Reframing Cox Proportional Hazards Model for Big Data and Neural Networks

Aliasghar (Arash) Tarkhan

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:

Noah Simon, Chair

Ali Shojaie

Alex Luedtke

Program Authorized to Offer Degree:
Biostatistics

University of Washington

Abstract

Reframing Cox Proportional Hazards Model for Big Data and Neural Networks

Aliasghar (Arash) Tarkhan

Chair of the Supervisory Committee:
Professor Noah Simon
Department of Biostatistics

In many medical and biomedical applications, we measure the outcome as a "time-to-event" (e.g., disease progression or death). The aim of this dissertation is to propose frameworks for survival analysis and prediction with survival data that include many observations, ultra-high dimensional features, or images. We propose frameworks that are computationally efficient and stable and are amenable to stochastic-based optimization algorithms. Our proposed frameworks scale up to extremely large datasets that do not fit into memory.

The aim of survival analysis is to assess the connection between the characteristics of a patient and the time-to-event outcome. To do this, it is common to assume a proportional hazards model and fit a proportional hazards regression (or Cox regression). A log-concave objective function known as the "partial likelihood" is maximized to fit the Cox proportional hazards model. For moderate-sized datasets, an efficient Newton-Raphson algorithm that leverages the structure of the objective function can be employed. In large datasets with lots of observations, this approach has two issues. First, the computational tricks that leverage the structure of the objective function can lead to computational instability. Second, the objective function does not naturally decouple over observations. Thus, the model can be computationally expensive to fit if the dataset does not fit into memory. In Chapter 2, we propose a novel modified framework for proportional hazards regression to address these

issues. The proposed framework results in an objective function amenable to stochastic gradient descent. We show that this simple modification allows us to efficiently fit survival models with extremely large datasets, including lots of observations. Our proposed framework facilitates training complex, e.g., neural-network-based models with survival data. We propose a straightforward neural network architecture for survival prediction.

The standard Cox model is known to behave poorly (the estimated coefficients may go to infinity) when the number of features is greater than the number of observations or even when the number of observations is greater than but close to the number of features. One way to handle this issue is to use regularization such that we have well-behaved solutions. The standard approaches (such as the lasso) for modifying Cox proportional hazards regression tend to fail for large-scale or ultra-high dimensional datasets because of computational instability and memory limits. In Chapter 3, we extend our proposed modification of the partial likelihood from Chapter 2 to the penalized partial likelihood to address these issues. In particular, our proposed framework enables data to be read off the hard drive in chunks to update our model sequentially. Therefore, our proposed modification facilitates fitting the penalized Cox model on larger datasets. We apply stochastic proximal gradient descent (SPGD) in our framework to fit Cox regression models with a convex combination of l_1 (lasso) and l_2 (ridge regression) penalties, also known as the *elastic-net* penalty.

In Chapter 4 of the dissertation, we tackle a problem that is a bit different from the survival analysis we discussed above. In computational pathology, training deep neural networks using giga-pixel whole-slide images (WSIs) is a challenging task. The fundamental challenge is the absence of annotation at the patch (instance) level because of the high cost and the time-consuming nature of hand labeling. This challenge is typically mitigated by pooling instances that rely on only the slide-level labels. For example, we see this with typical

weakly supervised learning methods, MIL, and attention-based MIL. A WSI typically has hundreds of thousands of image patches, each of which may carry different information about the slide label/class. Training a deep neural network with thousands of image patches per slide is computationally expensive. We propose an adaptive sampling strategy that aims to save computation by adaptively selecting a subset of highly predictive instances. We also study other sampling strategies that aim to reduce computation and compare them with our proposed strategy. Although we proposed the adaptive sampling strategy for the classification problem, this idea is quite general and can easily apply to survival prediction.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I want to express my warmest gratitude to my adviser, Professor Noah Simon, for the invaluable knowledge and support during my studies. Graduate school could be difficult and time-consuming. But when I reflect on my time at graduate school, I realize how fortunate I was to have worked with him. Without his vision, inspiration, trust, and patience, I could not complete this work. He provided an excellent example of scholars and mentors I hope to pursue in my future career.

I would like to express my sincere gratitude to my committee members, Professors Ali Shojaie, Alex Luedtke, Thomas Bengtsson, and Archis Ghate. It was an honor for me to have them on my dissertation committee. They are acknowledged professionals in my field of study and gave me excellent guidance and insights. Their suggestions and contributions strengthened my dissertation.

I would like to thank the Genentech company for supporting part of my dissertation research. I sincerely thank my mentors and coworkers, Dr. Thomas Bengtsson, Dr. Jian Dai, Dr. Trung Kien Nguyen, and Dr. Anil Yuce, for their mentorship, support, and guidance. It was a tremendous honor to know all of them and have a chance t learn from them. With their assistance, I broadened my research to more areas, such as oncology and histopathology.

I would like to sincerely thank the faculties, staff, students, and coworkers in the Biostatistics department at the University of Washington for their financial support of this research. Without my mentors, colleagues, and family, I could not have completed my degree successfully.

