

Wayne State University Dissertations

1-1-2016

Novel Machine Learning Methods For Modeling Time-To-Event Data

Bhanukiran Vinzamuri
Wayne State University,

Follow this and additional works at: https://digitalcommons.wayne.edu/oa_dissertations

 Part of the [Computer Sciences Commons](#), and the [Library and Information Science Commons](#)

Recommended Citation

Vinzamuri, Bhanukiran, "Novel Machine Learning Methods For Modeling Time-To-Event Data" (2016). *Wayne State University Dissertations*. 1600.
https://digitalcommons.wayne.edu/oa_dissertations/1600

This Open Access Dissertation is brought to you for free and open access by DigitalCommons@WayneState. It has been accepted for inclusion in Wayne State University Dissertations by an authorized administrator of DigitalCommons@WayneState.

**NOVEL MACHINE LEARNING METHODS FOR MODELING
TIME-TO-EVENT DATA**

by

BHANUKIRAN VINZAMURI

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2016

MAJOR: COMPUTER SCIENCE

Approved By:

Advisor

Date

DEDICATION

This dissertation is dedicated to parents and my Guru.

ACKNOWLEDGEMENTS

I would like to extend my gratitude to my advisor Dr. Chandan K. Reddy for mentoring me throughout my PhD. I have learned a lot from his perseverance and his desire to excel in this extremely competitive world. I would also like to thank him for improving my technical writing skills and being approachable to discuss issues.

I am grateful to Dr. Ming Dong, Dr. Dongxiao Zhu and Dr. Kazuhiko Shinki for agreeing to be on my committee and I thank them for providing their valuable feedback on my research.

I would also like to thank all the PhD and Masters students of the Data Mining and Knowledge Discovery (DMKD) lab who have helped in building a productive and friendly work environment. Finally, I am indebted to my parents, sister and guru for providing necessary emotional and financial support to ensure the completion of my PhD.

TABLE OF CONTENTS

DEDICATION	ii
ACKNOWLEDGEMENTS	iii
LIST OF TABLES	vii
LIST OF FIGURES	ix
CHAPTER 1: INTRODUCTION	1
1.1 Time-to-event Data	1
1.1.1 An Illustrative Example	1
1.1.2 Statistical Interpretation	3
1.1.3 Main Challenges	6
1.2 Our Contributions	7
1.3 Organization	8
CHAPTER 2: PREDICTIVE MODELS FOR TIME-TO-EVENT DATA	10
2.1 Non-Parametric Methods	11
2.2 Semi-Parametric and Parametric Methods	13
2.2.1 The Proportional Hazards (PH) Model	14
2.2.2 Parametric Methods	17
2.3 Machine Learning Methods	18
2.4 Limitations of Existing Methods	19
CHAPTER 3: REGULARIZED SURVIVAL REGRESSION MODELS	21
3.1 Motivation	21
3.2 Preliminaries	25
3.3 Cox Regression with Correlation-based Regularization	27
3.3.1 FEAR-COX Algorithm	28
3.3.2 OSCAR-COX Algorithm	29
3.4 Experimental Results	35
3.4.1 Experimental Setup	35

3.4.2	Evaluation Metrics	38
3.4.3	Redundancy in Features	39
3.4.4	Visualizing Sparsity of Models	41
3.4.5	Scalability Experiments	43
3.4.6	Biomarker Validation	43
3.4.7	Discussion on Clinical Implications	45
CHAPTER 4: REPRESENTATION BASED SURVIVAL REGRESSION .		47
4.1	Motivation	47
4.2	Preliminaries	50
4.3	Calibration using the Inverse Covariance Matrix	52
4.3.1	REC Algorithm	53
4.3.2	TREC Algorithm	55
4.3.3	Algorithm Analysis	57
4.4	Experimental Results	60
4.4.1	Datasets Description	61
4.4.2	Performance Evaluation	63
4.4.3	Integrating REC and TREC with Survival Regression	64
4.4.4	Improvement in AUC with Imputed Censoring	65
4.4.5	Parameter Sensitivity Analysis	67
CHAPTER 5: STRUCTURED MODEL FOR RIGHT CENSORED DATA		70
5.1	Motivation	70
5.2	Preliminaries	72
5.3	The Proposed SLIREC Algorithm	74
5.3.1	Event Matrix Generation	74
5.3.2	Structured Regularization based Linear Regression	76
5.3.3	Optimization	79
5.3.4	Theoretical Analysis	82

5.4	Experimental Results	84
5.4.1	Implementation Details	84
5.4.2	Evaluating Importance of Structured Regularization	85
5.4.3	Evaluation using Survival Models	86
5.4.4	Goodness of Survival Prediction	88
5.4.5	Scalability Experiments	88
CHAPTER 6: ACTIVE LEARNING BASED SURVIVAL REGRESSION .		92
6.1	Motivation	92
6.2	Preliminaries	96
6.3	Active Learning with Regularized Survival Analysis	98
6.3.1	RegCox: Regularized Cox Regression	98
6.3.2	Model Discriminative Gradient-Based Sampling	101
6.3.3	Proposed ARC Algorithm	102
6.3.4	Flow Diagram of ARC	103
6.4	Experimental Results	104
6.4.1	Datasets Description	104
6.4.2	Implementation Details	106
6.4.3	Goodness of Prediction	107
6.4.4	Comparison of Sampling Strategies	109
6.4.5	Importance of Censored Samples	109
CHAPTER 7: CONCLUSIONS AND FUTURE WORK		112
APPENDIX : LIST OF PUBLICATIONS		115
REFERENCES		124
ABSTRACT		125
AUTOBIOGRAPHICAL STATEMENT		127

LIST OF TABLES

Table 2.1	Commonly used parametric distributions in survival analysis.	17
Table 3.1	Notations used in this chapter.	24
Table 3.2	An example to demonstrate right censoring for 30-day readmission . . .	24
Table 3.3	Description of EHRs used in our experiments.	38
Table 3.4	Redundancy of features selected by FEAR-COX and OSCAR-COX against feature selection algorithms.	40
Table 3.5	Survival AUC values of FEAR-COX and OSCAR-COX against state-of-the-art algorithms.	41
Table 3.6	Brier score values of FEAR-COX and OSCAR-COX against state-of-the-art algorithms.	41
Table 3.7	Statistical association between biomarkers and heart failure readmission.	45
Table 4.1	Notations used in this chapter.	51
Table 4.2	Kickstarter data statistics for 18,143 projects.	61
Table 4.3	Description of censored statistics in the Kickstarter projects.	61
Table 4.4	Basic Statistics for EHRs.	62
Table 4.5	Comparison of Survival AUC values for different regularized Cox regression algorithms without and with TREC applied on kickstarter, EHR and synthetic censored datasets.	64
Table 4.6	Comparison of Survival AUC values for different survival algorithms without and with TREC applied on kickstarter, EHR and synthetic censored datasets.	65
Table 5.1	Notations used in this chapter.	72
Table 5.2	Description of the datasets used.	84
Table 5.3	Survival AUC and standard deviation values for the SLIREC algorithm compared to other survival regression models.	89
Table 5.4	Integrated Brier score values for the SLIREC algorithm compared to other survival regression models.	89
Table 6.1	Notations used in this chapter.	97
Table 6.2	Description of the datasets.	104
Table 6.3	Comparison of Survival AUC (std) values in ARC w.r.t. different regularizers.	107

Table 6.4 Comparison of MSE (std) values of ARC w.r.t. different regularizers. . . 108

LIST OF FIGURES

Figure 1.1	A sample illustration of survival data.	2
Figure 2.1	Categorization of machine learning methods for time-to-event data. . .	10
Figure 2.2	Kaplan Meier curve for data in Figure 1.1.	12
Figure 3.1	Correlation Heat maps for visualizing the diverse correlation structure present in EHRs	22
Figure 3.2	Patient readmission cycle at a hospital.	36
Figure 3.3	Boxplot visualizing the regression coefficients of the sparse variables selected by the regularized Cox regression algorithms.	42
Figure 3.4	Scalability w.r.t the number of instances.	44
Figure 3.5	Scalability w.r.t the number of features.	44
Figure 4.1	Flow diagram for the proposed calibrated survival analysis approach on a EHR dataset.	49
Figure 4.2	Percentage of right censored instances in EHR and Kickstarter datasets.	61
Figure 4.3	Survival AUC plots obtained for calibrated synthetic and EHR datasets using REC, TREC, SoftImpute and Misglasso methods.	66
Figure 4.4	Survival AUC plots obtained for calibrated kickstarter datasets using REC, TREC, SoftImpute and Misglasso methods.	67
Figure 4.5	Runtime on Kickstarter dataset using L_1 , L_2 norms in TREC	68
Figure 4.6	Iterations for convergence using L_2 norm based TREC	68
Figure 5.1	Illustrative example of SLIREC algorithm on a sample right censored dataset.	73
Figure 5.2	Visualizing structure in the event matrices for two survival datasets. . .	76
Figure 5.3	Performance comparison using CCA, OPLS, SSL and SLIREC methods on various survival datasets.	87
Figure 5.4	Measuring improvement in runtime before and after applying the approximation scheme in SLIREC for Lung dataset.	90
Figure 5.5	Measuring improvement in runtime before and after applying the approximation scheme in SLIREC for DLBCL dataset.	91
Figure 6.1	Survival Regression viewed as a binary classification problem.	93
Figure 6.2	Active learning cycle for time-to-event data.	94
Figure 6.3	Block diagram of the active learning framework with KEN-COX regression.	103

Figure 6.4	Readmission probabilities for patients computed within 30, 60 and 90 days post discharge from index hospitalization.	106
Figure 6.5	Comparison of the active learning rates of ARC with 4 different regularizers over real-world and synthetic datasets.	110
Figure 6.6	Censoredness plot for Breast and Colon datasets.	111

CHAPTER 1: INTRODUCTION

Knowledge extraction from time-to-event data [1–4] is an upcoming field of research which has wide utility in different real-world applications such as healthcare, finance and engineering. Time-to-event data is different from other forms of relational data, as this data has a unique temporal structure. In such data, a domain expert monitors the occurrence of a defined event of interest. In addition, the duration of study is also defined here within which the expert monitors the event occurrence.

1.1 Time-to-event Data

In many studies, the outcome of interest is related to the timing of the occurrence of an event. This can be explained by considering a clinical setting, where one may be interested in measuring how long a chronically ill patient survives after receiving a certain treatment. In another scenario, one may be interested in determining which of the three drugs, compared to a placebo, provides symptom relief most rapidly.

Imagine that a hospital is interested in monitoring the survival status of a patient following a first heart attack. The study could begin when the first patient, following his or her first heart attack, is randomly assigned to a follow-up program, with additional patients enrolled through time. Conversely, the study could begin with a cohort of subjects, each of whom has had their first heart attack, and were randomly assigned to a follow-up program. In either case, there are potentially three outcomes that could occur with each patient, with the event of interest being the death of the patient. These are (1) the patient dies; (2) the patient drops out of the study thereby becoming a loss to follow-up which could occur for any number of reasons, such as relocating geographically; or (3) the event of interest does not occur to the patient during the period of study. These three mutually exclusive events are the foundation for survival analysis studies.

1.1.1 An Illustrative Example

We present an illustration of time-to-event survival data which demonstrates the three cases mentioned above. In Figure 1.1, we present a simple example where 5 patients are

studied and the event of interest is the death of the patient. Subject C and Subject E have the event of interest recorded for both of them, whereas Subject B and Subject D survive the entire observation period. Their survival time is known to be for a length of time that is greater than the length of the study. This is known as *type I censoring*. Subject A drops out of the study after 6 months.

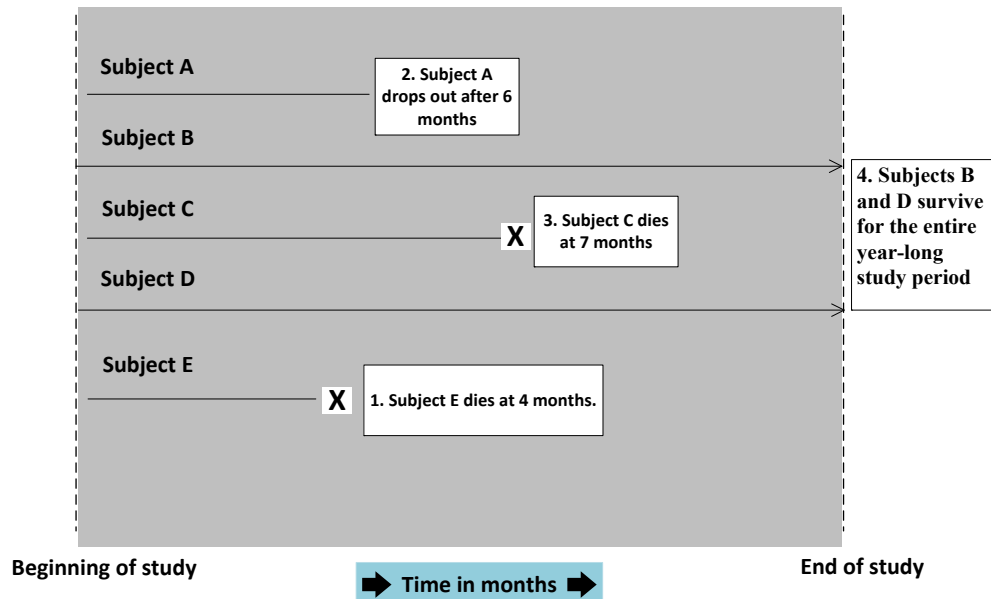


Figure 1.1: A sample illustration of survival data.

Such instances for which the exact endpoints are not known, because the subject dropped out of the study, was withdrawn from the study, or survived beyond the termination of the study are called *right censored data*, because the survival times extend beyond the right tail of the distribution of survival times. Generally, for purposes of analysis, a dichotomous, or indicator, variable is used to distinguish survival times of those subjects who experience the event of interest and those that do not because of one of the censoring mechanisms described above. Typically, this variable is called a status variable, with a zero indicating that an event did not occur and hence the survival time is censored, and a 1 indicating that the event of interest did occur.

Although, a vast majority of published research using survival analysis methods is clinical

in nature, it should be noted that there are many non-clinical uses for survival analysis as well. With the advent of computer-based statistical programs to help with complex calculations, the use of survival analysis methodologies has increased demonstrably among many disciplines. For example, engineers may wish to know the time it takes for a battery to lose its charge, a quality-control scientist at a manufacturing plant may wish to understand at which point machines need to be recalibrated, or an ecologist may want to estimate how long the average carcass remains in a study area before it is scavenged.

1.1.2 Statistical Interpretation

We now present the mathematical concepts for interpreting time-to-event data, and we also emphasize on data distributions commonly encountered in such analyses. Time-to-event data are distributed temporally, such that events occur either at some point, or within some interval of time. These events are considered to represent a random variable having some probability of occurrence at each time period for each subject in the study. To set up the framework for survival data with right censoring, two random variables need to be defined namely, T_{surv} and T_{cens} . The former corresponding to the survival time and the latter corresponding to the censoring time. A common censoring scheme applied to survival data is *administrative censoring*, where the censoring time is determined by the termination of the study. The crucial condition for the kind of survival analysis discussed here is that the survival time T_{surv} and the censoring time T_{cens} are independent. For both the random variables the cumulative distribution functions $F_{surv}(t)=P(T_{surv} \leq t)$ and $F_{cens}=P(T_{cens} \leq t)$. The survival function $S(t)$ and the censoring function $G(t)$ are defined as given in Eq. (1.1). The observed survival time is labeled as the minimum of T_{surv} and T_{cens} . The censoring indicator δ is set to 0 if $T=T_{cens}$ and 1 if $T=T_{surv}$. A related concept is the cumulative hazard function denoted by $H(t)$. $H(t)$ and $S(t)$ are closely related as in $H(t)=-\ln(S(t))$ and $S(t)=\exp(-H(t))$

The cumulative distribution function (cdf) represents the probability that an event time is less than or equal to some specified measurement time t . $F(t)$ is an increasing function

that runs from a value of zero (it is assumed theoretically that no events have occurred at the initiation of the study), to a value of 1 (it is assumed theoretically that all events have occurred at the conclusion of the study). In the context of survival analysis, a closely related function that is more commonly used than $F(t)$ is a function that runs from a value of 1 (it is assumed that all subjects at the initiation of the study have survived to that point) to a value of zero (it is assumed theoretically that none of the subjects have survived when the study ends, though some subjects may be censored). Conveniently, this is known as the survival distribution, $S(t)$, and is mathematically related to the cumulative distribution function as mentioned in Eq. (1.1).

$$F(t) = P(T \leq t) \tag{1.1}$$

$$S(t) = 1 - F_{surv}(t) = P(T_{surv} > t)$$

$$G(t) = 1 - F_{cens}(t) = P(T_{cens} > t)$$

The probability distribution function is represented by the set of probabilities that specify the possible values of a random variable. In the context of survival analysis, this density function represents the probability of an event occurring in a defined interval of time. Although, fully appreciating the intricacies of this probability distribution requires knowledge of calculus, we can illustrate its meaning conceptually by using some of the properties of the normal distribution. When we calculate the probabilities for the normal distribution, we are interested in calculating the area under a curve that was bounded by two values. Similarly, in survival analysis we are interested in calculating the probability of an event bounded by an interval of time, say Δt and then finding our probability as the interval becomes very small, that is as $\Delta t \rightarrow 0$. Hence, the probability distribution function, $f(t)$, is defined by Eq. (1.2).

$$f(t) = \frac{\Delta F(t)}{\Delta t} = -\frac{\Delta S(t)}{\Delta t} \tag{1.2}$$

That is, the set of probabilities of events that occur in an infinitesimally small interval of

time defines the probability function. It is also possible to find this function by examining what happens during a change in $F(t)$, say $\Delta F(t)$, or a change in $S(t)$, say $\Delta S(t)$, in a given interval of time.

The mean survival time is another important metric of interest in survival analysis, which is defined as $\mu = \mathbb{E}[X]$ and which maybe further expressed as $\mu = \int_0^\infty S(t)dt$. This gives the expected lifetime for an individual with a given survival function which is sometimes needed in survival analysis to estimate the life expectancy. Although the hazard function is difficult to estimate directly in survival analysis, it plays an important role in understanding the process of survival. A decreasing hazard function implies that the prognosis gets better as you live longer, and an increasing hazard function implies that the prognosis gets worse as you live longer.

Finally, a function that is often encountered in survival analysis is the hazard function, $h(t)$. This function is used to define the instantaneous probability of an event occurring, given that the subject has survived up to a given time t . This function is defined as in Eq. (1.3).

$$h(t) = \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t} \quad \Delta t \rightarrow 0 \quad (1.3)$$

$$h(t) = \frac{f(t)}{S(t)}$$

This function is based on a conditional probability, wherein we are interested in calculating the probability of an event occurring given that the subject has already survived until a particular time point. The condition of having already survived to a given time means that the probability of surviving into the future is influenced by having already survived previous time periods. This idea can be very important in some situations, where surviving the early stages of a disease may dramatically decrease the potential of an event occurring in the near future. As an example, consider cancer where non-recurrence, or remission, for a period of 5 years generally increases survivorship. This function can also be expressed in terms of the two functions previously defined in Eq. (1.3). This expression is defined so

because, the hazard function can exceed 1, so it is not truly a probability, though it is based on the conditional probability of an event occurring. The hazard function is often defined in survival analysis by a known distribution such as the lognormal, exponential, or a Weibull distribution.

1.1.3 Main Challenges

We now explore the challenges that will be addressed in this dissertation while building models for time-to-event data which are as follows.

- **Correlation:** In longitudinal data, it is observed that the data exhibits different kinds of correlations which are as follows (i) *Inter-event correlation*: Instances for which the events are observed tend to be correlated with each other. For example, two patients who were readmitted within 30-days of discharge for a disease, most often have a lot of similarity in the actions which triggered their readmission. (ii) *Intra-event correlation*: The covariates of an electronic health record (EHR) for a patient have a non-uniform effect in determining the survival status, which is why this *intra-event correlation* is an extremely important factor in predicting event occurrence.
- **Missing information:** During the period of observation, the events are not observed for all the patients, because of several reasons such as loss of follow-up or early termination of the study. In such cases, these patients do not have any time-to-event labels associated with them and they are called as right censored instances, as they only have a censoring time associated with them. This causes a significant problem while learning models from time-to-event data.
- **Lack of adaptability:** Prominent machine learning methods such as linear regression offer several benefits when applied for prediction on right censored data. However, existing linear regression-based methods are non-intuitive and they rely on user specified parameters to interpret the censoredness from the data. In such scenarios, it is extremely desirable to extend the linear regression model to right censored data, so that it can interpret the inherent patterns (such as the underlying structure of the data)

and use this knowledge extensively for prediction. The motivation for this approach is that it does not rely on user specified parameters and it can make the linear regression model more adaptable to different distributions of events and censored instances in the data.

- **Insufficient training instances:** Models built on such data rely heavily on the quality of training data available. Instances for which the event is observed have an event label defined and they can be added to the model directly. However, including censored instances in the model which do not have labels, has to be decided judiciously by assessing the influence of adding the instance in the model. This is a non-trivial task and the model has to include only those censored instances in the training data, which are having a significant impact on the model performance.

1.2 Our Contributions

We now mention the major contributions of the proposed machine learning models for time-to-event data. They are as follows.

- We address the problem of building survival models which can infer *intra-event correlation* by proposing two diverse regularizers. We address two forms of *intra-event correlation* which are feature based correlation and grouped correlation, respectively. Correlation among features in survival data is addressed using the FEAture Regularized Cox (FEAR-COX) algorithm. Grouped correlation (structured sparsity) among covariates in survival data is addressed using the Octagonal Shrinkage Clustering Algorithm Regression (OSCAR-COX). The performance of these algorithms is studied exhaustively on longitudinal EHRs obtained from a large hospital. These regularizers are also compared to state-of-the-art regularization methods in the literature.
- We propose a representation learning method for imputing times for the censored instances, which estimates the censored times by inferring the correlation pattern among different censored instances. This method uses a novel two-dimensional imputation approach which incorporates the *inter-event* and *intra-event correlation* to estimate

the time-to-event label for censored instances using a sparse inverse covariance based imputation method. This is called the Transposable REgularized covariance based calibration method (TREC). This learned new representation of the original survival dataset using TREC is then used for subsequent survival analysis.

- We present a method called Structured regularization based LInear REgression algorithm for right Censored data (SLIREC) which infers the *underlying structure* of the survival data directly and uses this knowledge to guide the base linear regression model. This structured approach is more robust compared to the standard statistical and Cox-based methods, as it can automatically adapt to different distributions of events and censored instances in the dataset which is very useful when dealing with different real-world datasets.
- We propose an active learning based survival regression method which can efficiently identify important censored instances from the survival dataset which are contributing most in order to build an effective survival model. This active learning method is generic as it uses the gradient of the loss function employed in the learning algorithm. We implement this active learning approach using the regularized Cox regression framework to present the Active Regularized Cox (ARC) algorithm.

1.3 Organization

This dissertation is organized as follows. In Chapter 2, we present an overview of popular survival analysis methods such as non-parametric, semi-parametric and ensemble methods, and conduct an in-depth literature study on different models for time-to-event data in these three categories. In Chapter 3, we present our regularized Cox regression algorithms which proposes two correlation-based regularizers to handle diverse intra-event correlation in longitudinal data. In Chapter 4, we present our representation learning based survival regression algorithm which is successful in learning a novel representation for survival data. This algorithm infers the time-to-event label using an imputation-based inverse covariance method. In Chapter 5, we present our Structured regularization based LInear REgression algorithm

for right Censored data (SLIREC) which extends the linear regression model by making it more adaptable to right censored data. In Chapter 6, we present our active learning-based survival regression approach, which is the first method to successfully use the active learning methodology for survival data to obtain a model with more informative and lesser number of training instances. Finally, in Chapter 7, we draw conclusions from all these algorithms and briefly discuss methods for extending them.

CHAPTER 2: PREDICTIVE MODELS FOR TIME-TO-EVENT DATA

In this section, we present the related work on existing machine learning methods proposed for time-to-event data. We provide a flow diagram in Figure 2.1 which represents different kinds of methods described in this chapter.

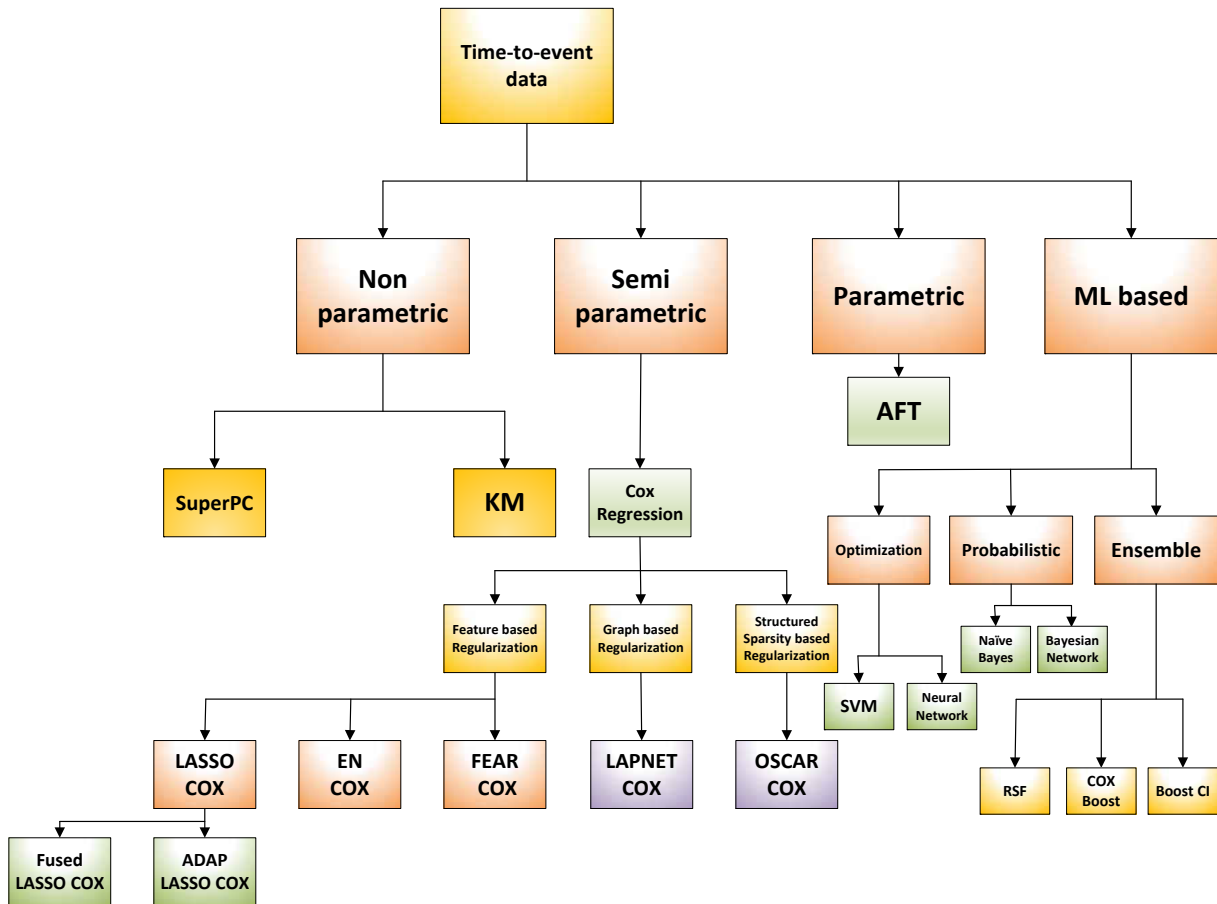


Figure 2.1: Categorization of machine learning methods for time-to-event data.

In this chapter, we discuss about different non-parametric estimation methods [4, 5]. This is followed by explaining the most important semi-parametric method studied in the survival analysis literature called Cox regression [6]. We study its partial log-likelihood formulation in detail. We segregate the existing methods for time-to-event data into three categories, namely, non-parametric, semi-parametric and parametric methods. Non-parametric methods such as the Kaplan-Meier (KM) and Nelson-Aalen (NA) estimator directly conduct inference

on the data without making any assumptions about the distribution. Semi-parametric methods make a trade-off between non-parametric and parametric methods by trying to extract information from the covariates present in the dataset, and they do not make any additional assumptions on the distribution of the hazard function [1–3].

Parametric methods, on the other hand, make assumptions a priori on the distribution of the functions involved completely and conduct maximum likelihood estimation for learning the model parameters directly. These methods make assumptions which seem to be confined to a fixed distribution alone while conducting survival analysis which need not be the case with survival data sampled from multiple distributions. This is the main motivation to prefer semi-parametric methods over parametric methods in survival analysis. We now study each of these methods in detail in the following sections.

2.1 Non-Parametric Methods

Non-parametric methods are used frequently for survival analysis as they make no assumptions about the hazard function and conduct estimation. We start by explaining the Kaplan-Meier (KM) estimator. The starting point for the KM estimator is considering a sample of n independent observations $(t_1, \delta_1), (t_2, \delta_2), \dots, (t_n, \delta_n)$ from (T, δ) . The following notation is introduced from the field of counting processes.

$$\hat{S}_{KM}(t) = \prod_{s \leq t} \left(1 - \frac{\Delta \bar{N}(s)}{\bar{Y}(s)} \right) \quad (2.1)$$

Let $Y_i(t) = \mathbf{1}\{t_i \geq t\}$, $\bar{Y}(t) = \sum_{i=1}^n Y_i(t)$, $N_i(t) = \mathbf{1}\{t_i \leq t, \delta_i = 1\}$, $\bar{N}(t) = \sum_{i=1}^n N_i(t)$. The risk set $R(t) = \{i; t_i \geq t\}$ represents the set of instances who are at risk at a given time t , and $\Delta \bar{N}(t)$ is the number of events at time t . This KM estimator is one of the most widely used non-parametric estimators of the survival function. It can be interpreted as a conditional survival function resulting from a partitioning of the time scale and estimating the survival function on each partitioning. In Figure 2.2, we plot the KM curve for the data considered in Figure 1.1. This KM estimator can also be defined similarly for the censoring function

$\hat{G}_{KM}(t)$ and the hazard functions $\hat{H}_{KM}(t)$.

$$H(t) = \int_0^t h(s) ds \quad (2.2)$$

$$\hat{H}_{KM}(t) = -\ln(\hat{S}_{KM}(t)) = \sum_{s \leq t} \ln\left(1 - \frac{\Delta \bar{N}(s)}{\bar{Y}(s)}\right)$$

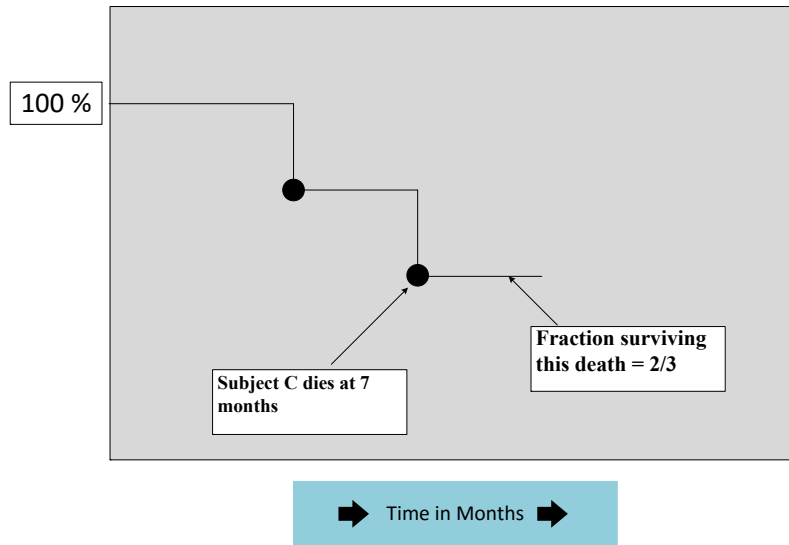


Figure 2.2: Kaplan Meier curve for data in Figure 1.1.

We now look at other methods for non-parametric survival analysis such as supervised principal components (SuperPC). This is a method which selects features from survival data which have a direct effect on the time-to-event. The steps involved in this algorithm involve computing the standard regression coefficients for each feature and then form a reduced data matrix consisting of only those features whose univariate coefficient exceeds a pre-defined threshold in absolute value. Subsequently, the selected principal components of the reduced data matrix are used in a regression model to predict the time-to-event outcome [7].

Multiple imputation for censored data is a method where the failure times are imputed using an asymptotic data augmentation scheme based on the current estimates and the baseline survival curve [8]. Once this is done a standard procedure such as Cox regression is

applied to the imputed data to update the estimates. A similar problem has been dealt with in the crowdsourcing domain which predicts the time-to-event directly using the survival function [9, 10]. Misglasso is an extension to the approach for imputing missing values by using the graphical lasso algorithm [11–13]. Other popular approaches include the SoftImpute algorithm which uses a nuclear norm minimization subject to constraints to fill the missing entries [14].

Risk stratified imputation in survival analysis is another approach which performs stratified imputation of missing time-to-events based on groups of patients who are similar to each other. The stratification is done to ensure that not too many samples are imputed, and all the imputation is done among censored instances which are similar to each other. An auxiliary variable approach to multiple imputation in survival analysis is proposed here [15] with the goal to improve efficiency using Monte Carlo methods.

