Fast Fourier Transformation (FFT), to extract useful information from the time series (*Bracewell*, 1999). Researchers and practitioners in Control Theory strive to calculate solutions for proper corrective action from the controller (inputs) that result in system stability. A set of past inputs and outputs is observed, and new inputs are set in such a way as to try to achieve a desired output (*Goodwin et al.*, 1979).

In data mining community, the practitioners look at the time series for exploring different tasks. Namely, clustering, classification, and some other tasks (*Fu*, 2011; *Esling and Agon*, 2012). The clustering task is to group unlabeled data such that the similarities within each group is minimized and the dissimilarity between each group is maximized (*Liao*, 2005). In the classification context of medical applications, the time series data are collected from several groups of individuals such as healthy and diseased patients. The classification task is to exploit such temporal information to determine the class membership of unseen patients and determine whether the patient belongs to the healthy group.

Several techniques have been proposed for time series classification. The techniques can be categorized into two groups: model-based and instance-based learning strategies. In model-based learning strategy techniques, the objective is to learn a classification model from the training instances and use that model to classify unlabeled instances. Such techniques include decision tree, neural network, and support vector machines. In instance-based learning strategy, the objective is to classify the unlabeled instance based on the similarity between the unlabeled

instance and all training (labeled) instances. One of the most widely used technique in that group is the K-Nearest Neighbor (KNN) (*Dasarathy, 1990; Cover and Hart, 1967*). It has been shown that it is hard to beat KNN on time series classification task (*Keogh and Kasetty, 2002; Lee et al., 2012*).

1.1 Motivation

In the context of classification (either model-based or instance-based technique) of unknown time series (time series with an unknown label), models utilize the whole time series with the unknown label to predict it based on the information learned from training data. In many time-sensitive applications, such as health informatics, it is critical to classify the time series at the earliest possible time point and retain similar accuracy to the model that utilizes the full-length time series for classification. For example, A study of the clinical data of newborn infants who had diagnosed with sepsis and sepsis-like disease, had abnormal heartbeat time series patterns 24 hours before the clinical suspicion of sepsis (*Griffin and Moorman, 2001*). Therefore, early diagnosis techniques might reduce the severity of many life-threatening diseases, by allowing effective treatment to be administered before the diseases fully manifested which would greatly reduce the risk of death.

The framework of the early classification of time series is outlined in Figure 1.1. In an early classification context, the objective is to provide patient-specific classification of unknown time series as early as possible. Therefore, instead of utilizing the whole time series, the early classification model looks at a portion (current stream) of the unknown time series and determines whether it is able to predict the label of the whole time series without looking at the rest of the time series. If the early classification model is able to predict at the time point which is at the end of the current stream, the label is predicted. Otherwise,

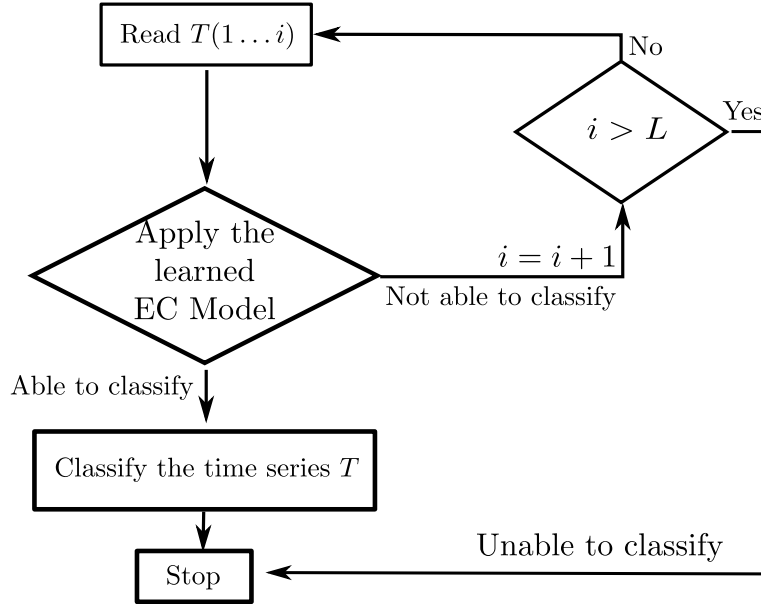
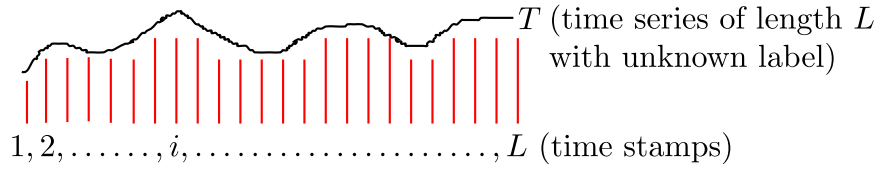


Figure 1.1: Early Classification Framework. Early classification (EC) model looks at a portion $T(1 \dots i)$ of length i of the unknown time series T . The EC model is then applied to the portion $T(1 \dots i)$ and the process is repeated (adding new time points) until the label is predicted, or the end of the time series is reached without obtaining a prediction.

the model requires more data for the unknown time series and looks at a larger segment, and does so until it is able to predict the label of the time series.

There are some work that have been proposed for early classification of time series. The details of these methods are reviewed in Section 1.2. However, the proposed early classification models either (1) are designed for univariate time series or (2) do not provide interpretable results. (1) In medical application, several variables have to be considered to have a clear explanation about the dynamic behavior of the disease. For example, sepsis, a medical condition characterized by uncontrolled inflammatory response due to infection, is one of the main causes of

deaths in the intensive care units, with over 750,000 cases annually in the United States alone (*Zuev et al.*, 2006). One of the main reasons for such a high number of death cases lies in limited understanding and knowledge about the complex inflammatory response mechanism, which has led to only a few effective sepsis therapies. That argues the need for early classification model for multivariate time series. (2) Interpretability is a fundamental desirable quality in predictive models (*Giraud-Carrier*, 1998; *Vellido et al.*, 2012). Time series models, regardless how sophisticated, can effectively be rendered powerless unless they can be interpreted by human experts (*Vellido et al.*, 2012). An interpretable model should be able to pinpoint exactly why a particular prediction was made, and provide the reason in a clear and natural way. For example, in medical domain, physicians tend to prefer interpretable methods rather than black-box methods. For this reason, the interpretability is a paramount quality that machine learning methods should aim to achieve if they are to be applied in practice.

One more aspect has to be considered for medical applications is the need for predictions to be accompanied by a measure which allows to judge about the uncertainty or belief in the prediction. Knowing the uncertainty associated with a given prediction is especially important in clinical diagnosis where data mining methods assist clinical experts in making decisions and optimizing therapy.

The objective of our work is to build multivariate time series predictive models that provides interpretable early classification of time series and retain accuracy similar to the models the utilizes the full time series for classification. Finally, we propose a method that provides uncertainty estimation for interpretable early classification method.

1.2 Related Work

Some traditional time series classification methods were (or could be) adapted for the purpose of early classification by applying the method at each time point. The methods are inflexible as the method trained on time series of length t , the prediction is always done at the t^{th} time point. A method that utilizes hidden Markov models (HMM) with less states than the time points is proposed (*Lin et al.*, 2008). The model is evaluated at each time point. They showed that the best results were obtained when using all time points. A linear dynamical system (LDS) and support vector machines (SVM) are used to model time series for gene expression (*Borgwardt et al.*, 2006). The method is applied at each time point and the best results were obtained when using most of the time points. These results provide evidence that there is a need for methods designed specifically for early classification.

The problem of early classification of time series is formulated recently (*Xing et al.*, 2009, 2011a). They introduced a novel concept of minimum prediction length and developed an early classification on time series (ECTS) method. ECTS makes early predictions and at the same time retains the accuracy comparable with that of a 1-nearest neighbor classifier using the entire time series. The disadvantage of ECTS is that it does not extract and summarize patterns from training data; thus, users may not be able to gain deep insights from the classification results. In many domains, it is really important not only to provide classification as early as possible but also to provide interpretable classification.

A method, called early distinctive shapelet classification (EDSC), is proposed for early classification (*Xing et al.*, 2011b). The EDSC method utilizes local shapelets (*Ye and Keogh*, 2009), which are segments of time series remaining in the same space of the input data and thus are highly interpretable. The

local shapelets are extracted to distinctly manifesting a target class locally and early so that they are effective for early classification. Experiments showed that the extracted local shapelets are highly interpretable and can achieve effective early classification. However, the method is proposed for early classification of univariate time series.

Several methods are proposed to provide real-time classification accompanied with uncertainty. An approach for early classification of signals using a generative classifier was developed recently (*Anderson et al.*, 2012). The model is based on the quadratic discriminant analysis (QDA) classifier and provides a reliability bound on the classifier's decision for every time point. Experiments showed that the approach is both early and reliable, and provides the user with a parameter to choose the trade-off between reliability and earliness. A patient risk is represented as a time series and the uncertainty is estimated as the distance between the evolving approximate daily risk of a patient and the hyperplane learned by the SVM (*Wiens et al.*, 2012). The model has been shown to be accurate on medical dataset. The disadvantage of these methods is that they are not interpretable ("black-box") for the domain experts which limits their applicability in the clinical domain.

In Chapter 2, we discuss a method on early classification on univariate time series (Section 2.1) and introduce our ECM method for early classification of multivariate time series (*Ghalwash et al.*, 2012, 2013b) (Section 2.2). In Chapter 3, we introduce interpretable methods for early classification on univariate (Section 3.1) and multivariate (Section 3.2) time series (*Ghalwash and Obradovic*, 2012; *Ghalwash et al.*, 2013b). Our method for uncertainty estimation (*Ghalwash et al.*, in reviewb) is introduced in Chapter 4. Finally, an application for interpretable early classification with uncertainty estimation (*Ghalwash et al.*, 2013a) on sepsis dataset is introduced in Chapter 5.

1.3 Datasets

Time series gene expression experiments have been used in a variety of biomedical applications (*Tchagang et al.*, 2009). These experiments consist of building expression profiles, which are essentially functions that model the changes in the expression levels of various genes. Such changes are collected by performing multiple microarray experiments over a period of time (*Spellman et al.*, 1998). Each experiment usually measures the expression level of multiple genes at a given time point. In this context, the gene expression time series classification problem is the process of determining the class of a previously unseen time series, based on the expression levels recorded for the time series in the training dataset. These classes can be defined in a variety of ways to help answer different kinds of questions. For example, as noted by *Borgwardt et al.* (2006), “*Will patient X respond well to a certain therapy or drug treatment? Is patient X going to develop sepsis in the next few hours? Is patient X recovering from a disease?*”

In our experiments throughout the thesis, we have used several gene expression datasets to evaluate the proposed models. We explain these gene expression datasets in the following sections.

1.3.1 Drug Response Dataset

A clinical dataset, which we will refer to as MS70, was generated to study the changes in cellular functions in multiple sclerosis patients in response to drug therapy with IFN β (*Baranzini et al.*, 2005). The dataset contains time series gene expression values for 52 patients. The patients were classified as good responders (33 patients) or bad responders (19 patients) to the drug. Blood samples were taken every 3 months in the first year and every 6 months in the second year. Some patients miss certain measurements, especially at the 7th time point. Thus,

the gene expression values were measured an average of 5-7 times for each subject.

Genes expression for good and bad subjects are presented in Figure 1.2.

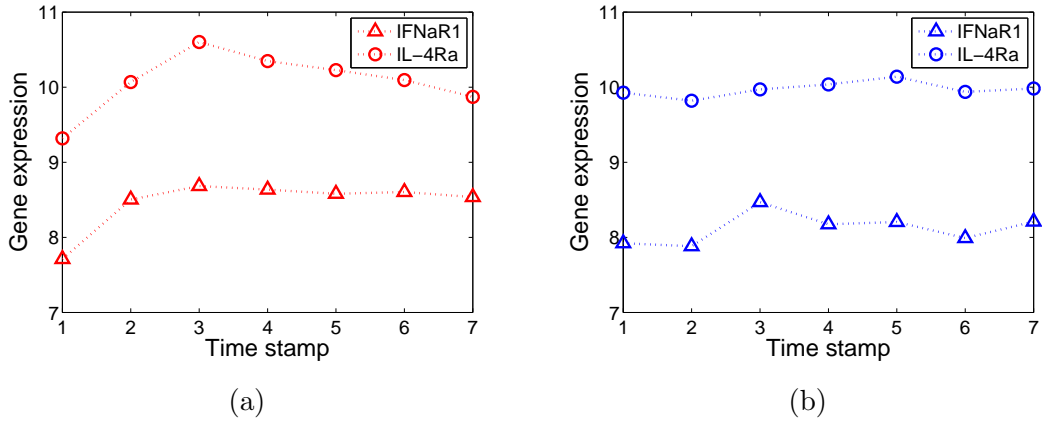


Figure 1.2: Two patients from the drug response dataset. (a) Two genes for a good responder subject were observed over 7 time points. (b) Two genes for an bad responder subject were observed over 7 time points. The dataset has 70 genes but we present only two genes for simplicity.

In order to adhere to the limitations of clinical settings (in which only a small, pre-specified number of genes is provided) and to be able to effectively compare our results with those attained in other studies, several datasets comprised of a fairly small number of genes were generated. The identification of triplets of genes for a Bayes classifier of time series gene expression data of multiple sclerosis patients' response to a drug was performed in (*Baranzini et al.*, 2005). Previous research identified 12 genes in terms of triplets. Hence, we generated four datasets: Baranzini3A and Baranzini3B, which consist of one triplet of the best two triplets of genes; Baranzini6, which has the top two triplets; and Baranzini12, which has all 12 genes identified by all triplets. A discriminative HMM has been developed and applied to the MS70 dataset to reveal the genes that are associated with the good or bad responders to the therapy (*Lin et al.*, 2008). A total of 9 genes were found that are associated with the therapy, 7 of which are identified using the last time stamp. Hence, we constructed two datasets, called Lin9 and Lin7,

Table 1.1: The list of the genes used in our experiments for the drug response datasets.

Dataset	Genes
Baranzini3A	Caspase 2, Caspase 10, FLIP
Baranzini3B	Caspase 2, Caspase 3, IRF4
Baranzini6	Caspase 7, Caspase 10, IRF2, IRF4, IRF6, IL-4Ra
Baranzini12	Caspase 2, Caspase 3, Caspase 7, Caspase 10 Flip, IRF2, IRF4, IRF6, IL-4Ra, IL12Rb1, STAT4, MAP3K1
Lin9	Caspase 2, Caspase 3, Caspase 10, IL-4Ra IL12Rb2, MAP3K1, IRF8, Jak2, RAIDD
Lin7	Caspase 2, Caspase 3, Caspase 10 IL-4Ra, MAP3K1, Jak2, RAIDD
Costa17	Caspase 2, Caspase 3, Caspase 10, Caspase 5 MAP3K1, STAT4, IRF2, IRF4, IRF5, IRF8, BAX, Tyk2 IL-4Ra, IL-2Rg, IFN-gRb, IFNaR2, Jak2

consisting of 9 and 7 genes, respectively. The entire MS70 dataset, containing all 70 genes, was also used for our experiments. A mixture of hidden Markov models has been developed to identify the genes that are associated with patient response to the treatment (*Costa et al.*, 2009). A total of 17 relevant genes were found, so we constructed a dataset called Costa17 that is comprised of the 17 genes. Table 1.1 contains the list of the genes used in our experiments for the drug response datasets.

1.3.2 Viral Challenge Datasets

Two datasets for blood gene expression from human viral studies with influenza A (H3N2) and live rhinovirus (HRV) to distinguish individuals with symptomatic acute respiratory infections from uninfected individuals are used in our experiments (*Zaas et al.*, 2009).

H3N2 dataset: A healthy volunteer intranasal challenge with H3N2 was performed in 17 subjects. Of those subjects, 9 became symptomatic and 8 remained asymptomatic. Blood samples were taken from each subject at 16 time points. Some subjects have missed certain measurements at time points 1, 5, 6, and/or 7. Hence, the gene expression values were measured on average 14-16 times for each subject. 30 genes were identified, in ranked order, as contributing to respiratory infection (*Zaas et al.*, 2009). We used 23 unique genes from that list that were found in the available dataset.

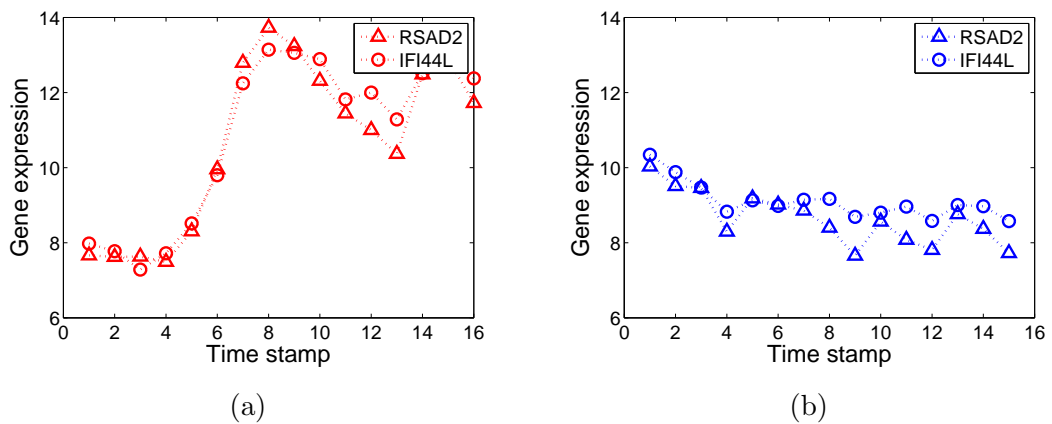


Figure 1.3: Two patients from the H3N2 dataset. (a) Two genes for a symptomatic subject were observed over 16 time points. (b) Two genes for an asymptomatic subject were observed over 15 time points. The dataset has 23 genes but we present only two for simplicity.

HRV dataset: A healthy volunteer intranasal challenge with HRV was performed in 20 subjects. Of those subjects, 10 became symptomatic and 10 remained asymptomatic. Blood samples were taken from each subject at 14 time points. Some subjects have missed certain measurements at time points 8-11. Hence, the gene expression values were measured on average 6-14 times for each subject. 30 genes were identified, in ranked order, as contributing to respiratory infection *Zaas et al.* (2009). We used 26 unique genes from that list that were found in the available dataset.

1.3.3 Myocardial Infarction Dataset

PTB (*Bousseljot et al.*, 1995) is an ECG database available at the Physionet website¹ *Goldberger et al.* (2000). In this application, we are interested in distinguishing between ECG signals of individuals with myocardial infarction (368 records) and those of healthy controls (80 records). The dataset consists of records of the conventional 12 leads ECGs (i, ii, iii, avr, avl, avf, v1, v2, v3, v4, v5, v6) together with the 3 Frank lead ECGs (vx, vy, vz).

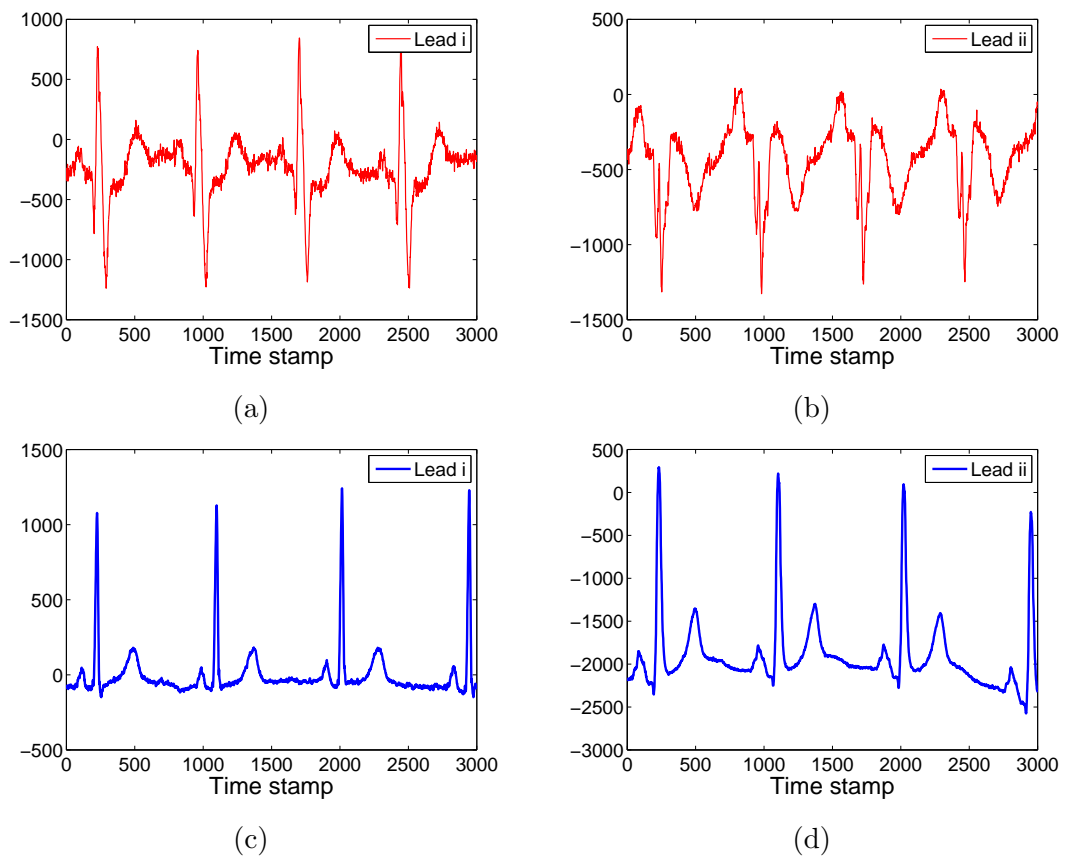


Figure 1.4: Two patients from the PTB dataset. (a) and (b) Lead i and ii for a unhealthy subject were measures over 3000 time points. (c) and (d) Lead i and ii for a healthy subject were measures over 3000 time points. The dataset has 15 leads but we present only two for simplicity.

¹<http://www.physionet.org/physiobank/database/ptbdb>

1.3.4 Sepsis Dataset

To significantly reduce the chance of a clinical failure and to save on the costs of clinical trials, biomedical researchers use computer simulations of body processes (often called virtual patients) to perform preliminary tests of hypotheses before they prove them in real patient studies. Virtual patients are generated using a carefully determined mathematical model to simulate the process of interest. A significant advantage of having a virtual patient model for experiments is the possibility of testing different approaches for finding adequate therapies on the same virtual patient and comparing the outcomes. In order to follow a real-life scenario, virtual patient models are accompanied with well-defined constraints in therapy that are in accordance with clinical practice (*Clermont et al.*, 2010; *Ristovski et al.*, 2012; *Song et al.*, 2012).

The mathematical model for inflammatory response to an infection is derived in (*Song et al.*, 2012). A mathematical model defines the dynamics of concentration of 19 variables (states) among which 8 are observable (Lsel - Lselectin; HMGB1 - high-mobility group protein B-1; CRT - creatinine; ALT - alanine aminotransferase; $TNF\alpha$ - tumor necrosis factor- α ; IL-1 - interleukin-1 β ; IL-6 - interleukin-6; IL-10 - interleukin-10) and 11 are hidden (CLP - cecal ligation and puncture; B - bacteria; Nt - peritoneal neutrophil; Nr - resting blood neutrophil; Np - primed blood neutrophil; Na - activated blood neutrophil; PI - systemic proinflammatory response; AI - systemic anti-inflammatory response; Ns - neutrophil sequestered in lung capillaries; Nl - lung neutrophil).

The model is also capable of modeling interactions between organs. This mathematical model is based on the system of Ordinary Differential Equations (ODE) whose details are presented in (*Song et al.*, 2012). Since it models measurable concentrations of cytokines, the 19-states mathematical model are capa-

ble of simulating blood purification system by hemoadsorption device (a column packed with beads that adsorb cytokines). It is assumed that a patient's blood is redirected through the hemoadsorption device where pro- and anti-inflammatory particles are removed.

Mathematical model simulates hemoadsorption as a first-order elimination of activated neutrophils (Na) as well as pro- and anti-inflammatory mediators (PI and AI) from the circulating blood. The parameters of the dynamics of adsorption were determined in (*Song et al.*, 2012). We assume that the device has two states - ON and OFF. ON state means that the device is attached to the patient and that it cleans blood with the rate specified in (*Song et al.*, 2012). OFF state means that device is detached from the patient. The ON/OFF states of blood purification device are controllable by clinicians.

Variability in the population of virtual patients is obtained by random initialization of three parameters in ODE and by random initialization of the states' initial conditions. In all of the simulations, t is an hourly step that starts from $t = 0$ when patient state and parameters are initialized. Then, patient state evolves according to ODE through the simulation time of 200 hours. According to *Song et al.* (2012) there are two possible outcomes at the end of simulation time. A patient is in survival group if (1) the number of bacteria (B) is less than B_{min} which was set to $1.0e5$, and (2) the value of systemic inflammation (PI) is less than 0.5. Otherwise, a patient is in non-survival group. Evolution of the patient to the final state can be modulated by applying blood purification device.

Real Data

We obtained real data from (*Song et al.*, 2012). Real data contain measurements over time of 8 observable states from mathematical model. As such, real data can be used for calibrating/testing the mathematical model. Experiments

to obtain real data were designed to evaluate long-term (one week) survival rate in a model of sepsis that resulted in a mortality rate similar to that observed clinically. The modified cecal ligation and puncture (CLP) protocol, 25% ligated length of cecum and 20-gauge needle, two-puncture, was used by (*Song et al.*, 2012) to induce sepsis in 23 rats. Plasma cytokines (tumor necrosis factor (TNF), interleukin(IL)-1b, IL-6 and IL-10), Lselectin (Lsel), high mobility group box1 (HMGB1), creatinine (CRT) and alanine aminotransferase (ALT) were measured from 0.8 ml blood samples at 18, 22, 48, 72, 120, 144, and 168h after CLP. No treatment was applied to any of 23 rats. Seven rats out of these 23 survived up to 7 days, being considered as the survivor population; the remaining 16 animals died and were considered as the non-survivor population.

Generation of Virtual Patient Population in Agreement with Real Data

Virtual patients are generated in accordance with real data as in (*Song et al.*, 2012). We generated each virtual patient according to the following 3-step protocol:

1. We randomly sample parameters of a mathematical model in consistence with valid ranges described in (*Song et al.*, 2012).
2. For chosen parameters we simulate the evolution of 19-states over time and determine the outcome (survival or non-survival).
3. We calculated the likelihood that evolution of 8 observable states follows evolution of real data (*Song et al.*, 2012). If the likelihood is high then the virtual patient has been "accepted" as valid. Otherwise, a generated patient has been rejected.

Following this protocol we generated 10000 sham (no treatment) virtual patients. A group of 5000 virtual patients belonged to the survival population, while another

group of 5000 virtual patients belonged to the non-survival population. Statistics of simulated data for eight measurable states together with observations from real data are presented in Figure 1.5.

1.4 Terminology

We introduce some terminologies we use throughout the thesis.

Definition 1.4.1 (Time series). *A time series $T = \{t_1, t_2, \dots, t_L\}$ of length L is defined as a sequence of real values sampled at L time stamps. The length of the time series is defined as $\text{len}(T) = L$. Each value of the time series is referred to as $T[i] \forall i = \{1, 2, \dots, L\}$.*

Definition 1.4.2 (Time series dataset). *Each time series is associated with a class label $c \in C$ where C is a finite set of class labels. A dataset D is a collection of M pairs $\{(T_i, c_i) : i = 1 \dots M\}$ where T_i is the time series number i and $c_i = \text{Class}(T_i)$ is its class.*

Definition 1.4.3 (Time series subsequence). *Given a time series $T = \{t_1, t_2, \dots, t_L\}$, a subsequence $s = T[i \dots i+l-1] = \{t_i, t_{i+1}, \dots, t_{i+l-1}\}$ is a sampling of contiguous positions of T of length $l < L$.*

Definition 1.4.4 (Euclidean distance between two time series). *Given two subsequences s and h where $\text{len}(s) = \text{len}(h) = l$, the Euclidean distance between s and h is defined as:*

$$\text{dist}(s, h) = \sqrt{\sum_{k=1}^l (s[k] - h[k])^2}$$

Definition 1.4.5 (Euclidean distance between a subsequence and a time series). *For a given time series T of length L and a subsequence s of length l , the distance between s and T is defined as the minimum distance between s and*

all subsequences of T of length l . Therefore, we slide a window of length l over the time series T to extract all subsequences $\{h_1, h_2, \dots, h_{L-l+1}\}$ of length l . The distance between s and T is computed as:

$$\text{dist}(s, T) = \min_{\forall i \in \{1, 2, \dots, L-l+1\}} \text{dist}(s, h_i) \quad (1.1)$$

Definition 1.4.6 (Multivariate time series). An N -dimensional (multivariate) time series \mathbf{T} of length L is defined as $\mathbf{T} = [T^1, T^2, \dots, T^N]$ where T^j is the j^{th} dimension of \mathbf{T} and $T^j[k]$ is the value of the j^{th} dimension of \mathbf{T} at time stamp k .