I would like to thank all of my friends, those whom I knew from Iran, and those with

whom I made friends here in the united states, and more specifically in Seattle and at the University of Washington. With them, I had a happier stay far from my family and finish graduate school smoother.

Finally, I would like to thank my wonderful family in Iran for their endless love and support. My parents and siblings were always there for me.

DEDICATION

to my dear parents and siblings.

Chapter 1

INTRODUCTION

It is common in biological contexts to investigate the relationship between patients' characteristics and time to an event of interest (such as disease progression, recovery, or death) [1]. Survival analysis aims to study such relationships [2]. Patients may leave the study prior to experiencing the event of interest, resulting in often-partial patient information in such situations. Cox proportional hazards model has been widely used to take this partial information into account [3]. Cox model maximizes a log-concave function known as the "partial likelihood" to estimate parameters.

With the growth of electronic medical records, biomedical datasets comprising a large number of observations are becoming more common [4]. Common methods for fitting the Cox model use the sequential structure of the partial likelihood to speed up computation [5, 6]. This may lead to computational instability. In addition, the partial likelihood does not decouple naturally across individuals or groups of individuals. Thus, if the dataset is too large to fit into memory, fitting the model might become computationally expensive. This also implies that the objective function based on the partial likelihood is not amenable to stochastic gradient-based optimization techniques [7]. This decoupling issue makes naïve approaches to fitting complex models, e.g., neural-networks, impractical for time-to-event data. Cox model may become computationally expensive or may perform poorly when handling large survival data, especially those including imaging data.

The standard Cox model performs poorly when the number of features is greater than the number of observations or even when the numbers are close to each other. There were some

efforts in the literature to resolve this issue and improve the computational aspects of Cox model [8, 9, 10, 5, 10]. Unfortunately, the optimization procedure used by these methods (and more generally those that engage with the full partial likelihood) requires that the entire dataset be read into memory before the model is fit. This creates problems when engaging with ultra-high dimensional datasets [11, 12]. These issues become even more severe when working with giga-pixel images commonly used in oncology applications, e.g., Whole-Slide Images (WSIs) [13, 14].

Automated visual inspection of sampled tissues through high-resolution Whole-Slide Images (WSIs) from biopsies has become the gold standard for diagnosing various diseases in oncology [13, 14]. However, WSIs are giga-pixel images and directly training the deep neural networks with them is infeasible because of the memory constraint. One immediate solution is to train such networks by sampling smaller regions of WSIs, also known as patches or tiles. But the challenge is the lack of pixel-level annotation and that the labels are only available at the WSI (patient) level. Multiple instance learning (MIL) [15, 16] and its attention-based version [17], as typical weakly supervised learning methods, have been proposed to tackle this challenge [15, 16]. But the remaining challenge is that not all the hundreds of thousands of tiles within a WSI are predictive of the outcome (e.g., disease outcome). This makes training of deep neural networks for tasks such as classification and survival prediction time-consuming and computationally expensive when using WSIs [18, 19].

In this dissertation, we propose a re-framed Cox proportional hazards model for big data and neural networks. In Chapter 2, we propose a modified framework for the Cox model that is amenable to optimization via stochastic-gradient based algorithms. In Chapter 3, we extend our proposed framework for the penalized Cox model to fit the Cox model with the number of features more than the number of observations. In Chapter 4, we propose an adaptive sampling strategy for training deep multiple instance learning neural networks and compare it with different sampling strategies. Although we proposed our sampling strategy

for the classification problems, it is easily extendable to survival prediction via training deep MIL neural networks.

Chapter 2

REFRAMING COX MODEL FOR BIG DATA AND NEURAL NETWORKS

2.1 Introduction

It is commonly of interest in biomedical settings to characterize the relationship between characteristics of an individual and their risk of experiencing an event of interest (e.g., progression of a disease, recovery, death, etc; see [1]). Outcomes of this type are known as "time-to-event" outcomes, and characterizing such relationships is known as survival analysis [2]. In such applications, we often only have partial information on patients due to censoring (e.g., they might leave the study before experiencing the event of interest).

Cox proportional hazards regression model (hereafter, simply called Cox model) [3] is the most common tool for conducting survival analyses. The Cox model assumes a particular semi-parametric relationship between the risk at each given time of experiencing an event and the features of a patient (e.g. age, sex, treatment assignment, etc). To estimate parameters in this model, the Cox model maximizes a log-concave function known as the "partial likelihood". Once estimated, this model can provide a person-specific hazard of experiencing an event (as a function of their features). Such predictions are often used in personalized medicine, e.g., in the development of prognostic and predictive biomarkers [20]. To maximize the partial likelihood, it is most common to use a second-order algorithm such as Newton-Raphson [21] for datasets with few features (with potentially many observations). Traditionally, Cox model has been used on data-sets with relatively few features, though penalized extensions have been developed for high-dimensional applications [5].

