

**ON LEVERAGING REPRESENTATION LEARNING TECHNIQUES FOR DATA  
ANALYTICS IN BIOMEDICAL INFORMATICS**

---

A Dissertation  
Submitted to  
the Temple University Graduate Board

---

In Partial Fulfillment  
of the Requirements for the Degree of  
DOCTOR OF PHILOSOPHY

---

by  
Xi Hang Cao  
August 2019

Examining committee members:

Dr. Zoran Obradovic, Advisory Chair, Dept. of Computer and Information Sciences  
Dr. Slobodan Vucetic, Dept. of Computer and Information Sciences  
Dr. Richard Souvenir, Dept of Computer and Information Sciences  
Dr. Avi Kaplan, External Member, College of Education

# ABSTRACT

Representation Learning is ubiquitous in state-of-the-art machine learning workflow, including data exploration/visualization, data preprocessing, data model learning, and model interpretations. However, the majority of the newly proposed Representation Learning methods are more suitable for problems with a large amount of data. Applying these methods to problems with a limited amount of data may lead to unsatisfactory performance. Therefore, there is a need for developing Representation Learning methods which are tailored for problems with “small data”, such as, clinical and biomedical data analytics. In this dissertation, we describe our studies of tackling the challenging clinical and biomedical data analytics problem from four perspectives: data preprocessing, temporal data representation learning, output representation learning, and joint input-output representation learning.

Data scaling is a critical component in data preprocessing. The objective in data scaling is to scale/transform the raw features into reasonable ranges such that the machine learning model equally exploits each feature of an instance. For example, in a credit fraud detection task, a machine learning model may utilize a person’s credit score and annual income as features, but because the ranges of these two features are different, a machine learning model may consider one more heavily than another. In this dissertation, I thoroughly introduce the problem in data scaling and describe an approach for data scaling which can intrinsically handle the outlier problem and lead to better model prediction performance.

Learning new representations for data in the unstandardized form is a common task in

data analytics and data science applications. Usually, data come in a tubular form; namely, the data is represented by a table in which each row is a feature (row) vector of an instance. However, it is also common that the data are not in this form; for example, texts, images, and video/audio records. In this dissertation, I describe the challenge of analyzing imperfect multivariate time series data in healthcare and biomedical research and show that the proposed method can learn a powerful representation to encounter various imperfections and lead to an improvement of prediction performance.

Learning output representations is a new aspect of Representation Learning, and its applications have shown promising results in complex tasks, including computer vision and recommendation systems. The main objective of an output representation algorithm is to explore the relationship among the target variables, such that a prediction model can efficiently exploit the similarities and potentially improve prediction performance. In this dissertation, I describe a learning framework which incorporates output representation learning to time-to-event estimation. Notably, the approach learns the model parameters and time vectors simultaneously. Experimental results do not only show the effectiveness of this approach but also show the interpretability of this approach from the visualizations of the time vectors in 2-D space.

Learning the input (feature) representation, output representation, and predictive modeling are closely related to each other. Therefore, it is a very natural extension of the state-of-the-art by considering them together in a joint framework. In this dissertation, I describe a large-margin ranking-based learning framework for time-to-event estimation with joint input embedding learning, output embedding learning, and model parameter learning. In the framework, I cast the functional learning problem to a kernel learning problem, and by adopting the theories in Multiple Kernel Learning, I propose an efficient optimization algorithm. Empirical results also show its effectiveness on several benchmark datasets.

*To my wife Qiqing.*

*To my parents Jingyun and Lexia.*

*To my brother Jianyong, my sister-in-law Yingshi, and my niece Ariel.*

*To my parents-in-law Shichang and Jun*

*To all of my friends*

*Thank you for your constant support and love  
throughout the writing of this dissertation  
and for always being there for me.*

# ACKNOWLEDGEMENTS

I would like to express my special appreciation and sincere gratitude to my advisor Dr. Zoran Obradovic for his continuous support during my PhD studies and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. Special thanks to Dr. Slobodan Vucetic, Dr. Richard Souvenir and Dr. Avi Kaplan, great professors and researchers from whom I learned a lot, for serving on my dissertation committee and for providing me with useful comments and valuable suggestions.

I would especially like to thank my teammate and co-author on most of my papers, Chao Han, for his inexhaustible source of ideas, time, effort and discussions that lead to our achievements and papers. I would like to thank a lot to my other co-authors Dr. Ivan Stojkovic, Dr. Mohamed Ghalwash, Dr. Ting Dai, Dr. Tong Perez, Dr. Jennifer Cromley, Dr. Kyle Mara, Dr. Michael Balsai, Dr. Ken Kim, Dr. Heng Luo, Dr. Ping Zhang, Dr. Lun Yang, Dr. Xioajing Du, Dr. Paul Ratazzi, Dr. Allen Kindman, Shoumik Roychoudhury, Nima Asadi, Martin Pavlovski, Branimir Ljubic, and Lucus Glass for their valuable ideas, discussions and contributions to our joint papers.

Special thanks to professors Paul LaFollette, Jr. and David Shulman, for whom I was working as a teaching assistant. Their experience and enthusiasm were of great inspiration throughout my years of teaching. I am also grateful to all my students at the Department of Computer and Information Sciences.

I thank my fellow labmates, postdocs and visiting scholars at the Center for Data Analytics and Biomedical Informatics: Ana K., Ana S., Branimir, Branko, Chao, Djordje, Dusan, Fang, Ivan, Jelena, Jesse, Jumanah, Kosta, Marija, Martin, Milan B., Milan V., Milos J., Milos J., Milos R., Mladen, Mohamed, Mohammad, Nancy, Nima, Noor, Nouf, Shoumik, Tamiris, Thomas, Tijana, Vladan, Vladisav, for sharing their knowledge and perspectives during our lab meetings.

I would also like to thank my friends, colleagues and mentors at HRL Laboratory and U.S. Air Force Research Lab for the summer internship/fellowship opportunities in their groups and leading me working on diverse exciting projects. Those were invaluable and unforgettable experiences.

I thank professors and staff at the Department of Computer and Information Sciences for making such a friendly and pleasant working environment. I would especially like to thank Julie Srocki, Hailey King, Christopher Bryant, and professor Justin Y. Shi, for all the help and for always being there for us graduate students.

I would like to acknowledge the support of IQVIA, Temple Student Retention Project, DARPA THOR Project, and DARPA Sepsis Project during my studies.

Finally, I would like to thank all my family and friends back home in Zhongshan China, Honolulu Hawaii, and here in Philadelphia and across the United States of America for being with me throughout all these years.

# TABLE OF CONTENTS

<b>ABSTRACT</b>	<b>ii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>LIST OF TABLES</b>	<b>xi</b>
<b>LIST OF FIGURES</b>	<b>xiii</b>
<b>1 INTRODUCTION</b>	<b>1</b>
<b>2 A ROBUST DATA SCALING ALGORITHM TO IMPROVE CLASSIFICATION ACCURACIES IN BIOMEDICAL DATA</b>	<b>5</b>
2.1 Introduction . . . . .	5
2.2 Background . . . . .	6
2.2.1 Data Scaling in Classification Modeling . . . . .	7
2.2.2 Commonly Used Data Scaling Algorithms . . . . .	8
2.3 Methods . . . . .	9
2.3.1 Approximation of the Cumulative Density Function . . . . .	10
2.3.2 Parameter Initialization . . . . .	12
2.3.3 Qualitative Comparisons of Data Scaling algorithms . . . . .	13
2.4 Conclusion . . . . .	21
<b>3 LEARNING A DYNAMIC-BASED REPRESENTATION FOR MULTIVARIATE BIOMARKER TIME SERIES CLASSIFICATIONS</b>	<b>25</b>
3.1 Introduction . . . . .	25
3.2 Background . . . . .	26

3.2.1	Time series data in biomedical research . . . . .	26
3.2.2	Imperfections in biomedical time series data . . . . .	27
3.2.3	Using a linear dynamical system to represent a multivariate time series . . . . .	28
3.2.4	Contributions . . . . .	28
3.3	Related Work . . . . .	30
3.3.1	Related work in learning LDS . . . . .	30
3.3.2	Related work in learning representation from MTS's . . . . .	31
3.4	Methods . . . . .	32
3.4.1	Notations . . . . .	32
3.4.2	Learning an LDS from an imperfect MTS . . . . .	32
3.4.3	Learning the dynamics matrix from complete and accurate measurements . . . . .	34
3.4.4	Learning an LDS from noisy MTS measurements . . . . .	35
3.4.5	Learning an LDS from partially observed MTS measurements . . . . .	36
3.4.6	Solving the optimization problem . . . . .	36
3.4.7	Classifying LDS's via a Kernel method . . . . .	38
3.5	Experiments and Results . . . . .	42
3.5.1	Experiments on synthetic MTS data . . . . .	42
3.5.2	Experiments on real-life datasets . . . . .	46
3.5.3	Data preprocessing . . . . .	48
3.5.4	Experiments on using all available data in the datasets . . . . .	48
3.5.5	Experiments on the H3N2 dataset with imposed imperfectness . . . . .	49
3.6	Conclusion . . . . .	52
<b>4</b>	<b>TIME-TO-EVENT ESTIMATION BY RE-DEFINING TIME</b>	<b>53</b>
4.1	Introduction . . . . .	53

4.2	Background . . . . .	54
4.3	Related Work . . . . .	56
4.4	Method . . . . .	58
4.4.1	Notations . . . . .	58
4.4.2	Problem formulation . . . . .	59
4.4.3	Objective Function Formulation . . . . .	60
4.4.4	Optimization . . . . .	62
4.5	Experiments . . . . .	66
4.5.1	Setup . . . . .	66
4.5.2	Experiment #1: comparisons to state-of-the-art models on benchmark datasets . . . . .	67
4.5.3	Experiment #2: Regimes identification by visualizing the learned time concept vectors . . . . .	70
4.5.4	Experiment #3: model scalability evaluation on synthetic datasets . . . . .	76
4.6	Conclusion . . . . .	78
<b>5</b>	<b>LEARNING INPUT AND OUTPUT KERNELS FOR TIME-TO-EVENT ESTIMATION ON HIGH-DIMENSIONAL DATA</b>	<b>79</b>
5.1	Introduction . . . . .	79
5.2	Background . . . . .	80
5.3	Related Work . . . . .	82
5.4	Method . . . . .	84
5.4.1	Notations . . . . .	84
5.4.2	Problem formulation . . . . .	85
5.4.3	Optimization . . . . .	86
5.5	Experiments . . . . .	91
5.5.1	Setup . . . . .	91
5.5.2	Comparisons to State-of-the-art Models on Benchmark Datasets . . . . .	92

5.6 Conclusion . . . . .	97
<b>6 CONCLUSION</b>	<b>98</b>
<b>BIBLIOGRAPHY</b>	<b>102</b>

# LIST OF TABLES

2.1	Summary of datasets used in experiments (sorted by the no. of subjects in ascending order) . . . . .	22
2.2	Results of 16 datasets (16 binary classification tasks) using different data scaling algorithms and classification models. The performances are measured by the average proportion of correct classification in 5-fold cross-validations. The means and 95% confidence intervals are included. Column names: None - no data scaling; Minmax - Min-max algorithm; Zscore - Z-score algorithm; GL - GL algorithm. Best performances are emphasized in bold . . . . .	23
2.3	Results of 16 datasets (16 binary classification tasks) using different data scaling algorithms and classification models. The performances are measured by the average Area Under the ROC in 5-fold cross-validations. The means and 95% confidence intervals are included. Column names: None - no data scaling; Minmax - Min-max algorithm; Z-score - Z-score algorithm; GL - GL algorithm. Best performances are emphasized in bold . . . . .	24
3.1	Notations and definitions . . . . .	33
3.2	Summary of 3 datasets . . . . .	47
3.3	Leave-one-out cross validation accuracies on 3 binary classification tasks. Best performances are in bold . . . . .	49
4.1	Notations and definitions. . . . .	59
4.2	Summaries of the 7 benchmark datasets (ordered by # example). . . . .	68
4.3	Concordance Indices (CI's) and corresponding standard deviations of all the models; The ranking of a model among all compared models in each dataset is included. . . . .	71
5.1	Notations and definitions. . . . .	85
5.2	The predetermined input kernels used in our experiments . . . . .	92

5.3	Summaries of the 7 benchmark datasets (ordered by # example). . . . .	93
5.4	Concordance Indices (CI's) and corresponding standard deviations of all the models; The ranking of a model among all compared models in each dataset is included. . . . .	94

# LIST OF FIGURES

2.1	Fitting of the ECDF using the GL algorithm: an example showing the approximation of an ECDF using a generalized logistic (GL) function. . .	14
2.2	Behavior of data scaling algorithms without outliers. When there is no outlier in the data, the behavior of the Min-max algorithm, Z-score algorithm and the GL algorithm is very similar. . . . .	16
2.3	Behavior of data scaling algorithms with outliers. When there is an outlier in the data, the behaviors of the Min-max algorithm and Z-score algorithm are significantly affected, but the impact of the outlier on the GL algorithm is neglectable. . . . .	17
2.4	An 2D illustration on how the GL algorithm can affect the classification accuracy. (a) raw data without scaling; (b) data scaled by the Min-max algorithm; (c) data scaled by the Z-score algorithm; (d) data scaled by the GL algorithm. . . . .	18
3.1	Examples of trajectories generated by the same LDS with noisy initial conditions and similarity plots of generated trajectories and learned dynamics matrices. a) and b) two example trajectories generated by the same LDS with noisy initial states; the same color indicates the same state. c) and d) similarity plots of generated trajectories and learned dynamics matrices; the same color indicates the trajectories generated from the same LDS. . .	44
3.2	Examples of trajectories generated by the same LDS with noisy dynamics matrices and similarity plots of generated trajectories and learned dynamics matrices. a) and b) two example trajectories generated by the same LDS with noisy initial states; the same color indicates the same state. c) and d) similarity plots of generated trajectories and learned dynamics matrices; the same color indicates the trajectories generated from the same LDS. . . . .	45

3.3	Means and standard deviations of each methods as the percentage of missing time points increases from 20% to 80%. The vertical line indicates one standard deviation. . . . .	50
3.4	Means and standard deviations of each methods as the percentage truncating increases from 20% to 80%. The vertical line indicates one standard deviation. . . . .	51
4.1	The proposed model first determines the corresponding time concept of each example and then infers the event occurrence time. . . . .	55
4.2	The Critical Difference Diagram shows the average rankings and overall rankings of all the compared models in the measure of Concordance Index. . . . .	72
4.3	2D scatter plots of the learned time concept vectors in the VDV datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes. . . . .	73
4.4	2D scatter plots of the learned time concept vectors in the VDV datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes. . . . .	73
4.5	2D scatter plots of the learned time concept vectors in the MCL datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes. . . . .	74
4.6	2D scatter plots of the learned time concept vectors in the NSBCD datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes. . . . .	74
4.7	2D scatter plots of the learned time concept vectors in the AML datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes. . . . .	75
4.8	2D scatter plots of the learned time concept vectors in the DLBCL datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes. . . . .	75
4.9	2D scatter plots of the learned time concept vectors in the DBCD datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes. . . . .	76

4.10	Learning time of the proposed model with different $n, m, T$ , and $d$ settings.	
	a) learning time changes as $n$ changes while other variables are constant;	
	b) learning time changes as $m$ changes while other variables are constant;	
	c) learning time changes as $T$ changes while other variables are constant;	
	d) learning time changes as $d$ changes while other variables are constant. .	77
5.1	The proposed model first determines the corresponding time concept of each example and then infers the event occurrence time. . . . .	82
5.2	The Critical Difference Diagram shows the average rankings and overall rankings of all the compared models in the measure of Concordance Index.	96

# CHAPTER 1

## INTRODUCTION

Representation Learning has been a very hot research area in Machine Learning. While high-capacity models, such as Deep Neural Networks, have achieved excellent performance in a variety traditionally challenging tasks, for example, computer vision, machine translation, text mining; however, they have a significant drawback: a massive volume of data is required to train the model. In this thesis, we focus on Representation Learning techniques which are tailored to address the learning challenges in problems with a small amount of data; for example, clinical and biomedical data analytics.

In Chapter 2, an approach for learning a representation based on data scaling and its applications in biomedical data analytics are described. Machine learning models have been adapted in biomedical research and practice for knowledge discovery and decision support. While mainstream biomedical informatics research focuses on developing more accurate models, the importance of data scaling draws less attention. We propose the Generalized Logistic (GL) algorithm that scales data uniformly to an appropriate interval by learning a generalized logistic function to fit the empirical cumulative distribution function of the data. The GL algorithm is simple yet effective; it is intrinsically robust to outliers, so it is particularly suitable for diagnostic/classification models in clinical/medical applications

where the number of samples is usually small; it scales the data in a nonlinear fashion, which leads to a potential improvement in accuracy. To evaluate the effectiveness of the proposed algorithm, experiments on 16 binary classification tasks with different variable types and cover a wide range of applications are conducted. The resultant performance in terms of area under the receiver operating characteristic curve (AUROC) and percentage of correct classification showed that models learned using data scaled by the GL algorithm outperform the ones using data scaled by the Min-max and the Z-score algorithm, which are the most commonly used data scaling algorithms. The proposed GL algorithm is simple and effective. It is robust to outliers, so no additional denoising or outlier detection step is needed in data preprocessing. Empirical results also show that models learned from data scaled by the GL algorithm have higher accuracy compared to the commonly used data scaling algorithms.

In Chapter 3, an approach of learning the representations for biomedical multivariate time-series data and its applications are described. Time series in healthcare practices and biomedical research are typically multivariate, i.e., multiple biomarkers are observed simultaneously at a time. However, they tend to be short, noisy, unaligned, irregularly sampled, partially observed, and with only limited samples. These imperfections pose a challenge for mining information from data. In this work, we propose to use dynamic-based representations to present such imperfect multivariate time series. Specifically, we propose an approach to learn a corresponding Linear Dynamical System (LDS) for a multivariate time series example and use the set of system parameters as a representation for that example. Such a representation can capture interactions of different variables and provide a unified view of multivariate time series with different lengths, different missingness mechanisms, and different starting points. Other techniques are then used to mine useful information and perform learning tasks based on the new representation. For example, we use support vector machine classification models with LDS kernels in time series classification tasks. To evaluate the effectiveness of the proposed approach, we conducted exper-

iments on both synthetic data sets and real-life datasets. The results in synthetic datasets demonstrated that the proposed approach could correctly learn the similarities of underlying linear dynamical systems. Our real-life data sets included human influenza A (H3N2), Rhinovirus (HRV), and respiratory syncytial virus (RSV) gene expression time series. The accuracies in the leave-one-out symptomatic/asymptomatic diagnostic tasks showed that our approach outperformed three baseline algorithms. Moreover, in experiments where various levels of imperfections were imposed on the H3N2 dataset, the accuracies of other baseline methods degraded significantly, but the accuracy of our approach remained high.

Chapter 4 and 5 focus on output Representation Learning and its applications in the problem of time-to-event estimation. The primary goal of a time-to-event estimation model is to infer the occurrence time of a target event accurately. Most existing studies focus on developing new models to utilize the information in the censored observations effectively. In chapter 4, an output Representation Learning model is described to tackle the time-to-event estimation problem. The model relaxes a fundamental constraint that the target variable, time, is a univariate number which satisfies a partial order. Instead, the proposed model interprets each event occurrence time as a time concept with a vector representation. We hypothesize that the model is more accurate and interpretable by capturing 1) the relationships between features and time concept vectors and 2) the relationships among time concept vectors. We also propose a scalable framework to learn the model parameters and time concept vectors simultaneously. Besides, similarity information among time concept vectors helped in identifying time regimes, thus leading to a potential knowledge discovery related to the human cancer datasets considered in our experiments. The model described in Chapter 5 complements some of the drawbacks in the model in Chapter 4. The proposed model adopts a kernel-based large-margin learning framework and simultaneously learns an input (feature vector) kernel and an output (event time) kernel to leverage the similarities among features and the similarities among event times. Both of the models are evaluated and analyzed in experiments on 7 benchmark

datasets, and they are compared to 15 traditional and state-of-the-art models. The results demonstrated the efficiency and effectiveness of the proposed models.

# CHAPTER 2

## A ROBUST DATA SCALING ALGORITHM TO IMPROVE CLASSIFICATION ACCURACIES IN BIOMEDICAL DATA

### 2.1 Introduction

Machine learning models have been adapted in biomedical research and practice for knowledge discovery and decision support. While mainstream biomedical informatics research focuses on developing more accurate models, the importance of data preprocessing draws less attention. We propose the Generalized Logistic (GL) algorithm that scales data uniformly to an appropriate interval by learning a generalized logistic function to fit the empirical cumulative distribution function of the data. The GL algorithm is simple yet effective; it is intrinsically robust to outliers, so it is particularly suitable for diagnostic/classification models in clinical/medical applications where the number of samples is usually small; it scales the data in a nonlinear fashion, which leads to potential improvement in accuracy. To evaluate the effectiveness of the proposed algorithm, we conducted experiments on 16 binary classification tasks with different variable types and cover a wide range of applications. The resultant performance in terms of area under the receiver operation

characteristic curve (AUROC) and percentage of correct classification showed that models learned using data scaled by the GL algorithm outperform the ones using data scaled by the Min-max and the Z-score algorithm, which are the most commonly used data scaling algorithms. The proposed GL algorithm is simple and effective. It is robust to outliers, so no additional denoising or outlier detection step is needed in data preprocessing. Empirical results also show models learned from data scaled by the GL algorithm have higher accuracy compared to the commonly used data scaling algorithms.

## 2.2 Background

There is an increasing interest in research and development of machine learning and data mining techniques for aid in biomedical studies as well as in clinical decision making Kourou et al. (2015) Swan et al. (2013) Kelchtermans et al. (2014) Foster et al. (2014). Typically, statistical learning methods are performed on the data of observed cases to yield diagnostic or prognostic models that can be applied in future cases in order to infer the diagnosis or predict the outcome. Such learned models might be used to assist physicians in guiding their decisions, and are sometimes shown to outperform the experts' prediction accuracy Maltoni et al. (2005). Furthermore, such models can discover previously unrecognized relations between the variables and outcome improving knowledge and understanding of the condition. Such discoveries may result in improved treatments or preventive strategies. Given that predictive models compute predictions based on information of a particular patient, they are also promising tools for achieving the goal of personalized medicine.

Predictive models have huge potential because of their ability to generalize from data. Even though predictive models lack the skills of a human expert, they can handle much larger amounts of data and can potentially find subtle patterns in the data that a human could not. Predictive models rely heavily on training data, and are dependent on data

quality. Ideally, a model should extract the existing signal from the data and disregard any spurious patterns (noise). Unfortunately, this is not an easy task, since data are often far from perfect; some of the imperfections include irrelevant variables, small numbers of samples, missing values, and outliers.

Therefore, data preprocessing is common and necessary in order to increase the ability of the predictive models to extract useful information. There are various approaches targeting different aspects of data imperfection; such as imputations for missing values, smoothing for removing the superimposed noise, or excluding the outlier examples. Then there are various transformations of variables, from common scaling and centering of the data values, to more advanced feature engineering techniques. Each of those techniques can make a significant improvement in predictive model performance when learned on the transformed data.

### *2.2.1 Data Scaling in Classification Modeling*

In the machine learning and data mining community, data scaling and data normalization refer to the same data preprocessing procedure, and these two terminologies are used interchangeably; their aim is to consolidate or transfer the data into ranges and forms that are appropriate for modeling and mining Han et al. (2011). Models trained on scaled data usually have significantly higher performance compared to the models trained on unscaled data, so data scaling is regarded as an essential step in data preprocessing. Data scaling is particularly important for methods that utilize distance measures, such as nearest neighbor classification and clustering. In addition, artificial Neural Network models require the input data to be normalized, so that the learning process can be more stable and faster Haykin (2009).

*Confusions of Gene Expression Normalization* In medicine, gene expression data obtained from microarray technology are widely used for disease/cancer diagnoses. Usually, a

normalization step is conducted for the purpose of identifying and removing sources of systematic variation in the measured fluorescence Dudoit et al. (2002), before the data are ready for analysis. However, the gene expression normalization step is not equivalent to the data scaling step that we study in this context. In most cases, a normalized gene expression dataset needs to be processed/scaled by a data scaling step before learning a classification model. The models that are learned from gene expression data with scaling usually outperform the models that are learned from gene expression data without scaling, with considerable margins.

### 2.2.2 Commonly Used Data Scaling Algorithms

Two data scaling algorithms are widely used: Min-max algorithm and Z-score algorithm.

*Min-max Algorithm* In the Min-max algorithm, the original data are linearly transformed. We denote  $x_{min}$  and  $x_{max}$  as the minimum and the maximum of a variable in the samples. The Min-max algorithm maps a value,  $v$ , of this variable to a value,  $v'$ , using the following formula:

$$v' = \frac{v - x_{min}}{x_{max} - x_{min}} + x_{min}. \quad (2.1)$$

The Min-max algorithm scales a variable in the training samples in the interval of  $[x_{min}, x_{max}]$  to  $[-1, 1]$  (or  $[0, 1]$ ) by using a linear mapping. However, when the unseen/testing samples fall outside of the training data range of the variable, the scaled values will be out of the bounds of the interval  $[-1, 1]$  (or  $[0, 1]$ ), and that may pose problems in some applications; in addition, it is very sensitive to outliers, as shown in latter sections.

*Z-score Algorithm* In the Z-score algorithm, the new value,  $v'$ , of a variable, is scaled from the original value,  $v$ , using the formula:

$$v' = \frac{v - \bar{x}}{\sigma_x}, \quad (2.2)$$

where  $\bar{x}$  and  $\sigma_x$  are the mean and standard deviation of the variable values in the training samples, respectively. After the scaling, the new values will have value 0 as the mean, and value 1 as the standard deviation. This algorithm does not map the original data into an interval, and it is also sensitive to outliers. When the number of examples is small, especially in scenarios in biomedical research, the mean and standard deviation calculated from the data may not be able to approximate the true mean and standard deviation well, so future input values will be scaled poorly.

## 2.3 Methods

The idea of the GL algorithm for data scaling is adapted from the histogram equalization technique, and it can map both the original and future data into a desired interval. The algorithm has no assumption on the sample distribution and utilizes generalized logistic functions to approximate cumulative density functions. Since it maps data into a uniformly distributed range of values, the points that were previously densely concentrated on some interval become more discernible, which allows more room for representation of the subtle differences between them. In addition, the GL algorithm reduces the distance of outliers from other samples, which makes the algorithm robust to the outliers. This advantage is particularly significant in diagnostic/classification modeling in medicine and healthcare, where the number of samples is usually small, and outliers have a huge impact on the model training, leading to poor accuracy.

In a preliminary study Cao and Obradovic (2015), the GL algorithm was effective in classifying tasks with microarray gene expression data. In this manuscript we have significantly extended our preliminary work in the following ways:

1. providing a thorough description of the proposed GL algorithm as well as intuitive and qualitative explanations of scenarios where the new algorithm is superior to the Min-max and Z-score algorithms;

2. extending the GL algorithm to include a much better and more general parameter initialization for the non-convex optimization, which is a critical part of the algorithm for fitting the generalized logistic function to the empirical cumulative distribution function;
3. empirically demonstrating that the GL algorithm is not only effective in gene expression classification tasks, but also in a broad variety of different diagnostic/classification tasks with different types of variables.

### *Data Scaling Formula*

We model the values of a variable in the samples as a random variable (r.v.)  $X$ . In the GL algorithm, the scaled value  $v'$  of a value,  $v$ , is obtained by

$$v' = P_X(v), \quad (2.3)$$

where  $P_X(\cdot)$  is the cumulative density function (CDF) of the r.v.  $X$ .

Using a CDF as a mapping can be also seen in the Histogram Equalization technique Gonzalez and Woods (2008) in the field of Digital Image Processing for image contrast enhancement. The difference of the GL algorithm versus the Histogram Equalization technique is that we do not only use the CDF to scale the data, but also learn/approximate the functional expression of the CDF, so that it can be used to scale unseen values.

#### *2.3.1 Approximation of the Cumulative Density Function*

From the data, we do not know the exact functional form of the cumulative density function (CDF) of an variable whose value is represented by the r.v.  $X$ ; therefore, we need to approximate the CDF. We can find the empirical cumulative density function (ECDF) using the formula

$$\hat{P}_X(v) = \frac{1}{n} \sum_{i=1}^n 1_{x_i \leq v}, \quad (2.4)$$

where  $\hat{P}_X(v)$  is the ECDF at a value  $v$ ,  $n$  is the number of samples, and  $x_i$  is the value of the variable in the  $i^{th}$  sample.

Unfortunately, in most cases, the ECDF has no functional form expression. Moreover, the original data tend to be noisy, so the ECDF is usually very bumpy. Therefore, we use a generalized logistic (GL) function to approximate the ECDF. It has been proven that a logistic function can be used to accurately approximate the CDF of a normal distribution Bowling et al. (2009). In this algorithm, we do not make any assumption on the distribution of the data; therefore, we use a more general form of the logistic function, called the generalized logistic (GL) function

$$L(x) = \frac{1}{(1 + Qe^{-B(x-M)})^{1/\nu}}. \quad (2.5)$$

Compared to the logistic function used in Bowling et al. (2009), this GL function provides the flexibility to approximate a more variety of distributions. One of the notable properties of (2.5) is that it maps the values in the interval  $(-\infty, \infty)$  to the interval  $(0,1)$ . This property makes our GL algorithm robust to outliers, and guarantees that the scaled data will be in  $(0,1)$ .

In order to approximate the ECDF, we need to learn the parameters  $Q$ ,  $B$ ,  $M$ , and  $\nu$  from the data, so that the GL function could best fit the ECDF. The sum of squared differences of the GL function and the ECDF can be represented by

$$\eta = \sum_{i=1}^n \|L(x_i) - \hat{P}_X(x_i)\|^2. \quad (2.6)$$

The best set of parameters is the minimizer of  $\eta$ , so the key to find the most appropriate GL function to approximate the ECDF is to solve an optimization problem

$$\underset{B, M, Q, \nu}{\text{minimize}} \eta(B, M, Q, \nu). \quad (2.7)$$

Because (2.5) and (2.6) are differentiable, the derivatives of  $\eta$  with respect to the parame-

ters can be easily obtained, as shown in the following:

$$\begin{aligned}\frac{d\eta}{dB} &= \sum_{i=1}^n -T_1 \frac{Qe^{-B(x_i-M)}(x_i-M)}{T_2}, \\ \frac{d\eta}{dM} &= \sum_{i=1}^n T_1 \frac{BQe^{-B(x_i-M)}}{T_2}, \\ \frac{d\eta}{dQ} &= \sum_{i=1}^n T_1 \frac{e^{-B(x_i-M)}}{T_2}, \\ \frac{d\eta}{d\nu} &= \sum_{i=1}^n -T_1 \frac{\ln(Qe^{-B(x_i-M)} + 1)}{\nu^2(Qe^{-B(x_i-M)} + 1)^{1/\nu}},\end{aligned}$$

where

$$\begin{aligned}T_1 &= 2(\hat{P}_X(x_i) - L(x_i)) \\ T_2 &= \nu(Qe^{-B(x_i-M)} + 1)^{1/\nu+1}.\end{aligned}$$

Therefore, a local minimum of (2.7) can be solved efficiently by any gradient descent optimization algorithms.