Hereafter, we use the terms ‘multidimensional’ and ‘multivariate’ interchangeably. We use the superscript j to refer to the dimension j , the subscript i to refer to the index of the time series, the bold symbol to refer to the multivariate case, and italic symbol for the univariate case.

Definition 1.4.7 (Multivariate time series subsequence). Given a multivariate time series $\mathbf{T} = [T^1, T^2, \dots, T^N]$, an N -dimensional subsequence \mathbf{s} of \mathbf{T} of length $l < L$ is defined as $\mathbf{s} = [s^1, s^2, \dots, s^N]$, where s^j is a subsequence of T^j of length l (Definition 1.4.3).

Definition 1.4.8 (Multivariate euclidean distance between two multivariate time series). Given two multivariate subsequences $\mathbf{s} = [s^1, s^2, \dots, s^N]$ and $\mathbf{h} = [h^1, h^2, \dots, h^N]$ where $\text{len}(\mathbf{s}) = \text{len}(\mathbf{h}) = l$, the Euclidean distance between \mathbf{s} and \mathbf{h} is defined as:

$$\text{dist}(\mathbf{s}, \mathbf{h}) = [\text{dist}(s^1, h^1), \text{dist}(s^2, h^2), \dots, \text{dist}(s^N, h^N)]$$

where $\text{dist}(s^j, h^j)$ is defined in Definition 1.4.4.

Simply, the distance between two multivariate time series is a vector of

distances where each component in the distance vector is the distance between the corresponding dimensions of the two multivariate time series.

Definition 1.4.9 (Multivariate euclidean distance between a subsequence and a time series). *The distance between an N -dimensional subsequence \mathbf{s} and N -dimensional time series \mathbf{T} is a vector of N Euclidean distances and is defined as:*

$$\text{dist}(\mathbf{s}, \mathbf{T}) = [\text{dist}(s^1, T^1), \text{dist}(s^2, T^2), \dots, \text{dist}(s^N, T^N)] \quad (1.2)$$

where $\text{dist}(s^j, T^j)$ is defined as in Equation 1.1.

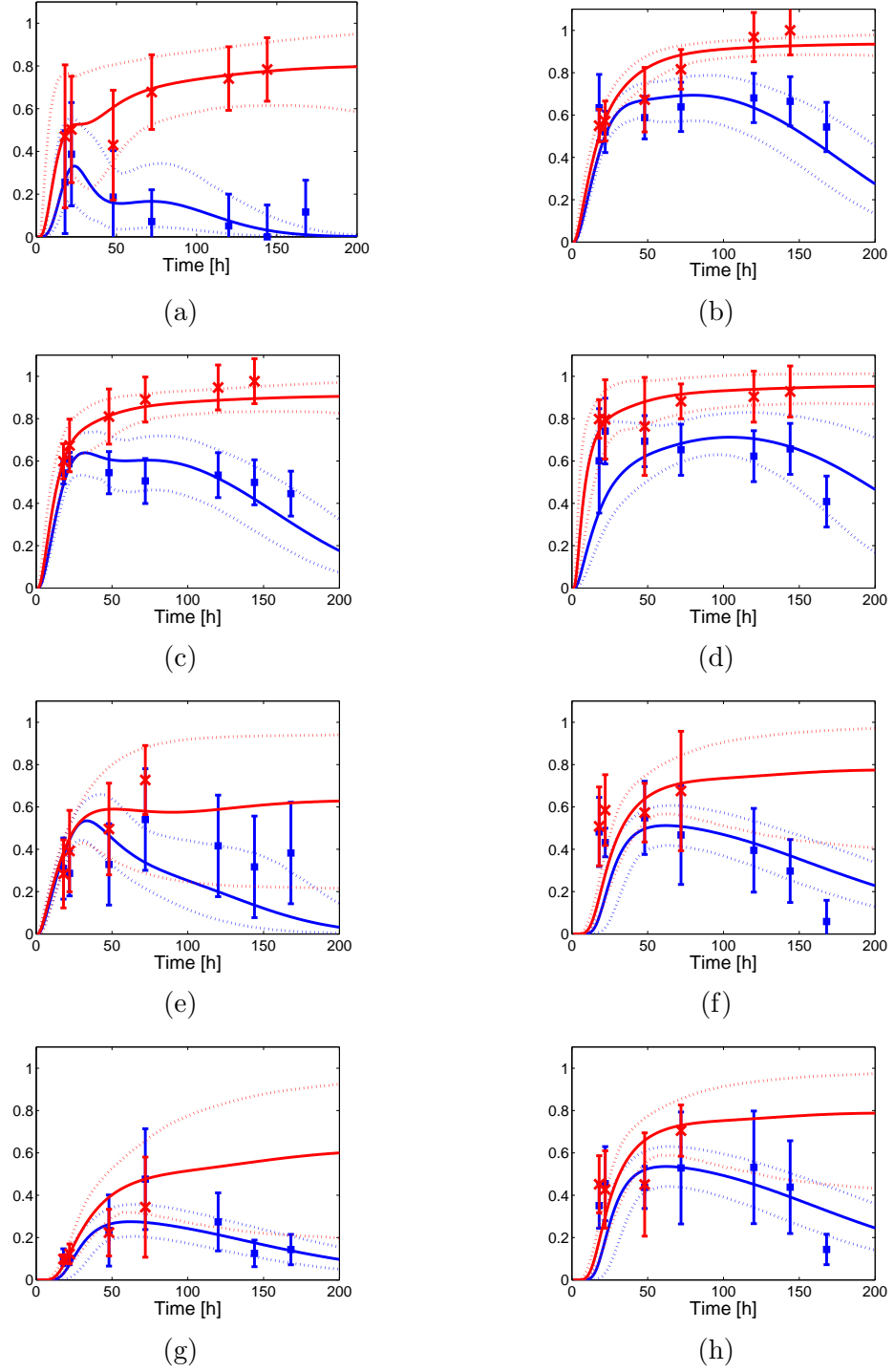


Figure 1.5: An agreement between simulated and real data. a) $TNF\alpha$, b) IL-1, c) IL-6, d) IL-10, e) Lsel, f) HMGB1, g) CRT, h) ALT. Solid lines - mean values of simulation outputs of 5000 virtual patients in survival (blue) and non-survival (red) groups. Dotted lines - region of 95% simulation uncertainty (95% of virtual patients are within the region). Error bars - real observations from animal study experiments.

CHAPTER 2

EARLY CLASSIFICATION

The idea of early classification of time series is to learn a model from the time series training data and then use that model to predict the label of time series with unknown label by observing only few points from the test time series (as shown in Figure 1.1) instead of using the full-length test time series. For that purpose, we start on Section 2.1 by briefly presenting the work of *Xing et al.* (2009) on early classification of univariate time series. Then, on Section 2.2 we present our work on early classification of multivariate time series (*Ghalwash et al.*, 2012, 2013b).

2.1 Early classification of univariate time series

Instance-based classifiers, such as the 1-nearest neighbor (1NN) classifier, classify the unlabeled instance based on the similarity between the unlabeled instance and all training (labeled) instances. The choice of distance metric is critical to the performance of 1NN classifiers. The Euclidean distance is shown to be superior on accuracy comparing to other similarity measurements (*Keogh and Kasetty*, 2002). *Xing et al.* (2009) extended the 1-NN classifier to make early classification of univariate time series.

Using 1-NN classifier, a time series T is classified by measuring the similarities between T and all time series in the training data. Then, T is classified as the class of the most similar time series to T . To achieve early classification,

Xing et al. (2009) proposed a method called Early Classification of Time Series (ECTS), which utilizes only the prefixes of the time series when computing the 1-NN.

If a time series T_1 is a 1-NN of time series T_2 using the full-length time series (all time points of the time series), then ECTS checks if T_1 is a 1-NN of T_2 using a prefix of the time series. If so, then the prefix of the time series could be used instead of the full-length time series to achieve the early classification. For that end, *Xing et al.* (2009) introduced a concept of reverse 1-NN.

The reverse of 1-NN of time series T is the set of all time series where T is considered to be the 1-NN. Then, ECTS checks if the prefix of the time series T could be used to obtain the same set of reverse 1-NN of T . If the same set is obtained, then the prefix (minimum prediction length MPL) of the time series T will be used in the classification process instead of the full-time series. Therefore, for each time series in the training, the time series is associated with a number MPL that determines the length of the prefix of the time series that could be used to obtain the same set of the reverse 1-NN using the full-length time series. The number MPL which differs from one time series to another is used to achieve early classification.

In the classification process, for a time series T with unknown label, the label is determined using the earliest MPL concept and the label of T is the label of the time series that is most similar to T based on the prefix of length MPL. If no such a time series is found, ECTS cannot make reliable prediction at the current time stamp, and have to wait for more values of T .

However, the learned $\text{MPL}(t)$ may not be stable and robust enough to make accurate classification. ECTS address this issue by clustering the time series and computing the MPL for each cluster instead of each time series. The reader is encouraged to read the original paper (*Xing et al.*, 2009) for more details.

The ECTS had been evaluated on seven univariate time series datasets from the UCR time series archive (*Keogh et al.*, 2011) and had been shown to be very competitive to the 1-NN classifier in term of accuracy. However, the ECTS provides the prediction as early as using only 44%-90% of the time series length.

The ECTS is only applicable to univariate time series. In medical application, several variables have to be considered to have a clear explanation about the dynamic behavior of the disease. For example, sepsis is one of the main causes of deaths in the intensive care units with over 750,000 cases annually in the United States alone (*Zuev et al.*, 2006). One of the main reasons for such a high number of death cases lies in limited understanding and knowledge about the complex inflammatory response mechanism, which has led to only a few effective sepsis therapies. That argues the need for early classification model for multivariate time series. In the next chapter, we introduce an early classification method for multivariate time series.

2.2 Early classification of multivariate time series

In this study, we propose using a hybrid model that combines a generative model (HMM) with a discriminative model (SVM) for early classification of a multivariate time series (*Ghalwash et al.*, 2012, 2013b). We first give a brief introduction to HMM and SVM and then introduce our hybrid model in the following sections.

HMM

Hidden Markov Models (HMMs) are powerful statistical models for modeling time-series data that can be characterized by an underlying process generating an observable time series (*Rabiner, 1989; Rabiner and Juang, 1986*).

Definition 2.2.1. *The HMM, referred to as Λ , is defined as tuple $\Lambda = (S, \Pi, A, B)$ where*

- *S is a finite set of states,*
- *$\Pi = \{\pi_i\}$ are the initial state probabilities,*
- *$A = \{a_{ij}\}$ are the state transition probabilities, where a_{ij} is the probability of transition from state i to state j .*
- *$B = \{b_i\}$ is the emission probabilities, where b_i is the probability of generating a real value at state i .*

Lin et al. (2008) were one of the forerunners of using HMMs for classification of gene expression time series. They showed that using 2 states allows for efficient alignment of patients with different response rate, as the 2 states represent 2 phases of the response (disease in our context). Therefore, in our context we assume that the cardinality of the set of states S is 2, see Figure 2.1. The emission probability is assumed to be drawn from a multivariate Gaussian distribution whose dimension equals the number of genes. In other words, $b_i \sim \mathcal{N}(\mu_i, \sigma_i)$ where μ_i and σ_i are the mean and standard deviation of the multivariate Gaussian distribution of the state i . Hence, the covariance matrix is assumed to be diagonal to avoid overfitting and to allow for fast computation of the model.

It is worth noting that, despite the novelty and efficiency of Lin's model, it was incapable of performing early classification of the time series. In their

paper, several experiments were done to perform classification at each time point using shorter time series. Their model was not patient-specific since the length of segments considered was very inflexible; that is, if the model is trained on time series of length 3, the prediction is always done at the 3rd time point. Our ECM model addresses this issue directly, and was, to the best of our knowledge, one of the first models that is capable of patient-specific early classification of multivariate gene expression time series.

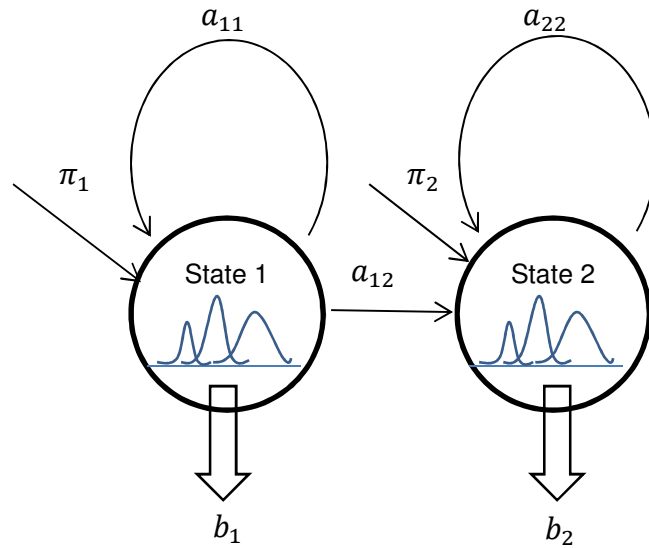


Figure 2.1: HMM Architecture. π_i is the initial state probability. a_{ij} is the transition probability from state i to state j . b_i is the emission state probability which is drawn from a multivariate Gaussian distribution of the state i .

SVM

Support Vector Machines (SVMs) belong to the family of kernel-based techniques that have proven to be extremely effective for machine learning purposes. They are a popular tool as they have a number of useful features, such as having theoretical guarantees of performance and not being affected by the curse of dimensionality. In instances where the data are not linearly separable, SVMs can

be used to transform the dataset by mapping it to a high-dimensional feature space via the kernel trick (*Boser et al.*, 1992). After this transformation has been performed, the SVM finds the maximal margin hyper-plane, expressed as a linear combination of support vectors of the training set, which is then used to classify the data:

$$f(\mathbf{x}) = \text{sgn}\left(\sum_i^M y_i \alpha_i \phi(\mathbf{x}_i \mathbf{x} + \delta)\right) \quad (2.1)$$

The set of parameters α_i is computed by solving a convex quadratic programming problem with linear constraints (*Burges*, 1998). Several kinds of kernel functions are often used with SVMs, including polynomial kernels, Radial Basis Function (hereinafter RBF) kernels, and sigmoid kernels. δ can be obtained using the Karush-Kuhn-Tucker condition (*Burges*, 1998).

2.2.1 Hybrid Model for Early Classification

The HMM Λ is used to learn the distribution of the patterns in a time series gene expression training dataset. The SVM takes the outputs of the HMM models as input: log likelihoods of segments in the time series gene expression training dataset. The role of SVM is to determine the combination of HMM models that could be trusted when classifying the time series based on the current stream.

The reasoning behind using a hybrid model is that if a generative model encounters a short segment that is repeated often in both classes (which can be a fairly common occurrence in gene expression data), it will likely not be able to distinguish between the two classes effectively. By applying a discriminative model on top of the generative model’s output, we are able to overcome this problem and improve overall performance. The details of how the hybrid model is trained and applied to the test data are provided next.

Data Preparation

The dataset is first divided into training and test data. The test data remains untouched until the very end, and all of the following procedures are applied to only the training data (see Figure 2.2(a)). When we mention test time series in the Training Phase, we are referring to the portion of the training data held out for testing for a particular fold of the internal cross-validation.

Training Phase

In general, several HMMs are trained on short time series segments and their log likelihoods on validation data are generated as features for the SVM. We opted for using an RBF kernel SVM.

Step 1: Train HMMs of specific length Given a dataset of time series of length L from one class, we extract all disjoint segments of length l , starting from position 0, and an HMM is trained on those segments. We refer to that HMM as Λ_1^l . We then shift the starting position by one time point, extract all disjoint segments of length l starting from position 1, and train another HMM on those segments. We call it Λ_2^l . We vary the length and the appropriate shifts in order to capture every possible pattern since these patterns can have different lengths and start at different positions.

Example 2.2.1. *An example of training different HMMs is illustrated in Figure 2.3 using only one time series for simplicity. A time series of length 24 is used for training HMMs. All time series segments of length 6 are extracted such that the first segment starts at position 1, the next segment starts at position 7, and so on. Therefore, we extract 4 disjoint time series segments such that the first one starts at position 1, and then train HMM Λ_1^6 on these segments. Then, we extract*

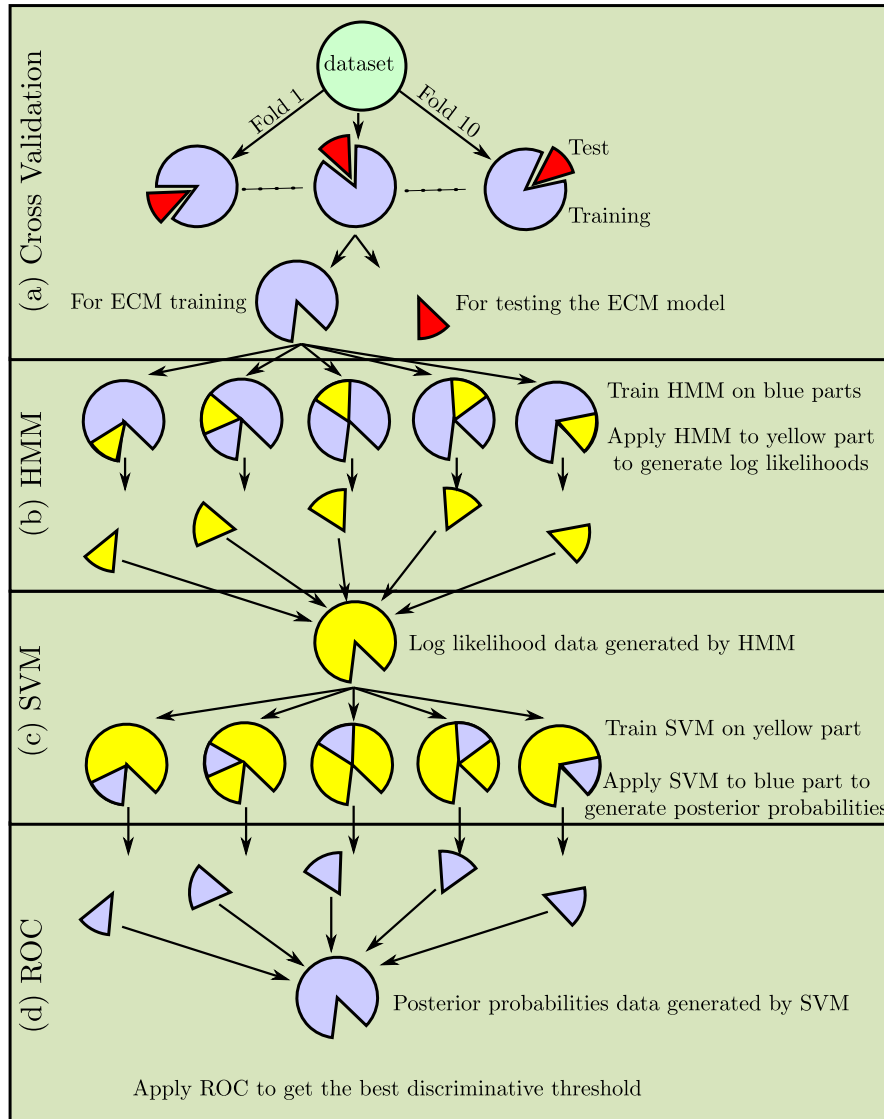


Figure 2.2: ECM training process. (a) The dataset is divided into 10 partitions for 10-cross validation process. For each fold, ECM is trained on 9 partitions and then tested on 1 partition. (b) For each training data, the data are divided into 5 partitions where HMM is trained on 4 partitions and applied on one partition to generate log likelihood data for SVM. (c) The log likelihood data are divided (same partitions as in HMM) into 5 partitions where SVM is trained on 4 partitions and applied on one partition to generate the posterior probabilities. (d) The ROC curve approach is applied to the posterior probabilities data to identify the best discriminative margin threshold for SVM.

all disjoint segments of length 6 such that the first segment starts at position 1. In this case, we extract only 3 disjoint segments. Then, we train another HMM

Λ_2^6 on these segments. We repeat that process 6 times. Clearly, there is no need to extract all disjoint segments where the first segment starts at position 7 because it will be a subset of the segments extracted from the position 1. The total number of the trained HMMs is 6.

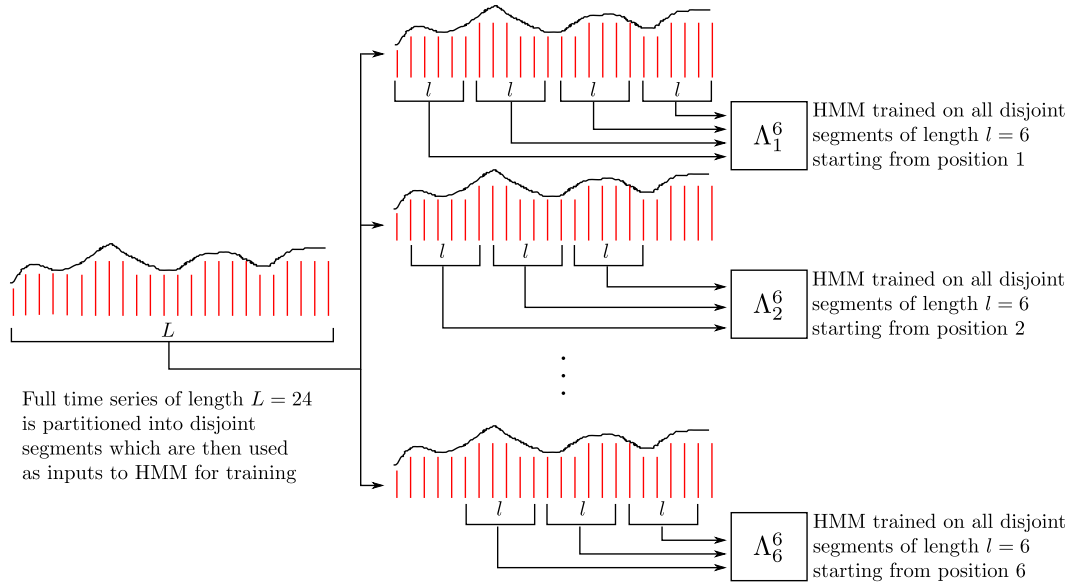


Figure 2.3: Step 1 of the algorithm is explained on time series of length $L = 24$, and for HMM trained on all segments of length $l = 6$.

Therefore, the number N_l of the HMM models, Λ_i^l where $i = \{1, \dots, N_l\}$, trained on all segments of length l at every starting position is

$$N_l = \begin{cases} l & \text{if } l \leq L/2, \\ L - l + 1 & \text{otherwise.} \end{cases}$$

Step 2: Train HMMs for different lengths We repeat the above procedure for different segment lengths l up to $l + k$. k is a user-defined parameter, set small enough to be able to capture all possible patterns, but $l + k$ should not exceed 50% of the time series to still be able to identify the pattern as early as possible.

Hence, the maximum number of models to be generated is

$$N_l + N_{l+1} + \dots + N_{l+k} = l + (l + 1) + \dots + (l + k) = (k + 1)(k/2 + l).$$

The same technique is applied to the time series of the other class, resulting in a maximum of N models where

$$N = 2(k + 1)(k/2 + l).$$

Step 3: Generate features for SVM We apply all trained HMMs on the test data as follows: first, we take the shortest time series segment O extracted from the training data. If the length of that segment is l , we then read l time points from the test time series (O_l). Next, we ask all N HMMs to generate their log likelihoods on the test time series we are currently examining. In general, an HMM is able to generate a log likelihood on the test time series if the model is trained on segments of length shorter than or equal to the length of the test time series. If the length of the current test time series is greater than l , then the Λ_i^l model uses only the last l time points of the current time series. The log likelihoods generated in this fashion, $\log P(O_l|\Lambda_j); j = \{1, \dots, N\}$, are then considered as features for an SVM (see Figure 2.2(b)).

The next part of this step is to read one more time point from the time series, so that the current test time series length is incremented by one. All models are asked to provide their log likelihoods on this time series to be another row for the SVM. This process is repeated until we have read all of the time points of the test time series. After this stage, we will have generated N_l instances with dimensionality N for the SVM, all of which have the class label of the test time series. Finally, we repeat this procedure for all test time series.

We repeat steps 1 - 3 five times to obtain 5-cross validations. Here we reinforce the fact that the HMMs are not used for prediction of the class of the time series, but rather as data for the SVM, which is made aware of the class labels independent of the HMMs.

After the cross validation is done, the log likelihoods generated by HMMs are ready to be used as features for the SVM. The parameters of the SVM are optimized using cross validation on the log likelihoods generated by the HMMs. Then another 5-cross validation is conducted for SVM to generate a score for each example. We use the same partitions of the 5-cross validations used for the HMM training phase to avoid any bias (see Figure 2.2(c)). When testing the SVM, the model computes a posterior probability estimate $P(y|O_l)$ of the class y for each example O_l . Namely, in a binary classification context, an SVM gives two probabilities: $P(y_1|O_l)$ and $P(y_2|O_l)$ (note that $P(y_2|O_l) = 1 - P(y_1|O_l)$). The scores for all examples are computed as the difference between the two probabilities $P(y_1|O_l) - P(y_2|O_l)$. Using the scores of all examples and their labels, we identify (using a standard ROC curve approach) the best threshold δ that maximizes the balanced accuracy (average between sensitivity and specificity)(see Figure 2.2(d)). Finally, both HMM and SVM are trained on training data which although being the same is essentially different - HMMs have time series as input, while SVM has log likelihoods, the output of HMMs, as input.

Testing Phase

We note here that the test data allocated in the very beginning of the procedure is never used for training either the SVM nor HMM models (see Figure 2.2(a)). The sketch for the test phase is illustrated in Figure 2.4. For a given time series with an unknown label, we read l time points (*the current stream*) from the test time series. We then ask all N HMM models to provide log likelihoods.

We use their outputs as inputs to the SVM, as described above. The score of the current time series is then computed using the probabilities generated by the SVM as $P(y_1|O_t) - P(y_2|O_t)$. If the score is greater than the threshold δ , then the current time series is most likely from class y_1 , otherwise, it is from class y_2 . We then ask if the probability estimate for the selected class is high enough to be confident about the prediction. A user-defined parameter *Conf* is used to measure this level of confidence. If the probability estimate is higher than the confidence level *Conf*, we stop at the current time point and predict the time series. Otherwise, we read one more time point from the current time series and repeat the procedure. If we reach the end of the time series and are not able to classify the time series, we mark that test time series as “not covered time series”.

Whole Process

The above training and testing phases are repeated 10 times using 10 different partitions (10 cross validations). We report the average of the following measures: coverage, earliness, relative accuracy, and accuracy. The measures are defined as follows:

Definition 2.2.2 (Coverage). *Coverage is the percentage of subjects who were classified (how many time series out of the test time series data were classified).*

Definition 2.2.3 (Earliness). *Earliness is the fraction of the time points used in the test for classification.*

Definition 2.2.4 (Relative Accuracy). *Relative Accuracy is computed as the average between sensitivity and specificity relative to the covered subjects. For example, if the method was able to classify 8 subjects out of 10 subjects and all 8 subjects were classified correctly then the coverage would be 80% and the relative accuracy would be 100%.*

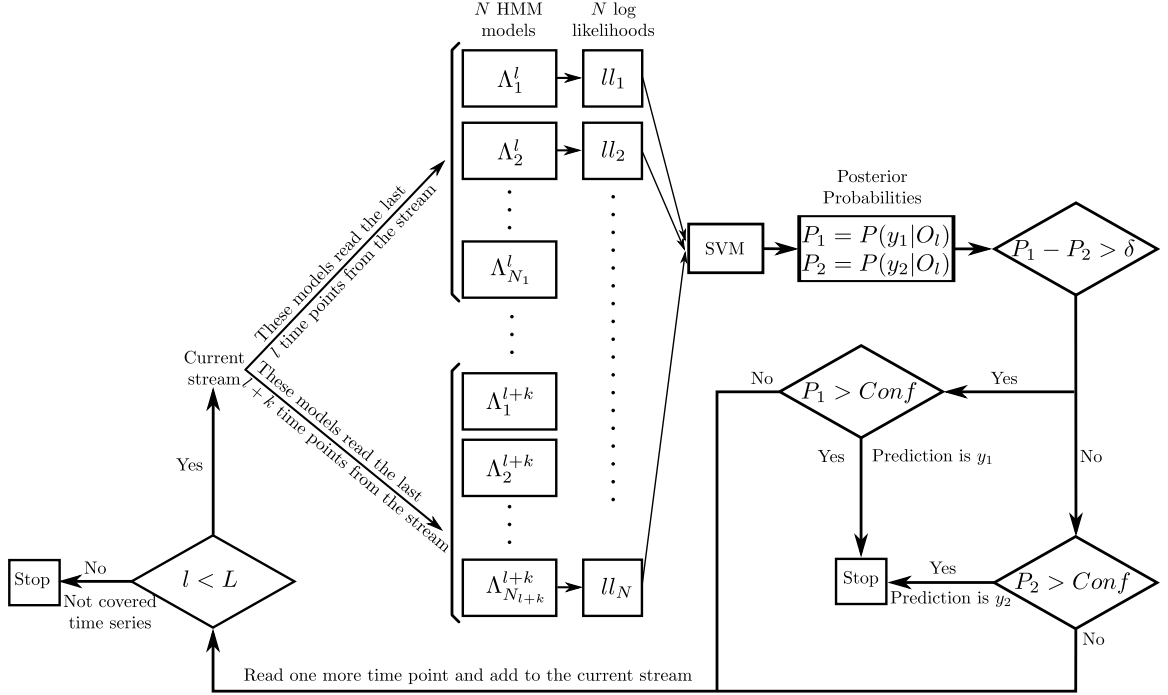


Figure 2.4: ECM Test phase. Each HMM model Λ_i^l reads the last l time points from the current stream and generates log likelihood u_i . The log likelihoods are then passed to SVM to generate the posterior probabilities of the time series being generated from each class. The margin between the two probabilities is compared to the threshold δ to determine the class membership. If the probability of the predicted class is greater than the confidence level $Conf$ then the process is stopped. Otherwise, we read one more time point and then the process is repeated until the prediction is obtained or we reach the end of the time series.