It is increasingly common to have biomedical datasets with a large number of observations, especially with increasing use of electronic medical records [4]. Although the Cox model

has been widely used for small-to-moderate numbers of observations, common methodologies for fitting the Cox model faces issues in datasets with many observations. In particular, in fitting the Cox model, it is common to leverage the sequential structure of the partial likelihood to vastly speed up computation (from $O(n^2)$ to $O(n)$; see [5]). However, when there are a large number of observations, this can lead to computational instability (which we illustrate in this chapter).

The second issue is that the partial likelihood does not naturally decouple over individuals or subsets of individuals. Thus, if the dataset does not fit into memory, the model can be very computationally expensive to fit: Standard distributed optimization methods such as those based on the alternating direction method of multipliers [22] cannot directly be used. This additionally means that the objective is not directly amenable to stochastic gradient-based optimization methods [7]. Unfortunately, to fit more complex neural network-based models, it is most common to use stochastic-gradient-based optimization [23]. This decoupling issue makes it impossible, or at least very impractical, to fit neural-network-based models with time-to-event data.

In this chapter, we propose a simple framework for conducting survival analysis using the Cox model [11, 12]. We built our framework upon an objective function that is a modification of the usual partial likelihood function. In particular, this modified objective function decouples over subsets of observations, and allows us to employ stochastic-gradient-based algorithms that engage only a subset of our data at each iteration. We show the parameters estimated by this new objective function are equivalent to the original parameters when the assumptions of the Cox model hold (and may actually be more robust with model mis-specification). In addition, our new objective function is amenable to optimization via stochastic-gradient-based algorithms that are computationally efficient and stable for this objective: This approach can easily scale to datasets that are too large to fit into memory. We further discuss how our new framework can be implemented with both streaming [24] and non-streaming datasets. We discuss extending our framework to use mini-batches and present some recommendations for improved performance in practice. We also extend our

methodology to produce asymptotically valid confidence intervals. We additionally illustrate how our methodology can be used to fit neural-network-based models with time-to-event outcome.

2.2 Related work

Although there are fully-parametric (e.g., [25]) and non-parametric (e.g., [26]) approaches for survival analysis, we focus on those that are based on the Cox model (as those are most relevant to our proposal). Others have proposals for fitting linear Cox models with large data [21, 27, 28]; however, those approaches are not amenable to stochastic optimization, which is crucial for engaging with complex prediction methods such as those with neural networks (or for applications with data stored in a distributed manner). Authors in [29] presented SGD-based algorithms for a variety of applications, including the Cox model. However, their algorithm suffers from two issues: It cannot accommodate streaming data, and in fact requires $\sim n^2$ computation (which we are aiming to avoid with our proposal). Authors in [30] proposed directly maximizing the concordance index [31]. While this is an interesting predictive target, it moves us away from generative parameters in the Cox model. As this objective function is discontinuous, confidence intervals may be difficult to obtain. Authors in [32, 33] connected neural networks to the log-partial likelihood. However, they engaged with [non-stochastic] gradient descent, which, as we will discuss in this chapter, is not easily amenable to very large and/or distributed datasets. Another set of work converts the survival prediction problem into a classification problem by dividing the continuous time-to-event into non-overlapping intervals [34, 35, 36]. This is an interesting line of work which we give empirical comparisons to in Section 2.6.2: However, the need to choose a grid for discretization is a potential downside. The work of [37] is most closely related to ours: As with [32] and [33], they connect neural networks with the partial likelihood; however, they note that a stochastic gradient-like optimization method will be needed to scale to large datasets. They propose a heuristic for an "approximate gradient". They do not justify the heuristic (and in fact, their stochastic gradient is not unbiased for their objective function, so

there is no guarantee, based on previous work, that any of the results of SGD-based methods will hold). Our re-framing of proportional hazards modeling generalizes and justifies their heuristic — it both identifies why it should work and proves that it will. Additionally, our reformulation can be combined in a straightforward way with distributed computing tools [38]. See Appendix A.1 for more detail on this. Our contributions in this chapter are:

- We propose a new optimization framework based on a modified version of the full log-partial likelihood that is amenable to stochastic/online optimization, requires $O(n)$ computation, and is not susceptible to floating-point error.

- We prove that the parameter estimated by our framework is equivalent to the generative parameter in a correctly specified Cox model.

- We prove that for a linear Cox model, our proposed estimator is statistically rate optimal (that is, MSE converges at a rate of n^{-1}).

- We theoretically and empirically show that for a linear Cox model, our framework accommodates construction of asymptotically valid confidence intervals.

- We empirically show that our framework may be more computationally efficient and stable than the current gold standard implementation for fitting the Cox model when the dataset is large.

- We propose a simple framework that facilitates flexible survival modeling with neural networks.