### 2.3.2 Parameter Initialization

The optimization problem described in (2.7) is non-convex, so in order to achieve a good local minimum (or even global minimum) of the objective function, the values of the parameters should be carefully initialized; i.e. determine  $B_0, M_0, Q_0, \nu_0$ , which are the initialization of the parameters for the gradient descent iterations. By looking at the structure of the GL function, we can see that parameter  $M$  determines the ‘‘center’’ of the GL curve; therefore, parameter  $M$ , should be close to the median of the sample values. We first arrive at:

$$M_0 = \hat{P}_X^{-1}(0.5) = x_{med}, \quad (2.8)$$

where  $x_{med}$  denotes the median value of the variable in the samples. From  $L(x_{med}) \approx \hat{P}_X(x_{med}) \approx 0.5$ , we have  $L(x_{med}) = \frac{1}{(1+Q_0e^{-B_0(x_{med}-x_{med})})^{1/\nu_0}} \approx 0.5$ , noting that we

replace  $M_0$  by  $x_{med}$  because of (2.8). We obtain:

$$\nu_0 = \log_2(1 + Q_0). \quad (2.9)$$

It is reasonable to assume that the minimum value in the samples will be scaled to a value close to 0.1, that is  $L(x_{min}) \approx \hat{P}_X(x_{min}) \approx 0.1$ , we have  $L(x_{min}) = \frac{1}{(1+Q_0e^{-B_0(x_{min}-x_{med})})^{1/\nu_0}} \approx 0.1$ , where  $x_{min}$  denotes the minimum value of the variable in the samples. We obtain:

$$B_0 = \frac{\ln((1 + Q_0)^{\log_2(10)} - 1) - \ln(Q_0)}{x_{med} - x_{min}}. \quad (2.10)$$

Now,  $\nu_0$  and  $B_0$  are dependent on  $Q_0$ . We further assume that the maximum value in the sample will be scaled to a value close to 0.9, that is  $L(x_{max}) \approx \hat{P}_X(x_{max}) \approx 0.9$ , thus  $L(x_{max}) = \frac{1}{(1+Q_0e^{-B_0(x_{max}-x_{med})})^{1/\nu_0}} \approx 0.9$ , where  $x_{max}$  denotes the maximum value of the variable in the samples. Combining (2.9) and (2.10), we obtain the following equation in terms of  $Q_0$ :

$$\frac{1}{1 + Q_0e^{(\ln((1+Q_0)^{\log_2(10)}-1)-\ln(Q_0))\frac{x_{max}-x_{med}}{x_{min}-x_{med}}}} = 0.9^{\log_2(1+Q_0)}, \quad (2.11)$$

and the most suitable value for  $Q_0$  is the root of equation (2.11). The root can be resolved numerically and quickly by using the Newton's method. With this initialization, we could find a set of parameters which make the GL function fit the ECDF well, as shown in Figure. 2.1.

### 2.3.3 Qualitative Comparisons of Data Scaling algorithms

In this section, we will intuitively and qualitatively discuss the scenarios where the GL algorithm is superior to the commonly used data scaling algorithms.

*The GL Algorithm is Robust to Outliers* During the data collection period, the data might be corrupted for various reasons; e.g., system error, human error, sample contamination, etc. Therefore, a data de-noising or outlier detection procedure may be necessary in the data

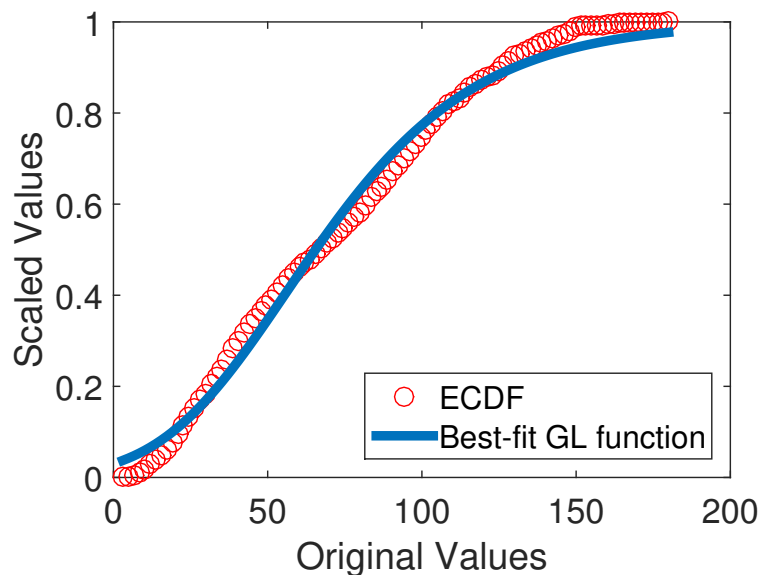


FIGURE 2.1: Fitting of the ECDF using the GL algorithm: an example showing the approximation of an ECDF using a generalized logistic (GL) function.

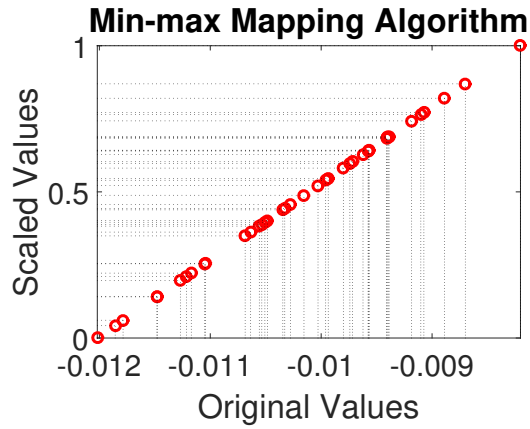
preprocessing step. The GL algorithm is intrinsically capable of handling situations where there are noisy samples and outliers in the samples. As Figure 2.2 shows, in the situation that there are no outliers in samples, all data scaling algorithms perform similarly. However, when an outlier exists in the data, as shown in Figure 2.3, the Min-max algorithm and the Z-score algorithm are affected by the outlier - the original values in the normal range are squeezed after the scaling. In contrast, the outlier's impact to the GL algorithm is neglectable, as shown in Figure 2.3c. Outliers are samples deviate strongly from the majority of (normal) samples, so the number of outliers will be always much smaller than the number of normal samples, and therefore, the contribution of outliers to the CDF of the samples is neglectable. However, outliers do not necessarily need to be the result of measurement errors, but may also occur due to variability, and represent completely valid instances. There are applications that are particularly concerned with such anomalies in the observations as they may carry valuable information about some rare modality of the processes responsible for its generation. For such applications, algorithms for outlier detection are utilized to interrogate the data and bring the focus to the rare signal in the data,

and our data preprocessing algorithm is inappropriate to use for such purposes. Nevertheless, regardless of the outliers' origin (error or variability), for the supervised task of classification, outliers are typically detrimental for classification accuracy, and their removal/correction is very welcome, if not necessary Acuna and Rodriguez (2004).

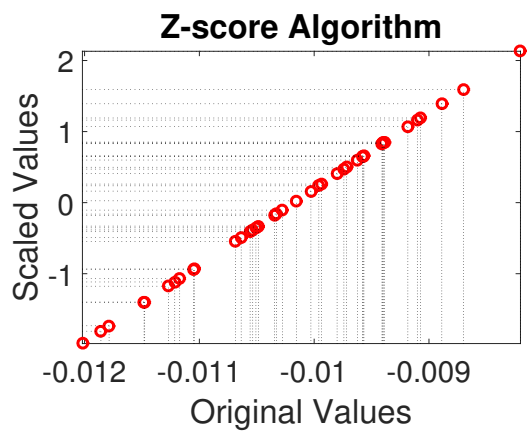
*The GL Algorithm can Improve Classification Accuracy* One of the complications which leads to poor classification accuracy is that the samples in different classes are dense and “crowded” near the decision boundary (otherwise, the accuracy would be expected to be high). Therefore, although in the training stage, the model can perfectly distinguish samples in different classes, in the testing stage, the model may make mistakes. Figure 2.4a (a) shows an artificially generated data of two groups (red v.s. blue), and we can imagine those samples are used to test the classifier. Although the two groups of data are separable, a trained classifier may make mistakes because these data are not seen in the training. One way to improve the classification in the test is to enlarge the separation the data from two groups near the decision boundary. The intuition is that if the separation of two groups is by a large margin, it allows a wider variety of decision boundaries to separate the data. Because the Min-max algorithm and the Z-score algorithm are linear mappings, after the data are scaled, their relative distance will not change (Figure 2.4 (b)&(c)). In contrast, the GL algorithm is a nonlinear mapping; it will enlarge the distance of the dense samples that are located near the decision boundary, and squeeze the samples that are located away from the decision boundary (Figure 2.4 (d)). This effect reduces the classifier's potential of making mistakes, thus improving the accuracy.

## Experiments and Results

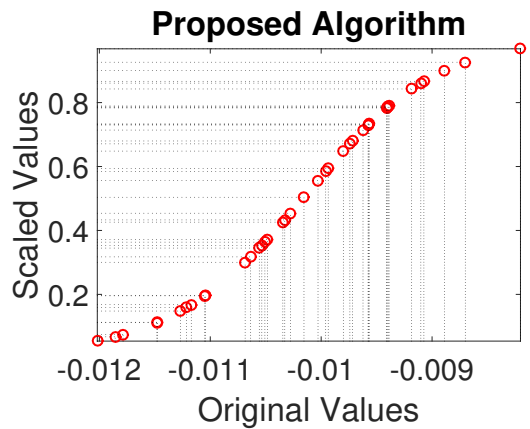
*Descriptions of Datasets* We have included 16 datasets in our experiments. The tasks associated with the datasets cover a broad variety of diagnostic/classification problems in biomedical research. The information of the datasets, including the number of samples,



(a)

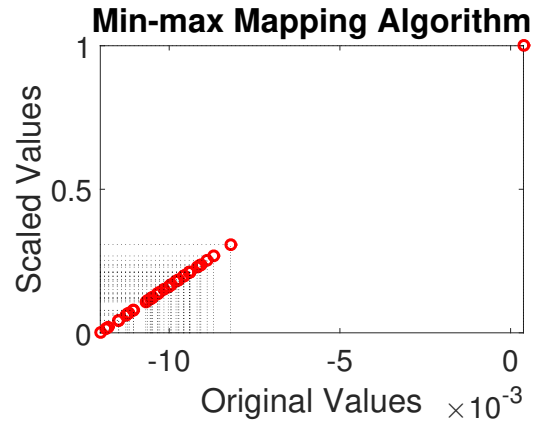


(b)

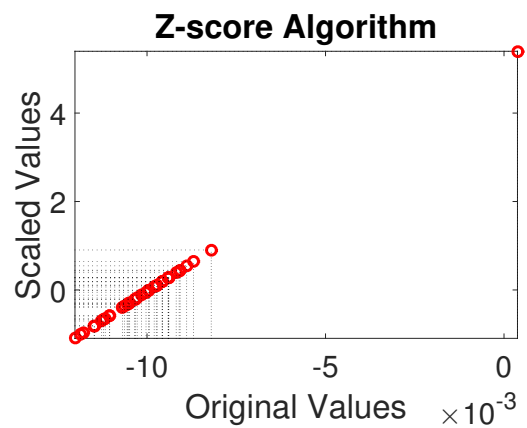


(c)

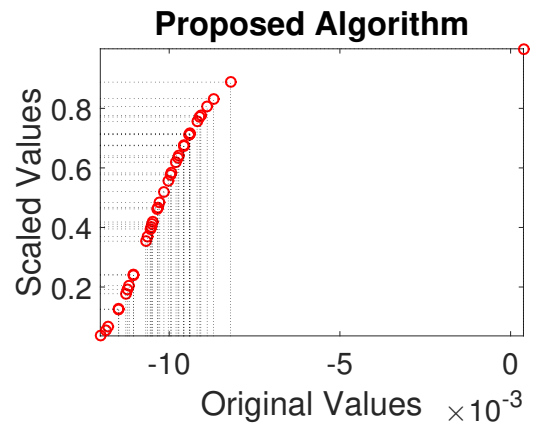
FIGURE 2.2: Behavior of data scaling algorithms without outliers. When there is no outlier in the data, the behavior of the Min-max algorithm, Z-score algorithm and the GL algorithm is very similar.



(a)



(b)



(c)

FIGURE 2.3: Behavior of data scaling algorithms with outliers. When there is an outlier in the data, the behaviors of the Min-max algorithm and Z-score algorithm are significantly affected, but the impact of the outlier on the GL algorithm is neglectable.

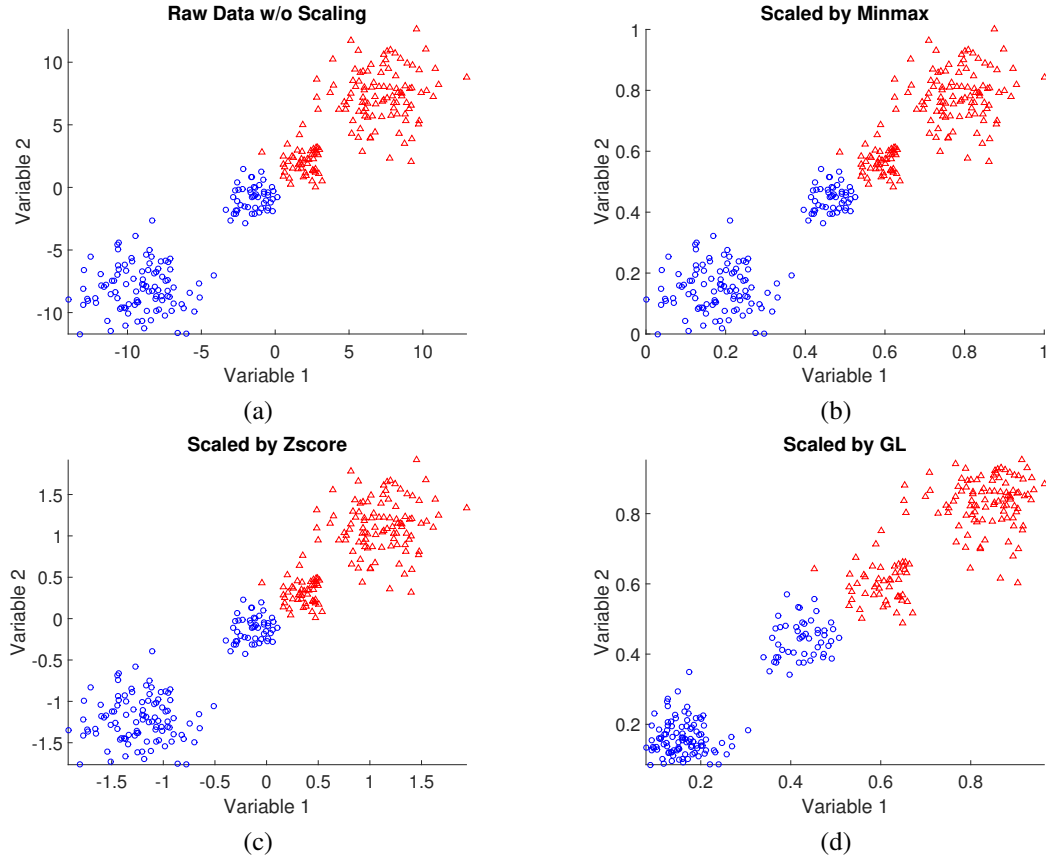


FIGURE 2.4: An 2D illustration on how the GL algorithm can affect the classification accuracy. (a) raw data without scaling; (b) data scaled by the Min-max algorithm; (c) data scaled by the Z-score algorithm; (d) data scaled by the GL algorithm.

variable types, and tasks, are summarized in Table 2.1. Among them, *LSVT*, *Pima Indian diabetes*, *Parkinsons*, *Wdbc*, *Breast tissue*, and *Indian liver* were downloaded from the UCI dataset repository (<https://archive.ics.uci.edu/ml/datasets.html>). These 6 datasets were selected because the majority of their variables are continuous, so that the data scaling algorithms could be applied (non-continuous variables were deleted). If a dataset was originally associated with a multiclass classification task, we will formulate a binary classification task as one-class-vs-others. The datasets, *Breast cancer*, *Colon cancer*, *Lung cancer*, *Prostate cancer*, and *Myeloma* were made available by *Statnikov et al.* Statnikov et al. (2005), and we downloaded the datasets from the supplementary material website (<http://www.gems-system.org/>). The datasets *DL-*

*BCL* and *Leukemia* were downloaded from the Kent Ridge Bio-medical Dataset Repository (<http://datam.i2r.a-star.edu.sg/datasets/krbd>); we removed the variables with missing values in the *DLBCL* dataset, so 715 variables were used in our experiments. The datasets *GSE 25869*, *GSE 27899IL*, and *GSE 29490*, were downloaded from the Gene Expression Omnibus Repository Edgar et al. (2002). We converted the datasets to *.mat* format, and made them available to the public; please refer to Section *Availability of data and material* for details.

*Evaluation Methods* To assess how different data scaling algorithms affect classification performances, we used Logistic Regression (LR) and Support Vector Machine (SVM) as the classification models. These two classification models have been used extensively in biological and medical research due to their simplicity and accessibility. The program code of the experiments was implemented in MATLAB 8.4. The results were obtained using 5-fold cross-validations. One of the performance metrics we used was the area under the receiver operation characteristic curve (AUROC), which has been commonly used for binary classification performance evaluations; one of the advantages of AUROC is that its value does not depend on a classification score threshold. To have a more complete comparison of different data scaling methods and classification models, we also used accuracy (proportion of correct classifications). The threshold we used to determine the class labels (and thus, the accuracy) of the testing set samples was obtained by selecting a score which could maximize the accuracy in the training set; if multiple, or a range of scores could achieve the maximum accuracy, we would select the minimum. The mean value and 95% confidence interval of the AUROC of each binary classification task can be found in Table 2.3, and the mean value and 95% confidence interval of the proportion of correct classifications can be found in Table 2.2. Due to the large number of variables, the AUROC's and accuracies of datasets *GSE 25869*, *GSE 27899IL*, and *GSE 29490* on Logistic Regression model were not available.

*Results and Discussions* In most of the classification tasks, models learned with unscaled data have the worst performances. This is consistent with our expectations. In general, an appropriate data processing step (i.e., data scaling) is able to improve the accuracy of a model. Comparing the GL algorithm to the Z-score algorithm and the Min-max algorithm, in most tasks, models learned with the data scaled by the GL algorithm achieved the best average AUROC's and the best average accuracies. Specifically, in the experiments, out of the 29 task-model cases (16 tasks; 2 models per task, but LR was not available in 3 tasks), the GL algorithm achieve the best AUROC's in 27 cases and the best accuracies in 25 cases. The advantage of the GL algorithm was more notable in datasets with a small number of samples, such as *colon*, *lung*, and *prostate*, in which the existence of outliers may significantly affects the model performance. For example, in the colon cancer diagnostic task, while using the SVM classifier, the model learned using GL scaled data achieved a 0.822 AUROC, while the best AUROC achieved by the SVM classifier from other data scaling methods was 0.725; it was a 13.4% of improvement. The improvements of AUROC using the data scaled by the GL algorithm were less notable in the tasks *Parkinsons*, *Wdbc*, *Indian liver*, and *Pima Indians Diabetes*. One of the reasons was that the number of samples in those data sets was relatively large, so the negative effects of outliers became less significant; another possible reason was that before the contributors uploaded the data set, they might have performed a preprocessing step to correct/remove abnormal samples. It is worthwhile to point out that, in three task-model cases (i.e., RL and SVM in the *Parkinsons* task, and SVM in the *Pima Indians Diabetes* task), although the GL algorithm achieved the best AUROC's, it did not achieve the best accuracies. That might be due to the the threshold selection rule in our experiments; while the AUROC's of different task-model cases were close, the ranking of the accuracies would be very sensitive to the selected threshold.

## 2.4 Conclusion

In this article, we present a simple yet effective data scaling algorithm, the GL algorithm, to scale data to an appropriate interval for diagnostic and classification modeling. In the GL algorithm, the values of a variable are scaled in the  $(0,1)$  interval using the cumulative density function of the variable. Since obtaining the functional expression of the CDF is difficult, a generalized logistic GL function is used to fit the empirical cumulative distribution function, and the optimized GL function is used for data scaling. The GL algorithm is intrinsically robust to outliers, so it is particularly suitable for diagnostic/classification models in clinical/medical applications, where the number of samples is usually small; it scales the data in a nonlinear fashion, which leads to improvement of accuracy. Experimental results show that models learned using data scaled by the GL algorithm generally outperform the ones using the Min-max algorithm and the Z-score algorithm, which are currently the most commonly used data scaling algorithms.

Table 2.1: Summary of datasets used in experiments (sorted by the no. of subjects in ascending order)

Dataset	No. of subjects (pos/neg)	Var. type	No. of var.	Task
<i>GSE 27899IL</i>	10/10	DNA methylation	27578	diagnose ulcerative colitis
<i>Prostate cancer</i>	14/9	microarray gene expression	15009	diagnose prostate cancer
<i>Colon cancer</i>	15/11	microarray gene expression	15009	diagnose colon cancer
<i>Lung cancer</i>	20/7	microarray gene expression	15009	diagnose lung cancer
<i>Breast cancer</i>	17/15	microarray gene expression	15009	diagnose breast cancer
<i>Leukemia</i>	11/27	microarray gene expression	7129	diagnose leukemia
<i>GSE 29490</i>	20/7	DNA methylation	26916	diagnose colorectal carcinoma
<i>GSE 25869</i>	14/9	DNA methylation	27570	diagnose gastric cancer
<i>Breast tissue</i>	21/85	impedance measurements	9	diagnose breast tumor
<i>LSVT</i>	42/84	wavelet and frequency based measurements	310	assessment of treatments in Parkinson
<i>DLBCL</i>	88/72	microarray gene expression	715	diagnose DLBCL
<i>Myeloma</i>	137/36	microarray gene expression	12625	diagnose bone lesions
<i>Parkinsons</i>	147/48	vocal based measurements	22	diagnose Parkinson disease
<i>Wdbc</i>	212/357	nuclear feature from image	30	diagnose breast tumor
<i>Indian liver</i>	414/165	biochemistry based measurements	9	diagnose liver disease
<i>Pima Indians diabetes</i>	268/500	clinical measurements	8	diagnose diabetes

Table 2.2: Results of 16 datasets (16 binary classification tasks) using different data scaling algorithms and classification models. The performances are measured by the average proportion of correct classification in 5-fold cross-validations. The means and 95% confidence intervals are included. Column names: None - no data scaling; Minmax - Min-max algorithm; Zscore - Z-score algorithm; GL - GL algorithm. Best performances are emphasized in bold

dataset	Method	None	Minmax	Zscore	GL
GSE27899IL	LR	NA $\pm$ NA	NA $\pm$ NA	NA $\pm$ NA	NA $\pm$ NA
	SVM	0.770 $\pm$ 0.054	<b>0.780 <math>\pm</math> 0.134</b>	<b>0.780 <math>\pm</math> 0.134</b>	<b>0.780 <math>\pm</math> 0.134</b>
Prostate Cancer	LR	0.609 $\pm$ 0.000	0.757 $\pm$ 0.132	0.722 $\pm$ 0.078	<b>0.765 <math>\pm</math> 0.100</b>
	SVM	0.635 $\pm$ 0.078	0.748 $\pm$ 0.156	0.748 $\pm$ 0.156	<b>0.835 <math>\pm</math> 0.114</b>
Colon Cancer	LR	0.577 $\pm$ 0.000	0.877 $\pm$ 0.064	0.877 $\pm$ 0.064	<b>0.923 <math>\pm</math> 0.054</b>
	SVM	0.677 $\pm$ 0.178	0.900 $\pm$ 0.042	0.915 $\pm$ 0.034	<b>0.946 <math>\pm</math> 0.042</b>
Lung Cancer	LR	0.741 $\pm$ 0.000	0.859 $\pm$ 0.062	0.852 $\pm$ 0.052	<b>0.896 <math>\pm</math> 0.062</b>
	SVM	0.778 $\pm$ 0.052	0.859 $\pm$ 0.034	0.859 $\pm$ 0.034	<b>0.867 <math>\pm</math> 0.040</b>
Breast Cancer	LR	0.773 $\pm$ 0.000	0.918 $\pm$ 0.040	<b>0.955 <math>\pm</math> 0.000</b>	<b>0.955 <math>\pm</math> 0.000</b>
	SVM	0.827 $\pm$ 0.040	0.909 $\pm$ 0.000	0.909 $\pm$ 0.000	<b>0.936 <math>\pm</math> 0.050</b>
Leukemia	LR	0.710 $\pm$ 0.000	0.956 $\pm$ 0.030	0.965 $\pm$ 0.030	<b>1.000 <math>\pm</math> 0.000</b>
	SVM	0.939 $\pm$ 0.030	0.965 $\pm$ 0.030	0.965 $\pm$ 0.030	<b>1.000 <math>\pm</math> 0.000</b>
GSE29490	LR	NA $\pm$ NA	NA $\pm$ NA	NA $\pm$ NA	NA $\pm$ NA
	SVM	0.942 $\pm$ 0.034	0.954 $\pm$ 0.034	0.958 $\pm$ 0.034	<b>0.979 <math>\pm</math> 0.000</b>
GSE25869	LR	NA $\pm$ NA	NA $\pm$ NA	NA $\pm$ NA	NA $\pm$ NA
	SVM	0.891 $\pm$ 0.038	0.891 $\pm$ 0.044	0.894 $\pm$ 0.034	<b>0.897 <math>\pm</math> 0.034</b>
Breast tissue	LR	0.778 $\pm$ 0.016	0.930 $\pm$ 0.010	<b>0.930 <math>\pm</math> 0.016</b>	0.927 $\pm$ 0.016
	SVM	0.681 $\pm$ 0.220	0.932 $\pm$ 0.024	0.926 $\pm$ 0.020	<b>0.942 <math>\pm</math> 0.008</b>
LSVT	LR	0.500 $\pm$ 0.000	0.870 $\pm$ 0.012	0.824 $\pm$ 0.038	<b>0.915 <math>\pm</math> 0.002</b>
	SVM	0.500 $\pm$ 0.000	0.873 $\pm$ 0.036	0.858 $\pm$ 0.036	<b>0.908 <math>\pm</math> 0.006</b>
DLBCL	LR	0.567 $\pm$ 0.014	0.571 $\pm$ 0.014	0.579 $\pm$ 0.032	<b>0.602 <math>\pm</math> 0.074</b>
	SVM	0.594 $\pm$ 0.082	0.592 $\pm$ 0.064	0.585 $\pm$ 0.044	<b>0.600 <math>\pm</math> 0.100</b>
Myeloma	LR	0.792 $\pm$ 0.000	0.805 $\pm$ 0.020	0.804 $\pm$ 0.018	<b>0.805 <math>\pm</math> 0.026</b>
	SVM	0.794 $\pm$ 0.006	0.809 $\pm$ 0.014	0.807 $\pm$ 0.026	<b>0.813 <math>\pm</math> 0.020</b>
Parkinsons	LR	0.865 $\pm$ 0.006	<b>0.894 <math>\pm</math> 0.022</b>	0.891 $\pm$ 0.016	0.868 $\pm$ 0.006
	SVM	0.880 $\pm$ 0.016	<b>0.884 <math>\pm</math> 0.006</b>	0.877 $\pm$ 0.020	0.868 $\pm$ 0.016
Wdbc	LR	0.878 $\pm$ 0.002	0.965 $\pm$ 0.010	0.963 $\pm$ 0.012	<b>0.971 <math>\pm</math> 0.012</b>
	SVM	0.960 $\pm$ 0.010	0.979 $\pm$ 0.004	0.976 $\pm$ 0.002	<b>0.980 <math>\pm</math> 0.008</b>
Indian Liver	LR	0.716 $\pm$ 0.002	0.727 $\pm$ 0.014	0.733 $\pm$ 0.008	<b>0.736 <math>\pm</math> 0.006</b>
	SVM	0.719 $\pm$ 0.006	0.720 $\pm$ 0.014	0.718 $\pm$ 0.010	<b>0.720 <math>\pm</math> 0.008</b>
Pima Indians Diabetes	LR	0.490 $\pm$ 0.070	0.738 $\pm$ 0.010	0.738 $\pm$ 0.010	<b>0.740 <math>\pm</math> 0.012</b>
	SVM	0.734 $\pm$ 0.052	<b>0.765 <math>\pm</math> 0.008</b>	0.753 $\pm$ 0.040	0.748 $\pm$ 0.034