Definition 2.2.5 (Accuracy). *Since we report the coverage and the accuracy, it might not be easy to directly compare the methods. For example, if one method has better accuracy and less coverage and the second has better coverage and less accuracy, it is not clear which one is better. Therefore, in order to obtain fair comparison, the method classifies the non-covered examples using the majority of examples in the training data or randomly in case of a tie.*

2.2.2 Experiments

All of the procedures have been implemented in C++. We used libSVM, which provides probability estimates (Chang and Lin, 2011). All experiments are

conducted on a PC Intel Core i7 2.8 GHz with 8GB RAM.

Evaluation of ECM on the drug response dataset

In order to have a possibility of effectively comparing our method with other methods proposed in literature, we used the 8 datasets described in Section 1.3.1. Three of these datasets (Baranzini6, Baranzini12, and Lin9) were used to determine the best confidence value to be used by our model, since it was the user-defined parameter in our method. In this set of experiments, we trained our model with a distinct value for the confidence parameter (0.4, 0.5, and 0.6) on Baranzini6, Baranzini12, and Lin9, respectively. Note that although the possible values of the confidence parameter are in the $[0,1]$ range, we noticed that accuracy drops for values of confidence parameter smaller than 0.4 and coverage suddenly drops when we choose values greater than 0.6. Both trends could be easily explained: if we allow decisions when we are not confident, our model will decide early, but the accuracy will not be sufficient. On the other hand, if our requests for confidence are too high, we have generally higher relative accuracy scores, at the expense of coverage and earliness, and we might end up without decisions until the last time stamp. Therefore, we omitted values below 0.4 and above 0.6, for the set of experiments dealing with gene expression classification. Finally, we opted to use a value of 0.5 for the confidence parameter, as this value seemed to offer the best middle-ground of the various tradeoffs evident in Table 2.1.

To further explain the effectiveness of our approach we show a real case from the Baranzini3A dataset. Figure 2.5 illustrates a 3-dimensional gene expression time series (genes Caspase 2, Caspase 10, and FLIP) observed at 6 time steps. In the left panel, a patient who is a poor responder to the drug is correctly classified by our hybrid approach at the 3rd time point. In the right panel, a patient who is good responder to the drug is correctly classified at the 4th time point. Although

Table 2.1: Classification accuracy, coverage, and earliness in prediction at different levels of confidence ($Conf$) on various datasets. Earliness is the percentage of the time points used in the classification phase with respect to the full-length time series.

		Baranzini6	Baranzini12	Lin9
Relative Accuracy	$Conf = 0.4$	62	85	87
	$Conf = 0.5$	66	85	85
	$Conf = 0.6$	71	83	86
Coverage	$Conf = 0.4$	96	100	100
	$Conf = 0.5$	92	100	100
	$Conf = 0.6$	86	98	94
Earliness	$Conf = 0.4$	44	43	43
	$Conf = 0.5$	49	43	44
	$Conf = 0.6$	51	46	50

the patterns of both patients look similar to each other such that it is hard to distinguish between them by eye, our hybrid model was able to classify them correctly and early.

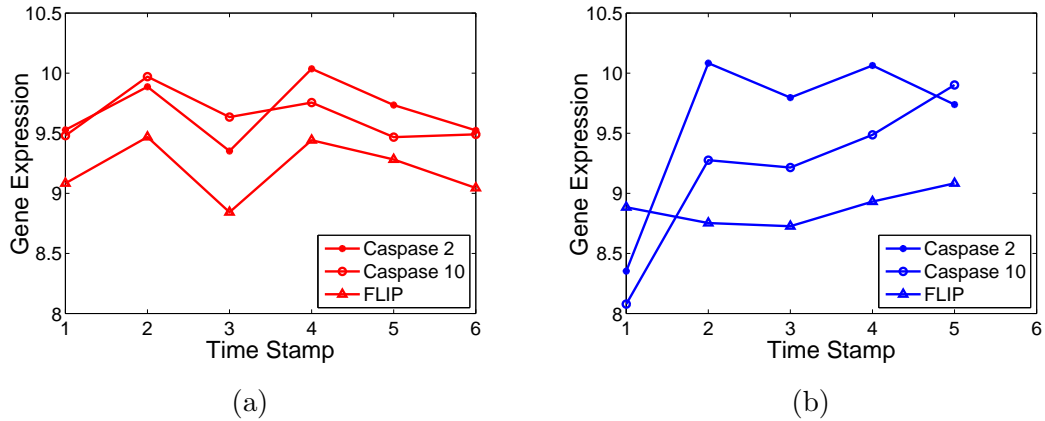


Figure 2.5: Time series for 3 gene expression for a patient who is a poor responder (left panel) and good responder (right panel) to the drug are represented. The subjects have been correctly classified at the 3rd and 4th time point, respectively.

Figure 2.6 compares different values for the user parameter confidence, which measures the confidence level the user expects from the model. Since the relative

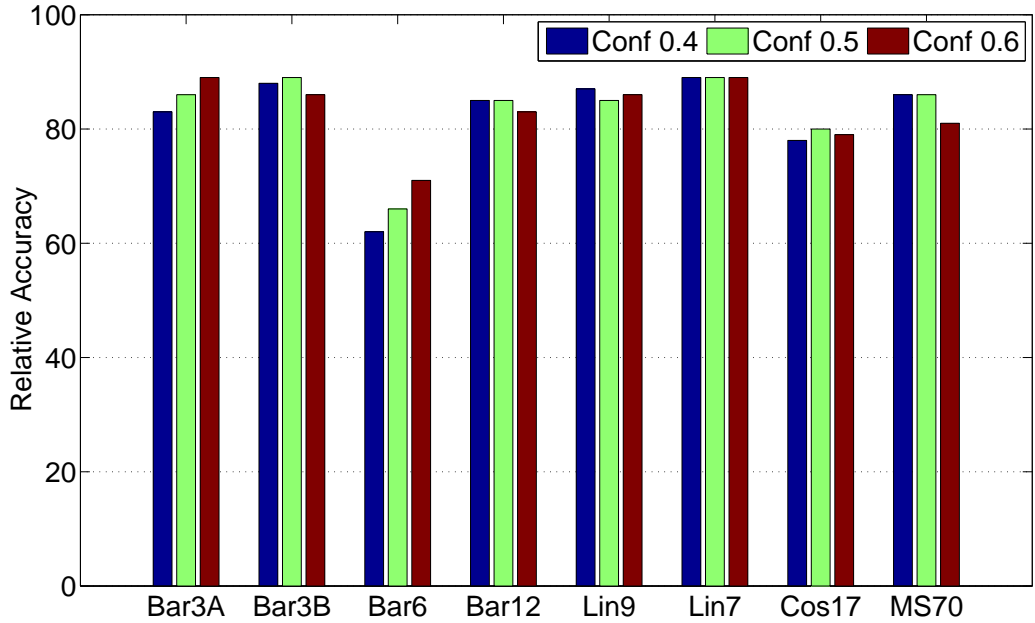


Figure 2.6: Measuring the sensitivity of the user parameter confidence on the relative accuracy for different datasets.

accuracy is barely affected by the value of confidence (in the specified 0.4 - 0.6 range) over the eight datasets, it is clear that the model is insensitive to the parameter within that range, and the results are fairly robust. The same comparison has been conducted to see the effect of different values of the confidence parameter on the accuracy and th earliness measures, as shown in Figures 2.7 and 2.8, respectively.

The outcome of applying our model to the remaining 5 datasets (out of the 8 that we initially created) is shown in Table 2.2. The accuracy of our model on each of the datasets is reported, along with the performance of the best models presented in literature when applied to the same datasets. Note that the accuracy of the “best” models is based on using the entire time series, while the accuracy of our model is based on using, on average, only 3/7 of the testing time series and obtaining patient-specific decisions. This important fact emphasizes the effectiveness of our method: even though the best models on average outperform our model in

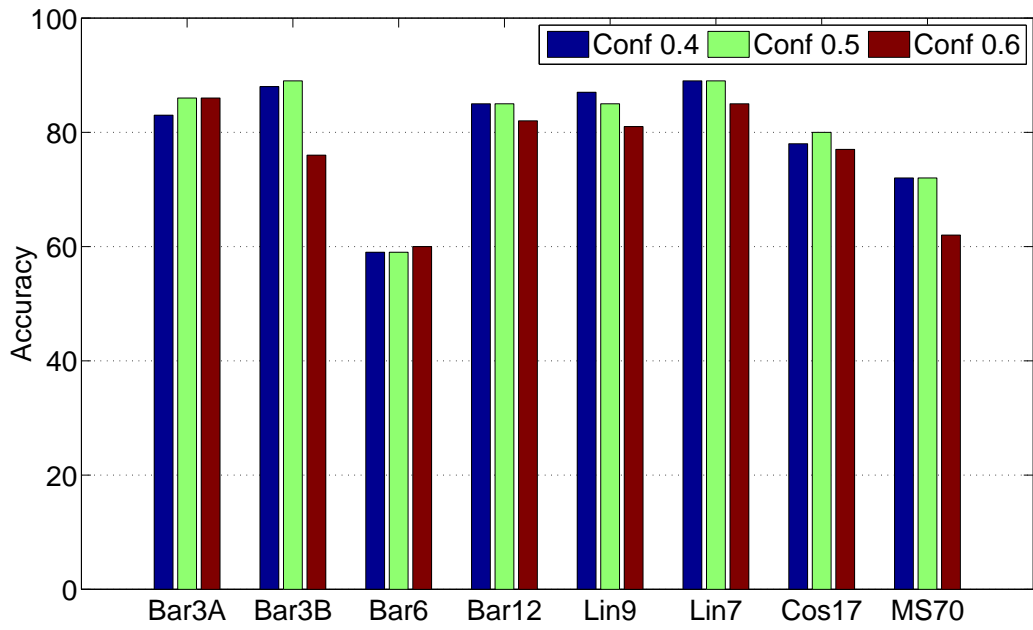


Figure 2.7: Measuring the sensitivity of the user parameter confidence on the accuracy for different datasets.

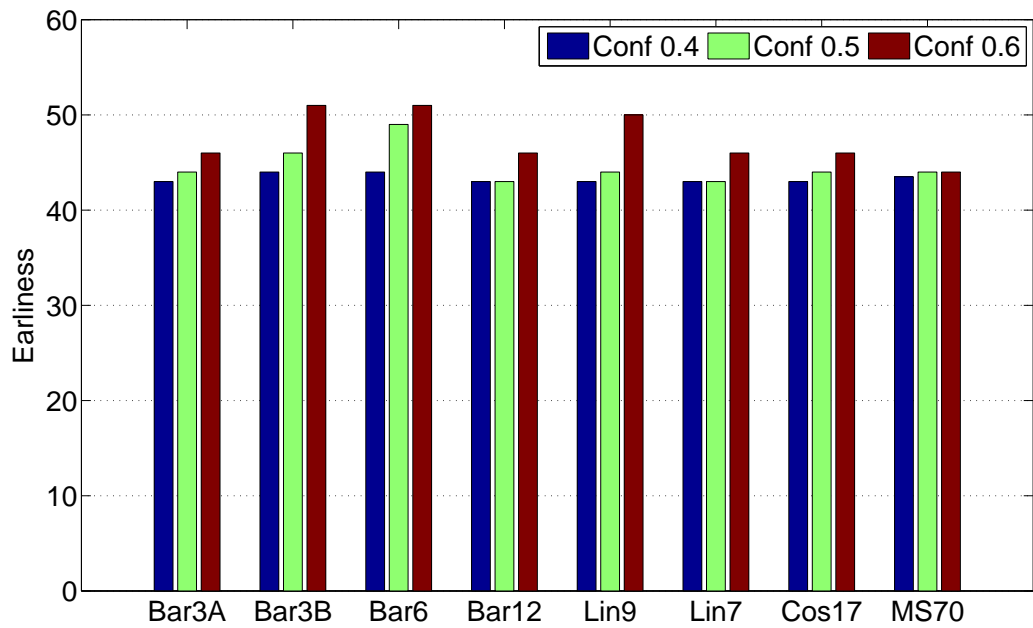


Figure 2.8: Measuring the sensitivity of the user parameter confidence on the earliness for different datasets.

terms of accuracy, we are able to provide competitive accuracy scores while using a significantly smaller portion of the full time series. In two of the datasets we were actually able to outperform the best models on accuracy, in spite of utilizing less than a full time-series when testing, reinforcing our model’s efficiency.

Table 2.2: Classification accuracy and earliness in prediction (in parentheses) for the best model in literature and our approach on various datasets. The mark \sim means on average. Wherever our approach achieved higher accuracy column Better? has a \checkmark mark, otherwise there is a \times mark.

Data set	Best	ECM	Better?
Baranzini3A	87.8 (7/7)	86.2(\sim 3/7)	\times
Baranzini3B	87.5(7/7)	88.5 (\sim 3/7)	\checkmark
Lin7	85.0(7/7)	88.7 (\sim 3/7)	\checkmark
Costa17	92.7 (7/7)	79.7(\sim 3/7)	\times
MS70	81.4 (7/7)	72.0(\sim 3/7)	\times

Figure 2.9, which provides the accuracy of the model developed by *Lin et al.* (2008) when using a shorter time series of fixed length in training and testing, further displays the strength of our model.

Evaluation of ECM on the sepsis therapy dataset

To test the performance of ECM on a sepsis-related dataset we opted for a sepsis model of the acute inflammatory response dataset, which consists of 4 variables varying over time (*Day et al.*, 2010). The authors adopted a subsystem approach to ensure that the interactions of the model variables are consistent with biological observations. Therefore, the model is able to capture a variety of clinically relevant scenarios associated with the inflammatory response to infection. The model displays three physiologically relevant equilibrium points, which correspond to biological states of health, aseptic death, and septic death. We chose to use this model since, to the best of our knowledge, there is currently no

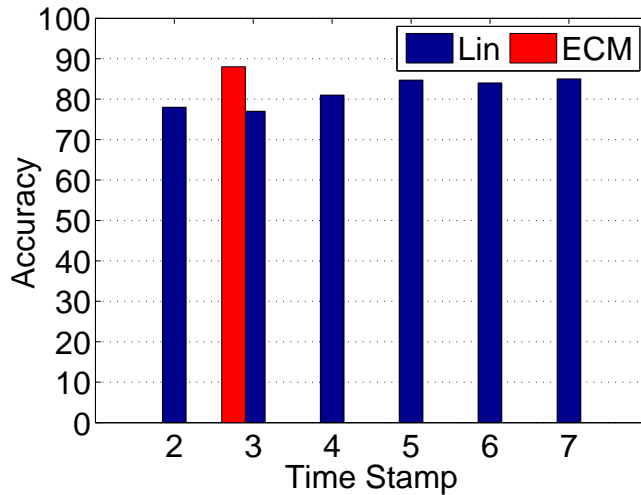


Figure 2.9: Comparison of the accuracy when using a shorter time series of fixed length in training and testing (Lin’s model) and ECM (different segment lengths and different shifts) on Lin7 dataset. ECM accuracy at the average earliness, consisting of 3 time points, is shown. Lin’s results on all time stamps were worse than ECM results.

publically available clinical dataset possessing the characteristics necessary for our experiments, and we used this model to generate a small number (50) of patients to further adhere to the restrictions of a real-world scenario.

We first obtained the relevant statistics of the threshold-based approach that is used for examining patients in hospitals. This approach relies on monitoring the value of one characteristic variable, and labeling the patient as non-healthy as soon as it passes the threshold of 0.05. Although this is a fairly simple approach, it is currently considered to be the state-of-the-art and is used in hospitals. However, the approach of this model is too aggressive, and even patients who do not need therapy are treated. To test this model in comparison to ECM, a population of 50 virtual patients is generated by the mathematical model (*Day et al., 2010*) on the simulation time of 24 hours (1 day) with hourly observations of all 4 given variables. Virtual patient state was determined from the values of the variables at the end of the total simulated time spent in the hospital. The population

Table 2.3: Classification results of threshold-based method and ECM on virtual septic patient data. The false negative (FN), false positive (FP), true negative (TN), true positive (TP) statistics are reported. Positive subjects are the septic patients and negative subjects are the healthy patients.

Model	FN	FP	TP	TN	Earliness
Threshold	0	7	21	22	2-3
ECM	0	3	21	26	2

consisted of 29 virtual patients classified as healthy and 21 classified as septic or aseptc (needed treatment). The results are shown in Table 2.3, and it is evident that

- the threshold appropriately treats unhealthy patients in the second or third hour,
- our approach appropriately treats the unhealthy patients in exactly the second hour,
- at the same time, healthy patients are not treated at all until the model confirms that they do not need treatment (which occurs by the fourth hour at the latest), at which point the remainder of the time series no longer needs to be considered and the patients are released from the emergency room.
- both the threshold and our method do not generate any false negatives, which is a vital statistic in the medical field (every false negative is essentially a patient that ends up dying due to misclassification),

The effectiveness of our method becomes clear when examining the false positive and earliness scores: using our method, 4 virtual patients were spared an incorrect diagnosis of sepsis. In reality that would effectively save real patients money on treatment and preserve their health (since unnecessary treatment can

cause damage and eventually death). Furthermore, the threshold-based method predicted all of the unhealthy patients at either the second or the third time step, whereas our method was able to do so without needing the third time step.

2.2.3 Conclusion

As evident by the experiments on the drug response dataset and the sepsis dataset, the proposed novel application of a hybrid HMM and SVM model was shown to be able to handle multivariate gene expression time series data, and is capable of performing patient-specific early classification on this kind of data with unprecedented results. On datasets gathered from real-world scenarios, the ECM model achieved relative accuracy scores that were on average slightly lower than the best results presented in literature when models were tested on the same datasets (although on two of these datasets, our model had a higher accuracy score). However, while the models from literature attained their best results after observing the full time series, our model was able to keep up by utilizing only 40% of the whole time series, thus displaying its early classification potential. Our approach was also able to significantly outperform the only other method designed specifically for early classification of multivariate time series. The performance of our model on a set of virtual patients was even more impressive: when compared to results obtained by applying the decision criteria used in the vast majority of hospitals, our model was able to correctly classify a larger proportion of patients, without misclassifying (and thus killing) any patients. Furthermore, it was able to do so as early or even earlier than the standard threshold-based approach without generating any false negatives, suggesting that there are no downsides in using our method over the threshold-based approach.

In closing, one of the assumptions of using HMMs is that there are enough

examples available for the model to accomplish a sufficient amount of learning. Because medical data are often expensive to obtain, this assumption cannot always be met when real data are involved (in some real-life scenarios, the data contain about 15 examples, as in the viral infection dataset (Section 1.3.2)), rendering our ECM approach useless. Another issue with the ECM model is the interpretability. Although the model was shown to be more accurate than other methods, the physicians tend to not use the model because the model does not provide why a particular prediction was made. In general, physicians tend to prefer interpretable models rather than black-box models. These issues limit the applicability of the ECM model in practice.

In the next chapter we introduce models that learn interpretable discriminative patterns from the dataset and use these patterns effectively for early classification.

CHAPTER 3

INTERPRETABLE EARLY CLASSIFICATION

In Chapter 2, we introduced the ECM model that was shown to be more effective than the best models in literature on several datasets. However, the ECM model is not interpretable which limits its applicability in practice. In this chapter, we introduce a model that extracts interpretable discriminative patterns and use them for early classification. In Section 3.1, we introduce the model that was proposed recently for early classification of univariate time series (*Xing et al.*, 2011b). Then, in Section 3.2, we extend the model to multivariate time series (*Ghalwash and Obradovic*, 2012; *Ghalwash et al.*, in review).

3.1 Interpretable early classification of univariate time series

In this section we briefly describe a method, called Early Distinctive Shapelet Classification (EDSC), that is proposed recently for interpretable early classification of univariate time series (*Xing et al.*, 2011b). The central idea of the method is to extract discriminative patterns (shapelets) for early classification. The shapelets, which are subsequences of time series remaining in the same space of the input data and thus are highly interpretable, have been used as features representing the characteristic of training time series (*Ye and Keogh*, 2009).

Example 3.1.1. *Suppose we have a dataset of 6 individuals. The time series for 3*

unhealthy subjects (red) and 3 healthy subjects (blue) are illustrated in Figure 3.1. The shapelet (pattern) is the time series segment that represents the characteristic of the class. The extracted shapelet from each class is drawn as solid lines. It is clear that each shapelet represents the property of the represented class and they have different shapes.

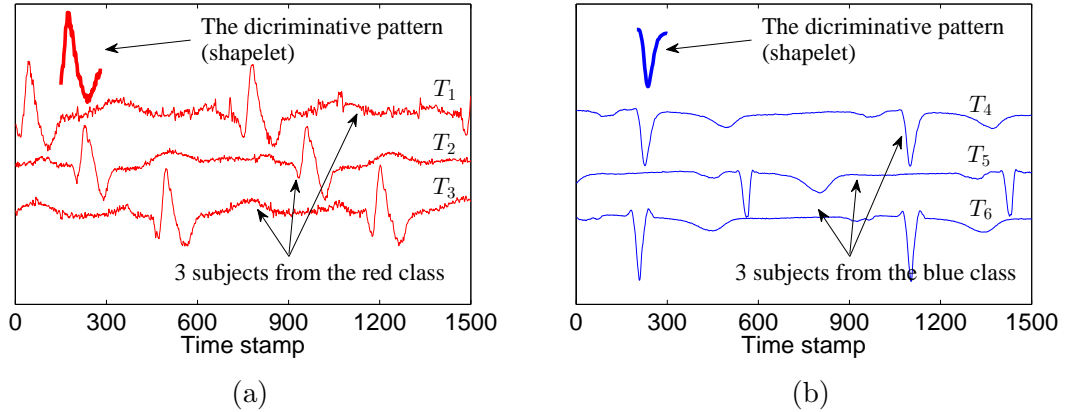


Figure 3.1: 3 subjects from the red class (T_1, T_2, T_3) and blue class (T_4, T_5, T_6) are shown in plots 3.1a and 3.1b, respectively. The discriminative shapelet from each class is represented as solid line. Clearly, the shapelet represents the characteristic of its class.

Ideally, a perfect shapelet of a class is the one representing all time series in the class but does not cover any time series in the other classes (as shown in Figure 3.1). These shapelets are used for interpretable time series classification. Based on the shapelet, the EDSC method was proposed, that utilizes local shapelets which are extracted to distinctly manifesting a target class locally and early so that they are effective for early classification (*Xing et al.*, 2011b). Therefore a time series with unknown label is classified as soon as the match is found between the time series and one of the extracted distinctive shapelets. Hence, there is no need to continue reading the rest of test time series, which complies with the framework of early classification mentioned in Figure 1.1.

Given a dataset D of univariate time series where each time series is associ-

ated with a class label, the task is to classify the time series as early as possible. The EDSC method addresses this problem in four steps:

1. extracts all time series subsequences of different lengths (shapelets). For each shapelet, a distance threshold is computed such that the shapelet discriminates well between classes,
2. ranks all the shapelets using some utility function that incorporates earliness and accuracy performance of the shapelet,
3. prunes the shapelets by selecting the top shapelets that cover all time series,
4. classifies unknown time series based on the closest matching shapelet.

We describe the phases of the EDSC method in the subsequent sections.

Step 1: Extracting all Shapelets

The core of the EDSC method is the shapelet, which is a time series subsequence that discriminates well between classes.

Definition 3.1.1 (Shapelet). *The shapelet is defined as $S = (s, l, \delta, c)$ where s is a time series subsequence of length l , c is the class label of the shapelet which is called the target class. The other classes (\bar{c}) are called the non-target classes. δ is a distance threshold that determines if the shapelet matches the time series in hand.*

Therefore, if the distance between the shapelet S and the current stream of the time series is less than δ then the shapelet matches or covers the time series.

Example 3.1.2. *The extracted shapelet from the blue class in Figure 3.1 is illustrated in Figure 3.2. The shapelet S is a time series segment s (solid line) extracted from the time series T_4 from position 190 to position 290. The length*

l of the shapelet is 101. The label or class of the shapelet is blue. The distance threshold δ determines whether the shapelet matches or covers the time series. So, if the shapelet matches time series with unknown label, the time series will be classified as the blue class.

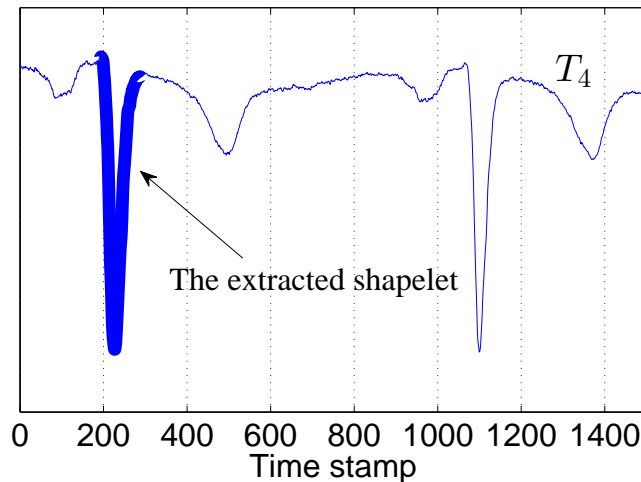


Figure 3.2: The shapelet $S = (s, l, \delta, c)$ is a time series segment s (solid line) extracted from the blue time series T_4 from position 190 to position 290. The length l of the shapelet is 101. The label or class of the shapelet is blue.

Suppose that we have extracted a shapelet $S = (s, l, \delta, c)$ where all parameters are defined except the distance threshold δ (we will explain later how to extract all shapelets). To compute the distance threshold, we need to compute the distances between the subsequence s and all time series in the dataset. To compute the distance between a subsequence s of length l and a time series T of length L (where $l < L$), we slide a window of length l over the time series T to extract all subsequences $\{h_1, h_2, \dots, h_{L-l+1}\}$ of length l . Then, the distance is computed as (see Figure 3.3):

$$\text{dist}(s, T) = \min_{\forall i \in \{1, 2, \dots, L-l+1\}} \text{dist}(s, h_i) \quad (3.1)$$

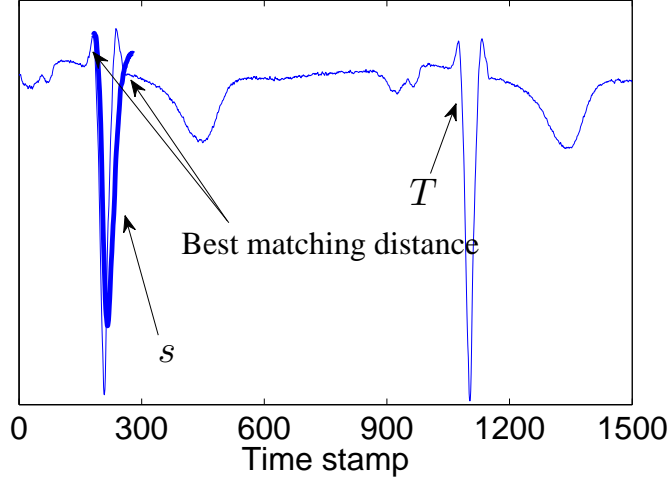


Figure 3.3: Best matching distance $dist(s, T)$ between the segment s (bold line) and time series T (thin line).

So, we compute the distance d_i between the shapelet S and each time series T_i in the dataset for each $i = \{1, 2, \dots, M\}$ where M is the number of the time series in the dataset. Each distance d_i is represented as a point on the order line as in Figure 3.4.

Example 3.1.3. The distances between the blue shapelet in Figure 3.2 and all 6 time series in Figure 3.1 are represented as points on the order line in Figure 3.4. Since the shapelet is extracted from the time series T_4 , the distance $d_4 = 0$. It is clear that the best distance threshold is between d_6 and d_3 .