2.2 Semi-Parametric and Parametric Methods

Cox regression is a semi-parametric method which uses the proportionality hazards (PH) assumption. It is widely used because of its effective performance and ease of availability. However, due to its maximum likelihood-based formulation, Cox regression tends to overfit data. This problem is solved by introducing regularizers into Cox regression to reduce the variance of the obtained solution.

The Lasso regularizer which is based on the L_1 norm was integrated with the partial log-likelihood function and the corresponding optimization problem was solved using the iteratively reweighted least squares algorithm. Lasso provides sparse solutions, but when selection has to be conducted over several correlated variables it selects one variable and does not consider the remaining variables. The elastic net regularizer which uses a convex combination of the L_1 and L_2 norms is effective for correlated survival data and the elastic net Cox (EN-COX) algorithm is implemented using a cyclic coordinate descent algorithm [16]. The computation in this algorithm is accelerated by approximating the Hessian computation involved. To incorporate more feature-based information, graph regularization has also been

used with Cox regression, where the graph Laplacian is used as a penalty [17–19]. The graph laplacian can successfully capture feature similarities through its structure and this penalty tries to keep similar coefficient values for connected features in the graph. Such graph-based regularized Cox models can also be stabilized using the regularizers built on the feature graph. The Jaccard graph is used here to obtain stability in the model. In a similar way, the adaptive lasso regularizer can also be used with Cox regression which improves performance over the LASSO-COX model [20–22]. In this algorithm, LASSO-COX is applied on the survival dataset, then the inverse of the obtained regression coefficients are used as weights to run further rounds of the LASSO-COX algorithm. This approach was observed to be more biased towards features initialized with higher weights, but it obtained superior performance in many cases. Fused-lasso is a similar regularizer which imposes sparsity on the model by imposing temporal smoothness among the regularizer coefficients to ensure stability of the coefficient values. Regularizers such as *scout* were also integrated with Cox regression [23]. These come under the supervised covariance-based regression models which consider the covariance matrix over the features and impose sparsity on it. The inverse covariance matrix represents the partial correlations between different features present in the dataset and this method can be considered as a minor variant of the feature correlation-based regularizers described above.

2.2.1 The Proportional Hazards (PH) Model

A very popular model in survival analysis is the proportional hazards (PH) model where individual specific hazard functions $h_i(t)$ are learned, and the proportional hazards assumption is made as given in Eq. (2.5). In this model, c_i is constant and $h_0(t)$ is a baseline hazard function which is left unspecified. The c_i term can be replaced with $\exp(X^T\beta)$ to obtain the Cox PH model [24].

$$\begin{aligned}
 h_i(t) &= c_i h_0(t) \\
 h(t|X) &= h_0(t) \exp(X^T\beta)
 \end{aligned}
 \tag{2.3}$$

The effect of the covariates on the hazard can be modeled by taking $c_i = \exp(X_i^T \beta)$. The survival function implied by the model is given in Eq. (2.4) where $H_0(t) = \int_0^t h_0(s) ds$ is the cumulative baseline hazard and the baseline survival function is given as $S_0(t) = \exp(-H_0(t))$.

$$S(t|X) = \exp(-\exp(X^T \beta) H_0(t)) = S_0(t)^{\exp(X^T \beta)} \quad (2.4)$$

A key feature of Cox regression model is that the hazard function of two individuals with covariates X' and X'' respectively are proportional. This can be expressed as in Eq. (2.5) and this ratio is constant over time. This ratio is called the relative risk for an individual with covariates X' compared to X''

$$\frac{h(t|X')}{h(t|X'')} = \frac{h_0(t) \exp(X' \beta)}{h_0(t) \exp(X'' \beta)} = \exp[\beta^T (X' - X'')] \quad (2.5)$$

The Cox regression model learns the regression coefficient vectors in survival analysis using a method called the partial likelihood estimation. We now explain this concept by first looking at the complete log-likelihood using the information summarized in the dataset in the form of triplets (T, δ, X) as in Eq. (2.6)

$$L(h_0, \beta) = \sum_{i=1}^n (-H_0(t_i) \exp(x_i^T \beta) + \delta_i (\ln(h_0(t_i)) + x_i^T \beta)) \quad (2.6)$$

Expanding this equation using $H_0(t) = \sum_{s \leq t} h_0(s)$ gives us Eq. (2.7) and for a fixed value of β this expression is maximal for the Breslow estimator, which is given as in Eq. (2.8).

$$L(H_0, \beta) = \sum_t (-h_0(t) \sum_i Y_i(t) \exp(x_i^T \beta) + \ln(h_0(t)) \Delta \bar{N}(t) + \sum_i \Delta N_i(t) x_i^T \beta) \quad (2.7)$$

Substituting the Breslow estimator in Eq. (2.7) gives us Eq. (2.9). In this equation, $pl(\beta)$ represents the partial log-likelihood which is given in Eq. (2.10).

$$\hat{h}_0(t|\beta) = \frac{\Delta \bar{N}(t)}{\sum_i Y_i(t) \exp(x_i^T \beta)} \quad (2.8)$$

A simplified and more often used version of the partial likelihood is provided in Eq. (2.11).

$$L(\hat{h}_0(\beta), \beta) = pl(\beta) + \sum_t \left(-\Delta \bar{N}(t) + \ln(\Delta \bar{N}(t)) \right) \quad (2.9)$$

$$pl(\beta) = \sum_{i=1}^n \int_0^{\infty} \ln \left(\frac{\exp(x_i^T \beta)}{\sum_j Y_j(t) \exp(x_j^T \beta)} \right) dN_i(t) \quad (2.10)$$

$$pl(\beta) = \prod_{i=1}^k \frac{\exp(\beta^T X_i)}{\sum_{j \in R_i} \exp(\beta^T X_j)} \quad (2.11)$$

Finally, we also present the logarithmic version of this partial likelihood which is often used for MLE estimation in Cox regression in Eq. (2.12).

$$l(\beta) = \log(pl(\beta)) = \sum_{i=1}^k \beta^T X_i - \sum_{i=1}^k \log \left(\sum_{j \in R_i} \exp(\beta^T X_j) \right) \quad (2.12)$$

The computation of the partial log-likelihood is changed if we consider tied event times in survival data. For all the k unique time-to-event values we use d_i to represent the number of times t_i re-occurs in the survival data and D_i to represent the set of indices with time t_i . In addition, we also let $s_i = \sum_{j \in D_i} X_j$ then the approximations proposed by Breslow and Effron can be found in Eq. (2.13).

$$l(\beta)_{breslow} = \prod_{i=1}^k \frac{\exp(\beta^T s_i)}{\sum_{j \in R_i} \exp(\beta^T X_j)^{d_i}} \quad (2.13)$$

$$l(\beta)_{effron} = \prod_{i=1}^k \frac{\exp(\beta^T s_i)}{\prod_{j=1}^{d_i} [\sum_{h \in R_i} \exp(\beta^T X_h) - \frac{j-1}{d_i} \sum_{l \in D_i} \exp(\beta^T X_l)]}$$

The computation in Cox regression consists of maximizing the partial log-likelihood given in Eq. (2.12) with or without the ties adjustment as given in Eq. (2.13), and then using the estimated regression coefficient vector in the estimating functions given in Eq. (2.14) to obtain the time dependent survival function. There are two ways to estimate the survival function one is using the NA estimator and the other is using the analogue of the KM

estimator which is called the Product-Limit (PL) estimator. These are provided in Eq. (2.14).

$$\begin{aligned}\hat{S}_{NA}(t|X, \hat{\beta}) &= \exp\left(-\hat{H}_0(t)\exp(x^T \hat{\beta})\right) \\ \hat{S}_{PL}(t|X, \hat{\beta}) &= \prod_{s \leq t} \left(1 - \exp(x^T \hat{\beta}) \hat{h}_0(s)\right)\end{aligned}\tag{2.14}$$

2.2.2 Parametric Methods

Parametric methods differ from semi-parametric methods, as these assume that the hazard distribution is specified. The accelerated failure time (AFT) model is one of the popular parametric survival models. We look at its formulation briefly. The AFT model assumption is stated in Eq. (2.15) where S_0 is the baseline survival function. From this equation it is seen that covariates act multiplicatively on time so that their effect is to accelerate or decelerate time-to-event relative to the baseline survival function.

$$S(t|X) = S_0(t)^{\exp(X^T \beta)}\tag{2.15}$$

An equivalent popular formulation of AFT-model is the following linear regression formulation for the log-transformed event time $\log(T)$ given X as in Eq. (2.16).

$$\log(T) = -X^T \beta + \epsilon.\tag{2.16}$$

In this equation ϵ is assumed to be independent of X . The AFT model is appealing due to its direct relationship between the failure time and the covariates. The semi-parametric version of the AFT model is computationally intensive, but if we are willing to specify a form for the baseline function, then the AFT model is fully parametric. We also specify some of the commonly used parametric distribution in survival analysis in Table 2.1.

Table 2.1: Commonly used parametric distributions in survival analysis.

Distribution	Hazard Rate	Survival Function	Probability Density Function
Exponential	λ	$\exp(-\lambda t)$	$\lambda \exp(-\lambda t)$
Weibull	$\alpha \lambda t^{\alpha-1}$	$\exp(-\lambda t^\alpha)$	$\alpha \lambda t^{\alpha-1} \exp(-\lambda t^\alpha)$
Log-Normal	$\frac{f(t)}{S(t)}$	$1-I(\lambda t, \beta)$	$\frac{\lambda^\beta t^{\beta-1} \exp(\lambda t)}{\tau(\beta)}$

Elastic net Buckley James (EN-BJ) [25] is a method which directly models the response for events using the least squares method, and for the censored instances the response variable is imputed using the conditional expectation values given the corresponding censoring times and covariates. This algorithm uses the elastic-net regularization term with this AFT model and was applied on high-dimensional genomic data obtaining good performance.

2.3 Machine Learning Methods

In this section, we present the machine learning methods used for analyzing time-to-event data. We categorize these methods into three categories, namely, (i) Bayesian-based (ii) Optimization-based and (iii) Ensemble-based. Censored Naive Bayes (CensNB) is a bayesian approach which applies the standard Naive Bayes algorithm for censored data [26]. In this algorithm, the conditional survivor function is learned by initializing the functions using non-parametric densities, which are then subsequently smoothed using a weighted loess smoother. These models use an approach called inverse probability of censoring weighting (IPCW) for each of the records in the dataset. This is a method which applies weights to the censored instances in order to account for censoring when compared to uncensored instances (events). Similarly, bayesian networks based data imputation has also been used to enhance the performance of survival trees. The imputation on missing instances is done using the bayesian network computed on complete instances and the model has shown to perform well in clinical trials.

We now look at ensemble methods for time-to-event data. The first method in this category is CoxBoost which applies the boosting based paradigm to the Cox regression algorithm by building a set of weak learners and learning their weights iteratively. A more refined boosting framework for survival data is based on boosting the concordance index (BoostCI) [27]. This method is based on optimizing the evaluation metric such as concordance index (survival AUC) directly, rather than optimizing the maximum likelihood, which has been observed to perform better in several scenarios. The algorithm computes the negative gradient of the concordance index and fits it separately to each of the components of

X and continues this until convergence is observed.

Random Survival Forests (RSF) is an ensemble method which uses a forest of survival trees for prediction [28]. The basic intuition of the RSF algorithm is explained as follows. The algorithm begins by drawing B bootstrap samples from the original data where 37% of the data is excluded in each sample which is also called out-of-bag (OOB) data. A survival tree is grown for each sample, where at each node we randomly select p candidate variables. The node is split using the candidate variables that maximizes survival difference between the daughter nodes. A constraint is used so that no terminal node has less than d_0 unique deaths. An ensemble cumulative hazard function is calculated using the cumulative hazard function for each tree and the OOB data is used to evaluate the model. This approach was found to provide competitive performance for many survival datasets.

Apart from this method other extensions to the linear regression model have been studied extensively in the context of multi-response prediction. These include methods such as canonical correlation analysis (CCA) [29], orthonormalized partial least-squares (OPLS) [30] and shared subspace learning (SSL) [31] which attempt to reduce the dimensionality of high-dimensional data and try to learn a projected representation onto the lower dimensional space. Subsequently, regression models built on the learned projected space perform more effectively.

2.4 Limitations of Existing Methods

We now look at the limitations of the methods discussed above. Non-parametric methods such as KM, NA, CensNB and SuperPC are flexible to use, but they do not conduct any form of inference on the survival data. Real-world survival data often needs some additional interpretation through the form of methodical inference of its properties, so that effective models can be built on them. In this dissertation, in Chapter 4, we study two different representation learning-based algorithms which modify the representation of survival data by capturing inherent properties such as *intra-event* and *inter-event* correlations in the dataset. This helps in deciphering patterns in the survival data which can enhance the predictive

power of the base model.

Semi-parametric regression methods such as Cox regression suffer from the overfitting problem, due to its MLE formulation. Traditional real-world survival data has complex patterns which cannot be deciphered using simple regularizers such as the lasso and ridge. In Chapter 3, we study more advanced properties in survival data such as structured sparsity in the form of grouped correlation to propose regularizers to handle such data.

Survival data consists of both events and censored instances, and the model is built using information from both these sources. However, the reliability of the information obtained from censored instances is ambiguous, as they do not have defined time-to-event labels. The models mentioned above directly use the censored times for these instances during model building, which is inappropriate. In this problem, it is extremely important to determine the influence of a censored instance on the model before including it in the training model. The survival models mentioned above do not address this labelling problem with survival data which is crucial to build reliable and effective models. These issues are addressed in detail in Chapter 6.

CHAPTER 3: REGULARIZED SURVIVAL REGRESSION MODELS

3.1 Motivation

The necessity to build correlation-based regularized Cox regression algorithms can be explained by considering the heterogeneous nature of electronic health records (EHRs) [32–35]. A typical EHR can be obtained by concatenating data from several resources such as demographics, comorbidities, procedures, medications, labs and insurance information. We segregate all this information from a real EHR cohort considered in this dissertation, and we plot the canonical correlation heatmaps between each of these groups. Canonical correlation captures the correlation patterns among multi-dimensional datasets with the same number of instances and different number of features by calculating the weights of the projection vectors which maximize the correlation between these two datasets in the projected space. This makes canonical correlation heat maps ideal to visualize the diverse correlation structure in EHRs.

The correlation heat maps in Figure 3.1 indicate that the intensity of correlation among the 6 different subgroups of EHRs are high, and this correlation pattern should be effectively utilized by an algorithm to obtain accurate predictions. One can also observe that as the correlation patterns are not uniform, which indicates the necessity to build complex regularizers that can account for such heterogeneous and non-uniform grouped correlation structure in EHRs.

In this chapter, we propose two algorithms which integrate novel regularizers in the Cox regression framework, which addresses two forms of intra-event correlation, which are feature correlation and grouped correlation, respectively. We present a generalized framework which converts Cox regression to a modified least squares problem using the gradient and Hessian information from the partial log-likelihood of Cox regression. This framework can be integrated with any regularizer using the corresponding regularized least squares version solver. For example, the traditional *shooting* LASSO least squares solver [36] can be integrated di-

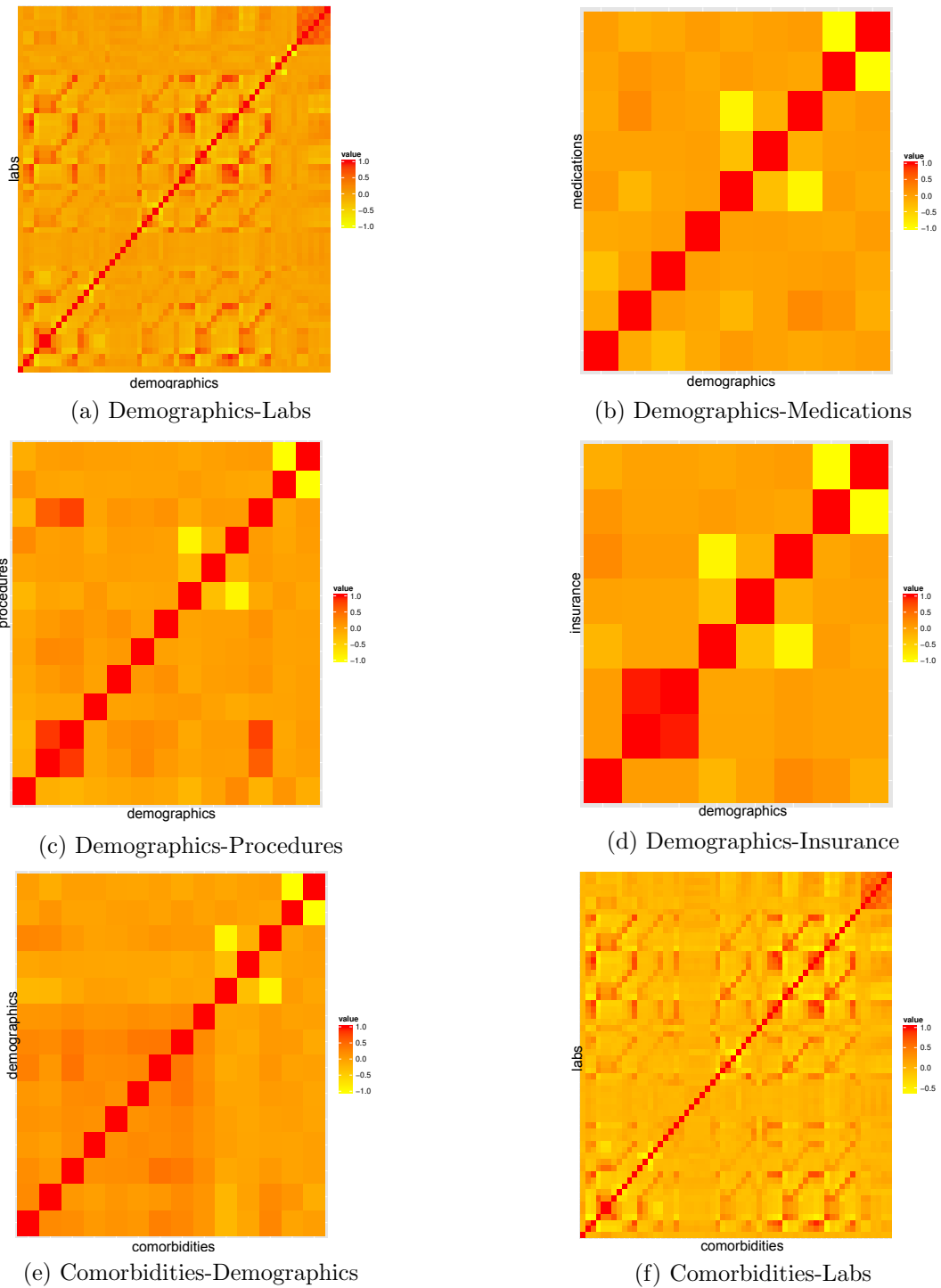


Figure 3.1: Correlation Heat maps for visualizing the diverse correlation structure present in EHRs

rectly with this framework to provide a more effective solution for LASSO-COX. We use this framework exhaustively, and study a total of 7 regularized Cox regression methods of which we implement 4 methods, namely, (fused-lasso (FLASSO-COX), adaptive-lasso (ALASSO-COX), feature regularized (FEAR-COX) and oscar (OSCAR-COX) using this least squares framework.

We propose a *Feature Regularized Cox regression* (FEAR-COX) algorithm which uses a novel feature-based regularizer with the modified least squares formulation of Cox regression. Our experimental results demonstrate that this method is more effective than the elastic net at handling correlated features. The novel pairwise feature similarity regularizer in this method is obtained using a convex formulation which uses a positive semi-definite matrix. We propose a *graph-based OSCAR* (Octagonal Shrinkage and Clustering Algorithm for Regression) regularized Cox regression method (OSCAR-COX) which uses the oscar regularizer [37] based least squares solver with the modified least squares formulation of Cox regression. This method is effective since it can capture structured sparsity (grouped correlation) among the feature sets in EHRs. It exploits the graph structure of the features in the dataset to capture this unique phenomenon in EHRs.

We demonstrate the improved discriminative ability of FEAR-COX and OSCAR-COX using standard evaluation metrics in survival analysis such as concordance index (c-index) and brier score. We also demonstrate the non-redundancy of the features selected by the sparse models of FEAR-COX and OSCAR-COX and also visualize the sparsity of our proposed models. In addition, we use the parsimonious models from FEAR-COX and OSCAR-COX to identify important biomarkers for heart failure readmission from EHRs. We validate the biomarkers identified using well known survey studies from the clinical informatics literature.

We now present a synthetic example which illustrates how patients are right censored in an EHR setting. In Table 3.2, we consider a simple EHR dataset consisting of 4 instances. In this example, the time is measured in days. The censoring time is set to 30 days for all

Table 3.1: Notations used in this chapter.

Name	Description
X	$n \times m$ matrix of feature vectors
T	$k \times 1$ vector of sorted unique failure times
R_i	risk set of all patients j such that $(t_j \geq t_i)$
d_i	number of patients readmitted within time t_i
δ	$n \times 1$ vector of censored statu.
$\hat{\beta}$	$m \times 1$ regression coefficient vector
$S(\cdot), G(\cdot), h(\cdot)$	survival, censoring and hazard functions
$F_{surv}(\cdot), G_{cens}(\cdot)$	cumulative survival and censoring functions
P	Positive semi-definite feature regularizer matrix
E	incidence graph on feature set

the patients. One can observe that instances with patient ID 122 and 21 are not censored and hence δ is set to 1 with the survival time equivalent to the time to event of interest (T). Instances with patient ID 61 and 45 are censored with δ set to 0. In this manner, right censoring is applied on the instances in the dataset.

Table 3.2: An example to demonstrate right censoring for 30-day readmission

Patient ID	T	Event	δ	Interpretation
122	2	HF Readmission	1	Patient readmitted after 2 days
61	30	End of Study	0	Patient not readmitted even 30 days after discharge
45	6	Drop from Study	0	Lost follow up of patient 6 days after discharge
21	4	HF Readmission	1	Patient readmitted after 4 days

With a brief description of the survival regression framework, we now introduce some notations that will help in comprehending the Cox regression framework in Table 3.3. Given a dataset X which consists of n data points. Let x_i denote the i^{th} feature vector. Let $T = \{t_1 < t_2 < t_3 < \dots < t_k\}$ represent the set of sorted k unique time-to-event values. δ_i represents the censoring status for the i^{th} patient. $\delta_i=1$ represents the occurrence of an event and $\delta_i=0$ represents a censored instance.

3.2 Preliminaries

The likelihood term in Cox regression can be written as in Eq. (3.1) and the partial log likelihood is defined using Eq. (3.2).

$$l(\beta) = \prod_{i=1}^n \left\{ \frac{\exp(x_i^T \beta)}{\sum_{j \in R_i} \exp(x_j^T \beta)} \right\}^{\delta_i} \quad (3.1)$$

$$L(\beta) = \log(l(\beta)) = \sum_{i=1}^n \delta_i x_i^T \beta - \sum_{i=1}^n \delta_i \log \left(\sum_{j \in R_i} \exp(x_j^T \beta) \right) \quad (3.2)$$

In the partial log likelihood equation, R_i is the set of all patients who are in the risk set of the i^{th} patient. In this Equation, the covariate values for the j^{th} individual is represented using x_j .

$$\frac{\partial L(\beta)}{\partial \beta_j} = \sum_{i=1}^n \delta_i x_{ij} - \sum_{i=1}^n \delta_i \frac{\sum_{l \in R_i} \exp(x_l^T \beta) x_{lj}}{\sum_{l \in R_i} \exp(x_l^T \beta)} \quad (3.3)$$

$$\begin{aligned} \frac{\partial^2 L(\beta)}{\partial \beta_j \partial \beta_k} = & - \sum_{i=1}^n \delta_i \left[\frac{\sum_{l \in R_i} \exp(x_l^T \beta) x_{lj} x_{lk}}{\sum_{l \in R_i} \exp(x_l^T \beta)} \right. \\ & \left. - \frac{\sum_{l \in R_i} (\exp(x_l^T \beta) x_{lj}) \sum_{l \in R_i} (\exp(x_l^T \beta) x_{lk})}{\sum_{l \in R_i} (\exp(x_l^T \beta))^2} \right] \end{aligned} \quad (3.4)$$

We now explain the procedure to convert Cox regression to a modified least squares problem. We define these additional notations to explain our interpretation of Cox as a modified least squares problem. In Algorithm 3.1, M is a triangular matrix obtained after applying the Cholesky factorization on H . M is called the pseudo-design matrix. z is denoted as the pseudo-response vector.

In Algorithm 3.1, the calculation of a pseudo-design matrix (M) and a pseudo-response matrix(z) helps us solve Cox regression problem as a modified least squares problem. This is very helpful considering that there are state-of-the-art least squares solvers which can then be used for solving different variants of Cox regression problems. Similarly, penalized Cox regression problems can also be converted into penalized least squares problems which then

become easier to solve with the existing solvers. However, the Hessian calculation in Cox regression is computationally expensive which can hinder the performance of Algorithm 3.1. It is so because for each of the m^2 elements of the matrix we have to compute the risk sets individually. So we use a trick here to improve the performance of the algorithm by accelerating the Hessian computation. We set the H matrix to be equivalent to the diagonal matrix of the elements of the diagonal of $-\frac{\partial^2 L(\beta)}{\partial \beta_j \partial \beta_k}$.

Finally in Algorithm 3.1, we estimate the values of $\hat{\beta}$ iteratively until convergence is obtained. Once the regression coefficient vector $\hat{\beta}$ is estimated, we can obtain the baseline hazard function using Eq. (3.5). After obtaining the baseline hazard function, we can compute the hazard function $h(t)$ for any given time t . This gives us the hazard or survival probability estimates at any given time t with the trained Cox model.

$$h_0(t) = \sum_{i:t_i \leq t} \frac{\delta_i}{\sum_{j \in R_i} \exp(\hat{\beta}^T x_j)} \quad (3.5)$$

Algorithm 3.1: Cox regression as a modified least squares problem

- 1 **Input:** Time-to-event labels T , Censored survival data X , Censoring Indicator δ , Number of instances n , tolerance parameter tol , Maximum iterations $itermax$.
 - 2 **Output:** Regression coefficient vector $\hat{\beta}$
 - 3 **Initialize** β ;
 - 4 Derive the partial log-likelihood function $L(\beta)$ using Eq. (3.2);
 - 5 **for** $j=1$ to $itermax$ **do**
 - 6 Set $G = -\frac{\partial L(\beta)}{\partial \beta_j}$ using Eq. (3.3);
 - 7 Set $H = -\frac{\partial^2 L(\beta)}{\partial \beta_j \partial \beta_k}$ using Eq. (3.4);
 - 8 Compute $M \leftarrow \text{cholesky}(H)$;
 - 9 Compute $z \leftarrow (M^T)^{-1} \cdot (H\beta - G)$;
 - 10 Solve $\arg \min_{\beta} (z - M\beta)^T (z - M\beta)$;
 - 11 **if** $\|\beta - \hat{\beta}\|_2 < tol$ **then**
 - 12 | break;
 - 13 **end**
 - 14 Set $\beta = \hat{\beta}$;
 - 15 **end**
-

3.3 Cox Regression with Correlation-based Regularization

In this section, we describe the algorithms developed by combining two novel correlation-based regularizers with Cox regression. We integrate both these regularizers following the same paradigm explained in Algorithm 3.1 where we convert Cox regression into its equivalent least squares formulation (LSQ). This conversion helps us solve a regularized Cox regression problem as a regularized least squares problem itself. This is a very *critical part* of all the regularized Cox regression methods we discuss in this chapter including those presented in Section 3.4. We do not explicitly mention this conversion in the discussion below and only discuss the solutions for the regularized least squares variants itself to great detail. Finally, we use these derived regularized least squares solvers in Step 9 of Algorithm 3.1 to obtain the desired regularized Cox regression algorithm.

We now briefly discuss about how the regularizers presented in this section differ from the most commonly used regularizers. Generally, most regularizers considered in the literature are convex loss functions because of their desirable properties. The motivation for applying convex functions in the clinical domain arises from the success achieved by using convex non-smooth functions such as the L_1 and the $L_{2,1}$ norms for different applications [38]. Their properties of sparsity and group sparsity have proven to be very effective for such applications.

Our novel regularizers are functions which use the L_1 , L_2 , and L_∞ norms. Regularizers also need a parameter which governs their importance in the framework. In general, this is denoted as λ and is also called the regularization parameter. In this section, we present the regularizers and their corresponding optimization routines used in the different variants of the modified least squares formulation of Cox regression. This is followed by exploring the feature based penalty and the Octagonal Shrinkage and Clustering Algorithm for Regression (OSCAR) penalty. We now explain how these penalties can be integrated in the modified least squares formulation of Cox regression, and then we present efficient solvers which are later integrated in Algorithm 3.1 to obtain the FEAR-COX and OSCAR-COX algorithms,

respectively.

3.3.1 FEAR-COX Algorithm

In this section, we define the feature-based regularizer for the modified least squares formulation of Cox regression, and we then discuss the cyclic coordinate descent method for solving this optimization problem. This regularizer is defined in the context of the least squares problem as follows. Consider a linear model as given in Eq. (3.6) where $X \in \mathbb{R}^{n \times m}$ is the data matrix, $y \in \mathbb{R}^n$ is the response vector and β is the regression coefficient vector. We can assume that the model is standardized which implies that $1^T y = 0$, $1^T X_i = 0$ and $X_i^T X_i = 1$

$$y = X\beta + \epsilon. \quad (3.6)$$

The general problem being solved is given in Eq. (3.7). In this equation $J(\beta)$ is a non-negative valued penalty function. λ is a non-negative complexity and regularization parameter. For the least squares formulation $J(\beta) = 0$. For ridge regression $J(\beta) = \|\beta\|_2^2$ and for the lasso $J(\beta) = \|\beta\|_1$. The overall minimization formulation will be as follows

$$\hat{\beta} = \arg \min_{\beta} \|y - X\beta\|^2 + \lambda J(\beta) \quad (3.7)$$

We propose a feature-based convex regularizer and plug it into the regression framework. The regularizer is used to incorporate the pairwise feature similarity into the regression framework. Let $P \in \mathbb{R}^{m \times m}$ be a positive semi-definite matrix.

$$J(\beta) = |\beta|^T P |\beta| \quad (3.8)$$

$$\hat{\beta} = \arg \min_{\beta} \|y - X\beta\|^2 + \lambda |\beta|^T P |\beta|$$

$J(\beta)$ is defined as in Eq. (3.8). This is followed by defining the formulation of the feature-

based regularizer in Eq. (3.8).

$$\begin{aligned}
L(\beta) &= \|y - X\beta\|^2 + \lambda |\beta|^T P |\beta| & (3.9) \\
&= y^T y - 2q^T \beta + \beta^T Q \beta + \lambda \sum_{i,j} P_{ij} |\beta_i \beta_j| \\
\frac{\partial L}{\partial \beta_i} &= -2q_i + 2Q_i^T \beta + 2\lambda \operatorname{sgn}(\beta_i) \sum_{j=1}^m P_{ij} |\beta_i \beta_j|
\end{aligned}$$

$$(Q_{ii} + P_{ii})\beta_i + \operatorname{sgn}(\beta_i)\lambda \sum_{j \neq i} P_{ij} |\beta_j| = q_i - \sum_{j \neq i} Q_{ij} \beta_j$$

In Eq. (3.9), we provide the cyclic coordinate descent steps used in solving for β . We set $Q=X^T X$ and $q=X^T y$. This is followed by setting the derivative to zero and solving for the i^{th} coordinate of β keeping the remaining $(i-1)$ components constant. In this formulation, S is the soft-thresholding function and is defined as $S_\lambda(x)=\operatorname{sgn}(x)\max(|x| - \lambda, 0)$.

To implement the FEAR-COX algorithm, we follow the steps outlined in Algorithm 3.1 and replacing the Equation in Step 9 of Algorithm 3.1 by the FEAR-COX solver procedure provided in Algorithm 3.2. This replacement in Algorithm 3.1 makes use of a more efficient regularizer in the form of the feature-based formulation to learn the corresponding regression coefficient vector.

3.3.2 OSCAR-COX Algorithm

Structured sparsity (grouped correlation) in EHRs as illustrated in Figure 3.1 is a phenomenon which is difficult to capture using regular sparsity inducing norms such as the LASSO and elastic net. In practical applications, one often knows a structure on the coefficient vector in addition to sparsity. For example, in group sparsity, one assumes that variables in the same group tend to be zero or nonzero simultaneously. If meaningful structures exist, we show that one can take advantage of such structures to improve the standard sparse learning based Cox regression methods. In this algorithm, we incorporate the OSCAR (Octagonal Shrinkage and Clustering Algorithm for Regression) regularization [37] into the Cox regression framework.