Table 2.3: Results of 16 datasets (16 binary classification tasks) using different data scaling algorithms and classification models. The performances are measured by the average Area Under the ROC in 5-fold cross-validations. The means and 95% confidence intervals are included. Column names: None - no data scaling; Minmax - Min-max algorithm; Z-score - Z-score algorithm; GL - GL algorithm. Best performances are emphasized in bold

dataset	Method	None	Minmax	Zscore	GL
GSE27899IL	LR	NA $\pm$ NA	NA $\pm$ NA	NA $\pm$ NA	NA $\pm$ NA
	SVM	0.768 $\pm$ 0.104	0.814 $\pm$ 0.084	0.814 $\pm$ 0.074	<b>0.824 <math>\pm</math> 0.058</b>
Prostate Cancer	LR	0.464 $\pm$ 0.000	0.749 $\pm$ 0.130	0.689 $\pm$ 0.156	<b>0.761 <math>\pm</math> 0.108</b>
	SVM	0.573 $\pm$ 0.198	0.725 $\pm$ 0.232	0.713 $\pm$ 0.244	<b>0.822 <math>\pm</math> 0.194</b>
Colon Cancer	LR	0.500 $\pm$ 0.000	0.895 $\pm$ 0.092	0.892 $\pm$ 0.082	<b>0.962 <math>\pm</math> 0.046</b>
	SVM	0.670 $\pm$ 0.184	0.940 $\pm$ 0.058	0.937 $\pm$ 0.050	<b>0.981 <math>\pm</math> 0.020</b>
Lung Cancer	LR	0.450 $\pm$ 0.000	0.839 $\pm$ 0.096	0.834 $\pm$ 0.108	<b>0.890 <math>\pm</math> 0.050</b>
	SVM	0.397 $\pm$ 0.274	0.716 $\pm$ 0.136	0.710 $\pm$ 0.152	<b>0.774 <math>\pm</math> 0.182</b>
Breast Cancer	LR	0.324 $\pm$ 0.000	0.809 $\pm$ 0.038	<b>0.821 <math>\pm</math> 0.020</b>	0.819 $\pm$ 0.022
	SVM	0.708 $\pm$ 0.158	0.793 $\pm$ 0.052	0.795 $\pm$ 0.042	<b>0.812 <math>\pm</math> 0.038</b>
Leukemia	LR	0.500 $\pm$ 0.000	0.988 $\pm$ 0.014	0.990 $\pm$ 0.006	<b>1.000 <math>\pm</math> 0.000</b>
	SVM	0.935 $\pm$ 0.034	0.992 $\pm$ 0.010	0.991 $\pm$ 0.008	<b>1.000 <math>\pm</math> 0.000</b>
GSE29490	LR	NA $\pm$ NA	NA $\pm$ NA	NA $\pm$ NA	NA $\pm$ NA
	SVM	0.983 $\pm$ 0.012	0.984 $\pm$ 0.034	0.985 $\pm$ 0.034	<b>0.994 <math>\pm</math> 0.004</b>
GSE25869	LR	NA $\pm$ NA	NA $\pm$ NA	NA $\pm$ NA	NA $\pm$ NA
	SVM	0.935 $\pm$ 0.024	0.937 $\pm$ 0.020	0.938 $\pm$ 0.016	<b>0.943 <math>\pm</math> 0.014</b>
Breast tissue	LR	0.520 $\pm$ 0.006	0.961 $\pm$ 0.032	<b>0.961 <math>\pm</math> 0.044</b>	0.940 $\pm$ 0.054
	SVM	0.713 $\pm$ 0.108	0.968 $\pm$ 0.006	0.970 $\pm$ 0.014	<b>0.972 <math>\pm</math> 0.010</b>
LSVT	LR	0.500 $\pm$ 0.000	0.875 $\pm$ 0.008	0.846 $\pm$ 0.022	<b>0.921 <math>\pm</math> 0.012</b>
	SVM	0.500 $\pm$ 0.000	0.879 $\pm$ 0.012	0.863 $\pm$ 0.014	<b>0.919 <math>\pm</math> 0.020</b>
DLBCL	LR	0.601 $\pm$ 0.038	0.608 $\pm$ 0.038	0.610 $\pm$ 0.048	<b>0.660 <math>\pm</math> 0.062</b>
	SVM	0.616 $\pm$ 0.050	0.622 $\pm$ 0.052	0.619 $\pm$ 0.052	<b>0.654 <math>\pm</math> 0.054</b>
Myeloma	LR	0.500 $\pm$ 0.000	0.729 $\pm$ 0.044	0.739 $\pm$ 0.072	<b>0.746 <math>\pm</math> 0.038</b>
	SVM	0.573 $\pm$ 0.098	0.748 $\pm$ 0.052	0.747 $\pm$ 0.054	<b>0.750 <math>\pm</math> 0.054</b>
Parkinsons	LR	0.875 $\pm$ 0.012	0.896 $\pm$ 0.054	0.893 $\pm$ 0.058	<b>0.906 <math>\pm</math> 0.048</b>
	SVM	0.882 $\pm$ 0.010	0.875 $\pm$ 0.010	0.885 $\pm$ 0.024	<b>0.891 <math>\pm</math> 0.018</b>
Wdbc	LR	0.942 $\pm$ 0.002	0.982 $\pm$ 0.004	0.978 $\pm$ 0.006	<b>0.993 <math>\pm</math> 0.004</b>
	SVM	0.990 $\pm$ 0.002	0.994 $\pm$ 0.002	0.993 $\pm$ 0.004	<b>0.995 <math>\pm</math> 0.000</b>
Indian Liver	LR	0.680 $\pm$ 0.002	0.743 $\pm$ 0.008	0.742 $\pm$ 0.008	<b>0.746 <math>\pm</math> 0.010</b>
	SVM	0.636 $\pm$ 0.068	0.696 $\pm$ 0.008	0.692 $\pm$ 0.034	<b>0.695 <math>\pm</math> 0.008</b>
Pima Indians Diabetes	LR	0.604 $\pm$ 0.004	0.827 $\pm$ 0.004	0.827 $\pm$ 0.004	<b>0.834 <math>\pm</math> 0.006</b>
	SVM	0.826 $\pm$ 0.004	0.828 $\pm$ 0.006	0.828 $\pm$ 0.006	<b>0.834 <math>\pm</math> 0.006</b>

# CHAPTER 3

## LEARNING A DYNAMIC-BASED REPRESENTATION FOR MULTIVARIATE BIOMARKER TIME SERIES CLASSIFICATIONS

### 3.1 Introduction

Time series in healthcare practices and biomedical research are typically multivariate, i.e. multiple biomarkers are observed simultaneously at a time. However, they tend to be short, noisy, unaligned, irregularly sampled, partially observed, and with only limited samples. These imperfections pose a challenge for mining information from data. In this work, we propose to use dynamic-based representations to present such imperfect multivariate time series. Specifically, we propose an approach to learn a corresponding Linear Dynamical System (LDS) for a multivariate time series example and use the set of system parameters as a representation for that example. Such a representation is able to capture interactions of different variables and provide a unified view of multivariate time series with different lengths, different missingness mechanisms, and different starting points. Other techniques are then used to mine useful information and perform learning tasks based on the new rep-

resentation. For example, we use support vector machine classification models with LDS kernels in time series classification tasks. To evaluate the effectiveness of the proposed approach, we conducted experiments on both synthetic data sets and real-life datasets. The results in synthetic datasets demonstrated that the proposed approach could correctly learn the similarities of underlying linear dynamical systems. Our real-life data sets included human influenza A (H3N2), Rhinovirus (HRV), and respiratory syncytial virus (RSV) gene expression time series. The accuracies in the leave-one-out symptomatic/asymptomatic diagnostic tasks showed that our approach outperformed three baseline algorithms. Moreover, in experiments where various levels of imperfections were imposed on the H3N2 dataset, the accuracies of other baseline methods degraded significantly, but the accuracy of our approach remained high.

## 3.2 Background

### *3.2.1 Time series data in biomedical research*

Data mining and machine learning techniques have great potential to improve healthcare quality by allowing effective knowledge extraction from observed data. Among different forms of data, time series data are very important for both medical research and clinical practice. A multivariate time series (MTS) is a sequence of observations on multiple variables in time. An efficient data mining model for time series analysis would capture the dynamics of the ongoing disease progression and interactions among biological system components and thus lead to an accurate forecasting of health state changes Omranian et al. (2015). More advanced methodologies can reveal hidden phenomena in the biological/physiological mechanism of a subject and provide the insights for proper treatments Henn et al. (2013).

### 3.2.2 Imperfections in biomedical time series data

To evaluate the health of a patient, we need to measure quantities related to the patient's physiological states in time. Some of the measurements can be made frequently; for example, heart rate, blood pressure, oxygen level, ECG, EEG, etc. Other measurements are made infrequently, for not only the cost but also the burdens to patients. Examples of these measurements include laboratory tests, CT images, MR Images, gene expressions etc. Many techniques (e.g. Fourier transform, Wavelet transform, etc. Rangayyan (2015)) have been developed to analyze frequently made measurements where data points are sufficient, there are no missing values, and samples are regularly spaced. In most cases, these techniques are not applicable to time series measurements with imperfections; i.e. short, noisy, unaligned, irregularly sampled, partially observed, and with limited samples. A typical example of an imperfect time series is the blood cell count (e.g. white blood cell, cytokine, etc.) time series. Usually, blood cell counts are measured through time; however, they are measured *infrequently*, such that typically, a very *limited number* of samples are available. In addition, such measurements are often *noisy* due to uncontrollable factors. Sample contamination and other human errors may also cause missing measurements of some biomarkers, so the data are *partially observed* (only a subset of variables is observed at a time point). Clinical measurements are usually made by humans (nurses), so the data are *irregularly spaced*. The onset of the disease is often unknown, so the time series data are usually *unaligned*. In emergency situations, decisions should be made in a short time, so the time series may be very *short*. However, these clinical data are closely related to patients' health states; therefore, developing methodologies to efficiently analyze them could tremendously help healthcare practitioners to better serve patients.

### 3.2.3 Using a linear dynamical system to represent a multivariate time series

In this section, we introduce briefly the basics of a linear dynamical system (LDS) and how a multivariate time series (MTS) can be represented by an LDS.

In our work, we consider the LDS in the following form:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t), \quad (3.1)$$

where  $t \in \mathbb{R}$  denotes time,  $\mathbf{x}(t) \in \mathbb{R}^d$  denotes an  $d$ -dimensional state vector,  $\mathbf{A} \in \mathbb{R}^{d \times d}$  is the dynamic matrix, and  $\dot{\mathbf{x}}(t)$  is the derivative of the state vector with respect to time. If an MTS consists of samples taken from the state trajectories generated by an LDS, then we can create a connection between the MTS and LDS. Therefore, we can represent an MTS by the parameters of its corresponding LDS. We argue that we can use the initial condition,  $\mathbf{x}_0$ , and dynamics matrix,  $\mathbf{A}$ , to represent an MTS, for the trajectories of the states are intrinsically determined by these two parameters. In the case that the MTS's are not aligned, the initial conditions of the LDS are unknown, and we can then use the dynamics matrix,  $\mathbf{A}$ , to represent an MTS. For example, in the time series classification tasks in our conducted experiments, we firstly learned an LDS (represented by a parameter set  $\{\mathbf{x}_0, \mathbf{A}\}$ ) from each MTS example, and then used the learned parameters as representations for model training and inference.

### 3.2.4 Contributions

In this work, we propose a method to learn informative representations from imperfect MTS's. The proposed method is able to overcome several important challenges of the MTS data in biomedical/healthcare researches and applications.

*Contribution 1: the proposed approach learns an efficient representation from an MTS which is short and with a limited number of irregularly spaced samples.* Many physiological variables, such as heart rate, blood pressure, and electrocardiogram, can be measured by specific devices. Measurements of these variables can be made noninvasively by ma-

chines at small costs; therefore, samples of the variables are usually acquired regularly and frequently. However, there are variables that are also very costly and need to be acquired by humans; as a result, only a *limited number* of samples are available. Moreover, these samples are *irregularly spaced* due to the schedule of the humans. Existing widely adapted time series analysis techniques usually require the samples of the variables to be acquired frequently and regularly. Inspired by Candes et al. (2006), which showed that stable signals can be recovered from incomplete and inaccurate measurements, the proposed method employs an optimization-based approach to learn a continuous-time, time-invariant linear dynamical system to fit an MTS which has irregularly spaced samples; in addition, the LDS is regularized by both smoothness and ridge loss to ensure that the learned model overcomes the problem of overfitting.

*Contribution 2: The proposed approach learns an efficient representation from an MTS with noisy, partially observed, and unaligned samples.* Because of the limitations of the acquisition techniques and laboratory equipments, measurements tend to be *noisy*; in addition, contaminations and human errors may cause missing measurements, so the variables at a time point may be *partially observed*. The multivariate time series of different subjects/patients are usually *unaligned*; this is because the onsets of the conditions are usually unknown, hospitalization times are different, and observation lengths are different. The proposed approach uses Prediction Error Methods (PEM) to fit the state trajectories of an LDS to the MTS and simultaneously estimate the initial condition (first observation of the time series), which is corrupted by noise. There are many interpolation methods proposed to estimate the missing values in time series data; however, if the underlying missing mechanism is unknown, interpolations would introduce bias. In our method, when fitting the state trajectories to an MTS, we only consider the available values, thus, this approach does not suffer from the bias introduced by interpolations. We assume that the LDS's are *time-invariant*; therefore, even though the onsets of the conditions are unknown, the LDS's still represent the dynamics of the time series; also, because of the time-invariances

of the LDS's, the observation lengths of the time series will have no impact on the learning. Because an LDS can be represented by a dynamics matrix, and the dimensions of the dynamics matrix are determined by the number of variables in the time series; therefore, an LDS is a unified representation of time series with various lengths.

### 3.3 Related Work

#### 3.3.1 *Related work in learning LDS*

Linear dynamical systems have been extensively used in various fields, including engineering, medicine, economics, etc. Learning an LDS from data is a long-lasting research topic. An Expectation-Maximization approach was proposed to learn the parameters of an LDS from data, and the relationships among LDS, factor analysis, and hidden Markov models were studied Ghahramani and Hinton (1996); Shumway and Stoffer (1982). Subspace methods were proposed to learn the LDS parameters by fitting the state observations Katayama (2006); Van Overschee and De Moor (2012). However, the above methods assume that the states are completely observed (no missing observations) and samples are regularly spaced; moreover, when the observation durations are short, the parameters learned using the above methods are susceptible to overfitting. Recently, regularization frameworks were proposed to address this problem. A framework was developed to regularize the largest eigenvalue of the dynamics matrix  $\mathbf{A}$  and learn the parameters using a spectral algorithm Boots et al. (2007). More recently, an L1-regularization framework Städler et al. (2013), a low-rank regularization framework Liu and Hauskrecht (2015), and a matrix factorization based framework with regularizations Liu and Hauskrecht (2016) were also proposed. While these frameworks overcome the overfitting problem and are able to learn an LDS efficiently from fully observed state trajectory samples, they do not explicitly handle the situations in which states are partially observed. Moreover, since they are based on the formulation of a discrete-time LDS, they require that the state trajectory

samples are regularly spaced. In this work, we proposed a learning algorithm to learn the dynamics matrix,  $\mathbf{A}$ , and the initial condition,  $\mathbf{x}_0$ , from an imperfect MTS (i.e. state trajectory samples), based on the formulation of a continuous-time LDS. Therefore, our learning algorithm does not require that the observations are regularly spaced.

### 3.3.2 *Related work in learning representation from MTS's*

Learning representations from data is crucial for many machine learning tasks Bengio et al. (2013). Time series Representation Learning does not only aim to reduce the storage of a large amount of time series data, but more importantly, it aims to extract informative features for classification, prediction, or clustering. The Discrete Fourier Transform (DFT) is one of the most well-known representations for time series. The first few coefficients of the DFT were proposed to represent a time series and showed promising results in searching and indexing time series in databases Agrawal et al. (1993). The Discrete Wavelet Transform (DWT) is proposed as a good alternative to the DFT Chan and Fu (1999), for the DWT is able to capture both the global and local shapes. However, these two representations require a rigorous sampling rate and completed observations. Piecewise Aggregate Approximation (PAA) Keogh et al. (2001) is an efficient representation for long-duration time series. Symbolic Aggregate Approximation (SAX) Lin et al. (2003) is a representation based on PAA; it is obtained by discretizing the PAA coefficients of a time series into some predefined symbols. The aim of both PAA and SAX is to reduce the length of a long time series, so they are not applicable to short time series. The concept of shapelet and a time series representation called shapelet transform were proposed Ye and Keogh (2009). A generalized shapelet-based method on multivariate time series showed advantages in early classification tasks Ghalwash and Obradovic (2012). The shapelet-based representations are able to capture the class-specific local features of time series; however, they are not applicable to a time series that is partially observed or irregularly sampled. Discrete-time Linear Dynamical Systems (DTLDS's) were proposed as kernels for a Sup-

port Vector Machine classifier in the tasks of MTS classification, and demonstrated that DTLDS's were efficient as representations for MTS's Borgwardt et al. (2006); however, the learned DTLDS's were susceptible to overfitting due to the lack of regularization, and are only applicable to MTS's with fixed lengths, due to the constraints in the learning algorithm.

## 3.4 Methods

In this section, we describe our approach to learn an LDS from an imperfect multivariate time series and use the LDS-based representation for the multivariate time series classification problem using a kernel support vector machine.

### 3.4.1 Notations

In this paper, scalars are denoted by lowercase alphabets (e.g.,  $t$ ). Vectors are represented by boldface alphabets (e.g.,  $\mathbf{x}$ ). Matrices are represented by boldface uppercase alphabets (e.g.,  $\mathbf{A}$ ). The  $(i, j)^{th}$  element of a matrix,  $\mathbf{A}$ , is denoted by its lowercase alphabet with a subscript, namely  $a_{ij}$ . The identity matrix is denoted by  $\mathbf{I}$  with suitable dimensions in equations. We list the main symbols in Table 3.1.

### 3.4.2 Learning an LDS from an imperfect MTS

The LDS in the form of (3.1) is also known as the first order continuous-time autoregressive (AR) model. A general framework to estimate the higher-order continuous-time AR model was proposed in Harvey and Stock (1985); however, this framework lacked regularizations, so the learned model is highly susceptible to overfitting; more importantly, this general framework is not applicable to the imperfect MTS for which our algorithm is proposed.

Using the Euler forward method, (3.1) can be approximated by

$$\frac{\mathbf{x}(t+h) - \mathbf{x}(t)}{h} = \mathbf{A}\mathbf{x}(t), \quad (3.2)$$

Notation	Definition
$m$	The number of time points in a time series is $m + 1$
$t_i$	$i = 0, 1, 2, \dots, m$ , the time stamp of the $i$ -th time point
$d$	The Number of biomarkers measured at a time
$h$	A small time increment
$n_i$	The number of $h$ 's in between $t_0$ and $t_i$
$\mathbf{x}(t_i) \in \mathbb{R}^d$	The $i$ -th biomarker measurement vector in a time series
$\hat{\mathbf{x}}(t_i) \in \mathbb{R}^d$	The approximation of $\mathbf{x}(t_i)$
$\mathbf{o}_i \in \{0, 1\}^d$	The encoding of observed measurement at the $i$ -th time point
$\mathbf{x}_0 \in \mathbb{R}^d$	Initial condition of a time series
$\mathbf{A} \in \mathbb{R}^{d \times d}$	Dynamics matrix
$\mathbf{O}_i \in \mathbb{R}^{d \times d}$	A diagonal matrix with $\mathbf{o}_i$ as the diagonal

Table 3.1: Notations and definitions

where  $h$  is a small time increment. The state vector  $h$  units ahead of current time  $t$  is approximated by

$$\mathbf{x}(t + h) = (\mathbf{I} + h\mathbf{A})\mathbf{x}(t). \quad (3.3)$$

Let's say in an MTS, there are  $m + 1$  samples  $\{\mathbf{x}(t_0), \mathbf{x}(t_1), \dots, \mathbf{x}(t_m)\}$  which are taken from the state trajectories of an LDS at time point  $\{t_0, t_1, t_2, \dots, t_m\}$ .  $\mathbf{x}(t_i) \in \mathbb{R}^d$  is a state vector consists of  $d$  biomarker measurements at time  $t_i$ . Because  $h$  is small, we can approximate the future time point as the current time point with integer multiples of  $h$ 's ahead. Namely,

$$t_i \simeq t_0 + n_i \cdot h, \quad i = 1, 2, \dots, m$$

where  $\{n_i | n_i = \lceil (t_i - t_0)/h \rceil, i = 1, 2, \dots, m\}$  are positive integers. If the initial state vector is known, then the state vector at time  $t_i$  can be approximated by

$$\hat{\mathbf{x}}(t_i) = \mathbf{x}(t_0 + n_i \cdot h) = (\mathbf{I} + h\mathbf{A})^{n_i} \mathbf{x}(t_0) \quad (3.4)$$

and the squared error of the  $i$ th approximation and the sample can be computed as

$$\begin{aligned}\|\mathbf{e}_i\|^2 &= \|\hat{\mathbf{x}}(t_i) - \mathbf{x}(t_i)\|_2^2 \\ &= \|(\mathbf{I} + h\mathbf{A})^{n_i} \mathbf{x}(t_0) - \mathbf{x}(t_i)\|_2^2,\end{aligned}\tag{3.5}$$

where  $\|\cdot\|_2$  is a vector  $L_2$ -norm.

### 3.4.3 Learning the dynamics matrix from complete and accurate measurements

To learn the dynamics matrix,  $\mathbf{A} \in \mathbb{R}^{d \times d}$ , we adopt the framework of regularized risk minimization Murphy (2012). Namely, we formulate an optimization problem in which the objective consists of an error term and regularization terms. The dynamics matrix is learned by minimizing such an objective; as shown in (3.6).

$$\underset{\mathbf{A}}{\text{minimize}} \quad J(\mathbf{A}) = J_e(\mathbf{A}) + J_r(\mathbf{A}) + J_s(\mathbf{A});\tag{3.6}$$

where

$$J_e(\mathbf{A}) = \frac{1}{2m} \sum_{i=1}^m \|(\mathbf{I} + h\mathbf{A})^{n_i} \mathbf{x}_0 - \mathbf{x}(t_i)\|_2^2,\tag{3.7}$$

$$J_r(\mathbf{A}) = \frac{\lambda_1}{2d^2} \|\mathbf{A}\|_F^2,\tag{3.8}$$

$$J_s(\mathbf{A}) = \frac{\lambda_2 h}{2n_m} \sum_{j=1}^{n_m-1} \|(\mathbf{I} + h\mathbf{A})^{j+1} \mathbf{x}_0 - (\mathbf{I} + h\mathbf{A})^j \mathbf{x}_0\|_2^2.\tag{3.9}$$

The error term, denoted by  $J_e$ , is the sum of squared errors (3.5) normalized by the number of time points,  $m$ . There are two regularization terms in the objective: the term  $J_r$  denotes the ridge loss (Frobenius norm) of the dynamics matrix; minimizing such a loss aims to prevent the overfitting of the model Vapnik and Vapnik (1998); the term  $J_s$  aims to ensure that the approximated time series are smooth Hamilton (1994). Theoretically, we can apply any appropriate regularization terms (e.g. matrix 2-norm/spectral norm in Boots et al. (2007) and nuclear norm in Liu and Hauskrecht (2015)) in this optimization formulation; however, in the considerations of computational efficiency and simplicity, we use

the above regularizations which are differentiable, so that we can employ gradient descent optimization methods to solve the problem.

The limitations of this formulation are that: 1) it assumes all the measurements are accurate, especially the initial measurement; 2) it assumes all the measurements are complete, which means all the variables must be observed in a measurement. However, in many cases, these two assumptions are too strong; therefore, we have to further extend the model to accommodate the cases in which the MTS is *noisy* and *partially observed*.

#### 3.4.4 Learning an LDS from noisy MTS measurements

If measurements are noisy, we have to formulate the objective function differently. In an LDS, the initial condition of the variables is extremely important because it is always one of the multipliers for approximating the time points, as shown in (3.4). Therefore, in order to learn the dynamics matrix, we need to also learn the initial state. To learn the initial state and the dynamics matrix simultaneously, we formulate the objective function of the optimization as follows:

$$\begin{aligned} \underset{\mathbf{x}_0, \mathbf{A}}{\text{minimize}} \quad J(\mathbf{x}_0, \mathbf{A}) &= J_e(\mathbf{x}_0, \mathbf{A}) + J_r(\mathbf{A}) \\ &+ J_s(\mathbf{x}_0, \mathbf{A}) + J_i(\mathbf{x}_0). \end{aligned} \tag{3.10}$$

The first three terms,  $J_e(\mathbf{x}_0, \mathbf{A})$ ,  $J_r(\mathbf{A})$ , and  $J_s(\mathbf{x}_0, \mathbf{A})$ , in this formulation are the same as defined in (3.7), (3.8), and (3.9). However, under the formulation in (3.6), we are only interested in solving the dynamics matrix,  $\mathbf{A}$ , so  $J_e(\mathbf{A})$  and  $J_s(\mathbf{A})$  were functions of variable  $\mathbf{A}$  alone. In a more realistic formulation, (3.10), we are interested in both  $\mathbf{x}_0$  and  $\mathbf{A}$ , and thus  $J_e(\mathbf{x}_0, \mathbf{A})$  and  $J_s(\mathbf{x}_0, \mathbf{A})$  are functions of both variables,  $\mathbf{x}_0$  and  $\mathbf{A}$ . We use the regularization term,

$$J_i(\mathbf{x}_0) = \frac{\lambda_3}{2d} \|\mathbf{x}_0 - \mathbf{x}(t_0)\|_2^2 \tag{3.11}$$

to ensure that the optimized initial observation is closed to the measurement.

### 3.4.5 Learning an LDS from partially observed MTS measurements

When only a subset of variables are observed, the error term in the objective function can be written as:

$$\begin{aligned}
 J_e(\mathbf{x}_0, \mathbf{A}) &= \frac{1}{2m} \sum_{i=1}^m \|\mathbf{O}_i \mathbf{e}_i\|_2^2 \\
 &= \frac{1}{2m} \sum_{i=1}^m \|\mathbf{O}_i [(I + h\mathbf{A})^{n_i} \mathbf{x}_0 - \mathbf{x}(t_i)]\|_2^2,
 \end{aligned} \tag{3.12}$$

where  $\mathbf{O}_i = \text{diag}(\mathbf{o}_i)$  is a diagonal matrix, and  $\mathbf{o}_i = [o_{i1}, o_{i2}, \dots]$  indicates whether the biomarkers are observed at time point,  $t_i$ ; namely,

$$o_{ij} = \begin{cases} 1, & \text{if at time point } i, \text{ the } j^{\text{th}} \text{ biomarker is observed} \\ 0, & \text{otherwise.} \end{cases}$$

### 3.4.6 Solving the optimization problem

It is challenging to simultaneously find  $\mathbf{x}_0$  and  $\mathbf{A}$  to minimize the objective function, especially when the objective function involves their products, as shown in (3.7). In order to solve this optimization problem, we employ an iterative strategy. Specifically, the algorithm minimizes the objective function, and finds the minimizers iteratively: in one iteration, the algorithm treats  $\mathbf{x}_0$  as known, and solves for the optimal  $\mathbf{A}$ ; in the following iteration, the algorithm treats  $\mathbf{A}$  as known and solves for the optimal  $\mathbf{x}_0$ . In each iteration, we can solve the sub-problem efficiently using a gradient descent approach.

Using the identities described in Gallier (2011) and Chen and Zadrozny (2001), results in the first-order differentiation of each term in the objective function w.r.t.  $\mathbf{A}$ , while

treating  $\mathbf{x}_0$  was a constant:

$$\frac{\partial J_e}{\partial \mathbf{A}} = \frac{1}{m} \sum_{i=1}^m \mathbf{O}_i [f_1(\mathbf{A}, h, n_i) - f_1(\mathbf{A}, h, 0)] \quad (3.13)$$

$$\times f_2(\mathbf{A}, h, n_i);$$

$$\frac{\partial J_r}{\partial \mathbf{A}} = \frac{\lambda_1}{d^2} \text{vec}(\mathbf{A})^T; \quad (3.14)$$

$$\frac{\partial J_s}{\partial \mathbf{A}} = \frac{\lambda_2 h}{n_m} \sum_{i=1}^{n_m-1} [f_1(\mathbf{A}, h, i+1) - f_1(\mathbf{A}, h, i)] \quad (3.15)$$

$$\times [f_2(\mathbf{A}, h, i+1) - f_2(\mathbf{A}, h, i)];$$

where

$$f_1(\mathbf{A}, h, n) = ((\mathbf{I} + h\mathbf{A})^n \mathbf{x}_0)^T (\mathbf{x}_0^T \otimes \mathbf{I}); \quad (3.16)$$

$$f_2(\mathbf{A}, h, n) = \sum_{k=1}^n \binom{n}{k} h^k \sum_{j=1}^k (\mathbf{A}^T)^{k-j} \otimes \mathbf{A}^{j-1}. \quad (3.17)$$

Here, the symbol  $\mathbf{I}$  represents the identity matrix with the same size of  $\mathbf{A}$ ,  $\text{vec}(\cdot)$  is the *vectorization operator* on a matrix, and the symbol  $\otimes$  represents the *Kronecker product operator*.

While treating  $\mathbf{A}$  as a constant, the first differentiation of the objective function terms w.r.t. the initial condition,  $\mathbf{x}_0$ , are

$$\frac{\partial J_e}{\partial \mathbf{x}_0} = \frac{1}{m} \sum_{i=1}^m \mathbf{O}_i [(\mathbf{I} + h\mathbf{A})^{n_i} \mathbf{x}_0 - \mathbf{x}(t_i)]^T (\mathbf{I} + h\mathbf{A})^{n_i} \quad (3.18)$$

$$\begin{aligned} \frac{\partial J_s}{\partial \mathbf{x}_0} &= \frac{\lambda_2 h}{n_m} \sum_{j=1}^{n_m-1} \mathbf{x}_0^T [(\mathbf{I} + h\mathbf{A})^{j+1} - (\mathbf{I} + h\mathbf{A})^j]^T \quad (3.19) \\ &\times [(\mathbf{I} + h\mathbf{A})^{j+1} - (\mathbf{I} + h\mathbf{A})^j] \mathbf{x}_0 \end{aligned}$$

$$\frac{\partial J_i}{\partial \mathbf{x}_0} = \frac{\lambda_3}{d} (\mathbf{x}_0 - \mathbf{x}(t_0))^T \quad (3.20)$$

The iterative procedure to learn  $\mathbf{A}$  and  $\mathbf{x}_0$  from an imperfect MTS is summarized in Procedure 1. The regularization parameters are user-specified. In our experiments, we

simply set  $\{\lambda_1, \lambda_2, \lambda_3\}$  as  $\{1, 1, 1\}$ . The objective function is non-convex; therefore, the solution may not be at the global minima. For MTS classification tasks, where a parameter set is learned from an MTS example, in order to have fair comparisons between dynamic matrices learned from MTS examples, we do not recommend to initialize  $\mathbf{x}_0$  and  $\mathbf{A}$  randomly. In our experiments, we initialized the dynamics matrix as a zero-matrix and initialize the initial condition as the initial sample. In the optimization procedure, we need to set parameters  $\{\eta, Tol, N_{iter}\}$ , where  $\eta$  is the update *step size*, *Tol* is the *optimality tolerance*, and  $N_{iter}$  is the *maximum iteration number*. In our experiments, we empirically found that setting  $\eta = 10^{-2}$  allowed a good balance of convergence and convergent rate, and setting  $\{Tol, N_{iter}\} = \{10^{-4}, 20\}$  allowed a good balance of computational time and optimality.

### 3.4.7 Classifying LDS's via a Kernel method

After learning the dynamics matrices and the initial conditions of all imperfect MTS examples in the data set, we would like to use them as representations of the MTSs for classification tasks. In previous studies, various metrics were developed to quantify the similarities/dissimilarities of two dynamical systems. Examples include the Martin distance Martin (2000) and subspace angles De Cock and De Moor (2002). In this study, we adapted the kernel-based framework proposed in Vishwanathan et al. (2007) as the basis of our LDS classification model, as this framework is a generalization of the previous two examples and can be directly integrated into a support vector machine classifier.