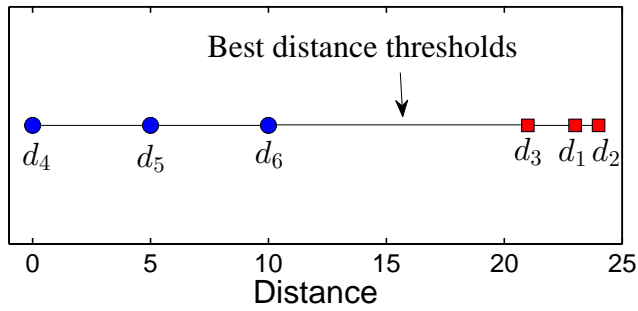


Figure 3.4: The distances d_i between the blue shapelet and all 6 time series T_i are represented as points on the order line. The best distance threshold δ is between d_6 and d_3 .

The distance threshold δ is computed such that it discriminates well between the classes. Several methods have been used for finding such best distance threshold. Namely, *Xing et al.* (2011b) proposed using Cheybshev’s inequality and kernel density estimation.

Xing et al. (2011b) showed the accuracy performance of the EDSC method on several datasets using the distance threshold learning methods they proposed. It has been shown that both learning methods are equally good and achieved similar accuracy performance. However, the Cheybshev’s inequality learning method is much faster than the kernel density estimation learning method. Therefore, we present the Cheybshev’s inequality as the main learning method for the EDSC method.

Cheybshev’s inequality The distance threshold δ is computed such that the distance between the shapelet and any non-target time series (time series of the other classes than c) is highly unlikely less than the threshold. In other words, the distance threshold is computed such that the probability of a non-target time series that matches the shapelet S is less than $1/k^2$ where k is a user defined parameter. Let X be a random variable with finite expected value μ and non-zero variance σ^2 . Then, for any real number $k > 0$, the Chebyshev’s inequality ensures that (*Allen* (1990))

$$Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

The EDSC method computes the distance d_i between the shapelet and each non-target time series and treat these distances as the random variable X . Therefore, the shapelet’s distance threshold δ is learned as:

$$\delta = \max(\mu - k * \sigma, 0) \tag{3.2}$$

Now, all parameters of the shapelet $S = (s, l, \delta, c)$ are defined except that how the shapelet or the time series segment is extracted from the data. Simply, the EDSC method iterates over all time series in D and extracts all subsequences of length l , where l is the length of the potential shapelets. The EDSC method varies l between $minL$ and $maxL$ which are user parameters. Then, for each time series segment of different length, the distance threshold for that particular shapelet is computed using Cheybshev’s inequality.

Step 2: Ranking the Shapelets

The number of the extracted shapelets from the dataset is large. Therefore, to find discriminative shapelets, the EDSC method assigns a score to each shapelet to rank them. The score of the shapelet incorporates both the earliness and the accuracy performance of the shapelet.

The earliness defines how early, on average, the shapelet matches the target time series (*the shapelet **matches** the time series T if $dist(s, T) \leq \delta$*). Technically, the earliness between the shapelet $S = (s, l, \delta, c)$ and the time series T is defined as

$$EML(S, T) = \min_{\forall i \in \{1, 2, \dots, L-l+1\}} dist(s, h_i) \leq \delta$$

where $len(T) = L$ and h_i are all subsequences of the time series T of length l .

Then, using the earliness, the weighted recall of the shapelet is computed as

$$WRecall(S) = \frac{1}{\|T_{\bar{c}}\|} \sum_{T \in D} \frac{1}{\sqrt[\alpha]{EML(S, T)}}$$

where α is a user defined parameter that determines the importance of the earliness, and $\|T_{\bar{c}}\|$ is the number of non-target time series. The utility score of the

shapelet is defined as

$$Utility(S) = \frac{2 \times Precision(S) \times WRecall(s)}{Precision(S) + WRecall(S)} \quad (3.3)$$

where $Precision$ is the fraction of the matched time series that are relevant (target time series) and computed as

$$Precision(S) = \frac{\|\{d_i \leq \delta \wedge Class(T_i) = c\}\|}{\|\{d_i \leq \delta\}\|}$$

where $d_i = dist(s, T_i)$ and $Class(T_i)$ is the class or label of the time series T_i .

The shapelets are sorted descending based on the utility score defined in Equation 3.3 so that the first shapelet in the list is the one that has the highest accuracy performance considering how early the shapelet appears in the training data.

Step 3: Pruning the Shapelets

The EDSC method sorts the shapelets descending based on their utility scores. Then, it starts with the highest ranked shapelet and removes all time series from the dataset that are matched by the shapelet. The shapelet $S = (s, l, \delta, c)$ **covers** the time series T if the shapelet matches the time series ($dist(s, T) \leq \delta$) and have the same class as the shapelet. Then, EDSC stores the shapelets and the next highest shapelet is considered. If the shapelet covers any of the remaining time series, the shapelet will be added to the extracted list and all covered time series will be removed. The method iteratively does so until all time series in the dataset are covered. In this manner, the EDSC method ends up with a small list of shapelets that is used for the classification purposes.

Step 4: Classification Phase

The classification process of the EDSC method follows the framework of the early classification described in Figure 1.1. The EDSC method initially reads a portion of length $minL$ from the test time series (where $minL$ is the length of the shortest extracted shapelet). The highest-ranked shapelet is considered. If the shapelet matches the time series then the time series is classified as the class of the shapelet and the prediction is made. Otherwise, the next shapelet from the ranked list is considered and the process of checking each shapelet is repeated. If none of the shapelets match the current portion of the test time series the method reads one more time stamp and continues classifying the time series (Figure 1.1). If the method reaches the end of the time series and none of the shapelets matches it, EDSC marks the time series as a not-classified example. The training phase of the EDSC method is summarized in Algorithm 1.

Algorithm 1: EDSC

Input: A training dataset D of M time series; $minL$, $maxL$ are the minimum and the maximum length of the shapelets

Output: *ShapeletList*: A list of all shapelets

- 1 *ShapeletList* = *empty* {sorted list of all shapelet, the shapelets are sorted descending based on the utility scores}
 - 2 **for** each time series $T \in D$ **do** { T is of length L }
 - 3 **for** $l \leftarrow minL$ **to** $maxL$ **do** {for each shapelet length}
 - 4 **for** $k \leftarrow 1$ **to** $L - l + 1$ **do** {for each starting position}
 - 5 $s := T[k \dots k + l - 1]$
 - 6 $X = \text{dist}(s, D)$ {distance between s and each non-target time series in D }
 - 7 $\delta = \text{ChebyshevInequality}(X)$ {Equation 3.2}
 - 8 $S := (s, l, \delta, \text{Class}(T))$
 - 9 $\text{ComputeUtilityScore}(S)$ {Equation 3.3}
 - 10 $\text{Add}(S, \text{ShapeletList})$ {sorted list}
 - 11 *ShapeletList* = $\text{PruneShapelets}(\text{ShapeletList})$
-

Table 3.1: Evaluation of the EDSC method, using the Chebyshev’s inequality as distance threshold method and the weighted recall as the utility score method, on each gene of the drug response and viral infection datasets. The best accuracy is reported.

Dataset	gene	Accuracy	Relative Accuracy	Coverage	Earliness
H3N2	LOC26010	77.78	85.71	100	38.34
HRV	RSAD2	42.86	80.00	55.56	52.50
Baranzini3A	Caspase 10	12.00	100.00	12.25	42.86
Baranzini3B	Caspase 3	26.09	80.00	31.38	40.26
Baranzini6	Caspase 10	12.00	100.00	12.25	42.86
Baranzini12	Caspase 3	26.09	80.00	31.38	40.26
Lin9	Caspase 3	26.09	80.00	31.38	40.26
Costa17	Caspase 3	26.09	80.00	31.38	40.26

Experiments

The EDSC method was evaluated on seven univariate time series datasets from the UCR time series archive (*Keogh et al., 2011*). The results showed that the local shapelets extracted by the EDSC methods are highly interpretable and can achieve effective early classification. The experiments results are omitted here and the reader are highly encouraged to read the original paper (*Xing et al., 2011b*). However, the method is applicable only to univariate time series not multivariate time series. We applied the method to our multivariate time series drug response datasets (Section 1.3.1) and viral infection dataset (Section 1.3.2). We applied the EDSC method on each gene of the dataset separately and report the gene that gives the best accuracy for each dataset. The results are shown in Table 3.1. The EDSC methods has relatively high relative accuracy but very low coverage (the number of patients that the method was able to classify). The results show that the univariate early classification EDSC method does not give good accuracy on multivariate time series datasets. In the next section, we extend the interpretable EDSC method to multivariate time series case.

3.2 Interpretable early classification of multivariate time series

In medical application, several variables have to be considered to have a clear explanation about the dynamic behavior of the disease. For example, sepsis, a medical condition characterized by uncontrolled inflammatory response due to infection, is one of the main causes of deaths in the intensive care units, with over 750,000 cases annually in the United States alone (Zuev *et al.*, 2006). One of the main reasons for such a high number of death cases lies in limited understanding and knowledge about the complex inflammatory response mechanism, which has led to only a few effective sepsis therapies. In this section, we propose two methods for early classification of multivariate time series that address the drawbacks of the univariate EDSC method (Ghalwash and Obradovic, 2012; Ghalwash *et al.*, in review).

3.2.1 Multivariate Shapelet Detection

We generalize the definition of local shapelets to a multivariate context and accordingly propose a method for early classification of multivariate time series. The proposed method is called *Multivariate Shapelets Detection* (MSD). A multivariate shapelet consists of multiple segments, where each segment is extracted from exactly one dimension. The test time series is then classified based on the multivariate shapelet that best match the test time series.

As in the EDSC method, the core of the MSD method is the shapelet. We give the notion of the multivariate shapelet using the following example and then define the multivariate shapelet in a formal way.

Example 3.2.1. *Figure 3.5 shows an example of a 3-dimensional time series of*

length 15. It shows an example of an extracted 3-dimensional shapelet of length 4. The multivariate shapelet is composed of three segments where each segment is extracted from a different dimension from the time series from position 6 to position 9.

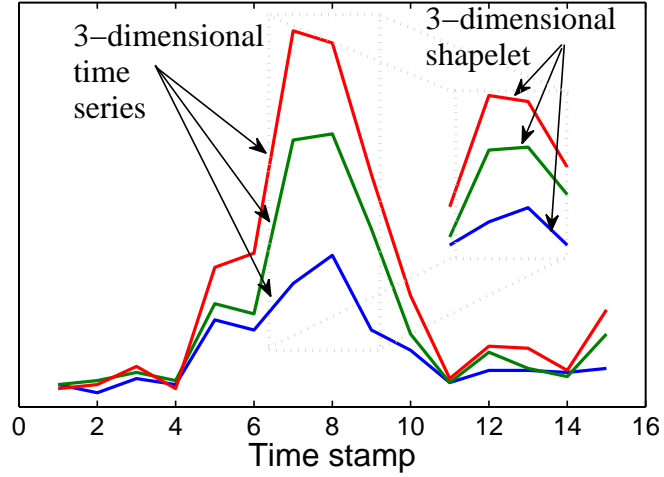


Figure 3.5: A 3-dimensional shapelet of length 4 is illustrated in the top-right part of the figure. The shapelet has three dimensions where each dimension (red, green and blue lines) is extracted from the corresponding dimension of the multivariate time series. The multivariate shapelet is extracted from position 6 to position 9.

Definition 3.2.1 (N -dimensional shapelet). An N -dimensional shapelet (N -shapelet) of length l is defined as $\mathbf{f} = (\mathbf{s}, l, \Delta, c_f)$ where c_f is the class of the shapelet. The vector $\mathbf{s} = [s^1, s^2, \dots, s^N]$ where s^j is the j^{th} dimension of the shapelet. The distance threshold $\Delta = [\delta^1, \delta^2, \dots, \delta^N]$ where δ^j is computed (as explained later) so that for each N -dimensional time series $\mathbf{T} = [T^1, T^2, \dots, T^N]$ of class c_f , the distance between the corresponding dimensions of the time series and the shapelet is less than or equal to the distance threshold. In other words,

$$\forall(\mathbf{T}, c_f) \in D \Rightarrow \text{dist}(s^j, T^j) \leq \delta^j \quad \forall j = 1 \dots N$$

The basic idea of the MSD method is to extract all multivariate shapelets and

then prunes the shapelets and keeps only the informative multivariate shapelets for early classification. Therefore, in a dataset of N -dimensional time series, the method extracts all N -dimensional shapelets $\mathbf{f} = (\mathbf{s}, l, \Delta, c_f)$. The method assumes that all subsequences s^j of the shapelet are extracted from the same starting position. For example, in Figure 3.5, the subsequences s^1 , s^2 , and s^3 (red, green, and blue lines) start from the same starting position 6 and have the same length 4. Hence, we slide a window of length l over the time series. At each time stamp k , a subsequence s^j of length l starting from the time point k is extracted from the j^{th} dimension to construct $\mathbf{s} = [s^1, s^2, \dots, s^N]$. An example of a 3-dimensional shapelet is shown in Figure 3.5. The the distance threshold parameter Δ of the shapelet \mathbf{f} is undefined yet. To define Δ , we need to compute the minimum distance between \mathbf{f} and every time series \mathbf{T} in the dataset. Since the shapelet and the time series are multivariate, then we compute the distance between the corresponding dimensions of the shapelet and the time series. Therefore, the distance between \mathbf{f} and \mathbf{T} is a vector of distances (N -dimensional distance) and is computed as in Equation 1.2:

$$\text{dist}(\mathbf{s}, \mathbf{T}) = [\text{dist}(s^1, T^1), \text{dist}(s^2, T^2), \dots, \text{dist}(s^N, T^N)]$$

Example 3.2.2. *Figure 3.6 shows an example of a 3-dimensional shapelet $\mathbf{f} = (\mathbf{s}, l, \Delta, c_f)$ where the length of the shapelet $l = 4$, $\mathbf{s} = [s^1, s^2, s^3]$, and $\Delta = [\delta^1, \delta^2, \delta^3]$. Based on Definition 1.4.5, the distance between the subsequence s^1 and the time series T^1 is the minimum distance between s^1 and all subsequences of length 4 of the time series T^1 . So, $\text{dist}(s^1, T^1) = 8.66$. Also, $\text{dist}(s^2, T^2) = 28.28$ and $\text{dist}(s^3, T^3) = 36.06$. Therefore, the distance $\text{dist}(\mathbf{f}, \mathbf{T}) := \text{dist}(\mathbf{s}, \mathbf{T}) = [8.66, 28.28, 36.06]$.*

So, we compute the distance between the shapelet and each time series in

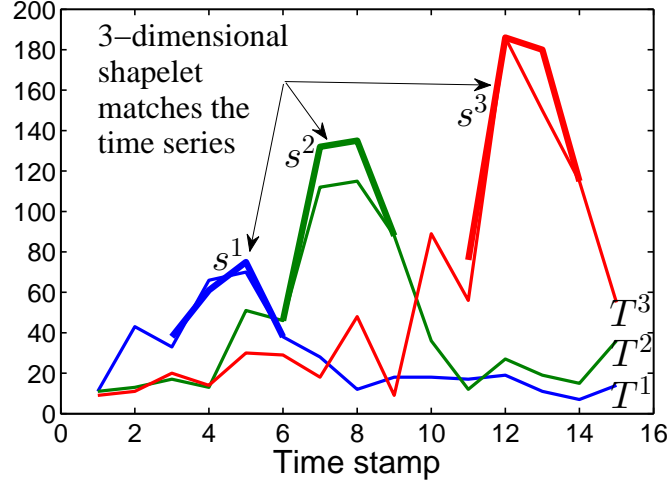


Figure 3.6: A 3-dimensional shapelet $\mathbf{f} = ([s^1, s^2, s^3], 4, [\delta^1, \delta^2, \delta^3], c_f)$ matches the time series $\mathbf{T} = [T^1, T^2, T^3]$. The distance between s^1 and T^1 is 8.66, between s^2 and T^2 is 28.28, and between s^3 and T^3 is 36.06. Therefore, $\text{dist}(\mathbf{f}, \mathbf{T}) = [8.66, 28.28, 36.06]$.

the dataset. Each distance is a vector of length N (the number of dimensions of the time series) as in Example 3.2.2. To compute the distance threshold of a shapelet, we need to sort the distances to find the threshold that maximizes the information gain. So, we need to provide a way to compare two multi-dimensional distances. The following definition defines how to compare two multidimensional distances.

Definition 3.2.2 (Ordered distances). *Two multidimensional distances $\mathbf{d}_1 = [d_1^1, d_1^2, \dots, d_1^N]$ and $\mathbf{d}_2 = [d_2^1, d_2^2, \dots, d_2^N]$ are defined to be ordered according to the following criterion:*

$$\mathbf{d}_1 < \mathbf{d}_2 \Leftrightarrow d_1^j < d_2^j \quad \forall j = 1 \dots N \quad (3.4)$$

Equation 3.4 requires all N dimensions of \mathbf{d}_1 to be less than all corresponding N dimensions of \mathbf{d}_2 . For example, if we match a shapelet with a time series then we would require all N dimensions of the time series to be less than the shapelet's threshold (N -dimensional vector).

Example 3.2.3. *Suppose that the extracted shapelet is $\mathbf{f} = ([s^1, s^2, s^3], 4, \Delta, c_f)$ as in Figure 3.6, where $\Delta = [\delta^1, \delta^2, \delta^3] = [10, 30, 40]$. Then, the shapelet \mathbf{f} matches the time series \mathbf{T} if the distance $\text{dist}(\mathbf{f}, \mathbf{T}) < \Delta$ which means $\text{dist}(s^j, T^j) < \delta^j \forall j = \{1, 2, 3\}$. In other words, each component of the multivariate shapelet has to match the corresponding dimension of the time series: the red component of the shapelet has to match the red dimension of the time series, the blue component of the shapelet has to match the blue dimension of the time series, and the green component of the shapelet has to match the green dimension of the time series.*

This way, the method would try to find a pattern very similar to the shapelet at hand, which could lead to overfitting. In order to prevent overfitting, Definition 3.2.2 is relaxed and redefined to be partially ordered according to the following definition.

Definition 3.2.3 (Partially ordered distances). *Two multidimensional distances $\mathbf{d}_1 = [d_1^1, d_1^2, \dots, d_1^N]$ and $\mathbf{d}_2 = [d_2^1, d_2^2, \dots, d_2^N]$ are defined to be partially ordered according to the following criterion:*

$$\mathbf{d}_1 <_{Perc} \mathbf{d}_2 \Leftrightarrow d_1^{q_j} < d_2^{q_j} \quad \forall j = 1 \dots Perc \times N \quad (3.5)$$

where $Perc \in]0, 1]$.

Therefore, \mathbf{d}_1 is less than \mathbf{d}_2 if a certain number of dimensions (not all dimensions) satisfy the “less than” operation. In other words, the multivariate shapelet match the time series if some components of the shapelet match the corresponding dimensions of the time series.

We explain the 4 steps of the MSD method to extract discriminative multivariate shapelets and use them for early classification.

Step 1: Extracting all Multivariate Shapelets

The MSD iterates over all time series in the dataset D . Then, for each time series \mathbf{T} , it slides a windows of specified length l over the time series. Therefore, for each starting position k over the time series \mathbf{T} , the MSD method extracts a subsequence $s^j = T^j[k \dots k + l - 1]$ of length l starting from the position k from each dimension $j = \{1, 2, \dots, N\}$ of the time series and consider all these components s^j as an N -dimensional shapelet $\mathbf{f} = ([s^1, s^2, \dots, s^N], l, \Delta, c_f)$ where c_f is the label of the time series \mathbf{T} and Δ will be computed later.

Example 3.2.4. *Suppose that we have a 3-dimensional time series \mathbf{T} of length 15 as in Figure 3.5. The total number of the extracted 3-dimensional shapelets of length 4 is 12. An example of one of those 3-dimensional shapelets is illustrated in the top-right part of Figure 3.5. The illustrated shapelet is defined as $\mathbf{f} = ([s^1, s^2, s^3], 4, \Delta, c_f)$ where $s^j = T^j[6 \dots 9]$, c_f is the label of the time series T , and Δ will be defined later.*

Suppose that the dataset D has M multivariate time series where the maximum length of the time series is L . Then, the maximum total number of the extracted multivariate shapelets from the dataset is $M \sum_{l=\min L}^{\max L} l(L - l + 1)$ where $\min L$ and $\max L$ are user parameters define the minimum and maximum length of the extracted shapelet, respectively. Then, for each extracted shapelet, the distance threshold Δ of the extracted shapelet is computed (Algorithm 2, line 7).

Xing et al. (2011b) proposed Cheybshev’s inequality as a distance threshold learning method for univariate shapelet. The same method could be adapted for multivariate distance threshold considering Definition 3.2.3 for comparing multivariate distances. *Ye and Keogh (2009)* proposed information gain for learning distance threshold for univariate shapelet. We propose the information gain method

Algorithm 2: Extraction of N -dimensional shapelets

Input: A training dataset D of M N -dimensional time series; $minL$; $maxL$ (user parameters for minimum and maximum shapelet's length)

Output: A list *ShapeletList* of all extracted multivariate shapelets

```
1 ShapeletList = empty {sorted list of all shapelet}
2 for  $i = 1$  to  $M$  do {each subject time series}
3   for  $l = minL$  to  $maxL$  do {each shapelet's length}
4     for  $k = 1$  to  $L - l + 1$  do {each start position}
5       Define  $s^j = T^j[k \dots k - l + 1] \forall j = \{1, 2, \dots, N\}$ 
6       Define shapelet  $\mathbf{f} = ([s^1, s^2, \dots, s^N], l, \Delta, c_f)$ 
7       Compute distance threshold  $\Delta$ 
8       Compute utility score for  $\mathbf{f}$ 
9       Add  $\mathbf{f}$  to ShapeletList
10 ShapeletList = PruneShapelets(ShapeletList)
```

for multivariate shapelet.

To compute the distance threshold of a shapelet, we need to compute the distance between the shapelet and each time series in the dataset. Let us assume that we have a dataset of M multivariate time series and a shapelet $\mathbf{f} = (\mathbf{s}, l, \Delta, c_f)$ where $\mathbf{s} = [s^1, s^2, \dots, s^N]$. All parameters of the shapelet \mathbf{f} are defined except the distance threshold Δ . Let us assume that all time series of the class c_f are called target time series and all other time series are called non-target time series. Then, we compute the distance between the shapelet \mathbf{f} and each time series using Definition 1.4.9. Let us assume that $\mathbf{d}_i = [d_i^1, d_i^2, \dots, d_i^N]$ is the distance between the shapelet \mathbf{f} and the time series $\mathbf{T}_i = [T_i^1, T_i^2, \dots, T_i^N]$ where $d_i^j = dist(s^j, T_i^j) \forall i = \{1, 2, \dots, M\}$ and $j = \{1, 2, \dots, N\}$.

Multivariate Cheybshev's inequality: The distance threshold Δ of a multivariate shapelet $\mathbf{f} = (\mathbf{S}, l, \Delta, c_f)$ is computed such that the probability of a non-target time series (time series of the other classes than c_f) that matches the shapelet \mathbf{f} is less than $1/k^2$ where k is a user defined parameter. Let X^j be a random variable that represents the distances d_i^j between the shapelet and all

non-target time series in the dataset D . Let μ^j and σ^j be the finite expected value and non-zero standard deviation of X^j , respectively. Then, for any real number $k > 0$, the Chebyshev's inequality ensures that (*Allen (1990)*)

$$Pr(|X^j - \mu^j| \geq k\sigma^j) \leq \frac{1}{k^2}$$

The Chebyshev's inequality considers only the distance \mathbf{d}_i between the shapelet and each *non-target* time series for each dimension separately. It treats these distances as the random variable X^j . Therefore, the multivariate shapelet's distance threshold $\Delta = [\delta^1, \delta^2, \dots, \delta^N]$ is computed such that:

$$\delta^j = \max(\mu^j - k * \sigma^j, 0) \tag{3.6}$$

In other words, we compute the distance between the shapelet and each non-target time series. Then, the threshold for each dimension is computed separately. As evident by our experiments that this distance threshold method does not provide accurate results. Therefore, we propose multivariate information gain which outperformed the Chebyshev's inequality on the medical datasets considered in this thesis.

Multivariate information gain: The basic idea is to find the shapelet's distance threshold that maximizes the information gain and divides the dataset into two groups: target and non-target time series (*Ye and Keogh, 2009*). First, the entropy of the dataset is computed as

$$Entropy = - \sum_{c \in C} \frac{m_c}{M} \log\left(\frac{m_c}{M}\right) \tag{3.7}$$

where C is the set of all classes of the time series, m_c is the number of time series of class c , and M is the number of all time series. To compute the distance threshold Δ of the shapelet \mathbf{f} , the MSD method sorts the distances \mathbf{d}_i between the shapelet and all time series \mathbf{T}_i considering Definition 3.2.3 for comparing the distances. Then, it finds the mid point between two consecutive distances as a candidate for the threshold. The mid point between two consecutive distances \mathbf{d}_i and \mathbf{d}_p is computed as $\mathbf{d}_{\text{mp}} = [\delta^1, \delta^2, \dots, \delta^N]$ where $\delta^j = \frac{d_i^j + d_p^j}{2} \quad \forall j = \{1, 2, \dots, N\}$.

The dataset is then divided into two datasets D_L and D_R . The dataset D_L contains all time series such that the distance between the shapelet and time series is less than or equal to the candidate threshold (Definition 3.2.3). The dataset D_R contains the rest of the time series. Then the entropies E_L and E_R of the datasets D_L and D_R are computed, respectively. By comparing the entropy before and after the split, we obtain a measure of information gain which is computed as

$$IG = Entropy - \frac{M_L}{M} E_L - \frac{M_R}{M} E_R \quad (3.8)$$

where M_L and M_R are the number of time series in D_L and D_R , respectively. Therefore, we compute the information gain for each candidate threshold and choose the distance threshold $\Delta = [\delta^1, \delta^2, \dots, \delta^N]$ that has the maximum information gain for the shapelet.

Now, all parameters of the shapelet $\mathbf{f} = (\mathbf{s}, l, \Delta, c_f)$ are well defined. The number of the extracted shapelets from the time series dataset is large. Therefore, the MSD method assigns a score to each shapelet (Algorithm 2, line 8) such that it sorts the shapelets and then prunes them to select few discriminative shapelets (Algorithm 2, line 10) for the early classification task. The score assigned to the shapelet should consider the earliness and the accuracy performance of the shapelet on the training data.

Step 2: Ranking all Multivariate Shapelets

Xing et al. (2011b) proposed weighted F_1 score (Equation 3.3) as a utility score for univariate shapelet. The same method could be adapted for multivariate shapelet considering Definition 3.2.3 for comparing multivariate distances. We propose the information gain method for assigning a utility score for the multivariate shapelet.

The utility score of a shapelet should incorporate the earliness and the distinctiveness properties. First, the MSD method computes how early each component of the shapelet matches the corresponding dimension of the time series. Let us assume that we have a shapelet $\mathbf{f} = (\mathbf{s}, l, \Delta, c_f)$ and a time series $\mathbf{T}_i = [T^1, T^2, \dots, T^N]$ of length L . Then, the earliness of the component s^j of the shapelet is defined as

$$EML(s^j, T^j) = \min_{\forall k \in \{1, 2, \dots, L-l+1\}} dist(s^j, h_k^j) \leq \delta^j$$

where h_k^j is a subsequence of T^j . If the distance between s^j and all subsequences h_k^j are greater than the threshold δ^j then the $EML(s^j, T^j)$ is defined to be ∞ .

EML measures how early the component s^j of the shapelet \mathbf{f} has classified the corresponding dimension T^j of the time series \mathbf{T} . Since, we are using only a percentage $Perc$ when comparing two multivariate distances (Definition 3.2.3) and the same principle when the shapelet matches a time series, then we should use the same principle when computing the earliness. Therefore, the earliness between a multivariate shapelet and a time series is computed based on $Perc$ of the components of the shapelets. Whenever a $Perc$ of the components of the shapelet match the time series ($dist(s^j, T^j) \leq \delta^j$), then the earliness of the shapelet is computed based on only those matched components. In other words, the earliness

is computed as:

$$EML(\mathbf{f}, \mathbf{T}) = \max_{j=\{1,2,\dots, Perc \times N\}} EML(s^{q_j}, T^{q_j})$$

where $EML(s^{q_j}, T^{q_j}) \neq \infty$. If the number of matched components of the shapelet is less than $Perc \times N$ then $EML(\mathbf{f}, \mathbf{T}) := \infty$.

Example 3.2.5. Figure 3.7 shows an example of a 3-dimensional shapelet \mathbf{f} where only two components of the shapelet (blue and green) match the time series. If $Perc = 0.65$ then the shapelet matches the time series if at least $0.65 \times 3 \approx 2$ components match the time series. Based on that definition, the earliness $EML(\mathbf{f}, \mathbf{T})$ is computed based on only those two matched components (blue and green components). Since, the EML of the blue and green components are 6 and 9, respectively. Therefore, $EML(\mathbf{f}, \mathbf{T}) = 9$.