2.3 Methodology

2.3.1 Cox Proportional Hazards Regression Model

The Cox model, proposed by [3], is a commonly used semi-parametric regression model for evaluating the association between the time until some event of interest and a set of variables

measured on a patient. More formally, suppose on each patient we measure an event time T, and a vector of numeric features $X = (X_1, \ldots, X_p)$. The Cox model engages with the so-called hazard function

$$h(x,t) = \frac{p(t|x)}{S(t|x)},$$

where $p(t|x) = \frac{d}{dt}P(T < t|X = x)$ and $S(t|x) = P(T > t|X = x)$. The hazard function, $h(x,t)$, can be thought of as the probability density of having an event at time t, given that a patient (with covariates x) has not had an event up to that time. The Cox model assumes a particular form for the hazard function,

$$h(t, \boldsymbol{x}; \boldsymbol{\beta}^*) = h_0(t) \exp\left(f_{\boldsymbol{\beta}^*}(\boldsymbol{x})\right), \tag{2.1}$$

where $f_{\boldsymbol{\beta}^*}$ is a specified function of parameters $\boldsymbol{\beta}^* = (\beta_1^*, \beta_2^*, \ldots, \beta_k^*)$ that determines the role played by $\boldsymbol{x}$ in the hazard; and $h_0(t)$ is a baseline hazard function (independent of covariates). Note that $f_{\boldsymbol{\beta}^*}(\boldsymbol{x})$ may be assumed to be of different forms in different applications: For instance, in many scenarios k is taken to be p, and the simple linear model $f_{\boldsymbol{\beta}}(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\beta} = \beta_1 x_1 + \cdots + \beta_p x_p$ is used. This model assumes that the manner in which a patient's covariates modulate their risk of experiencing an event is independent of time. In particular, it is encoded entirely in $f_{\boldsymbol{\beta}^*}$. This simplifies the estimation and interpretation of the predictive model.

Our aim is to use data to estimate $\boldsymbol{\beta}^*$. In particular, we will assume that we have a dataset with n independent observations drawn from model (2.1): $\mathcal{D}^{(n)} = \{\mathcal{D}_i = (y_i, \boldsymbol{x}^{(i)}) | i = 1, 2, \ldots, n\}$. For the moment we assume that there is no censoring (all event times are observed), and no ties (all event times are unique). Estimation is conducted using the log-partial-likelihood:

$$pl^{(n)}(\boldsymbol{\beta}|\mathcal{D}^{(n)}) = \log\left(\prod_{i=1}^{n} \frac{h(\boldsymbol{x}^{(i)};\boldsymbol{\beta})}{\sum_{j\in\mathcal{R}_i} h(\boldsymbol{x}^{(j)};\boldsymbol{\beta})}\right) = \sum_{i=1}^{n}\left(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)}) - \log\left(\sum_{j\in\mathcal{R}_i}\exp\left(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(j)})\right)\right)\right), \tag{2.2}$$

where $\mathcal{R}_i = \{j \,|\, t_j \geq t_i\}$ is the "risk set for patient i". Note that aside from $\mathcal{R}_i$, the expression in (2.2) is independent of the event times. Extending this partial likelihood to

deal with censoring, left-truncation [39] and ties is quite straightforward (see Appendix A.2 for more details of this extension); however, for ease of exposition we do not include these details here.

2.3.2 Estimation by maximizing the log-partial-likelihood

Using the log-partial-likelihood from equation (2.2), an estimate of $\boldsymbol{\beta}^*$ can be obtained as

$$\hat{\boldsymbol{\beta}}^{(n)} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ -pl^{(n)}(\boldsymbol{\beta}|\mathcal{D}^{(n)}) \right\}. \tag{2.3}$$

When linear $f_{\boldsymbol{\beta}}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\beta} = \beta_1 x_1 + \cdots + \beta_p x_p$ is used, our objective function in (2.3) is convex in $\boldsymbol{\beta}$, and thus the tools of convex optimization can be applied to find $\hat{\boldsymbol{\beta}}^{(n)}$ (see *Corollary 1* in Appendix A.3 for the proof of convexity; included for completeness). In the current gold standard `survival` package in `R` [40], Newton-Raphson is used to minimize (2.3) with linear f. In the later sections, we will refer to this implementation as `coxph()`. For linear f, one can show that $\left\| \hat{\boldsymbol{\beta}}^{(n)} - \boldsymbol{\beta}^* \right\|_2^2 = O_p\left(n^{-1}\right)$ which is rate optimal (as is standard for estimation in parametric models; see [41]).