A kernel of two LDS's,  $(\mathbf{x}_0, \mathbf{A})$  and  $(\mathbf{x}'_0, \mathbf{A}')$ , can be defined as

$$\begin{aligned}
 &k((\mathbf{x}_0, \mathbf{A}), (\mathbf{x}'_0, \mathbf{A}')) \\
 &:= \mathbf{x}_0^T \left[ \int_0^\infty \exp(\mathbf{A}t)^T \mathbf{W} \exp(\mathbf{A}'t) \mu(t) dt \right] \mathbf{x}'_0,
 \end{aligned} \tag{3.21}$$

where  $\exp(\mathbf{A}t)$  is a matrix exponential, matrix  $\mathbf{W}$  is a positive semi-definite matrix which is used to weight the different variables in the MTS's, and the function  $\mu(t)$  is a discount

---

**Procedure 1** Iteratively learning  $\mathbf{A}$  and  $\mathbf{x}_0$  from an Imperfect MTS
 

---

**Input:**

- 1: An MTS with time stamp set  $\{t_0, t_1, \dots, t_m\}$  and state vectors  $\{\mathbf{x}(t_0), \mathbf{x}(t_1), \dots, \mathbf{x}(t_m)\}$
- 2: A small time increment  $h$  ▷ A user-decide variable
- 3:  $\eta, Tol$  and  $N_{iter}$  ▷ Step size, tolerance and max iteration number

**Output:**

- 1: Estimated initial state vector,  $\mathbf{x}_0$
- 2: Estimated dynamics matrix,  $\mathbf{A}$

**Procedure:**

- 1: initialize  $\mathbf{A}$ , and initialize  $\mathbf{x}_0$  as  $\mathbf{x}(t_0)$
  - 2:  $J_{old} \leftarrow J(\mathbf{x}_0, \mathbf{A})$  as defined in (3.10)
  - 3: compute  $\{n_1, n_2, \dots, n_i, \dots, n_m\}$ , where  $n_i = (t_i - t_0)/h$
  - 4:  $Counter_1 \leftarrow 0$
  - 5: **repeat**
  - 6:      $counter_1 ++$
  - 7:     *// Use BCGD Aglorithm and treat  $\mathbf{x}_0$  as a constant*
  - 8:      $counter_2 \leftarrow 0$
  - 9:     **repeat**
  - 10:          $counter_2 ++$
  - 11:         compute  $\frac{\partial J_e}{\partial \mathbf{A}}$  using (3.13)
  - 12:         compute  $\frac{\partial J_r}{\partial \mathbf{A}}$  using (3.14)
  - 13:         compute  $\frac{\partial J_s}{\partial \mathbf{A}}$  using (3.15)
  - 14:          $\frac{\partial J}{\partial \mathbf{A}} \leftarrow \frac{\partial J_e}{\partial \mathbf{A}} + \frac{\partial J_r}{\partial \mathbf{A}} + \frac{\partial J_s}{\partial \mathbf{A}}$
  - 15:          $\mathbf{A} \leftarrow \mathbf{A} - \eta \frac{\partial J}{\partial \mathbf{A}}$
  - 16:         **until**  $\|\frac{\partial J}{\partial \mathbf{A}}\| < Tol$  or  $counter_2 > N_{iter}$
  - 17:         *// Use BCGD Aglorithm and treat  $\mathbf{A}$  as a constant*
  - 18:          $counter_2 \leftarrow 0$
  - 19:         **repeat**
  - 20:              $counter_2 ++$
  - 21:             compute  $\frac{\partial J_e}{\partial \mathbf{x}_0}$  using (3.18)
  - 22:             compute  $\frac{\partial J_s}{\partial \mathbf{x}_0}$  using (3.19)
  - 23:             compute  $\frac{\partial J_i}{\partial \mathbf{x}_0}$  using (3.20)
  - 24:              $\frac{\partial J}{\partial \mathbf{x}_0} \leftarrow \frac{\partial J_e}{\partial \mathbf{x}_0} + \frac{\partial J_r}{\partial \mathbf{x}_0} + \frac{\partial J_s}{\partial \mathbf{x}_0}$
  - 25:              $\mathbf{x}_0 \leftarrow \mathbf{x}_0 - \eta \frac{\partial J}{\partial \mathbf{x}_0}$
  - 26:             **until**  $\|\frac{\partial J}{\partial \mathbf{x}_0}\| < Tol$  or  $counter_2 > N_{iter}$
  - 27:              $J_{new} \leftarrow J(\mathbf{x}_0, \mathbf{A})$
  - 28:              $\Delta J \leftarrow J_{old} - J_{new}$
  - 29:              $J_{old} \leftarrow J_{new}$
  - 30:         **until**  $\|\Delta J\| < Tol$  or  $counter_1 > N_{iter}$
  - 31:         **Return**  $\mathbf{x}_0$  and  $\mathbf{A}$
- 

function. This kernel can be seen as a special inner product of the trajectories of the two LDS's.

In our study, since we do not have any prior knowledge of the weights of the states in

an LDS, we assume the states are equally weighted; therefore, without loss of generality, we replace matrix  $\mathbf{W}$  with an identity matrix  $\mathbf{I}$ . The definition of the discount function,  $\mu(t)$ , is problem specific. In general, there are two popular choices, one is the Dirac delta function,  $\mu(t) = \delta(t - \tau)$ , and the other one is the exponential decay function,  $\mu(t) = e^{-\lambda t}$ .

When  $\mu(t) = \delta(t - \tau)$ , the kernel is reduced to the inner product of the state vectors at time,  $\tau$ . Namely, the kernel defined in (3.21), with  $\mathbf{W}$  replaced by  $\mathbf{I}$ , is reduced to

$$\begin{aligned} k((\mathbf{x}_0, \mathbf{A}), (\mathbf{x}'_0, \mathbf{A}')) \\ := \mathbf{x}'_0{}^T \exp(\mathbf{A}\tau)^T \exp(\mathbf{A}'\tau) \mathbf{x}'_0. \end{aligned} \tag{3.22}$$

This kernel is useful if we know the MTS's (or LDS's) are the most distinguishable at time,  $\tau$ , *a priori*. In our case, we used the exponential decay function, in which there is no discount at time zero, and the discount become large as the time series progress. Using the exponential decay function, the kernel defined in (3.21), with  $\mathbf{W}$  replaced by  $\mathbf{I}$ , can be written as

$$\begin{aligned} k((\mathbf{x}_0, \mathbf{A}), (\mathbf{x}'_0, \mathbf{A}')) \\ := \mathbf{x}'_0{}^T \left[ \int_0^\infty \exp(\mathbf{A}t)^T \exp(\mathbf{A}'t) e^{-\lambda t} dt \right] \mathbf{x}'_0, \end{aligned} \tag{3.23}$$

where  $\lambda$  is a hyper-parameter, which controls how fast the discount increases. However, there is a restriction on the value of  $\lambda$ ; that is  $\lambda > 2\Lambda$ , where  $\Lambda = \max(\|\mathbf{A}\|_2, \|\mathbf{A}'\|_2)$ . The symbol  $\|\cdot\|_2$  represents the L-2 norm operation, and when the operand is a matrix, it is also called the spectral norm, which is the largest singular value of the matrix. Such a restriction on  $\lambda$  could ensure the convergence of the integral.

Even though the integral in (3.23) converges, it is hard to compute because of the infinite sum. To simplify the integral, we first let

$$\mathbf{M} = \int_0^\infty \exp(\mathbf{A}t)^T \exp(\mathbf{A}'t) e^{-\lambda t} dt, \tag{3.24}$$

and then, by assuming both  $\mathbf{A}$  and  $\mathbf{A}'$  are non-singular and using integration by parts, we

arrived at

$$\begin{aligned}
\mathbf{M} &= \int_0^\infty \exp(\mathbf{A}t)^T \exp(\mathbf{A}'t) e^{-\lambda t} dt \\
&= (\mathbf{A}^T)^{-1} e^{-\lambda t} (\exp(\mathbf{A}t))^T \exp(\mathbf{A}'t) \Big|_0^\infty \\
&\quad - \int_0^\infty (\mathbf{A}^T)^{-1} e^{-\lambda t} \exp(\mathbf{A}t)^T \exp(\mathbf{A}'t) (\mathbf{A}' - \lambda \mathbf{I}) dt \\
&= (\mathbf{A}^T)^{-1} \\
&\quad - (\mathbf{A}^T)^{-1} \left[ \int_0^\infty e^{-\lambda t} \exp(\mathbf{A}t)^T \exp(\mathbf{A}'t) dt \right] (\mathbf{A}' - \lambda \mathbf{I}) \\
&= (\mathbf{A}^T)^{-1} - (\mathbf{A}^T)^{-1} \mathbf{M} (\mathbf{A}' - \lambda \mathbf{I}).
\end{aligned}$$

By multiplying both sides by  $(\mathbf{A}^T)^{-1}$  and arranging the terms, we obtained an equation of  $\mathbf{M}$ :

$$\mathbf{A}^T \mathbf{M} + \mathbf{M} \mathbf{A}' - \lambda \mathbf{M} = -\mathbf{I}. \quad (3.25)$$

Solving for  $\mathbf{M}$  is an easier task; by vectorizing both sides of (3.25) and organizing the terms, we obtain

$$[\mathbf{I} \otimes \mathbf{A}^T + (\mathbf{A}')^T \otimes \mathbf{I} - \lambda \mathbf{I}'] \text{vec}(\mathbf{M}) = -\text{vec}(\mathbf{I}),$$

where  $\mathbf{I}'$  is an identity matrix whose number of column is the same as the dimensionality of  $\text{vec}(\mathbf{M})$ . Then we can solve  $\text{vec}(\mathbf{M})$  as

$$\text{vec}(\mathbf{M}) = [\lambda \mathbf{I}' - \mathbf{I} \otimes \mathbf{A}^T - (\mathbf{A}')^T \otimes \mathbf{I}]^{-1} \text{vec}(\mathbf{I}). \quad (3.26)$$

Combining (3.26), (3.24), and the vectorized (3.23), we get

$$\begin{aligned}
k((\mathbf{x}_0, \mathbf{A}), (\mathbf{x}'_0, \mathbf{A}')) &= \text{vec}(\mathbf{x}_0^T \mathbf{M} \mathbf{x}'_0) \\
&= [\mathbf{x}'_0 \otimes \mathbf{x}_0]^T [\lambda \mathbf{I}' - \mathbf{I} \otimes \mathbf{A}^T - (\mathbf{A}')^T \otimes \mathbf{I}]^{-1} \text{vec}(\mathbf{I})
\end{aligned} \quad (3.27)$$

We would like to point out that the first equality in (3.27) holds true because  $k((\mathbf{x}_0, \mathbf{A}), (\mathbf{x}'_0, \mathbf{A}'))$  is a scalar, and its vectorization is a scalar. Comparing to (3.23), (3.27) is computable, but

we need to inverse the matrix,  $\lambda \mathbf{I}' - \mathbf{I} \otimes \mathbf{A}^T - (\mathbf{A}')^T \otimes \mathbf{I}$ . At the first sight, one may think the matrix inversion could cost high computation overhead; however, this matrix has a sparse and block diagonal structure, so we can exploit the structure and compute its inverse rather cheaply.

In the cases where the MTS's are not aligned, namely, the initial states of the LDS's were observed at different times among individual MTS's after the onset, the terms  $\mathbf{x}_0$  and  $\mathbf{x}'_0$  in (3.27) seem to become meaningless. Therefore, we can define the kernel between dynamics matrices by

$$k(\mathbf{A}, \mathbf{A}') = tr(\mathbf{M}) \tag{3.28}$$

With this computable kernel defined, we can employ the support vector machine model for classification.

### 3.5 Experiments and Results

To evaluate our approach, we have conducted experiments on both synthetic and real-life data.

#### 3.5.1 Experiments on synthetic MTS data

The purpose of the experiments on synthetic MTS data is to check whether the dynamics matrices learned by our approach can preserve the similarities of the underlying LDS's. Based on the state trajectory generation procedure, we had two experiments.

##### *Sub-Experiment 1: noisy initial states*

In this experiment, we generated the state trajectories based on the following procedure:

1. Randomly generate  $K$  LDS's; i.e.  $K$  tuples of  $(\mathbf{x}_0, \mathbf{A})$ , where  $\mathbf{x}_0 \in \mathbb{R}^5$  and  $\mathbf{A} \in \mathbb{R}^{5 \times 5}$ .

2. Add random noise to a initial state and generate state trajectories using Euler forward method (eq. 3.4), and repeat  $L$  times for each  $(\mathbf{x}_0, \mathbf{A})$  tuple.

For ease of visualization, we set  $K = 3$  and  $L = 10$  in our experiments; therefore, there are 3 different LDS's, and each LDS is used to generate 10 trajectories with noisy initial states. MTS's were obtained by sampling the trajectories at 20 equally spaced time points. At the end, we have 30 MTS's with 20 time points and equal lengths. We applied our algorithm on the MTS's to learn the dynamics matrices. The pairwise similarity of dynamics matrices,  $sim_{\mathbf{A}}$ , is computed by (3.29).

$$sim_{\mathbf{A}}(i, j) = \|\mathbf{A}_i - \mathbf{A}_j\|_F \quad (3.29)$$

Based on the pairwise similarities, we projected the learned dynamics matrices on a 2D space using multidimensional scaling Borg and Groenen (2005). Each object in the 2D space represents a learned dynamics matrix, thus, ideally, dynamics matrices learned from the MTS's sampled from the same LDS (with noisy initial states) should form a cluster. For comparison, we also computed the pairwise similarity of the generated trajectories,  $sim_{\mathbf{x}}$ , by using (3.30)

$$sim_{\mathbf{x}}(i, j) = \sum_k \|\mathbf{x}_i(t_k) - \mathbf{x}_j(t_k)\|_2 \quad (3.30)$$

Although the dynamics matrices were the same, trajectories generated from two noisy initial states varied significantly, as shown in Figure 3.1a and 3.1b, in which Example 1 and Example 2 are trajectories generated from the same LDS, with the same dynamics matrix but noisy initial states. These variations of trajectories are also shown in the trajectory similarity plot, as shown in Figure 3.1c), in which the symbol of Example 1 and the symbol of Example 2 are far apart. Despite the variations in the trajectories, the learned dynamics matrices preserved the similarities of the LDS's, as shown in Figure 3.1d), in which the symbol of Example 1 and the symbol of Example 2 are close. Therefore, our learning approach is able to learn the underlying LDS despite the initial states being noisy.

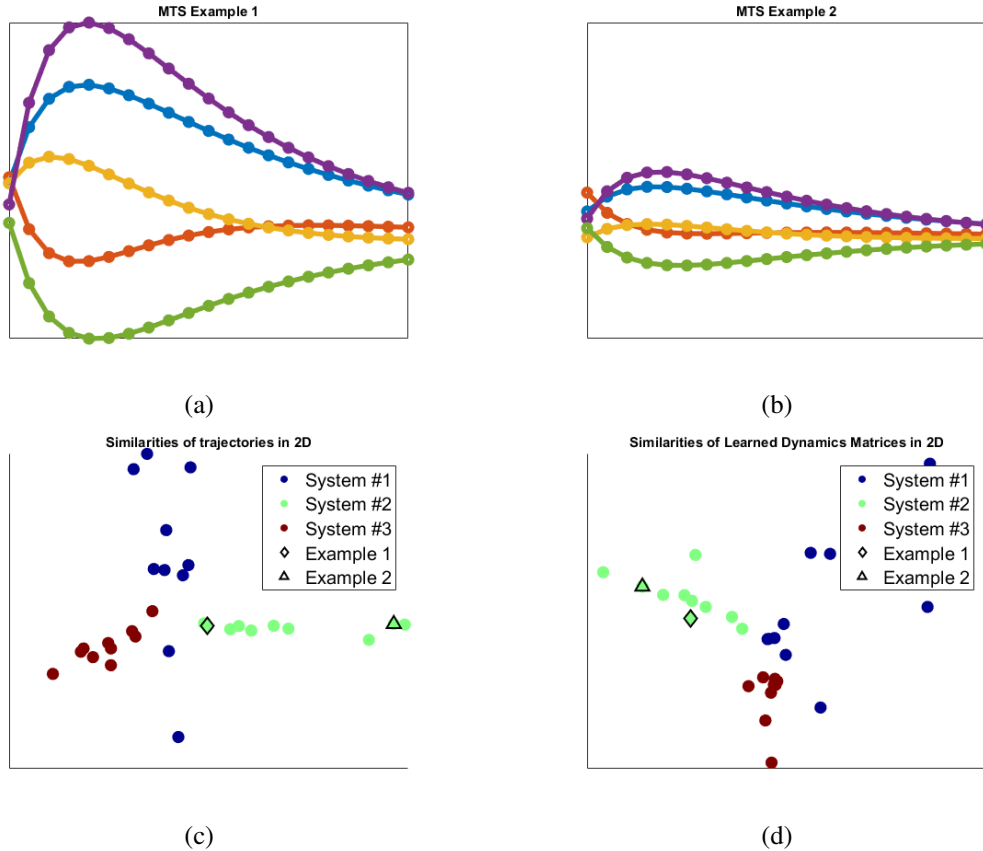


FIGURE 3.1: Examples of trajectories generated by the same LDS with noisy initial conditions and similarity plots of generated trajectories and learned dynamics matrices. a) and b) two example trajectories generated by the same LDS with noisy initial states; the same color indicates the same state. c) and d) similarity plots of generated trajectories and learned dynamics matrices; the same color indicates the trajectories generated from the same LDS.

### *Sub-Experiment 2: noisy dynamics matrices*

This experiment was similar to the above experiment; however, in the trajectory generation procedure, instead of making the initial states noisy, we made the dynamics matrices noisy. Using the same parameter settings (i.e.  $K = 3$  and  $L = 10$ ), we generated 30 trajectories and thus 30 MTS's. Dynamics matrices were learned from the generated MTS's.

Trajectories generated from the same LDS with noisy dynamics matrices also varied, as shown in Figure 3.2a and 3.2b, in which, Example 1 and Example 2 are trajectories generated from the same LDS, with the same initial states but noisy dynamics matrices.

The trajectories generated with noisy dynamics matrix may vary significantly, as shown in Figure 3.2c, in which Example 1 and Example 2 are far apart. But the dynamics matrices learned by our algorithm preserved the similarities much better, as shown in Figure 3.2d, in which the clusters of the 3 LDS's are very distinct.

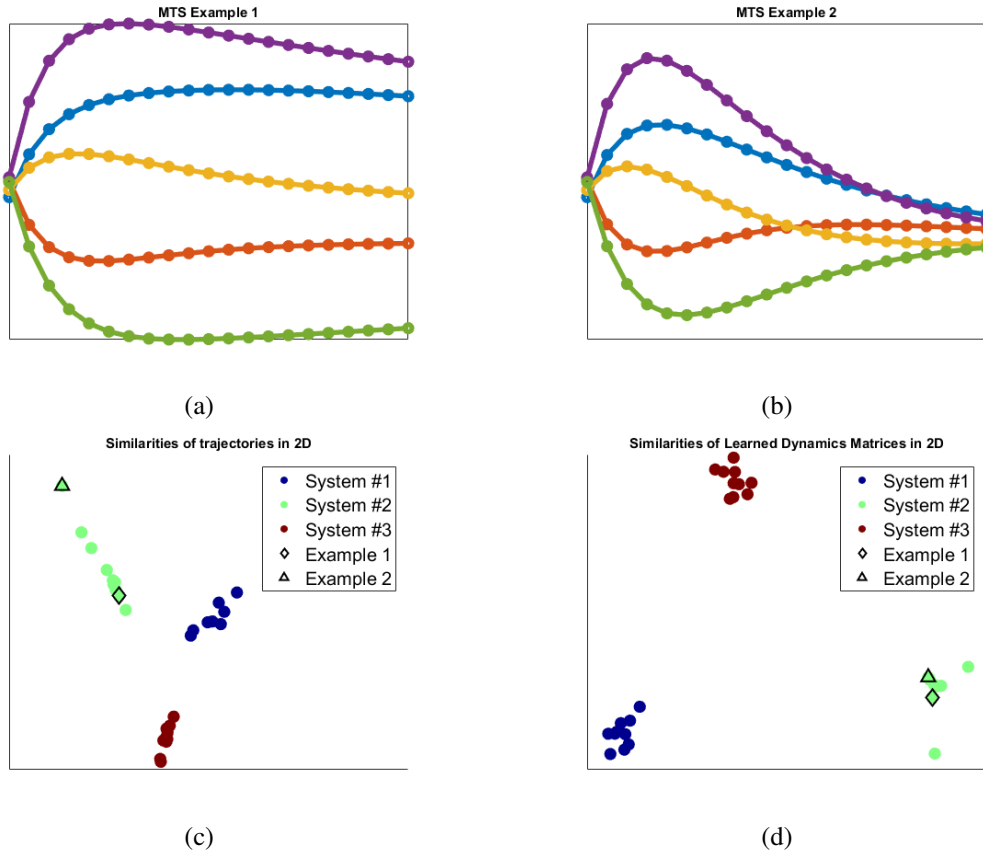


FIGURE 3.2: Examples of trajectories generated by the same LDS with noisy dynamics matrices and similarity plots of generated trajectories and learned dynamics matrices. a) and b) two example trajectories generated by the same LDS with noisy initial states; the same color indicates the same state. c) and d) similarity plots of generated trajectories and learned dynamics matrices; the same color indicates the trajectories generated from the same LDS.

### 3.5.2 Experiments on real-life datasets

#### *Datasets and baselines*

To evaluate the efficiency of using LDS's as representations of imperfect MTS's, we conducted experiments on three human blood gene expression time series datasets, influenza A (H3N2), Rhinovirus (HRV), and respiratory syncytial virus (RSV) Zaas et al. (2009). The number of MTS's in the H3N2, HRV, and RSV datasets were 17, 20, and 20, respectively. At each time point of the MTS, expression levels of multiple genes were measured. The number of genes in each dataset was originally over 10,000. In our experiments, in each dataset, we only included the genes suggested by a previous study Zaas et al. (2009); therefore, the number of genes included in the H3N2, HRV, and RSV datasets were 23, 26, and 24, respectively. Each MTS in the datasets has a different number of time points (i.e., the number of temporal gene expression measurements was different from subject to subject). In the H3N2 dataset, most of the subjects had 16 temporal gene expression measurements, and thus there were 16 time points available in the MTS's of those subjects; a small number of subjects in the H3N2 dataset had 15 time points available in their MTS's. The numbers of temporal gene expressions in the subjects in the HRV dataset varied significantly, ranging from 7 to 14. The numbers of temporal gene expressions in the subjects in the RSV dataset varied even more significantly, ranging from 6 to 21. In the MTS's of the H3N2 dataset, every two consecutive gene expressions were spaced by 1 time unit. However, the gene expressions in MTS's of the HRV and RSV datasets were not equally spaced. The interval of two consecutive gene expressions may be separated as much as 7 units and as little as 1 unit in the HRV dataset. The interval of two consecutive gene expressions may be separated as much as 10 units and as little as 1 unit in the RSV dataset. A summary of the datasets is given in Table 3.2. MTS's in the H3N2 dataset are close to "perfect"; namely, most of them have the same lengths, and samples are regularly spaced. In contrast, the RSV dataset is a good demonstration of an imperfect

MTS dataset, in which both lengths and sampling intervals of the MTS’s vary significantly. Within each dataset, we formulated a binary classification task to determine whether an MTS was from an individual with symptomatic acute respiratory infection or an individual with no infection.

Table 3.2: Summary of 3 datasets

	H3N2	HRV	RSV
Number of genes	23	26	24
Number of MTS’s (pos/neg)	9/8	10/10	9/11
Number of time points (range)	15-16	7-14	6-21
Sampling interval (range)	1	1-7	1-10

For comparison, in our experiments, we also included three baseline representations/methods:

- **stat + SVM** Using statistical summaries of temporal samples as time series features has been proven effective in biomedical applications Henry et al. (2015). In our experiments, for each variable in an MTS instance, we computed four statistical measures (*mean, standard deviation, maximum, minimum*) to summarize the time series. That is, if there are  $n$  variables, in this approach, an MTS is represented by a  $4n$ -dimensional feature vector. After the features were generated, we used a support vector machine (SVM) model as the classifier. The kernels used in the SVM model were linear, radial basis function, and polynomial of order 3. The best result among the different kernels will be reported. The hyperparameter in the model was determined by a nested leave-one-out cross-validation in the training set.
- **PAA + INN** Piecewise Aggregate Approximation (PPA) Keogh et al. (2001) is an efficient representation for long time series. In our experiment, for each (MTS) instance, we applied PAA on each variable, and then concatenated all the variables as a feature vector. The feature vectors of all the instances have the same dimension-

alities. After the transformation, we used *1-nearest-neighbor* as the classification model.

- ***DTLDS-kernel SVM*** In our experiments, we implemented this baseline based on Borgwardt et al. (2006), in which a singular value decomposition based approach is used to learn a discrete time linear dynamical system (DTLDS) from an MTS with regularly spaced samples. Then a classification was performed on an SVM classifier by constructing a kernel using learned DTLDS's. This method is similar to our proposed method; however, it is not applicable to time series with imperfectness; therefore, in order to use this method on the MTS's used in the experiments, interpolation and(or) truncation of the time series were necessary. More details are given in the following section about data preprocessing.

### 3.5.3 *Data preprocessing*

Different genes are expressed in different scales; therefore, we needed to normalize the MTS before learning and modeling. In our experiments, we scaled each gene in all MTS into the (0,1) interval Cao and Obradovic (2015). In some of the baseline models (i.e. *PAA + INN* and *DTLDS-kernel SVM*), gene expressions are required to be measured regularly; therefore, for MTS's with irregularly spaced measurements, we performed linear interpolations. In addition, in *PAA + INN* and *LDS-kernel SVM* models, MTS's are required to be of equal length. In our experiments, we truncated the MTS's in a dataset to the length of the shortest MTS in that dataset.

### 3.5.4 *Experiments on using all available data in the datasets*

In this experiment, we ran all the baseline methods and the proposed method on all three datasets and used all available time points in each dataset. Due to the limited number of MTS in each dataset (Table 3.2), in our experiments, the results were obtained by using leave-one-out cross-validations. The prediction accuracy of each representation and model

is shown in Table 3.3.

Table 3.3: Leave-one-out cross validation accuracies on 3 binary classification tasks. Best performances are in bold

	H3N2	HRV	RSV
<i>PAA + INN</i>	<b>1.000</b>	0.800	0.750
<i>stat + SVM</i>	<b>1.000</b>	0.850	0.750
<i>DTLDS-kernel SVM</i>	<b>1.000</b>	0.750	0.650
<i>proposed</i>	<b>1.000</b>	<b>1.000</b>	<b>0.800</b>

All the representations/methods performed equally extremely well on H3N2. That is not surprising because instances in that dataset are of very high quality (see Table 3.2): the lengths of the MTS’s are almost equal, samples are acquired in regular intervals, and all MTS’s have a decent number of time points. In contrast, the accuracies on HRV and RSV were negatively affected by the imperfectness of MTS’s, such that all the baseline methods could not achieve as high accuracy as on H3N2. Our proposed method has the best accuracies across all datasets. It is worth pointing out that even in an imperfect MTS HRV dataset, our proposed method still achieved 100% accuracy, while the second best accuracy was only 85%.

### 3.5.5 Experiments on the H3N2 dataset with imposed imperfectness

When using all available time points in the H3N2 dataset, all the models achieved perfect classification accuracies. In the following experiments, we manually imposed imperfectness into the H3N2 dataset to characterize the performance of four methods when affected by various levels of defects seen in real-life applications.

**MTS’s with missing time points** In this experiment, we removed completely at random approximately 20%, 40%, 60%, and 80% of the time points from the original MTS’s. Because each MTS in the dataset has 15 to 16 time points, after the removal, there were 13, 10, 7, and 4 time points left. We repeated the removal process for 5 times, and each time, different time points were removed. By doing this, we can emulate imperfectness

such as irregular sampling intervals and potentially unaligned MTS's. Then we applied the classification methods on the imperfect MTS's. The mean accuracies of each method and their standard deviations as the percentage missing increases are shown at Figure 3.3.

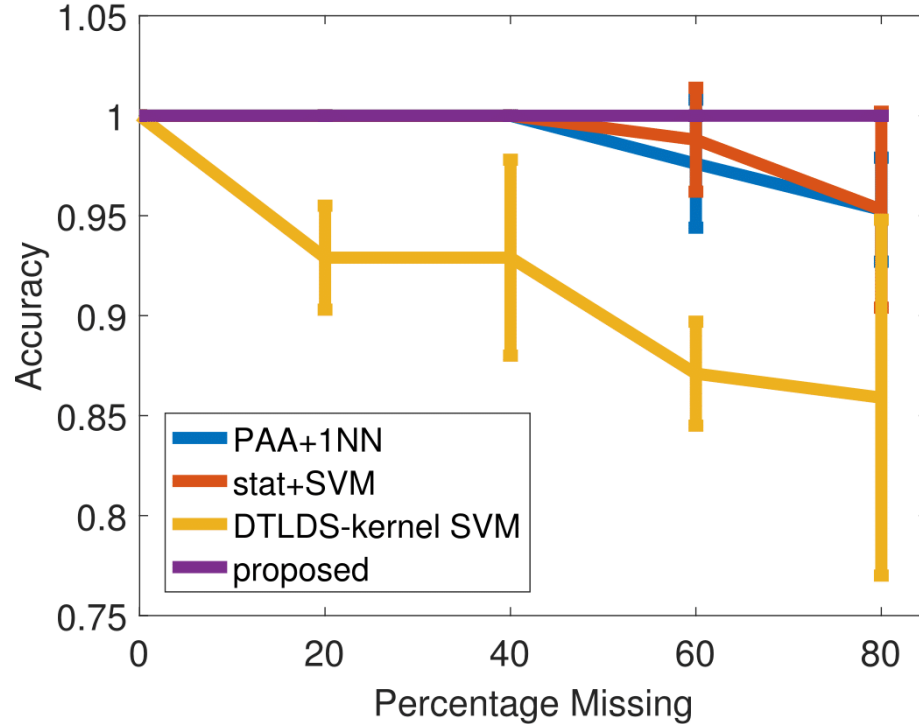


FIGURE 3.3: Means and standard deviations of each methods as the percentage of missing time points increases from 20% to 80%. The vertical line indicates one standard deviation.

From Figure 3.3, we notice that the *DTLDS-kernel SVM* was affected the most by irregular sampling intervals. The performances of *PAA+1NN* and *stat+SVM* were affected minimally, as they both kept 100% accuracy when up to 40% of time points were missing, and their accuracies only degraded around 5% when 80% of time points were missing. Our proposed method achieved 100% accuracy even when 80% of data were missing.

**MTS's with truncations** In this experiment, we imposed imperfectness by randomly truncating the leading and ending time points from individual MTS's; specifically, we randomly truncated 25%, 50%, and 75% time points at the beginning, at the end, or both (at the beginning and the end) from the original MTS's. By doing this, we obtained MTS's which are unaligned, short, and with a limited number of samples. After the truncation,

each MTS had 12, 8, and 4 regularly spaced time points. We repeated the truncation process 5 times, and at each time, different portions of the leading and ending time points would be truncated. The mean accuracies of each method and their standard deviations as the percentage truncation increases are shown at Figure 3.4.