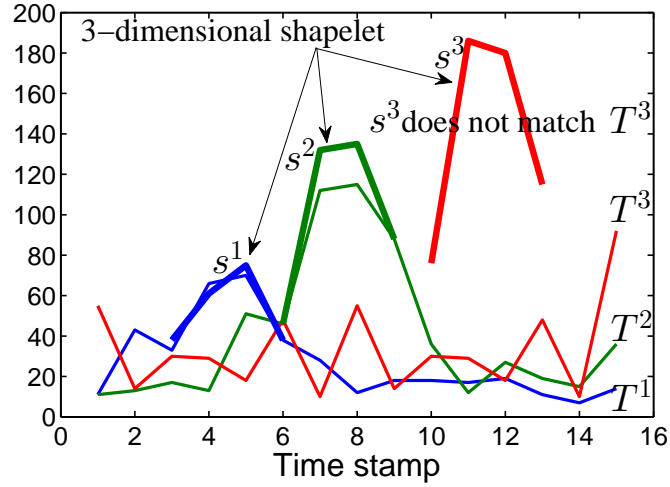


Figure 3.7: Two components (blue and green) of the shapelet match the time series. Therefore, the earliness EML is computed based on only those two components. $EML(\mathbf{f}, \mathbf{T}) = 9$.

Weighted F_1 score: The weighted F_1 score considers the accuracy performance of the shapelet as well as the earliness on the training dataset. To compute the

score, we compute the *EML* between the shapelet and each time series in the dataset using Equation 3.2.1. Then, the utility score of the shapelet is computed using Equation 3.3.

Weighted information gain: The weighted information gain of the shapelet is computed as follows:

1. Compute the distance between the shapelet $\mathbf{f} = (\mathbf{s}, l, \Delta, c_f)$ and every time series \mathbf{T}_i in the dataset D .
2. Split the dataset D into two datasets D_L and D_R such that D_L contains all time series where $dist(\mathbf{f}, \mathbf{T}_i) \leq \Delta$ (Definition 3.2.3) and D_R contains the rest of the time series.
3. For each time series T in the dataset D_L , if $Class(\mathbf{T}) = c_f$, then \mathbf{T} is weighted by $EML(\mathbf{f}, \mathbf{T})$. Otherwise, the time series is weighted by 1.
4. Compute M_L as the weighted count of the number of time series in the dataset D_L and M_R is the size of the dataset D_R .
5. Compute the weighted information gain using Equation 3.8.

The following theorem proves that the weighted information gain incorporates the earliness and assigns high utility score to the shapelet that has better earliness given the same accuracy performance.

Theorem: If \mathbf{f}_1 and \mathbf{f}_2 are two shapelets that have the same distance threshold (same splitting point), the same class, and different earliness (\mathbf{f}_1 has better earliness than \mathbf{f}_2), then \mathbf{f}_1 has better weighted information gain than \mathbf{f}_2 .

Proof: Suppose that the number of target time series in D_L is N_T and the number of non-target time series in D_L is N_{NT} . Without loss of generality, since \mathbf{f}_1

has better earliness than \mathbf{f}_2 , suppose that for every target time series \mathbf{T} in D_L , $EML(\mathbf{f}_1, \mathbf{T}) = P_1$ and $EML(\mathbf{f}_2, \mathbf{T}) = P_2$ such that $P_1 < P_2$. The weighted count M_{L1} and M_{L2} of the time series in D_L for \mathbf{f}_1 and \mathbf{f}_2 is $P_1 N_T + N_{NT}$ and $P_2 N_T + N_{NT}$, respectively. Since $P_1 < P_2$, then $M_{L1} < M_{L2}$. Hence the weighted information gain of \mathbf{f}_1 is greater than the weighted information gain of \mathbf{f}_2 .

After computing the score for each shapelet, the method sorts them in descending order according to their utility scores and then selects a subset of shapelets for the early classification task.

Step 3: Pruning all Multivariate Shapelets

Two methods for pruning the shapelet have been used by the MSD method:

Cover pruning method: The MSD method sorts the shapelets descending based on their utility scores. Then, it starts with the highest ranked shapelet and removes all time series from the dataset that are matched by the shapelet. The shapelet $\mathbf{f} = (\mathbf{s}, l, \Delta, c_f)$ **covers** the time series \mathbf{T} if the shapelet matches the time series ($dist(\mathbf{f}, \mathbf{T}) \leq \Delta$ based on Definition 3.2.3) and have the same class as the shapelet. Then, MSD stores the shapelets (Algorithm 2, line 9) and the next highest shapelet is considered. If the shapelet covers any of the remaining time series, the shapelet will be added to the extracted list and all covered time series will be removed. The method iteratively does so until all time series in the dataset are covered. In this manner, the MSD method ends up with a small list of shapelets that is used for the classification purposes.

Top x pruning method: The second method simply involves keeping the top x shapelets from each class where x is a user-defined parameter. In our experiments,

we used the top 5, 10, 15 and 20 shapelets from each class.

Step 4: Classification Phase

The classification process of the MSD method follows the framework of the early classification described in Figure 1.1. The MSD method initially reads a portion of length $minL$ from the test time series (where $minL$ is the length of the shortest extracted shapelet). The highest-ranked shapelet is considered. If the shapelet matches the time series then the time series is classified as the class of the shapelet and the prediction is made. Otherwise, the next shapelet from the ranked list is considered and the process of checking each shapelet is repeated. If none of the shapelets match the current portion of the test time series the method reads one more time stamp and continues classifying the time series (Figure 1.1). If the method reaches the end of the time series and none of the shapelets matches it, MSD marks the time series as a not-classified example.

Experiments

In all experiments we set $minL = 3$ and $maxL = 0.6 \times L$ where L is the maximum time series' length in the dataset. Since the number of subjects was small, bootstrapping was used for estimating the generalization error (*Lendasse et al.*, 2003; *Jain et al.*, 1987). We sample with replacement a subset (75%) from the original dataset. We train our model on the sample data and then test it on the subjects that are not used in the training data. This process is repeated 1000 times and the final reported statistics is the median of the statistics over all bootstrap samples. We report the median instead of the average since the distribution of the statistics is skewed and not symmetric.

In the results, we report the median of the accuracy (Definitions 2.2.4 and 2.2.5), the coverage (Definition 2.2.2), and the earliness (Definition 2.2.3). Note

that the earliness varied from test example to another. In other words, each test example could be classified at different time point, so that our method is patient-specific and there is no fixed length of the time series used for classification.

Real Case: First, we show the effectiveness of the MSD method on a single patient from the H3N2 dataset. Figure 3.8b shows genes RSAD2 and IFI44L observed at 15 time steps for an asymptomatic test subject from H3N2 data that is correctly and early classified by MSD at the 5th time point. The MSD method used a shapelet of length 5 to classify the test subject. In Figure 3.8a, MSD used a shapelet of length 6 that was extracted from the time series of a symptomatic subject, so it correctly classified the symptomatic test subject at the 8th time point (it used only 50% of the time series' length to classify the test subject).

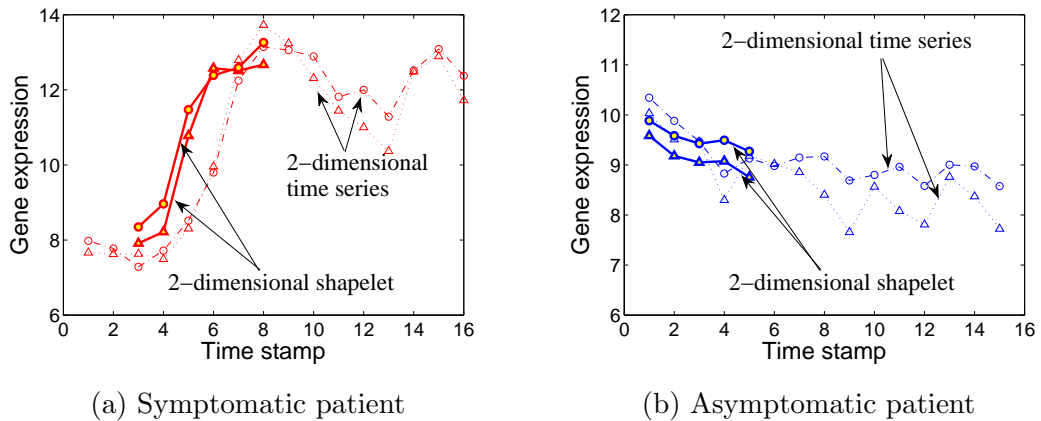


Figure 3.8: The effectiveness of the MSD method is illustrated on a single patient from H3N2. (a) A 2-dimensional H3N2 symptomatic test subject (genes RSAD2 and IFI44L observed at 16 time steps) has been correctly classified by MSD method at the earliest possible time stamp number 8. (b) A 2-dimensional H3N2 asymptomatic test subject (genes RSAD2 and IFI44L observed at 15 time steps) has been correctly classified by MSD method at the 5th time point. Red lines represent time series of the symptomatic subject. Blue lines represent time series of the asymptomatic subject. Shapelets are represents by solid markers.

Evaluation of MSD on viral infection and drug response datasets: The MSD method was evaluated on the viral and drug response datasets using all genes

defined by the dataset. In Table 3.2, we report the median of the coverage, the accuracy, the relative accuracy, and the earliness.

Table 3.2: The performance of the MSD method on 8 datasets. The accuracy (Acc), relative accuracy (Rel Acc), coverage (Cov), and the earliness (Ear) are reported. The list of the parameters (*Perc* and Pruning method) that have been used for each method is provided in the last two columns. The MSD method achieved good accuracy on most of the datasets using a small fraction of the time series.

Dataset	# genes	Acc	Rel Acc	Cov	Ear	<i>Perc</i>	Pruning
H3N2	23	77.78	85.71	100	62.50	0.2	Cover
HRV	26	70.00	71.43	100	40.00	0.6	Top10
Baranzini3A	3	70.00	73.91	95.83	46.26	0.7	Top15
Baranzini3B	3	66.67	68.00	100	44.81	0.7	Top15
Baranzini6	6	70.83	70.83	100	42.86	0.6	Top15
Baranzini12	12	66.67	66.67	100	42.86	0.7	Top10
Lin9	9	67.86	69.57	100	44.00	0.9	Top20
Costa17	17	68.00	69.23	100	45.24	0.3	Top5

As shown in Table 3.2, it is clear that the MSD method achieved high accuracy using a small fraction of the time series. For example, MSD on the H3N2 dataset covered approximately 100% of the dataset, and out of the covered time series it achieved 85.71% accuracy using 62% of the time series' length. On another benchmark dataset called Lin9, the method developed in (*Lin et al., 2008*) achieved 85% accuracy using the full time series while our MSD method achieved approximately 68% accuracy using less than half of the time series' length on average.

Comparison between MSD and 2 alternative methods We compare the MSD method with the EDSC method (using the Chebyshev's inequality as distance threshold method and the weighted recall as utility score method). Since

EDSC is a univariate method, it has been tested on each gene in the dataset and the best accuracy achieved is reported. In addition, we compare the MSD method with the ECM method (Section 2.2). The ECM method is developed for multivariate time series therefore it is applicable directly to the dataset in hand.

Since we report both the accuracy and the earliness for each method, it might be not easy to compare all the three methods based on two evaluation measures. Therefore, we propose an evaluation measure that incorporates both the earliness (Ear) and the accuracy (Acc). We use F_β -measure as the weighted average between Acc and Ear . F_β -measure is defined as:

$$F_\beta = (1 + \beta^2) \frac{Acc \cdot (1 - Ear)}{\beta^2(1 - Ear) + Acc}$$

where smaller values of β put more weight on the earliness and larger values of β put more weight on the accuracy. Note that we use $(1 - Ear)$ because we want to penalize larger values of Ear . In our experiments, we used the balanced F_1 -score, which gives both the accuracy and the earliness the same weight. F_1 -score reaches its best value at 1 and worst score at 0.

As shown in Table 3.3, the MSD method significantly outperformed the EDSC method on all datasets except the H3N2 dataset, where they have similar accuracy but the EDSC method provided the classification much earlier. The reason of achieving less accurate results using MSD method as compared to the EDSC method may be due to the non-robustness of the MSD method to noisy variables so that MSD does not extract meaningful features from the multivariate data in an automated fashion. Therefore, Equation 3.5 is affected by the noise in the variables which may lead to poor discrimination among the classes.

On the other hand, MSD outperformed ECM on the viral infection dataset but not on many drug response datasets. The reason for that is ECM is composed

of generative model HMM which suffers from the limited number of patients in the training data. Viral infection dataset has approximately less than 20 patients that is why MSD outperformed ECM on these datasets. However, on the drug response dataset, the time series is too short (approximately 5-7 data points) to learn discriminative shapelet for early classification. We believe that is the reason that ECM outperformed MSD on these datasets.

Comparison of different threshold learning methods for MSD: The MSD method was evaluated on the viral infection and drug repose datasets using two different distance threshold methods. Namely, we compared the proposed distance threshold method multivariate information gain with the Chebyshev’s inequality method. We note that using Chebyshev’s inequality gives better relative accuracy, but on the other hand it gives worse coverage, which reduces the overall accuracy. We applied a paired t-test of the null hypothesis that the difference between the bootstrap accuracies for both methods are a random sample from a normal distribution with mean 0 and unknown variance, against the alternative that the mean is not 0. We applied the t-test at the 99% significance level. The results are shown in Table 3.4. Using Chebyshev’s inequality as a threshold method outperformed the information gain in only two datasets (Lin9 and Costa17) where in Costa17 the difference is not significant.

Comparison of different utility methods for MSD: The MSD method was evaluated on the viral infection and drug repose datasets using two utility score methods. Namely, we compared the proposed weighted information gain method with the weighted F_1 method. We applied a t-test to measure the significance of the difference at the 99% significance level. The results are shown in Table 3.5. Using weighted F_1 score as a utility score, the method outperformed the weighted

information gain in only one dataset (Baranzini6).

Run-time analysis: In Table 3.6, we show the run time of the MSD method on viral infection and drug response datasets. All experiments were conducted on a PC Intel Core i7 2.8 GHz with 8GB RAM. It is evident that the run time grows exponentially with the number of examples and the time series length.

Conclusion

The MSD method address one limitation of ECM which is interpretability. Physicians prefer to use interpretable method rather than more accurate black-box method. If there is enough number of observations for each patient as in the viral infection dataset, then MSD turns out to be as accurate as ECM that suffers from the limited number of patients. However, MSD might not be robust to noisy variables so that MSD does not extract meaningful features from the multivariate data in an automated fashion. Therefore, Equation 3.5 is affected by the noise in the variables which may lead to poor discrimination among the classes. In addition, the components of the extracted multivariate shapelet have the constraints of the same starting time point and are required to be of the same length as in Figure 3.5. In the next section, we propose an interpretable method that addresses all the aforementioned issues of MSD.

3.2.2 Extraction of Interpretable Multivariate Patterns for Early Discrimination

We propose an optimization-based method for building predictive models on multivariate time series data and mining relevant temporal Interpretable Patterns for Early Diagnosis (IPED) (*Ghalwash et al.*, in reviewa). The problem is formulated in IPED as an optimization-based binary classification task addressed

in three steps. First, the time series data are transformed into a binary matrix representation suitable for application of classification methods. Then, the binary matrix is used to define a full-dimensional shapelet for each class via solving a proposed novel convex-concave optimization problem. Then, the dimensionality of the extracted full-dimensional shapelets is reduced by solving a mixed integer discrete optimization problem. The extracted low-dimensional shapelets (key shapelets) are then used for early interpretable classification in challenging clinical applications. We are able to make our results interpretable by using only a few patterns from the observed time series data.

The IPED method addresses three issues in the MSD method. First, the components of the multivariate shapelet do not have the constraints of the same starting time point and are not required to be of the same length. Therefore, the onset of each pattern could be different, which simulates a real life scenario. Second, the extracted multivariate shapelet contains only the relevant dimensions of the observed multivariate time series data. We first start by sketching the IPED method to have a clear notion of the overall steps of the method and then we describe each step separately in more details.

Method Sketch A recent study notes “*transforming the data are the simplest way of achieving improvement in problems where the discriminating features are based on similarity in change and similarity in shape*” (Bagnall et al., 2012). Following this principle, we extract all time series subsequences of different lengths from each dimension of the multivariate time series. These subsequences are called univariate shapelets (or simply shapelets). The shapelet is used for discriminating between groups of time series. Therefore, we compute a distance threshold for each shapelet to maximize the information gain (Ye and Keogh, 2009). Using the shapelets, we construct a binary matrix, as in Figure 3.9a, where each element

of the matrix indicates the presence of the shapelet (column) in the subject time series (row) based on the shapelet’s distance threshold. The shapelets are organized into N groups such that each group contains all shapelets extracted from one dimension.

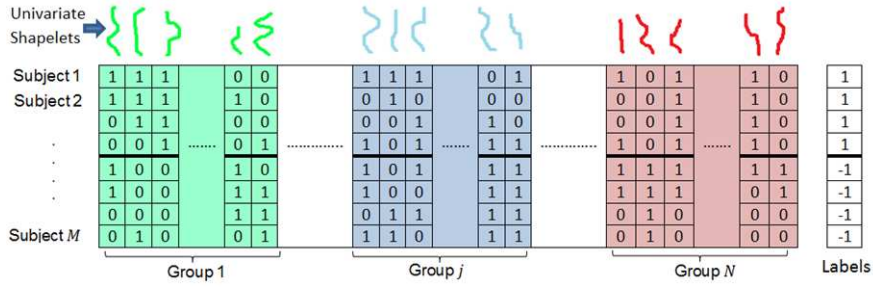
Next, for each class we extract a multivariate shapelet from the binary matrix such that only one shapelet from each group is extracted, as in Figure 3.9b. We call the extracted multivariate shapelet a *full-dimensional shapelet* because it contains a representative from each dimension of the observed time series. The problem of extracting a full-dimensional shapelet that maximizes the accuracy is formalized as an optimization problem, which is solved using convex-concave procedure (*Yuille and Rangarajan, 2003*).

As humans are able to estimate relatedness of only a few variables at a time (*Jennings et al., 1982*), and some variables of the time series might be irrelevant for early classification, we extract *key shapelets* from the full-dimensional shapelet as in Figure 3.9c. We formalize this problem as a mixed integer optimization problem. Clearly, the resulting dimension of the key shapelet is less than or equal to the dimension of the observed time series.

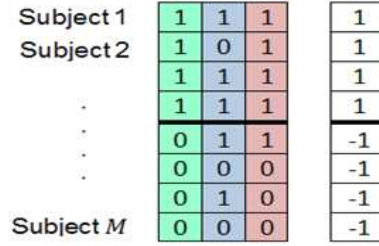
Then, we repeat that process to extract several full-dimensional shapelets and key shapelets for each class. Finally, we use all the extracted key shapelets, as a representative for the class, for early classification.

Step 1: Problem Transformation to a Binary Matrix Representation

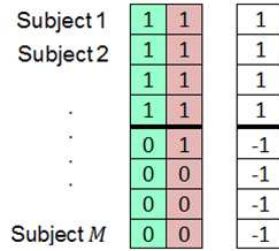
The IPED method starts by extracting all univariate shapelets from the multivariate time series dataset D . An effective method to extract all univariate shapelets is described in (*Chang et al., 2012b; Rakthanmanon and Keogh, 2011*). The IPED method iterates over all dimensions j of all M time series, and for each subject univariate time series T_i^j it extracts all subsequences S_{ikl}^j (univariate



(a) Constructing the binary matrix



(b) Extracting full-dimensional shapelet



(c) Extracting key shapelet

Figure 3.9: IPED framework. (a) Transformation of N -dimensional time series observations on M subjects (rows) into a binary matrix using shapelets (columns). The elements of the matrix indicate the presence of the shapelet in the subject. Each group (different color) contains all shapelets extracted from a single dimension. (b) Extraction of a full-dimensional shapelet from the binary matrix such that only one univariate shapelet from each group is extracted. (c) Extraction of key shapelets (fewer dimensional shapelet) from the full-dimensional shapelet.

shapelets) of length l that start at the k^{th} time point. Each univariate shapelet will be referred to as S_{ikl}^j which means the subsequence of length l that is extracted from j^{th} dimension of the time series T_i from the start position k .

Our objective is to extract a few key shapelets and to use them for discriminating between classes. The discriminative shapelets are those that are present

in the time series of one class but not in time series of another. Therefore, we transform our problem into a problem of selecting patterns from binary matrix which indicate presence of shapelets in time series. The transformation process is summarized in Algorithm 3.

Algorithm 3: Transformation of N -dimensional Time Series

Input: A training dataset D of M N -dimensional time series; $minL$; $maxL$ (user parameters for minimum and maximum shapelet’s length)

Output: A binary matrix F where rows represent subjects and columns represent shapelets

```

1 for  $j = 1$  to  $N$  do {each dimension}
2   for  $i = 1$  to  $M$  do {each subject time series}
3     for  $l = minL$  to  $maxL$  do {each shapelet’s length}
4       for  $k = 1$  to  $L - l + 1$  do {each start position}
5          $dist = ComputeDist(S_{ikl}^j, D)$ 
6         Compute distance threshold  $d_{ikl}^j$ 
7         Construct column feature  $F_{ikl}^j$ 

```

For each extracted univariate shapelet S_{ikl}^j we compute the distance between the shapelet and j^{th} -dimension of each of M multidimensional time series in the dataset using the function $ComputeDist$. The distance between S_{ikl}^j and time series T^j is previously defined in Definition 1.4.5 (the minimum distance between S_{ikl}^j and each subsequence of length l from T^j). The function $ComputeDist$ returns an array $dist$ of length M where the m^{th} -element represents the distance between the shapelet S_{ikl}^j and the time series T_m^j as in Figure 3.10.

Next, we determine the distance threshold d_{ikl}^j (line 6) such that if the distance between the shapelet and any time series is less than threshold we say that the shapelet is present in (matches) the time series. The distance threshold is computed so that it maximizes the separation of the dataset into two classes by maximizing information gain (Ye and Keogh, 2009) (an ideally discriminative shapelet is the one that is present in all time series of one class but not in any

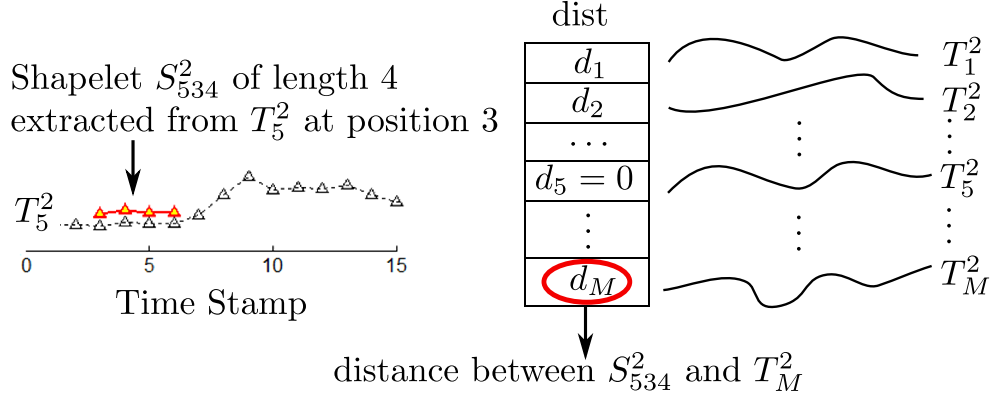


Figure 3.10: In this example, the shapelet S_{534}^2 (short red line) of length 4, is extracted from the 2^{nd} dimension of the time series \mathbf{T}_5 , of length 15, from the starting position 3. $dist$ is an array of distances d_m between the shapelet S_{534}^2 and the 2^{nd} dimension of each time series $\mathbf{T}_m : m = \{1, 2, \dots, M\}$. Note that the distance between S_{534}^2 and T_5^2 is zero because the shapelet is extracted from T_5^2 .

time series of another class), see Equation 3.8. Note that for a shapelet S_{ikl}^j that is extracted from j -th dimension we only determine its presence in j -th dimension of time series of all patients. We say that a shapelet S_{ikl}^j is present in (*matches*) the time series T^j if distance between S_{ikl}^j and T^j is less than a threshold d_{ikl}^j .

Consequently, we construct a binary vector F_{ikl}^j (line 7) which depicts the presence of the shapelet S_{ikl}^j in all training subjects (e.g. a column in Figure 3.9a). The m^{th} element in the vector F_{ikl}^j equals 1 if the distance between the shapelet S_{ikl}^j and the time series T_m^j is less than the shapelet's distance threshold d_{ikl}^j . Otherwise, it is 0. We call binary vector F_{ikl}^j the profile of the shapelet S_{ikl}^j (*Lines et al.*, 2012). For example, each column in Figure 3.9a represents a univariate shapelet profile.

Algorithm 1 returns a binary matrix F of size $M \times R$ where M is the number of subjects and R is the number of shapelets, where $R = NM \sum_{k=\min L}^{\max L} k(L-k+1)$. The columns of the matrix F are partitioned into N groups such that each group j contains all extracted shapelets from the dimension j as in Figure 3.9a.

Step 2: Extracting Full-dimensional Shapelets

The binary matrix F contains profiles for all shapelets from all dimensions of the time series. Having in mind the need for interpretability, for each class, our objective is to retain exactly one “class representative” univariate shapelet from each dimension of multivariate time series such that classification accuracy is maximized. The naïve approach to extract such representative shapelets is to look at each dimension separately and extract the shapelet which maximizes the accuracy (or minimizes classification loss) within the observed group. However, with this approach the possible interactions among shapelets from different groups are not taken into account.

We propose a novel method that simultaneously finds exactly one representative shapelet in each group. In order to mathematically define the problem that we are solving, let us assume that we have N groups of shapelets, where R^j is the number of shapelets in group j , $j = \{1, \dots, N\}$. We then assign weight w_i^j to each shapelet $i = \{1, \dots, R^j\}$ in group j which measures the importance of the shapelet in the classification. If we do not impose any restrictions on weights $W = \{w_i^j\}$, the optimal weights can be found by minimization of logistic loss over M subjects in the training data. The objective becomes minimization of logistic loss \mathcal{L}_1 with respect to the weights W .

$$\underset{W}{\text{minimize}} \underbrace{\sum_{m=1}^M \log(1 + e^{\overbrace{-c_m \sum_{j=1}^N \sum_{i=1}^{R^j} (w_i^j \cdot f_{im}^j)}^{\mathcal{I}_m}})}_{\mathcal{L}_1}} \quad (3.9)$$

where f_i^j represents the profile of the shapelet i from the group j , i.e. one column in the binary matrix F in Figure 3.9, f_{im}^j represents the m^{th} component of f_i^j , and c_m is the label for the m^{th} subject.

To be able to identify exactly one shapelet within each group we solve the following optimization problem with constraints on the weights W

$$\underset{W}{\text{minimize}} \quad \mathcal{L}_1 \quad (3.10)$$

$$\text{subject to} \quad 0 \leq w_i^j \leq 1, \quad \forall i, j, \quad (3.11)$$

$$\sum_{i=1}^{R^j} w_i^j = 1, \quad \forall j, \quad (3.12)$$

$$\max_{i=1 \dots R^j} w_i^j = 1, \quad \forall j. \quad (3.13)$$

Equation 3.11 indicates that all weights are bounded. The lower bound has to be 0 as we are interested only to extract representative shapelets for the positive class. The upper bound can be any positive real number. For simplicity we set the upper bound to 1. We need to impose this upper bound to be able to assure that exactly one weight within a group is not 0 and all other weights in the same group are equal to 0 by Equations 3.12 and 3.13. Equation 3.12 restricts weights to be normalized making sum of all weights within a group to be equal to 1. When Equation 3.12 is used together with Equation 3.13, which says that maximum weight within a group has to be equal to 1, we achieve that exactly one w_i^j within a group j is 1 while all other weights within a group j have to be 0.

The constrained optimization problem 3.10-3.13 is hard to solve since the *max* function is not differentiable. To be able to apply standard convex apparatus, we propose a transformation of 3.10-3.13 into a new optimization problem in which we: 1) relax hard equality constraints by introducing penalized terms in the objective function and 2) approximate the *max* function with convex differentiable *log-sum-exp* function. The objective function with relaxed hard equality

constraints becomes:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \overbrace{\mathcal{L}_1 + C_1 \cdot \sum_{j=1}^N \left(\sum_{i=1}^{R^j} w_i^j - 1 \right)^2}^{\mathcal{L}_2} \\ & + C_2 \cdot \sum_{j=1}^N \left(1 - \max_{i=1 \dots R^j} w_i^j \right) \end{aligned} \quad (3.14)$$

$$\text{subject to} \quad 0 \leq w_i^j \leq 1, \quad \forall i, j. \quad (3.15)$$

where we set the penalization parameters C_1 and C_2 to some values ($C_1 = C_2 = 0.1$ in our experiments). The intuition behind these penalization terms is that we would like to penalize the difference between sum of weights and 1, as well as maximum weight in each group and 1. We use quadratic penalty in the first term, as the sum of weights can be both greater or less than 1. We do not need quadratic penalty in the second term, as $\max_{i=1 \dots R^j} w_i^j$ is always less than or equal to 1 because of constraint 3.15. For that reason, we need to penalize the difference between 1 and max of weights without the need for squaring the term.