Algorithm 3.2: Solver for the FEAR-COX Algorithm.

```

1 Input: Feature Vector  $X$ , Response variable  $y$ , Regularization parameter  $\lambda$ , PSD
   Matrix  $P$ , Maximum number of iterations  $numiter$ , Tolerance  $tol$ ,
   Initialized coefficient vector  $\beta^0$ 
2 Output: Regression vector  $\beta$ 
3 Initialize  $\beta^0$ ;
4  $Q \leftarrow X^T X, q \leftarrow X^T y, \beta^{old} \leftarrow \beta \leftarrow \beta^0$ ;
5 for  $j=1$  to  $numiter$  do
6   for  $i=1$  to  $m$  do
7      $\beta_i \leftarrow \frac{S(Q_{ii}\beta_i - Q_i^T \beta + q_i, \lambda(P_i^T |\beta| - P_{ii} |\beta_i|))}{Q_{ii} + \lambda P_{ii}}$ ;
8   end
9   if  $\|\beta - \beta^{old}\|_2 < tol$  then
10     $\beta \leftarrow \text{diag}(1 + \frac{P_{ii}}{Q_{ii}})\beta$ ;
11    return;
12  end
13   $\beta^{old} \leftarrow \beta$ ;
14 end

```

OSCAR performs variable selection for regression with many highly correlated predictors. The advantage of using this penalty over other penalties such as the elastic net and LASSO is that this method promotes equality of coefficients which are similarly related to the response. OSCAR obtains the advantages of both individual sparsity due to the L_1 norm and the group sparsity because of the pairwise L_∞ norm. It can select features and form different groups of features. In this way, OSCAR also does supervised clustering of the features. In this chapter, we use the modified Graph OSCAR (GOSCAR) regularizer [37, 39, 40] in the Cox regression formulation. The formulation of the GOSCAR penalty is given in Eq. (3.10). In this formulation, E is the incidence matrix of the feature graph and $L(\beta)$ is the loss function which is the modified least squares loss function derived from the partial log likelihood of Cox regression and λ_1 and λ_2 are the regularization parameters.

In this manner, a pairwise feature regularizer is added to the Cox regression formulation. OSCAR has proven to be more effective than the elastic net in handling correlation among variables and hence is more suited for EHR data as illustrated in Figure 3.1. For the sake of simplicity, we refer to the GOSCAR-COX algorithm as OSCAR-COX throughout this

chapter.

$$\hat{\beta} = \arg \min_{\beta} L(\beta) + \lambda_1(\|\beta\|_1) + \lambda_2(\|E\beta\|_1) \quad (3.10)$$

In contrast to the FEAR-COX algorithm, the formulation in OSCAR-COX is non-smooth. This problem can be solved using the alternate direction method of multipliers (ADMM) [41] method effectively. The ADMM method has proven to have a very fast convergence rate and is particularly useful for our problem. We now explain the OSCAR regression algorithm for regularized linear regression which can be used for solving the modified least squares formulation of OSCAR-COX. This solution will use the ADMM formulation for fast and efficient convergence. We now explain the formulation for the alternate direction method of multipliers (ADMM).

The ADMM routine is used to solve problems of the form in Eq. (3.11). The variables $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$ where $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$ and $c \in \mathbb{R}^p$. f and g are assumed to be convex functions.

$$\begin{aligned} \arg \min_{x,z} f(x) + g(z) \\ \text{s.t. } Ax + Bz = c \end{aligned} \quad (3.11)$$

ADMM method uses a variant of the augmented lagrangian method and reformulates the problem as given in Eq. (3.12). The update rule steps which are iteratively processed in this method are given in Eq. (3.13). In Eq. (3.14), we provide a basic formulation of the OSCAR penalty in the linear regression setting explained above.

$$\begin{aligned} L_{\rho}(x, z, \mu) = f(x) + g(z) + \mu^T(Ax + Bz - c) \\ + \frac{\rho}{2} \|Ax + Bz - c\|^2 \end{aligned} \quad (3.12)$$

The update rule for ADMM is given by

$$\begin{aligned}
x^{k+1} &:= \arg \min_x L_\rho(x, z^k, \mu^k) \\
z^{k+1} &:= \arg \min_z L_\rho(x^{k+1}, z, \mu^k) \\
\mu^{k+1} &:= \mu^k + \rho(Ax^{k+1} + Bz^{k+1} - c)
\end{aligned} \tag{3.13}$$

The GOSCAR regression algorithm uses the OSCAR penalty with a least squares loss function. This is a modified form of the OSCAR penalty as given in Eq. (3.14) with the addition of the incidence matrix E for the graph. We now explain the steps needed to solve this modified OSCAR regression problem using the ADMM procedure. The ADMM formulation for the GOSCAR regression algorithm is given in Eq. (3.15).

$$\arg \min_{\beta} \|y - X\beta\|^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \sum_{i < j} \max\{|\beta_i|, |\beta_j|\} \tag{3.14}$$

$$\begin{aligned}
&\arg \min_{\beta, q, p} \frac{1}{2} \|y - X\beta\|^2 + \lambda_1 \|q\|_1 + \lambda_2 \|p\|_1 \\
&\quad s.t \quad \beta - q = 0, E\beta - p = 0
\end{aligned} \tag{3.15}$$

$$\begin{aligned}
L_\rho(\beta, q, p, \mu, v) &= \frac{1}{2} \|y - X\beta\|^2 \\
&\quad + \lambda_1 \|q\|_1 + \lambda_2 \|p\|_1 + \mu^T(\beta - q) \\
&\quad + v^T(E\beta - p) + \frac{\rho}{2} \|\beta - q\|^2 + \frac{\rho}{2} \|E\beta - p\|^2
\end{aligned} \tag{3.16}$$

$$\begin{aligned}
\beta^{k+1} &= \arg \min_{\beta} \frac{1}{2} \|y - X\beta\|^2 + (\mu^k + E^T v^k)^T \beta \\
&\quad + \frac{\rho}{2} \|\beta - q^k\|^2 + \frac{\rho}{2} \|E\beta - p^k\|^2
\end{aligned} \tag{3.17}$$

Algorithm 3.3: OSCAR Solver for the modified least squares formulation of Cox Regression.

1 **Input:** Feature Vector X , Response y , Incidence Graph E , Regularization parameters λ_1, λ_2 , Auxiliary parameter ρ , Maximum number of iterations $itermax$

2 **Output:** Regression vector β^k

3 **Initialize** $p^0 \leftarrow 0, q^0 \leftarrow 0, \mu^0 \leftarrow 0, v \leftarrow 0$;

4 **for** $k=1$ to $itermax$ **do**

5 Compute β^{k+1} using Eq. (3.17);

6 Compute q^{k+1} using Eq. (3.18);

7 Compute p^{k+1} using Eq. (3.18);

8 Compute μ^{k+1}, v^{k+1} using Eq. (3.18);

9 $k \leftarrow k + 1$;

10 Continue until *convergence*;

11 **end**

$$q^{k+1} = \arg \min_q \frac{\rho}{2} \|q - \beta^{k+1}\|^2 + \lambda_1 \|q\|_1 - (\mu^k)^T q \quad (3.18)$$

$$= S_{\lambda_1/\rho}(\beta^{k+1} + \frac{1}{\rho}\mu^k)$$

$$p^{k+1} = S_{\lambda_2/\rho}(E\beta^{k+1} + \frac{1}{\rho}v^k) \quad (3.19)$$

$$\mu^{k+1} = \mu^k + \rho(\beta^{k+1} - q^{k+1})$$

$$v^{k+1} = v^k + \rho(E\beta^{k+1} - p^{k+1})$$

In this formulation, q and p are the slack variables. E is the incidence matrix of the graph. In Eq. (3.16), we obtain the augmented lagrangian function in terms of the variables and lagrange multipliers μ and v . In this equation, ρ is the scalar augmented lagrangian parameter which is derived using cross validation.

In Eq. (3.18), S_λ is the soft thresholding function. $S_\lambda(x) = \text{sgn}(x) \max(|x| - \lambda, 0)$. To implement the OSCAR-COX algorithm, we follow the steps outlined in Algorithm 3.1 and replace the Equation in Step 9 of Algorithm 3.1 by the OSCAR solver for the least squares formulation of Cox provided in Algorithm 3.3.

The essence of both FEAR-COX and OSCAR-COX is the same that they are solving a regularized least squares problem derived from Cox regression. However, the difference arises in the usage of novel regularizers and different optimization methods for solving the regularized least squares problem. In terms of the regularization, the difference between the FEAR and OSCAR regularizers lies in their uniqueness in handling correlated variables. FEAR uses a feature-based regularizer in its formulation to handle correlated variables effectively. In this algorithm, the choice of P is important, but we do not study the performance with different formulations of P which is intended for future work. It uses the L_2 norm based formulation in its regularizer.

In OSCAR, we use the L_1 norm and a pairwise L_∞ norm term. The pairwise L_∞ function encourages similar coefficient values for correlated variables. OSCAR is also effective at handling structured sparsity which cannot be inherently detected using the elastic net regularizers. This discussion helps us understand that both these algorithms handle correlated variables in their own unique ways.

We now discuss the complexity of these algorithms. FEAR-COX uses the cyclic coordinate descent approach with a convex smooth composite loss function which is theoretically known to converge from the literature of coordinate descent. The time complexity of one iteration of FEAR-COX is $O(m)$ which is for the soft-thresholding operation.

OSCAR-COX uses the ADMM method coupled with a convex loss function. The theory of augmented lagrangian based multipliers can be used easily here to prove the convergence of this approach using the ADMM steps provided earlier in this section. In OSCAR-COX, the Cholesky factorization only needs to be computed once, and each iteration involves solving one linear system and two soft-thresholding operations. The time complexity of the soft-thresholding operation is $O(m)$. Due to the sparsity of computing the incidence matrix E , its time complexity is $O(me)$, where e is the number of edges in the feature graph. Thus the time complexity for one iteration of OSCAR-COX is $O(m(m + n) + me)$.

3.4 Experimental Results

In this section, we discuss the experimental results obtained by using the proposed FEAR-COX and OSCAR-COX regression algorithms on 9 real-world EHRs. We evaluate the goodness of these algorithms in terms of non-redundancy in feature selection, discriminative ability measured using the survival AUC (concordance index) and Brier score metrics, respectively. We compare the performance of our proposed regularizers against state-of-the-art regularizers such as adaptive-lasso (ALASSO) [20, 42], laplacian net (LAPNET) [43] and fused-lasso (FLASSO) [44]. We also provide brief implementation details for these algorithms and the parameter settings are also explained. Our feature selection analysis compares the goodness of the features selected using our methods and compares them to those obtained from other prominent feature selection methods for censored data. We also plot the sparse important variables included in the models and conduct a study on the biomarkers obtained by using the proposed algorithms and validate those biomarkers using survey articles from existing clinical literature.

3.4.1 Experimental Setup

In this section, we provide the description of the components of the EHRs used in this chapter followed by briefly explaining the implementation details for our proposed regularized Cox regression algorithms. We will describe various kinds of variables present in our data acquired from Henry Ford Health System (HFHS) Detroit, Michigan USA. We also present a flowchart diagram which represents how these variables are collected from a patient at HFHS in Figure 3.2. The patient readmission cycle consists of the different stages a patient goes through from the initial admission to the next readmission [45, 46]. The different kinds of information obtained from the patient beginning from the admission to discharge includes demographics, comorbidities, medications, procedures and pharmacy claims. All these constitute an EHR for that particular hospitalization of the patient. The entire set of important variables which constitute an EHR are classified in the literature under the following broad categories.

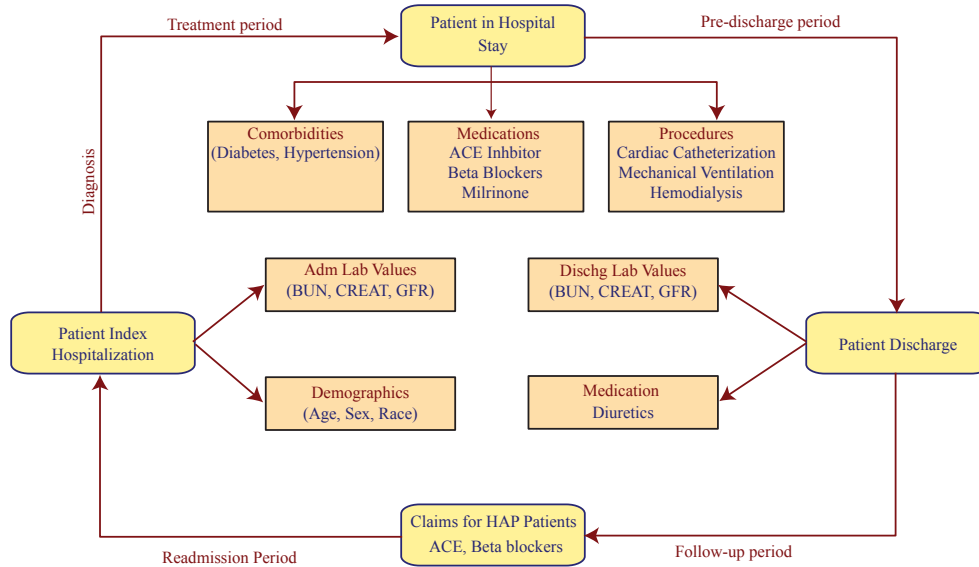


Figure 3.2: Patient readmission cycle at a hospital.

- *Socio-demographic Variables*: These variables in this category include age, sex, race, marital status, health insurance, and income. This also consists of follow-up information on the patients after being discharged from the hospital.
- *Comorbid Conditions*: These variables are considered to be one of the most important factors for determining the readmission risk. The most commonly used conditions under this category are represented using binary variables for different conditions associated with heart failure such as diabetes, hypertension, atrial fibrillation, myocardial infarction, and chronic lung disease.
- *Serum Biomarkers*: These variables include certain laboratory variables which are associated with the readmission risk. Some of the important variables in this category include BUN, Creatinine, serum sodium (NA) and hematocrit or haemoglobin (HGB). We extracted unique labs from EHRs for our current analysis which includes most of the lab variables that are important candidates for being associated with readmission risk.
- *Medications and Procedures*: The variables under this category include medications such as Beta-blockers, ACE (angiotensin-converting-enzyme) inhibitors, and ARB (an-

giotensin receptor blockers). The procedures that are important include cardiac catheterization, hemodialysis and mechanical ventilation.

We computed features which signify the % of *abnormal* labs for a patient (ALR). We construct this feature by using the measured lab values for the patient throughout and comparing it with the lower and upper bound values present in the *Labs*. The average abnormality score computed over all the labs for a patient helps in understanding the aberration present in the labs for a patient. The idea behind constructing this feature is based on domain knowledge from the literature which indicates that >25% decrease in GFR and >25% increase in BUN values was associated with worse survival rate and higher readmission [47].

For the procedures, we created variables for each distinct procedure conducted for the patient. This feature represents the number of times these individual procedures were conducted for the patient. In the medications, we follow the same protocol as done for the procedures and we created two new variables for the distinct medications. In summary, it can be seen that following this procedure summarizes the complex clinical data into a succinct representation which is then used for readmission risk prediction [48].

In Table 3.3, we provide the details about the number of records in the EHRs. The variation in the number of columns for these EHRs arises from the difference in the number of common lab tests, procedures and medications administered to the patients during their different readmissions. In this longitudinal data, we observe the phenomenon that as the readmission index increases the number of patients readmitted decreases.

The FEAR-COX and OSCAR-COX algorithms were implemented in the R programming environment. We convert the original Cox regression problem into its modified least squares formulation. We then implemented the cyclic coordinate descent and ADMM procedures for FEAR-COX and OSCAR-COX. We also used the *igraph* R package for constructing the incidence matrix to be used in OSCAR and for computing the graph laplacian for laplacian net cox algorithm. In addition, we also implemented the FLASSO-COX using the *genlasso* R package [44]. We used the *coxNet* R package for the EN-COX algorithm. We use the

Table 3.3: Description of EHRs used in our experiments.

EHRs	# Features	# Instances
EHR-0	77	4416
EHR-1	76	3409
EHR-2	76	2748
EHR-3	76	2208
EHR-4	75	1800
EHR-5	75	1463
EHR-6	77	1248
EHR-7	75	1055
EHR-8	75	855

sbrier function from the *ipred* R package to compute the Brier score [49]. Feature selection using supervised principal components analysis is done using the *superpc* R package.

In the OSCAR-COX algorithm, the augmented lagrangian parameter ρ was set to 3 and we use the same values for both the regularization parameters λ_1 and λ_2 which was set to 2. These values were determined in a greedy manner by choosing the values that gave us the best performance. In the FEAR-COX algorithm, the elastic net parameter α was varied from 0.1 to 0.7 with increments of 0.05. We observed that an α value of 0.6 gave us the best performance.

3.4.2 Evaluation Metrics

In this section, we explain the evaluation metrics used for our experimental results. Popular metrics used in survival analysis, such as time-based AUC and survival AUC [49] aim at evaluating the relative risk of an event for two instances, than predicting the absolute survival times for these instances. These metrics are introduced below.

$$\begin{aligned} \text{AUC}(T_c) &= P(\hat{Y}_i < \hat{Y}_j | Y_i < T_c, Y_j > T_c) \\ &= \frac{1}{\text{num}(T_c)} \sum_{Y_i < T_c} \sum_{Y_j > T_c} \mathbf{I}(\hat{Y}_i < \hat{Y}_j) \end{aligned} \quad (3.20)$$

In Eq. (3.20), we define the time-based AUC estimated at any given time T_c . $\text{num}(T_c)$ denotes the number of comparable pairs at time T_c and \mathbf{I} is an indicator function. $\text{AUC}(T_c)$

can be used to define the Survival AUC metric which measures the weighted average of the time-based AUC as given in Eq. (3.21). In this equation, T_e represents the set of all possible event times in the dataset, and num represents the cumulative number of comparable pairs calculated over all event times.

$$\text{Survival AUC} = \frac{1}{num} \sum_{T_c \in T_e} AUC(T_c) \cdot num(T_c) \quad (3.21)$$

We also evaluate our model by computing the brier score [49] at any given time T_c using Eq. (3.22). This corresponds to the squared difference between the event indicator variable for instance i (δ_i) and its corresponding survival prediction \hat{Y}_i . This is averaged over all instances to obtain $BS(T_c)$ and integrated over (T_e) to obtain the integrated brier score (IBS).

$$BS(T_c) = \frac{1}{n} \sum_{i=1}^n (\delta_i - \hat{Y}_i)^2 \quad (3.22)$$

$$IBS = \frac{1}{max(T_e)} \int_0^{max(T_e)} BS(T_c) dT_c$$

The values of the integrated brier score and survival AUC range between 0 and 1. It should be noted that a good survival regression model will have high survival AUC and low brier score. In Eq. (3.23), we provide the formula for the redundancy metric. In this Equation, ρ_{ij} is the Pearson correlation coefficient, F is the set of features selected by the corresponding parsimonious model and m is the number of features present in the dataset.

$$\text{Redundancy} = \frac{1}{m(m-1)} \sum_{f_i, f_j \in F, i > j} \rho_{ij} \quad (3.23)$$

3.4.3 Redundancy in Features

In the proposed regularized Cox regression algorithms, we use sparsity inducing norms with specific mathematical structure to handle correlation among attributes. Due to the sparsity induced, these methods also perform feature selection implicitly. We compare the goodness of the features selected by these methods against state-of-the-art feature selection

methods. The metric we use for comparison is the redundancy of features given in Eq. (3.23). In Table 3.4, we compute the redundancy scores using several regularized Cox regression algorithms and supervised principal components (SuperPC) [7].

Supervised principal components is a generalization of principal components regression. The principal components are the linear combinations of the features that capture the directions of largest variation in a dataset. To find linear combinations that are related to an outcome variable, we retain only those features whose score exceeds a threshold. A principal components based analysis is carried out using only the data from these selected features.

The redundancy values in Table 3.4 indicate that FEAR-COX is unanimously providing the best set of top ranked features for all the datasets. These features are non-redundant in terms of representation and also effective for prediction. The survival AUC values in Table 3.5, indicate that OSCAR-COX has higher discriminative ability compared to other methods as it can infer structured sparsity effectively.

Table 3.4: Redundancy of features selected by FEAR-COX and OSCAR-COX against feature selection algorithms.

Method	EHRs								
	0	1	2	3	4	5	6	7	8
SuperPC	0.0134	0.0147	0.0137	0.0139	0.0129	0.0117	0.011	0.0149	0.0136
EN-COX	0.04832	0.0453	0.0458	0.0491	0.0458	0.0455	0.0420	0.04793	0.0466
LASSO-COX	0.0478	0.0466	0.0453	0.0486	0.0454	0.0451	0.0414	0.0472	0.0464
ALASSO-COX	0.0171	0.0156	0.018	0.0215	0.0186	0.0077	0.009	0.0157	0.0138
LAPNET-COX	0.0461	0.0471	0.0458	0.0491	0.0458	0.0455	0.042	0.0479	0.0466
FLASSO-COX	0.0483	0.0471	0.0458	0.0491	0.0458	0.0455	0.042	0.0479	0.0466
FEAR-COX	0.0053	0.005	0.0078	0.0074	0.007	0.0053	0.0056	0.005	0.0046
OSCAR-COX	0.0483	0.0471	0.0458	0.0491	0.0458	0.0455	0.042	0.0479	0.046

In Table 3.6, we report the values obtained at time 30 to assess the goodness of our predictions for the 30-day readmission problem. The values provided in Table 3.6 clearly indicate that FEAR-COX and OSCAR-COX are building models which are giving the best predictions at time 30 compared to other regularized Cox regression algorithms.

Table 3.5: Survival AUC values of FEAR-COX and OSCAR-COX against state-of-the-art algorithms.

Method	EHRs								
	0	1	2	3	4	5	6	7	8
CensNB	0.5611	0.56622	0.567	0.5633	0.574	0.583	0.58490	0.5756	0.5771
EN-COX	0.5957	0.6036	0.600	0.611	0.611	0.6094	0.6049	0.6059	0.6081
LASSO-COX	0.5569	0.5624	0.5531	0.5483	0.5440	0.5641	0.56818	0.5527	0.5271
ALASSO-COX	0.595	0.601	0.596	0.608	0.6076	0.6049	0.5971	0.5998	0.5974
LAPNET-COX	0.5563	0.573	0.600	0.611	0.6110	0.5953	0.6049	0.6059	0.6081
FLASSO-COX	0.5637	0.5838	0.5633	0.5456	0.5809	0.588	0.5761	0.5700	0.5592
FEAR-COX	0.587	0.5949	0.5846	0.5923	0.5944	0.6094	0.588	0.5933	0.5848
OSCAR-COX	0.605	0.613	0.586	0.627	0.6076	0.5900	0.611	0.608	0.62

Table 3.6: Brier score values of FEAR-COX and OSCAR-COX against state-of-the-art algorithms.

Method	EHRs								
	0	1	2	3	4	5	6	7	8
CensNB	0.56	0.54	0.545	0.562	0.5498	0.4879	0.5132	0.4677	0.522
EN-COX	0.4075	0.40626	0.3823	0.3649	0.3547	0.3057	0.3264	0.338	0.325
LASSO-COX	0.393	0.394	0.3695	0.3582	0.3225	0.3204	0.3092	0.3030	0.2876
ALASSO-COX	0.404	0.403	0.379	0.333	0.3474	0.3269	0.3284	0.3272	0.3083
LAPNET-COX	0.4364	0.415	0.3824	0.3649	0.3547	0.3055	0.3263	0.3383	0.3254
FLASSO-COX	0.3944	0.3914	0.366	0.3490	0.3261	0.3238	0.3107	0.3028	0.2824
FEAR-COX	0.3932	0.388	0.3638	0.3519	0.322	0.3181	0.3089	0.3023	0.2821
OSCAR-COX	0.3925	0.3878	0.3648	0.3482	0.322	0.3152	0.3084	0.299	0.2800

3.4.4 Visualizing Sparsity of Models

In this section, we analyze the sparsity of the models obtained using our regularized Cox regression algorithms. We use all the 7 regularized Cox regression algorithms and apply them on the 9 longitudinal EHRs. We then obtain some of the features retained in each model and plot their corresponding regression coefficient values over the longitudinal EHRs. In Figure 3.3, we plot the sparsity of the solutions obtained and compare them with respect to the coefficients obtained using our proposed sparse models.

This experiment demonstrates how each of these regularized Cox regression models are interpreting different novel and important biomarkers in their unique way. One can notice that as *OSCAR-COX* promotes equality of coefficients among the same group of variables

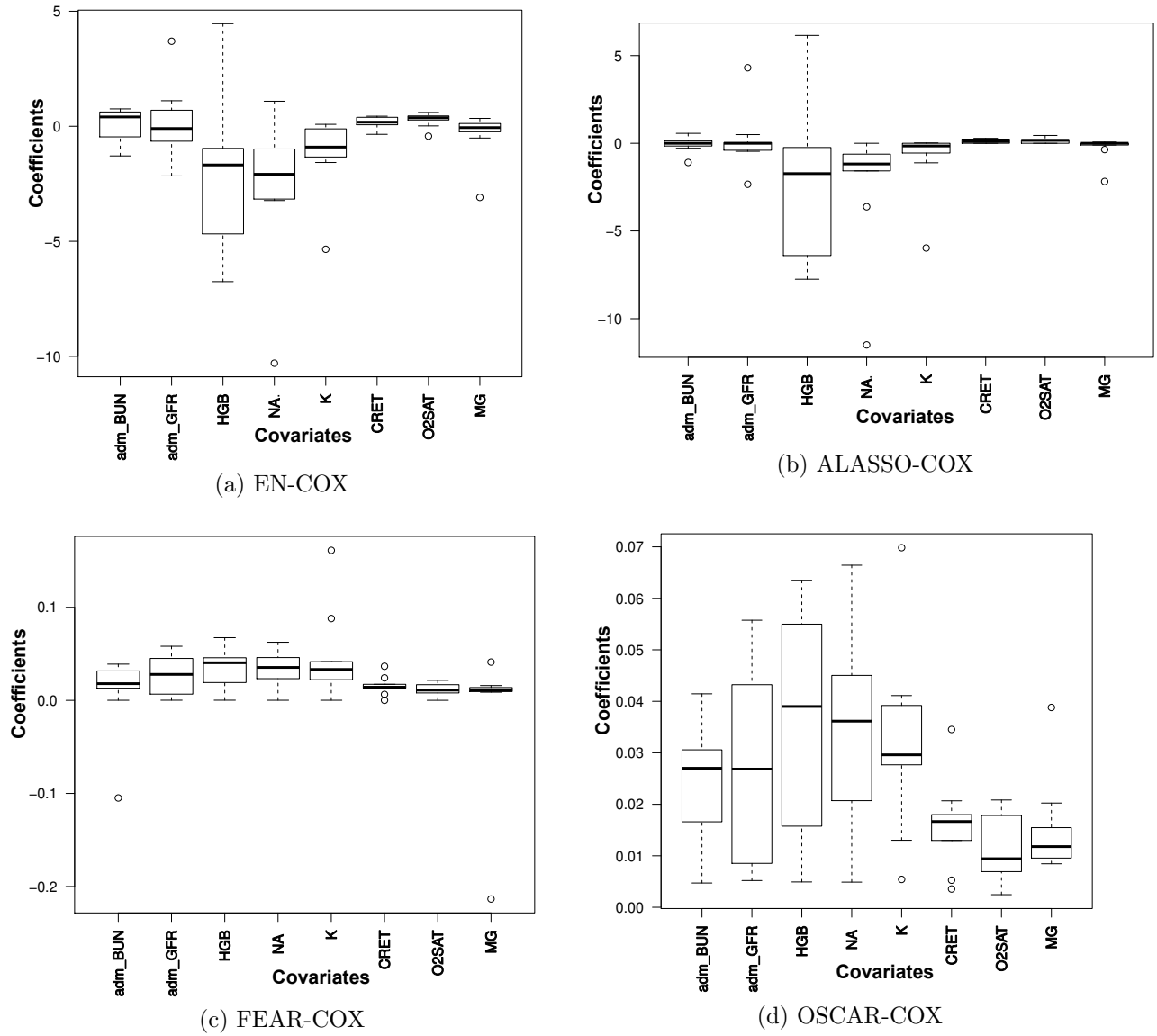


Figure 3.3: Boxplot visualizing the regression coefficients of the sparse variables selected by the regularized Cox regression algorithms.

such as labs in EHRs which explains why there is more similarity among these values in the boxplot. Similarly, one can also notice the ALASSO-COX tends to be more biased towards variables with higher initialized weights during model building which gives *HGB* higher importance. This kind of analysis of these algorithms can help a domain expert decide which algorithm to use as per their clinical requirements.

3.4.5 Scalability Experiments

In this section, we study the scalability of our proposed FEAR-COX and OSCAR-COX algorithms when the number of instances and features in EHRs are varied. We sample EHRs from our cohort and estimate the time needed to determine the final regression coefficient vector for both these algorithms. In Figure 3.4 and Figure 3.5, we provide the scalability plots for FEAR-COX and OSCAR-COX w.r.t. the number of instances and features, respectively. To obtain these plots we sampled different set of instances and features in an increasing order, and we obtained the time needed to build our proposed regularized Cox models. The x-axis represents the selected number of instances and features and the y-axis represents the time taken in seconds. These plots indicate that FEAR-COX is relatively robust with increasing number of instances as its complexity is linear. The time taken for OSCAR-COX follows a similar trend with an increasing number of instances and features which indicates its consistency. However, OSCAR-COX has quadratic runtime complexity, but it tends to build more effective models in terms of performance metrics such as survival AUC and brier score. Hence, there is a trade-off between complexity and performance.

3.4.6 Biomarker Validation

Biomarkers are important indicators (variables) of the progression of a disease in a real-world clinical setting. In this section, we provide a comparative analysis of the biomarkers obtained by applying our methods on the EHRs. We begin by explaining how we created the baseline to evaluate the biomarkers obtained.

Baseline generation: In a popular clinical review article [50], the authors conducted a survey over medical journal articles to determine the important variables for predicting

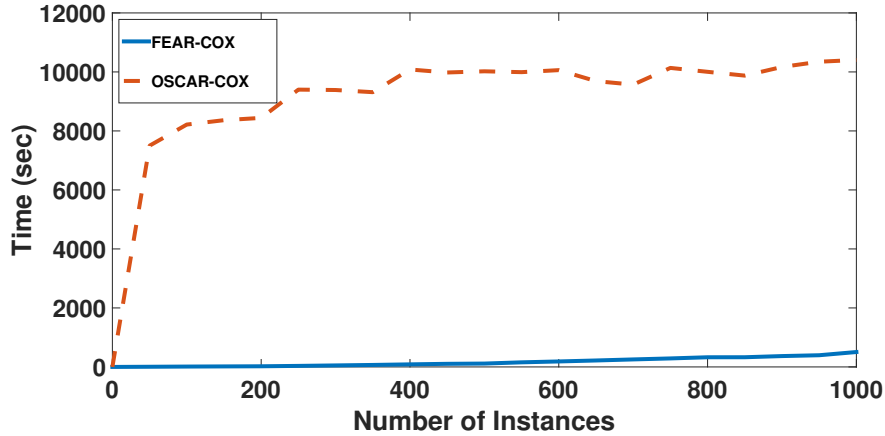


Figure 3.4: Scalability w.r.t the number of instances.

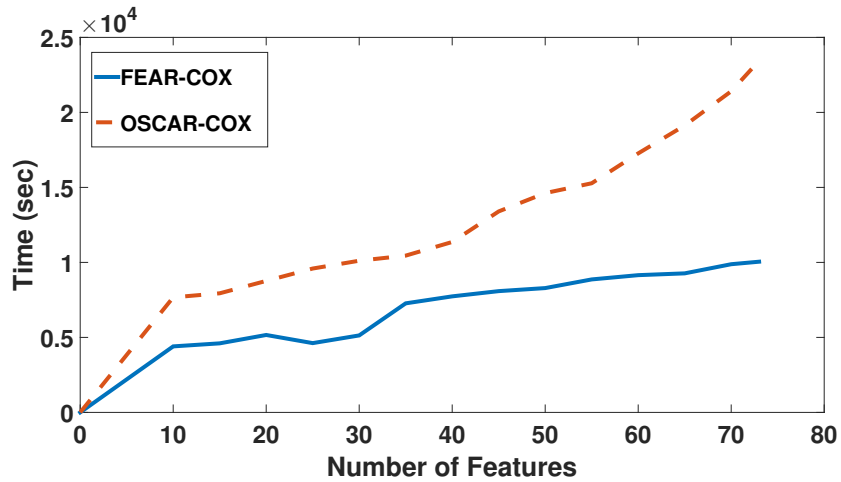


Figure 3.5: Scalability w.r.t the number of features.

readmission risk for heart failure. The survey statistics included capturing the % of studies where the clinical variable was included in the model, % of studies where the variable was included and found to be statistically associated with readmission risk and other related measurements. In Table 3.7, the second column represents the % of clinical studies which reported a statistical association between the candidate variable and heart failure readmission risk. We obtained these numbers directly from the clinical review article [50]. We use this number as the baseline and sort the important biomarkers in the descending order of their statistical association. For each important biomarker, we use a ✓ mark to represent that this variable is selected in the parsimonious feature model and ✗ to mark its absence from

the model. The regression models we consider in this experiment are those of LASSO-COX, EN-COX, FEAR-COX and OSCAR-COX, respectively. We also consider the top 20 variables with highest absolute regression coefficient values from the non-sparse models in this experiment.

We observe that FEAR-COX ranks 6 out of 7 variables as its features in the model, and OSCAR-COX identifies all 7 important baseline biomarkers and uses them in its model. LASSO-COX ranks 5 of these biomarkers in its top ranked feature list. EN-COX identifies only 4 out of the 7 important biomarkers. This proves that our methods identify clinically relevant variables from the entire set and retain those variables in their parsimonious models effectively.