In current commonly used software implementations, the ordered structure of the loss (as well as the gradient, and hessian) are leveraged to improve computational efficiency. In particular, we examine the gradient,

$$\nabla_\beta \Big\{ -pl^{(n)}(\boldsymbol{\beta}|\mathcal{D}^{(n)}) \Big\} = -\sum_{i=1}^{n} \Big(\dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)}) - \frac{\sum_{j \in \mathcal{R}_i} \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(j)}) \exp\left(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(j)})\right)}{\sum_{j \in \mathcal{R}_i} \exp\left(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(j)})\right)} \Big), \tag{2.4}$$

where $\dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}) = \nabla_\beta \{ f_{\boldsymbol{\beta}}(\boldsymbol{x}) \}$ is the gradient of $f_{\boldsymbol{\beta}}(\boldsymbol{x})$ with respect to $\boldsymbol{\beta}$. While a naïve calculation would have $O(n^2)$ computational complexity because of the nested summations, this is not necessary. In the case that the times are ordered $t_1 < t_2 < \ldots < t_n$, we see that $R_i = R_{i+1} \cup \{i\}$. This allows us to use cumulative sums and differences to calculate the entire gradient in $O(n)$ computational complexity, with a single $O(n \log(n))$ complexity sort required at the beginning of the algorithm [5]. In discussing computational complexity, we assume that elementary operations (e.g. addition, subtraction, multiplication, and exponentiation) are $O(1)$. We also assume p is fixed and so, e.g., $x_i^\top \beta = O(1)$. If we imagined an

asymptotic regime with growing p, then our $O(n)$ statement on the computational cost for computing the gradient (once the data are sorted) should be modified to $O(np)$. The same updating trick mentioned above for the gradient can be used for calculating the Hessian. Unfortunately, however, when employing this strategy, the algorithm becomes susceptible to round-off issues, especially with a larger number of observations, n, and features, p, as seen in Section 2.6.1. Appendix A.12 discusses round-off error issues in more detail in standard implementations of log-partial-likelihood maximization e.g., `coxph()`.

Additional inspection of the gradient in (2.4) shows why stochastic-gradient-based methods cannot be used to decouple gradient calculations over observations in the sample: While the gradient can be written as a sum over indices $i = 1, \ldots, n$, the denominator for the $i = 1$ term involves all observations in the dataset. In the next section, we propose a novel simple modification of (2.3) that admits an efficient stochastic-gradient-based algorithm for estimating $\boldsymbol{\beta}^*$.

2.3.3 Estimation using SGD: BigSurvSGD

We begin by reformulating our problem. We consider a population parameter $\boldsymbol{\beta}^{(s)}$, defined as the population minimizer of the expected partial likelihood of s random patients (which we will refer to as "strata of size s"):

$$\boldsymbol{\beta}^{(s)} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \Big\{ \mathbb{E}_s[-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})] \Big\}. \tag{2.5}$$

Here, we think of $\mathcal{D}^{(s)}$ as a draw of s random patients from the population. Note that the minimum value for s is 2; otherwise, expression (2.4) (i.e., the gradient) becomes zero for all $\boldsymbol{\beta}$. By including a superscript s in $\boldsymbol{\beta}^{(s)}$, we note that this parameter may depend on s. In fact, when the assumptions of the Cox model hold (2.1), then $\boldsymbol{\beta}^{(s)} = \boldsymbol{\beta}^*$ for all s. The proof of this is quite simple, with details given in Appendix A.3.

To estimate $\boldsymbol{\beta}^*$, we select a small fixed s ($s \ll n$) and directly apply stochastic gradient descent to the population optimization problem (2.5). In practice, this will amount to calculating stochastic gradients using random strata of size s. One may note that for s

small, there are $\binom{n}{s}$ such strata to choose. However, results for stochastic gradient descent indicate that under strong convexity of (2.5), on the order of only m steps (strata) should be required to obtain a rate optimal estimator (converging at a rate of m^{-1} in MSE). See Appendix A.3 for the proof of strong convexity of (2.5) and the convergence rate $O(m^{-1})$.

In Section 2.6.1, we empirically investigate the effect of strata size s on the efficiency of our estimator. Based on these results, it seems choosing $s = 20$ results in negligible efficiency loss. See Appendix A.13 for more recommendations on how to choose strata size s.

Pairwise concordance ($s = 2$)

An interesting special case is when we choose strata of size $s = 2$, and look at pairs of patients. Then, in the case of no censoring, the population minimizer in (2.5), i.e., $\boldsymbol{\beta}^{(2)}$ maximizes the expectation of the pairwise log-partial likelihood

$$\begin{aligned} pl^{(2)}(\boldsymbol{\beta}|\mathcal{D}^{(2)}) = &\log\Big(\frac{\exp\left(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(1)})\right)}{\exp\left(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(1)})\right) + \exp\left(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(2)})\right)}\Big) 1(t_1 < t_2) \\ &+ \log\Big(\frac{\exp\left(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(2)})\right)}{\exp\left(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(1)})\right) + \exp\left(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(2)})\right)}\Big) 1(t_2 < t_1). \end{aligned} \tag{2.6}$$

This log-partial-likelihood can be thought of as a smoothed version of the standard concordance measure used in the concordance index [31]. Thus, even when the proportional hazards model does not hold, the parameter $\boldsymbol{\beta}^{(2)}$ maintains a useful interpretation as the population minimizer of the average smoothed concordance index. Also (2.6) is similar to the objective function for conditional logistic regression (CLR) with strata size $s = 2$ [42]. In the deep learning literature, neural network models with a conditional logistic outcome layers are often referred to as Siamese Neural Networks [43].