To get a differentiable optimization function we need to approximate \max with a smooth function. We start with the following lower bound for the \max function (*Boyd and Vandenberghe, 2004*)

$$\max_{i=1 \dots R^j} w_i^j \geq \log \left(\sum_{i=1}^{R^j} e^{w_i^j} \right) - \log R^j. \quad (3.16)$$

Then we have

$$1 - \max_{i=1 \dots R^j} w_i^j \leq 1 - \underbrace{\log \left(\sum_{i=1}^{R^j} e^{w_i^j} \right) + \log R^j}_{\mathcal{M}^j}. \quad (3.17)$$

This means that the second penalization term is upper bounded with smooth

log-sum-exp function. Penalizing the right hand side of (3.17) assures that the maximum is close to 1. If we combine (3.17) and (3.14) we get an optimization problem:

$$\underset{W}{\text{minimize}} \quad \mathcal{L}_1 + C_1 \cdot \mathcal{L}_2 + C_2 \cdot \mathcal{L}_3 \quad (3.18)$$

$$\text{subject to} \quad 0 \leq w_i^j \leq 1, \quad \forall i, j. \quad (3.19)$$

where $\mathcal{L}_3 = \sum_{j=1}^N \mathcal{M}^j$. The objective function (3.18) is convex-concave. \mathcal{L}_1 and \mathcal{L}_2 are convex while \mathcal{L}_3 is concave as it is equal to negative convex *log-sum-exp* function. Therefore, we can apply the convex-concave procedure (CCCP) (Collobert *et al.*, 2006; Yuille and Rangarajan, 2003) to find the global optimal solution. The application of CCCP to find optimal solution is shown in Algorithm 4.

Algorithm 4: Extract Full-Dimensional Shapelet

1 Initialize W^0

2 **repeat**

$$\mathbf{3} \quad W^{t+1} = \underset{W}{\operatorname{argmin}} \quad \overbrace{\mathcal{L}_1 + C_1 \cdot \mathcal{L}_2 + C_2 \cdot W \cdot \left(\frac{d\mathcal{L}_3}{dW} \right)_{W=W^t}}^{\mathcal{J}}$$

4 **until** Convergence of W $\{W^{t+1} - W^t \leq 0.01\}$

The term $(d\mathcal{L}_3/dW)_{W=W^t}$ is the derivative of \mathcal{L}_3 at the point W^t . The advantage of CCCP is that no additional hyper-parameters are needed. Furthermore, each update is a convex minimization problem and can be solved using classical and efficient convex apparatus.

Since we now have a smooth, differentiable objective function \mathcal{J} with only inequality constraints, we can use the trust-region-reflective algorithm for solving the problem (Coleman and Li, 1996). In order to quickly solve the problem we compute first derivatives of the objective function with respect to the weights W ,

and approximate Hessian with diagonal matrix (*Djuric et al., 2012*) as follows

$$\frac{\partial \mathcal{J}}{\partial w_i^j} = \sum_{m=1}^M \frac{\mathcal{I}_m \cdot (-y_m \cdot f_{im}^j)}{1 + \mathcal{I}_m} + 2 \cdot C_1 \cdot \left(\sum_{i=1}^{R^j} w_i^j - 1 \right) - C_2 \cdot \left(\frac{\exp^{w_i^j}}{\sum_{i=1}^{R^j} \exp^{w_i^j}} \right)_{W=W^t}, \quad (3.20)$$

$$\frac{\partial^2 \mathcal{J}}{\partial w_i^j \partial w_i^j} = \sum_{m=1}^M \frac{\mathcal{I}_m}{(1 + \mathcal{I}_m)^2} \cdot (y_m \cdot f_{im}^j)^2 + 2 \cdot C_1. \quad (3.21)$$

Step 3: Extracting Key Shapelets

As humans are able to perceive only a few variables at a time (*Jennings et al., 1982*), using the full-dimensional shapelets for early classification has several drawbacks, evidenced by our experiments reported in the experiment section. First, if the number of the dimensions of the time series is high, then it would be implausible for all components of the shapelet to cover all dimensions of the time series. This will affect the earliness of the method and the decision of the classification would come late, if it comes at all. Second, if some of the dimensions are irrelevant to the target, then these irrelevant dimensions will be subsequently inherited to the full-dimensional shapelet and would affect the overall accuracy of the method. Therefore, we propose a method to extract automatically, for each class, discriminating representative key shapelets by extracting all relevant dimensions from the full-dimensional shapelet.

Let's simplify the idea of the key shapelet with the following example. Assume that we have 8 subjects where the class labels of the subjects are defined in c . Suppose we have extracted a large dimensional (3-dimension) shapelet S with its profile \hat{S} as representative for the positive class (as in Figure 3.9b). We would like S to have maximum coverage for the positive class and minimum coverage for the negative class (*that is, present in all time series of the positive class but not*

in the time series of the negative class).

$$c = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \quad \hat{S} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \hat{S}' = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The full-dimensional shapelet S covers or matches a time series if each component of S covers the corresponding dimension of the time series. Since the first row of \hat{S} has three ones, it means that each component of the shapelet S covers the corresponding dimension of the time series for subject 1. Therefore, the shapelet S covers the first subject. The same for the 3rd and 4th subjects. The shapelet S does not cover the 2nd subject because the 2nd component does not cover the corresponding dimension of the time series of the 2nd subject. Then, S has positive coverage 75% (covers 3/4 of the subjects in the positive class). It is clear from the second column in \hat{S} that the 2nd component in S is noisy because it is represented three times in the positive class and two times in the negative class (1st, 3rd, 4th, 5th and 7th subjects).

Obviously, we can extract a key shapelet S' with only two components from S . S' has better positive coverage than S because it covers all positive subjects and does not cover any negative subject. Sometimes we can extract multiple key shapelets from one full-dimensional shapelet.

A recent paper proposed a novel mixed integer optimization approach for

extracting interpretable compact rules for the classification task (*Chang et al.*, 2012a; *Bertsimas et al.*, 2012). Thus, we adapt the approach to selecting several key shapelets from a full-dimensional shapelet. Here we review the method along with our modifications.

It is worth mentioning that we could minimize the logistic loss with L_1 and L_2 regularization terms (elastic net) instead of using the integer discrete optimization problem because the penalized logistic regression is much faster. However, properties of penalized logistic regression are not suitable for early classification which we will explain later in this section why elastic net does not work for our application. However, the results when applying penalized logistic regression to our datasets were worse than using a novel mixed integer optimization approach.

Let $b \in \{0, 1\}^N$ be a binary vector that encodes the presence of the N components in the key shapelet. For example, S' is encoded by the binary vector $b = [101]$ which means that only the first and third components are in the key shapelet S' . Let x_i be defined as

$$x_i = \begin{cases} 1 & \text{if } S' \text{ covers the subject } i, \\ 0 & \text{otherwise.} \end{cases}$$

The problem of selecting a key shapelet from the full-dimensional shapelet S can

then be formulated as the following optimization problem

$$\underset{b,x}{\text{maximize}} \quad \sum_{i_+} x_i - R_1 \sum_{i_-} x_i - R_2 \sum_{j=1}^N b_j \quad (3.22)$$

$$\text{subject to} \quad x_i \leq 1 + (S_{ij} - 1)b_j, \forall i, j, \quad (3.23)$$

$$x_i \geq 1 + (S_i - \mathbf{1}_N)^T b, \forall i, \quad (3.24)$$

$$\sum_{j=1}^N b_j \geq \mathcal{B}, \quad (3.25)$$

$$b_j \in \{0, 1\}, \quad (3.26)$$

$$0 \leq x_i \leq 1. \quad (3.27)$$

where $\mathbf{1}_N$ is an N -dimensional column vector of all ones and $i = \{1 \dots M\}, j = \{1 \dots N\}$ unless otherwise stated. i_+ are indices for all positive subjects and i_- are indices for all negative subjects. R_1, R_2 , and \mathcal{B} are parameters that will be discussed later.

The first term of the objective function (3.22) corresponds to maximizing the coverage of the shapelet for the positive class while the second term ensures that the shapelet is not represented in the negative class. The last term controls the sparsity of the multivariate shapelet. So, if there are several optimal solutions that maximize the coverage for the positive class and minimize the coverage in the negative class, we choose the one that has less components. R_1 and R_2 control the weights for those terms and have been set as $R_1 = 10$ to emphasize the importance of non-coverage for the negative class, and $R_2 = \frac{0.1}{MN}$ as was set previously (*Chang et al., 2012a*).

For the constraint (3.23), we have two cases for b_j . If $b_j = 0$ then the constraint is just $x_i \leq 1$ which has no effect. If $b_j = 1$ then the constraint becomes $x_i \leq S_{ij}$ which means that $x_i = 0$ if the subject i is not covered by the component

j . The constraint (3.24) explains the case when the subject i is covered by the component j . Therefore, the constraints (3.23) and (3.24) combined ensure that $x_i = 1$ if and only if all the components of the shapelet cover all the corresponding dimensions of the subject i (*this is the main reason why this approach outperformed the elastic net approach in early classification. Elastic net allows some components of the shapelets to be zero. Although it seems to be an advantage for elastic net, it did not work in early classification context*).

The constraint (3.25) determines the minimum number of components of the multivariate shapelet. Complex shapelets (high values of \mathcal{B}) increase the accuracy performance of the method because \mathcal{B} dimensions of the time series would be covered. However, this might affect the earliness. Therefore, the multivariate shapelet can not be neither too simple nor too complex.

By solving the problem (3.22), we obtain one key shapelet. Sometimes, there are several optimal key shapelets that could be extracted from the same full-dimensional shapelet. Therefore, we need to resolve the problem (3.22) to obtain more key shapelets. However, we need to make sure that the second time we solve the problem, we will not get one of the previous solutions. So, each time we solve (3.22) we add the following constraint

$$1 - \sum_{j:b_j^*=0} b_j - \sum_{j:b_j^*=1} (1 - b_j) \leq 0, \quad (3.28)$$

where b^* is the previous solution. The constraint (3.28) ensures that the new solution does not equal to the previous solution. So, the final key shapelets would have different components.

Therefore, we initially solve problem (3.22)-(3.27) and get an optimal solution. Then we add a new constraint (3.28) and resolve the problem (3.22)-(3.27) to get a new optimal key shapelet. We repeat this process by solving (3.22) until

the solution has positive coverage (number of positive subjects covered by the shapelet using function $PosCoverage$) less than a threshold or we exceed a certain number of iterations. After we extract all key shapelets, we rank them such that the first ranked shapelet has maximum positive coverage. The process is shown in Algorithm 5.

Algorithm 5: Extract Key Shapelets

Input: full-dimensional shapelet profile \hat{S}
Output: List of key shapelets

- 1 **while** $PosCoverage(S') \geq CovThreshold$ **do**
- 2 $S' = \text{Solve problem (3.22)}$
- 3 Add S' to *Solutions*
- 4 Add new constraint (3.28)
- 5 Rank *Solutions*

Up to this point we have extracted several key shapelets that cover subjects in the positive class. We can iterate this procedure (extracting a full-dimensional shapelet followed by extracting key shapelet) for additional I iterations. Having in mind the need for interpretability and selecting only key shapelets as well as the need to avoid overfitting with too complex model (*Mueen et al.*, 2011), we consider $I = 2$ iterations. We then flip the labels of the subjects and do the same process again by iteratively extracting a full-dimensional shapelet and then key shapelets for the other class. The whole IPED procedure is summarized in Algorithm 6.

Algorithm 6: IPED

Input: A time series dataset D

- 1 Transform D into binary matrix (Algorithm 3)
- 2 **for** $c = 1$ **to** C **do**
- 3 Consider c as the positive class
- 4 **for** $i = 1$ **to** I **do**
- 5 Extract full-dimensional shapelet (Algorithm 4)
- 6 Extract key shapelets (Algorithm 5)

Key Shapelets for Early Classification

After running Algorithm 6, we end up with several key shapelets for each class. For a time series with unknown label, the IPED method initially reads $minL$ (minimum length of any shapelet which is user parameter) time stamps from the test time series. First, the highest-ranked key shapelet is considered. If any of the components of the key shapelet covers the corresponding dimension of the current stream of the test multivariate time series, we mark the component of the shapelet that covers the time series. *Recall that the component of the shapelet covers a time series if the distance between the time series and the component of the shapelet is less than the distance threshold of the component.* If all components of the shapelet are marked, then the time series is classified as the class of the shapelet and the process stops. Otherwise, next key shapelet from the ranked list is considered and the process is repeated. If none of the shapelets cover the current stream of the test time series, the method reads one more time stamp and continues classifying the time series. If the method reaches the end of the time series and none of the shapelets cover it, the method marks the time series as an unclassified subject.

We note that the components of the multivariate shapelets may classify the time series at different time points. So, one component of the multivariate shapelet may cover the time series at time point t_1 and the other component may cover the time series at t_2 where $t_2 > t_1$. However, the final decision is made only when all components cover the time series, which is the last time point t_2 . Therefore, the test time series could be classified after reading a number of time points greater than the shapelet's length.

Experiments

As described before in the conclusion section of the MSD method, to learn discriminative shapelet we should have enough number of observations for the time series. Therefore, we evaluate the IPED method on the real datasets: the viral infection and PTB dataset. Later in the next chapter, we apply the method on the synthetic sepsis dataset as an application for the IPED method on the clinical data.

We compare the IPED method to 3 alternative interpretable methods. Logical Shapelet (LS) (Mueen *et al.*, 2011), Early Distinctive Shapelet for Classification (EDSC) (Xing *et al.*, 2011b), and Multivariate Shapelet Detection (MSD) (Ghalwash *et al.*, 2012). Mueen *et al.* (2011) considers combination (conjunction and disjunction) of shapelets which are called logical shapelet for more expressiveness for the time series. As noted by Mueen *et al.* (2011), *for the sake of simplicity and to guard against over fitting with too complex model, they only consider two cases, only AND, and only OR combinations.* LS is designed for classification of univariate time series and EDSC for early classification of univariate time series. Therefore, we applied both methods on two settings: 1) on each variable of the dataset separately. 2) on a new variable that is constructed by concatenating all the variables in the dataset. We report the one that gives the best result. However, LS is not designed for early classification, therefore, it utilizes the full time series of the test data. The MSD method is designed for early classification of multivariate time series so it is applicable directly to our datasets.

The coverage threshold used in Algorithm 5 to retrieve more key shapelets is set to be 50% of the training class subjects, to ensure that more training subjects are covered and to allow for variability among subjects. The *maxIter* is set to 5 to limit the computational time. For the MSD method, the parameters are optimized

using internal cross validations.

Interpretability of IPED To show the benefit of extracting a key shapelet and to explain some features of the IPED method, we show a real case from the H3N2 dataset. We used two-dimensional shapelets just for simplicity of the presentations. The experiments in the next section show that a 3-dimensional shapelet is more reasonable for our application.

In Figure 3.11, we show four genes observed over time for the same symptomatic subject. The red and blue genes are informative because we have jump in the gene expression, while the other two genes are non-informative.

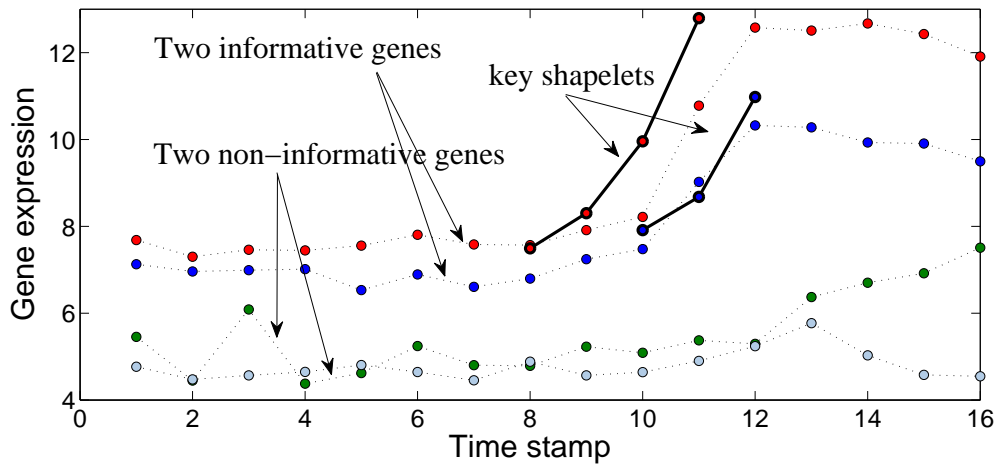


Figure 3.11: The effectiveness of the IPED method is illustrated on a single patient from H3N2. Four genes for a symptomatic test subject were observed over 16 time points. Two genes are informative for early classification (red and blue) while the other two gene are non-informative. The key shapelet covers the subject at 12th time point. We add displacement in the gene expression to make the visualization easier.

The IPED method is interpretable because the key shapelet is interpreted as “symptomatic subject is identified when we observe high increase in the red gene accompanied with high increase in the blue gene”.

We use this example to explain how IPED overcomes the issues presented

in the MSD method. First, the IPED method has extracted a key shapelet which contains only the two informative genes. This shows that the key shapelet is more accurate than the full-dimensional shapelet, as in the case of the MSD method, because it contains only the relevant dimensions. Second, each component of the key shapelet is of different length and the onset of each component is different. For instance, the 1st component of the shapelet covers the corresponding dimension of the subject time series at the 11th time point while the 2nd component covers the corresponding dimension at the 12th time point when the decision is made. These are the issues in the MSD method and we believe it affects the efficiency of MSD.

Model Complexity The parameter \mathcal{B} determines the complexity of the multivariate shapelet. We assessed the sensitivity of IPED to the parameter \mathcal{B} . For each value of \mathcal{B} , we compute the four evaluation measures. The results are shown in Figure 3.12. As expected, when increasing the number of components of the shapelet, the accuracy of the model increases but the coverage decreases and the classification decisions are provided later. Since our objective is to provide the classification as early as possible and to maintain comparable accuracy performance, we use $\mathcal{B} = 3$ in the remaining experiments for all datasets to guard against overfitting and not to use different values of \mathcal{B} for different datasets.

Evaluation of IPED We compare IPED to MSD, LS and EDSC on the 3 datasets described earlier using the four aforementioned evaluation measures. As shown in Figure 3.13, our IPED method has better performance than the alternative MSD method on the PTB dataset. In addition, IPED outperforms all other classifiers and provides the classification much earlier, using approximately half of the time series.

For the viral infection datasets, IPED has comparable results with the MSD

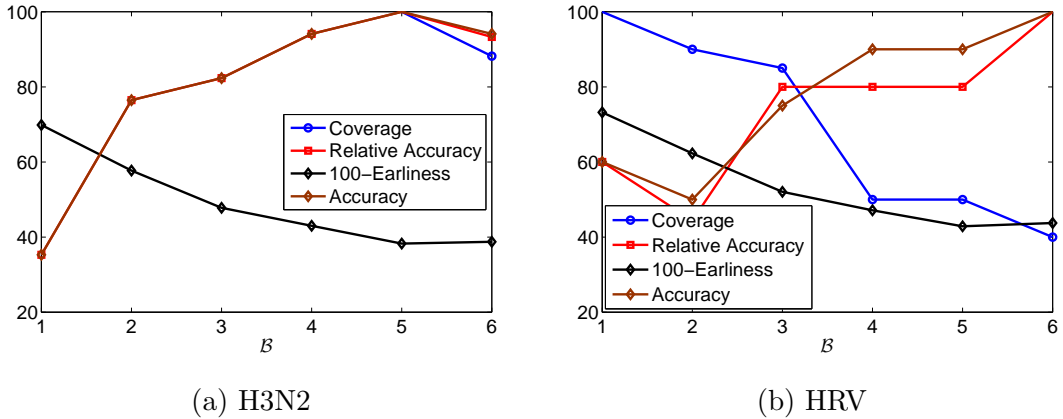


Figure 3.12: The sensitivity of the model complexity \mathcal{B} on H3N2 (left) and HRV (right) datasets.

method. For univariate classifiers, LS and EDSC, the results obtained by concatenating all variables are worse than the results obtained on one variable. It shows that LS and EDSC are not appropriate for multivariate time series. LS is the most accurate of the classifiers. However, LS provides the classification at the end of the time series while IPED provides comparable results even earlier.

Since the early classification is important in the medical domain, we have shown that on a longer time series datasets such as PTB, the classification was provided using only half of the time series, which could have significant impact by providing treatment to the patient at an earlier time. In addition, the classification accuracy of IPED was comparable to or even better than those classifiers that use the full time series.

Runtime Complexity Analysis The IPED method has three steps to extract key shapelets from the time series data. The first step is the shapelets extraction. This step is performed using exhaustive search to extract all univariate shapelets from all dimensions of the time series data. In our experiments on the PTB dataset (the largest dataset in our experiments), the shapelet extraction step took less than a minute. However, there are some recent works to speed up the process

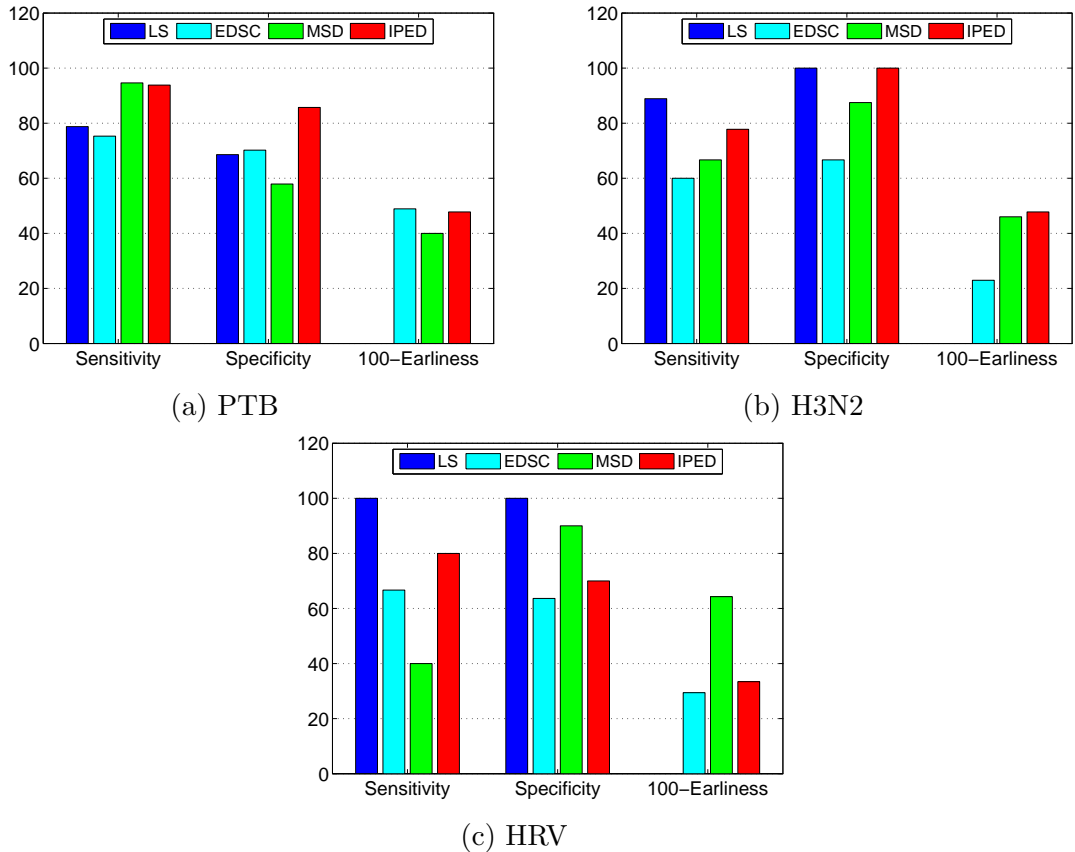


Figure 3.13: Comparison between IPED and the alternative methods on three datasets. LS and EDSC methods are applied to each variables separately and to a new variable that is constructed by concatenating all the variables, and the best result is reported. Sensitivity, specificity and the 100-earliness are reported for each method (the higher is the better). LS is the Logical Shapelet method. EDSC is the Early Ddistinctive Shapelet for early Classification of univariate time series. MSD is the Multivariate Shapelet Detection method. IPED is our proposed method. LS utilizes the full time series so that the 100-earliness for the LS method is zero.

of that step (*Rakthanmanon and Keogh, 2011; Chang et al., 2012b*). We did not use them because it is not the main point of our paper.

The second step of the IPED method is extraction of a full-dimensional shapelet by solving a convex-concave problem. This is the most time-consuming part of the method because it requires solving a convex optimization on a dataset that is constructed using all extracted univariate shapelets. On the PTB dataset, it took about 15 minutes to solve the optimization problem.

The third step is the extraction of the key shapelets from the full dimensional shapelet. This part is very fast (2-3 seconds) because the dimension of the input data to the problem is the same as the number of dimensions of the original time series (which in our case was 15-26). Therefore, the most time is consumed to apply the second step of the method. However, this is only for training the model, which is done offline.

Using the extracted key shapelets for early classification (which is done online), is a very fast operation, because the number of extracted shapelets is small. In our case, the early classification of the test time series for each patient takes less than a second to classify the patient.

Evaluation of the Second Optimization To evaluate the importance of our mixed integer optimization formalization, we conducted two experiments. First, we removed the second optimization completely and kept only the first optimization, which we call “*None*”. In the second experiment, we replaced the mixed integer optimization with the penalized (elastic net) logistic loss as the objective function, which we call “*EN*”. The results are shown in Table 3.7. Using the mixed integer programming as the second optimization in the IPED method outperformed all other alternatives because *EN* and *None* have very low coverage which affects the overall accuracy.

Conclusion

We proposed an optimization-based method for building predictive models on multivariate time series data and mining relevant temporal interpretable patterns for early classification (IPED). The IPED method starts with transforming the multivariate time series data into a binary matrix representation over the span of all extracted shapelets from all the dimensions of the time series. Then, the

matrix is used to build predictive models via solving the proposed optimization formulations. The IPED method extracts a full-dimensional shapelet for each class from the binary matrix by solving a convex-concave optimization problem. Then, IPED extracts key shapelets for each class by solving a mixed integer optimization problem. The extracted key shapelets are low-dimensional shapelets. These key shapelets are then used for early interpretable classification. We are able to make our results interpretable by using only a few patterns from the observed time series data.

Our IPED method addresses three issues in the state-of-the-art MSD method. First, the components of the multivariate shapelet do not have the constraints of the same starting time point and are not required to be of the same length. Therefore, the onset of each pattern could be different, which simulates a real life scenario. Second, the extracted multivariate shapelet contains only the relevant dimensions of the observed multivariate time series data.

We showed that the IPED method provides accurate and interpretable early classification. However, it is important in medical applications to provide a measure that allows the physicians to judge about the belief in the prediction. In the next chapter, we propose a method to provide uncertainty estimate for the shapelet-based early classification model like EDSC, MSD, and IPED.

Table 3.3: Evaluation of MSD, EDSC and ECM on viral infection and drug response datasets.

Dataset	Method	Acc	Rel Acc	Cov	Ear	F_1
H3N2	EDSC	77.78	85.71	100	38.34	0.69
	MSD	77.78	85.71	100	62.50	0.69
	ECM	40.68	85.71	100	23.50	0.53
HRV	EDSC	42.86	80.00	55.56	52.50	0.45
	MSD	70.00	71.43	100	40.00	0.65
	ECM	31.24	85.71	100	25.50	0.44
Baranzini3A	EDSC	12.00	100.00	12.25	42.86	0.20
	MSD	70.00	73.91	95.83	46.26	0.61
	ECM	86.00	86.00	100	44.00	0.68
Baranzini3B	EDSC	26.09	80.00	31.38	40.26	0.36
	MSD	66.67	68.00	100	44.81	0.60
	ECM	89.00	89.00	100	46.00	0.67
Baranzini6	EDSC	12.00	100.00	12.25	42.86	0.20
	MSD	70.83	70.83	100	42.86	0.63
	ECM	59.00	66.00	91.00	49.00	0.55
Baranzini12	EDSC	26.09	80.00	31.38	40.26	0.36
	MSD	66.67	66.67	100	42.86	0.62
	ECM	85.00	85.00	100	43.00	0.68
Lin9	EDSC	26.09	80.00	31.38	40.26	0.36
	MSD	67.86	69.57	100	44.00	0.61
	ECM	85.00	85.00	100	44.50	0.67
Costa17	EDSC	26.09	80.00	31.38	40.26	0.36
	MSD	68.00	69.23	100	45.24	0.61
	ECM	80.00	80.00	100	44.00	0.66

Table 3.4: Comparison of different threshold learning methods for MSD.