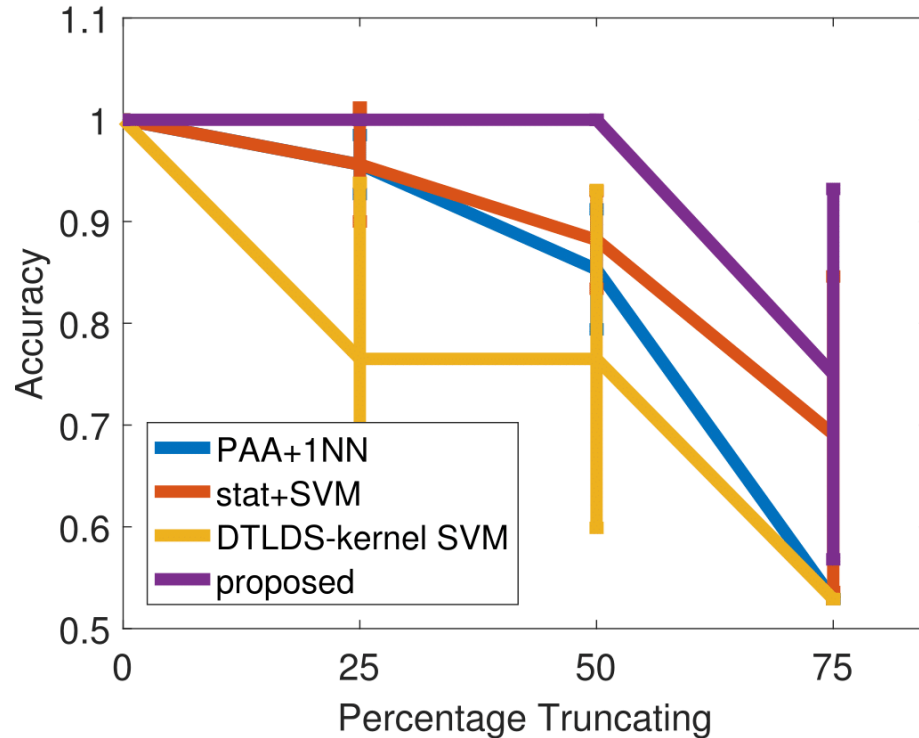


FIGURE 3.4: Means and standard deviations of each methods as the percentage truncating increases from 20% to 80%. The vertical line indicates one standard deviation.

In contrast to the previous experiment, the accuracies of all the methods were affected more in this experiment, although the MTS’s had roughly the same number of time points. When time points were removed uniformly and randomly, the length of an MTS might not be reduced; however, in this experiment, we did not only reduce the number of time points in an MTS but also reduced its length. Therefore, methods such as *PAA+INN* and *stat + SVM*, which are highly dependent on the trend and the values (usually the minimal/maximum values) at the beginning/end of a process, were hurt more by this kind of data deficiency. Our proposed method kept its perfect classification accuracy until the

percentage of truncating exceed 50%. The proposed method was able to continue performing well because the learned LDS's were able to capture the unique characteristics of the dynamics in the MTS's from different classes.

### 3.6 Conclusion

In this paper, we proposed a method to learn continuous-time LDS's from MTS's with various forms of imperfectness; i.e. limited time points, irregular sampling intervals, unaligned, noisy, partially observed, and short spanned. By adopting a powerful LDS kernel formulation, we employed a support vector machine model for classification tasks. Empirical results on three diagnostic tasks with different levels of imperfectness provided evidence that our proposed method is effective and able to outperform alternative methods.

# CHAPTER 4

## TIME-TO-EVENT ESTIMATION BY RE-DEFINING TIME

### 4.1 Introduction

The primary goal of a time-to-event estimation model is to accurately infer the occurrence time of a target event. Most existing studies focus on developing new models to effectively utilize the information in the censored observations. In this paper, we propose a model to tackle the time-to-event estimation problem from a completely different perspective. Our model relaxes a fundamental constraint that the target variable, time, is a univariate number which satisfies a partial order. Instead, the proposed model interprets each event occurrence time as a time concept with a vector representation. We hypothesize that the model will be more accurate and interpretable by capturing 1) the relationships between features and time concept vectors and 2) the relationships among time concept vectors. We also propose a scalable framework to simultaneously learn the model parameters and time concept vectors. Rigorous experiments and analysis have been conducted to demonstrate the efficiency and effectiveness of the proposed model. Furthermore, similarity information among time concept vectors helped in identifying time regimes, thus leading to a potential

knowledge discovery related to the human cancer datasets considered in our experiments.

## 4.2 Background

The primary goal of a time-to-event estimation model is to accurately infer the occurrence time of a target event. Time-to-event data analysis has been an active research topic due to its tremendous application values in a variety of disciplines including biology, healthcare, engineering, economics, and sociology Tierney et al. (2007). Time-to-event estimation is also called survival analysis Klein and Moeschberger (2006), reliability analysis, duration modeling, and event history analysis. The most unique characteristic of the time-to-event estimation problem is the presence of censored examples in the data. A censored example is an example whose event occurrence time is unobserved due to observation window limits or losing track during the observation window. The most common censoring cases are left censoring and right censoring. In the left censoring case, the event occurs before the beginning (left edge) of the observation window; similarly, in the right censoring case, the event occurs after the end (right edge) of the observation window. In this paper, we only consider the right censoring case.

Because of the uniqueness of the time-to-event data, most existing models focus on developing algorithms for effectively extracting information from the censored examples. Recently, machine learning methods have been adopted for time-to-event modeling Wang et al. (2017b), for instance, Survival Tree Models Gordon and Olshen (1985); LeBlanc and Crowley (1992); Bou-Hamad et al. (2011), Support Vector Machine for censored data Khan and Zubek (2008); Van Belle et al. (2007, 2011), Random Survival Forest Ishwaran et al. (2011), and Survival Boosting Trees Hothorn et al. (2005). Highly innovative new approaches have been proposed; for example, active learning Vinzamuri et al. (2014), transfer learning Li et al. (2016b), multi-task survival analysis Li et al. (2016a); Wang et al. (2017a), deep survival analysis Ranganath et al. (2016) and adversarial learning

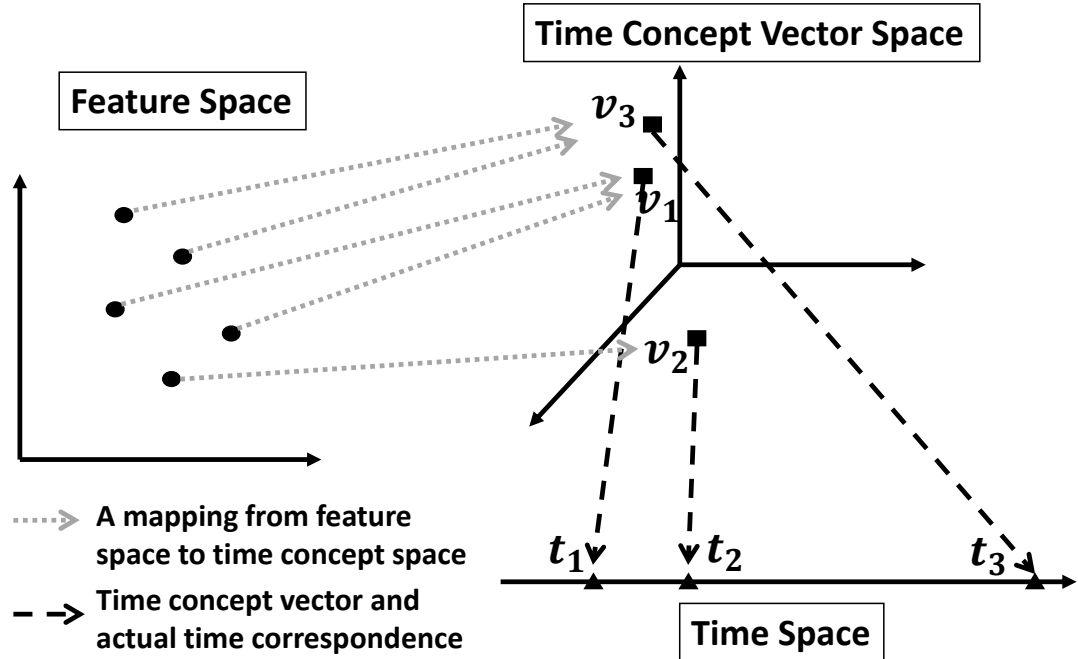


FIGURE 4.1: The proposed model first determines the corresponding time concept of each example and then infers the event occurrence time.

Chapfuwa et al. (2018).

In this paper, we approach the time-to-event estimation problem from a different perspective. In a conventional time-to-event problem formulation, the target variable *time* is univariate and satisfies a partial order; i.e., if  $t_1 < t_2 < t_3$ , then  $t_2 - t_1 < t_3 - t_1$ . In our model, instead of directly using *time* as the target variable, we propose to treat each discretized event occurrence time as a concept with a vector representation (called a time concept vector), and use the vector representation to replace *time* as the target. Our hypothesis is that an event occurrence time should be treated as a word/concept which carries an abstract meaning. Therefore, instead of univariate numbers, multidimensional vectors are more suitable for capturing the similarities among the concepts representing the event occurrence times. By effectively exploiting the relations between features and time concepts, we can obtain a more accurate model for time-to-event estimation. Furthermore, the similarity among time concept vectors may reveal information for knowledge discovery;

for example, time regime identification. The high-level idea of the proposed model is illustrated in Figure 4.1. In the figure, each example is represented by a dot in a 2D feature space, each time concept is represented by a dot in a 3D vector space, and each event occurrence time is represented in a 1D time space (time line). Instead of learning a model to directly assign each example an event occurrence time (i.e.  $\mathbb{R}^2 \rightarrow \mathbb{R}$ ), the proposed model assigns each example a time concept (i.e.  $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ ), and then the actual event time is inferred by the known concept-time correspondence. Because the time concept vector of an event occurrence time is unknown *a priori*, we developed a framework to learn the model parameters and the time concept vectors jointly. The contributions of our paper are summarized as follows:

- We propose a new time-to-event estimation model which indirectly infers event occurrence times via time concept vectors.
- We develop an efficient framework to learn model parameters and time concept vectors jointly.
- We conduct rigorous experiments to demonstrate the effectiveness, interpretability, and scalability of the proposed model.

### 4.3 Related Work

The time-to-event model (or survival analysis model) is one of the most fruitful research topics in statistics. For a comprehensive survey on this topic, one can refer to Wang et al. (2017b) and references therein. In this section, we will briefly summarize some of the most widely used models.

The Cox proportional hazard model Cox (1992) is one of the earliest and one of the most influential models in the topic of survival analysis. The parameters in the Cox model are determined by optimizing a partial likelihood function. Models which use the partial

likelihood as the optimization objective are considered as Cox-based models. The basic formulation of the Cox model is highly subject to overfitting in high-dimensional data. To address this shortcoming, variants of the Cox model based on different regularizations are proposed; for example, LASSO-Cox Tibshirani (1997), is a Cox-based model with an  $L_1$ -norm regularization, and EN-Cox Simon et al. (2011) is a Cox-based model with the elastic net regularization.

Parametric models are another class of widely used survival analysis models. In parametric models, the event occurrence time is usually assumed to satisfy an underlying distribution of which the density, survivorship, hazard and cumulative hazard functions are easy to derive. The model parameters are then determined by optimizing a likelihood function based on the given data. Some popular choices of the underlying distributions include Logistic, Weibull, Log-Gaussian, and Log-Logistic Lee and Wang (2003).

The time-to-event problem is similar to the traditional regression problem in the sense that the target variable is continuous. However, the traditional regression model cannot be directly applied to the time-to-event problem due to the existence of censored examples. The Tobit model Tobin (1958) is one of the earliest linear regression models which incorporate the censored examples in the optimization objective. The Buckley-James (BJ) regression model Buckley and James (1979) handles the censored data by using an auxiliary Kaplan-Meier estimator Kaplan and Meier (1958). The variant, BJ-EN, is proposed in Wang et al. (2008) for high-dimensional data.

Recently, machine learning techniques are adopted for time-to-event estimation problems Wang et al. (2017b). A survival tree model was proposed in Gordon and Olshen (1985), in which the Wasserstein metric was used to estimate the homogeneity and as the choice of splitting criterion. Support vector machines (SVM) were also adopted for survival analysis. In Khan and Zubek (2008), a SVM-based model was proposed for censored data by using an updated asymmetric loss function. In Van Belle et al. (2007), an SVM-based model was proposed using a health index as a proxy for the censored time. In Van Belle et al.

(2011), an SVM-based model was proposed to incorporate ranking and regression to solve the time-to-event estimation problem. Ensemble models were also adopted to solve the time-to-event estimation problem. In Ishwaran et al. (2011), a random survival forests was proposed to use survival trees Gordon and Olshen (1985) as weak learners. In Hothorn et al. (2005), a boosting algorithm was proposed to incorporate censored data. An active regularized cox regression model was proposed Vinzamuri et al. (2014) to use an active learning algorithm to incorporate the Cox model by using a discriminative gradient sampling strategy. A transfer learning survival analysis model was proposed Li et al. (2016b) to improve the Cox PH model by transferring knowledge from the source domain to the target domain in the context of survival analysis. Multi-task learning model for survival analysis (MTLSA) Li et al. (2016a) model introduced an indicator matrix which allows it to take the advantage of multi-task learning and be able to simultaneously learn from both uncensored and censored examples. In Ranganath et al. (2016), a hierarchical generative approach to survival analysis in the context of the Electronic Health Records was proposed. In Chapfuwa et al. (2018), a time-to-event model as proposed to focus on the estimation of time-to-event distributions by using adversarial generative approach.

## 4.4 Method

### 4.4.1 Notations

In this paper, scalar variables are denoted by letters (e.g.,  $t$  and  $N$ ), vector variables are represented by boldface letters (e.g.,  $\mathbf{x}$ ), matrix variables are represented by boldface uppercase letters (e.g.,  $\mathbf{V}$ ). The  $j$ -th column vector and the  $i$ -th row vector of  $\mathbf{V}$  are denoted by  $\mathbf{V}_{:j}$  and  $\mathbf{V}_{i,:}$ , respectively. We list the main symbols used in our subsequent derivations in Table 4.1.

Notation	Definition
$n$	The number of examples in a dataset
$m$	The dimensionality of a feature vector
$d$	The dimensionality of a time vector
$\mathcal{T}$	$\mathcal{T} = \{t_1, t_2, \dots, t_j, \dots, t_{ \mathcal{T} }\}$ , the set of distinct time points in the dataset
$\mathbf{x}_i \in \mathbb{R}^m$	The feature vector of the $i$ -th instance
$y_i \in \mathcal{T}$	The last observation time of the $i$ -th example
$s_i \in \{0, 1\}$	The status of the $i$ -th example in its last observation (0: no event; 1: event)
$\mathbf{V} \in \mathbb{R}^{d \times  \mathcal{T} }$	A matrix whose $j$ -th column vector, $\mathbf{V}_{:j} \in \mathbb{R}^d$ , is the concept vector of time point, $t_j$
$\mathbf{W} \in \mathbb{R}^{d \times m}$	A linear transformation matrix

Table 4.1: Notations and definitions.

#### 4.4.2 Problem formulation

In the time-to-event model learning, each example in the dataset is represented by a triplet,  $(\mathbf{x}_i, y_i, s_i)$  with  $i \in \{1, 2, \dots, n\}$  being the index of the example and  $n$  being the number of examples. Within each triplet,  $\mathbf{x}_i \in \mathbb{R}^m$  denotes a  $m$ -dimensional feature vector,  $y_i \in \mathcal{T}$  denotes the last observation time, and  $s_i \in \{0, 1\}$  denotes the status of the example at its last observation time. If  $s_i = 1$ , the target event occurs at time  $y_i$  and subsequent observations are not necessary. If  $s_i = 0$ , the target event has not occurred up to time  $y_i$ , and subsequent observations are not available; thus, we call this example censored, and the event may occur at anytime  $t > y_i$ . The symbol,  $\mathcal{T} = \{t_1, t_2, \dots, t_{|\mathcal{T}|}\}$  with  $|\mathcal{T}| < \infty$ , denotes the set of all possible event occurrence times. In this context, the time-to-event

task is to learn a probabilistic model,

$$p(y|\mathbf{x}, \boldsymbol{\theta}),$$

with model parameter,  $\boldsymbol{\theta}$ .

#### 4.4.3 Objective Function Formulation

##### *Time-to-event probability*

In our model, we compute the time-to-event probability, i.e., the probability that “the target event occurs at time  $t_j$ ”, by

$$\begin{aligned} p(y = t_j|\mathbf{x}, \boldsymbol{\theta}) &= p(y = t_j|\mathbf{x}, \mathbf{W}, \mathbf{V}) \\ &= \frac{\exp(-\|\mathbf{W}\mathbf{x} - \mathbf{V}_{:j}\|_2^2)}{\sum_{j'=1}^{|\mathcal{T}|} \exp(-\|\mathbf{W}\mathbf{x} - \mathbf{V}_{:j'}\|_2^2)}, \end{aligned} \quad (4.1)$$

with model parameter  $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{V}\}$ , where  $\mathbf{W} \in \mathbb{R}^{d \times m}$  denotes the linear transformation matrix which maps a vector from the  $m$ -dimensional feature space to the  $d$ -dimensional time concept vector space;  $\mathbf{V} \in \mathbb{R}^{d \times |\mathcal{T}|}$  denotes a matrix whose  $j$ -th column vector,  $\mathbf{V}_{:j}$ , is the vector representation of the event occurrence time,  $t_j$  with  $j = 1, 2, \dots, |\mathcal{T}|$ .

When an example is censored, the target event’s exact occurrence time is unknown, therefore (4.1) is not appropriate for calculating the time-to-event probability. Instead, since we know the censoring time and the fact that the event has not occurred up to that time, we can model the probability of “the target event will occur after time  $t_j$ ” by

$$\begin{aligned} p(y > t_j|\mathbf{x}, \boldsymbol{\theta}) &= p(y > t_j|\mathbf{x}, \mathbf{W}, \mathbf{V}) \\ &= \frac{\sum_{k=j+1}^{|\mathcal{T}|} \exp(-\|\mathbf{W}\mathbf{x} - \mathbf{V}_{:k}\|_2^2)}{\sum_{j'=1}^{|\mathcal{T}|} \exp(-\|\mathbf{W}\mathbf{x} - \mathbf{V}_{:j'}\|_2^2)}. \end{aligned} \quad (4.2)$$

##### *Optimization Objective*

For notation simplicity,  $q_i$ , defined by (4.2), is used to denote the event probability for the  $i$ -th example if it is censored, otherwise,  $p_i$ , defined by (4.1), is used to denote the event

probability for the  $i$ -th example. Using the short-hand notations,  $p_i$  and  $q_i$ , the negative log-likelihood based on the dataset can be written as

$$\ell(\mathbf{W}, \mathbf{V}) = \sum_{i=1}^n -(\mathbb{1}(s_i = 1) \log p_i + \mathbb{1}(s_i = 0) \log q_i),$$

where  $\mathbb{1}(\cdot)$  is an indicator function whose returned value is 1 if the condition is met, otherwise 0. It is important to note that the negative log-likelihood is a function of both the linear transformation matrix,  $\mathbf{W}$ , and the time concept vector matrix  $\mathbf{V}$ ; therefore, minimizing the negative log-likelihood could provide an optimal solution for our model. To avoid overfitting and to enhance the generalization of the model, we apply several regularization terms (note: detailed discussions on the regularization terms are provided in the following section) and propose to solve the following minimization problem:

$$\begin{aligned} \underset{\mathbf{W}, \mathbf{V}}{\text{minimize}} \quad & F(\mathbf{W}, \mathbf{V}) = \ell(\mathbf{W}, \mathbf{V}) + \lambda_1 \|\mathbf{W}\|_{2,1} \\ & + \lambda_2 \|\mathbf{V}\|_* + \frac{\lambda_3}{2} \|\mathbf{VD}\|_F^2, \end{aligned} \tag{4.3}$$

in which,  $\|\cdot\|_{2,1}$  denotes the  $L_{2,1}$ -norm,  $\|\cdot\|_*$  denotes the nuclear norm,  $\|\cdot\|_F$  denotes the Frobenius norm, and  $\{\lambda_1, \lambda_2, \lambda_3\}$  are the trade-off parameters. The definition of the matrix,  $\mathbf{D}$ , will be given in the following section.

### *Regularization*

The  $L_{2,1}$ -norm is known for inducing column sparsity. Therefore, in addition to preventing overfitting, the  $L_{2,1}$ -norm regularizer also acts as a feature-selection mechanism. In practice, especially in the time-to-event estimation tasks in the medical domain, the feature vectors are often very high-dimensional. Being able to select the most relevant features can not only improve the performance but also enhance the interpretability of the model.

The nuclear norm is a low-rank matrix inducing regularizer. It has a high utility value in our model. First, because the time concept vector dimensionality is unknown in our optimization algorithm, we intend to over-estimate the dimensionality in initialization.

Having a low-rank inducing regularizer could help determine the effective dimensionality. Second, the nuclear norm regularizer may help generate structures in the time concept vector space; i.e., similar time concepts vectors reside in the same linear subspace.

The Frobenius norm is used to enhance the temporal smoothness among time concept vectors. The time concept vectors corresponding to two consecutive time points are similar, and thus the cumulative differences of all consecutive time points should be small. The cumulative difference square is represented by

$$\sum_{j=1}^{|\mathcal{T}|-1} \|\mathbf{V}_{:j} - \mathbf{V}_{:j+1}\|_2^2 = \|\mathbf{V}\mathbf{D}\|_F^2,$$

where the matrix  $\mathbf{D} \in \mathbb{R}^{|\mathcal{T}| \times (|\mathcal{T}|-1)}$  with the  $(i,j)$ -th element being

$$D_{ij} = \begin{cases} 1, & i = j \\ -1, & i = j + 1 \\ 0, & \text{otherwise.} \end{cases}$$

#### 4.4.4 Optimization

There are two challenges to optimize our proposed objective function: first, the objective function is non-smooth; therefore, conventional gradient-based approaches (e.g. conjugate gradient methods) are not applicable; second, multiple variables (i.e.  $\mathbf{W}$  and  $\mathbf{V}$ ) are required to optimize, and their terms are not separable.

To address these challenges in this optimization problem, we propose an iterative proximal algorithm. In our proposed algorithm, we formulate two non-smooth sub-problems: in the first sub-problem, the time concept vector matrix,  $\mathbf{V}$ , is treated as a constant, and a proximal algorithm is applied to solve the optimal  $\mathbf{W}$ ; in the second problem the linear transformation matrix,  $\mathbf{W}$ , is treated as a constant, and a proximal algorithm is applied to solve the optimal  $\mathbf{V}$ . Then these two sub-problems are solved repeatedly until the overall objective converges. As the procedures for solving these two sub-problems are similar, we will give a detailed description on solving the first sub-problem, and a similar procedure

can be used to solve the second sub-problem.

*Sub-problem 1: solving  $\mathbf{W}$ , with  $\mathbf{V}$  as a constant*

By treating  $\mathbf{V}$  as a constant, the original optimization problem with objective function (4.3) is reduced to

$$\min_{\mathbf{W}} F_1(\mathbf{W}) = \ell(\mathbf{W}) + \lambda_1 \|\mathbf{W}\|_{2,1}. \quad (4.4)$$

It is important to note that the objective function  $F_1(\mathbf{W})$  is convex because its first term (a log-softmax function) and second term (a norm) are both convex. However, this problem is challenging due to the non-smoothness in the  $L_{2,1}$ -norm regularization. To solve this problem, we make use of an auxiliary function

$$\begin{aligned} Q(\mathbf{W}, \mathbf{W}') &= \ell(\mathbf{W}') + \text{tr}((\mathbf{W} - \mathbf{W}')^\top \Delta \ell(\mathbf{W}')) \\ &\quad + \frac{L}{2} \|\mathbf{W} - \mathbf{W}'\|_F^2 + \lambda_1 \|\mathbf{W}\|_{2,1}. \end{aligned} \quad (4.5)$$

**Proposition 1.** *Let  $L(\ell)$  denote the Lipschitz constant of  $\ell(\mathbf{W})$ , then for any  $L \geq L(\ell)$ , we have*

$$F_1(\mathbf{W}) \leq Q(\mathbf{W}, \mathbf{W}').$$

This is straightforward to prove based on the convexity and the smoothness of  $\ell(\mathbf{W})$  and the definition of the Lipschitz constant. It is also straightforward to verify

$$Q(\mathbf{W}', \mathbf{W}') = F_1(\mathbf{W}') \quad \text{and} \quad Q(\mathbf{W}, \mathbf{W}) = F_1(\mathbf{W}).$$

Therefore, letting  $\mathbf{W}' = \mathbf{W}^t$  and

$$\mathbf{W}^{t+1} = \arg \min_{\mathbf{W}} Q(\mathbf{W}, \mathbf{W}^t), \quad (4.6)$$

we arrive at the following relations:

$$F_1(\mathbf{W}^{t+1}) \leq Q(\mathbf{W}^{t+1}, \mathbf{W}^t) \leq Q(\mathbf{W}^t, \mathbf{W}^t) = F_1(\mathbf{W}^t).$$

Namely, (4.6) can be used as an updating rule for optimizing the objective function,  $F_1(\mathbf{W})$ . By completing the square in (4.5), we can find that (4.6) is equivalent to

$$\begin{aligned} \mathbf{W}^{t+1} &= \\ \arg \min_{\mathbf{W}} \frac{L}{2} \|\mathbf{W} - (\mathbf{W}^t - \frac{1}{L} \Delta \ell(\mathbf{W}^t))\|_F^2 + \lambda_1 \|\mathbf{W}\|_{2,1} & \quad (4.7) \\ &= \text{prox}_{\frac{\lambda_1}{L} \|\cdot\|_{2,1}}(\mathbf{W}^t - \frac{1}{L} \Delta \ell(\mathbf{W}^t)) \end{aligned}$$

where  $\text{prox}_{\frac{\lambda_1}{L} \|\cdot\|_{2,1}}$  is the proximal operator Parikh and Boyd (2013) of  $\frac{\lambda_1}{L} \|\cdot\|_{2,1}$ . Knowing the analytical form of the proximal operator for the scaled  $L_{2,1}$ -norm,  $\alpha \|\cdot\|_{2,1}$ , being

$$\text{prox}_{\alpha \|\cdot\|_{2,1}}(\mathbf{A}) = \mathbf{A}_{:j} \left(1 - \frac{\alpha}{\|\mathbf{A}_{:j}\|_2}\right)_+, \quad \forall j, \quad (4.8)$$

and combining (4.7) and (4.8), we obtain the updating rule for  $\mathbf{W}_{:j}^{t+1}$ ,  $\forall j$

$$\begin{aligned} \mathbf{W}_{:j}^{t+1} &= \\ (\mathbf{W}^t - \frac{1}{L} \Delta \ell(\mathbf{W}^t))_{:j} \left(1 - \frac{\lambda_1}{L \|\mathbf{W}^t - \frac{1}{L} \Delta \ell(\mathbf{W}^t)\|_{2,j}}\right)_+, & \quad (4.9) \end{aligned}$$

where  $(\cdot)_+ = \max(\cdot, 0)$ . The procedure for solving sub-problem 1 is described in Algorithm 2.

---

**Procedure 2** Sub-problem 1: solving  $\mathbf{W}$ , while treating  $\mathbf{V}$  as a constant

---

- 1: initialize  $\mathbf{W}^0$
  - 2:  $t \leftarrow 0$
  - 3: **repeat**
  - 4:      $\mathbf{W}_{:j}^{t+1} \leftarrow (\mathbf{W}_{:j}^t - \frac{1}{L} \Delta \ell(\mathbf{W}^t))_{:j} \left(1 - \frac{\lambda_1}{L \|\mathbf{W}^t - \frac{1}{L} \Delta \ell(\mathbf{W}^t)\|_{2,j}}\right)_+, \quad \forall j$
  - 5:      $t++$
  - 6: **until**  $t < \text{max\_iter}$  **and not** converge
  - 7: **Return**  $\mathbf{W}$
-

*Sub-problem 2: solving  $\mathbf{V}$ , with  $\mathbf{W}$  as a constant*

By treating  $\mathbf{W}$  as a constant, the original optimization problem with objective function (4.3) is reduced to

$$\min_{\mathbf{V}} F_2(\mathbf{V}) = \ell(\mathbf{V}) + \lambda_2 \|\mathbf{V}\|_* + \frac{\lambda_3}{2} \|\mathbf{V}\mathbf{D}\|_F^2. \quad (4.10)$$

Following the derivations as in solving sub-problem 1, we arrive at the updating rule of  $\mathbf{V}$  in a proximal operator form:

$$\mathbf{V}^{t+1} = \text{prox}_{\frac{\lambda_2}{L} \|\cdot\|_*} \left( \mathbf{V}^t - \frac{1}{L} (\Delta\ell(\mathbf{V}^t) + \lambda_3 \mathbf{V}^t \mathbf{D} \mathbf{D}^\top) \right). \quad (4.11)$$

The proximal operator of a scaled nuclear norm Parikh and Boyd (2013),  $\alpha \|\cdot\|_*$ , is

$$\text{prox}_{\alpha \|\cdot\|_*}(\mathbf{A}) = \mathbf{U} \text{diag}(\text{prox}_{\alpha \|\cdot\|_1}(\sigma(\mathbf{A}))) \mathbf{R}^\top, \quad (4.12)$$

where  $\sigma(\mathbf{A})$  denotes the vector of singular values of a matrix,  $\mathbf{A}$ ;  $\{\mathbf{U}, \text{diag}(\sigma(\mathbf{A})), \mathbf{R}^\top\}$  is the singular value composition of  $\mathbf{A}$ , and

$$\text{prox}_{\alpha \|\cdot\|_1}(\mathbf{a}) = (\mathbf{a} - \alpha \mathbf{1})_+ - (-\mathbf{a} - \alpha \mathbf{1})_+ \quad (4.13)$$

is the proximal operator for the scaled vector  $L_1$ -norm. The symbol,  $\mathbf{1}$ , denotes a one-vector with an appropriate dimension. Combining (4.11), (4.12), and (4.13), the updating rule of  $\mathbf{V}$  can be written as

$$\mathbf{V}^{t+1} = \mathbf{U} \text{diag} \left( \left( \sigma(\hat{\mathbf{V}}) - \frac{\lambda_2}{L} \mathbf{1} \right)_+ - \left( -\sigma(\hat{\mathbf{V}}) - \frac{\lambda_2}{L} \mathbf{1} \right)_+ \right) \mathbf{R}^\top, \quad (4.14)$$

where  $\{\mathbf{U}, \text{diag}(\sigma(\hat{\mathbf{V}})), \mathbf{R}^\top\}$  is the singular value decomposition of  $\mathbf{V}^t - \frac{1}{L} (\Delta\ell(\mathbf{V}^t) + \lambda_3 \mathbf{V}^t \mathbf{D} \mathbf{D}^\top)$ . The procedure of solving sub-problem 2 is given in Algorithm 3.

To solve the optimization problem with the objective function (4.3) involving both variables  $\mathbf{W}$  and  $\mathbf{V}$ , we propose to use Algorithm 4, in which sub-problems 1 and sub-problem 2 are solved iteratively until a convergence is reached.