3.4.7 Discussion on Clinical Implications

In this experiment, we consider the important biomarkers which were selected by our regularized Cox regression models and assess their importance from a clinical perspective. Based on our analysis, we received explicit feedback from the clinical expert who informed us that Haemoglobin (HGB), Creatinine (CREAT), Blood urea Nitrogen (BUN) and glomerular filtration rate (GFR) are all well described biomarkers in the heart failure literature [45, 47]. The relationship of Potassium (K) to readmission is complex due to its correlation with renal dysfunction and it is also a biomarker which has not been explored so far. The relationship of magnesium to hospitalization in heart failure patients is also not well described in the medical literature possibly because of its relation with diuretic medications and it deserves further investigation.

Table 3.7: Statistical association between biomarkers and heart failure readmission.

Variable	Assoc	LASSO-COX	EN-COX	FEAR-COX	OSCAR-COX
HGB	0.81	✘	✓	✓	✓
cardiac_cath	0.73	✓	✓	✓	✓
NA	0.71	✓	✓	✓	✓
BUN/CREAT	0.66	✓	✓	✓	✓
heart_failure	0.60	✓	✘	✓	✓
afib	0.60	✘	✘	✓	✓
pvd	0.56	✓	✘	✘	✓

Arterial oxygen saturation (O2SAT) was used in different home monitoring methods to assess its relationship with readmission risk. However, there are no conclusive studies so far which indicate a direct relationship of O2SAT with readmission risk which makes this an interesting biomarker to explore. The discovery of a new biomarker through such studies raises the possibility of a new target of interventions to be tested on patients. The biomarker can then be applied in real-time clinical scenarios where a simple intervention could be aimed at this biomarker to check if the changes in its magnitude reflected the improvement/exacerbation in the patient's health condition.

Transforming into Practice: Intervention Studies- The risk of readmission will be calculated at the time of discharge from the hospital and it can be used to make appropriate intervention decisions. Many of these interventions would be reasonable to try in a high-risk population (low-risk patients get routine standard current care which is brief education at the time of discharge and routine follow-up scheduled with their outpatient physician). Examples of interventions for high-risk patients can include: enrollment in nurse-driven disease management programs, home nursing visits, early discharge follow-up, outpatient intravenous diuretics, laboratory/biomarker monitoring and novel technology-based solutions. The objective here is to target more costly interventions to patients with the highest risk of heart failure readmission [51].

We hypothesize that the patients with an estimated probability of 30-day readmission between 50% and 90% are seen in specialized follow up clinic in 3 days, and they receive weekly home nursing visits through the first 6 weeks post-discharge. When deciding what the cutoff is (ie whether the target patients are those with estimated 30-day readmission risk greater than 50%, 70% or 90%) is a judgment call and could be influenced by what the risk estimates look like across the cohort and the cost/difficulty of the intervention being planned at that particular hospital facility.

CHAPTER 4: REPRESENTATION BASED SURVIVAL REGRESSION

4.1 Motivation

In this chapter, we address another important issue with survival data, which is learning a representation for survival data that can resolve the issue of missing time-to-events for censored instances. In standard survival analysis, these instances are labeled using the duration of the study or last known follow-up time [1, 2]. Such censored instances cannot provide much information to the survival algorithm. This inappropriate labelling can become a significant problem especially when the data has many censored instances ($\geq 40\%$ of overall number of instances). To overcome this problem, we propose to solve the missing time-to-events problem in censored data by developing an approach called *calibrated survival analysis* which can learn an appropriate label value for the censored instances. We now explain our motivation for developing this framework for calibrating time-to-event values for censored instances by explaining the inherent problem associated with censored data, and we also explain the two-dimensional correlation based structure in censored data using an example from the crowdfunding domain.

Censoring in data can be divided into two categories, namely, independent and dependent censoring. Independent censoring is a phenomenon where the covariates and censoring are assumed to be independent [1, 3]. In this assumption alone, traditional estimators such as Kaplan-Meier (KM) remain unbiased yielding true estimates. However, most datasets violate independent censoring and exhibit a phenomenon called *dependent censoring* where the covariates in the data and censoring are correlated with each other. In this scenario, the KM estimator is biased which effects the correctness of several other related survival analysis methods.

To address this issue in this chapter, we present an approach called *calibrated survival analysis* which employs a novel form of censoring called *imputed censoring*. The goal of imputed censoring is to reduce the bias in standard survival estimators, and this is ac-

complished by using a regularized inverse covariance based imputation algorithm. We use covariance-based imputation methods as they are well equipped to capture correlations between censored instances while performing imputation which other methods such as matrix factorization do not capture.

The correlation structure in censored data exhibit a unique phenomenon which can be explained by considering a typical crowdfunding scenario. In this scenario, we define an event of interest as the time taken by a project to reach its pre-defined goal amount and succeed [52]. Considering two projects which got censored one can notice that in order to impute the time-to-event labels for these instances the following factors need to be considered which are (i) time taken by instances similar to both of them to reach the goal amount (inter-event correlation) (ii) importance of similar features for both instances in determining the time-to-event (intra-event correlation). To account for both these phenomena, we develop a row and column-based regularization approach within an inverse covariance estimation procedure to appropriately estimate the time-to-event label. Our proposed *calibrated survival analysis approach* imputes the time-to-event labels for censored instances using a regularized inverse covariance matrix approach.

Another important motivation for proposing a calibrated survival analysis framework for censored data can be obtained from the theory of representation learning [53]. Representation learning attempts to learn a novel representation of the data which captures the inherent structure, so that any predictive algorithm can perform better on the learned new representation. In calibrated survival analysis, through imputed censoring, we are effectively learning a new representation of the original survival data by solving the bias problem explained earlier. We also state that imputed censoring preserves the original censored nature of the problem, and does not output a predictive model directly. Hence, our proposed approach can be used in conjunction with other existing predictive survival analysis methods.

This is considered to be an important advantage of this work since it does not compete

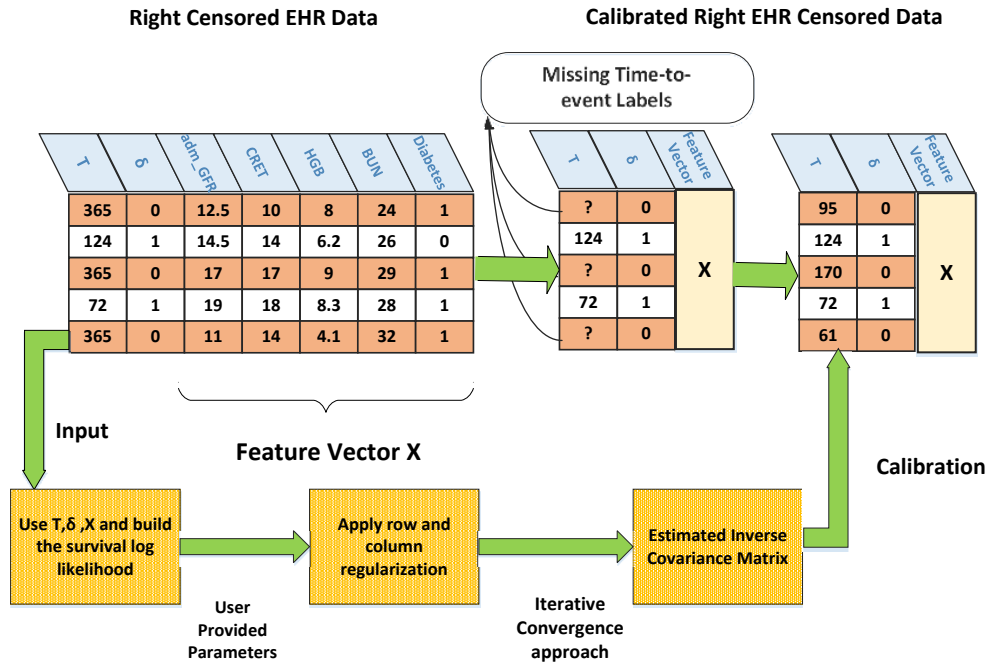


Figure 4.1: Flow diagram for the proposed calibrated survival analysis approach on a EHR dataset.

with existing method to perform better, rather it compliments any existing survival method and improves its performance. We now provide a flow diagram in Figure 4.1 which explains the process of conversion for a sample right censored EHR dataset into a calibrated censored EHR dataset. In Figure 4.1, we consider 5 patients who are being monitored for subsequent readmissions (events). The survival attributes are the time-to-event (T) and the censored status (δ) which are stacked at the front, and they are followed by their corresponding predictive EHR based attributes. One can observe that all the instances in this dataset have been followed up until 365 days past their discharge from their primary hospitalization, which is when we assume the follow-up period ended.

The event of interest here is the rehospitalization of the patient during the follow-up time. We observe that for patients 1, 3 and 5 the event was not observed during the entire follow-up period and their time-to-event labels are presumed to be missing. Most often, while studying such patients in survival analysis, the methods from the literature typically

assign the last known follow-up time (in this case 365) to label them [54].

Our calibrated survival analysis approach fills the gap in the current literature here by estimating the calibrated time-to-event values for these censored instances by exploiting instance-based and feature-based correlations among censored instances in order to effectively impute them. As illustrated in Figure 4.1, this estimation is done through an iterative convergence routine which forms the main part of our imputed censoring method.

Propose a calibrated survival analysis framework which uses a novel imputed censoring approach to model the time-to-event variable. This imputed censoring approach uses a row and column regularization-based inverse covariance estimation algorithm to impute the censored instances. The goal of this approach is to impute the missing time-to-events for the censored instances in order to build a more efficient representation of the survival data which an algorithm can leverage upon. The primary objective of our approach is to improve the representation of survival data to enhance the predictive ability of survival algorithms.

In this chapter, we study the formulation of our row and column based regularized inverse covariance method which is used in imputed censoring exhaustively. We discuss the properties of this algorithm using the L_1 and L_2 regularizers, but the framework can work with any regularizer with a defined L_p norm where ($p \geq 1$). We also evaluate the effectiveness of our calibration method by comparing the survival AUC (concordance index) values obtained using standard survival regression algorithms on the data with and without our time-to-event calibration. We also conduct experiments to assess the convergence and the impact of regularizers and regularization parameters on the performance of our algorithm.

4.2 Preliminaries

In this section, we explain an overview of our proposed method for converting a censored dataset into a calibrated censored dataset. We begin by presenting the table of notations used in this chapter in Table 4.1.

In this section, we present an overview of our method which can convert any given dataset with right censoring into a calibrated right censored dataset. In this approach, we build a

Table 4.1: Notations used in this chapter.

Name	Description
X	$\mathbb{R}^{n \times p}$ data matrix
Σ	$n \times n$ row covariance matrix
Δ	$p \times p$ column covariance matrix
T	time-to-event
v_i	mean of i^{th} row
μ_j	mean of j^{th} column
q_r	row regularizer
ρ_r	row penalty
q_c	column regularizer
ρ_c	column penalty

framework that uses both single and composite regularization by imposing regularizers and user provided penalty parameters on both the rows (single) and rows, columns (composite) of the feature matrix.

Regularization is used here to learn the sparse inverse covariance matrix. The learned covariance matrix captures the correlation structure among censored instances which is subsequently used for imputed censoring. Unlike other censored labeling schemes, the novelty of this framework lies in interpreting the missing time-to-event values using a mean-restricted matrix variate normal distribution which is represented as $N_{n,p}(v, \mu, \Sigma, \Delta)$. This distribution implies that the missing time-to-event labels are modeled with a mean $v_i + \mu_j$ along with variance $\Sigma_{ii}\Delta_{jj}$.

This modeling can be viewed as a random effects model defined as $T_{ij} = v_i + \mu_j + \epsilon_{ij}$ where $\epsilon_{ij} \sim N(0, \Sigma_{ii}\Delta_{jj})$ which has two additive fixed effects depending on the row and column means and a random effect whose variance depends on the product of the corresponding row and column covariances.

The goal of this method is to impute the time-to-event and in this process calibrating it to a more optimal value. This is called the *calibration* step of our method where we impute the time-to-event labels for the right censored instances rather than naively labelling

it with the duration of observation. We note here that censoring is still preserved in this dataset as these values are *calibrated* values and not the true observed values. If we knew the true time-to-event values for the right censored instances, then the dataset would have been independent of censoring which is unlikely in domains with higher number of censored instances.

Through this methodology, we are proposing a new way of handling time-to-event values for right censored instances, which can convert the dataset into a calibrated dataset which makes it more conducive to apply survival algorithms. Before presenting the algorithm, we first review the notation and phrases used throughout this chapter. For the sake of simplicity, we refer to right censoring as censoring. 1_n represents a unit vector of n entries. We use i to denote the row index and j to denote the column index. The observed and missing parts of row i are o_i and m_i , respectively, and o_j and m_j are the analogous parts of column index j . We let m and o denote the complete set of missing and observed elements, respectively. X here represents a $n \times p$ matrix. This includes the features, the censoring indicator δ and the time-to-event variable T .

Further expanding over this notation, x_{i,o_i} denotes the observed components of an uncensored instance i and x_{i,m_i} denote the missing components of a censored instance. This includes both the missing feature values and the missing time-to-event label information for the censored instance. For each observation, we partition the mean and covariance to correspond to the observed parts of observation i and denote them by μ_{o_i} and Δ_{o_i,o_i} , respectively.

4.3 Calibration using the Inverse Covariance Matrix

In this section, we present the two methods for calibrated survival analysis. We begin by explaining the REgularized inverse covariance based Calibration (**REC**) method, and then present the Transposable REgularized covariance based Calibration (**TREC**) method. Before exploring the inner details of these algorithms, we state explicitly that both these approaches are only meant to build a more effective representation of the original survival data. The time values in the calibrated censored dataset are not the predicted values, but are

only estimated by our iterative convergence framework in an effort to facilitate the process of building an efficient representation of the survival data.

4.3.1 REC Algorithm

In this section, we begin by explaining the *REC* method which receives the censored dataset as an input and outputs the calibrated Times, T_{calib} , which are used for learning the final model. This algorithm is designed using an *iterative convergence* style optimization procedure where we initialize the missing time-to-event values and update our estimates iteratively until convergence is observed.

We now present the regularized likelihood equation used in **REC** algorithm in Eq. (4.1) which uses a single column-based regularization term. One can notice that an important difference between this and the EM algorithm term is the regularization component used. Imputation is a part of the E-step of the algorithm in which the conditional expectation of the complete data log-likelihood is taken given the current parameter estimates. The computation in **REC** can be divided into two parts which are: (i) imputation-based calibration and (ii) covariance correction. We outline both these steps in Eq. (4.2) and Eq. (4.3), respectively.

$$\ell_{obs}(\mu, \Delta) = \frac{1}{2} \sum_{i=1}^n [\log |\Delta_{o_i, o_i}^{-1}| - (x_{o_i} - \mu_{o_i})^T \Delta_{o_i, o_i}^{-1} (x_{o_i} - \mu_{o_i})] - \rho_c \|\Delta^{-1}\|_{q_c} \quad (4.1)$$

The first step, imputation-based calibration, is given in Eq. (4.2). This step also involves the covariance -based correction term and the next step is given in Eq. (4.3). The covariance-based correction term is defined so because it is added to the cross products forming the

covariance matrix.

$$\begin{aligned} \hat{x}_{i,j} &= E(x_{i,j} \mid x_{i,o_i}, \mu', \Delta') & (4.2) \\ &= \begin{cases} \mu'_{m_i} + \Delta'_{m_i,o_i} \Delta'^{-1}_{o_i,o_i} (x_{i,o_i} - \mu'_{o_i}), & \text{if } j \in m_i \\ x_{i,j}, & \text{if } j \in o_i \end{cases} \\ c_{i,jj'} &= \begin{cases} \Delta'_{m_i,m_i} - \Delta'_{m_i,o_i} \Delta'^{-1}_{o_i,o_i} \Delta'_{o_i,m_i}, & \text{if } j, j' \in m_i \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

One can notice that in the covariance correction term $c_{i,jj'}$ is only non-zero when both j and j' are missing (censored in our context). The second step of our **REC** algorithm is the maximization step which is given in Eq. (4.4). In this maximization step, $\hat{\Delta}'$ is computed by replacing μ with $\hat{\mu}$ in $\hat{\Delta}'$.

$$E(x_{i,j}x_{i,j'} \mid x_{i,o_i}, \mu', \Delta') = \hat{x}_{i,j}\hat{x}_{i,j'} + c_{i,jj'} \quad (4.3)$$

In Algorithm 4.1, we follow an iterative convergence routine similar to the traditional EM algorithm and introducing a row-based regularization term and the corresponding covariance correction term. We set $q_r=1$ using the L_1 regularizer due to its formulation as the graphical lasso which can be solved using available techniques such as coordinate descent efficiently.

$$\begin{aligned} Q(\theta \mid \theta^k) &= \frac{n}{2} \log |\Delta^{-1}| - \frac{1}{2} \text{tr}(\hat{\Delta}' \Delta^{-1}) - \rho_c \|\Delta^{-1}\|_{q_c} & (4.4) \\ \hat{\Delta}'_{jj'} &= \sum_{i=1}^n [(\hat{x}_{ij} - \mu_j)(\hat{x}_{ij} - \mu_j) + c_{i,jj'}] \end{aligned}$$

This column-based regularization captures one aspect of imputing censored instances by considering the feature importance among different censored instances in determining their corresponding time-to-event labels while imputing them. With this background, we now look at our next algorithm which uses a composite row and column based regularization method which also captures the instance-wise correlation while imputing right censored instances.

Algorithm 4.1: REC Algorithm

- 1 **Input:** Features X , Status δ , Time T
 - 2 **Output:** Calibrated Times T_{calib}
 - 3 **Initialize** Set the missing values as $:\hat{x}_{i,m_i} = \sum_{i \in o_i} x_{ij} / n_i$;
 - 4 Set $\mu^{(0)}, \Delta^{(0)}$ as the empirical mean, covariance;
 - 5 Compute $E(x_{i,j} | x_{i,o_i}, \mu^k, \Delta^k)$ as in Eq. (4.2);
 - 6 Update Estimates: $\hat{\mu}_j$ & $\hat{\Delta}'_{jj}$;
 - 7 Maximize penalized log-likelihood w.r.t. Δ^{-1} to obtain the new estimate $\hat{\Delta}$;
 - 8 Repeat until convergence;
-

4.3.2 TREC Algorithm

In this section, we present the **TREC** algorithm which tries to learn the inverse covariance matrix from censored data by imposing row and column based regularization on the likelihood function. This is called the Transposable REgularized covariance based Calibration (**TREC**) method for censored data. The novelty of this framework lies in interpreting censoring as an imputation problem on the time-to-event variable by modeling its dependence on both row and column-based features [55]. The formulation for the log-likelihood function in **TREC** is given by Eq. (4.5).

$$\begin{aligned} \ell(v, \mu, \Sigma, \Delta) = & \frac{p}{2} \log |\Sigma^{-1}| + \frac{n}{2} \log |\Delta^{-1}| \\ & - \frac{1}{2} \text{Tr}(\Sigma^{-1}(X - v\mathbf{1}_{(p)}^T - \mathbf{1}_{(n)}\mu^T)\Delta^{-1}(X - v\mathbf{1}_{(p)}^T - \mathbf{1}_{(n)}\mu^T)^T) - \rho_r \|\Sigma^{-1}\|_{q_r} - \rho_c \|\Delta^{-1}\|_{q_c} \end{aligned} \quad (4.5)$$

In Eq. (4.5), q_r and q_c are either 1 or 2, which corresponds to either L_1 or L_2 regularizer. We consider these two choices as they are the most popular regularizers used in the literature. Considering the L_1 norm when q_r and q_c are set to 1, it is observed that solution obtained reaches a stationary point, but it is not guaranteed to be the global maximum.

This happens because of the large number of stationary points present on the likelihood surface when using the L_1 penalty. However, maximization with the L_1 penalties can be achieved by applying the graphical lasso algorithm [11]. This coordinate-wise maximization method used in the graphical lasso leads to a simple iterative algorithm, but it does not necessarily converge to a global maximum.

While considering the L_2 penalty problem on the other hand, the problem can be solved by taking the Eigenvalue decomposition and a global maximum can be found. This leads to a global maximum, but the solution does not have a simple iterative form as for the L_1 norm. However, in both the cases, we observe that better initialization of the row and column estimates can result in a faster convergence rate.

The optimal way of beginning such an assignment is by initializing them with their corresponding MLE estimates for faster convergence. In this regard, we now give the proof for the maximum likelihood estimate (MLE) of the mean parameters.

Theorem 1. *The MLE estimate for v and μ are*

$$\begin{aligned}\hat{v} &= \sum_{j=1}^p \frac{(X_{cj} - \hat{\mu}_j)}{p} \\ \hat{\mu} &= \sum_{i=1}^n \frac{(X_{ir} - \hat{v}_i)}{n}\end{aligned}\tag{4.6}$$

Proof. Expanding the trace term of $\ell(v, \mu, \Sigma, \Delta)$ w.r.t. μ and v and then taking partial derivatives, we get

$$\begin{aligned}\frac{\partial \ell}{\partial v} &= 2\Sigma^{-1}v1^T\Delta^{-1} - 2\Sigma^{-1}(X - 1\mu^T)\Delta^{-1} = 0 \\ \Rightarrow \hat{v}1^T &= X - 1\mu^T \\ \Rightarrow \hat{v} &= \frac{1^T(X - 1\mu^T)}{p} = \sum_{j=1}^p \frac{X_{cj} - \mu_j}{p}\end{aligned}$$

□

This proof can be extended in a similar manner to obtain $\hat{\mu}$ as well. With these MLE initial estimates derived, we now propose the **TREC** algorithm with the L_1 and L_2 norms as regularizers. The algorithm uses a strategy similar to block coordinate descent by maximizing on one block of coordinates at a given time, thus saving considerable mathematical and computational time. Conditional maximization (CM) is done with respect to one block of coordinates either Σ^{-1} or Δ^{-1} .

We now put these steps together and present the **TREC** Algorithm 4.2. In this Algorithm, we begin by initializing \hat{v} and $\hat{\mu}$ from the observed uncensored instances using the MLE estimates given in Eq. (4.6). We then use these values to initialize the time-to-event

Algorithm 4.2: TREC Algorithm

- 1 **Input:** Features X , Status δ , Time T , Regularization parameters ρ_r, ρ_c, q_r, q_c
 - 2 **Output:** Calibrated Times T_{calib}
 - 3 **Initialize:** Estimate \hat{v} and $\hat{\mu}$ from observed uncensored instances using Eq. (4.6);
 - 4 **if** T_{ij} *is missing* **then**
 - 5 | Set $T_{ij} = \hat{v}_i + \hat{\mu}_j$;
 - 6 **end**
 - 7 Start with nonsingular estimates $\hat{\Sigma}$ and $\hat{\Delta}$;
 - 8 Initialize matrices G, C, F, D ;
 - 9 Calculate $\hat{X}^T \hat{\Sigma}^{-1} \hat{X} + G(\hat{\Sigma}^{-1})$ as in Eq. (4.9);
 - 10 Update estimates of \hat{v} and $\hat{\mu}$;
 - 11 Maximize Q with respect to Δ^{-1} to obtain $\hat{\Delta}$ using gradient as given in Eq. (4.10, 4.11);
 - 12 Update estimates of \hat{v} and $\hat{\mu}$;
 - 13 Maximize Q with respect to Σ^{-1} to obtain $\hat{\Sigma}$ using gradient as in Eq. (4.10, 4.11);
 - 14 Repeat until convergence;
-

label and begin the computation as given in Eq. (4.9).

After convergence, the final values of \hat{v} and $\hat{\mu}$ are calculated, subsequently the calibrated time-to-event values T_{calib} are computed through our imputation step. Finally, a new survival model is built using X , δ and T_{calib} and a survival algorithm.

We now provide the details of the convergence and complexity of our **TREC** algorithm. The novelty of our framework lies in estimating both the row and column sparse inverse covariance matrices. The complexity associated with each column-wise computation is $O(np)$ and this computation over p columns amounts to a $O(np^2)$ time complexity. The resulting optimization problem is convex with respect to each term and it can be efficiently solved using block coordinate descent methods.

4.3.3 Algorithm Analysis

We now develop the steps involved in the block coordinate descent optimization algorithm mathematically, beginning with the observed data log-likelihood which we seek to maximize. We use this term $x_{o_j, j}^*$ to condense the likelihood equation to express it in a simpler form as

given in Eq. (4.7).

$$x_{o_j, j}^* = \Sigma_{o_j, o_j}^{-1/2} (x_{o_j, j} - v_{o_j}) \quad (4.7)$$

$$\begin{aligned} \ell(v, \mu, \Sigma, \Delta) &= \frac{1}{2} [\sum_{j=1}^p \log |\Sigma_{o_j, o_j}^{-1}| + \sum_{i=1}^n |\Delta_{o_i, o_i}^{-1}|] \\ &\quad - \frac{1}{2} \text{Tr} \left(\sum_{i=1}^n (x_{i, o_i}^* - \mu_{o_i})^T (x_{i, o_i}^* - \mu_{o_i}) \Delta_{o_i, o_i}^{-1} \right) \\ &\quad - \rho_r \|\Sigma^{-1}\|_{q_r} - \rho_c \|\Delta^{-1}\|_{q_c} \end{aligned}$$

We now derive a simple form to express each of our steps. One is expressed with respect to Σ^{-1} and the other with respect to Δ^{-1} as in Eq. (4.9). This is possible because of the structure of the matrix-variate model, specifically the trace term. The model parameters are represented using $\theta = \{v, \mu, \Sigma, \Delta\}$. The E step, denoted by $Q(\theta | \theta', X_o)$, is expressed in Eq. (4.8).

$$\begin{aligned} Q(\theta | \theta', X_o) &= E(\ell(v, \mu, \Sigma, \Delta) | X_o, \theta') \quad (4.8) \\ &\propto E[\text{Tr}(X^T \Sigma^{-1} \Delta^{-1} X) | X_o, \theta'] \\ &\propto \text{Tr}[E(X^T \Sigma^{-1} X | X_o, \theta') \Delta^{-1}] \\ &\propto \text{Tr}[E(X \Delta^{-1} X^T | X_o, \theta') \Sigma^{-1}] \end{aligned}$$

We now provide the proof for our theorem for obtaining the simple forms of the conditional maximization step which will be used in our blockwise algorithm.

Theorem 2. *The E step is proportional to the following form.*

$$\begin{aligned} E[\text{Tr}(X^T \Sigma^{-1} X \Delta^{-1}) | X_o, \theta'] &= \text{Tr}[(\hat{X}^T \Sigma^{-1} \hat{X} + G(\Sigma^{-1})) \Delta^{-1}] \quad (4.9) \\ &= \text{Tr}[(\hat{X} \Delta^{-1} \hat{X}^T + F(\Delta^{-1})) \Sigma^{-1}] \\ &\text{where } \hat{X} = E(X | X_o, \theta') \text{ and} \end{aligned}$$

$$G(\Sigma^{-1}) = \begin{pmatrix} \text{Tr}(C^{(11)} \Sigma^{-1}) & \dots & \text{Tr}(C^{(1p)} \Sigma^{-1}) \\ \vdots & \ddots & \vdots \\ \text{Tr}(C^{(p1)} \Sigma^{-1}) & \dots & \text{Tr}(C^{(pp)} \Sigma^{-1}) \end{pmatrix}$$

$$F(\Delta^{-1}) = \begin{pmatrix} \text{Tr}(D^{(11)}\Delta^{-1}) & \dots & \text{Tr}(D^{(1n)}\Delta^{-1}) \\ \vdots & \ddots & \vdots \\ \text{Tr}(D^{(n1)}\Delta^{-1}) & \dots & \text{Tr}(D^{(nn)}\Delta^{-1}) \end{pmatrix}$$

$$C^{(jj')} = \text{Cov}(X_{cj}, X_{cj'} | X_o, \theta')$$

$$D^{(ii')} = \text{Cov}(X_{ir}, X_{i'r} | X_o, \theta')$$

Proof. We first show that

$$E[\text{Tr}(X^T \Sigma^{-1} X \Delta^{-1}) | X_o, \theta'] = \text{Tr}[(\hat{X}^T \Sigma^{-1} \hat{X} + G(\Sigma^{-1})) \Delta^{-1}]$$

Let $A = X^T \Sigma^{-1} X$, then,

$$E[\text{Tr}(X^T \Sigma^{-1} X \Delta^{-1}) | X_o, \theta'] = \text{Tr}[E(A | X_o, \theta'), \Delta^{-1}]$$

$$\begin{aligned} E(A_{jj'} | X_o, \theta') &= E(X_{cj}^T \Sigma^{-1} X_{cj'} | X_o, \theta') \\ &= E\left[\sum_{k=1}^n \sum_{t=1}^n x_{tj} x_{tk'} \sigma_{tk}^{-1} | X_o, \theta'\right] \\ &= \sum_{k=1}^n \sum_{t=1}^n \hat{x}_{tj} \hat{x}_{tk'} \sigma_{tk}^{-1} + \sum_{k=1}^n \sum_{t=1}^n C_{tk}^{(jj')} \sigma_{tk}^{-1} \\ &= \hat{X}_{cj}^T \Sigma^{-1} \hat{X}_{cj'} + \text{Tr}(C^{(jj')} \Sigma^{-1}) \end{aligned}$$

Thus, $E(A | X_o, \theta') = \hat{X}^T \Sigma^{-1} \hat{X} + G(\Sigma^{-1})$

The proof showing

$$E[\text{Tr}(X^T \Sigma^{-1} X \Delta^{-1}) | X_o, \theta'] = \text{Tr}[(\hat{X}^T \Sigma^{-1} \hat{X} + F(\Delta^{-1})) \Sigma^{-1}]$$

is similar to the calculation above with $B = X \Delta^{-1} X^T$ and

$$E(B_{ii} | X_o, \theta') = \hat{X}_{ir} \Delta^{-1} \hat{X}_{i'r}^T + \text{Tr}(D^{(ii')} \Delta^{-1})$$

□

We now present the gradient equations that are used in **TREC** with the L_1 and L_2 norms

in Eq. (4.10) and Eq. (4.11).

$$\begin{aligned}\frac{\partial Q}{\partial \Delta^{-1}} &= \Delta - \frac{\hat{X}^T \Sigma^{-1} \hat{X} + G(\Sigma^{-1})}{n} - \frac{2\rho_c}{n} \text{sgn}(\Delta^{-1}) \\ \frac{\partial Q}{\partial \Sigma^{-1}} &= \Sigma - \frac{\hat{X} \Delta^{-1} \hat{X}^T + F(\Delta^{-1})}{p} - \frac{2\rho_r}{p} \text{sgn}(\Sigma^{-1})\end{aligned}\quad (4.10)$$

We now use a notation now through the remainder of this chapter to represent the regularizer being used in **TREC**. L_1 -**TREC** represents using the L_1 norm in **TREC**. The same notation can be extended to the L_2 -**TREC** algorithm. The gradient equation in this case are given as follows.

$$\begin{aligned}\frac{\partial Q}{\partial \Delta^{-1}} &= \Delta - \frac{\hat{X}^T \Sigma^{-1} \hat{X} + G(\Sigma^{-1})}{n} - \frac{4\rho_c}{n} \Delta^{-1} \\ \frac{\partial Q}{\partial \Sigma^{-1}} &= \Sigma - \frac{\hat{X} \Delta^{-1} \hat{X}^T + F(\Delta^{-1})}{p} - \frac{4\rho_r}{p} \Sigma^{-1}\end{aligned}\quad (4.11)$$

4.4 Experimental Results

In this section, we present the experimental results obtained using the proposed **REC** and **TREC** methods for calibrated survival analysis on EHRs, Kickstarter and synthetic datasets. In Figure 4.2, we present a bar graph which plots the censored statistics for the kickstarter and EHRs. One can clearly observe that the distribution of right censored instances is higher for the kickstarter data compared to the EHRs, which is an important characteristic of datasets from the crowdfunding domain.

In this section, we will discuss the data collection and pre-processing steps for the EHRs and kickstarter datasets. We conduct several experiments to study the importance of imputing censored instances using our methods. We provide plots which illustrate the improvements obtained in survival regression algorithms after applying our approach. Finally, we also study the effect of both the regularizers and regularization parameters on the runtime performance of our algorithms.

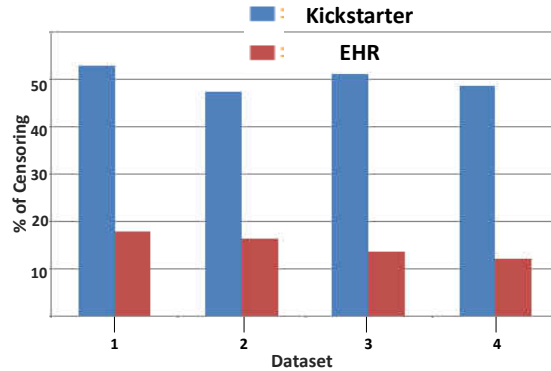


Figure 4.2: Percentage of right censored instances in EHR and Kickstarter datasets.

4.4.1 Datasets Description

We will first describe the various kinds of datasets used in our experiments. These include the Kickstarter data, EHRs and the synthetic datasets. We explain the data collection and pre-processing steps involved with each of these datasets briefly.

Table 4.2: Kickstarter data statistics for 18,143 projects.

Attr	Mean	Min	Max	StdDev
Goal	26,531	100	100,000,000	758,366
Pledged	11,023	100	6,224,955	78,550
backers	138	1	35,383	633
Days	31	1	60	10.5

Table 4.3: Description of censored statistics in the Kickstarter projects.