Dataset	Method	Acc	Rel Acc	Cov	Ear	<i>P</i> -value
H3N2	Chebyshev	66.67	85.71	87.50	58.33	6.3e-062
	Information Gain	77.78	85.71	100	62.50	
HRV	Chebyshev	55.56	100	57.14	60	4.9e-063
	Information Gain	70.00	71.43	100	40.00	
Baranzini3A	Chebyshev	65.38	82.61	80.95	55.10	2.7e-052
	Information Gain	70.00	73.91	95.83	46.26	
Baranzini3B	Chebyshev	62.96	76.92	84.81	53.90	1.4e-030
	Information Gain	66.67	68.00	100	44.81	
Baranzini6	Chebyshev	65.38	70.83	100.00	47.62	4.9e-052
	Information Gain	70.83	70.83	100	42.86	
Baranzini12	Chebyshev	66.67	69.23	100.00	45.58	0.036
	Information Gain	66.67	66.67	100	42.86	
Lin9	Chebyshev	68.18	77.27	91.67	50.55	3.2e-005
	Information Gain	67.86	69.57	100	44.00	
Costa17	Chebyshev	70.00	77.27	95.45	48.98	0.017
	Information Gain	68.00	69.23	100	45.24	

Table 3.5: Comparison of different utility methods for MSD. Weighted F_1 score and weighted information gain.

Dataset	Method	Acc	Rel Acc	Cov	Ear	P -value
H3N2	Weighted F_1 score	68.26	71.43	100	42.71	7.2e-104
	Weighted IG	77.78	85.71	100	62.50	
HRV	Weighted F_1 score	66.67	70.00	100	47.89	4.7e-163
	Weighted IG	70.00	71.43	100	40.00	
Baranzini3A	Weighted F_1 score	66.67	68.00	100	43.94	3.9e-085
	Weighted IG	70.00	73.91	95.83	46.26	
Baranzini3B	Weighted F_1 score	65.22	65.38	100	43.51	3.8e-071
	Weighted IG	66.67	68.00	100	44.81	
Baranzini6	Weighted F_1 score	72.00	72.00	100	42.86	9.6e-145
	Weighted IG	70.83	70.83	100	42.86	
Baranzini12	Weighted F_1 score	65.22	68.00	100	44.72	1.6e-093
	Weighted IG	66.67	66.67	100	42.86	
Lin9	Weighted F_1 score	65.38	66.67	100	43.51	0
	Weighted IG	67.86	69.57	100	44.00	
Costa17	Weighted F_1 score	67.00	68.00	100	43.51	1.6e-037
	Weighted IG	68.00	69.23	100	45.24	

Table 3.6: The run time of the MSD method for all datasets. The number of genes, number of examples, the time series length, and the run time in seconds are reported in the table.

Dataset	# genes	# examples	TS length	Time in seconds
H3N2	23	17	16	295.1
HRV	26	20	10	77.7
Baranzini3A	3	52	7	49.3
Baranzini3B	3	52	7	36.1
Baranzini6	6	52	7	41.1
Baranzini12	12	52	7	64.3
Lin9	9	52	7	48.8
Costa17	17	52	7	131.9

Table 3.7: Evaluation of different optimizations to extract the key shapelets. *EN* is the elastic net approach. *None* is the method without the second optimization. Cov is the coverage, Acc is the accuracy, and Ear is the earliness.

		Coverage	Accuracy	100-Earliness
H3N2	IPED	100	82.35	47.8
	EN	21.2	80.0	5.4
	None	17.6	60	4.5
HRV	IPED	85.0	80.6	47.2
	EN	19.0	54.3	9.5
	None	1	10	0.4
PTB	IPED	100	89.78	47.43
	EN	25	70.2	60.2
	None	1.17	90.2	4.2

CHAPTER 4

UNCERTAINTY ESTIMATE FOR SHAPELET-BASED EARLY CLASSIFICATION MODELS

In time-sensitive applications such as medical applications, some certain aspects have to be considered. Accurate and timely diagnosis is essential to allow physicians to design appropriate therapeutic strategies at early stages of diseases. In addition to attaining accurate and early classification, it is important to attain decisions that can be easily interpreted. The IPED method addresses all these issues by extracting interpretable shapelets for accurate early classification. However, in medical applications, it is crucial to accompany the predictions by a measure which allows the physicians to judge about the uncertainty or belief in the prediction. Knowing the uncertainty associated with a given prediction is especially important in clinical diagnosis where data mining methods assist clinical experts in making decisions and optimizing therapy.

We propose a method that provides uncertainty estimate for shapelet-based early classification method (*Ghalwash et al.*, in reviewb). Since, the uncertainty estimate method is based on the shapelet, therefore, it could be easily integrated with any shapelet-based method like EDSC, MSD and IPED. The uncertainty estimation should meet some requirements. In temporal data, by having more observations the uncertainty of the prediction would be reduced over time and the confidence of the prediction would increase. In addition, the uncertainty measure takes values on $[0,1]$ range where 1 gives the highest uncertainty estimate.

4.1 Uncertainty Estimation

We start by proposing a method to provide an uncertainty estimate to the interpretable early classification EDSC method since it is a univariate method and it is easy to explain it (*Ghalwash et al.*, in reviewb). Then, we show how to easily extend the method to multivariate shapelet-based method.

Instead of modeling the uncertainty of classifying a time series as class c , we model the confidence estimate and then the uncertainty is computed directly from the confidence estimate. Therefore, we model the confidence $ClassCon(c)$ of classifying a time series as class c and based on that the uncertainty is computed as

$$Uncert(c) = 1 - ClassCon(c) \quad (4.1)$$

Let us assume that we have a univariate shapelet $S = (s, l, \delta, c)$ (Definition 3.1.1) and a univariate time series T . If the distance d between the time series and the shapelet (Definition 1.4.5) is less than or equal to δ , then T is classified as the class of the shapelet. In this scenario, we did not measure how confident the EDSC method is about the classification of T .

Imagine that the shapelet S is represented as a point, as in Figure 4.1(a). The radius of the circle around the shapelet represents the shapelet's distance threshold δ . Assume that we have two time series T_1 and T_2 . If the distance between the shapelet and both time series T_1 and T_2 are less than the shapelet's distance threshold δ then T_1 and T_2 are represented as points inside the circle. Therefore, both time series are classified to be of the same class as the shapelet's class. However, the distance between T_1 and S is less than the distance between T_2 and S which reflects the fact that the shapelet is more certain about the classification of T_1 than the classification of T_2 . In Figure 4.1(a) different red dotted circles represent different levels of confidence. Intuitively, the shapelet is more

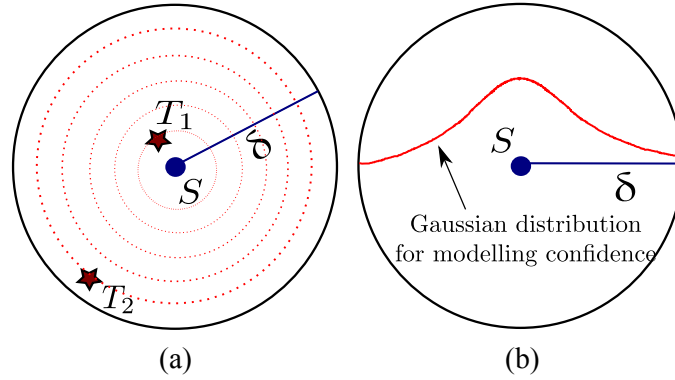


Figure 4.1: Modeling confidence. (a) The shapelet is represented as point and the radius of the circle around S represents the distance threshold δ . T_1 and T_2 are two time series segments with distances less than δ to the shapelets, therefore, they are predicted to have the same class as the shapelet's class. However, S is more certain about the prediction of T_1 than the prediction of T_2 . (b) The confidence of the shapelet is modelled as Gaussian where the shapelet is more certain about closer time series than time series that are δ -distance apart.

certain about the time series in the most inner red circle than time series in the bigger red circles. The uncertainty reaches the highest level when the time series lies on the boundary of the black circle.

4.1.1 Confidence Estimate

We model the confidence of the shapelet S as Gaussian distribution such that the confidence takes the highest value at the center of the circle (center of the distribution) and lower value at the boundary of the circle (time series with distance δ) as shown in Figure 4.1(b).

Technically, the confidence is modeled as a Gaussian function

$$G_S(d) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{(d-\mu)^2}{2\sigma^2}} \quad (4.2)$$

where S is the shapelet and d is the distance between the shapelet and the time series. The mean of the Gaussian is zero as in Figure 4.1(b) and the standard

deviation σ is chosen such that 95% of the matched distances are inside the circle. Therefore, we assign $\mu + 2\sigma = \delta \implies \sigma = \delta/2$.

When a time series matches the shapelet, i.e. $dist(S, T) = d \leq \delta$, then the confidence of the shapelet S , which referred to as $UnitCon_S$, is:

$$UnitCon_S(d) = \text{erfc}\left(\frac{d}{\sigma\sqrt{2}}\right) \quad (4.3)$$

where `erfc` is the complementary error function (*Andrews, 1998*) and it is used to convert the confidence into a probability.

Note that, each shapelet has its own threshold and hence has its own confidence distribution. Therefore, for any time series, we compute the distance d between the time series and the shapelet. If the distance is less than or equal to the threshold, the confidence is computed using Equation 4.3. If the distance is greater than the threshold, the confidence is not computed because the time series lies outside the region (circle) of the shapelet. Thus, the confidence is computed only when the shapelet matches the time series.

Let us assume that we have two shapelets S_1 and S_2 (from different classes) that match the time series T . Suppose that both shapelets give the same confidence level about their classification. Which one should we trust? Intuitively, we should trust the shapelet that performs well in the training data. Therefore, we need the confidence measure to incorporate the accuracy performance of the shapelet in the training data, having in mind that the final confidence measure should be in the range $[0, 1]$. Typically, the confidence is 1 only when the distance between the shapelet and the time series is zero, and the shapelet discriminates all training target time series from the non-target time series.

To incorporate the performance of the shapelet in the confidence measure, we compute the accuracy performance (F_1 score) of the shapelet in the training

data. To compute the F_1 score, we need to compute the precision and the recall of the shapelet $S = (s, l, \delta, c)$ as follows:

$$Precision(S) = \frac{\|\{d_i \leq \delta \wedge Class(T_i) = c\}\|}{\|\{d_i \leq \delta\}\|}$$

$$Recall(S) = \frac{\|\{d_i \leq \delta \wedge Class(T_i) = c\}\|}{\|\{Class(T_i) = c\}\|}$$

where d_i is the distance between the time series T_i and the shapelet S . The F_1 score of the shapelet is defined as

$$F_1(S) = \frac{2 \times Precision(S) \times Recall(S)}{Precision(S) + Recall(S)} \quad (4.4)$$

Then, the confidence is measured as:

$$Con_S(d) = UnitCon_S(d) * F_1(S) \quad (4.5)$$

Since F_1 takes value between 0 and 1, the highest value of the Con measure is 1 which satisfies the property of the confidence measure. For simplicity, when it is clear from the context, we write $Con(S)$ and remove d to refer to Equation 4.5. Equation 4.5 incorporates two measures: how far the time series from the shapelet and the performance of the shapelet in the training data.

Aggregated Class Confidence

If there is only one shapelet from the class c matches the time series, then the confidence estimate for predicting the time series as class c is the same as the shapelet's confidence and computed using Equation 4.5. Now, we consider the case of computing the class confidence when more than one shapelet match the time series. We need to find a way to aggregate confidences from multiple

shapelets to get one final confidence measure for each class. Intuitively, the class confidence should be greater than the confidence of any of the individual shapelet's confidence. Let us start with a simple case and then we generalize it.

Simple Case: Assume we have two shapelets S_1 and S_2 from class c that match the time series, then the confidence of classifying the time series as class c is computed as:

$$\begin{aligned}
 \text{ClassCon}(c) &= \text{Con}(S_1 \cup S_2) \\
 &= \text{Con}(S_1) + \text{Con}(S_2) - \text{Con}(S_1 \cap S_2) \\
 &= \text{Con}(S_1) + \text{Con}(S_2) - \text{Con}(S_1) * \text{Con}(S_2) \quad (4.6)
 \end{aligned}$$

where the assumption of the shapelets' independence is considered. The value of the class confidence ClassCon is greater than the confidence of any of the individual shapelets S_1 and S_2 which does make sense because our confidence for classifying the time series as class c is increased by having two matched shapelets for the class c .

General Case: Let us assume that $S^c = \{S_1, S_2, \dots, S_N\}$ be the set of all shapelets from class c that match the current time series. Then the classification confidence of the time series for class c is computed as:

$$\begin{aligned}
 \text{ClassCon}(c) &= \text{Con}(S^c) \\
 &= \text{Con}(S_1 \cup S_2 \cup \dots \cup S_N) \\
 &= \sum_{k=1}^N (-1)^{k+1} \sum_{\substack{I \subset \{1, 2, \dots, N\} \\ |I|=k}} \text{Con}(S_I) \quad (4.7)
 \end{aligned}$$

where the last sum runs over all subsets I of the indices $\{1, \dots, N\}$ which contain exactly k elements, and $S_I = \cap_{i \in I} S_i$.

Now, the class confidence $ClassCon$ satisfies all the properties of the confidence measure, i.e. takes values on the range $[0, 1]$ and has value higher than any individual shapelet.

Multivariate Shapelet: The above definition of the class confidence (Equation 4.7) is directly applicable to multivariate shapelets. Let us assume that we have two matched multivariate shapelets $\mathbf{f}_1 = ([s_1, s_2, s_3], l_1, \Delta_1, c)$ and $\mathbf{f}_2 = ([s_1, s_4, s_3], l_2, \Delta_2, c)$ where both multivariate shapelets have two common components s_1 and s_3 . Then, the 4 different matched components are s_1, s_2, s_3, s_4 . The class confidence of the time series based on those two matched multivariate shapelets is computed as

$$ClassCon(c) = Con(s_1 \cup s_2 \cup s_3 \cup s_4)$$

in which we apply Equation 4.7 directly.

Propagation of Uncertainty Over Time

Intuitively, the uncertainty estimate for classifying a time series would decrease as the time evolves and more information about the time series becomes available to the method. The uncertainty measure defined in Equation 4.1 satisfies this property.

Assume that at time t there are k shapelets match the time series and at time $t+1$ two more shapelets match the time series, then $k+2$ shapelets match the time series up to time $t+1$. Since the k shapelets are subset of the $k+2$ shapelets then the confidence of the $k+2$ shapelets has to be greater than the confidence

of the k shapelets (as in Equations 4.7), which means that the confidence at time $t + 1$ is greater than the confidence at time t . In other words, the uncertainty propagates over time and decreases as time evolves.

4.1.2 Uncertainty for Early Classification

To compute a reliable uncertainty estimate for the time series classification, we need to have more discriminative shapelets used for early classification. Therefore, we modify the EDSC method to obtain more discriminative shapelets. In particular, the pruning method described in Section 3.1 is modified.

The EDSC method sorts the shapelets descending based on their utility scores. Then, it starts with the highest ranked shapelet S and removes all time series from the dataset that are matched by the shapelet. Then, EDSC stores the shapelet S and *all other shapelets that have the same utility score as S (equally-performance shapelets)*. Then, the next highest shapelet is considered. If the shapelet covers any of the remaining time series, the shapelet and all other equally-performance shapelets will be added to the extracted list and all covered time series will be removed. The method iteratively does so until all time series in the dataset are covered. In this manner, the EDSC method ends up with a list of equally-performance shapelets that is used for the classification purposes. The list of the shapelets extracted from the modified EDSC is longer than the list extracted from the original EDSC. However, the longer list does not contain a lower-performance shapelet than the shorter list.

The classification process is also little bit different than the classification process described in Section 3.1. When we have time series with unknown label, we compute the distance between the current stream of the time series and *all* discriminative shapelets extracted by EDSC (we do not start with the highest one

until a match is found). Then we compute the uncertainty for each class based on *all matched* shapelets from that class (*Recall that the shapelet matches the time series if the distance between the shapelet and the time series is less than the distance threshold*). If we do not have any matched shapelet for a class then we do not have uncertainty associated with that class.

At each time point we compute the uncertainty for each class. The time series is classified based on the class that has minimum uncertainty. If the produced uncertainty is not satisfactory for the user, the method continues classifying the time series until a user defined level of uncertainty is obtained.

4.2 Experiments

We evaluated our method on three univariate ECG datasets from the UCR time series archive (*Keogh et al., 2011*). In all experiments we used the same parameters as in (*Xing et al., 2011b*).

Table 4.1: ECG datasets description. The number of patients and time points for each dataset. The numbers between parenthesis in the “No. Patients” column represent the number of training and test data, respectively.

Dataset	No. Patients	No. Time Points
ECG200	200(100/100)	96
TwoLeadECG	1162(23/1139)	82
ECGFiveDays	884(23/861)	136

4.2.1 Modified EDSC versus Original EDSC

We have modified the EDSC method to allow for more equally-performance shapelet to be included in the final list used for the early classification task. The

reason behind that is two folds: First, since the additionally included shapelet have the same accuracy and earliness performance on the training data, then it would be better to include them to allow for variability among patients patterns. Second, allowing more shapelet would result in reliable uncertainty estimate.

We first evaluate the effect of including more shapelets on the accuracy performance of the EDSC method without estimating the uncertainty. As shown in Table 4.2, the modified EDSC has better accuracy than the original EDSC and has slightly better earliness which validates our hypothesis that including more shapelets would be beneficial for allowing variability among patients.

Table 4.2: Evaluation of the modified EDSC on ECG datasets.

Measure	ECG200		TwoLeadECG		ECGFiveDays	
	Original	Modified	Original	Modified	Original	Modified
Coverage	90	92	85.5	96.7	95.1	96.4
Relative Accuracy	86.7	85.9	98.7	98.8	75.3	75.1
Accuracy	78	79	84.4	95.5	71.7	72.4
Earliness	17.6	17.2	39.8	39.2	48.4	48.4
# Shapelets	32	118	4	13	5	10

4.2.2 Case Study

We show the effectiveness of our uncertainty method on a real example from the TwoLeadECG dataset. In particular, we show how the method provides *more confident class prediction by having more shapelets* as in Equation 4.6.

The TwoLeadECG dataset has two classes, for simplicity, we call them the red and the blue classes. EDSC has extracted 13 shapelets, two from each class. A time series from the blue class is shown in Figure 4.2. The time series matches the first (blue) shapelet at time 34 with uncertainty 0.95 which means that the

uncertainty associated for the blue class is 0.95 and no uncertainty associated for the red class because no shapelet matches the time series up to that point.

Then, at the time point 35, the time series matches a red shapelet with uncertainty 0.94. Therefore, at this point the method classifies the time series as a blue class with uncertainty 0.95 and as a red class with uncertainty 0.94. The method has approximately similar uncertainty for each class and therefore it is not certain about any class. These uncertainties propagate until time point 45 where a new (blue) shapelet matches the time series. The uncertainties from the two matched blue shapelets are aggregated using Equation 4.7 to give uncertainty 0.84 which reveals the fact that the method is now more certain to classify the time series as blue class (correct classification).

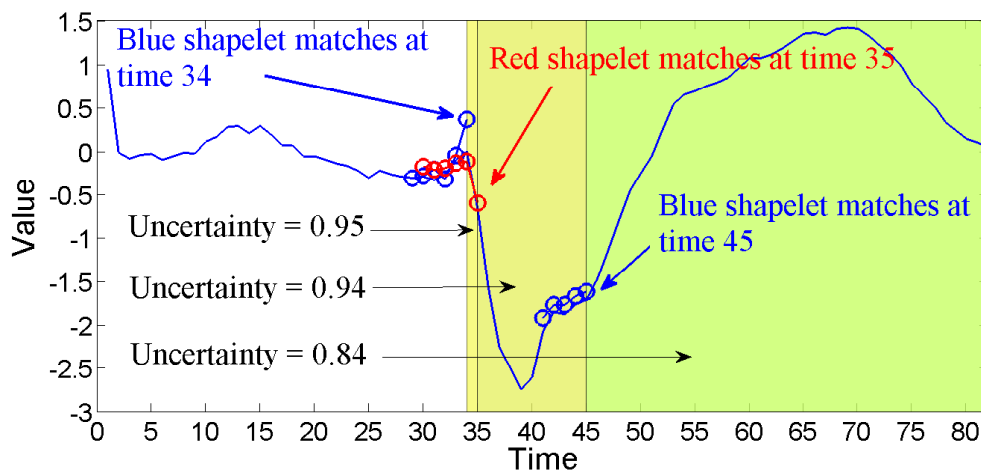


Figure 4.2: Aggregation of uncertainties from multiple shapelets for a time series from the TwoLeadECG dataset. A time series from the blue class is classified at the time point 34 with uncertainty 0.95. Then, a red shapelet matches the time series at the time point 35 with uncertainty 0.94. Therefore, the prediction is undecided due to similar uncertainties for red and blue classes. At time point 45, a new blue shapelet matches the time series. The uncertainties from the two blue shapelets are aggregated using Equation 4.7 resulting in uncertainty 0.84 for the blue class which is the correct class.

4.2.3 Uncertainty Propagation

In these set of experiments, we show how the uncertainty decreases over time for all patients. We use the ECG200 dataset as an example as shown in Figure 4.3a. EDSC classifies many patients using only the first 10 time points but with high uncertainty. These uncertainties decreases over time. For instance, as shown in Figure 4.3b, the patient 72 is classified at the time point 7 with high uncertainty 0.98 and this uncertainty reduced eventually until it reached the value 0.43 at the time point 12.

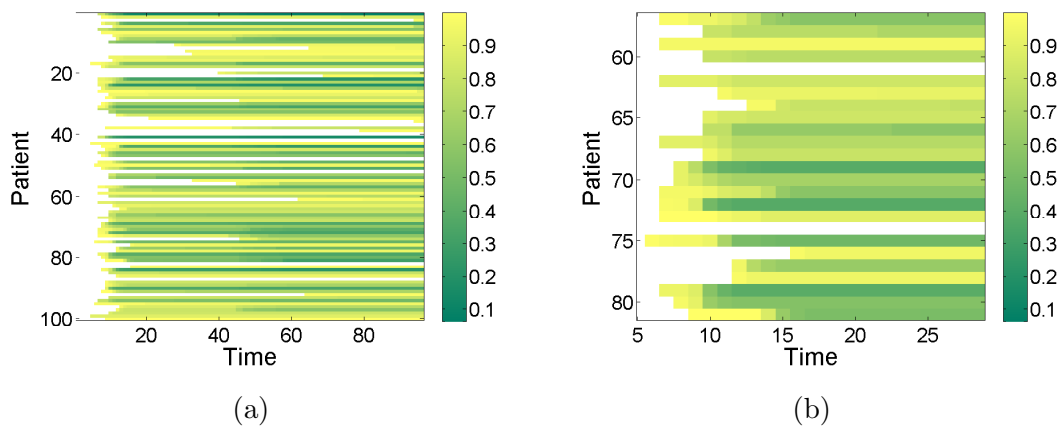


Figure 4.3: The left panel (a) shows the values of uncertainty over time for each patient from the ECG200 dataset. The white bar indicates that there is no classification at that point and hence there is no uncertainty. As shown in panel (a), the uncertainty for each patient reduces over time. The right panel (b) is the zoom-in version of panel (a), e.g. the method is highly uncertain (0.98) about the classification of the patient 72 at time 7. As time evolves, the method becomes more and more certain about the classification where the method classifies the patient with uncertainty 0.43 at time 12.

The average of uncertainties over all patients at each time point is computed for the three evaluation datasets. As shown in Figure 4.4, the methods starts with high uncertainty and then the uncertainty decreases over time until reaches the end of the time series.

Two aspects have to be pointed out for Figure 4.4. First, the average uncertainty over all patients sometimes decreases and then increases little bit. For

example, on ECG200 dataset, the uncertainty decreases up to the 60th time point and then increases up to the 70th time point. The reason is that the method covers more patients from time point 60 to time point 70. Therefore, the uncertainty for those new covered patients will increase the average uncertainty at these time points. However, as shown in Figure 4.3, the uncertainty for a specific patient does not increase. Second, the average uncertainty over all patients are relatively high. The reason is that only few shapelets match the time series which does not make the uncertainty decreases significantly. The illustration in Figure 4.2 shows that only two blue shapelets match the time series and therefore the final uncertainty in this case is still fairly high.

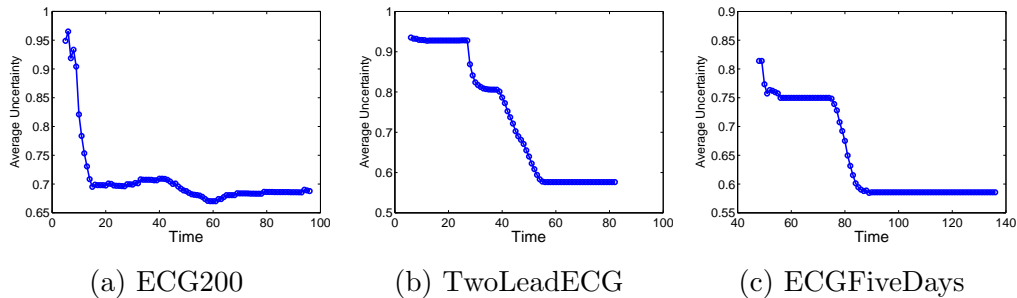


Figure 4.4: The average uncertainty over all patients at each time point for ECG200, TwoLeadECG and ECGFiveDay datasets. In all experiments the method starts with high uncertainty and then becomes certain about the classification as time evolves.

Since the uncertainty decreases over time, it brings the question of how the uncertainty is related to the classification accuracy and earliness. In the following section we show the effect of the uncertainty on the modified EDSC method’s accuracy and earliness.

4.2.4 Uncertainty versus evaluation measures

We study the effect of the uncertainty on accuracy, coverage, and earliness. Intuitively, when requiring low uncertainty we expect the method to become more

accurate but the classification will be delayed. In addition, lower uncertainty may affect the coverage of the method so the method may not be able to classify many patients even after reaching the end of the time series.

We show the effect of the uncertainty on the predefined three performance measures of the modified EDSC method. Namely, Coverage (Definition 2.2.2), Earliness (Definition 2.2.3), and Relative Accuracy (Definition 2.2.4).

As shown in Figure 4.5, lower values of uncertainty give more accurate results than higher values of uncertainty. However, the earliness and the coverage of the modified EDSC method are affected. In other words, the method covers small number of patients when requiring low uncertainty and the classification for those classified patients are provided later, i.e. the majority of the patients are classified with relatively high uncertainty (as in Figure 4.5). Therefore, there is a trade off between uncertainty from one side and accuracy, coverage, and earliness from the other side. This trade off varies from one dataset to another and from domain to another.

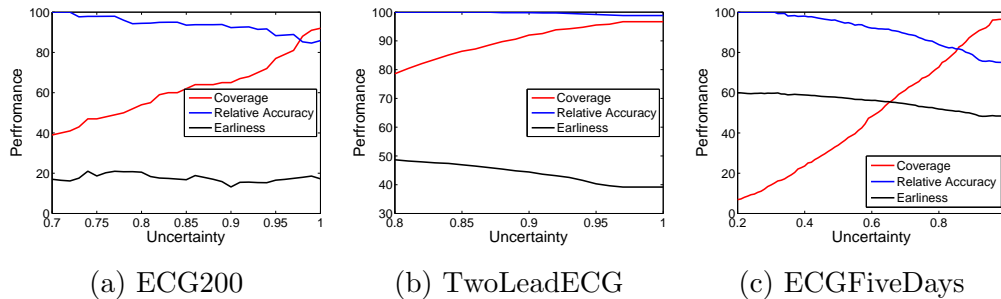


Figure 4.5: The effect of the uncertainty on coverage, earliness and accuracy measures evaluated on three dataset. Lower uncertainty results in more accurate results but reduces the coverage of the method and makes the modified EDSC method delays the classification.

Table 4.3: Categorization of time series classification methods. Four properties are used to categorize different methods: interpretability (Inter), earliness (Ear), uncertainty estimate (Uncer), and multivariate time series (Multi)

Reference	Method	Inter	Ear	Uncer	Multi
(<i>Lin et al.</i> , 2008)	HMM	X	X	X	✓
(<i>Borgwardt et al.</i> , 2006)	LDS/SVM	X	X	X	✓
(<i>Xing et al.</i> , 2009, 2011a)	ECTS	X	✓	X	X
(<i>Wiens et al.</i> , 2012)	-	X	✓	✓	✓
(<i>Anderson et al.</i> , 2012)	QDA	X	✓	✓	✓
(<i>Ye and Keogh</i> , 2009)	Shapelet	✓	X	X	X
(<i>Xing et al.</i> , 2011b)	EDSC	✓	✓	X	X
(<i>Ghalwash and Obradovic</i> , 2012)	MSD	✓	✓	X	✓
(<i>Ghalwash et al.</i> , in reviewa)	IPED	✓	✓	X	✓
(<i>Ghalwash et al.</i> , 2012, 2013b)	ECM	X	✓	X	✓

4.3 Related Work

Since we have developed a method that provides an uncertainty estimates for the shapelet-based early classification method, we discuss relationship of our method to the existing time series classification methods that address some of earliness, interpretability, and uncertainty estimates aspects. To the best of our knowledge, existing methods are limited to addressing at most two of these three aspects (earliness, interpretability, uncertainty estimates). Table 4.3 summarizes properties of several methods that successfully addressed some of these aspects. However, there is no yet a time series classification method that provides simultaneously these three aspects.

Adapting time series classification methods for early classification.