---

**Procedure 3** Sub-problem 2: solving  $\mathbf{V}$ , while treating  $\mathbf{W}$  as a constant

---

- 1: initialize  $\mathbf{V}^0$
  - 2:  $t \leftarrow 0$
  - 3: **repeat**
  - 4:  $\{\mathbf{U}, \text{diag}(\sigma(\hat{\mathbf{V}})), \mathbf{R}^\top\} \leftarrow SVD(\mathbf{V}^t - \frac{1}{L}(\Delta\ell(\mathbf{V}^t) + \lambda_3 \mathbf{V}^t \mathbf{D} \mathbf{D}^\top))$
  - 5:  $\mathbf{V}^{t+1} = \mathbf{U} \text{diag}((\sigma(\hat{\mathbf{V}}) - \frac{\lambda_2}{L} \mathbf{1})_+ - (-\sigma(\hat{\mathbf{V}}) - \frac{\lambda_2}{L} \mathbf{1})_+) \mathbf{R}^\top$
  - 6:  $t++$
  - 7: **until**  $t < \text{max\_iter}$  **and not** converge
  - 8: **Return**  $\mathbf{V}$
- 

---

**Procedure 4** An iterative proximal algorithm for solving (4.3)

---

- 1: initialize  $\mathbf{W}^0$  and  $\mathbf{V}^0$
  - 2: **repeat**
  - 3: Solve sub-problem 1 using Algorithm 2
  - 4: Solve sub-problem 2 using Algorithm 3
  - 5: **until** converge
  - 6: **Return**  $\mathbf{W}$  and  $\mathbf{V}$
- 

## 4.5 Experiments

In this section, we describe our conducted experiments on the proposed model. The first experiment focuses on demonstrating the effectiveness of the proposed model by comparing the performance with those from state-of-the-art models on several benchmark datasets for the time-to-event estimation tasks. In the second experiment, we visualize the learned time concept vectors and show a few interesting observations. The third experiment focuses on empirically characterizing the scalability of the proposed model by running on a number of synthetic learning tasks in different settings.

### 4.5.1 Setup

The proposed model is implemented in Python 3.6.6 based on the *Numpy* package. We also use the *Autograd* package for computing the derivatives; i.e.,  $\Delta\ell(\mathbf{W}^t)$  in (4.7) and  $\Delta\ell(\mathbf{V}^t)$  in (4.11). The code runs on a workstation with an Intel Core i7 CPU 4GHz

and 32 GB of RAM. Each of the hyper-parameters  $\{\lambda_1, \lambda_2, \lambda_3\}$  were determined by grid searches in values  $\{0, 0.1, 1.0\}$  using cross-validation within the training set. The time concept vector dimensionality,  $d$ , is set to  $d = 5|\mathcal{T}|$ , where  $\mathcal{T}$  is the set of event times in the training set, and the effective dimensionality of the time concept vector will be automatically determined by the regularizations.

#### 4.5.2 Experiment #1: comparisons to state-of-the-art models on benchmark datasets

In this experiment, we compare the performance, in terms of the concordance index, of the proposed model with the state-of-the-art baseline models on several benchmark datasets for the time-to-event estimation tasks.

##### *Datasets*

The 7 benchmark datasets used in this experiment are publicly available for download. Details information of the datasets and the task descriptions are provided as the follows:

- *Van de Vijver's Microarray Breast Cancer data (VDV)* Van't Veer et al. (2002) contains 4,707 gene expression values on 78 (44 censored) breast cancer patients for predicting occurrence of death in terms of year, up to 13 years.
- *Gene-expression profiles of lung adenocarcinoma (Lung)* Beer et al. (2002) contains 7,129 gene expression values on 86 (62 censored) early-stage lung adenocarcinoma patients for predicting occurrence of death in terms of month, up to 110 months.
- *Mantle Cell Lymphoma (MCL)* Rosenwald et al. (2003) contains 8,810 gene expression values on 92 (28 censored) MCL patients for predicting occurrence of death in terms of year, up to 14 years.
- *Norway/Stanford breast cancer data (NSBCD)* Sørli et al. (2003) contains 549 gene expression values on 115 (77 censored) breast cancer women for predicting occurrence of death in terms of month, up to 188 months.

- *Adult myeloid leukemia (AML)* Bullinger et al. (2004) contains 6,283 gene expression values on 116 (77 censored) AML patients for predicting occurrence of death in terms of month, up to 54 months.
- *Diffuse Large B-Cell Lymphoma (DLBCL)* Lossos (2008) contains 7,399 gene expression values on 240 (102 censored) DLBCL patients for predicting occurrence time of death in terms of year, up to 21 years.
- *Dutch Breast Cancer Data (DBCD)* van Houwelingen et al. (2006) contains 4,919 gene expression values on 295 (216 censored) breast cancer women for predicting occurrence time of death in terms of year, up to 18 years.

The 7 benchmark datasets are summarized in Table 4.2. To have a fair comparisons to the baseline models, we adopt the evaluation settings in Li et al. (2016a), in which 5-fold cross-validation is used when the number of examples is greater than 150 and 3-fold cross-validation otherwise.

Dataset	# example	# censored	# feature	# time
VDV	78	44	4705	13
Lung	86	62	7129	110
MCL	92	28	8810	14
NS-BCD	115	77	549	188
AML	116	49	6283	54
DL-BCL	240	102	7399	21
DBCD	295	216	4919	18

Table 4.2: Summaries of the 7 benchmark datasets (ordered by # example).

### *Baseline models*

Based on the popularity and accessibility, we compared our proposed model to 16 baseline models.

*Cox based models* The Cox proportional hazards model Cox (1992) is the most commonly used survival analysis model. Besides the basic formulation, the  $L_1$ -norm regularized variant, *LASSO-Cox* Tibshirani (1997), and the elastic net regularized variant, *EN-Cox* Simon et al. (2011), are also widely used.

*Censored regression models* Standard likelihood function estimation incorporates censored examples Lee and Wang (2003). Based on the assumptions of the underlining distributions, it has four variants: Weibull, Logistic, Log-Logistic, and Log-Gaussian.

*Linear models* Tobit model Tobin (1958) is an extension of linear regression that incorporates censored examples with parameters estimated by the maximum likelihood method rather than using least squares error. Buckley-James regression Wang et al. (2008) (BJ-EN), is a linear model which incorporates censored examples by estimating the censored value using the Kaplan-Meier estimator Kaplan and Meier (1958) with elastic net regularization.

*Pairwise ranking based models* Boosting concordance index Mayr and Schmid (2014) (Boost-CI) is a gradient boosting algorithm to optimize the smoothed version of the concordance index.

*Multi-task learning models* *MTLSA* is multi-task formulation tailored for handling censored examples Li et al. (2016a) by introducing an indication table and a non-negative non-increasing list structure constraint.

*Others* *SurvGB* Hothorn et al. (2005) is a gradient boosting model for survival analysis. *SurvSVM-Linear* Pölsterl et al. (2015) is a rank and regression based SVM adoption for survival analysis using a linear kernel. *SurvSVM-RBF* Pölsterl et al. (2016) is an efficient SVM-based survival analysis model using the radial basis function kernel. *BJ-Neural* is

a 2-layer neural network with Rectified Linear Unit Activation functions to optimized an objective of Buckley-James regression Wang et al. (2008).

*Performance metric*

Due to the presence of censored examples, concordance index (CI or c-index) Heagerty and Zheng (2005) is used to evaluate the performance of time-to-event estimation models.

The CI of a model,  $g$ , is defined as Steck et al. (2008):

$$\text{IC}(g) = \frac{1}{num} \sum_{s_i=1} \sum_{y_j > y_i} \mathbb{1}(g(\mathbf{x}_j) > g(\mathbf{x}_i)),$$

where  $\mathbb{1}(\cdot)$  is an indication function, and  $num$  is a normalization factor such that  $\text{CI}(g) \leq 1$ , and  $\text{CI}(g) = 1$  is the best possible performance. The CI’s and corresponding standard deviations of all baseline models and the proposed model on the benchmark datasets are shown in Table 4.3. The results of *SurvSVM*’s and *SurvGB* were obtained by using the the functions in the *Scikit-Survival* package. The result of *BJ-Neural* was obtained by using the *Scikit-Learning* package. Results of the other baseline models are adopted from Li et al. (2016a).

Figure 4.2 shows the Critical Difference Diagram Demšar (2006) of Experiment #1. The proposed method achieves the best overall ranking. Models with regularizations, i.e., *proposed*, *MTLSA*, *BJ-EN*, *CoxEN*, and *CoxLasso* have the best overall rankings; this may suggest that in datasets with relatively small number of examples, regularization is very important to prevent overfitting.

*4.5.3 Experiment #2: Regimes identification by visualizing the learned time concept vectors*

The proposed learning framework does not only learn the model parameters (i.e. matrix  $\mathbf{W}$ ) but also learns the time concept vectors (i.e. column vectors in matrix,  $\mathbf{V}$ ). In this experiment, we visualize the learned time concept vectors and point out some interesting findings based on cluster formations. To generate time concept vectors, we apply the

	VDV	Lung	MCL	NSBCD	AML	DLBCL	DBCDC
CoxPH	0.597 (7) ± 0.011	0.516 (13) ± 0.133	0.577 (11) ± 0.059	0.441 (12) ± 0.059	0.552 (8) ± 0.068	0.455 (13) ± 0.072	0.554 (11) ± 0.123
CoxLasso	0.648 (4) ± 0.028	0.670 (3) ± 0.091	0.682 (8) ± 0.07	0.591 (10) ± 0.109	0.600 (4) ± 0.031	0.634 (4) ± 0.042	0.688 (8) ± 0.043
CoxEN	0.642 (5) ± 0.068	0.665 (4) ± 0.07	0.673 (9) ± 0.073	0.605 (9) ± 0.1	0.572 (6) ± 0.06	0.649 (3) ± 0.039	0.721 (4) ± 0.031
Logistic	0.528 (8) ± 0.14	0.571 (11) ± 0.094	0.483 (12) ± 0.068	0.379 (13) ± 0.02	0.454 (15) ± 0.077	0.484 (12) ± 0.05	0.491 (13) ± 0.087
Weibull	0.316 (16) ± 0.132	0.429 (15) ± 0.01	0.474 (14) ± 0.075	0.305 (15) ± 0.153	0.529 (12) ± 0.055	0.251 (16) ± 0.063	0.456 (16) ± 0.105
Log-Gaussian	0.521 (10) ± 0.165	0.412 (16) ± 0.075	0.256 (16) ± 0.072	0.444 (11) ± 0.054	0.405 (16) ± 0.065	0.317 (15) ± 0.091	0.488 (14) ± 0.055
Log-Logistic	0.527 (9) ± 0.107	0.592 (9) ± 0.066	0.480 (13) ± 0.072	0.238 (16) ± 0.05	0.468 (14) ± 0.08	0.425 (14) ± 0.124	0.526 (12) ± 0.023
Tobit	0.519 (11) ± 0.158	0.469 (14) ± 0.136	0.459 (15) ± 0.032	0.373 (14) ± 0.021	0.473 (13) ± 0.076	0.497 (11) ± 0.053	0.487 (15) ± 0.076
BJ-EN	0.608 (6) ± 0.065	0.665 (4) ± 0.132	0.723 (3) ± 0.11	0.622 (8) ± 0.092	0.650 (3) ± 0.059	0.629 (5) ± 0.073	0.709 (6) ± 0.039
Boost-CI	0.665 (3) ± 0.059	0.571 (11) ± 0.093	0.705 (5) ± 0.096	0.626 (7) ± 0.083	0.582 (5) ± 0.05	0.608 (7) ± 0.03	0.710 (5) ± 0.043
MTLSA	0.701 (1) ± 0.033	0.633 (8) ± 0.075	0.727 (2) ± 0.096	0.682 (3) ± 0.045	0.715 (2) ± 0.049	0.653 (2) ± 0.071	0.758 (2) ± 0.03
SurvGB	0.509 (12) ± 0.085	0.586 (10) ± 0.07	0.650 (10) ± 0.048	0.634 (5) ± 0.089	0.539 (11) ± 0.051	0.564 (10) ± 0.056	0.665 (10) ± 0.043
SurvSVM-Linear	0.481 (14) ± 0.055	0.706 (1) ± 0.075	0.719 (4) ± 0.059	0.647 (4) ± 0.052	0.552 (8) ± 0.044	0.599 (8) ± 0.039	0.709 (6) ± 0.027
SurvSVM-RBF	0.491 (13) ± 0.05	0.656 (6) ± 0.053	0.704 (6) ± 0.063	0.725 (1) ± 0.04	0.566 (7) ± 0.029	0.611 (6) ± 0.025	0.753 (3) ± 0.033
BJ-Neural	0.467 (15) ± 0.073	0.643 (7) ± 0.066	0.687 (7) ± 0.075	0.628 (6) ± 0.042	0.542 (10) ± 0.047	0.579 (9) ± 0.037	0.679 (9) ± 0.038
Proposed	0.681 (2) ± 0.128	0.685 (2) ± 0.035	0.742 (1) ± 0.038	0.712 (2) ± 0.032	0.719 (1) ± 0.042	0.669 (1) ± 0.068	0.759 (1) ± 0.081

Table 4.3: Concordance Indices (CI’s) and corresponding standard deviations of all the models; The ranking of a model among all compared models in each dataset is included.

proposed model to each of the benchmark datasets using all available examples within a dataset. Using the best hyper-parameter settings found in *Experiment #1*. After each learning process is finished, the learned time concept vectors, which are high-dimensional, are projected on to a 2-dimensional space using t-SNE Maaten and Hinton (2008) for visualization. The 2D projected time concept vectors are indicated by their corresponding event occurrence times. The two axes in the 2D plot usually have no specific meanings, but the projections using t-SNE preserve the time concept vector distribution; in other words, time concept vectors that are in the same manifold in the high-dimensional space will be projected to the same cluster in the low-dimensional space.

Clusters can be easily identified in each scatter plot. For example, three major clusters

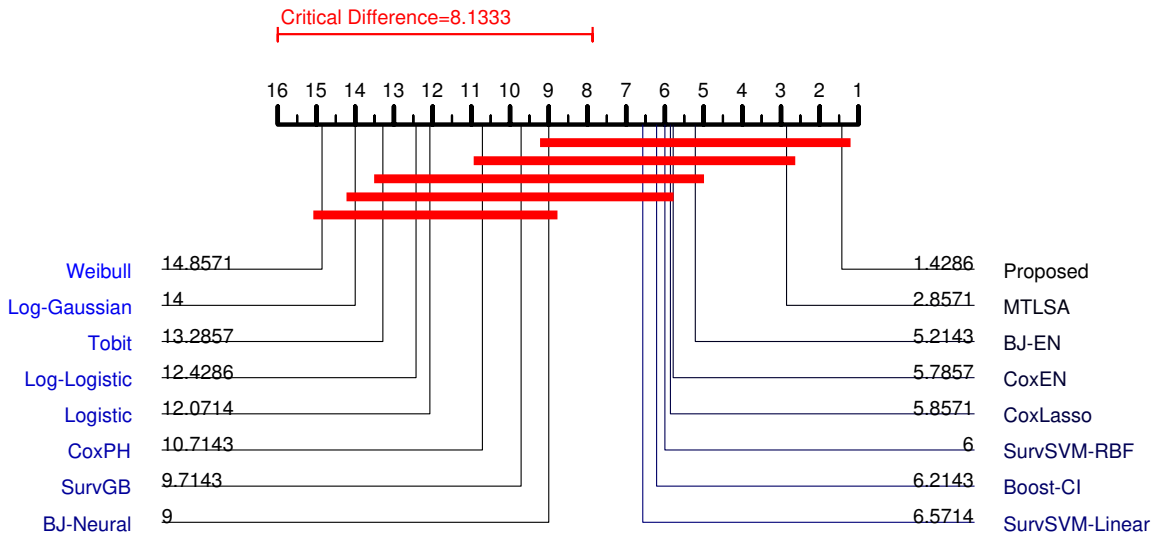


FIGURE 4.2: The Critical Difference Diagram shows the average rankings and overall rankings of all the compared models in the measure of Concordance Index.

can be identified in the *Lung* dataset (Figure 4.4) and two clusters can be identified in the *AML* dataset (Figure 4.7). The cluster formations reveal the time regime information. For example, in the *AML* dataset, one cluster consists of event occurrence time points 1 to 10, and another cluster consists of time points 11 and above; this may suggest that when a patient is estimated to have a survival time in-between 1 month and 10 months, they are not likely to achieve a complete remission, or are likely to suffer fatal complications of therapy. If a patient is estimated to have a survival time 11 months or above, the patient is more likely to achieve a durable complete remission. Another interesting finding is that early occurrence time concepts are able to form a cluster with the late occurrence time concepts. For example, in the plot of the *DBCD* time concepts, the early time concepts (0, 1, 2) are in the same cluster as late occurrence time concepts (11 and above), and time concepts 3-10 form a second cluster. Clinical experience bears this out: some patients will present very critically ill, with most known prognostic factors not in their favor, yet they survive and achieve a durable complete remission.

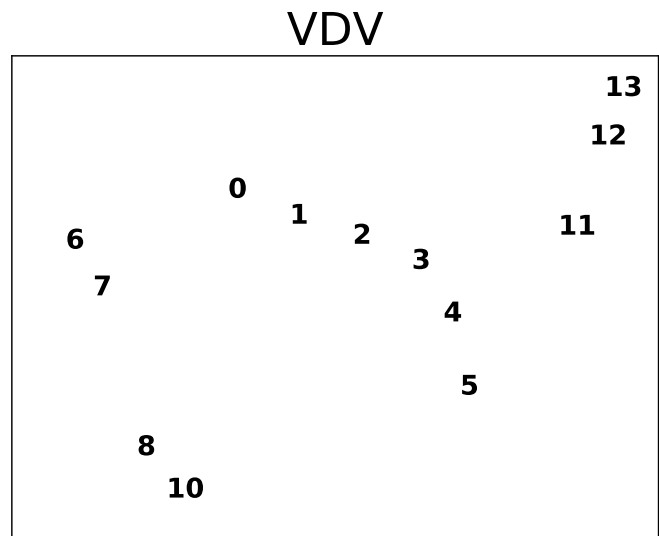


FIGURE 4.3: 2D scatter plots of the learned time concept vectors in the VDV datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes.

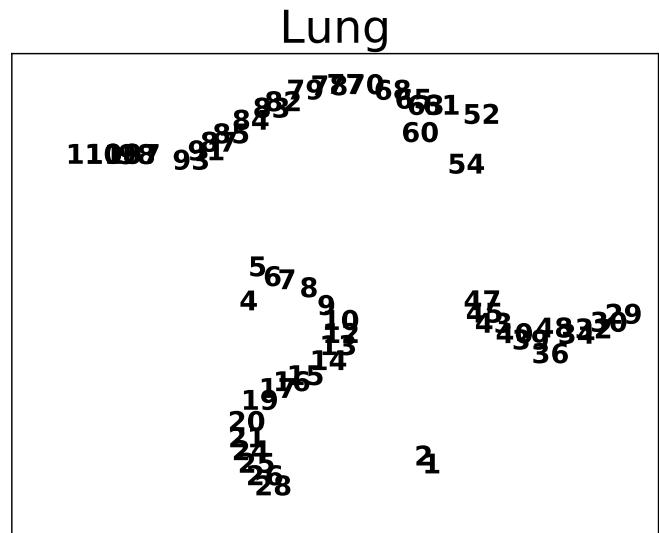


FIGURE 4.4: 2D scatter plots of the learned time concept vectors in the VDV datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes.

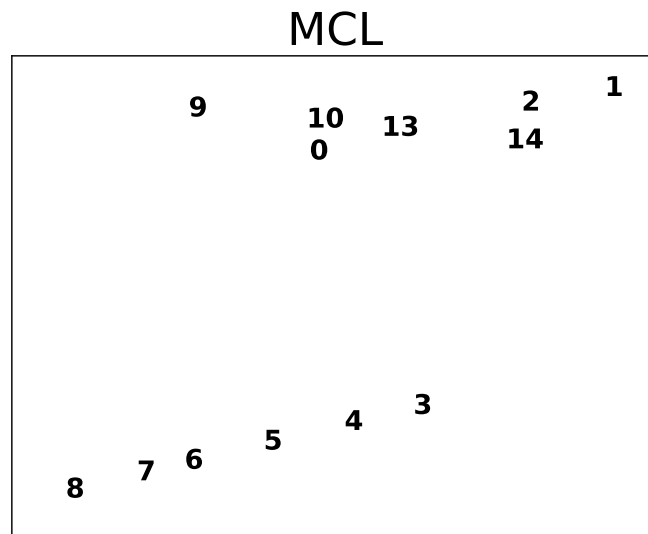


FIGURE 4.5: 2D scatter plots of the learned time concept vectors in the MCL datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes.

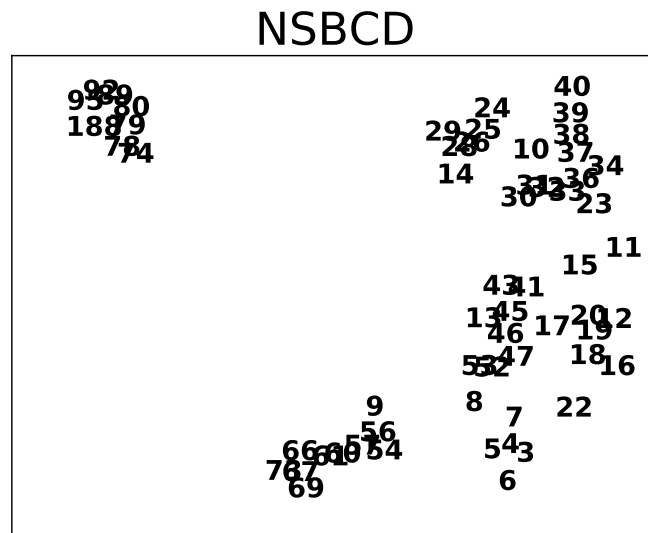


FIGURE 4.6: 2D scatter plots of the learned time concept vectors in the NSBCD datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes.

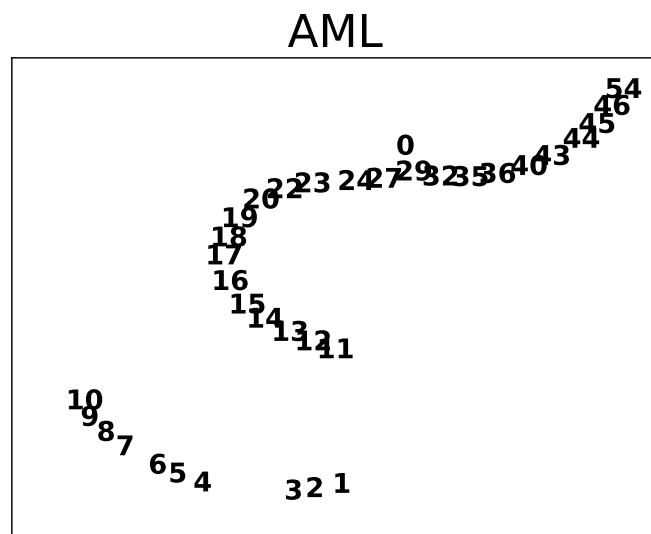


FIGURE 4.7: 2D scatter plots of the learned time concept vectors in the AML datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes.

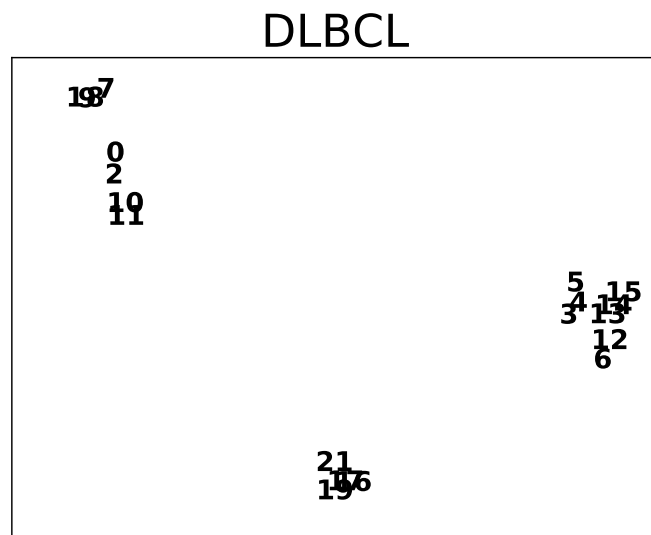


FIGURE 4.8: 2D scatter plots of the learned time concept vectors in the DLBCL datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes.

## DBCD

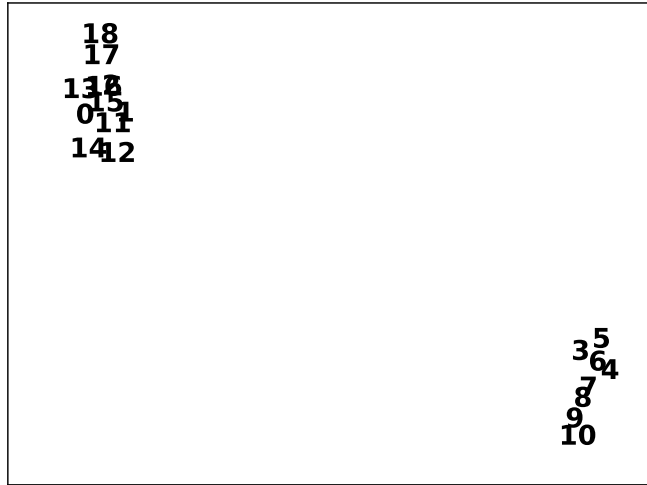


FIGURE 4.9: 2D scatter plots of the learned time concept vectors in the DBCD datasets. The numbers in each plot indicate the corresponding time concepts, and their cluster formations may help in revealing time regimes.

### 4.5.4 Experiment #3: model scalability evaluation on synthetic datasets

We empirically evaluate the scalability of the proposed model with respect to the number of examples ( $n$ ), the number of features ( $m$ ), the number of possible event times ( $T$ ), and the dimensionality of the time concept vector space ( $d$ ). In the synthetic dataset, the feature vectors ( $\mathbf{x}_i$ 's) are generated by using a uniform distribution in the  $(-1, 1)$  interval, the event occurrence times ( $y_i$ 's) are generated by using a discrete uniform distribution between one and the number of possible event occurrence times, and the event status ( $s_i$ 's) are generated by a Bernoulli distribution with mean 0.5. Multiple runs are conducted for each  $(n, m, T, d)$  setting; however, because the learning time variations of the runs within the same setting are very small, the error bars (standard deviations) of each point in the plots in Figure 4.10 are not obvious. Each curve in Figure 4.10a shows the learning time of the model as the number of examples ( $n$ ) takes  $\{100, 200, 500, 600\}$  while  $m, T$ , and  $d$  remain constant. Each curve in Figure 4.10b shows the learning time of the model as the number of features ( $m$ ) takes  $\{1000, 2000, 4000, 5000\}$  while  $n, d$ , and  $T$  remain constant. Each curve in

Figure 4.10c shows the learning time of the model as the number of possible event times ( $T$ ) takes  $\{25, 50, 75, 100\}$  while  $n$ ,  $m$ , and  $d$  remain constant. Each curve in Figure 4.10d shows the learning time of the model as the time concept dimensionality ( $d$ ) takes  $\{100, 150, 200, 250\}$  while  $n$ ,  $m$ , and  $T$  remain constant. These figures demonstrate that the learning time of the proposed model is approximately linear with respect to  $n$ ,  $m$ ,  $T$ , and  $d$ . Therefore, the time complexity of the proposed algorithm is approximately  $\mathcal{O}(nmTd)$ . That means, in each sub-figure, the slope of the top (red) curve is approximately 50 times as the slope of the bottom (blue) curve, and thus the bottom (blue) curve looks very “flat”.

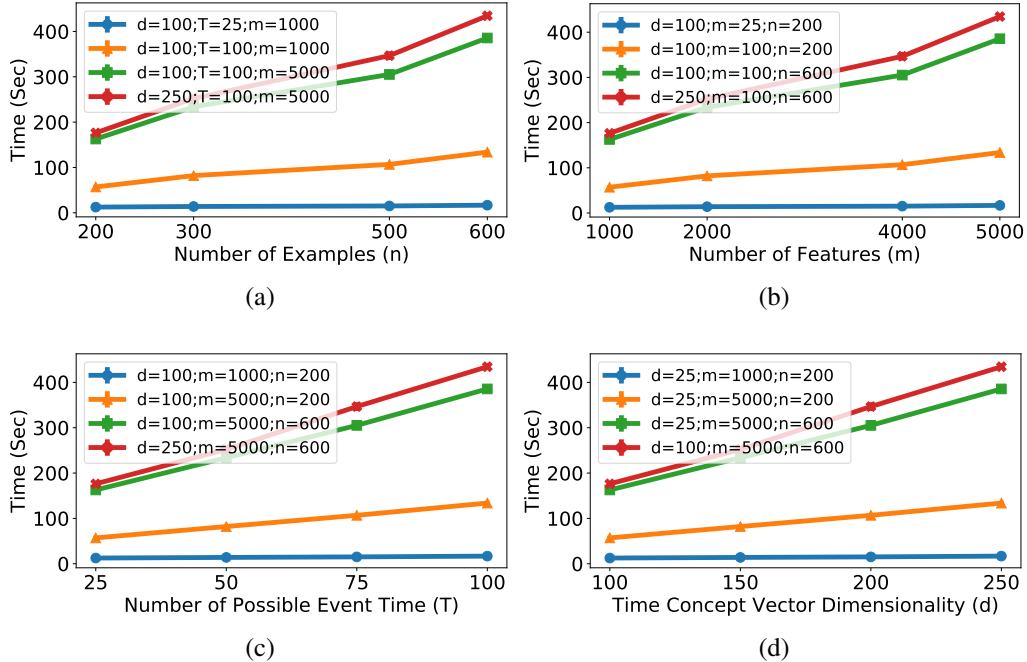


FIGURE 4.10: Learning time of the proposed model with different  $n$ ,  $m$ ,  $T$ , and  $d$  settings. a) learning time changes as  $n$  changes while other variables are constant; b) learning time changes as  $m$  changes while other variables are constant; c) learning time changes as  $T$  changes while other variables are constant; d) learning time changes as  $d$  changes while other variables are constant.

## 4.6 Conclusion

In this study, we propose a new model for the time-to-event estimation problem. In the proposed model, instead of using the actual time points, time concept vectors are used as the target. A scalable optimization framework is also developed to learn the model parameters and time concept vectors jointly. Empirical results show that the proposed model is effective and efficient. It yields results consistent with clinical observation. Using this methodology may reveal previously unrecognized associations between specific clinical characteristics and survival, generating hypotheses to drive further prospective investigation.