Name	Startdate	Enddate	# Projects	Censored(%)
Kick1	1/12/2013	1/1/2013	4175	52.99
Kick2	1/1/2014	15/3/2014	5229	47.36
Kick3	16/3/2014	31/4/2014	5720	51.25
Kick4	1/5/2014	30/6/2014	2969	48.58

For the experiments in this chapter, we obtained six months of Kickstarter (a popular crowdfunding platform) data from www.kickspy.com. This dataset spans from 12/15/13 to 06/15/14, which consists of projects characterized by 30 project-based attributes. The attributes in the kickstarter datasets include a number of static features such as project goal amount, duration, textual content, etc., and two dynamic features: per-day increase

Table 4.4: Basic Statistics for EHRs.

Readm	# Rows	# Columns	Censored(%)
EHR0	4417	77	22.20
EHR1	3410	77	17.98
EHR2	2749	77	16.44
EHR3	2209	77	13.63
EHR4	1801	76	12.05

in number of backers and pledged amount as given in Table 4.3. In this manner, a total of 18,143 projects with over 1 million backers were obtained and processed using the procedures followed in [56]. The attribute used to determine the outcome in the kickstarter datasets is the duration of the project.

Each project in our kickstarter database is tracked over a period of time until its goal date is reached or it obtains the goal amount. If a project reaches its goal amount (event in this scenario) in a specified duration (time-to-event) this is measured as a success. However, failure to reach the specified goal amount by the end of the study would imply that the instance has been censored (possibly attains the goal amount at a later time). With this notion of censoring, we present the percentage of censored instances in kickstarter data in Table 4.3.

We also procured electronic health records (EHRs) for patients admitted at the Henry Ford Health System, Detroit, Michigan over a period of 10 years. The event here is heart failure readmission and the duration is measured after the patient has been discharged from primary index hospitalization.

The basic statistics along with the right censored percentages are provided for 5 of our sample datasets in Table 4.4. Readm-index represents the index of readmission for the patients. EHR0 corresponds to the data for the index hospitalization. Similarly, EHR n represents the dataset for the n^{th} rehospitalization for the patient set considered. It should be noted that as n increases the number of patients will be reduced.

In addition, we also generated synthetic datasets by setting the pairwise correlation between any pair of covariates to vary from -0.5 to 0.5. Feature vectors of different dimen-

sionality are generated to construct three synthetic datasets. For each of these synthetic datasets, the generated failure times T are computed using a Weibull distribution.

We compare the effect of calibrated survival analysis on any given dataset before and after applying it, by evaluating the performance using a standard survival learner. In our experiments, for each dataset we create a new one after applying **TREC**-based calibration and this is labelled as **With**, and the version before applying **TREC**-based calibration is labelled as **Without**. We use this notation throughout this section.

4.4.2 Performance Evaluation

We will now describe the evaluation metrics used in this work along with some of the implementation details for both the algorithms proposed in this chapter as well as details pertaining to baseline comparison algorithms. We implemented both **REC** and **TREC** in the R programming language. As mentioned earlier we implemented the versions corresponding to both the L_1 and L_2 norms in **TREC**. The **glasso** R package was used for solving the graphical lasso problem for solving the corresponding subproblems in **REC** and **TREC**. The iterative blockwise gradient descent algorithm was implemented as the main optimization routine for solving **TREC**. The corresponding parameters for regularization in **REC** and **TREC** were determined through five-fold cross validation.

In this section, we will refer to L_2 -**TREC** as **TREC**, and this has been used for obtaining the results in this section. As mentioned earlier, we prefer the L_2 norm as it gives us a global maximum compared to the L_1 norm. So all the calibrated datasets have been generated using the L_2 -**TREC** and **REC** algorithms. We explicitly emphasize that if a regularizer is not mentioned with **TREC**, then it is assumed to be the L_2 -**TREC** algorithm itself. As **REC** uses the L_1 norm alone, we do not specify the norm explicitly, and it assumed that **REC** refers to using the L_1 norm formulation only.

We now briefly discuss the implementation details pertaining to baseline comparison algorithms. The software for CensNB is available at ¹ and we used our code for FEAR-COX

¹<https://sites.google.com/a/umn.edu/jwolfson/software>

and OSCAR-COX². The randomForestSRC and CoxNet R packages are used for running random survival forests and EN-COX, respectively.

We now briefly explain the baseline imputation algorithms used for comparing the performance of **REC** and **TREC**. The first baseline algorithm is SoftImpute [14] which is a method which uses the nuclear norm regularizer and iteratively replaces the missing elements with those obtained from a soft thresholded singular value decomposition (SVD). It tries to minimize the nuclear norm subject to certain constraints. The *softImpute* R package is used for the SoftImpute algorithm.

The other baseline method is Misglasso [13] which is a method that replaces the missing values using the standard graphical lasso by modifying the update step in the EM iteration. We implement the misglasso algorithm by using the graphical lasso R package (glasso).

Table 4.5: Comparison of Survival AUC values for different regularized Cox regression algorithms without and with **TREC** applied on kickstarter, EHR and synthetic censored datasets.

Dataset	EN-COX		FEAR-COX		OSCAR-COX	
	Without	With	Without	With	Without	With
Kick 1	0.81	0.83	0.79	0.80	0.81	0.84
Kick 2	0.81	0.86	0.81	0.84	0.83	0.87
Kick 3	0.80	0.82	0.79	0.833	0.81	0.833
Kick 4	0.77	0.81	0.78	0.81	0.82	0.85
EHR 0	0.58	0.60	0.61	0.605	0.63	0.643
EHR 1	0.592	0.609	0.611	0.62	0.62	0.65
EHR 2	0.598	0.605	0.624	0.611	0.618	0.599
EHR 3	0.59	0.595	0.611	0.63	0.607	0.62
EHR 4	0.618	0.633	0.641	0.655	0.644	0.66
Syn1	0.668	0.673	0.681	0.699	0.677	0.664
Syn2	0.872	0.902	0.890	0.910	0.927	0.943
Syn3	0.727	0.719	0.785	0.854	0.8	0.931

4.4.3 Integrating REC and TREC with Survival Regression

In this section, we present results which demonstrate the robustness of **REC** and **TREC**. The algorithms used in this section are Censored Naive Bayes (CensNB) [26], Boosted Con-

²<https://github.com/MLSurvival/>

Table 4.6: Comparison of Survival AUC values for different survival algorithms without and with **TREC** applied on kickstarter, EHR and synthetic censored datasets.

Dataset	CensNB		RSF		BoostCI		CoxBoost	
	Without	With	Without	With	Without	With	Without	With
Kick 1	0.77	0.80	0.80	0.799	0.78	0.78	0.83	0.86
Kick 2	0.78	0.81	0.81	0.833	0.74	0.733	0.80	0.82
Kick 3	0.76	0.75	0.732	0.758	0.76	0.72	0.79	0.82
Kick 4	0.72	0.77	0.796	0.812	0.72	0.744	0.84	0.83
EHR 0	0.57	0.59	0.599	0.611	0.51	0.55	0.61	0.619
EHR 1	0.575	0.58	0.583	0.609	0.54	0.56	0.62	0.62
EHR 2	0.57	0.60	0.581	0.591	0.575	0.591	0.637	0.665
EHR 3	0.60	0.63	0.611	0.636	0.62	0.62	0.64	0.648
EHR 4	0.621	0.659	0.633	0.627	0.66	0.69	0.66	0.652
Syn1	0.654	0.661	0.633	0.642	0.64	0.67	0.69	0.71
Syn2	0.847	0.867	0.852	0.905	0.83	0.89	0.87	0.933
Syn3	0.714	0.764	0.834	0.841	0.74	0.74	0.85	0.922

cordance Index (BoostCI) [27], CoxBoost, Elastic net Cox (ENCOX) [16], Feature regularized Cox (FEAR-COX), Oscar Cox (OSCAR-COX) [57] and Random Survival Forests (RSF) [28]. The results from Tables 4.5, 4.6 indicate that when **TREC** is applied on the censored dataset (**With**), the survival regression algorithm is able to give a better performance in comparison to using the original right censored dataset (**Without**).

We attribute this better performance to the fact that **TREC** models the censored missing time-to-event values using a row and column regularization method which infers the correlation patterns among censored instances which is needed to impute the time-to-event labels correctly.

The improvements in survival AUC values are prominent with using both regularized Cox regression algorithms as given in Table 4.5, and other survival algorithms as given in Table 4.6. These improvements also confirm that the performance of our approach does not depend on using any specific kind of survival regression algorithm.

4.4.4 Improvement in AUC with Imputed Censoring

In this experiment, we plot the survival AUC values of the learning algorithm when we gradually sample instances from the calibrated data (**With**) using different methods for imputing the missing time-to-event values in survival data. This experiment helps us

interpret how the calibrated samples are contributing towards building a more effective model as they are sampled iteratively. The approaches used for imputation in this experiment include SoftImpute [14], Misglasso [13], **REC** and **TREC**. In this experiment, we present the results for the synthetic datasets, kickstarter datasets and EHRs.

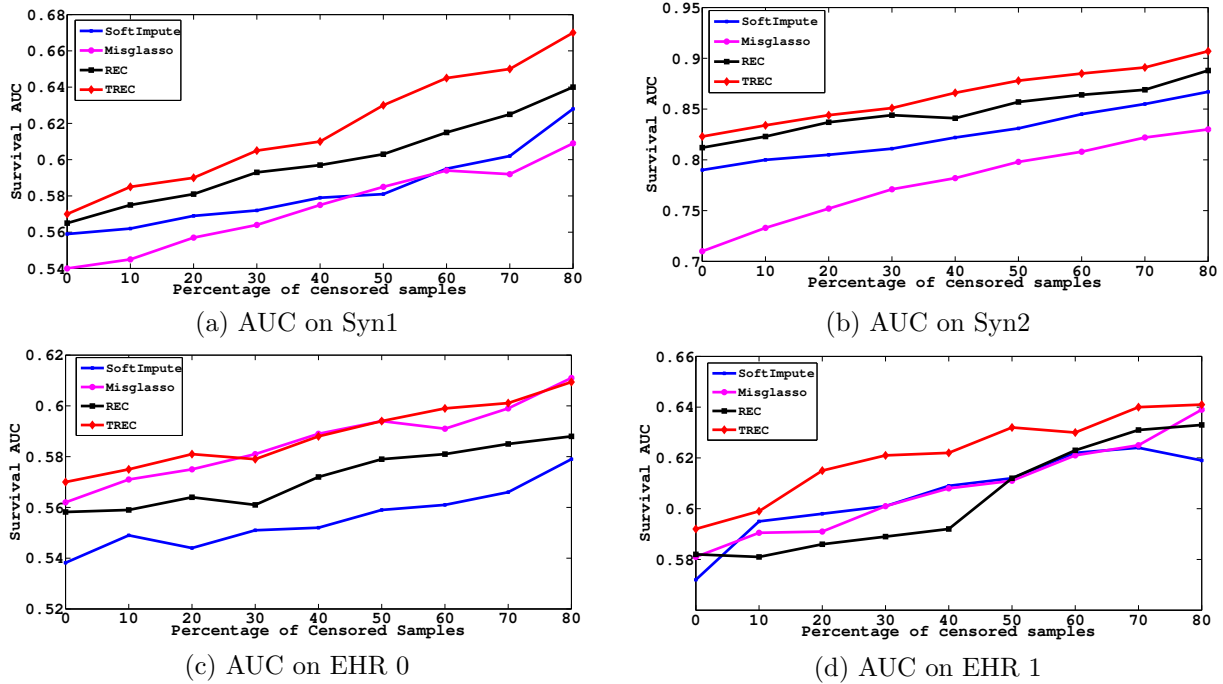


Figure 4.3: Survival AUC plots obtained for calibrated synthetic and EHR datasets using REC, TREC, SoftImpute and Misglasso methods.

The learning algorithm considered for this experiment was the (EN-COX) algorithm. As determined from the previous experiment the choice of the learning algorithm was not a part of our approach, so we can choose any arbitrary survival learner. We train the initial survival model using all the uncensored instances, and we continuously sample instances from a pool of censored instances and add them to retrain a survival model. These censored instances have been imputed using **REC** and **TREC**. Simultaneously, we also impute these instances iteratively using methods such as SoftImpute and Misglasso before training a new survival model.

As imputed censored instances are added to the training data from the censored pool, we retrain the model and plot the survival AUC values on this combined dataset of the initial

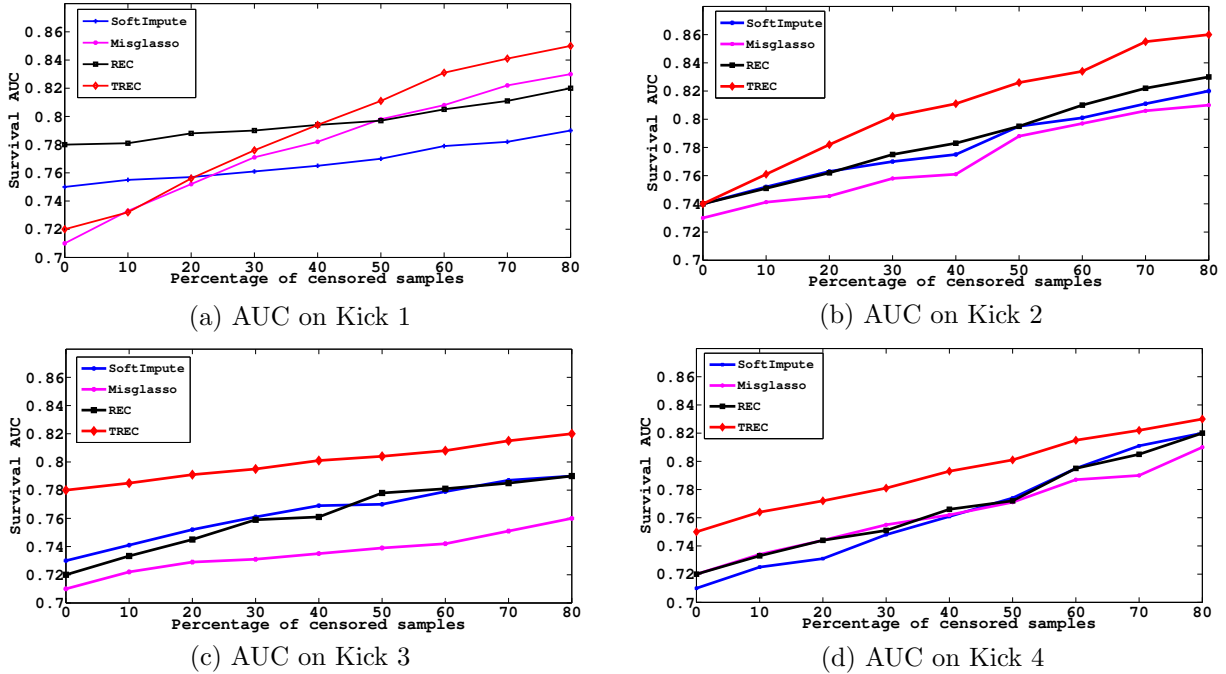


Figure 4.4: Survival AUC plots obtained for calibrated kickstarter datasets using REC, TREC, SoftImpute and Misglasso methods.

set of uncensored instances and the sampled censored instances. From the plots in Figure 4.3 and Figure 4.4, we observe that the survival AUC values improve for most of the cases, with the improvements being prominent for **TREC** compared to other competing methods, and **REC** stands as the second best method.

The better performance of **TREC** is because it is effective in interpreting the missing values in the time-to-event labels for censored instances, as it imputes these values considering the two-dimensional correlation structure within the covariance matrix in its formulation. Calibrated time-to-event labels tend to provide the survival model with more discriminative information which is evident from the improvement in the survival AUC values.

4.4.5 Parameter Sensitivity Analysis

In this section, we study the influence of the row and column regularizers and parameters on the convergence and runtime of the **TREC** algorithm. We study the runtime using both L_1 and L_2 regularizers in **TREC** to assess their time efficiency. We use one of the kickstarter datasets (Kick 1) for this experiment. The values of the row and column regularization

parameters were obtained using cross validation for this dataset.

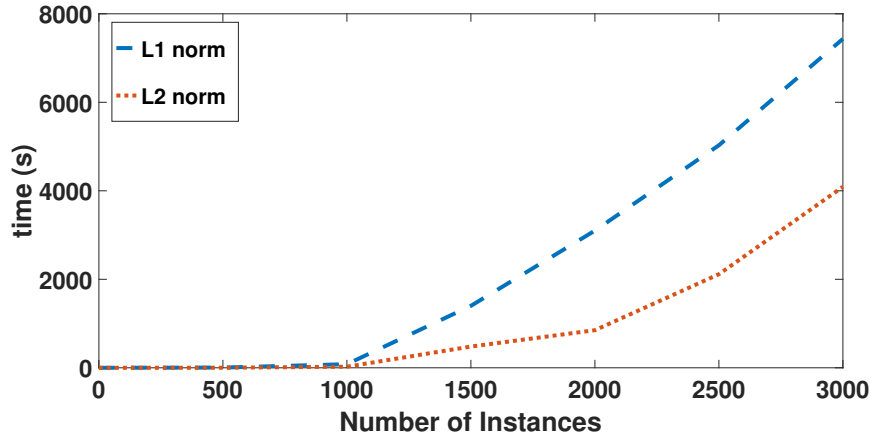


Figure 4.5: Runtime on Kickstarter dataset using L_1 , L_2 norms in **TREC**.

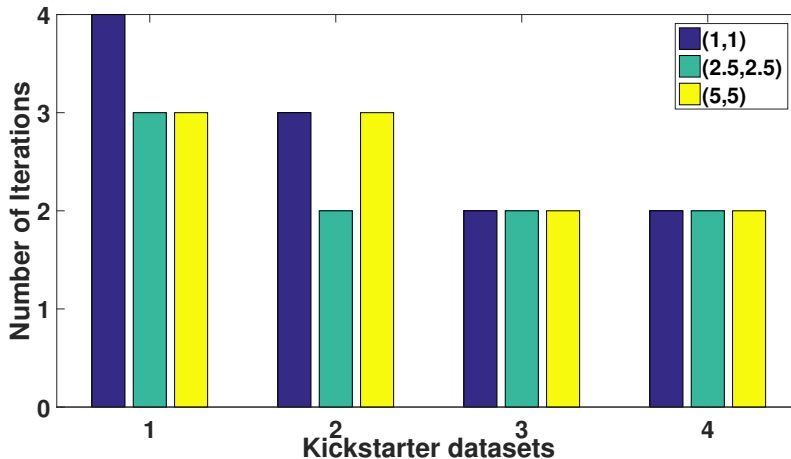


Figure 4.6: Iterations for convergence using L_2 norm based **TREC**.

In Figure 4.5 and Figure 4.6, we plot the runtime in seconds on the Y-axis, and the number of instances sampled from Kick 1 dataset are labeled on the X-axis. We run both our L_1 and L_2 norm based **TREC** algorithms separately to assess their runtime. In Figure 4.5, one can observe that among the two norms L_2 norm seems to be more time efficient compared to the L_1 norm. The L_1 norm uses the graphical lasso solver and the higher number of stationary points observed in this formulation results in higher runtime to obtain convergence. This makes the L_2 norm the more effective regularizer due to better scalability. However, the L_2 norm does not provide sparse solutions with respect to the inverse covariance matrix esti-

mated which affects the interpretability of the solution when dealing with high dimensional datasets. So there is a trade-off between choosing the L_1 and L_2 norms.

In another experiment, we also study the impact of the choice of the regularization parameters on the convergence of **TREC**. In Figure 4.6, the X-axis represents the indices of the 4 kickstarter datasets used in this chapter. The Y-axis represents the number of iterations needed for **TREC** to converge for each dataset using three sets of regularization parameter values. The legend in the figure indicates the values chosen for the regularization parameters ρ_r and ρ_c . We observe that the choice of regularization parameters does not affect the convergence, as there is no uniform pattern observed. So these experiments help us conclude that the choice of regularizer is important, but the value of these regularization parameters does not affect the convergence of **TREC** significantly. This also indicates that our framework is not sensitive to the row and column regularization parameters.

CHAPTER 5: STRUCTURED MODEL FOR RIGHT CENSORED DATA

5.1 Motivation

Predictive models have been built on right censored data using non-parametric, semi-parametric and parametric methods as seen in Chapter 2, 3 and 4. In particular, semi-parametric methods such as Cox regression [6] offer some distinct benefits of interpreting the censoredness from the data, but these methods also assume that the *proportional hazards (PH) assumption* [2] must be satisfied. This assumption states that the risk of occurrence of an event for two unique instances is related by a fixed multiplicative factor called the *baseline hazard rate*. It has been observed empirically that this assumption is not satisfied for all the real-world datasets which limits their applicability significantly.

Other approaches such as linear regression [58, 59] have also been used to learn effective prediction models for right censored data without assuming the PH condition. The advantages of using linear regression models in this domain are (i) it is a non-parametric approach and it does not make any assumptions about the distribution of the data, and (ii) it provides good performance for diverse real-world datasets. Linear regression methods account for right censored instances using weighted schemes and imputation methods [60]. However, weighted methods rely heavily on the convergence of instance weights which is not guaranteed making them biased, and imputation methods use an expectation maximization (EM) framework which adds a significant computational burden affecting the scalability. In contrast to these methods, in this chapter, we propose a Structured regularization based Linear REgression algorithm for right Censored data (SLIREC) which infers the *underlying structure* directly and uses this knowledge to guide the base linear regression model. Our motivation to use structure-based methods arises from the fact that they can effectively infer latent knowledge for prediction such as tree-based hierarchies [61] and graph-based relationships which is extremely crucial for prediction. This is also supported by the success obtained using structured sparsity based regularization methods in regression [62], and to

our knowledge this is the first work which addresses the issue of building structured sparsity based regression models for right censored data.

This structured approach is more robust compared to the standard statistical and Cox-based methods, as it can automatically adapt to different distributions of events and censored instances in the dataset which is very useful when dealing with different real-world datasets. Specific structured regularization methods such as sparse inverse covariance estimation (SICE) [11] can learn a sparse graphical model which explores the dependencies among different events in the right censored data. Our primary goal in this work is to extract such structural knowledge from right censored data using SICE and explore the utility of this knowledge for the linear regression model.

SICE is known to outperform generic regularization methods which do not exploit the structure, as they can efficiently interpret partial correlations among features, which is observed more frequently than absolute correlations. This is one of the primary reasons of their widespread usage in learning sparse graphical models and studying conditional feature independence. The major contributions of this chapter are summarized as follows.

- We propose a Structured regularization based LLinear REgression algorithm for right Censored data (SLIREC) which addresses the problem of building a linear regression model for right censored data by learning the sparse inverse covariance matrix, and uses this structural knowledge in a regularization framework to guide the base linear regression model.
- We formulate SLIREC as a bi-convex optimization problem based on the block-coordinate descent algorithm, and we accelerate the inner computations involved using an efficient approximation scheme based on the proximal-Newton [63] method, which improves the runtime complexity of this algorithm significantly.
- We evaluate the performance of SLIREC by comparing its goodness with respect to regularized Cox regression methods, and also illustrate the improvement obtained through SLIREC by comparing its performance to regression methods which vary in their ability

to infer the structure. In addition, we also present results based on survival regression metrics such as the Brier score and concordance index [49].

This chapter is organized as follows. In Section 5.2, we provide an overview of our approach. In Section 5.3, we present the details of the proposed SLIREC algorithm and discuss the optimization. In Section 5.4, we present the experimental results obtained by applying SLIREC on different benchmark datasets.

5.2 Preliminaries

In this section, we begin by explaining the notations used in this chapter. This is then followed by a brief description of the proposed SLIREC algorithm for time-to-event data. In Table 5.1, we present the notations and their brief implications. Let X and Y be symmetric $p \times p$ matrices, then $X \succ 0$ and $X \succeq 0$ implies that X is positive definite and positive semidefinite, respectively.

Table 5.1: Notations used in this chapter.

Name	Description
X	$\mathbb{R}^{n \times p}$ survival covariates matrix
T	$\mathbb{R}^{n \times 1}$ times response vector
δ	binary $n \times 1$ event indicator vector
k	number of events
T_e	$\mathbb{R}^{k \times 1}$ sorted unique time-to-event vector
Y	$\mathbb{R}^{n \times k}$ multi-response event matrix
$\hat{S}_{KM}(\cdot)$	survival Kaplan-Meier function
λ_1	scalar regularization parameter
Λ	$\mathbb{R}^{k \times k}$ symmetric weight matrix
β	$\mathbb{R}^{p \times k}$ regression coefficients matrix
Ω	$\mathbb{R}^{k \times k}$ events inverse covariance matrix

T represents the observed response time vector for all the instances with T_i being the response time for instance i . T_e represents the set of unique time-to-event values sorted in ascending order and both T_s, T_c represents an arbitrary time value. The vectorized listing of the elements of a $p \times p$ matrix is denoted by $vec(X)$ and the Kronecker product of matrices X and Y is denoted as $X \otimes Y$. G and H represent the gradient and Hessian matrices for a matrix function. We also use the matrix norm notation in this chapter where $\|X\|_1 = \sum_{i,j} |X_{ij}|$

and $|X|$ represents the determinant of the matrix and the absolute value when X is a scalar. $\|X\|_{1,\Lambda} = \sum_{i,j} \lambda_{ij} |X_{ij}|$ represents the weighted element-wise ℓ_1 norm where Λ is a symmetric non-negative weighted matrix, with $\Lambda = [\lambda_{ij}]$ and $\lambda_{ij} > 0$ for off-diagonal elements and $\lambda_{ii} = 0$ for diagonal entries. This is also referred to as the weighted matrix norm in our SLIREC formulation.

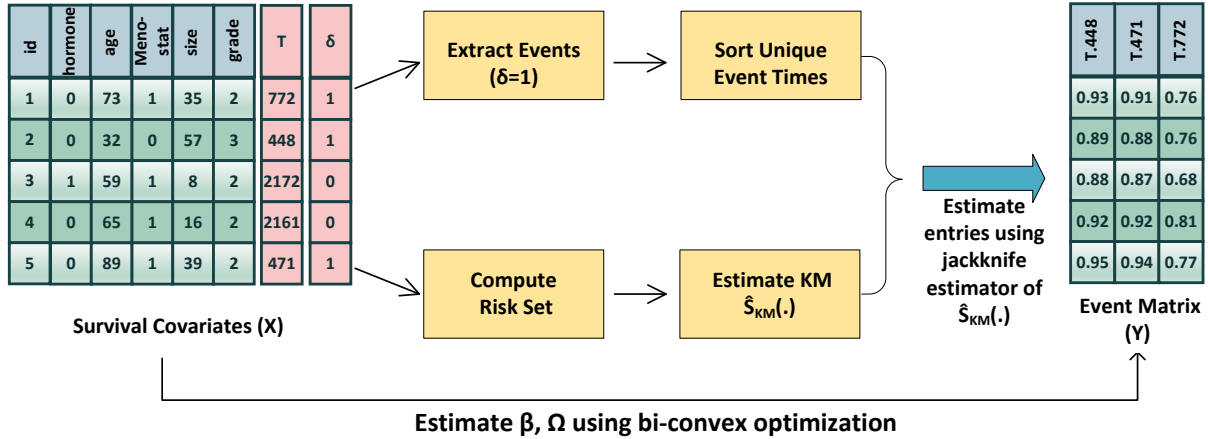


Figure 5.1: Illustrative example of SLIREC algorithm on a sample right censored dataset.

In Figure 5.1, we present an illustrative example on a sample cancer survival dataset, and also explain the intuition behind the steps used in the SLIREC algorithm. The first step of SLIREC is the event matrix generation where we initially extract events ($\delta = 1$) using T and δ . In this example, the event matrix generated on the right consists of three columns corresponding to three events at different time points (448, 471, 772). We then learn the Kaplan-Meier (KM) estimator [5, 24, 32] of the survival function ($\hat{S}_{KM}(\cdot)$) which measures the probability of the event not yet occurring, and estimate it for instance 1 at these three time points. Thus, we obtain three probability values after computing $\hat{S}_{KM}(\cdot)$ to build a $\mathbb{R}^{1 \times 3}$ event probability vector.

Once this is done, we assess the influence of instance 1 on $\hat{S}_{KM}(\cdot)$ at all three event times by recomputing the KM function without considering instance 1. This is called the *jackknife* method of estimation in the statistics literature [64]. We then obtain 3 influence values for instance 1 corresponding to each time to obtain a $\mathbb{R}^{1 \times 3}$ event influence vector.

This procedure of computing the influence is then repeated for the remaining instances at all event times to obtain a $\mathbb{R}^{5 \times 3}$ *event influence matrix*, where each cell value represents the influence of this instance in determining the event at the given time. For simplicity, we refer to this event influence matrix as the event matrix in the rest of this chapter.

Subsequently, this event matrix (Y) is used to fit a multi-response linear regression model on the survival covariates (X). This step of SLIREC also involves applying structured regularization on Y using sparse inverse covariance estimation, and use this to guide the base linear regression model for obtaining predictions \hat{Y} . One of the unique features of this approach is that the adopted structured regularization will enhance the predictiveness of the linear regression model. This feature will be illustrated later in our experiments also.

5.3 The Proposed SLIREC Algorithm

In this section, we present the SLIREC algorithm by explaining the two stages involved in this approach, namely the event matrix generation and the linear regression algorithm which imposes structured regularization on the learned event matrix. We also present the details of the optimization involved and discuss the complexity of the SLIREC algorithm.

5.3.1 Event Matrix Generation

In this section, we present the process of generating an event matrix from right censored data using the Kaplan-Meier (KM) estimator. We begin by explaining the formulation of the KM estimator, and we also explain the jackknife estimation method used for estimating the pseudo-response variables to populate the event matrix [65]. The novelty of this approach is that it is a non-parametric method of learning a multi-response representation of the dataset and it estimates pseudo-response values even for the right censored instances. The key contribution of this method is that it relies solely on using only T and δ to obtain these pseudo-responses making it more viable to apply linear regression-based prediction models for right censored data, which could not be done before. An important feature of the generated event matrix is that there are individual response variables in the event matrix (Y) corresponding to each unique time-to-event variable as shown in Figure 5.1. We now explain

the procedure of estimating the columns of Y using the non-parametric KM estimator.

The starting point for the KM estimator is a sample of n independent observations from a survival dataset with k unique events with the event times ordered as $T_1 < T_2 < \dots < T_k$. At any arbitrary time T_s , we can define the number of events observed as e_s , and the number of instances with event times greater than or equal to T_s at risk are represented by r_s . The conditional probability of surviving beyond time T_s can then be defined as $p(T_s)$.

$$p(T_s) = \frac{r_s - e_s}{r_s} \quad (5.1)$$

$$\hat{S}_{KM}(T_c) = \prod_{T_s < T_c} p(T_s)$$

In this equation, the KM estimator $\hat{S}_{KM}(T_c)$ can be interpreted as the probability of the event not yet occurring until an arbitrary time T_c . This is calculated by estimating the cumulative probability of the event not occurring at each of the preceding time intervals ($T_s < T_c$), and subsequently multiplying all these preceding probability values to obtain the final probability. This method of probability estimation is completely non-parametric and it does not make any assumptions about the survival covariates. This makes it more suitable for our approach compared to semi-parametric and parametric estimation methods.

We now explain the procedure used to estimate the values in the event matrix (Y) using $\hat{S}_{KM}(T_c)$, which represents the KM estimator of the survival function evaluated at a specific event time T_c . The entry in the event matrix for instance j at time point T_c i.e. $(Y_j(T_c))$ can then be defined as given in Eq. (5.2), where $\hat{S}_{KM}^{-j}(\cdot)$ is the KM estimator based on the instances other than the j^{th} instance.

$$Y_j(T_c) = n\hat{S}_{KM}(T_c) - (n - 1)\hat{S}_{KM}^{-j}(T_c) \quad (5.2)$$

As mentioned earlier, this method of estimation is called the *jackknife approach*, where we estimate the change in $\hat{S}_{KM}(\cdot)$ at every event time T_c before and after removing instance j from the dataset. This difference, as calculated by Eq. (5.2), represents the influence of instance j on predicting the occurrence of the event at time T . This influence computation is

done for all the instances to estimate the entries in the event matrix. This method depends on estimating the $\hat{S}_{KM}(\cdot)$ function alone which makes its computation much simpler.

The goal of this step of the SLIREC algorithm was to resolve the issue of missing response variables for right censored instances in survival data. This was done by generating a complete multi-response matrix which consists of the estimated pseudo responses for both censored instances and events which were obtained using the KM estimator of the survival function. This multi-response matrix is now fitted onto the survival covariates using the linear regression model. However, applying linear regression directly to predict each individual response variable is not intuitive, as this approach would completely ignore the effect of other response variables on modeling the current response.