Several time series classification methods are *adapted* for early classification by applying the method at each time point. The methods are inflexible as the

method trained on time series of length t , the prediction is always done at the t^{th} time point. A method that utilizes hidden Markov models (HMM) with less states than the time points is proposed (*Lin et al.*, 2008). The model is evaluated at each time point. They showed that the best results were obtained when using all time points. A linear dynamical system (LDS) and support vector machines (SVM) are used to model time series for gene expression (*Borgwardt et al.*, 2006). The method is applied at each time point and the best results were obtained when using most of the time points. These results provide evidence that there is need for methods designed specifically for early classification.

Interpretable time series classification.

The shapelets, which are subsequences of time series, has been used as features representing the characteristic of training time series (*Ye and Keogh*, 2009). Ideally, a perfect shapelet of a class is the one representing all time series in the class but does not cover any time series in other classes. These shapelets are used for interpretable time series classification. However, the method does not provide neither uncertainty estimate nor early classification which are addressed by our proposed method.

Early classification.

The problem of early classification of time series is formulated recently (*Xing et al.*, 2009, 2011a). They introduced a novel concept of minimum prediction length and developed an early classification on time series (ECTS) method. ECTS makes early predictions and at the same time retains the accuracy comparable with that of a 1-nearest neighbor classifier using the entire time series. The disadvantage of ECTS is that it does not extract and summarize patterns from training data; thus, users may not be able to gain deep insights from the classification results.

Interpretable early classification.

A method, called early distinctive shapelet classification (EDSC), is proposed for early classification (*Xing et al.*, 2011b). The EDSC method utilizes local shapelets, which are segments of time series remaining in the same space of the input data and thus are highly interpretable. The local shapelets are extracted to distinctly manifesting a target class locally and early so that they are effective for early classification. Experiments showed that the extracted local shapelets are highly interpretable and can achieve effective early classification. However, the EDSC method does not provide reliability (uncertainty) estimate for the prediction task. This limitation is addressed by our method proposed in this article.

Early classification with uncertainty estimate.

An approach for early classification of signals using a generative classifier was developed recently (*Anderson et al.*, 2012). The model is based on the quadratic discriminant analysis (QDA) classifier and provides a reliability bound on the classifier's decision for every time point. Experiments showed that the approach is both early and reliable, and provides the user with a parameter to choose the trade-off between reliability and earliness. A patient risk is represented as a time series and the uncertainty is estimated as the distance between the evolving approximate daily risk of a patient and the hyperplane learned by the SVM (*Wiens et al.*, 2012). The model has been shown to be accurate on medical dataset. The disadvantage of these methods is that they are not interpretable (“black-box”) for the domain experts.

4.4 Conclusion

We proposed a method that provides uncertainty estimate for the shapelet-based early classification method. The uncertainty estimate provides a measure to judge about the prediction of the method. We extended the interpretable early classification method (EDSC) and proposed a method to measure the uncertainty with the classification. The proposed uncertainty measure takes values in the range $[0,1]$ and propagates over time. We evaluated our method on three medical datasets and showed how the uncertainty decreases over time by having more shapelets match the time series which makes the method provides more certain classification results.

In the next chapter we use the uncertainty estimate for the IPED method as an application for blood purification treatment for septic patients.

CHAPTER 5

APPLICATION - SEPSIS THERAPY OPTIMIZATION

Sepsis, a medical condition characterized by uncontrolled inflammatory response due to infection, is one of the main causes of deaths in the intensive care units, with over 750,000 cases annually in the United States alone (*Zuev et al.*, 2006). One of the main reasons for such a high number of death cases lies in limited understanding and knowledge about the complex inflammatory response mechanism, which has led to only a few effective sepsis therapies. The single approved anti-sepsis drug therapy was withdrawn from global markets in fall 2011 following the failure of its worldwide trial to demonstrate improved patient outcome (*Angus*, 2011). In the absence of adequate therapy, the patient is treated with standard broad-spectrum antibiotics and/or intravenous fluids with dosages adjusted manually. Inadequate treatment and the fact that sepsis is often diagnosed too late result in a mortality rate of 30-35%, and for every hour that the administration of appropriate treatment is delayed, the mortality rate increases by about 7% (*Thiel et al.*, 2010).

Blood purification has recently been proposed as a potentially beneficial therapy for septic patients (*Rimmele and Kellum*, 2011). This therapy is based on the dialysis-like principle, where the blood is purified by a device attached to patient (Figure 5.1). The goal of purification is to remove harmful particles from patient blood, leading the patient to a healthy state. Preliminary studies on animal models indicate the success of blood purification techniques in sepsis treatment

(*Song et al.*, 2012). In conducted animal studies clinicians had full control of the exact time of sepsis induction. Based on that information and theoretical assumptions about sepsis progress, clinicians determined start and duration of therapy. For example, in (*Song et al.*, 2012) the duration of therapy was fixed to 4 hours while the onset of therapy was set at 18 hours after sepsis induction (the clinicians' assumption was that after the 18th hour sepsis would be fully manifested). The onset of therapy was the same for all animal models involved in the experiment, although more personalized approach to sepsis diagnosis would be far more effective. Also, fixed onset of therapy with respect to sepsis induction does not reflect real clinical practice where clinicians have limited knowledge of sepsis stage at the moment when a patient shows up in an emergency room. Despite its importance, no work has been done to provide correct and timely sepsis diagnosis in conjunction with optimal blood purification therapy.

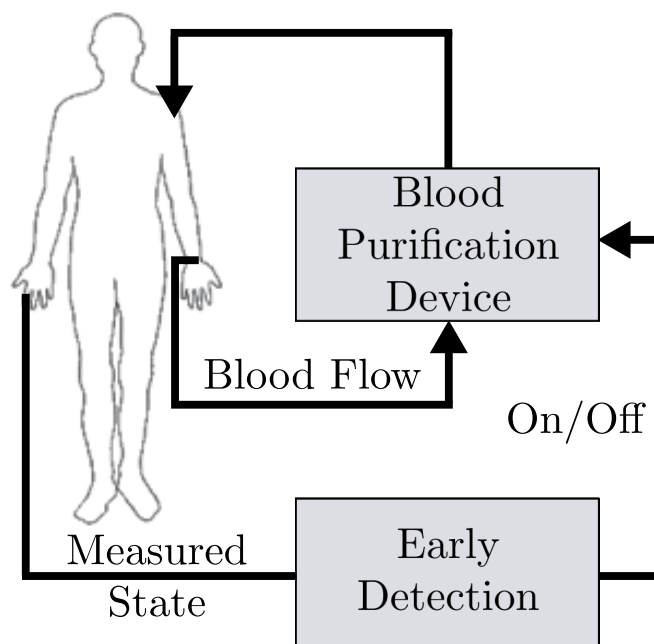


Figure 5.1: Schematic diagram of dialysis-like blood purification device accompanied with early detection module.

The time needed for sepsis detection can be significantly reduced by us-

ing the information from multivariate time series data collected from the patient. Therefore, we use the IPED method (*Ghalwash et al.*, in reviewa) that extracts multivariate shapelet from the data and use them for identification of septic patients from healthy patients. The IPED classification method is accompanied with uncertainty estimate proposed in Chapter 4. Usually, the practitioner do not release a patient from the intensive care unit unless the patient is confirmed to some extent to be healthy. Following this principle, we use lower uncertainty to identify healthy patients and high uncertainty to identify septic patients. Lower values for the uncertainty measure indicates that the IPED method will not classify the patient as healthy unless the method is confident about the classification. Contrary, the IPED method uses high uncertainty for classify septic patients which indicates that if there is any doubt that patient is going to develop sepsis in the near future then the IPED method immediately classifies the patient as septic patient and the physician starts to treat the patient (*Ghalwash et al.*, 2013a).

We follow the steps in Section 1.3.4 to generate 10,000 virtual patients who received no treatment. A group of 5,000 patients belong to the survival population and the remaining 5,000 patients belong to the non-survival population. We sampled randomly 30 patients from the 10,000 patients for training the IPED method and used the remaining 9,970 patients for testing. We chosen 30 patients to simulate the real-life where small number of patients is provided for training. We repeated that process, sampling 30 patients for training and applying the trained model on the 9,970 patients, three times having three values for each reported measure. The average and standard deviation of the measure are reported.

5.1 Analysis of Onset and Duration of Blood Purification Therapy

We would like to analyze the effect of varying onset and duration of application of blood purification device. In (*Song et al.*, 2012) was suggested that optimal treatment should start at 18 hours after sepsis induction (roughly the time point in Figure 1.5 where non-survival group becomes distinguishable from survival group). It was also suggested that optimal treatment should last for 4 hours. We used a group of 5,000 virtual patients that would be in the non-survival population if no treatment was applied.

5.1.1 Therapy Onset

We tested how treatment efficacy depends on the starting time of the therapy. We applied the blood purification device continuously for 4 hours, as suggested in (*Song et al.*, 2012), with starting time varying from 2 to 30 hours after sepsis induction. We report percentage of rescued patients (patients for whom outcome after treatment was healthy, the higher percentage of rescued the better). The result is presented in Figure 5.2, blue line. We see that if treatment had been initiated later than 12 hours after sepsis induction, the percentage of rescued patients decreases with every hour. By applying therapy earlier (before 12 hours after sepsis induction), the percentage of rescued patients also decreases with every hour of early therapy. The graph shows that there is a critical time point around 10 hour at which the therapy is the most efficient.

This finding is in strong agreement with theoretical considerations of the sepsis stages and treatment effects. Sepsis treatment requires both a strong pro-inflammatory phase for the clearance of pathogen (Figure 5.3, area A) and an

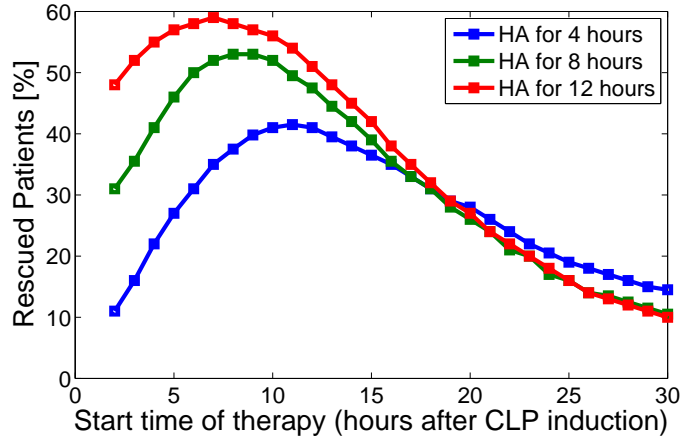


Figure 5.2: Blood purification therapy efficacy (percentage of rescued patients) with respect to therapy starting time and the duration of treatment.

anti-inflammatory phase for recovery (Figure 5.3, area C). A stage of an adversary influence of the pro-inflammatory response, which is disproportionate and counterproductive, is presented in Figure 5.3, area B. An inadequate treatment in either the pro-inflammatory (area A) or the immune-recovering anti-inflammatory phase (area C) might do more harm than good, while delayed treatment when immune response is counterproductive (area B) may significantly reduce the chance of survival.

5.1.2 Therapy Duration

We tested how treatment efficacy depends on the duration of continuous application of blood purification device. In addition to previous experiment, we applied the blood purification device continuously for 8 and 12 hours with varying therapy starting times. The plot in Figure 5.2 shows that with increased duration of HA application the percentage of rescued patients increases.

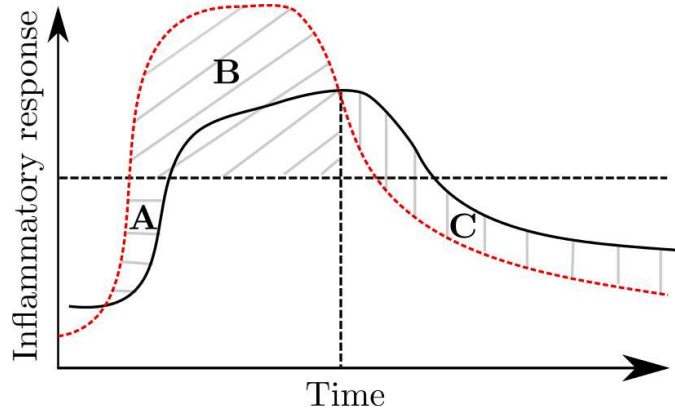


Figure 5.3: Theoretical considerations of the sepsis stages and treatment effects. $Time = 0$ - occurrence of an infection; red dotted line - pro-inflammatory response; blue solid line - anti-inflammatory response; black dashed horizontal line - response beyond which the process becomes adversary; black dashed vertical line - a tip-over point beyond which any therapy might be counterproductive; horizontal pattern (area A) - anti-inflammatory therapy likely harmful, pro-inflammatory therapy beneficial; diagonal pattern (area B) - likely maximal benefit from anti-inflammatory therapy; vertical pattern (area C) - anti-inflammatory response restoring patient state, any therapy likely harmful.

5.2 Treatment with IPED Therapy Onset

We applied our early classification method IPED to the virtual patients generated by the 19-states sepsis model. We used 30 patients (20 survival patients and 10 non-survival patients) for training and patients to test the model. The true positive rate of our method was 100%, meaning that all non-survival patients were predicted correctly and early. We then tested standard 18th hour blood purification therapy versus therapy initiated at IPED suggested time.

Known Sepsis Induction Time

This experiment assumes that sepsis induction time is known (common in laboratory conditions but uncommon in clinical practice). We applied 4-hour blood purification therapy on 5,000 non-survival patients and successfully rescued 31% of patients that would otherwise die (Figure 5.4, blue bar). When the 4-hour

blood purification therapy was started at IPED suggested time the percentage of rescued patients increased to 34%. This result shows that in laboratory conditions when sepsis induction time is known there is a benefit of using IPED to suggest start time of therapy.

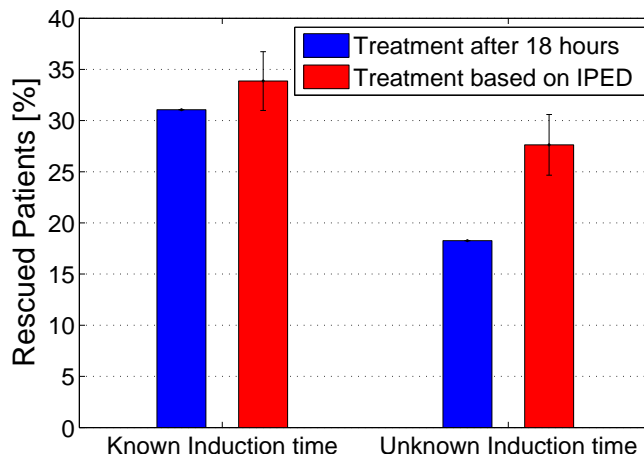


Figure 5.4: Percentage of rescued patients. Known sepsis induction time - patients are admitted at the time of the CLP-induced sepsis. Unknown sepsis induction time - patients are admitted with some delay after CLP-induced sepsis (uniformly sampled from 5-12 hours after the induction).

Unknown Sepsis Induction Time

In clinical practice the sepsis induction time is unknown. To simulate this the time when the patient visits the ICU was sampled uniformly from 5-12 hours after the sepsis induction.

We applied the 4-hour blood purification therapy at 18th hour from time when patient showed up. With this procedure the number of rescued patients is around 18% (Figure 5.4). When we applied the 4-hour therapy starting from IPED suggested time with respect to time when patient showed up we get an improvement to around 28% of rescued patients which again shows the huge benefit of using IPED to suggest start time of therapy.

5.3 Hybrid Treatment Strategy

We realized that patient who were suggested by IPED to receive therapy after 18th hour of sepsis induction (as many as one fourth of total population) may benefit if standard and IPED strategies were combined. Here we propose hybrid treatment strategy:

1. If a patient is recommended for treatment by IPED before the 18th hour, then the treatment is provided at the recommended time before the 18th hour.
2. If a patient is recommended for treatment by IPED after the 18th hour, then the treatment is provided at the 18th hour.

In Figure 5.5, we represent the percentage of IPED rescued patients with late treatment recommendation when the 4-hour therapy was provided at the recommended time (at the 19th hour or after as recommended by our IPED method). Percentage of rescued patients is around 22%. On the other hand, in the same figure we show the percentage of hybrid rescued patients with late treatment recommendation when the 4-hour therapy was provided at the 18th hour. The percentage of hybrid rescued patients is around 27%. Therefore, the hybrid treatment strategy saves more lives when compared to treatment solely based on IPED.

5.4 Conclusion

We applied our IPED method to make early diagnosis for septic patients. We showed that combination of early diagnosis and blood purification therapy can rescue more patients than standard approach for blood purification therapy. We also showed that a hybrid therapy that combines strengths of early and standard approaches and further improves the percentage of rescued patients.

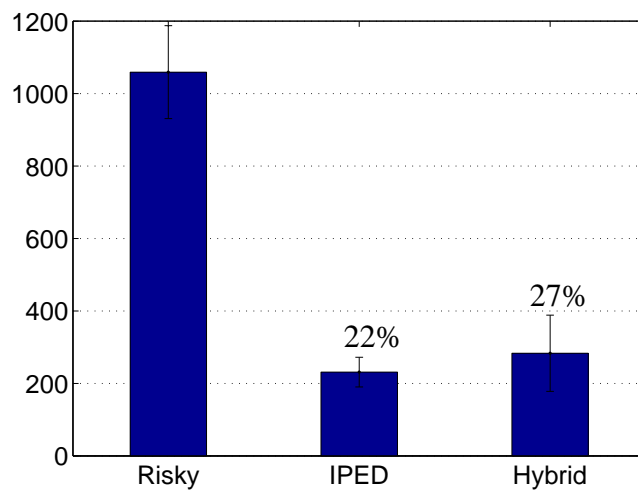


Figure 5.5: Hybrid treatment strategy. Risky - patients from non survival group where IPED predicts treatment after the 18th hour; IPED - rescued patients when the 4-hour therapy was provided at the recommended time (at 19th hour or after as recommended by our IPED method); Hybrid - rescued patients when the 4-hour therapy was provided at the 18th hour.

BIBLIOGRAPHY

- Allen, A. O., *Probability, Statistics, and Queuing Theory with Computer Science Applications*, Academic Press, Inc., San Diego, CA, 1990.
- Anderson, H. S., N. Parrish, K. Tsukida, and M. R. Gupta, Reliable early classification of time series, in *IEEE Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, 2012.
- Andrews, L. C., *Special functions of mathematics for engineers*, Spie Press, Oxford Science Publications, New York, 1998.
- Angus, D. C., The search for effective therapy for sepsis: Back to the drawing board?, *JAMA*, 306(23), 2614–2615, doi:10.1001/jama.2011.1853, 2011.
- Bagnall, A., L. Davis, J. Hills, and J. Lines, Transformation based ensembles for time series classification, in *Proceedings of the 12th SIAM International Conference on Data Mining*, pp. 307–319, 2012.
- Baranzini, S. E., et al., Transcription-based prediction of response to ifn using supervised computational methods, *PLoS Biol*, 3(1), 166–176, doi:10.1371/journal.pbio.0030002, 2005.
- Bertsimas, D., A. Chang, and C. Rudin, Ordered rules for classification: A discrete optimization approach to associative classification, in *Proceedings of ICML Workshop on Optimization in Machine Learning*, 2012.
- Borgwardt, K. M., S. Vishwanathan, and H.-P. Kriegel, Class prediction from time series gene expression profiles using dynamical systems kernels, in *Pacific Symposium on Biocomputing*, vol. 11, pp. 547–558, 2006.

- Boser, B. E., I. M. Guyon, and V. N. Vapnik, A training algorithm for optimal margin classifiers, in *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pp. 144–152, ACM, New York, NY, USA, 1992.
- Bousseljot, R., D. Kreisler, and A. Schnabel, Nutzung der ekg-signaldatenbank cardiodat der ptb über das internet, *Biomedizinische Technik*, p. 317, 1995.
- Box, G. E. P., G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, Wiley; 4th edition, 2008.
- Boyd, S., and L. Vandenberghe, *Convex Optimization*, Cambridge university press, 2004.
- Bracewell, R. N., *The Fourier Transform and Its Applications*, McGraw-Hill; 3 edition, 1999.
- Burges, C. J. C., A tutorial on support vector machines for pattern recognition, *Data Mining Knowledge Discovery*, 2, 121–167, 1998.
- Chang, A., D. Bertsimas, and C. Rudin, Ordered rules for classification: A discrete optimization approach to associative classification, Poster in Proceedings of Neural Information Processing Systems Foundations (NIPS), 2012a.
- Chang, C., and C. Lin, LIBSVM: A library for support vector machines, *ACM Transactions on Intelligent Systems and Technology*, 2, 1–27, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>, 2011.
- Chang, K.-W., B. Deka, W.-M. W. Hwu, and D. Roth, Efficient pattern-based time series classification on gpu, in *IEEE International Conference on Data Mining*, 2012b.

- Clermont, G., J. Rubin, and J. Day, Using nonlinear model predictive control to find optimal therapeutic strategies to modulate inflammation, *Mathematical Biosciences and Engineering*, 7(4), 739–763, 2010.
- Coleman, T. F., and Y. Li, An interior trust region approach for nonlinear minimization subject to bounds, *SIAM Journal on Optimization*, 6, 418–455, 1996.
- Collobert, R., F. Sinz, J. Weston, and L. Bottou, Trading convexity for scalability, in *International Conference of Machine Learning*, 2006.
- Costa, I. G., A. Schnhuth, C. Hafemeister, and A. Schliep, Constrained mixture estimation for analysis and robust classification of clinical time series, *Bioinformatics*, 25(12), i6–i14, 2009.
- Cover, T., and P. Hart, Nearest neighbor pattern classification, *IEEE Transactions on Information Theory*, IT-13(1), 21–27, 1967.
- Dasarathy, B. V., *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, IEEE Computer Society Press, 1990.
- Day, J., J. Rubin, and G. Clermont, Using nonlinear model predictive control to find optimal therapeutic strategies to modulate inflammation, in *Mathematical Biosciences and Engineering (MBE), ISDA '06*, vol. 7, pp. 739–763, 2010.
- Djuric, N., M. Grbovic, and S. Vucetic, Convex kernelized sorting, in *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 893–899, 2012.
- Esling, P., and C. Agon, Time-series data mining, *ACM Computing Surveys*, 45(12:1–12:34), 18571874, 2012.
- Fu, T., A review on time series data mining, *Engineering Applications of Artificial Intelligence*, 24, 164–181, 2011.

- Ghalwash, M. F., and Z. Obradovic, Early classification of multivariate temporal observations by extraction of interpretable shapelets, *BMC Bioinformatics*, 13(195), 2012.
- Ghalwash, M. F., D. Ramljak, and Z. Obradovic, Early classification of multivariate time series using a hybrid hmm/svm model, in *IEEE International Conference on Bioinformatics and Biomedicine*, Philadelphia, PA, 2012.
- Ghalwash, M. F., D. Ramljak, and Z. Obradovic, Early diagnosis and its benefits in sepsis blood purification treatment, in *International Workshop on Data Mining for Healthcare*, Philadelphia, PA, 2013a.
- Ghalwash, M. F., D. Ramljak, and Z. Obradovic, Patient-specific early classification of multivariate observations, *International Journal of Data mining and Bioinformatics*, in press, 2013b.
- Ghalwash, M. F., V. Radosavljevic, and Z. Obradovic, Extraction of interpretable multivariate patterns for early diagnostics, in reviewa.
- Ghalwash, M. F., V. Radosavljevic, and Z. Obradovic, Uncertainty estimation for interpretable early classification, in reviewb.
- Giraud-Carrier, C., Beyond predictive accuracy: What?, in *ECML-98 Workshop on Upgrading Learning to Meta-Level: Model Selection and Data Transformation*, pp. 78–85, 1998.
- Goldberger, A. L., et al., Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals, *Circulation*, 101, e215–e220, 2000.
- Goodwin, G. C., P. J. Ramadge, and P. E. Caines, Discrete time multivariable

- adaptive control, in *18th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, pp. 335 – 340, 1979.
- Griffin, M., and J. Moorman, Toward the early diagnosis of neonatal sepsis and sepsis-like illness using novel heart rate analysis, *Pediatrics*, 107(1), 97–104, 2001.
- Jain, A. K., R. C. Dubes, and C.-C. Chen, Bootstrap techniques for error estimation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5), 628–633, 1987.
- Jennings, D. L., T. M. Amabile, and L. Ross, *Judgment Under Uncertainty: Heuristics and Biases*, Cambridge Press, 1982.
- Keogh, E., and S. Kasetty, On the need for time series data mining benchmarks: A survey and empirical demonstration, in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 102–111, 2002.
- Keogh, E., Q. Zhu, B. Hu, X. X. Y. Hao, L. Wei, and C. A. Ratanamahatana, The UCR time series classification and clustering homepage http://www.cs.ucr.edu/~eamonn/time_series_data/, 2011.
- Lee, Y., C. Wei, T. Cheng, and C. Yang, Nearest-neighbor-based approach to time-series classification, *Decision Support Systems*, 53(1), 207–217, 2012.
- Lendasse, A., V. Wertz, and M. Verleysen, Model selection with cross-validations and bootstraps - application to time series prediction with rbf models, in *Artificial Neural Networks and Neural Information Processing ICANN/ICONIP 2003*, pp. 573–580, Springer-Verlag, 2003.

- Liao, T., Clustering of time series dataa survey, *Pattern Recognition*, 38(11), 18571874, 2005.
- Lin, T., N. Kaminski, and Z. Bar-Joseph, Alignment and classification of time series gene expression in clinical studies, *Bioinformatics*, 24, i147–i155, doi: 10.1093/bioinformatics/btn152, 2008.
- Lines, J., L. M. Davis, J. Hills, and A. Bagnall, A shapelet transform for time series classification, in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 289–297, 2012.
- Mueen, A., E. Keogh, and N. Young, Logical-shapelets: An expressive primitive for time series classification, in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1154–1162, 2011.
- Rabiner, L. R., A tutorial on HMM and selected applications in speech recognition, *Proceedings of the IEEE*, 77(2), 257–286, 1989.
- Rabiner, L. R., and B. H. Juang, An introduction to hidden markov models, *IEEE ASSP Magazine*, 3(1), 4–16, 1986.
- Rakthanmanon, T., and E. Keogh, Fast-shapelets: A fast algorithm for discovering robust time series shapelets, in *Proceedings of 11th SIAM International Conference on Data Mining*, 2011.
- Rimmele, T., and J. A. Kellum, Clinical review: Blood purification for sepsis, *Crit Care*, 15(1), 205, 2011.
- Ristovski, K., V. Radosavljevic, and Z. Obradovic, A data mining approach for optimization of acute inflammation therapy, in *IEEE International Conference on Bioinformatics and Biomedicine*, 2012.

- Song, S. O. K., J. Hogg, Z.-Y. Peng, R. S. Parker, J. A. Kellum, and G. Clermont, Ensemble models of neutrophil trafficking in severe sepsis, *PLoS Computational Biology*, 8(3), 2012.
- Spellman, P. T., G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher, Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization, *Molecular Biology of the Cell*, 9, 3273–3297, 1998.
- Tan, P., M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Pearson Addison Wesley, 2006.
- Tchagang, A. B., K. V. Bui, T. McGinnis, and P. V. Benos, Extracting biologically significant patterns from short time series gene expression data, *BMC Bioinformatics*, 10, 2009.
- Thiel, S. W., J. M. Rosini, W. Shannon, J. A. Doherty, S. T. Micek, and M. H. Kollef, Early prediction of septic shock in hospitalized patients, *Journal of hospital medicine: an official publication of the Society of Hospital Medicine*, 5(1), 19–25, 2010.
- Vellido, A., J. D. Martin-Guerrero, and P. J. Lisboa, Making machine learning models interpretable, in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2012.
- Wiens, J., J. Guttag, and E. Horvitz, Patient risk stratification for hospital-associated c. diff as a time-series classification task, in *Neural Information Processing System (NIPS)*, 2012.
- Xing, Z., J. Pei, and P. S. Yu, Early prediction on time series: A nearest neighbor

- approach, in *Proceedings of the 21st international joint conference on Artificial intelligence*, pp. 1297–1302, 2009.
- Xing, Z., J. Pei, and P. S. Yu, Early classification on time series, *Knowl Inf Syst*, 31, 105–107, 2011a.
- Xing, Z., J. Pei, P. S. Yu, and K. Wang, Extracting interpretable features for early classification on time series, in *Proceedings of 11th SIAM International Conference on Data Mining*, pp. 439–451, 2011b.
- Ye, L., and E. Keogh, Time series shapelets: A new primitive for data mining, in *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2009.
- Yuille, A., and A. Rangarajan, The concave-convex procedure (CCCP), in *Neural Computation*, vol. 15, pp. 915–936, 2003.
- Zaas, A. K., et al., Gene expression signatures diagnose influenza and other symptomatic respiratory viral infections in humans, *Cell Host and Microbe*, 6(3), 207–217, 2009.
- Zuev, S. M., S. F. Kingsmore, and D. D. Gessler, Sepsis progression and outcome: A dynamical model, *Theoretical Biology and Medical Modelling*, 2006.