# CHAPTER 5

## LEARNING INPUT AND OUTPUT KERNELS FOR TIME-TO-EVENT ESTIMATION ON HIGH-DIMENSIONAL DATA

### 5.1 Introduction

Time-to-Event estimation is also commonly known as survival analysis, whose primary goal is to accurately infer the occurrence time of a target event. The most unique characteristic of the time-to-event estimation problem is the presence of censored examples in the data. Therefore, many existing studies focus on developing new models to effectively utilize the information in the censored observations and optimize a ranking-based objective. In this paper, we propose a model to tackle the time-to-event estimation problem from an output Representation Learning perspective. Our model relaxes a fundamental constraint that the target/output variable, time, is a univariate number which satisfies a partial order. Instead, the proposed model adopts a kernel-based large-margin learning framework and simultaneously learns an input (feature vector) kernel and an output (event time) kernel to leverage the similarities among features and the similarities among event times. Experiments and analysis have been conducted to compare the proposed model to 15 traditional

and state-of-the-art models across 7 benchmark datasets. The results demonstrated the efficiency and effectiveness of the proposed model.

## 5.2 Background

Time-to-event data analysis has been an active research topic due to its tremendous application values in a variety of disciplines, including but not limited to, biology, health-care, engineering, economics, and sociology Tierney et al. (2007). Based on different application purposes, time-to-event estimation is also called survival analysis Klein and Moeschberger (2006), reliability analysis, duration modeling, and event history analysis. The primary goal of a time-to-event estimation model is to accurately infer the occurrence time of a target event or the risk of the event occurrence, and it is a common assumption that the higher the risk, the shorter the time to the occurrence of the event. Therefore, instead of directly estimating the target event occurring time, many existing models, for instance, the notable Cox Proportional-Hazards Model Cox (1992), estimate the risk of the target event occurrence. The most unique characteristic of the time-to-event estimation problem is the presence of censored examples in the data. A censored example is an example whose event occurrence time is unobserved due to observation window limits or losing track during the observation window. The most common censoring case is right censoring in which the event occurs after the end (right edge) of the observation window. In this paper, we only consider the right censoring case.

The event time of a censored example is not completely missing; therefore, most existing models focus on how to effectively make use of the information from the censored examples. Recently, machine learning methods have been adopted for time-to-event modeling Wang et al. (2017b), for instance, Survival Tree Models Gordon and Olshen (1985); LeBlanc and Crowley (1992); Bou-Hamad et al. (2011), Support Vector Machine for censored data Khan and Zubek (2008); Van Belle et al. (2007, 2011), Random Survival Forest

Ishwaran et al. (2011), and Survival Boosting Trees Hothorn et al. (2005). Highly innovative new approaches have been proposed; for example, active learning Vinzamuri et al. (2014), transfer learning Li et al. (2016b), multi-task survival analysis Li et al. (2016a); Wang et al. (2017a), deep survival analysis Ranganath et al. (2016) and adversarial learning Chapfuwa et al. (2018).

In this paper, we approach the time-to-event estimation problem from an Representation Learning perspective. In a conventional time-to-event problem formulation, the target variable *time* is univariate and satisfies a partial order; i.e., if  $t_1 < t_2 < t_3$ , then  $t_2 - t_1 < t_3 - t_1$ . In the proposed approach, instead of directly using *time*, i.e.  $t_1$ ,  $t_2$ , and  $t_3$ , we use their embedded versions, i.e.  $\psi(t_1)$ ,  $\psi(t_2)$ , and  $\psi(t_3)$ , with  $\psi(\cdot)$  as the embedding function. Therefore, the partial ordering relation may not hold in the embedding space; for example, in the original time space, we have  $t_2 - t_1 < t_3 - t_1$ , but in the embedding space, we may have  $\|\psi(t_2) - \psi(t_1)\|_2 > \|\psi(t_3) - \psi(t_1)\|_2$ . We hypothesize that the embeddings are more suitable for capturing the similarities among the event occurrence times, and by effectively exploiting the relations between feature embeddings and time embeddings, we can obtain a more accurate model for time-to-event estimation.

To effectively exploit relations between the observed features and the event time, the proposed approach aims to learn both the input (feature) embedding function and output (time) embedding function (as illustrated in Figure 5.1) along with the model parameter via a ranking-based large-margin objective. By applying the Mercer Theorem Mercer (1909) and the Representer Theorem Schölkopf et al. (2001), the proposed approach learns the corresponding input kernel and output kernel for the input embedding function and output embedding function, respectively. Subsequent steps are then applied to convert the constrained optimization problem to an unconstrained one, and thus off-the-shelf efficient subgradient-based optimization routines can be applied to solve the the problem.

The contributions of our paper are summarized as follows:

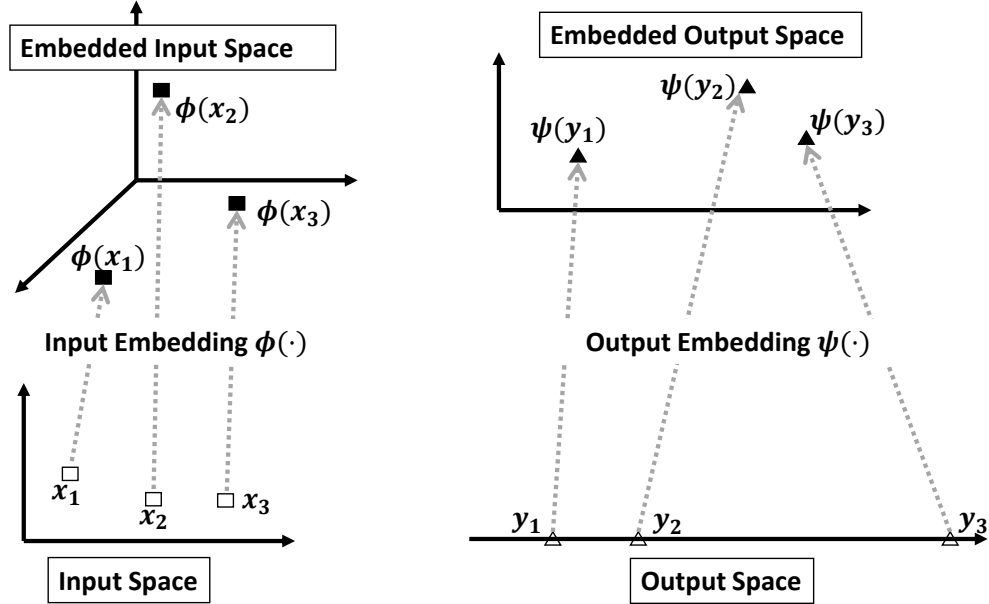


FIGURE 5.1: The proposed model first determines the corresponding time concept of each example and then infers the event occurrence time.

- We propose a new time-to-event estimation model and a learning framework to jointly learn the input kernel, the output kernel and model parameters.
- We develop an efficient optimization framework to learn model parameters and kernels
- We conduct experiments on 7 benchmark datasets and compare the proposed model to 15 traditional and state-of-the-art models and the results demonstrate the effectiveness of the proposed model.

### 5.3 Related Work

The time-to-event model (also called survival analysis) has been one of the most active research topics in statistics, for its vast application values across different fields. For a comprehensive survey on this topic, one can refer to Wang et al. (2017b) and references therein. In this section, we will briefly summarize some of the most widely used models.

The Cox proportional hazard model Cox (1992) is considered as one of the earliest and most influential models in the history of survival analysis research. The parameters in the Cox model are determined by optimizing a partial likelihood function; therefore, models which use the partial likelihood as the optimization objective are considered as Cox-based models. The basic formulation of the Cox model is highly subject to overfitting in high-dimensional data. To address this shortcoming, variants of the Cox model based on different regularizations are proposed; for example, LASSO-Cox Tibshirani (1997), is a Cox-based model with an  $L_1$ -norm regularization, and EN-Cox Simon et al. (2011) is a Cox-based model with the elastic net regularization.

Parametric models are another widely used class of survival analysis models. In parametric models, the event occurrence time is usually assumed to satisfy an underlying distribution of which the density, survivorship, hazard and cumulative hazard functions are easy to derive. The model parameters are then determined by optimizing a likelihood function based on the given data. Some popular choices of the underlying distributions include Logistic, Weibull, Log-Gaussian, and Log-Logistic Lee and Wang (2003).

The time-to-event problem is similar to the traditional regression problem in the sense that the target variable is continuous. However, the traditional regression model cannot be directly applied to the time-to-event problem because of the existence of censored examples. The Tobit model Tobin (1958) is one of the earliest linear regression models which incorporate the censored examples in the optimization objective. The Buckley-James (BJ) regression model Buckley and James (1979) handles the censored data by using an auxiliary Kaplan-Meier estimator Kaplan and Meier (1958). The variant, BJ-EN, is proposed in Wang et al. (2008) for high-dimensional data.

Recently, machine learning techniques are adopted for time-to-event estimation problems Wang et al. (2017b). A survival tree model was proposed in Gordon and Olshen (1985), in which the Wasserstein metric was used to estimate the homogeneity and as the choice of splitting criterion. Support vector machines (SVM) were also adopted for survival

analysis. In Khan and Zubek (2008), a SVM-based model was proposed for censored data by using an updated asymmetric loss function. In Van Belle et al. (2007), an SVM-based model was proposed using a health index as a proxy for the censored time. In Van Belle et al. (2011), an SVM-based model was proposed to incorporate ranking and regression to solve the time-to-event estimation problem. Ensemble models were also adopted to solve the time-to-event estimation problem. In Ishwaran et al. (2011), a random survival forests was proposed to use survival trees Gordon and Olshen (1985) as weak learners. In Hothorn et al. (2005), a boosting algorithm was proposed to incorporate censored data. An active regularized cox regression model was proposed Vinzamuri et al. (2014) to use an active learning algorithm to incorporate the Cox model by using a discriminative gradient sampling strategy. A transfer learning survival analysis model was proposed Li et al. (2016b) to improve the Cox PH model by transferring knowledge from the source domain to the target domain in the context of survival analysis. Multi-task learning model for survival analysis (MTLSA) Li et al. (2016a) model introduced an indicator matrix which allows it to take the advantage of multi-task learning and be able to simultaneously learn from both uncensored and censored examples. In Ranganath et al. (2016), a hierarchical generative approach to survival analysis in the context of the Electronic Health Records was proposed. In Chapfuwa et al. (2018), a time-to-event model as proposed to focus on the estimation of time-to-event distributions by using adversarial generative approach.

## 5.4 Method

### 5.4.1 Notations

In this paper, scalar variables are denoted by letters (e.g.,  $t$  and  $N$ ), vector variables are represented by boldface letters (e.g.,  $\mathbf{x}$ ), matrix variables are represented by boldface uppercase letters (e.g.,  $\Phi$ ). The  $j$ -th column vector and the  $i$ -th row vector of  $\Phi$  are denoted by  $\Phi_{:,j}$  and  $\Phi_{i,:}$ , respectively. We list the main symbols used in our subsequent derivations

Notation	Definition
$N$	The number of examples in a dataset.
$d$	The dimensionality of the feature vectors.
$\mathbf{x}_n \in \mathbb{R}^d$	The feature vector of the $n$ -th instance, for $n = (1, \dots, N)$ .
$y_n \in \mathbb{R}$	The last observation time of the $n$ -th example, for $n = 1, \dots, N$ .
$s_n \in \{0, 1\}$	The status of the $n$ -th example in its last observation (0: no event; 1: event), for $n = (1, \dots, N)$ .
$\phi(\cdot)$	The input embedding function.
$\psi(\cdot)$	The output embedding function.
$\mathbf{w}$	The model parameter for the ranking function.
$\Phi \in \mathbb{R}^{N \times N}$	The input kernel; i.e. $\Phi_{i,j} = \Phi(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$ , for $i = 1, \dots, N$ and $j = 1, \dots, N$ .
$\Psi \in \mathbb{R}^{N \times N}$	The output kernel; i.e. $\Psi_{i,j} = \Psi(y_i, y_j) = \psi(y_i)^\top \psi(y_j)$ , for $i = 1, \dots, N$ and $j = 1, \dots, N$ .
$\mathcal{P}$	The set of the comparable index pairs in the optimization objective function.

Table 5.1: Notations and definitions.

in Table 5.1.

#### 5.4.2 Problem formulation

In the time-to-event model learning, each example in the dataset is represented by a triplet,  $(\mathbf{x}_n, y_n, s_n)$  with  $n \in \{1, 2, \dots, N\}$  being the index of the example and  $N$  being the number of examples. Within each triplet,  $\mathbf{x}_n \in \mathbb{R}^d$  denotes a  $d$ -dimensional feature vector,  $y_n \in \mathbb{R}$  denotes the last observation time, and  $s_n \in \{0, 1\}$  denotes the status of the example at its last observation time. If  $s_n = 1$ , the target event occurs at time  $y_n$  and subsequent observations are not necessary. If  $s_n = 0$ , the target event has not occurred up to time  $y_i$ , and subsequent observations are not available; thus, we call this example censored, and the event may occur at anytime  $t > y_i$ .

The objective of the proposed approach is to learn a ranking function with parameter  $f(\mathbf{x}; \mathbf{w})$  from which the output ranking scores are consistent with the event time in the

dataset; namely,  $y_i > y_j \implies f(\mathbf{x}_i; \mathbf{w}) > f(\mathbf{x}_j; \mathbf{w})$ .

### *A Ranking-based Large-Margin Objective*

Similar to RankSVM Joachims (2002) and StructSVM Joachims et al. (2009), we can setup the following 1-soft large-margin objective:

$$\begin{aligned} \min_{\substack{\mathbf{w}, \psi(\cdot), \phi(\cdot) \\ \{\xi_{(i,j)}\}_{(i,j) \in \mathcal{P}}}} & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{(i,j)} \xi_{(i,j)} \\ \text{subject to} & \quad \mathbf{w}^\top \phi(\mathbf{x}_i) - \mathbf{w}^\top \phi(\mathbf{x}_j) \geq \|\psi(y_i) - \psi(y_j)\|_2^2 - \xi_{(i,j)}, \\ & \quad \xi_{(i,j)} \geq 0, \\ & \quad \forall (i, j) \in \mathcal{P}, \end{aligned} \tag{5.1}$$

in which the ranking function is defined as  $f(\mathbf{x}, \mathbf{w}) \triangleq \mathbf{w}^\top \phi(\mathbf{x})$ , a linear function with respect to the input embedding. The constraints encapsulate the margin requirements: the ranking score difference between two examples has to be at least the Euclidean distance square of the corresponding output embeddings. This optimization formulation aims to simultaneously optimize the ranking function parameter,  $\mathbf{w}$ , the input embedding function,  $\phi(\cdot)$ , the output embedding function,  $\psi(\cdot)$ , and the slack variables,  $\xi_{(i,j)}$ ,  $\forall (i, j) \in \mathcal{P}$ ; where  $(i, j)$  is a comparable pair in the set:

$$\mathcal{P} = \{(i, j) \mid (y_i > y_j \wedge s_j = 1)\}.$$

The trade-off hyper-parameter  $C$  is problem-specific.

### *5.4.3 Optimization*

#### *The Failure of the Dual Formulation*

In many margin-based learning problems, for instance, Joachims (2002); Schölkopf et al. (2001), the optimization problem's dual form may be easier to solve compared to the primal problem. In our case the dual form of the optimization problem (5.1) can be written

as the following:

$$\begin{aligned} & \max_{\boldsymbol{\lambda}, \boldsymbol{\nu}} \quad g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \\ & \text{subject to} \quad \boldsymbol{\lambda} \geq 0, \boldsymbol{\nu} \geq 0, \end{aligned} \tag{5.2}$$

where

$$\begin{aligned} \boldsymbol{\lambda} &= (\cdots \lambda_{(i,j)} \cdots), \forall (i,j) \in \mathcal{P} \\ \boldsymbol{\nu} &= (\cdots \nu_{(i,j)} \cdots), \forall (i,j) \in \mathcal{P} \end{aligned}$$

and

$$\begin{aligned} g(\boldsymbol{\lambda}, \boldsymbol{\nu}) &= \inf_{\substack{\boldsymbol{w}, \psi(\cdot), \phi(\cdot) \\ \{\xi_{(i,j)}\}_{(i,j) \in \mathcal{P}}}} \frac{1}{2} \|\boldsymbol{w}\|_2^2 + C \sum_{(i,j)} \xi_{(i,j)} + \sum_{(i,j)} \nu_{(i,j)} \xi_{(i,j)} \\ &+ \sum_{(i,j)} \lambda_{(i,j)} (\|\psi(y_i) - \psi(y_j)\|_2^2 - \xi_{(i,j)} - \boldsymbol{w}^\top (\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j))), \end{aligned} \tag{5.3}$$

where  $\lambda_{(i,j)}$  and  $\nu_{(i,j)}$  with  $\forall (i,j) \in \mathcal{P}$ , are the Lagrangian multipliers for the inequality constraints in (5.1). In a glance, the dual formulation 5.2, is easier to solve because there are only box constraints involved. Unfortunately, the optimizing objective 5.3 does not have an analytical expression with respect to the Lagrangian multipliers because the forms of the embedding functions,  $\phi(\cdot)$  and  $\psi(\cdot)$  are unknown. Therefore, solving the (5.2) is also very difficult.

### *Our approach*

We first rewrite (5.1) using a hinge loss:

$$\min_{\boldsymbol{w}, \psi(\cdot), \phi(\cdot)} \frac{1}{2} \|\boldsymbol{w}\|_2^2 + C \sum_{(i,j) \in \mathcal{P}} \max(0, \|\psi(y_i) - \psi(y_j)\|_2^2 - \boldsymbol{w}^\top (\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j))) \tag{5.4}$$

By using the the Representer Theorem Schölkopf et al. (2001), we can express the ranking function parameter as:

$$\boldsymbol{w} = \sum_{n=1}^N \beta_n \phi(\boldsymbol{x}_n). \tag{5.5}$$

By using the Mercer Theorem Mercer (1909), and by letting  $\Phi \succeq 0$  be the kernel matrix corresponds to the Hilbert space defined by the input embedding function, we have:

$$\Phi_{ij} = \Phi_{ji} = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j). \quad (5.6)$$

Similarly, by letting  $\Psi \succeq 0$  as the kernel matrix corresponds to the Hilbert space defined by the output embedding function, we have:

$$\Psi_{ij} = \Psi_{ji} = \psi(y_i)^\top \psi(y_j). \quad (5.7)$$

Combining (5.5), (5.6), and (5.7), we can rewrite the 2-norm of the ranking function parameter as:

$$\begin{aligned} \|\mathbf{w}\|_2^2 = \mathbf{w}^\top \mathbf{w} &= \left( \sum_{h=1}^N \beta_h \phi(\mathbf{x}_h) \right)^\top \left( \sum_{m=1}^N \beta_m \phi(\mathbf{x}_m) \right) \\ &= \sum_{h=1}^N \sum_{m=1}^N \beta_h \beta_m \phi(\mathbf{x}_h)^\top \phi(\mathbf{x}_m) \\ &= \boldsymbol{\beta}^\top \Phi \boldsymbol{\beta}, \end{aligned} \quad (5.8)$$

the Euclidean distance of the output embeddings as:

$$\begin{aligned} \|\psi(y_i) - \psi(y_j)\|_2^2 &= (\psi(y_i) - \psi(y_j))^\top (\psi(y_i) - \psi(y_j)) \\ &= \psi(y_i)^\top \psi(y_i) + \psi(y_j)^\top \psi(y_j) - \psi(y_i)^\top \psi(y_j) - \psi(y_j)^\top \psi(y_i) \\ &= \Psi_{ii} + \Psi_{jj} - \Psi_{ij} - \Psi_{ji} \\ &= \Psi_{ii} + \Psi_{jj} - 2\Psi_{ij}, \end{aligned} \quad (5.9)$$

the ranking score difference as:

$$\begin{aligned} \mathbf{w}^\top (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) &= \sum_{n=1}^N \beta_n \phi(\mathbf{x}_n)^\top (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) \\ &= \sum_{n=1}^N \beta_n \Phi_{ni} - \sum_{n=1}^N \beta_n \Phi_{nj} \\ &= \boldsymbol{\beta}^\top (\Phi_{\cdot i} - \Phi_{\cdot j}). \end{aligned} \quad (5.10)$$

By plugging (5.8), (5.9), and (5.10) in (5.4), we can transform (5.4) to the following optimization problem:

$$\begin{aligned} \min_{\beta, \Psi, \Phi} \quad & \frac{1}{2} \beta^\top \Phi \beta + C \sum_{(i,j) \in \mathcal{P}} \max(0, \Psi_{ii} + \Psi_{jj} - 2\Psi_{ij} - \beta^\top (\Phi_{:i} - \Phi_{:j})) \\ \text{subject to} \quad & \Psi \succcurlyeq 0, \\ & \Phi \succcurlyeq 0. \end{aligned} \tag{5.11}$$

The symbol,  $\succcurlyeq$ , represents a Linear Matrix Inequality (LMI), and the LMI's constrain that the kernel matrices are positive semi-definite (PSD). The optimization problems (5.1) and (5.11) are equivalent. Instead of explicitly solving for the ranking function parameter,  $w$ , (5.11) aims to optimize the  $\beta$  from (5.5). Since directly solving for the input embedding function,  $\phi(\cdot)$ , and the output embedding function,  $\psi(\cdot)$ , is infeasible, (5.11) aims to optimize their corresponding kernel matrices,  $\Phi$  and  $\Psi$ , respectively.

However, the optimization problem (5.11) is not scalable because both  $\Phi \in \mathbb{R}^{N \times N}$  and  $\Psi \in \mathbb{R}^{N \times N}$ , in which the number of variables to optimize is in  $\mathcal{O}(N^2)$ , and thus it is quadratic with the number of examples in the dataset; in addition, the PSD constraints on the kernel matrices make the optimization be challenging. To mitigate the drawbacks in (5.11), we adapt the formulation of Multiple Kernel Learning (MKL) Lanckriet et al. (2004); Rakotomamonjy et al. (2008), in which, the optimized kernel is a convex combination of a set of predetermined kernels, i.e., we can represent

$$\Psi = \sum_{k=1}^K a_k \Psi_k, \text{ with } \sum_{k=1}^K a_k = 1 \text{ and } a_k \geq 0, \forall k, \tag{5.12}$$

$$\Phi = \sum_{l=1}^L b_l \Phi_l, \text{ with } \sum_{l=1}^L b_l = 1 \text{ and } b_l \geq 0, \forall l, \tag{5.13}$$

in which  $\{\Psi_k\}_{k=1}^K$  is a set of predetermined output kernels, and  $\{\Phi_l\}_{l=1}^L$  is a set of predetermined input kernels. Instead of solving  $\Psi$  and  $\Phi$  explicitly, we can solve for their

optimal convex combinations, and thus, we reformulate (5.11) as:

$$\begin{aligned}
\min_{\beta, \mathbf{a}, \mathbf{b}} \quad & \frac{1}{2} \beta^\top \hat{\Psi}(\mathbf{a}) \beta \\
& + C \sum_{(i,j) \in \mathcal{P}} \max(0, \hat{\Phi}(\mathbf{b})_{ii} + \hat{\Phi}(\mathbf{b})_{jj} - 2\hat{\Phi}(\mathbf{b})_{ij} - \beta^\top (\hat{\Psi}(\mathbf{a})_{:i} - \hat{\Psi}(\mathbf{a})_{:j})) \\
\text{subject to} \quad & \mathbf{a}^\top \mathbf{e} = 1, \mathbf{b}^\top \mathbf{e} = 1, \\
& \mathbf{a} \geq 0, \mathbf{b} \geq 0,
\end{aligned} \tag{5.14}$$

where  $\mathbf{a} = (a_1, \dots, a_K)$ ,  $\mathbf{b} = (b_1, \dots, b_L)$ ,  $\mathbf{e}$  is a vector with all 1's, and let

$$\begin{aligned}
\hat{\Psi} : \mathbb{R}^K &\rightarrow \mathbb{R}^{N \times N} \text{ be a function } \hat{\Psi}(\mathbf{a}) = \sum_{k=1}^K a_k \Psi_k, \\
\hat{\Phi} : \mathbb{R}^L &\rightarrow \mathbb{R}^{N \times N} \text{ be a function } \hat{\Phi}(\mathbf{b}) = \sum_{l=1}^L b_l \Phi_l,
\end{aligned}$$

We can further simplify (5.14) by eliminating the constraints with the following changes of variables:

$$a_k = \sigma(\boldsymbol{\gamma})_k = \frac{e^{\gamma_k}}{\sum_{k=1}^K e^{\gamma_k}}, \text{ for } k = (1, \dots, K), \boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_K) \in \mathbb{R}^K; \tag{5.15}$$

$$b_l = \sigma(\boldsymbol{\mu})_l = \frac{e^{\mu_l}}{\sum_{l=1}^L e^{\mu_l}}, \text{ for } l = (1, \dots, L), \boldsymbol{\mu} = (\mu_1, \dots, \mu_L) \in \mathbb{R}^L. \tag{5.16}$$

Thus, we have

$$\hat{\Psi}(\mathbf{a}) = \hat{\Psi}(\sigma(\boldsymbol{\gamma})), \tag{5.17}$$

and

$$\hat{\Phi}(\mathbf{b}) = \hat{\Phi}(\sigma(\boldsymbol{\mu})). \tag{5.18}$$

By combining (5.17), (5.18), and (5.14), we arrive at the final form of the optimization

problem

$$\begin{aligned}
\min_{\beta, \gamma, \mu} \quad & \frac{1}{2} \beta^\top \hat{\Psi}(\sigma(\mu)) \beta \\
& + C \sum_{(i,j) \in \mathcal{P}} \max(0, \hat{\Psi}(\sigma(\gamma))_{ii} + \hat{\Psi}(\sigma(\gamma))_{jj} - 2\hat{\Psi}(\sigma(\gamma))_{ij} \\
& \quad - \beta^\top (\hat{\Phi}(\sigma(\mu))_{:i} - \hat{\Phi}(\sigma(\mu))_{:j}))
\end{aligned} \tag{5.19}$$

which is unconstrained (i.e.  $\beta \in \mathbb{R}^N$ ,  $\gamma \in \mathbb{R}^K$ , and  $\mu \in \mathbb{R}^L$ ). To solve this unconstrained optimization problem, we use a sub-gradient descent approach.

## 5.5 Experiments

### 5.5.1 Setup

The proposed model is implemented in Python 3.6.6 based on the *Numpy* package. We also use the *Autograd* package for computing the derivatives. In the proposed model, there is only one hyper-parameter,  $C$ , as shown in (5.19). The value of the hyper-parameter was determined by a grid search in values  $\{0.001, 0.01, 0.1, 1.0, 10.0, 100.0\}$  using cross-validation within the training set.

The predetermined input kernels are shown in Table. 5.2. There are four types of kernels used: *Constant Kernel*, *Linear Kernel*, *Polynomial Kernel*, and *Radial Basis Function (RBF) Kernel*. In the Polynomial Kernel, there are three parameters, and each of them has 3 values, so there are  $3 \times 3 \times 3 = 27$  combinations. In the RBF Kernel, the kernel parameter has 5 values, so 5 RBF kernels are included. Therefore, in total, there are  $1 + 1 + 27 + 5 = 34$  predetermined input kernels used.

Using the same types of kernels and the same parameter settings as the predetermined input kernels, there are also 34 predetermined output kernels used.

Name	Definition	Parameter settings
Constant Kernel	$\Phi(\mathbf{x}_i, \mathbf{x}_j) = 1$	n/a
Linear Kernel	$\Phi(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$	n/a
Polynomial Kernel	$\Phi(\mathbf{x}_i, \mathbf{x}_j) = (\eta \mathbf{x}_i^\top \mathbf{x}_j + c_0)^p$	$\eta = \{0.1, 0.5, 1.0\}$ , $c_0 = \{-1, 0, 1\}$ , $d = \{3, 4, 7\}$
RBF Kernel	$\Phi(\mathbf{x}_i, \mathbf{x}_j) = \exp(\eta \ \mathbf{x}_i - \mathbf{x}_j\ ^2)$	$\eta = \{\frac{0.1}{d}, \frac{0.5}{d}, \frac{1}{d}, \frac{5}{d}, \frac{10}{d}\}$

Table 5.2: The predetermined input kernels used in our experiments

### 5.5.2 Comparisons to State-of-the-art Models on Benchmark Datasets

In this experiment, we compare the performance, in terms of the concordance index, of the proposed model with the state-of-the-art baseline models on several benchmark datasets for the time-to-event estimation tasks.

#### Datasets

The 7 benchmark datasets used in this experiment are publicly available for download. Details information of the datasets and the task descriptions are provided as the follows:

- *Van de Vijver’s Microarray Breast Cancer data (VDV)* Van’t Veer et al. (2002) contains 4,707 gene expression values on 78 (44 censored) breast cancer patients for predicting occurrence of death in terms of year, up to 13 years.
- *Gene-expression profiles of lung adenocarcinoma (Lung)* Beer et al. (2002) contains 7,129 gene expression values on 86 (62 censored) early-stage lung adenocarcinoma patients for predicting occurrence of death in terms of month, up to 110 months.
- *Mantle Cell Lymphoma (MCL)* Rosenwald et al. (2003) contains 8,810 gene expression values on 92 (28 censored) MCL patients for predicting occurrence of death in terms of year, up to 14 years.

- *Norway/Stanford breast cancer data (NSBCD)* Sørli et al. (2003) contains 549 gene expression values on 115 (77 censored) breast cancer women for predicting occurrence of death in terms of month, up to 188 months.
- *Adult myeloid leukemia (AML)* Bullinger et al. (2004) contains 6,283 gene expression values on 116 (77 censored) AML patients for predicting occurrence of death in terms of month, up to 54 months.
- *Diffuse Large B-Cell Lymphoma (DLBCL)* Lossos (2008) contains 7,399 gene expression values on 240 (102 censored) DLBCL patients for predicting occurrence time of death in terms of year, up to 21 years.
- *Dutch Breast Cancer Data (DBCD)* van Houwelingen et al. (2006) contains 4,919 gene expression values on 295 (216 censored) breast cancer women for predicting occurrence time of death in terms of year, up to 18 years.

The 7 benchmark datasets are summarized in Table 5.3. To have a fair comparisons to the baseline models, we adopt the evaluation settings in Li et al. (2016a), in which 5-fold cross-validation is used when the number of examples is greater than 150 and 3-fold cross-validation otherwise.

Dataset	# example	# censored	# feature	# time
VDV	78	44	4705	13
Lung	86	62	7129	110
MCL	92	28	8810	14
NSBCD	115	77	549	188
AML	116	49	6283	54
DLBCL	240	102	7399	21
DBCD	295	216	4919	18

Table 5.3: Summaries of the 7 benchmark datasets (ordered by # example).