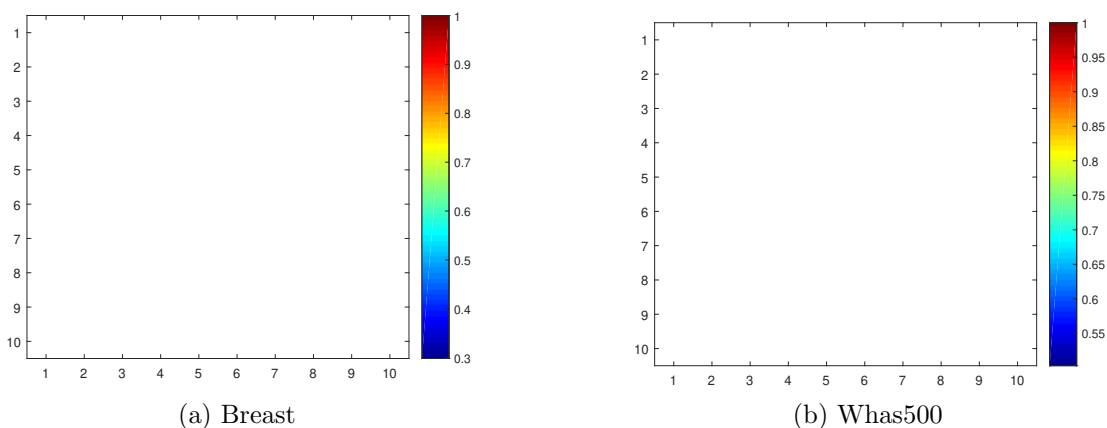


Figure 5.2: Visualizing structure in the event matrices for two survival datasets.

To resolve this problem, we resort to a structured regularization approach which can model the effects of multiple response variables by capturing their inherent structure. We visualize this correlation structure for two survival datasets, namely, Breast and Whas500 in Figure 5.2. We now explain our method which can infer this underlying structure effectively in the next section.

5.3.2 Structured Regularization based Linear Regression

In this section, we present the Structured regularization based Linear REgression algorithm for right Censored data (SLIREC) which learns a sparse linear regression model

by simultaneously learning the structure of the event matrix present in the data. In this method, two regularizers, one for the regression coefficient matrix and the other for the inverse covariance matrix over the event matrix are used. The method estimates the sparse inverse covariance matrix, and uses the learned graphical model structure to supplement the predictive ability of the original linear regression model. This is also referred to as supervised covariance-based regression. We now define the SLIREC likelihood function using the two variables (β, Ω) where $\Omega = \Sigma^{-1}$ is called as the inverse covariance matrix (also called a precision matrix).

$$L(\beta, \Omega) = \text{tr} \left[\frac{1}{n} (Y - X\beta)^T (Y - X\beta) \Omega \right] - \log |\Omega| \quad (5.3)$$

$$\hat{Y} = X\hat{\beta} + \epsilon.$$

Eq. (5.3), gives the formulation of the SLIREC likelihood function in terms of two variables β and Ω . We also present the multi-response linear regression estimation equation where the multi-response predictions \hat{Y} are obtained using X , $\hat{\beta}$ and ϵ is the error term. While estimating $\hat{\beta}$ and $\hat{\Omega}$ it is desirable to learn sparse inverse covariance matrices, as their computation can be very expensive when dealing with high-dimensional datasets. To induce sparsity, we use the weighted matrix norm penalty on the inverse covariance matrix which is defined as $\|\Omega\|_{1,\Lambda} = \sum_{i,j} \lambda_{ij} |\Omega_{ij}|$ and $\Lambda = [\lambda_{ij}]$ with $\lambda_{ij} > 0$ for the off-diagonal elements and $\lambda_{ii} = 0$ for diagonal entries. The intuition of using this regularizer is that it penalizes off-diagonal entries alone, and it also ensures that the optimal solution for Ω has a finite solution. This penalty also has an effect of reducing the number of parameters for Ω .

$$(\hat{\beta}, \hat{\Omega}) = \arg \min_{\beta, \Omega} \left\{ L(\beta, \Omega) + \|\Omega\|_{1,\Lambda} + \lambda_1 \|\beta\|_1 \right\} \quad (5.4)$$

We impose the L_1 penalty on β in this formulation to build interpretable models. The regularizers used here are the L_1 norm and the weighted matrix norm. This optimization problem is not convex; however, solving for either β or Ω with the other one fixed is convex. We now solve Eq. (5.4) using the block-coordinate descent algorithm [66]. The solution for

Eq. (5.4) with $(\beta=\beta_0)$ gives Eq. (5.5) which is a L_1 -penalized covariance estimation problem.

$$\hat{\Omega}(\beta_0) = \arg \min_{\Omega} \left\{ \text{tr}(S\Omega) - \log|\Omega| + \|\Omega\|_{1,\Lambda} \right\} \quad (5.5)$$

$$S = \frac{1}{n}(Y - X\beta_0)^T(Y - X\beta_0)$$

There are solvers such as the graphical lasso (glasso) [11] which can solve this problem. However, it is observed that for high-dimensional data the glasso solver does not scale well. To overcome this problem, we use an efficient second-order approximation based algorithm to solve Eq. (5.5), which uses the symmetric structure of the Hessian for obtaining faster convergence.

$$\hat{\beta}(\Omega_0) = \arg \min_{\beta} \left\{ \text{tr} \left[\frac{1}{n}(Y - X\beta)^T(Y - X\beta)\Omega_0 \right] + \lambda_1 \|\beta\|_1 \right\} \quad (5.6)$$

The solution for Eq. (5.6), which solves for $\hat{\beta}$ by keeping Ω fixed as Ω_0 can be obtained using the cyclic-coordinate descent algorithm. The coordinate descent steps can be simply obtained through the directional derivatives which are then used along with the soft-thresholding operator. The coordinate-descent procedure is guaranteed to converge provided the inverse covariance matrix is positive semi-definite which is ensured through our estimation procedure.

As outlined in Algorithm 5.1, the first step of our approach is to use T along with the event indicator δ to learn the event matrix Y . This event matrix is generated using the procedure explained in Section 5.3.1. Subsequently, the survival covariates X and the event matrix Y are used within the SLIREC algorithm. The scalar regularization parameter λ_1 and the symmetric weight matrix Λ are provided as inputs while solving the individual convex optimization problems with respect to β and Ω , respectively.

We iteratively estimate $\hat{\beta}$ and $\hat{\Omega}$ until convergence. This algorithm uses the block-coordinate descent optimization method, and $\hat{\Omega}$ is estimated using an efficient second-order

approximation approach which is explained in the next section. The intuition behind using Algorithm 5.2 for solving Eq. (5.5) is to obtain faster convergence of the solution. After convergence of both these parameters, we use X and $\hat{\beta}$ to obtain the final prediction matrix using Eq. (5.3). This prediction matrix \hat{Y} is a $\mathbb{R}^{n \times k}$ matrix where each cell represents the probability of event of interest not yet occurring at the specified time for each instance. We explain the optimization steps we used in our algorithm to improve its efficiency in the next section.

Algorithm 5.1: SLIREC Algorithm

- 1 **Input:** Survival covariates matrix $X \in \mathbb{R}^{n \times p}$, Times vector T , Event indicator δ , regularization parameter λ_1 , weight matrix $\Lambda \in \mathbb{R}^{k \times k}$, tolerance parameter ϵ , maximum iterations max_iter
 - 2 **Output:** Regression coefficients matrix $\hat{\beta}_{iter+1} \in \mathbb{R}^{p \times k}$, Events inverse covariance matrix $\hat{\Omega}_{iter+1} \in \mathbb{R}^{k \times k}$
 - 3 **Initialize** $\hat{\beta}_0, \hat{\Omega}_0$;
 - 4 Generate event matrix Y using T and δ using Eq. (5.1) and Eq. (5.2);
 - 5 **for** $iter = 0, \dots, max_iter$ **do**
 - 6 Compute $\hat{\beta}_{iter+1}$ by solving Eq. (5.6) using $\hat{\Omega}_{iter}$;
 - 7 Compute $\hat{\Omega}_{iter+1}$ by solving Eq. (5.5) using Algorithm 5.2 and $\hat{\beta}_{iter+1}$;
 - 8 **if** $\|\hat{\beta}_{iter+1} - \hat{\beta}_{iter}\|_1 < \epsilon$ **then**
 - 9 | break;
 - 10 **end**
 - 11 **end**
-

5.3.3 Optimization

In this section, we present an effective second-order approximation-based algorithm to solve the optimization problem in Eq. (5.5). The intuition for using a second-order approximation is to obtain faster convergence rates compared to first-order methods which converge at a slow rate of $O(1/\sqrt{n})$. However, to avoid the intensive Hessian computation, approximation methods such as the quasi-Newton method and the proximal-Newton method have been proposed [63]. The solver used by us here is a variant of this proximal-Newton method which is explained below.

We represent the composite objective function in this equation as $f(\Omega)$. This objective

function is composed of two parts such that $f(\Omega) \equiv g(\Omega) + h(\Omega)$ where $g(\Omega)$ is convex and $h(\Omega)$ is convex, but not differentiable (non-smooth). This minimization problem can be solved using the second-order Taylor expansion of $g(\Omega)$. The second-order expansion for the log-determinant function is given in Eq. (5.7) where t indicates the iteration index for the sequence of Ω values generated.

$$f(\Omega) \equiv g(\Omega) + h(\Omega) \quad (5.7)$$

$$g(\Omega) = \text{tr}(S\Omega) - \log|\Omega|$$

$$h(\Omega) = \|\Omega\|_{1,\Lambda}$$

$$\bar{g}_{\Omega_t}(\Delta) \equiv g(\Omega_t) + \text{vec}(\nabla g(\Omega_t))^T \text{vec}(\Delta) + \frac{1}{2} \text{vec}(\Delta)^T \nabla^2 g(\Omega_t) \text{vec}(\Delta)$$

$$D_t^* = \arg \min_{\Delta} \{\bar{g}_{\Omega_t}(\Delta) + h(\Omega_t + \Delta)\}$$

We now define the Newton direction D_t^* for the objective function $f(\Omega)$ which can then be written as the solution of the regularized quadratic program given in Eq. (5.7) and Eq. (5.8). This Newton direction computation problem can be expressed as a lasso problem as shown in Eq. (5.8) with the gradient and Hessian formulations given in Eq. (5.9). Subsequently, the standard coordinate descent method is used to solve this equation to obtain the Newton direction.

$$\arg \min_{\Delta} \frac{1}{2} \| H^{\frac{1}{2}} \text{vec}(\Delta) + H^{-\frac{1}{2}} b \|^2 + \|\Omega_t + \Delta\|_{1,\Lambda} \quad (5.8)$$

In this Eq. (5.9), $H = \nabla^2 g(\Omega_t)$ and $b = \text{vec}(\nabla g(\Omega_t))$. The Hessian and Gradient matrices G and H can be written as given in Eq. (5.9).

$$S = \frac{1}{n} (Y - X\beta_0)^T (Y - X\beta_0) \quad (5.9)$$

$$G = \nabla g(\Omega_t) = S - \Omega_t^{-1}$$

$$H = \nabla^2 g(\Omega_t) = \Omega_t^{-1} \otimes \Omega_t^{-1}$$

In Algorithm 5.2, we describe the algorithm used for solving Eq. (5.5). The algorithm

is based on obtaining the Newton direction D_t^* by solving a lasso problem using coordinate descent, and using the obtained Newton direction along with an Armijo-rule based step size selection (α) to obtain the next positive definite iterate (Ω_t). The calculation of the Newton direction is simplified when Ω is a diagonal matrix, as the Hessian matrix $H = \Omega_t^{-1} \otimes \Omega_t^{-1}$ is also a diagonal matrix, therefore the time complexity for the Newton direction update reduces from $O(k^3)$ to $O(k^2)$, which is why this is referred to as a second-order approximation method.

We now discuss the complexity of the SLIREC algorithm by analyzing the complexity of both the stages involved. The time needed to generate the event matrix representation using the KM approach is constant in general and does not depend upon the number of events in the dataset. The block-coordinate descent method used in the bi-regularized linear regression model needs np units of time for the coordinate descent step and the computation of the events inverse covariance matrix takes k^2 units of time. Hence, the overall time complexity of the SLIREC algorithm can be calculated as $O(np + k^2)$.

Algorithm 5.2: Efficient Solver for Eq. (5.5) in SLIREC

```

1 Input: Event matrix  $Y \in \mathbb{R}^{n \times k}$ , parameters  $\sigma, \beta, \delta$ 
2 Output: Events inverse covariance matrix  $\Omega_{t+1} \in \mathbb{R}^{k \times k}$ 
3 for  $t = 1, \dots$ , do
4   for  $\alpha = 1, \beta, \beta^2, \dots$  do
5     Compute the Cholesky factorization  $LL^T = \Omega_t + \alpha D_t^*$ ;
6     if  $\Omega_t + \alpha D_t^* \succ 0$  then
7       Compute  $f(\Omega_t + \alpha D_t^*)$  from  $L$  and  $\Omega_t + \alpha D_t^*$ ;
8       if  $f(\Omega_t + \alpha D_t^*) \leq f(\Omega_t) + \alpha \sigma \delta$  then
9         break;
10      end
11    end
12     $\Omega_{t+1} = \Omega_t + \alpha D_t^*$ ;
13  end
14 end

```

5.3.4 Theoretical Analysis

In this section, we discuss the proof for the line search condition which ensures the descent property by finding the next positive definite iterate. The Armijo line search rule is stated in Eq. (5.10) where we try step sizes $\alpha \in \{\beta^0, \beta^1, \dots\}$ with a constant decrease rate $0 < \beta < 1$, until we find the smallest k with $\alpha = \beta^k$ such that $\Omega_t + \alpha D_t^*$ is positive definite and satisfies the decrease condition for $0 < \sigma < 0.5$. We now provide the theorem and proof for the Armijo condition.

$$f(\Omega_t + \alpha D_t^*) \leq f(\Omega_t) + \alpha \sigma \delta \quad (5.10)$$

$$\delta = \text{tr}(\nabla g(\Omega_t)^T D_t^*) + \|\Omega_t + D_t^*\|_{1,\Lambda} - \|\Omega_t\|_{1,\Lambda}$$

Theorem 3. *For the symmetric inverse covariance matrix $\Omega_t \succ 0$, and the symmetric Newton direction D_t^* , there exists an $\bar{\alpha} > 0$ such that for all $\alpha < \bar{\alpha}$, the matrix $\Omega_t + \alpha D_t^*$ satisfies the line search condition given in Eq. (5.10).*

Proof. We use the fact that the matrix weighted norm satisfies the inequality given in Eq. (5.11). This inequality can be proved trivially by considering the convex nature of the $\|\cdot\|_{1,\Lambda}$ norm.

$$\begin{aligned} \|\Omega_t + \alpha D_t^*\|_{1,\Lambda} &= \|\alpha(\Omega_t + D_t^*) + (1 - \alpha)\Omega_t\|_{1,\Lambda} \\ &\leq \alpha \|\Omega_t + D_t^*\|_{1,\Lambda} + (1 - \alpha) \|\Omega_t\|_{1,\Lambda} \end{aligned} \quad (5.11)$$

$$\begin{aligned} f(\Omega_t + \alpha D_t^*) - f(\Omega_t) &= g(\Omega_t + \alpha D_t^*) - g(\Omega_t) \\ &\quad + \|\Omega_t + \alpha D_t^*\|_{1,\Lambda} - \|\Omega_t\|_{1,\Lambda} \\ &\leq g(\Omega_t + \alpha D_t^*) - g(\Omega_t) \\ &\quad + \alpha(\|\Omega_t + D_t^*\|_{1,\Lambda} - \|\Omega_t\|_{1,\Lambda}) \\ &= \alpha \text{tr}((\nabla g(\Omega_t))^T D_t^*) + O(\alpha^2) \\ &\quad + \alpha(\|\Omega_t + D_t^*\|_{1,\Lambda} - \|\Omega_t\|_{1,\Lambda}) \\ &= \alpha \sigma \delta \end{aligned} \quad (5.12)$$

This proof shows that the Armijo line search condition is satisfied by finding the next positive definite iterate which ensures the descent of the objective function. \square

We now explain the conditions which ensures that the approximation method used in SLIREC algorithm converges to a global optimum. Subsequently, we define the necessary conditions needed to be satisfied for convergence.

Theorem 4. *There exists a unique minimizer $\hat{\Omega}$ for Eq. (5.7).*

Proof. This can be proved using the fact that $H = \nabla^2 g(\Omega) = \Omega^{-1} \otimes \Omega^{-1}$ is convex since $\|\Omega\|_{1,\Lambda}$ is convex and $-\log|\Omega|$ is strongly convex; hence we have that $f(\Omega)$ is strongly convex and the minimizer $\hat{\Omega}$ for this function is unique from the property of strong convex functions. \square

We now briefly state the conditions required for the convergence of the solution Ω_t in Eq. (5.7) using some of the theory of strictly convex functions. In this regard, we also provide the Newton update step for the constrained minimization problems.

Theorem 5. *Assume f is strictly convex and f has a unique minimizer $\hat{\Omega}$ and that $\nabla^2 f(\Omega)$ is Lipschitz continuous. Then for all Ω_t sufficiently close to $\hat{\Omega}$, the sequence Ω_t generated by Eq. (5.13) converges quadratically to $\hat{\Omega}$.*

$$\begin{aligned} \Omega_{t+1} = \arg \min_{\Omega} & \nabla f(\Omega_t)^T (\Omega - \Omega_t) \\ & + \frac{1}{2} (\Omega - \Omega_t)^T \nabla^2 f(\Omega_t) (\Omega - \Omega_t) \end{aligned} \quad (5.13)$$

Proof. The objective function used in the SLIREC algorithm in Eq. (5.7) is convex and non-smooth, so before proving the convergence; we need to modify the formulation of this optimization problem. This conversion can be done by dividing the index set with $\lambda_{ij} \neq 0$ into three subsets which are given in Eq. (5.14). Using these three subsets which represent positive (P), negative (N) and zero sets (Z) respectively, we can now define a constrained minimization problem in Eq. (5.15) which satisfies all the conditions mentioned above, and whose optimum corresponds to the same as the global optimum of Eq. (5.7); hence proving the convergence.

$$\begin{aligned} P &= \{(i, j) | \Omega_{ij} > 0\} \\ N &= \{(i, j) | \Omega_{ij} < 0\} \\ Z &= \{(i, j) | \Omega_{ij} = 0\} \end{aligned} \quad (5.14)$$

$$\begin{aligned} \arg \min_{\Omega} & -\log|\Omega| + \text{tr}(S\Omega) + \sum_{(i,j) \in P} \lambda_{ij} \Omega_{ij} - \sum_{(i,j) \in N} \lambda_{ij} \Omega_{ij} \\ \text{s.t.} & \quad \Omega_{ij} > 0 \quad \forall (i, j) \in P \\ & \quad \Omega_{ij} < 0 \quad \forall (i, j) \in N \\ & \quad \Omega_{ij} = 0 \quad \forall (i, j) \in Z \end{aligned} \quad (5.15)$$

This completes explaining the conditions associated with attaining the global optimum and the convergence of the approximation method used in the SLIREC algorithm. \square

5.4 Experimental Results

In this section, we compare the performance of our proposed SLIREC algorithm against both survival and linear regression-based algorithms. Table 5.2 provides the description of the datasets used in our experiments. Primary biliary cirrhosis (PBC) dataset is from the Mayo Clinic trial in cirrhosis of the liver. Breast and Colon cancer datasets are from the German Breast Cancer Study Group. The Diffuse Large B-Cell Lymphoma (DLBCL) dataset consists of Lymphochip DNA microarrays from 240 biopsy samples of DLBCL tumors for studying the survival status over 21 years. The Norway Stanford Breast Cancer Data (NSBCD) consists of breast cancer measurements for 115 women with breast cancer observed for 188 months to monitor the death time. The Lung dataset consists of the gene expression profiles of 86 early stage lung adenocarcinoma patients for a period of 110 months. The electronic health record (EHR) is a proprietary dataset consists of clinical and behavioral variables for patients monitored for heart failure readmissions at a major hospital in the US. These datasets except for the EHR dataset were obtained from these websites ³ and ⁴.

Table 5.2: Description of the datasets used.

Dataset	# Instances	# Features	# Events	Censored(%)
PBC	418	17	109	61
Breast	686	8	270	56
Colon	888	13	364	50
whas1	481	14	180	52
whas500	500	22	162	43
DLBCL	240	7399	134	42
NSBCD	115	549	26	67
Lung	86	7129	24	72
EHR	789	183	297	54

5.4.1 Implementation Details

We implemented our entire code for the proposed SLIREC algorithm in the R programming language using several CRAN repositories. The computation of the event matrix was

³<https://www.umass.edu/statdata/statdata/stat-survival.html>

⁴<http://user.it.uu.se/~liuya610/download.html>

done using the *pseudo* R package which generates the pseudo-observations for survival analysis. The structured regularization component of SLIREC was implemented using the *mrce* R package. The second-order approximation method was implemented using the *quic* R package. The *survcomp*, *survival*, *ipred* R packages were used for obtaining the survival AUC, standard deviation and Brier score metrics. The *CoxBoost*, *randomforestSRC*, *superpc* and *Coxnet* R packages were used to implement the CoxBoost, RSF, SuperPC and regularized Cox regression algorithms, respectively. The linear regression based methods which can be applied for predicting multiple response variables of the event matrix were obtained using the Multi-Label Dimensionality Reduction (MLDR) Matlab package available from here⁵.

5.4.2 Evaluating Importance of Structured Regularization

In this section, we evaluate the importance of the structured regularization component of SLIREC by comparing its performance with various linear regression based methods which vary in their ability to infer the underlying structure. This experiment is feasible because after applying the event matrix generation step on the right censored data, we obtained a unique multi-response representation which can be used to fit linear regression based methods and our structured regularization based linear regression method. In this experiment, we compare with three different baseline linear regression based methods which are CCA, OPLS and SSL, respectively. These methods are briefly described below.

- *Canonical Correlation Analysis (CCA)* [29]: CCA is a method used to identify correlations between two sets of multi-dimensional variables, which are the survival covariates and the event matrix. This projected representation obtained by CCA is used in a multi-response linear regression framework.
- *Orthonormalized Partial Least Squares (OPLS)* [30]: OPLS is a method which considers two identical sets of multi-dimensional variables as done for CCA, and it creates orthogonal score vectors maximizing the covariance between different sets of variables.

The projected representation obtained by OPLS is then used within a multi-response

⁵<http://www.yelab.net/software/MLDR/>

linear regression framework.

- *Shared Subspace Learning (SSL)* [31]: SSL extracts shared structures for multi-response prediction by capturing the correlation among the different columns of the event matrix by extracting a lower dimensional subspace. Subsequently, the projections of the survival covariates and the event matrix onto the extracted lower-dimensional subspace are used within a multi-response linear regression framework.

We obtained the prediction matrix \hat{Y} given in Eq. (5.3) using CCA, OPLS, SSL and SLIREC methods. The survival prediction matrix \hat{Y} can then be used to obtain the time-based $AUC(T_c)$ values for modeling each response variable of Y using Eq. (3.20). This vector consists of the $AUC(T_c)$ values and is computed over each individual unique time-to-event for these datasets. In this manner, this vector of $AUC(T_c)$ values obtained for each dataset using each of the four algorithms can be visualized using a boxplot shown in Figure 5.3. From this figure, one can observe that, in all the cases, SLIREC performs the best, and its range of values is either better or comparable to those of SSL. The better performance of SLIREC and SSL is attributed to the fact that both these methods consider the structure of the event matrix and use this knowledge during prediction. CCA and OPLS, on the other hand, cannot leverage on this rich structural knowledge which reflects in their comparatively lower AUC values. This demonstrates the importance of using structured regularization based methods for building predictive models for right censored data.

5.4.3 Evaluation using Survival Models

In this section, we evaluate the performance of SLIREC by comparing its performance with several other state-of-the-art survival regression algorithms which are described below.

- *Elastic-net (EN-COX)* [16]: EN-COX integrates the elastic net penalty with the Cox partial log-likelihood loss function. KEN-COX [57] supplements EN-COX with an additional feature kernel term to capture more feature specific information of the survival covariates.
- *Laplacian Net Cox (LAPNET-COX)* [17]: LAPNET-COX computes the graph Lapla-

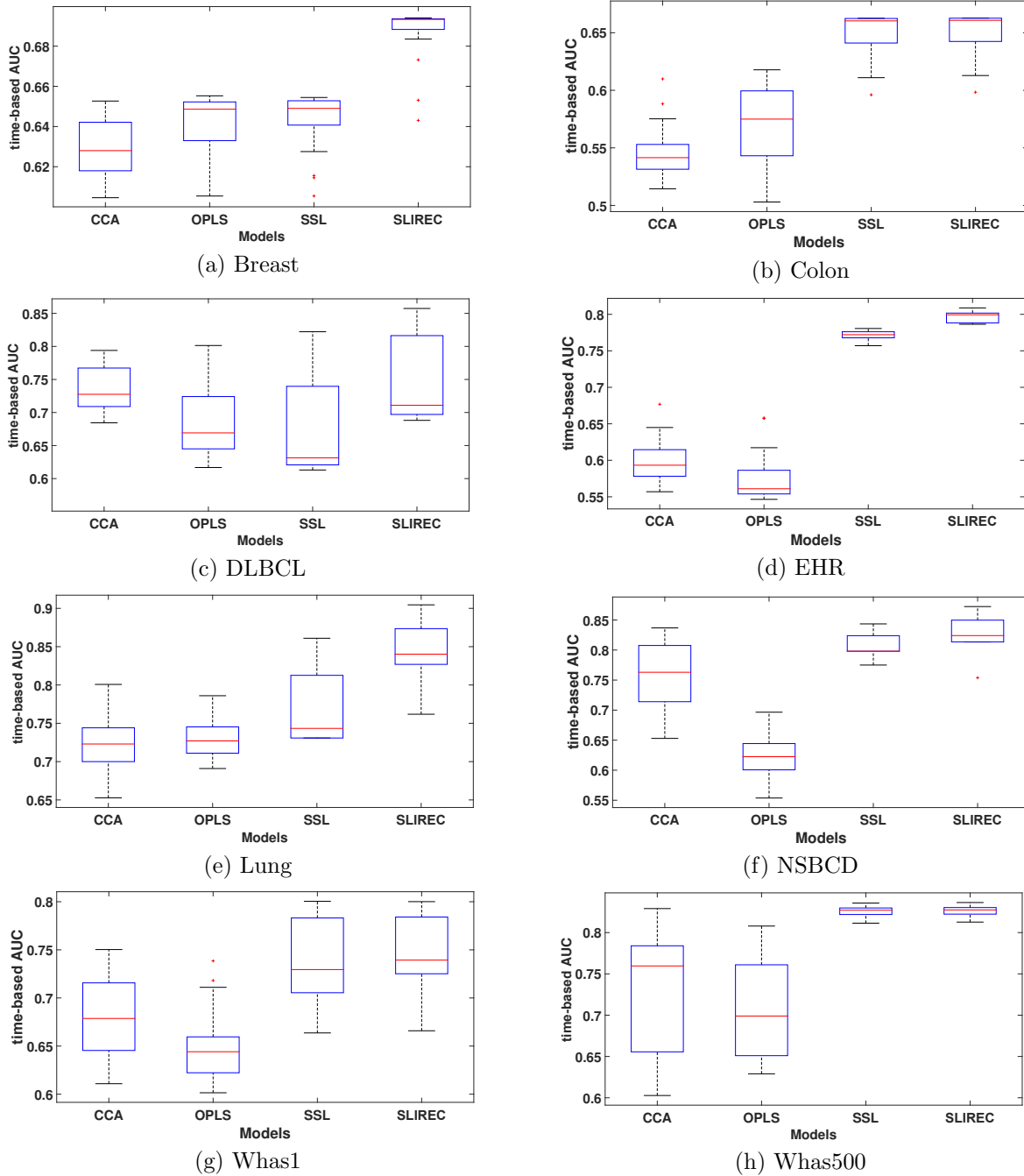


Figure 5.3: Performance comparison using CCA, OPLS, SSL and SLIREC methods on various survival datasets.

cian over the survival covariates and integrates this into the elastic-net Cox algorithm.

The graph Laplacian is used here to capture structural knowledge of the data.

- *Elastic Net Buckley James (EN-BJ)* [25]: EN-BJ uses a semi-parametric accelerated failure time (AFT) model with the elastic net regularization.
- *Random Survival Forests (RSF)* [28] and *CoxBoost* [67]: RSF and CoxBoost are ensemble based methods which use survival trees and boosting for prediction, respectively.

We report the survival AUC (and standard deviation values) obtained from various survival regression methods in Table 5.3. The regularization parameters were determined using a tuning set, after evaluating a sequence of values and the optimal value corresponding to the minimum MSE is chosen. Following this the values in Table 5.3 was obtained after using these regularization parameters and five-fold cross-validation to obtain standard deviation estimates. The results indicate that SLIREC performs better than competing algorithms in 6 out of 9 datasets, and performs competitively in the remaining cases. The better performance of SLIREC is attributed to its ability to use effective structured regularization, augmenting the predictive ability of the base sparse linear regression model.

5.4.4 Goodness of Survival Prediction

We also present the results on assessing the goodness of the survival predictions obtained from the SLIREC algorithm and other competing survival regression algorithms using the Brier score metric. As mentioned earlier, we prefer survival models with lower Brier score values. In Table 5.4, we present the integrated brier score (IBS) values for different benchmark datasets using various survival regression methods. The values in Table 5.4 indicate that SLIREC provides more effective predictions for 7 out of 9 datasets considered. This indicates that our approach can be used to obtain reliable survival predictions which are often needed in several real-world applications.

5.4.5 Scalability Experiments

In this section, we present the experimental results which assess the improvement in the runtime of SLIREC algorithm before and after applying the second-order approximation

Table 5.3: Survival AUC and standard deviation values for the SLIREC algorithm compared to other survival regression models.

Dataset	EN-COX	KEN-COX	LAPNET-COX	EN-BJ	RSF	CoxBoost	SLIREC
Breast	0.671 (0.017)	0.682 (0.087)	0.664 (0.014)	0.680 (0.065)	0.679 (0.029)	0.686 (0.014)	0.691 (0.014)
Colon	0.646 (0.054)	0.635 (0.019)	0.642 (0.021)	0.663 (0.027)	0.638 (0.051)	0.661 (0.016)	0.661 (0.016)
PBC	0.742 (0.031)	0.790 (0.025)	0.778 (0.030)	0.764 (0.028)	0.742 (0.088)	0.738 (0.018)	0.804 (0.005)
DLBCL	0.693 (0.028)	0.713 (0.028)	0.691 (0.049)	0.733 (0.058)	0.755 (0.121)	0.752 (0.009)	0.763 (0.071)
EHR	0.705 (0.015)	0.707 (0.013)	0.696 (0.091)	0.684 (0.034)	0.717 (0.070)	0.660 (0.126)	0.697 (0.082)
Lung	0.733 (0.014)	0.778 (0.030)	0.752 (0.005)	0.818 (0.088)	0.780 (0.062)	0.769 (0.011)	0.840 (0.013)
NSBCD	0.717 (0.011)	0.704 (0.004)	0.727 (0.011)	0.781 (0.047)	0.725 (0.051)	0.808 (0.074)	0.808 (0.155)
Whas1	0.753 (0.019)	0.766 (0.022)	0.741 (0.014)	0.763 (0.046)	0.726 (0.039)	0.755 (0.013)	0.786 (0.064)
Whas500	0.832 (0.020)	0.772 (0.015)	0.793 (0.061)	0.825 (0.029)	0.803 (0.029)	0.830 (0.044)	0.825 (0.036)

Table 5.4: Integrated Brier score values for the SLIREC algorithm compared to other survival regression models.

Dataset	EN-COX	KEN-COX	LAPNET-COX	EN-BJ	RSF	CoxBoost	SLIREC
Breast	0.573	0.558	0.551	0.421	0.588	0.581	0.419
Colon	0.495	0.434	0.477	0.448	0.506	0.510	0.492
PBC	0.652	0.573	0.515	0.495	0.611	0.635	0.331
DLBCL	0.634	0.658	0.619	0.664	0.631	0.668	0.672
EHR	0.319	0.445	0.382	0.403	0.419	0.690	0.183
Lung	0.620	0.696	0.677	0.523	0.514	0.248	0.215
NSBCD	0.544	0.547	0.546	0.546	0.562	0.381	0.371
Whas1	0.503	0.562	0.493	0.454	0.471	0.533	0.432
Whas500	0.631	0.585	0.602	0.440	0.623	0.359	0.314

technique described in Section 5.3.3. We consider two datasets in this experiment which are Lung and DLBCL. We iteratively sample instances with varying feature dimensionality from these datasets, and mark these values on the x-axis. We also measure the time taken (in seconds) for execution of SLIREC algorithm before and after applying the approximation technique for these sets of instances, and plot the corresponding time taken in seconds on the y-axis.

From Figure 5.4 and Figure 5.5, we observe that the runtime of SLIREC after applying the approximation is significantly smaller than the one before applying it. It is also observed that the runtime for SLIREC with the approximation does not vary significantly despite increasing the feature dimensionality. This is due to the acceleration provided by second-order approximation or Newton-based methods which are known to obtain super-linear convergence rates. This proves the importance of our approximation technique while applying the SLIREC algorithm on different datasets.