Our formulation has similarities to boosting [44] where weak learners are repeatedly trained on residuals from a current fitted model to improve performance, e.g., pairwise concordance in case of $s = 2$. Our approach is slightly different in that we do not "sketch" covariates, and we sub-sample observations (though, sub-sampling observations has increasingly become popular with boosting [45]). Our modified loss might fruitfully be engaged with boosting algorithm (using e.g., trees as base learners).

Optimization with SGD

Suppose that we have n_s independent (without overlap) strata, $D_1^{(s)}, \ldots, D_{n_s}^{(s)}$, each with s independent patients drawn from our population (with $s \geq 2$). For ease of notation, let I_m denote the indices of patients in strata $D_m^{(s)}$ for each $m \leq n_s$. For any $\boldsymbol{\beta}$ we have

$$\nabla_{\boldsymbol{\beta}} \mathbb{E}_s \left[pl^{(s)}(\boldsymbol{\beta} | \mathcal{D}_m^{(s)}) \right] = \mathbb{E}_s \left[\nabla_{\boldsymbol{\beta}} \left\{ pl^{(s)}(\boldsymbol{\beta} | \mathcal{D}_m^{(s)}) \right\} \right], \qquad \text{for all } m \leq n_s, \tag{2.7}$$

when $\boldsymbol{x}$ are drawn from a reasonable distribution (e.g. bounded); and $f_{\boldsymbol{\beta}}(\boldsymbol{x})$ is not too poorly behaved (e.g. Lipschitz). Note that expectation in (2.7) is taken w.r.t. the true joint distribution of outcomes and covariates. Here $\nabla_{\boldsymbol{\beta}} \left\{ pl^{(s)}(\boldsymbol{\beta} | \mathcal{D}_m^{(s)}) \right\}$ is defined analogously to (2.4) using $\mathcal{D}_m^{(s)}$

$$\nabla_\beta \Big\{ - pl^{(s)}(\boldsymbol{\beta} | \mathcal{D}_m^{(s)}) \Big\} = - \sum_{i \in I_m} \left(\dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)}) - \frac{\sum_{j \in \mathcal{R}_i^m} \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(j)}) \exp\left(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(j)}) \right)}{\sum_{j \in \mathcal{R}_i^m} \exp\left(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(j)}) \right)} \right), \tag{2.8}$$

where $R_i^m = \{ j \,|\, t_j \geq t_i \text{ and } i, j \in I_m \}$ are risk sets that include only patients in stratum m; and $\dot{f}_\beta$ denotes the gradient of f w.r.t. β. From here we can give the simplest version of our stochastic gradient descent (SGD) algorithm for (2.3). We choose an initial $\hat{\boldsymbol{\beta}}(0)$ (perhaps $= 0$), and at each iteration $m = 1, \ldots, n_s$, we update our estimate by

$$\hat{\boldsymbol{\beta}}(m) = \hat{\boldsymbol{\beta}}(m-1) + \gamma_m \times \nabla_\beta \left\{ pl^{(s)}(\hat{\boldsymbol{\beta}}(m-1) | \mathcal{D}_m^{(s)}) \right\}. \tag{2.9}$$

Here, γ_m is the learning rate (which should be specified in advance, or determined adaptively as discussed later in Section 2.3.3 and Appendix A.6). The computation time to run n_s steps of stochastic gradient descent according to (2.9) for linear $f_{\boldsymbol{\beta}}$ is about $sn_sp = np$ (where $n = sn_s$ is the total sample size). If ordering is not leveraged, then around nps computation is required. In contrast, Newton's algorithm for optimizing the full log-partial likelihood requires around np^2 computation per iteration when the round-off-error-prone updating rule is used (and around $n^2p + np^2$ if not). Additionally, with these small strata of size s, we are not prone to round-off issues when using stochastic optimization (because this sum is calculated separately for each small strata).

It has been shown that SGD algorithms for strongly convex objective functions are asymptotically more efficient if we use a running average of the iterates as the final estimate [46, 47]. Additionally in this case $\gamma_m = \gamma$ can be set to a fixed value (so long as it is sufficiently small). We denote the running average estimator by

$$\tilde{\boldsymbol{\beta}}(m) = \frac{1}{m}\sum_{i=1}^{m}\hat{\boldsymbol{\beta}}(i). \tag{2.10}$$

Note that the averaging process does not change the values of $\hat{\boldsymbol{\beta}}(m)$. In our simulations, we use the averaged $\tilde{\boldsymbol{\beta}}(m)$. Strong convexity of the objective in (2.5) depends on properties of $f_{\boldsymbol{\beta}}$ and (weakly) on the distribution of $\boldsymbol{x}$. For linear $f_{\boldsymbol{\beta}}$, and $\boldsymbol{x}$ with a non-degenerate distribution, this objective will be strongly convex (see Appendix A.4 for the proof of strong convexity of our objective function). In such cases, standard results [48] show that $\left\|\tilde{\boldsymbol{\beta}}(m) - \boldsymbol{\beta}^{(s)}\right\|_2^2 = O_p(m^{-1})$. As a reminder, this is the statistically optimal rate of convergence for estimating $\boldsymbol{\beta}^{(s)}$ (or equivalently, $\boldsymbol{\beta}^*$ when the Cox model assumptions holds) from m observations. See Appendix A.4 for the proof of this convergence rate $O(m^{-1})$ for averaging over iterates. When we use this averaged estimator, it has been shown that choosing the learning rate as $\gamma_m = \frac{C}{\sqrt{m}}$ (where C is a constant) gives us such an optimal convergence rate [49]. We choose this learning rate in our simulation studies in Section 2.6.1, where we tune the constant C to optimize the performance.