	VDV	Lung	MCL	NSBCD	AML	DLBCL	DBCDC
CoxPH	0.597 (7) ± 0.011	0.516 (13) ± 0.133	0.577 (11) ± 0.059	0.441 (12) ± 0.059	0.552 (8) ± 0.068	0.455 (13) ± 0.072	0.554 (11) ± 0.123
CoxLasso	0.648 (4) ± 0.028	0.670 (3) ± 0.091	0.682 (8) ± 0.07	0.591 (10) ± 0.109	0.600 (4) ± 0.031	0.634 (4) ± 0.042	0.688 (8) ± 0.043
CoxEN	0.642 (5) ± 0.068	0.665 (4) ± 0.07	0.673 (9) ± 0.073	0.605 (9) ± 0.1	0.572 (6) ± 0.06	0.649 (3) ± 0.039	0.721 (4) ± 0.031
Logistic	0.528 (8) ± 0.14	0.571 (11) ± 0.094	0.483 (12) ± 0.068	0.379 (13) ± 0.02	0.454 (15) ± 0.077	0.484 (12) ± 0.05	0.491 (13) ± 0.087
Weibull	0.316 (16) ± 0.132	0.429 (15) ± 0.01	0.474 (14) ± 0.075	0.305 (15) ± 0.153	0.529 (12) ± 0.055	0.251 (16) ± 0.063	0.456 (16) ± 0.105
Log-Gaussian	0.521 (10) ± 0.165	0.412 (16) ± 0.075	0.256 (16) ± 0.072	0.444 (11) ± 0.054	0.405 (16) ± 0.065	0.317 (15) ± 0.091	0.488 (14) ± 0.055
Log-Logistic	0.527 (9) ± 0.107	0.592 (9) ± 0.066	0.480 (13) ± 0.072	0.238 (16) ± 0.05	0.468 (14) ± 0.08	0.425 (14) ± 0.124	0.526 (12) ± 0.023
Tobit	0.519 (11) ± 0.158	0.469 (14) ± 0.136	0.459 (15) ± 0.032	0.373 (14) ± 0.021	0.473 (13) ± 0.076	0.497 (11) ± 0.053	0.487 (15) ± 0.076
BJ-EN	0.608 (6) ± 0.065	0.665 (4) ± 0.132	0.723 (3) ± 0.11	0.622 (8) ± 0.092	0.650 (2) ± 0.059	0.629 (5) ± 0.073	0.709 (6) ± 0.039
Boost-CI	0.665 (2) ± 0.059	0.571 (11) ± 0.093	0.705 (5) ± 0.096	0.626 (7) ± 0.083	0.582 (5) ± 0.05	0.608 (7) ± 0.03	0.710 (5) ± 0.043
MTLSA	0.701 (1) ± 0.033	0.633 (8) ± 0.075	0.727 (2) ± 0.096	0.682 (3) ± 0.045	0.715 (1) ± 0.049	0.653 (2) ± 0.071	0.758 (1) ± 0.03
SurvGB	0.509 (12) ± 0.085	0.586 (10) ± 0.07	0.650 (10) ± 0.048	0.634 (5) ± 0.089	0.539 (11) ± 0.051	0.564 (10) ± 0.056	0.665 (10) ± 0.043
SurvSVM-Linear	0.481 (14) ± 0.055	0.706 (2) ± 0.075	0.719 (4) ± 0.059	0.647 (4) ± 0.052	0.552 (8) ± 0.044	0.599 (8) ± 0.039	0.709 (6) ± 0.027
SurvSVM-RBF	0.491 (13) ± 0.05	0.656 (6) ± 0.053	0.704 (6) ± 0.063	0.725 (2) ± 0.04	0.566 (7) ± 0.029	0.611 (6) ± 0.025	0.753 (3) ± 0.033
BJ-Neural	0.467 (15) ± 0.073	0.643 (7) ± 0.066	0.687 (7) ± 0.075	0.628 (6) ± 0.042	0.542 (10) ± 0.047	0.579 (9) ± 0.037	0.679 (9) ± 0.038
Proposed	0.650 (3) ± 0.090	0.710 (1) ± 0.051	0.757 (1) ± 0.024	0.745 (1) ± 0.040	0.606 (3) ± 0.030	0.656 (1) ± 0.016	0.755 (2) ± 0.040

Table 5.4: Concordance Indices (CI's) and corresponding standard deviations of all the models; The ranking of a model among all compared models in each dataset is included.

### Baseline models

Based on the popularity and accessibility, we compared our proposed model to 16 baseline models.

*Cox based models* The Cox proportional hazards model Cox (1992) is the most commonly used survival analysis model. Besides the basic formulation, the  $L_1$ -norm regularized variant, *LASSO-Cox* Tibshirani (1997), and the elastic net regularized variant, *EN-Cox* Simon et al. (2011), are also widely used.

*Censored regression models* Standard likelihood function estimation incorporates censored examples Lee and Wang (2003). Based on the assumptions of the underlying distributions, it has four variants: Weibull, Logistic, Log-Logistic, and Log-Gaussian.

*Linear models* Tobit model Tobin (1958) is an extension of linear regression that incorporates censored examples with parameters estimated by the maximum likelihood method rather than using least squares error. Buckley-James regression Wang et al. (2008) (BJ-EN), is a linear model which incorporates censored examples by estimating the censored value using the Kaplan-Meier estimator Kaplan and Meier (1958) with elastic net regularization.

*Pairwise ranking based models* Boosting concordance index Mayr and Schmid (2014) (Boost-CI) is a gradient boosting algorithm to optimize the smoothed version of the concordance index.

*Multi-task learning models* *MTLSA* is multi-task formulation tailored for handling censored examples Li et al. (2016a) by introducing an indication table and a non-negative non-increasing list structure constraint.

*Others* *SurvGB* Hothorn et al. (2005) is a gradient boosting model for survival analysis. *SurvSVM-Linear* Pölsterl et al. (2015) is a rank and regression based SVM adoption for survival analysis using a linear kernel. *SurvSVM-RBF* Pölsterl et al. (2016) is an efficient SVM-based survival analysis model using the radial basis function kernel. *BJ-Neural* is a 2-layer neural network with Rectified Linear Unit Activation functions to optimized an objective of Buckley-James regression Wang et al. (2008).

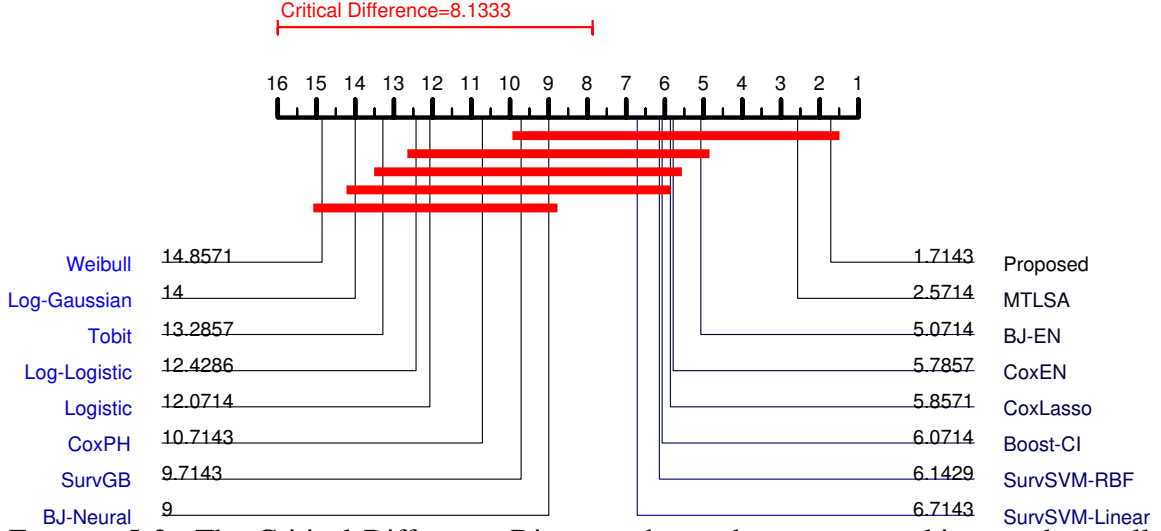


FIGURE 5.2: The Critical Difference Diagram shows the average rankings and overall rankings of all the compared models in the measure of Concordance Index.

### Performance metric

Due to the presence of censored examples, concordance index (CI or c-index) Heagerty and Zheng (2005) is used to evaluate the performance of time-to-event estimation models. The CI of a model,  $g$ , is defined as Steck et al. (2008):

$$\text{IC}(f) = \frac{1}{num} \sum_{s_j=1} \sum_{y_i > y_j} \mathbb{1}(f(\mathbf{x}_i; \mathbf{w}) > f(\mathbf{x}_j; \mathbf{w})),$$

where  $\mathbb{1}(\cdot)$  is an indication function, and  $num$  is a normalization factor such that  $\text{CI}(g) \leq 1$ , and  $\text{CI}(g) = 1$  is the best possible performance. The CI's and corresponding standard deviations of all baseline models and the proposed model on the benchmark datasets are shown in Table 5.4. The results of *SurvSVM*'s and *SurvGB* were obtained by using the the functions in the *Scikit-Survival* package. The result of *BJ-Neural* was obtained by using the *Scikit-Learn* package. Results of the other baseline models are adopted from Li et al. (2016a).

Figure 5.2 shows the Critical Difference Diagram Demšar (2006) of Experiment #1. The proposed method achieves the best overall ranking. Models with regularizations, i.e., *proposed*, *MTLSA*, *BJ-EN*, *CoxEN*, and *CoxLasso* have the best overall rankings; this may

suggest that in datasets with relatively small number of examples, regularization is very important to prevent overfitting.

## 5.6 Conclusion

In the paper, we propose a new model for the time-to-event estimation problem and an efficient learning framework. The proposed model uses the time-to-event data to learn an input kernel, an output kernel and a ranking function parameter jointly. Experiments have been conducted to compare the proposed model to 15 baseline traditional and state-of-the-art models across 7 benchmark datasets. Model performance shown the effectiveness of the proposed model.

# CHAPTER 6

## CONCLUSION

To conclude, in this dissertation, four different Representation Learning approaches are described in details. These approaches provide Representation Learning treatments in different steps in general data analytics workflow, including, data preprocessing (Chapter 2), input representation learning (Chapter 3 and 5), and output representation learning (Chapter 4). These four methods also make use of different Machine Learning paradigms, including, unsupervised learning (Chapter 2 and 3) and supervised learning (Chapter 4 and 5). More importantly, these approaches are heavily focused on applicability in real-world problems; therefore, they share the characteristics of scalable, interpretable, and easy-to-use. Each chapter is written in a structured way that each of them includes:

- a detailed background section to motivate the target problem and lead to the main idea of the approach;
- a comprehensive related work section to list major state-of-the-art related approaches;
- a thorough method section to provide the technical details of the method and algorithm;

- a rigorous experiment section to analyze the advantages and disadvantage of the proposed approach.

Therefore, while the chapters are related via the typical data analytics workflow, they can be seen as standalone treatments for specific problems encountered in biomedical informatics research.

In Chapter 2, We propose the Generalized Logistic (GL) algorithm that scales data uniformly to an appropriate interval by learning a generalized logistic function to fit the empirical cumulative distribution function of the data. This approach aims to learn a representation based on data scaling. The GL algorithm is simple yet effective; it is intrinsically robust to outliers, so it is particularly suitable for diagnostic/classification models in clinical/medical applications where the number of samples is usually small; it scales the data in a nonlinear fashion, which leads to a potential improvement in accuracy. The experimental performance in terms of area under the receiver operating characteristic curve (AUROC) and percentage of correct classification showed that models learned using data scaled by the GL algorithm outperform the ones using data scaled by the Min-max and the Z-score algorithm, which are the most commonly used data scaling algorithms.

In Chapter 3, we propose an approach of learning the representations for biomedical multivariate time-series data with real-work imperfections. Time series in healthcare practices and biomedical research are typically multivariate, i.e., multiple biomarkers are observed simultaneously at a time. However, they tend to be short, noisy, unaligned, irregularly sampled, partially observed, and with only limited samples. These imperfections pose a challenge for mining information from data. In this work, we propose to use dynamic-based representations to present such imperfect multivariate time series. Specifically, we propose an approach to learn a corresponding Linear Dynamical System (LDS) for a multivariate time series example and use the set of system parameters as a representation for that example. Such a representation can capture interactions of different

variables and provide a unified view of multivariate time series with different lengths, different missingness mechanisms, and different starting points. Other techniques are then used to mine useful information and perform learning tasks based on the new representation. For example, we use support vector machine classification models with LDS kernels in time series classification tasks. We have conducted experiments on both synthetic data sets and real-life datasets. The accuracy in the leave-one-out symptomatic/asymptomatic diagnostic tasks showed that our approach outperformed three baseline algorithms. Moreover, in experiments where various levels of imperfections were imposed on the H3N2 dataset, the accuracy of other baseline methods degraded significantly, but the accuracy of our approach remained high.

Chapter 4 and 5 focus on output Representation Learning and its applications in the problem of time-to-event estimation. The primary goal of a time-to-event estimation model is to infer the occurrence time of a target event accurately. Most existing studies focus on developing new models to utilize the information in the censored observations effectively. In chapter 4, an output Representation Learning model is described to tackle the time-to-event estimation problem. The model relaxes a fundamental constraint that the target variable, time, is a univariate number which satisfies a partial order. Instead, the proposed model interprets each event occurrence time as a time concept with a vector representation. We hypothesize that the model is more accurate and interpretable by capturing 1) the relationships between features and time concept vectors and 2) the relationships among time concept vectors. We also propose a scalable framework to learn the model parameters and time concept vectors simultaneously. Besides, similarity information among time concept vectors helped in identifying time regimes, thus leading to a potential knowledge discovery related to the human cancer datasets considered in our experiments. The model described in Chapter 5 complements some of the drawbacks in the model in Chapter 4. The proposed model adopts a kernel-based large-margin learning framework and simultaneously learns an input (feature vector) kernel and an output (event

time) kernel to leverage the similarities among features and the similarities among event times. Both of the models are evaluated and analyzed in experiments on 7 benchmark datasets, and they are compared to 15 traditional and state-of-the-art models. The results demonstrated the efficiency and effectiveness of the proposed models.

# BIBLIOGRAPHY

- Acuna, E. and Rodriguez, C. (2004), “A meta analysis study of outlier detection methods in classification,” *Technical paper, Department of Mathematics, University of Puerto Rico at Mayaguez*.
- Agrawal, R., Faloutsos, C., and Swami, A. (1993), *Efficient similarity search in sequence databases*, Springer, New York.
- Beer, D. G., Kardia, S. L., Huang, C.-C., Giordano, T. J., Levin, A. M., Misek, D. E., Lin, L., Chen, G., Gharib, T. G., Thomas, D. G., et al. (2002), “Gene-expression profiles predict survival of patients with lung adenocarcinoma,” *Nature medicine*, 8, 816.
- Bengio, Y., Courville, A., and Vincent, P. (2013), “Representation learning: A review and new perspectives,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35, 1798–1828.
- Boots, B., Gordon, G. J., and Siddiqi, S. M. (2007), “A Constraint Generation Approach to Learning Stable Linear Dynamical Systems,” in *Advances in Neural Information Processing Systems*, pp. 1329–1336.
- Borg, I. and Groenen, P. J. (2005), *Modern multidimensional scaling: Theory and applications*, Springer Science & Business Media, New York.
- Borgwardt, K. M., Vishwanathan, S., and Kriegel, H.-P. (2006), “Class prediction from time series gene expression profiles using dynamical systems kernels.” in *Pacific symposium on biocomputing*, vol. 11, pp. 547–558.
- Bou-Hamad, I., Larocque, D., Ben-Ameur, H., et al. (2011), “A review of survival trees,” *Statistics Surveys*, 5, 44–71.
- Bowling, S. R., Khasawneh, M. T., Kaewkuekool, S., and Cho, B. R. (2009), “A logistic approximation to the cumulative normal distribution,” *Journal of Industrial Engineering and Management*, 2, 114–127.
- Buckley, J. and James, I. (1979), “Linear regression with censored data,” *Biometrika*, 66, 429–436.

- Bullinger, L., Döhner, K., Bair, E., Fröhling, S., Schlenk, R. F., Tibshirani, R., Döhner, H., and Pollack, J. R. (2004), “Use of gene-expression profiling to identify prognostic subclasses in adult acute myeloid leukemia,” *New England Journal of Medicine*, 350, 1605–1616.
- Candes, E. J., Romberg, J. K., and Tao, T. (2006), “Stable signal recovery from incomplete and inaccurate measurements,” *Communications on pure and applied mathematics*, 59, 1207–1223.
- Cao, X. H. and Obradovic, Z. (2015), “A robust data scaling algorithm for gene expression classification,” in *Bioinformatics and Bioengineering (BIBE), 2015 IEEE 15th International Conference on*, pp. 1–4, IEEE.
- Chan, K.-P. and Fu, A. W.-C. (1999), “Efficient time series matching by wavelets,” in *Data Engineering, 1999. Proceedings., 15th International Conference on*, pp. 126–133, IEEE.
- Chapfuwa, P., Tao, C., Li, C., Page, C., Goldstein, B., Carin, L., and Henao, R. (2018), “Adversarial Time-to-Event Modeling,” *arXiv preprint arXiv:1804.03184*.
- Chen, B. and Zadrozny, P. A. (2001), “Analytic derivatives of the matrix exponential for estimation of linear continuous-time models,” *Journal of Economic Dynamics and Control*, 25, 1867–1879.
- Cox, D. R. (1992), “Regression models and life-tables,” in *Breakthroughs in statistics*, pp. 527–541, Springer.
- De Cock, K. and De Moor, B. (2002), “Subspace angles between ARMA models,” *Systems & Control Letters*, 46, 265–270.
- Demšar, J. (2006), “Statistical comparisons of classifiers over multiple data sets,” *Journal of Machine learning research*, 7, 1–30.
- Dudoit, S., Yang, Y. H., Callow, M. J., and Speed, T. P. (2002), “Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments,” *Statistica sinica*, pp. 111–139.
- Edgar, R., Domrachev, M., and Lash, A. E. (2002), “Gene Expression Omnibus: NCBI gene expression and hybridization array data repository,” *Nucleic acids research*, 30, 207–210.
- Foster, K. R., Koprowski, R., and Skufca, J. D. (2014), “Machine learning, medical diagnosis, and biomedical engineering research-commentary,” *Biomedical engineering online*, 13, 94.
- Gallier, J. (2011), *Geometric methods and applications: for computer science and engineering*, vol. 38, Springer Science & Business Media, New York.

- Ghahramani, Z. and Hinton, G. E. (1996), “Parameter Estimation for Linear Dynamical Systems,” *University of Toronto technical report CRGTR962*, 6, 1–6.
- Ghalwash, M. F. and Obradovic, Z. (2012), “Early classification of multivariate temporal observations by extraction of interpretable shapelets,” *BMC bioinformatics*, 13, 1.
- Gonzalez, R. and Woods, R. (2008), “Digital Image Processing: Pearson Prentice Hall,” *Upper Saddle River, NJ*.
- Gordon, L. and Olshen, R. A. (1985), “Tree-structured survival analysis.” *Cancer treatment reports*, 69, 1065–1069.
- Hamilton, J. D. (1994), *Time series analysis*, vol. 2, Princeton university press, Princeton.
- Han, J., Kamber, M., and Pei, J. (2011), *Data mining: concepts and techniques: concepts and techniques*, Elsevier.
- Harvey, A. and Stock, J. H. (1985), “The estimation of higher-order continuous time autoregressive models,” *Econometric Theory*, 1, 97–117.
- Haykin, S. S. (2009), *Neural networks and learning machines*, Pearson Education Upper Saddle River.
- Heagerty, P. J. and Zheng, Y. (2005), “Survival model predictive accuracy and ROC curves,” *Biometrics*, 61, 92–105.
- Henn, A. D., Wu, S., Qiu, X., Ruda, M., Stover, M., Yang, H., Liu, Z., Welle, S. L., Holden-Wiltse, J., Wu, H., et al. (2013), “High-resolution temporal response patterns to influenza vaccine reveal a distinct human plasma cell gene signature,” *Scientific reports*, 3.
- Henry, K. E., Hager, D. N., Pronovost, P. J., and Saria, S. (2015), “A targeted real-time early warning score (TREWScore) for septic shock,” *Science Translational Medicine*, 7, 299ra122–299ra122.
- Hothorn, T., Bühlmann, P., Dudoit, S., Molinaro, A., and Van Der Laan, M. J. (2005), “Survival ensembles,” *Biostatistics*, 7, 355–373.
- Ishwaran, H., Kogalur, U. B., Chen, X., and Minn, A. J. (2011), “Random survival forests for high-dimensional data,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 4, 115–132.
- Joachims, T. (2002), “Optimizing search engines using clickthrough data,” in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142, ACM.
- Joachims, T., Finley, T., and Yu, C.-N. J. (2009), “Cutting-plane training of structural SVMs,” *Machine Learning*, 77, 27–59.

- Kaplan, E. L. and Meier, P. (1958), “Nonparametric estimation from incomplete observations,” *Journal of the American statistical association*, 53, 457–481.
- Katayama, T. (2006), *Subspace methods for system identification*, Springer Science & Business Media, New York.
- Kelchtermans, P., Bittremieux, W., Grave, K., Degroeve, S., Ramon, J., Laukens, K., Valkenburg, D., Barsnes, H., and Martens, L. (2014), “Machine learning applications in proteomics research: How the past can boost the future,” *Proteomics*, 14, 353–366.
- Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001), “Dimensionality reduction for fast similarity search in large time series databases,” *Knowledge and Information Systems*, 3, 263–286.
- Khan, F. M. and Zubek, V. B. (2008), “Support vector regression for censored data (SVRC): a novel tool for survival analysis,” in *Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on*, pp. 863–868, IEEE.
- Klein, J. P. and Moeschberger, M. L. (2006), *Survival analysis: techniques for censored and truncated data*, Springer Science & Business Media.
- Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., and Fotiadis, D. I. (2015), “Machine learning applications in cancer prognosis and prediction,” *Computational and structural biotechnology journal*, 13, 8–17.
- Lanckriet, G. R., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2004), “Learning the Kernel Matrix with Semidefinite Programming,” *Journal of Machine learning research*, 5, 27–72.
- LeBlanc, M. and Crowley, J. (1992), “Relative risk trees for censored survival data,” *Biometrics*, pp. 411–425.
- Lee, E. T. and Wang, J. (2003), *Statistical methods for survival data analysis*, vol. 476, John Wiley & Sons.
- Li, Y., Wang, J., Ye, J., and Reddy, C. K. (2016a), “A multi-task learning formulation for survival analysis,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1715–1724, ACM.
- Li, Y., Wang, L., Wang, J., Ye, J., and Reddy, C. K. (2016b), “Transfer learning for survival analysis via efficient L2, 1-norm regularized Cox regression,” in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pp. 231–240, IEEE.
- Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003), “A symbolic representation of time series, with implications for streaming algorithms,” in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pp. 2–11, ACM.

- Liu, Z. and Hauskrecht, M. (2015), “A Regularized Linear Dynamical System Framework for Multivariate Time Series Analysis,” in *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, vol. 2015, p. 1798, NIH Public Access.
- Liu, Z. and Hauskrecht, M. (2016), “Learning linear dynamical systems from multivariate time series: A matrix factorization based framework,” in *Proceedings of the 2016 SIAM International Conference on Data Mining*, pp. 810–818, SIAM.
- Lossos, I. S. (2008), “Diffuse large B cell lymphoma: from gene expression profiling to prediction of outcome,” *Biology of Blood and Marrow Transplantation*, 14, 108–111.
- Maaten, L. v. d. and Hinton, G. (2008), “Visualizing data using t-SNE,” *Journal of machine learning research*, 9, 2579–2605.
- Maltoni, M., Caraceni, A., Brunelli, C., Broeckaert, B., Christakis, N., Eychmueller, S., Glare, P., Nabal, M., Vigano, A., Larkin, P., et al. (2005), “Prognostic factors in advanced cancer patients: evidence-based clinical recommendations—a study by the Steering Committee of the European Association for Palliative Care,” *Journal of Clinical Oncology*, 23, 6240–6248.
- Martin, R. J. (2000), “A metric for ARMA processes,” *IEEE Transactions on Signal Processing*, 48, 1164–1170.
- Mayr, A. and Schmid, M. (2014), “Boosting the concordance index for survival data—a unified framework to derive and evaluate biomarker combinations,” *PloS one*, 9, e84483.
- Mercer, J. (1909), “Xvi. functions of positive and negative type, and their connection the theory of integral equations,” *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209, 415–446.
- Murphy, K. P. (2012), *Machine learning: a probabilistic perspective*, MIT press, Cambridge, Boston.
- Omranian, N., Mueller-Roeber, B., and Nikoloski, Z. (2015), “Segmentation of biological multivariate time-series data,” *Scientific reports*, 5.
- Parikh, N. and Boyd, S. (2013), “Proximal algorithms,” *Foundations and Trends in optimization*, 1, 123–231.
- Pölsterl, S., Navab, N., and Katouzian, A. (2015), “Fast training of support vector machines for survival analysis,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 243–259, Springer.
- Pölsterl, S., Navab, N., and Katouzian, A. (2016), “An Efficient Training Algorithm for Kernel Survival Support Vector Machines,” *arXiv preprint arXiv:1611.07054*.

- Rakotomamonjy, A., Bach, F. R., Canu, S., and Grandvalet, Y. (2008), “SimpleMKL,” *Journal of Machine Learning Research*, 9, 2491–2521.
- Ranganath, R., Perotte, A., Elhadad, N., and Blei, D. (2016), “Deep survival analysis,” *arXiv preprint arXiv:1608.02158*.
- Rangayyan, R. M. (2015), *Biomedical signal analysis*, vol. 33, John Wiley & Sons, New Jersey.
- Rosenwald, A., Wright, G., Wiestner, A., Chan, W. C., Connors, J. M., Campo, E., Gascoyne, R. D., Grogan, T. M., Muller-Hermelink, H. K., Smeland, E. B., et al. (2003), “The proliferation gene expression signature is a quantitative integrator of oncogenic events that predicts survival in mantle cell lymphoma,” *Cancer cell*, 3, 185–197.
- Schölkopf, B., Herbrich, R., and Smola, A. J. (2001), “A generalized representer theorem,” in *International conference on computational learning theory*, pp. 416–426, Springer.
- Shumway, R. H. and Stoffer, D. S. (1982), “An approach to time series smoothing and forecasting using the EM algorithm,” *Journal of time series analysis*, 3, 253–264.
- Simon, N., Friedman, J., Hastie, T., and Tibshirani, R. (2011), “Regularization paths for Cox’s proportional hazards model via coordinate descent,” *Journal of statistical software*, 39, 1.
- Sørli, T., Tibshirani, R., Parker, J., Hastie, T., Marron, J. S., Nobel, A., Deng, S., Johnsen, H., Pesich, R., Geisler, S., et al. (2003), “Repeated observation of breast tumor subtypes in independent gene expression data sets,” *Proceedings of the national academy of sciences*, 100, 8418–8423.
- Städler, N., Mukherjee, S., et al. (2013), “Penalized estimation in high-dimensional hidden Markov models with state-specific graphical models,” *The Annals of Applied Statistics*, 7, 2157–2179.
- Statnikov, A., Tsamardinos, I., Dosbayev, Y., and Aliferis, C. F. (2005), “GEMS: a system for automated cancer diagnosis and biomarker discovery from microarray gene expression data,” *International journal of medical informatics*, 74, 491–503.
- Steck, H., Krishnapuram, B., Dehing-oberije, C., Lambin, P., and Raykar, V. C. (2008), “On ranking in survival analysis: Bounds on the concordance index,” in *Advances in neural information processing systems*, pp. 1209–1216.
- Swan, A. L., Mobasher, A., Allaway, D., Liddell, S., and Bacardit, J. (2013), “Application of machine learning to proteomics data: classification and biomarker identification in postgenomics biology,” *OmicS: a journal of integrative biology*, 17, 595–610.
- Tibshirani, R. (1997), “The lasso method for variable selection in the Cox model,” *Statistics in medicine*, 16, 385–395.

- Tierney, J. F., Stewart, L. A., Gherzi, D., Burdett, S., and Sydes, M. R. (2007), “Practical methods for incorporating summary time-to-event data into meta-analysis,” *Trials*, 8, 16.
- Tobin, J. (1958), “Estimation of relationships for limited dependent variables,” *Econometrica: journal of the Econometric Society*, pp. 24–36.
- Van Belle, V., Pelckmans, K., Suykens, J., and Van Huffel, S. (2007), “Support vector machines for survival analysis,” in *Proceedings of the Third International Conference on Computational Intelligence in Medicine and Healthcare (CIMED2007)*, pp. 1–8.
- Van Belle, V., Pelckmans, K., Van Huffel, S., and Suykens, J. A. (2011), “Support vector methods for survival analysis: a comparison between ranking and regression approaches,” *Artificial intelligence in medicine*, 53, 107–118.
- van Houwelingen, H. C., Bruinsma, T., Hart, A. A., van’t Veer, L. J., and Wessels, L. F. (2006), “Cross-validated Cox regression on microarray gene expression data,” *Statistics in medicine*, 25, 3201–3216.
- Van Overschee, P. and De Moor, B. (2012), *Subspace identification for linear systems: Theory—Implementation—Applications*, Springer Science & Business Media, New York.
- Van’t Veer, L. J., Dai, H., Van De Vijver, M. J., He, Y. D., Hart, A. A., Mao, M., Peterse, H. L., Van Der Kooy, K., Marton, M. J., Witteveen, A. T., et al. (2002), “Gene expression profiling predicts clinical outcome of breast cancer,” *nature*, 415, 530.
- Vapnik, V. N. and Vapnik, V. (1998), *Statistical learning theory*, vol. 1, Wiley New York, New York.
- Vinzamuri, B., Li, Y., and Reddy, C. K. (2014), “Active learning based survival regression for censored data,” in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 241–250, ACM.
- Vishwanathan, S., Smola, A. J., and Vidal, R. (2007), “Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes,” *International Journal of Computer Vision*, 73, 95–119.
- Wang, L., Li, Y., Zhou, J., Zhu, D., and Ye, J. (2017a), “Multi-task Survival Analysis,” in *2017 IEEE International Conference on Data Mining (ICDM)*, pp. 485–494, IEEE.
- Wang, P., Li, Y., and Reddy, C. K. (2017b), “Machine learning for survival analysis: A survey,” *arXiv preprint arXiv:1708.04649*.
- Wang, S., Nan, B., Zhu, J., and Beer, D. G. (2008), “Doubly penalized Buckley–James method for survival data with high-dimensional covariates,” *Biometrics*, 64, 132–140.

- Ye, L. and Keogh, E. (2009), “Time series shapelets: a new primitive for data mining,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 947–956, ACM.
- Zaas, A. K., Chen, M., Varkey, J., Veldman, T., Hero, A. O., Lucas, J., Huang, Y., Turner, R., Gilbert, A., Lambkin-Williams, R., et al. (2009), “Gene expression signatures diagnose influenza and other symptomatic respiratory viral infections in humans,” *Cell host & microbe*, 6, 207–217.