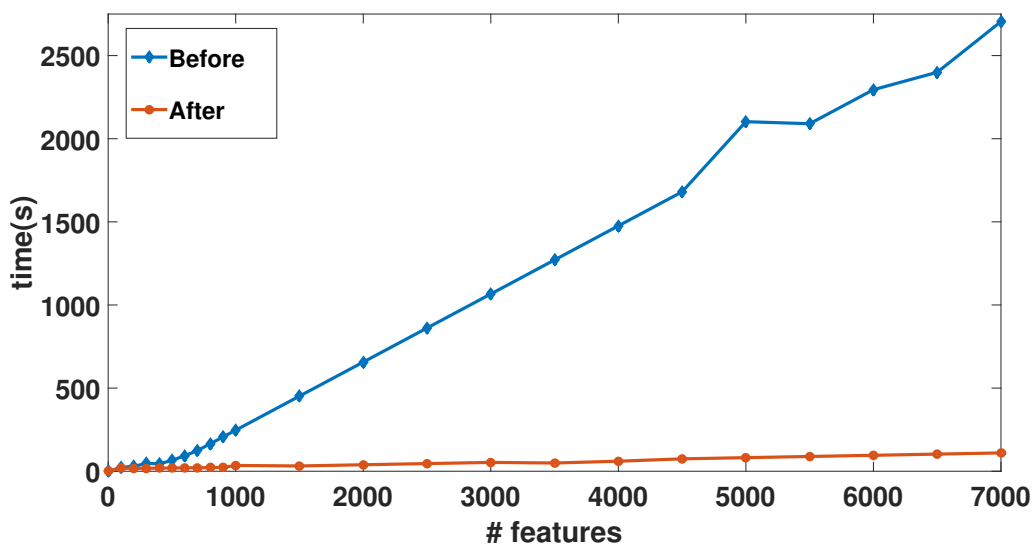


Figure 5.4: Measuring improvement in runtime before and after applying the approximation scheme in SLIREC for Lung dataset.

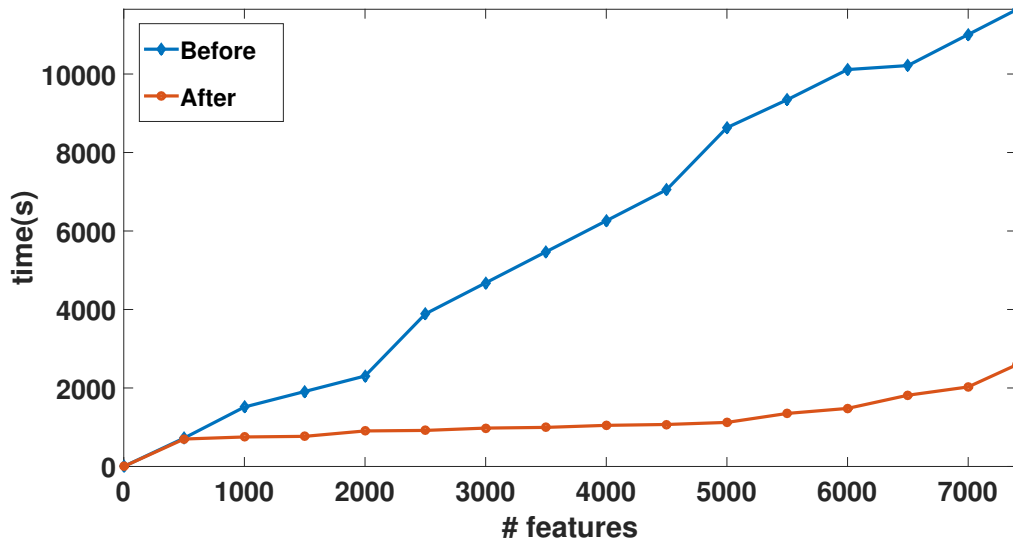


Figure 5.5: Measuring improvement in runtime before and after applying the approximation scheme in SLIREC for DLBCL dataset.

CHAPTER 6: ACTIVE LEARNING BASED SURVIVAL REGRESSION

6.1 Motivation

In this chapter, we present an approach for learning a model from time-to-event data which evaluates the impact of adding an instance to the model. This is an important problem because the quality of training data chosen determines the goodness of the learned model. Some methods which can be applied for acquiring a good set of training examples include active learning and semi-supervised learning methods. Semi-supervised learning methods rely on using side information in the form of pairwise constraints or co-training methods to build models. Active learning is different from semi-supervised learning, as the model learning process is more dynamic here, with instances being queried for labels at the end of each iteration. The oracle (labeling expert) is constantly involved in the active learning framework which is not the case with semi-supervised learning methods, as the expert-based information is provided at the beginning itself.

In the literature, active learning methods have been used more frequently for the binary classification problem rather than the regression problem where the outcome variable is continuous. The problem of learning a model from survival data is unique as it is both a classification as well as a regression problem. It can be viewed as a classification problem considering the fact that there are two well-defined classes which are events (positive class) and censored (negative class). Simultaneously, it can also be viewed as a regression problem, as we are trying to predict a continuous valued time-to-event label. In Figure 6.1, we provide a small illustration of learning a survival regression model on a sample synthetic dataset where it can be viewed as both a classification and a regression problem.

In Figure 6.1, we fit a Cox regression model with a Weibull base hazard rate using a single covariate on a set of points which consists of both events and censored instances. We then learn the Cox model on this data and plot the subsequent predicted time-to-event values. This is the regression component of learning the survival model. We also represent both

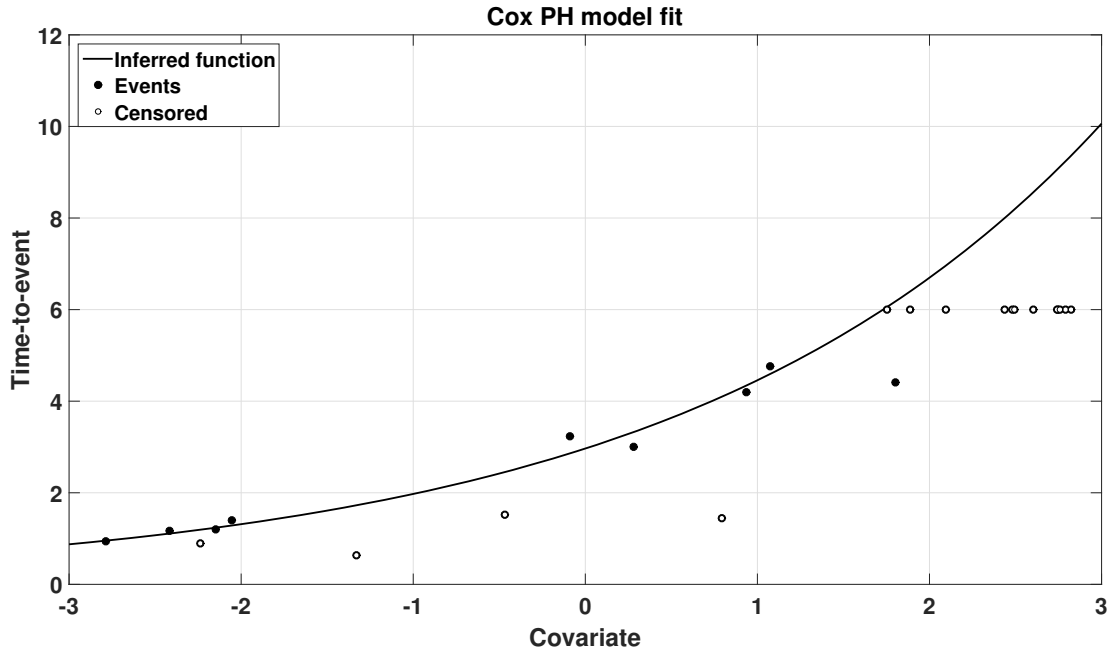


Figure 6.1: Survival Regression viewed as a binary classification problem.

the censored instances and the events present in the data in this plot which gives us the perception of this being a binary classification problem, where we have to classify if a given instance will be censored or be a possible event. This phenomenon of survival regression where it can be interpreted as a classification problem makes it conducive to apply binary classification based active learning methods such as uncertainty based sampling.

The reformulated classification problem of learning a survival model from data which consists of both events and censored instances can be viewed as a problem of learning a model from the data consisting of both labeled instances (events) and a unlabeled instances (censored instances). In such a scenario, we can build a model on a small training sample consisting of few events and censored instances whose labels are obtained from an oracle (ground truth). This model can then be applied on the remaining data to identify those instances which can be added to the model. Subsequently, the sampled instances are queried for their time-to-event labels using the oracle. In each iteration, a fixed number of instances are sampled and queried for their labels before being added to the model. This iterative learning procedure is carried on until the learning model stabilizes. This is called the active

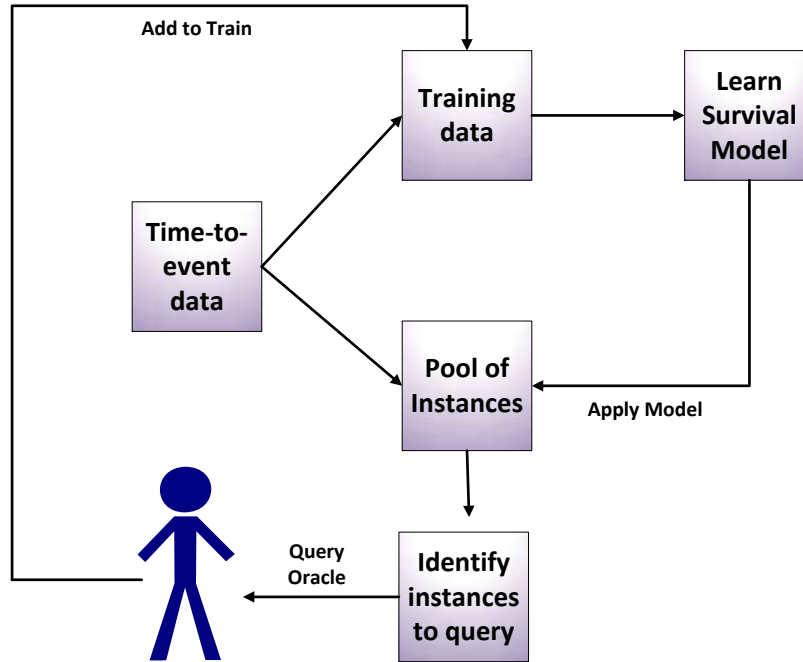


Figure 6.2: Active learning cycle for time-to-event data.

learning cycle and is given in Figure 6.2.

The advantage of using an active learning approach compared to using a semi-supervised learning method is that this allows the model to select instances which should be queried for labels and added to the training data. In this context, this method helps in building a model which is more *intelligible* compared to using a semi-supervised learning method. This form of learning also benefits by utilizing the information in censored instances more explicitly. In addition, the active learning framework does not depend on the base survival model being used which makes it very flexible to use it with different kinds of survival regression algorithms. In this chapter, we study this active learning method using several Cox regression algorithms. In the next section, we provide the motivation of using a Cox regression method and we also provide some real-world examples which would benefit by using active learning-based survival models.

Active learning from time-to-event data can be very useful in a wide range of applications where a domain expert (oracle) can be involved in the model building process. For example, in healthcare applications, the survival model can select instances by learning from a small

labeled set of instances and then query the expert to receive the time-to-event label before including it in the model. This expert feedback can help in refining the model which is particularly useful for healthcare applications such as predicting *30-day* readmission risk [45, 68]. In such applications, the domain expert can integrate domain knowledge into the survival model to build a more robust model.

Active learning in this domain is particularly challenging because the model must choose an instance from both censored and uncensored set of instances in the dataset and query the expert to obtain the time-to-event label. In general censored data mining tasks, censored instances are either deleted or the missing values are imputed to convert it into an *uncensored* problem. An important challenge here lies in utilizing the censored instance completely while building the active learning based survival regression model without deleting or modifying the instance.

Over the past few years, data mining methods have been tuned to predict from censored data. Machine learning methods such as neural networks [69], random forests [28] and support vector machine [70] based approaches have been applied to deal with censored data. These methods in particular can handle non-linear relations between the covariates in censored data. Survival regression methods such as Cox proportional hazards [6] and Accelerated failure time (AFT) [59] models are also used to build regression models from censored data.

Cox regression differs from other methods mentioned above since it estimates the relative risk rather than the absolute risk of occurrence of the event. In the healthcare scenario, this is highly useful for a doctor to compare two patients from the same cohort to identify who is at a relatively higher risk. Cox regression also has a simple formulation which consists of just estimating two quantities (i) the unspecified baseline hazard function and (ii) a linear function of the set of covariates. The major contributions of this chapter are as follows.

- We present an Active Regularized Cox regression (ARC) framework which effectively integrates active learning and Cox regression using a novel model discriminative gra-

dient sampling strategy and robust regularization. Regularization helps in providing good generalizability in ARC and the model discriminative gradient sampling encourages selecting appropriate instances to be labeled by the domain expert. ARC is tested on electronic health records (EHR), synthetic and publicly available survival datasets.

- Experimental results over different datasets indicate that ARC outperforms competing methods and attains very competitive AUC values. To our knowledge, this is the first work which combines active learning with Cox regression for predicting time-to-event outcomes in the 30-day readmission problem [45, 68] for heart failure.

This chapter is organized as follows. In Section 6.2, we present the preliminaries needed to comprehend the ARC algorithm. In Section 6.3, the algorithm for the coordinate majorization descent (CMD) based regularized Cox regression (*RegCox*) is provided and the proposed ARC algorithm is explained. The model discriminative gradient-based sampling strategy used in this approach is also explained. In Section 6.4, experimental analysis is conducted to evaluate ARC against different kinds of survival regression algorithms.

6.2 Preliminaries

In this section, we introduce the preliminaries on Cox regression and the notations needed to interpret the active regularized Cox regression (ARC) method. This is followed by reviewing some of the concepts of the Cox regression framework.

Cox regression is one of the most widely used survival analysis methods. It is a semi-parametric regression model which can accommodate both discrete and continuous measures of event times. It assumes that conditioned on the covariates X all risks are statistically independent, and that the hazard probability of the primary risk for individuals with covariates X is a function of the following parametrized form.

$$h(t|X) = h_0(t) \times \exp(X \cdot \beta) \tag{6.1}$$

$$h(t|X) = \underbrace{h_0(t)}_{\text{base hazard rate}} \times \underbrace{\exp(X_1\beta) \times \dots \times \exp(X_m\beta)}_{\text{proportional hazards}}$$

In Eq. (6.1), $X \cdot \beta = \sum_{\mu=1}^m X_{\mu}\beta_{\mu}$ with time independent parameters $\beta = (\beta_1, \dots, \beta_m)$.

Table 6.1: Notations used in this chapter.

Name	Description
X	$n \times m$ matrix of feature vectors
T	$n \times 1$ vector of failure times
K	number of unique failure times
δ	$n \times 1$ binary vector of censored status
R_i	set of all instances at risk at time T_i ($T_j > T_i$)
β	$m \times 1$ regression coefficient vector
$L(\beta)$	partial log-likelihood
$h(t X)$	conditional hazard probability
$h_0(t)$	base hazard rate
$S_0(t)$	base survival rate
$S(t X)$	conditional survival probability
Ke	column-wise kernel matrix

The function $h_0(t)$ is called the base hazard rate. It is the base hazard rate one would find for the trivial covariates $X = (0, 0, \dots, 0)$. The proportional hazards (PH) assumption in Cox regression also basically states that different covariates contribute each an independent multiplicative factor each to the primary risk hazard rate.

The effect of covariates are taken to be mutually independent and also independent of time. However, it is easy to incorporate time-dependent covariates also into the Cox regression model. In Cox regression, the goal is to find the most probable parameters $\beta = (\beta_1, \dots, \beta_m)$ and the most probable base hazard function $h_0(t)$.

β is estimated using the maximum likelihood estimation over the partial log-likelihood function. The base hazard function on the other hand is estimated using Eq. (3.5). This base hazard function is estimated for an arbitrary time t after calculating β . During estimation the Cox regression model does not assume knowledge of absolute risk and estimates only the relative risk.

This model is also referred to as the CoxPH (Proportional Hazards) model because of the proportional hazards assumption which states that the hazard for any individual is a fixed proportion of the hazard for any other individual. In Eq. (3.5), the formulae for

estimating the base survival function $S_0(t)$ and the conditional survival probability $S(t|X_i)$ were provided. This function models the probability of survival for an instance whereas the hazard probability models the probability of occurrence of the event of interest for an instance. Cox regression is one of the most popular survival regression models and its simple formulation makes it easier to integrate it with various other data mining techniques.

6.3 Active Learning with Regularized Survival Analysis

In this section, we explain the proposed Active Regularized Cox regression (ARC) framework. In Section 6.3.1, we explain a simple regularized Cox regression algorithm (*RegCox*) which uses the elastic net regularizer. A scalable coordinate majorization descent (CMD) based algorithm for solving this problem is provided. This is followed by explaining the model discriminative gradient based sampling strategy used in active learning. Finally, the ARC framework which combines active learning and regularized Cox regression using model discriminative gradient based sampling is explained.

6.3.1 RegCox: Regularized Cox Regression

Cox regression models have the tendency to overfit the dataset, which limits their generalizability to different scenarios [54]. Regularization is used to overcome the overfitting tendency of the models. The corresponding problem can be solved using unconstrained optimization methods such as gradient descent and coordinate descent (CD). However, in practice, these methods do not scale well. To alleviate this problem, we present a coordinate majorization descent (CMD) based algorithm for solving *RegCox* which is more efficient and scalable than the regular CD solver.

$$\begin{aligned}
 L(\beta) &= n^{-1} \sum_{i=1}^K -X_i\beta + \log\left(\sum_{m \in R_i} \exp(X_m\beta)\right) \\
 L'_j(\beta) &= n^{-1} \sum_{i=1}^K \left\{-X(i, j) + \frac{\sum_{m \in R_i} X(m, j)\exp(X_m\beta)}{\sum_{m \in R_i} \exp(X_m\beta)}\right\}
 \end{aligned} \tag{6.2}$$

In this section, we present the *RegCox* framework which is a generic regularized Cox regression framework which can use any standard regularizer such as the elastic net, kernel

elastic net [57] etc. We consider solving *RegCox* here with the specific instance of the elastic net regularization. In Eq. (6.2), $L(\beta)$ is the partial log-likelihood loss function in Cox regression and $L'_j(\beta)$ is the gradient of log-likelihood with respect to the j^{th} attribute. $G(\beta)$ is the composite function consisting of the log-likelihood and regularization term.

$$G(\beta) = L(\beta) + \sum_{j=1}^m \lambda(\alpha|\beta_j| + \frac{1}{2}(1-\alpha)\beta_j^2) \quad (6.3)$$

$$G(\beta_j) = L(\beta_j, k \neq j) + \lambda(\alpha|\beta_j| + \frac{1}{2}(1-\alpha)\beta_j^2)$$

To apply the CMD optimization, we define the objective function $G(\beta_j)$ in Eq. (6.3) for fixed λ , α and β_k . The majorization minimization principle [71] is applied here and instead of minimizing $G(\beta_j)$ in Eq. (6.3) an update of β_j is found such that the univariate function $G(\beta_j)$ is decreased. To write this updating formula for β_j some additional notation is defined using D_j in Eq. (6.4).

$$D_j = \sum_{i=1}^K \frac{1}{4n} \{ \max_{m \in R_i} (X(m, j)) - \min_{m \in R_i} (X(m, j)) \}^2 \quad (6.4)$$

$$\beta_j^{new} = \frac{S(D_j \beta_j - L'_j(\beta), \lambda \alpha)}{D_j + \lambda(1-\alpha)}$$

$$S(z, t) = (|z| - t)_+ \text{sign}(z)$$

In Eq. (6.2), the formulae for computing the j^{th} component of the log-likelihood gradient vector is provided. We use this notation to represent this gradient ($L'_j(\beta) = \frac{\partial}{\partial \beta} L_j(\beta)$). R_i represents the risk set at time point i . K represents the number of unique failure times. λ is the regularization parameter and α is the elastic net parameter ($0 < \alpha < 1$). $S(z, t)$ is the soft-thresholding function. The equation for estimating the regression coefficient vector β^{new} in *RegCox* using coordinate majorization descent (CMD) optimization is also provided.

In Algorithm 6.1, the regression coefficient vector for the j^{th} coordinate is estimated by keeping all other coordinate values fixed. The regularization parameter λ is determined through cross validation. The EN-COX is another instance of *RegCox* which we consider

in our ARC framework. LASSO-COX [21] can be considered as a special case of the elastic net regularizer for the value of α set to 1.

The third regularized Cox regression algorithm we consider in *RegCox* is the kernel elastic net Cox regression (KEN-COX). Kernel elastic net Cox regression supplements EN-COX [16] with a column wise kernel matrix information. A RBF kernel matrix (Ke) is computed over the features (columns) of the dataset, and this information is plugged into the elastic net regularizer. The formulation is provided in Eq. (6.5). In this formulation, we use a notation where $X(:, i)$ represents the i^{th} column vector of the matrix X. Finally, the fourth regularized Cox regression algorithm we consider in the ARC framework is the Laplacian net COX (LAPNET-COX) algorithm.

KEN-COX and LAPNET-COX can be solved by using the CMD procedure used for solving *RegCox*. The only modification required in Algorithm 6.1 is modifying the denominator in the equation for estimating β_j^{new} . The details and algorithm for solving KEN-COX are provided in [57]. The algorithm for the Laplacian net Cox (LAPNET-COX) algorithm can be found in [43].

Algorithm 6.1: Regularized Cox Regression (RegCox)

```

1 Input: Training Feature Vectors  $X$ , Censored variable  $\delta$ , Time-to-event  $T$ ,
   Regularization parameter  $\lambda$ 
2 Output: Regression coefficient vector  $\beta$ 
3 Initialize  $\beta$ ;
4 for  $iter=1:1:max$  do
5   Compute  $L(\beta)$ ,  $G(\beta)$  from  $X$ ,  $T$ ,  $\lambda$  and  $\alpha$  using Eq. (6.2) and Eq. (6.3);
6   for  $j = 1, \dots, m$  do
7     Set the objective function  $G(\beta_j)$  and apply the CMD procedure;
8     Compute the updating factor  $D_j$  for computing  $\beta_j^{new}$  using Eq. (6.4);
9      $\beta_j^{new} = \frac{S(D_j \beta_j - L'_j(\beta), \lambda \alpha)}{D_j + \lambda(1 - \alpha)}$ ;
10  end
11  Update  $\beta = \beta^{new}$ ;
12 end

```

$$\beta = \arg \min_{\beta} L(\beta) + \lambda(\alpha \|\beta\|_1) + \lambda(1 - \alpha)\beta^T Ke\beta \quad (6.5)$$

$$Ke(i, j) = \exp\left(\frac{-\|X(:, i) - X(:, j)\|_2^2}{2\sigma^2}\right)$$

6.3.2 Model Discriminative Gradient-Based Sampling

In this section, we explain the model discriminative gradient-based sampling strategy used by *RegCox* in ARC. In general regression problems, solving for the optimal parameter β which can minimize the empirical error is a widely used search approach. In this approach, the parameters are repeatedly updated according to the negative gradient of the loss $L(\beta)$ with respect to each training example (X_i, T_i, δ_i) . The equation for obtaining β is provided in Eq. (6.6). In this equation, α is called the learning rate.

$$\beta = \beta - \alpha \frac{\partial L_{X^+}(\beta)}{\partial \beta} \quad (6.6)$$

In active learning, model change is estimated after adding a new example X^+ to the training data with censored status δ^+ and time-to-event value T^+ . The empirical risk on the enlarged training set $D^+ = D \cup (X^+, T^+, \delta^+)$ is defined using Eq. (6.7).

$$C(X^+) = \alpha \frac{\partial L_{X^+}(\beta)}{\partial \beta} \quad (6.7)$$

The goal of our sampling strategy in active learning is then to choose the example that could maximally change the current model and this selection function can be formulated as

$$X^* = \arg \max_{X^+ \in pool} \|C(X^+)\| \quad (6.8)$$

However, in practice we do not know the true label (time-to-event) (T^+) of the sampled data point X^+ in advance. Therefore, we are not able to estimate the model change directly. Instead, the expected change is calculated over all possible K unique time-to-event labels from $\{T_1, T_2, \dots, T_K\}$ to approximate the true change.

$$X^* = \arg \max_{X \in pool} \sum_{k=1}^K h(T_k|X) \left\| \frac{\partial L_X(\beta)}{\partial \beta} \right\| \quad (6.9)$$

The impact of adding an instance X from the pool to the training data is calculated in Eq. (6.9). The absolute value of the gradient of the loss function with respect to the instance is weighted by the hazard probability $h(T_k|X)$ for that instance. This value is accumulated over all unique time-to-event values to obtain an estimate of the impact of X on the model. Finally, the instance X^* which can induce the maximum model change over all the instances in the pool is selected and assumed to be the *most discriminative* instance for active learning.

6.3.3 Proposed ARC Algorithm

Algorithm 6.2: ARC Algorithm

```

1 Input: Training Set  $Train$ , Unlabelled pool  $Pool$ , Time-to-event  $T$ , Censored
   status  $\delta$ , Active learning rounds  $max$ 
2 Output: Final labelled set  $Train$ 
3 for  $p=1:1:max$  do
4    $Model = RegCox(Train, \delta, T)$ ;
5   for each instance in  $Pool$  do
6     | Use model discriminative gradient sampling for each instance in  $Pool$ ;
7   end
8    $X^* = \arg \max_{X \in pool} \sum_{k=1}^K h(T_k|X) \left\| \frac{\partial L_X(\beta)}{\partial \beta} \right\|$ ;
9   Query oracle for label (time-to-event) of  $X^*$ ;
10   $Train \leftarrow Train \cup X^*$ ;
11   $Pool \leftarrow Pool \setminus X^*$ ;
12 end

```

In Algorithm 6.2, the basic ARC framework is explained. In line 4, the *RegCox* model is built using the training data and time-to-event values. In lines 5-7, the model is applied to all the instances in the unlabelled pool where Eq. (6.9) is applied. In line 8, the instance which makes the highest impact on the model is selected and the time-to-event label for this instance is requested. Finally, in lines 10, 11 the training data is updated to build the model at the end of the current active learning round.

Convergence and Complexity of ARC: The coordinate majorization descent (CMD) method mentioned earlier is used in *RegCox* and it is known to converge efficiently [71] which guarantees the convergence of ARC. However, convergence rates may vary with the kind of regularizer used. The time complexity of Cox regression is $O(mK)$ where m is the number

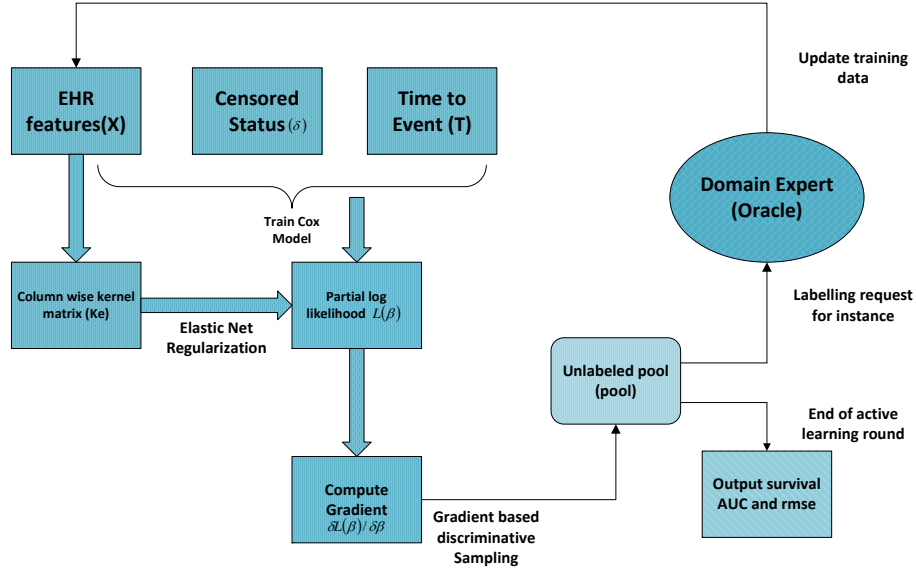


Figure 6.3: Block diagram of the active learning framework with KEN-COX regression.

of columns, K is the number of unique time-to-event values. The complexity of ARC can be computed as $O(nmK + nK)$ where n is the number of instances. The additional nK term here is because of the model discriminative gradient sampling step which is applied on the pool of unlabeled instances.

6.3.4 Flow Diagram of ARC

In Figure 6.3, ARC (KEN) combines KEN-COX with the model discriminative gradient-based sampling strategy. In the ARC (KEN) algorithm, a kernel matrix (Ke) is built on the features of the dataset and this is integrated with the log-likelihood function of Cox regression. A kernel elastic net regularization is employed to avoid overfitting. Model discriminative gradient-based sampling is then performed using the trained KEN-COX model and the instances available in the unlabeled pool to select the instance to be labelled by the end user/expert. The survival AUC and MSE values are output at the end of each active learning round.

6.4 Experimental Results

In this section, we present the experimental results obtained after applying ARC on various diverse datasets. Several real and synthetic survival datasets are used along with electronic health records to assess the performance of ARC. The data processing is explained in the experimental setup subsection. We provide different results which assess the goodness of fit, discriminative ability and learning rates.

6.4.1 Datasets Description

In this section, we demonstrate the performance of ARC on the following datasets. In Table 6.2, we provide the details of the datasets considered for our experiments.

Table 6.2: Description of the datasets.

Dataset	# Instances	# Features	Censored(%)	# Events
PBC	418	17	61	109
Breast	686	8	56	270
Colon	888	13	50	364
whas1	481	14	51.7	180
whas500	500	22	43	162
DLBCL	240	7399	42.5	134
NSBCD	115	549	66.96	26
Lung	86	7129	72.1	24
EHR	789	183	54.3	297
Syn1	500	15	40	300
Syn2	500	50	40	300
Syn3	100	50	40	60

- *Survival datasets:* Breast, Primary biliary cirrhosis (PBC) and Colon are survival datasets which are used directly from the standard survival R package. PBC data is from the Mayo Clinic trial in primary biliary cirrhosis (PBC) of the liver conducted between 1974 and 1984. Breast cancer dataset is from the German Breast Cancer Study Group. Colon cancer dataset is obtained from the survival R package. Whas1, Whas500 are Worcester Heart Attack Study datasets and DLBCL, NSBCD and Lung are high-dimensional gene-expression survival datasets. These datasets have the time-to-event and censored attributes provided along with the covariate values.

These datasets can be accessed from⁶ and ⁷.

- *EHRs*: We consider electronic health records (EHRs) of heart failure diagnosed patients for our analysis. This dataset was obtained for patients diagnosed with primary heart failure from Henry Ford Health System, Detroit, Michigan, USA for a duration of 10 years. For pre-processing this data, we construct features for all the distinct lab variables. To tackle the problem of multiple lab values for the same patient, we represent each lab by a set of summary statistics and apply a logarithmic transformation on these values to normalize them.

Time-to-event (30-day readmission) values are calculated using the prior admission and discharge dates. Patients are *right censored* using the 30 day readmission study period. This implies that if the difference between the last known follow up date and the previous admission date for a patient exceeds 30 days without the onset of a heart failure readmission, then this patient is right censored.

We present a snapshot of the distribution of readmission probabilities over this EHR dataset. In Figure 6.4, the readmission probabilities are plotted over a small sample of the EHR dataset for 30, 60 and 90 day readmission for heart failure. EN-COX model was trained on 200 random instances from one of our EHR datasets and the predicted survival probability values were obtained on a validation sample of 1000 instances. This hazard probability plot can help the readers understand the readmission trends present in this EHR dataset.

- *Synthetic datasets*: We generate synthetic datasets by setting the pairwise correlation ρ between any pair of covariates to vary from -0.5 to 0.5. We generate the feature vectors using this correlation and a normal distribution $N(0, 1)$. Feature vectors of different dimensionality are generated to construct four synthetic datasets. For each of these synthetic datasets, the generated failure times T are calculated using a Weibull distribution with γ set to 1.5. The Weibull distribution is used here to generate positive

⁶<https://www.umass.edu/statdata/statdata/stat-survival.html>

⁷<http://user.it.uu.se/~liuya610/download.html>

responses (failure times) to suit the constraints of synthetic survival data. Censoring for each dataset was set randomly to achieve 40% censoring in each synthetic dataset.

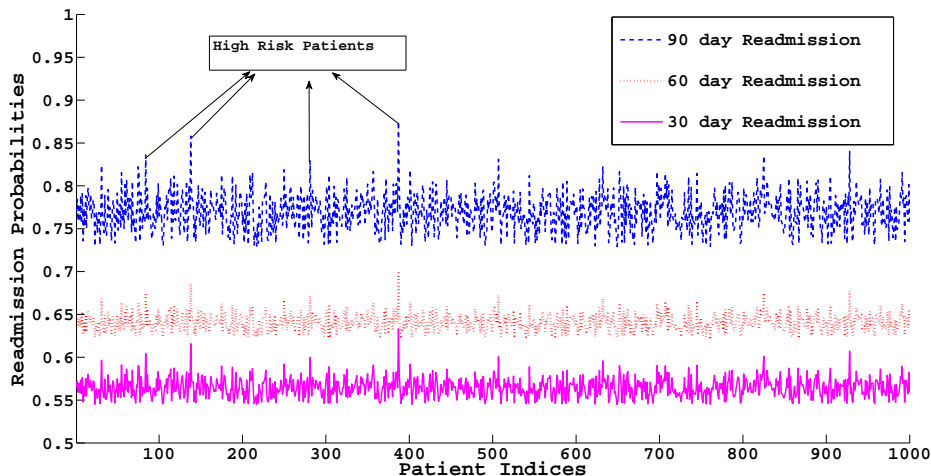


Figure 6.4: Readmission probabilities for patients computed within 30, 60 and 90 days post discharge from index hospitalization.