Streaming vs non-streaming

In the discussion above, we imagined that observations were arriving in a continual stream of strata, and that we were more concerned with the cost of computation than the cost of data collection. The algorithm we described engaged with each stratum only once (in calculating a single stochastic gradient). Algorithm 2 in Appendix A.6 details an implementation of a streaming algorithm in this imagined scenario. In practice, we generally have a fixed (though potentially large) number of observations (non-streaming), n, that are not naturally partitioned into strata. To employ SGD here, we randomly partition our observations into n_s disjoint strata of size s, and then carry out our updates in (2.9). In our work, we call such

a pass over a single random partition of the data, an epoch. In practice, we may want to take more than one epoch, and use mini-batches of multiple strata to improve performance. Algorithm 3 in Appendix A.6 details an implementation that takes multiple passes over the data and use mini-batches (we discuss additional bells and whistles in Appendix A.5).

In practice, we may also want to modify the learning rate over iterates to improve the performance of the SGD algorithm. We found using the `AMSGrad` [50] algorithm for adaptively selecting the learning rate using moments, using average over iterates as we discussed in Section 2.3.3), and taking ~ 100 epochs over the data generally results in strongest performance (see Appendix A.11.1).

2.4 Equivalence to U-statistic based optimization

While in this chapter we discuss obtaining an estimator by directly attempting to minimize the population objective function (2.5) using SGD, there is a corresponding empirical minimization problem. In particular, for strata of size s, one could define an estimator $\hat{\beta}^s$ by

$$\hat{\boldsymbol{\beta}}^{(s)} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \Big\{ - \sum_{\mathcal{D}^{(s)} \subset \mathcal{D}^{(n)}} pl^{(s)} \Big(\boldsymbol{\beta} | \mathcal{D}^{(s)} \Big) \Big\} \tag{2.11}$$

As in a standard U-statistic [51], this sum is taken over all subsets of s patients out of our original n patients (resulting in $\binom{n}{s} \sim n^s$ terms). This appears to be a difficult optimization problem, given the enormous number of terms. However, our approach shows that, in fact, only $\sim n$ of those terms need be considered to get a "good enough" approximate minimizer (and can be chosen randomly): The majority of terms contain redundant information. In fact, one could see this directly by noting that the objective function in (2.11) decouples over subsets. An application of stochastic gradient descent here would involve sampling strata with replacement (no indpendendent assumption among different subsets are needed). Assuming this empirical objective function is strongly convex, with averaging over iterates, would converge to a tolerance of $1/m$ after m steps. This approach, based on *incomplete U-statistics* [52] could be taken more generally for losses defined by U-statistics.

2.5 Inference

For linear $f_\beta(x)$, we can leverage the formulation of our estimator from Section 2.4 to construct confidence intervals. There are two approaches toward this end. The first approach uses large sample properties of U-statistic minimizers [53, 54]. In particular, these estimators are asymptotically linear [53], with an influence function based on their Hoeffding projection [51] which we can use to estimate the variance. The second approach is based on a non-parametric bootstrap which is discussed and justified by the multiplier U-process [55]. In practice, we use the non-parametric bootstrap to estimate the standard error of our coefficient estimates; and combine that with asymptotic normality to construct the confidence interval. Appendix A.7 contains details of the proposed implementation for calculating standard errors and confidence intervals, covering both theory and methods.

2.6 Results

2.6.1 Simulated experiments

We generate an event time that follows the Cox model (2.1) with simple linear $f_{\boldsymbol{\beta}}$ detailed below. We generate the baseline hazard $h_0(t)$ using an exponential distribution with parameters $\lambda = 1$. We generate the censoring and event times independently. The details of the data simulation procedure are given below [56]

$$
\begin{aligned}
&x_i \sim Uniform(-\sqrt{3}, \sqrt{3}) \text{ (unit-variance variable)},\\
&y_i \sim \exp(\mu = e^{-\boldsymbol{x}_i^T \boldsymbol{\beta^*}}) \text{ (time to event/censoring)}, \quad \boldsymbol{\beta^*} = \beta^* \mathbf{1}_{P\times 1}\\
&\delta_i \sim Bernoulli(p = 1 - p_c),\ p_c = Pr(t_i > c_i)
\end{aligned} \tag{2.12}
$$