6.4.2 Implementation Details

The ARC framework is implemented in R using the CoxNet package. While presenting the experimental results here, we test ARC in an academic setting without the involvement of a real domain expert. The instances which are sampled through the model discriminative gradient-based sampling scheme in ARC are automatically assigned to their appropriate time-to-event labels by our program. We report the results obtained using five-fold cross validation where we use four folds to build the active learning model. We divide the data in these four folds into the labelled and unlabelled pool respectively and conduct the active learning process to obtain a model with best possible training instances. We then use the fifth fold as a hold-out set on which we report the Survival AUC and MSE metrics. This is done in a cyclic fashion to report the standard deviation values also. We employ a notation through the remaining part of this chapter to represent different active learning algorithms in ARC. ARC (LASSO) represents integrating LASSO-COX with active learning. Similarly ARC (KEN) and ARC (LAPNET) represent integrating KEN-COX and LAPNET-COX

with active learning respectively. KEN-COX uses an additional σ parameter in its RBF kernel which is set to 0.3 for all the experiments. The codes for the ARC algorithm are available here. ⁸

6.4.3 Goodness of Prediction

In Table 6.3, we provide the survival AUC and the standard deviation (std) values obtained through five-fold cross validation after running the ARC framework on several public and synthetic survival datasets.

Table 6.3: Comparison of Survival AUC (std) values in ARC w.r.t. different regularizers.

Dataset	ARC (LASSO)	ARC (EN)	ARC (KEN)	ARC (LAPNET)
PBC	0.809 (0.022)	0.807 (0.020)	0.806 (0.058)	0.796 (0.046)
Breast	0.663 (0.034)	0.649 (0.053)	0.676 (0.043)	0.676 (0.044)
Colon	0.673 (0.030)	0.661 (0.027)	0.683 (0.059)	0.719 (0.034)
whas1	0.806 (0.042)	0.796 (0.017)	0.816 (0.094)	0.792 (0.003)
whas500	0.806 (0.032)	0.817 (0.012)	0.795 (0.024)	0.771 (0.076)
DLBCL	0.544 (0.065)	0.623 (0.035)	0.649 (0.066)	0.611 (0.028)
NSBCD	0.718 (0.058)	0.719 (0.091)	0.650 (0.061)	0.693 (0.041)
EHR	0.664 (0.028)	0.679 (0.037)	0.691 (0.082)	0.688 (0.011)
Syn1	0.541 (0.027)	0.638 (0.032)	0.602 (0.078)	0.658 (0.047)
Syn2	0.844 (0.025)	0.873 (0.033)	0.893 (0.045)	0.914 (0.062)
Syn3	0.676 (0.014)	0.680 (0.078)	0.676 (0.098)	0.590 (0.097)

We compare the goodness of fit of ARC (LASSO), ARC (EN), ARC (KEN) and ARC (LAPNET). The Martingale Residuals based mean squared error is also calculated for the

⁸<https://github.com/MLSurvival/>

Cox-based models using Eq. (6.10).

$$MSE = \frac{\sum_{i=1}^n (\delta_i - (\exp(X_i^T \beta) h_0(T')))^2}{n} \quad (6.10)$$

The mean square error (MSE) and std values for the survival regression models are calculated using five-fold cross validation. The MSE is used to assess the goodness of fit obtained by the Cox regression model. These values are also provided in Table 6.4.

Table 6.4: Comparison of MSE (std) values of ARC w.r.t. different regularizers.

Dataset	ARC (LASSO)	ARC (EN)	ARC (KEN)	ARC (LAPNET)
PBC	0.338 (0.068)	0.280 (0.025)	0.278 (0.060)	0.293 (0.059)
Breast	0.374 (0.035)	0.374 (0.026)	0.320 (0.106)	0.375 (0.021)
Colon	0.405 (0.058)	0.387 (0.030)	0.376 (0.024)	0.396 (0.033)
whas1	0.429 (0.077)	0.432 (0.043)	0.429 (0.053)	0.427 (0.046)
whas500	0.381 (0.045)	0.368 (0.051)	0.375 (0.029)	0.366 (0.050)
DLBCL	0.334 (0.035)	0.279 (0.034)	0.246 (0.025)	0.261 (0.031)
NSBCD	0.277 (0.052)	0.235 (0.041)	0.229 (0.048)	0.242 (0.050)
EHR	0.455 (0.038)	0.414 (0.032)	0.426 (0.069)	0.448 (0.063)
Syn1	0.404 (0.080)	0.398 (0.072)	0.396 (0.065)	0.397 (0.054)
Syn2	0.375 (0.056)	0.330 (0.065)	0.350 (0.072)	0.301 (0.066)
Syn3	0.403 (0.138)	0.263 (0.031)	0.288 (0.083)	0.289 (0.044)

The results in Table 6.3 and Table 6.4 shows that for all regularizers used ARC performs competitively. This also shows that ARC is a regularizer independent framework and it can accommodate any kind of regularizer to learn an active learning model.

6.4.4 Comparison of Sampling Strategies

In Figure 6.5, the learning curves are plotted over **20** active learning rounds for **6** datasets and the x-axis represents the active learning rounds and the y-axis represents the concordance index (Survival AUC). Depending on the size of the dataset being considered, we set the sampling size for each round in batch mode active learning. For each dataset, we consider integrating LASSO-COX, EN-COX, KEN-COX with two non-censoring sampling methods which are random sampling and uncertainty sampling.

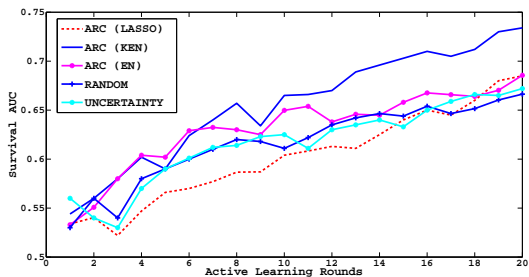
Random sampling selects instances to include in the active learning model at random. Uncertainty-based sampling selects those instances which the model is most uncertain about. We obtained these instances by building a naive bayesian classifier considering survival regression as a binary classification problem. Subsequently, those instances with low posterior probability difference margin were sampled and added to the model.

The learning curves in Figure 6.5 indicate that ARC (LASSO), ARC (EN), ARC (KEN) obtain models with good discriminative ability. The learning curves suggests that qualitative instances are being sampled from the pool and added to the training data in the active learning rounds. The results over all the datasets also show the effectiveness of ARC based sampling in comparison to uncertainty and random sampling.

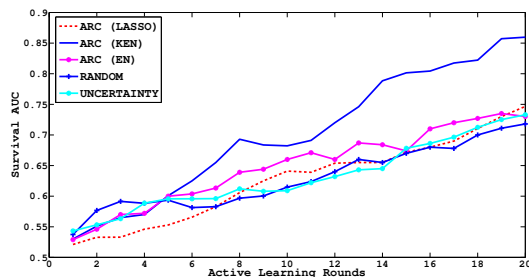
6.4.5 Importance of Censored Samples

In this section, we evaluate the contribution of the censored instances towards building the active learning model. This is important in order to understand how the model is capturing the censoredness of the instances while building the active learning model during successive iterations. We capture these statistics of the number of sampled instances with each of the four algorithms for the Breast and Colon survival datasets. The sampled instances are then divided into censored and event instances and these are plotted below.

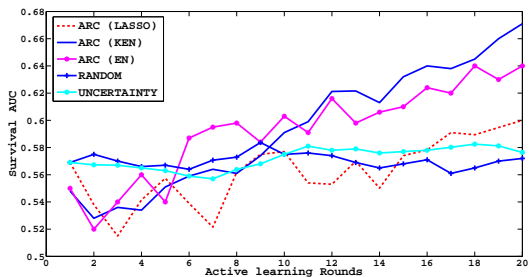
Figure 6.6, clearly demonstrates how the active learning model is sampling more number of censored instances than events as the number of rounds increases. This indicates that the model is trying to extract the censoredness which non-active learning methods cannot do.



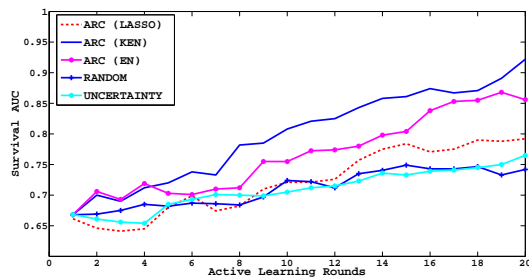
(a) Breast



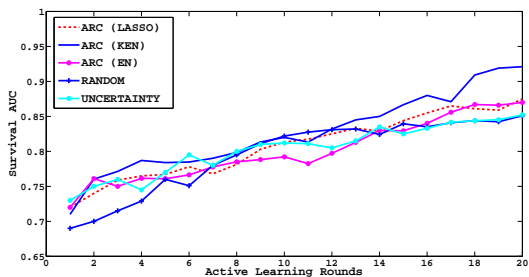
(b) Colon



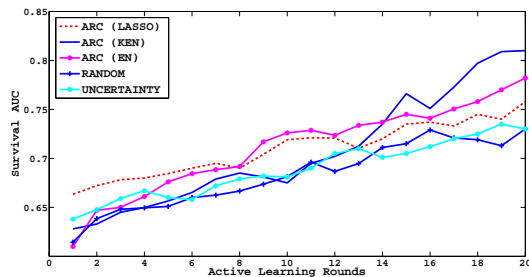
(c) EHR



(d) Syn1



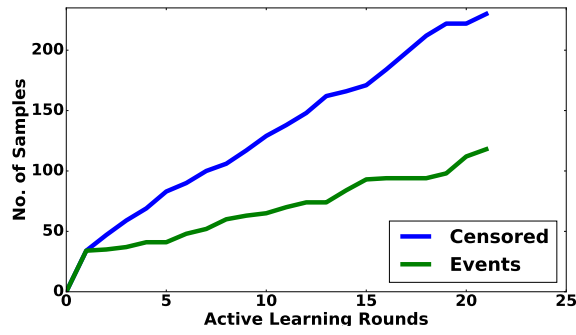
(e) Syn2



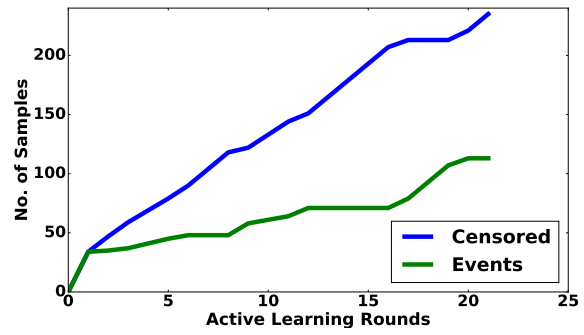
(f) Syn3

Figure 6.5: Comparison of the active learning rates of ARC with 4 different regularizers over real-world and synthetic datasets.

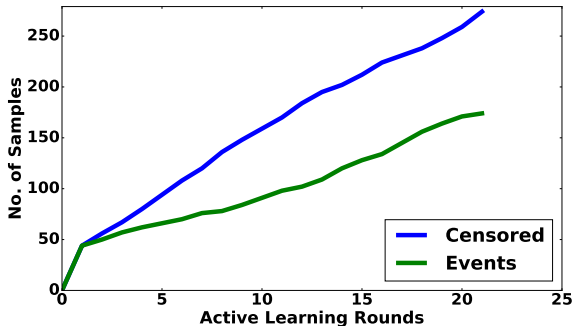
This justifies the importance of using active learning-based methods for right censored data.



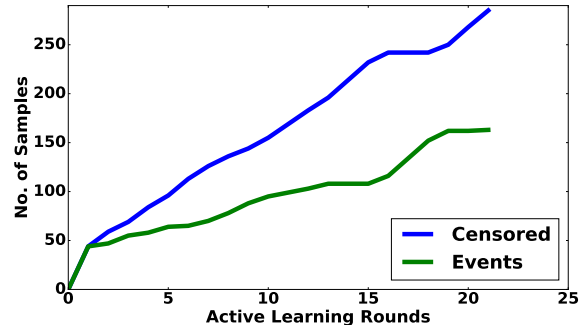
(a) Breast (ARC KEN)



(b) Breast (ARC LAPNET)



(c) Colon (ARC KEN)



(d) Colon (ARC LAPNET)

Figure 6.6: Censoredness plot for Breast and Colon datasets.

CHAPTER 7: CONCLUSIONS AND FUTURE WORK

In this chapter, we summarize the major contributions of the methods proposed in this dissertation, and we present ideas to extend each of these algorithms.

In Chapter 3, we proposed two different regularizers to capture intra-event correlation from survival data. These regularizers were integrated with the partial log-likelihood loss function of the Cox regression. The first regularizer we presented was intended to capture feature correlation among the attributes. This was called the Feature regularized (FEAR-COX) Cox regression. The second regularizer we used was the graph-based OSCAR regularizer which used the feature graph information present in the dataset to capture structured sparsity and we integrated it with Cox regression to obtain the OSCAR-COX algorithm. We compared the discriminative ability of these algorithms with respect to state-of-the-art regularized Cox regression models such as the fused-lasso, adaptive-lasso and laplacian net based Cox regression. We conducted feature analysis of the sparse set of features selected by these regularized Cox regression models. The results obtained indicate that our methods are effective at building better discriminative models, and the improvements affirm the importance of using novel regularizers with Cox regression. In addition, the sparse set of features obtained by our regularized Cox regression models can be useful for clinicians to assess important biomarkers during intervention studies conducted for readmission analytics. We can extend this work by building regularized Cox regression models with time-varying covariates which is generally observed in recurrent longitudinal patient data in hospitals. Another interesting direction would be to study the impact of these regularizers with respect to different hazard based survival models such as Weibull survival models.

In Chapter 4, we proposed a set of methods for performing survival analysis by calibrating the time-to-event labels for censored instances in the dataset. We motivated the necessity for this application by considering the two-dimensional correlation structure in censored data which needs to be inferred by a method before labelling these censored instances. These methods are very useful in several real-world scenarios such as (i) mining clinical

data to identify patient readmissions (ii) following projects in crowdfunding to determine their success. Traditional survival learners cannot be used directly for such data, since the time-to-event label information that is used for censored instances is incomplete. Erroneous time-to-event labels in such instances could misguide the learning algorithm which is undesirable. To overcome this problem, we introduce a transformation process which makes it easy for a domain expert to convert highly censored data to calibrated censored data which is more reliable for predictive analytics. We studied two methods in this paper, namely, Regularized Inverse Covariance-based Calibration (**REC**) and Transposable Regularized Inverse Covariance-based Calibration (**TREC**). **REC** uses a column-based regularization to account for intra-event correlation. **TREC** uses a composite row and column-based regularization to account for both inter-event and intra-event correlation. The experimental results reveal that both these methods help in improving the survival AUC of algorithms in comparison to other data imputation schemes. This work can be extended to interval-based censoring to identify methods to calibrate censored instances in that domain.

In Chapter 5, we proposed a novel solution for the problem of learning a linear regression model for data with right censoring. The uniqueness of our approach was that it can extract knowledge about the structure of the events and censored instances and induce this knowledge into the prediction model. This feature made our approach adaptable to variable proportions of censored instances and events in the data. The proposed SLIREC model was formulated as a bi-convex optimization problem and the block-coordinate descent method was used for solving it. We also used an approximation technique based on the proximal-Newton method in our computation to obtain orders of magnitude faster convergence. We evaluated the performance of Structured regularization linear regression model for censored data (SLIREC) using several diverse benchmark datasets consisting of high-dimensional gene expression measurements and electronic health records. Our experimental results demonstrated the efficiency of the structured regularization component of SLIREC compared to various linear regression and survival regression algorithms using metrics such as time-based

AUC, survival AUC and Brier score. We plan to extend this work by incorporating other matrix based regularizers such as the nuclear norm within the SLIREC framework for inferring the structure and assessing the improvement obtained.

In Chapter 6, we presented an active learning based method for building a survival model which assesses the importance of an instance before adding it to the model. This Active Regularized Cox regression (ARC) framework which integrates active learning with Cox regression using a novel model discriminative gradient based sampling strategy. This is useful in healthcare applications such as readmission risk prediction where in ARC can identify patient records to be labelled by a domain expert which can help in building survival models with expert feedback. In ARC, the domain expert provides a time-to-event label for the instance sampled by the model. This labelled instance is then added to the training data and the model is updated with the sampled set of instances at the end of each active learning round. We conducted several experiments to study the performance of ARC using four regularized Cox regression algorithms on various synthetic and public survival datasets. The results indicate that ARC is effective at building predictive models with good discriminative ability.

APPENDIX : LIST OF PUBLICATIONS

Journal Publications

1. Yan Li, Bhanukiran Vinzamuri and Chandan K. Reddy “Constrained Elastic Net based Knowledge Transfer for Healthcare Information Exchange.” Springer Data Mining and Knowledge Discovery Journal (DMKD), 29(4), pages 1094—1112, 2015.
2. Bhanukiran Vinzamuri, Yan Li and Chandan K. Reddy “Calibrated Survival Analysis using Regularized Inverse Covariance Estimation for Right Censored Data” *Journal Under Revision*.
3. Bhanukiran Vinzamuri and Chandan K. Reddy “ Feature Grouping-based Regression and its application for Survival Analysis”. *Journal Under Revision*

Conference Publications

1. Ping Wang, Karthik Padthe, Bhanukiran Vinzamuri and Chandan K. Reddy “ CRISP: Consensus Regularized Selection based Prediction” . Proceedings of ACM International Conference on Information and Knowledge Management (CIKM), Indianapolis, USA, 2016.
2. Yan Li, Bhanukiran Vinzamuri and Chandan K. Reddy “Regularized Weighted Linear Regression for High-dimensional Censored Data.” Proceedings of SIAM Conference on Data Mining (SDM), Miami, FL USA, 2016.
3. Bhanukiran Vinzamuri, Yan Li and Chandan K. Reddy “Active Learning based Survival Regression for Censored Data” . Proceedings of ACM Conference on Information and Knowledge Management (CIKM) Shanghai China, pages 241–250, 2014.
4. Vineeth Rakesh, Dilpreet Singh, Bhanukiran Vinzamuri and Chandan K. Reddy “Personalized Recommendation of Twitter Lists using Content and Network Information”

In Proceedings of the AAAI International Conference on Weblogs and Social Media (ICWSM), Michigan, USA, 2014.

5. Bhanukiran Vinzamuri and Chandan K. Reddy “Cox regression with correlation based regularization for electronic health records”. Proceedings of IEEE International Conference on Data Mining (ICDM), Dallas TX USA, pages 757–766, 2013.
6. Bhanukiran Vinzamuri, Jaegul Choo and Chandan K. Reddy “ Structured Regularization based Linear Regression for Right Censored Data”. *Submitted*
7. Bhanukiran Vinzamuri, Karthik Padthe and Chandan K. Reddy “Feature Grouping using Weighted ℓ_1 norm for High-Dimensional Data” *Submitted*.

Book Chapters

1. Chandan K. Reddy and Bhanukiran Vinzamuri “A Survey of Partitional and Hierarchical Clustering Algorithms.” Data Clustering: Algorithms and Applications (87), pages 87-110, 2013.

REFERENCES

- [1] John P Klein and Melvin L Moeschberger. *Survival analysis: techniques for censored and truncated data*. Springer Science & Business Media, 2005.
- [2] David W Hosmer Jr, Stanley Lemeshow, and Susanne May. *Applied survival analysis: Regression modelling of time to event data*, 2008.
- [3] Ching-Fan Chung, Peter Schmidt, and Ana D Witte. Survival analysis: A survey. *Journal of Quantitative Criminology*, 7(1):59–98, 1991.
- [4] H Koul, V v Susarla, and J Van Ryzin. Regression analysis with randomly right-censored data. *The Annals of Statistics*, pages 1276–1288, 1981.
- [5] Chandan K Reddy and Yan Li. A review of clinical prediction models. In *Healthcare Data Analytics*, pages 343–378. Chapman and Hall/CRC, 2015.
- [6] Peter Sasieni. Cox regression model. *Encyclopedia of Biostatistics*, 1999.
- [7] Eric Bair, Trevor Hastie, Debashis Paul, and Robert Tibshirani. Prediction by supervised principal components. *Journal of the American Statistical Association*, 101(473):119–137, 2012.
- [8] Wei Pan. A multiple imputation approach to cox regression with interval-censored data. *Biometrics*, 56(1):199–203, 2000.
- [9] Komal Kapoor, Mingxuan Sun, Jaideep Srivastava, and Tao Ye. A hazard based approach to user return time prediction. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1719–1728, 2014.
- [10] Jing Wang, Siamak Faridani, and Panagiotis Ipeirotis. Estimating the completion time of crowdsourced tasks using survival analysis models. *Crowdsourcing for search and data mining (CSDM)*, pages 31–38, 2011.
- [11] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.
- [12] Peter J Green. On use of the em for penalized likelihood estimation. *Journal of the Royal Statistical Society. Series B (Methodological)*, 52(3):443–452, 1990.

- [13] Nicolas Städler and Peter Bühlmann. Missing values: sparse inverse covariance estimation and an extension to sparse regression. *Statistics and Computing*, 22(1):219–235, 2012.
- [14] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(10):2287–2322, 2010.
- [15] Xiao-Li Meng and Donald B Rubin. Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80(2):267–278, 1993.
- [16] Noah Simon, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for coxs proportional hazards model via coordinate descent. *Journal of statistical software*, 39(5):1–13, 2011.
- [17] Wei Zhang, Takayo Ota, Viji Shridhar, Jeremy Chien, Baolin Wu, and Rui Kuang. Network-based survival analysis reveals subnetwork signatures for predicting outcomes of ovarian cancer treatment. *PLoS Comput Biol*, 9(3):1–16, 2013.
- [18] Shivapratap Gopakumar, Tu Dinh Nguyen, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Stabilizing sparse cox model using statistic and semantic structures in electronic medical records. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 331–343. Springer, 2015.
- [19] Kyu Ha Lee, Sounak Chakraborty, and Jianguo Sun. Survival prediction and variable selection with simultaneous shrinkage and grouping priors. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 8(2):114–127, 2015.
- [20] Hao Helen Zhang and Wenbin Lu. Adaptive lasso for cox’s proportional hazards model. *Biometrika*, 94(3):691–703, 2007.
- [21] Robert Tibshirani et al. The lasso method for variable selection in the cox model. *Statistics in medicine*, 16(4):385–395, 1997.
- [22] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC Press, 2015.

- [23] Daniela M Witten and Robert Tibshirani. Survival analysis with high-dimensional covariates. *Statistical methods in medical research*, 4:1–23, 2009.
- [24] John P Klein, Hans C Van Houwelingen, Joseph G Ibrahim, and Thomas H Scheike. *Handbook of survival analysis*. CRC Press, 2013.
- [25] Sijian Wang, Bin Nan, Ji Zhu, and David G Beer. Doubly penalized buckley–james method for survival data with high-dimensional covariates. *Biometrics*, 64(1):132–140, 2008.
- [26] Julian Wolfson, Sunayan Bandyopadhyay, Mohamed Elidrisi, Gabriela Vazquez-Benitez, David M Vock, Donald Musgrove, Gediminas Adomavicius, Paul E Johnson, and Patrick J O’Connor. A naive bayes machine learning approach to risk prediction using censored, time-to-event data. *Statistics in medicine*, 34(21):2941–2957, 2015.
- [27] Andreas Mayr and Matthias Schmid. Boosting the concordance index for survival data—a unified framework to derive and evaluate biomarker combinations. *PloS one*, 9(1):834–843, 2014.
- [28] Hemant Ishwaran, Udaya B Kogalur, Eugene H Blackstone, and Michael S Lauer. Random survival forests. *The annals of applied statistics*, 2(3):841–860, 2008.
- [29] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.
- [30] Liang Sun, Shuiwang Ji, and Jieping Ye. Canonical correlation analysis for multilabel classification: A least-squares formulation, extensions, and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):194–200, 2011.
- [31] Shuiwang Ji, Lei Tang, Shipeng Yu, and Jieping Ye. Extracting shared subspace for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 381–389, 2008.
- [32] Chandan K Reddy and Charu C Aggarwal. *Healthcare data analytics*, volume 36. CRC Press, 2015.

- [33] Debprakash Patnaik, Patrick Butler, Naren Ramakrishnan, Laxmi Parida, Benjamin J Keller, and David A Hanauer. Experiences with mining temporal event sequences from electronic medical records: initial successes and some challenges. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 360–368, 2011.
- [34] Jimeng Sun, Fei Wang, Jianying Hu, and Shahram Edabollahi. Supervised patient similarity measure of heterogeneous patient records. *ACM SIGKDD Explorations Newsletter*, 14(1):16–24, 2012.
- [35] Pranjul Yadav, Michael Steinbach, Vipin Kumar, and Gyorgy Simon. Mining electronic health records (ehr): A survey. 2015.
- [36] Wenjiang J Fu. Penalized regressions: the bridge versus the lasso. *Journal of computational and graphical statistics*, 7(3):397–416, 1998.
- [37] Howard D Bondell and Brian J Reich. Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with oscar. *Biometrics*, 64(1):115–123, 2008.
- [38] Xi Chen, Weike Pan, James T Kwok, and Jaime G Carbonell. Accelerated gradient method for multi-task sparse learning problem. In *Proceedings of Ninth IEEE International Conference on Data Mining*, pages 746–751. IEEE, 2009.
- [39] Jieping Ye and Jun Liu. Sparse methods for biomedical data. *ACM SIGKDD Explorations Newsletter*, 14(1):4–15, 2012.
- [40] Sen Yang, Lei Yuan, Ying-Cheng Lai, Xiaotong Shen, Peter Wonka, and Jieping Ye. Feature grouping and selection over an undirected graph. In *Graph Embedding for Pattern Analysis*, pages 27–43. Springer, 2013.
- [41] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.
- [42] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical*

- association*, 101(476):1418–1429, 2006.
- [43] Hokeun Sun, Wei Lin, Rui Feng, and Hongzhe Li. Network-regularized high-dimensional cox regression for analysis of genomic data. *Statistica Sinica*, 24(3):14–33, 2014.
- [44] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.
- [45] Adrian F Hernandez, Melissa A Greiner, Gregg C Fonarow, Bradley G Hammill, Paul A Heidenreich, Clyde W Yancy, Eric D Peterson, and Lesley H Curtis. Relationship between early physician follow-up and 30-day readmission among medicare beneficiaries hospitalized for heart failure. *Journal of American Medical Association (JAMA)*, 303(17):1716–1722, 2010.
- [46] Marlow B Hernandez, Randall S Schwartz, Craig R Asher, Elsy V Navas, Victor Totfalusi, Ivan Buitrago, Ankush Lahoti, and Gian M Novaro. Predictors of 30-day readmission in patients hospitalized with decompensated heart failure. *Clinical cardiology*, 36(9):542–547, 2013.
- [47] David E Lanfear, Edward L Peterson, Janis Campbell, Hemant Phatak, David Wu, Karen Wells, John A Spertus, and L Keoki Williams. Relation of worsened renal function during hospitalization for heart failure to long-term outcomes and rehospitalization. *The American journal of cardiology*, 107(1):74–78, 2011.
- [48] Yan Li, Bhanukiran Vinzamuri, and Chandan K Reddy. Constrained elastic net based knowledge transfer for healthcare information exchange. *Data Mining and Knowledge Discovery*, 29(4):1094–1112, 2015.
- [49] Wessel N Van Wieringen, David Kun, Regina Hampel, and Anne-Laure Boulesteix. Survival prediction using gene expression data: a review and comparison. *Computational statistics & data analysis*, 53(5):1590–1603, 2009.
- [50] Joseph S Ross, Gregory K Mulvey, Brett Stauffer, Vishnu Patlolla, Susannah M Bernheim, Patricia S Keenan, and Harlan M Krumholz. Statistical models and patient

- predictors of readmission for heart failure: a systematic review. *Archives of internal medicine*, 168(13):1371–1386, 2008.
- [51] Badri Padhukasahasram, Chandan K Reddy, Yan Li, and David E Lanfear. Joint impact of clinical and behavioral variables on the risk of unplanned readmission and death after a heart failure hospitalization. *PloS one*, 10(6):1–11, 2015.
- [52] Yan Li, Vineeth Rakesh, and Chandan K Reddy. Project success prediction in crowdfunding environments. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 247–256, 2016.
- [53] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- [54] ACC Coolen and L Holmberg. Principles of survival analysis, 2013.
- [55] Genevera I Allen and Robert Tibshirani. Transposable regularized covariance models with an application to missing data imputation. *The Annals of Applied Statistics*, 4(2):764–790, 2010.
- [56] Vineeth Rakesh, Jaegul Choo, and Chandan K Reddy. Project recommendation using heterogeneous traits in crowdfunding. In *Ninth International AAAI Conference on Web and Social Media*, pages 337–346, 2015.
- [57] Bhanukiran Vinzamuri and Chandan K Reddy. Cox regression with correlation based regularization for electronic health records. In *Proceedings of the IEEE 13th International Conference on Data Mining*, pages 757–766, 2013.
- [58] Jonathan Buckley and Ian James. Linear regression with censored data. *Biometrika*, 66(3):429–436, 1979.
- [59] LJ Wei. The accelerated failure time model: a useful alternative to the cox regression model in survival analysis. *Statistics in medicine*, 11(14-15):1871–1879, 1992.
- [60] Cheryl L Faucett, Nathaniel Schenker, and Jeremy MG Taylor. Survival analysis using auxiliary variables via multiple imputation, with application to aids clinical trial data.

- Biometrics*, 58(1):37–47, 2002.
- [61] Seyoung Kim and Eric P Xing. Tree-guided group lasso for multi-response regression with structured sparsity, with an application to eqtl mapping. *The Annals of Applied Statistics*, 6(3):1095–1117, 2012.
- [62] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Structured sparsity through convex optimization. *Statistical Science*, 27(4):450–468, 2012.
- [63] Jason Lee, Yuekai Sun, and Michael Saunders. Proximal newton-type methods for convex optimization. In *Advances in Neural Information Processing Systems*, pages 836–844, 2012.
- [64] Chien-Fu Jeff Wu. Jackknife, bootstrap and other resampling methods in regression analysis. *the Annals of Statistics*, pages 1261–1295, 1986.
- [65] Per Kragh Andersen and Maja Pohar Perme. Pseudo-observations in survival analysis. *Statistical methods in medical research*, 4:1–29, 2009.
- [66] Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009.
- [67] Harald Binder. Coxboost: Cox models by likelihood based boosting for a single survival endpoint or competing risks. *R package version*, 1, 2013.
- [68] Devan Kansagara, Honora Englander, Amanda Salanitro, David Kagen, Cecelia Theobald, Michele Freeman, and Sunil Kripalani. Risk prediction models for hospital readmission: a systematic review. *Journal of American Medical Association (JAMA)*, 306(15):1688–1698, 2011.
- [69] Elia Biganzoli, Patrizia Boracchi, and Ettore Marubini. A general framework for neural network models on censored survival data. *Neural Networks*, 15(2):209–218, 2002.
- [70] Faisal M Khan and Valentina Bayer Zubek. Support vector regression for censored data (svrc): a novel tool for survival analysis. In *Proceedings of IEEE International Conference on Data Mining*, pages 863–868, 2008.
- [71] Kenneth Lange, David R Hunter, and Ilsoon Yang. Optimization transfer using surro-

gate objective functions. *Journal of computational and graphical statistics*, 9(1):1–20, 2000.

ABSTRACT**NOVEL MACHINE LEARNING METHODS FOR MODELING
TIME-TO-EVENT DATA**

by

Bhanukiran Vinzamuri**August 2016****Advisor:** Dr. Chandan K. Reddy**Major:** Computer Science**Degree:** Doctor of Philosophy

Predicting time-to-event from longitudinal data where different events occur at different time points is an extremely important problem in several domains such as healthcare, economics, social networks and seismology, to name a few. A unique challenge in this problem involves building predictive models from right censored data (also called as survival data). This is a phenomenon where instances whose event of interest are not yet observed within a given observation time window and are considered to be right censored. Effective models for predicting time-to-event labels from such right censored data with good accuracy can have a significant impact in these domains. However, existing methods in the literature cannot capture various complexities present in real-world survival data such as feature groups and intra and inter-event correlations. To address such challenges, we briefly summarize the major contributions of the methods proposed here as (i) modeling intra-event correlations in survival data using structured sparsity-based regularizers, (ii) learning novel representations for survival data by inferring inter-event and intra-event correlations, (iii) extending linear regression-based methods to learn predictive models from right censored data and (iv) identifying censored instances and events from the data which are contributing extensively to learning a model with lesser number of training instances using active learning. We present

optimization-based algorithms corresponding to each of the aforementioned contributions in this dissertation utilizing diverse techniques such as regularization, representation learning and active learning. Our methods are tested on different real-world longitudinal datasets such as electronic health records (EHRs), crowdfunding data, gene-expression data and several publicly available synthetic survival datasets. The results demonstrate the goodness of these methods when compared to state-of-the-art survival analysis, classification and regression methods from the literature.

AUTOBIOGRAPHICAL STATEMENT

Bhanukiran Vinzamuri completed his B. Tech and Masters by Research in Computer Science from International Institute of Information Technology, Hyderabad (IIIT-H) in India in 2011. He enrolled in the PhD in Computer Science program at Wayne State University in August 2012. He has completed an internship at Mayo Clinic in 2015. His primary research interests include data mining, machine learning, biostatistics and healthcare informatics. He has published papers at top-tier conferences such as IEEE ICDM, ACM CIKM, SIAM SDM and journals such as Springer DMKD. He has also served as a co-reviewer for more than 50 conference and journal papers in Computer Science.