where $y_i = min(t_i, c_i)$, i.e., time to event or censoring whichever comes first. Here p_c, the probability of censoring, is a parameter we can tune. Note that $\mu = e^{-\boldsymbol{X}^T \boldsymbol{\beta^*}}$ is the scale parameter (and mean) of the exponential distribution. This implies that larger $\boldsymbol{x}^T \boldsymbol{\beta^*}$ generate smaller survival times. Although this is not written in the form of (2.1), it is still consistent with the Cox proportional hazards model assumptions, with $f_{\boldsymbol{\beta^*}}(\boldsymbol{x}) = \boldsymbol{\beta^*}^\top \boldsymbol{x}$. In

all comparisons, we include the performance of `coxph()` the gold standard `R` implementation of Newton's algorithm for maximizing the partial likelihood. Additionally, in all comparisons, we normalize all covariates to have variance 1.

Small data results: We first evaluate the statistical efficiency of our estimation procedure (using strata sizes of less than n). We choose $\beta^* = 1$ and evaluate mean-squared error (MSE) between estimators $\tilde{\beta}$ (i.e., average over iterates), and $\hat{\beta}^{(n)}$ (i.e., `coxph()`) and the truth, β^* over 1000 simulated datasets with up to 100 epochs run. The top left panel in Figure 2.1 illustrates MSE of $\tilde{\beta}$ with an adaptive learning rate (using `AMSGrad`) for different strata sizes s. Although the convergence rate is still $\sim n^{-1}$ for all strata sizes, we see that for small strata sizes (e.g., $s = 2$), there is some statistical inefficiency (in the constant multiplying the rate) when the model is correctly specified. However, there is nearly no statistical inefficiency for the larger strata sizes (strata sizes larger than $s = 20$) as compared to the full partial likelihood.

We next evaluate the performance of averaged SGD with a fixed learning rate, against averaged SGD with an adaptive learning rate (using `AMSGrad`) with a fixed strata-size of $s = 20$ for both. In addition, we try various numbers of epochs (from 10 to 100). The top right panel in Figure 2.1 shows performance over 1000 simulated datasets. We see that with enough epochs (around 100) both approches perform well. However, `AMSGrad` nearly reaches that performance with as few as 50 epochs, whereas using a fixed learning rate does not attain that performance with fewer than 100 epochs. For both of these methods, we tuned our (initial) learning-rate to be empirically optimal in these experiments. Hereafter, we use averaged SGD with `AMSGrad` and we simply refer to it as `BigSurvSGD`.

In Section 2.3.3, we discussed the potential computational instability of `coxph()` due to its updating formulae in small-to-moderate sized datasets and how our framework can avoid such instability. Here we empirically verify those claims. The bottom left and bottom right panels in Figure 2.1 compare the MSE and concordance index of `coxph()` and `AveAMSGrad` algorithms for small-to-moderate sample sizes (n) and varying number of features (p) over 1000 simulated datasets. As we see, `coxph()` performs poorly for larger p and n. For instance,

as we can see in the bottom left panel in Figure 2.1, `coxph()` with $(p = 50, n = 1000)$ performs worse than $(p = 50, n = 100)$. This is because of computational instability with the recursive updating formulas used by `coxph()` for larger sample sizes and numbers of features. One important aspect of these examples is that we include a large amount of signal (which increases as the number of features increases). With less signal, this instability is less pronounced unless very large sample sizes are used.

Big Data Results: We next consider the numerical stability of our framework (versus directly maximizing the full partial likelihood using Newton's algorithm). We generated 100 datasets with $\beta^* = 1$ and we used 100 epochs for the `AveAMSGrad` algorithm. The simulations were conducted on a quad-core Intel Core i7-2600 with 12 GB RAM. Figure 2.2 shows a surprising and unfortunate result for `coxph()`: We see that as sample size increases drastically, the performance of `coxph()` starts getting worse! In particular, for $p = 20$, `coxph()` is basically producing nonsense by the time we get to $1,000$ observations for this simulation setup. This indicates that for large datasets the current gold standard may be inadequate, though we do note that there is a large amount of signal in these simulations (more than we might often see in practice). In contrast, `BigSurvSGD` has no such issues and gives very good performance. As a reminder, the statistical performance of the output of `coxph()`, due to floating-point issues, degenerates much earlier. We want to be clear that we do not believe these issues are due to sloppiness in the `coxph()` function (the `survival` package is superbly done), but rather an issue with trying to optimize the full-partial likelihood. Furthermore, `coxph()` fails for the medium-to-large datasets as it is poorly equipped to deal with datasets that do not easily fit in memory (`R` unfortunately generally deals somewhat poorly with memory management). The right hand side of vertical blue line (with number of observations greater than $\sim 3 \times 10^6$) shows the area where `coxph()` fails due to out-of-memory issue with the system specifications we used. Our proposed algorithm scales up well with very large dataset: It reads the data in chunks from the hard-drive (allowing us to engage with datasets difficult to fit in memory) and it continuously shows improvement by increasing sample size. Note that by big data here, we mean lots

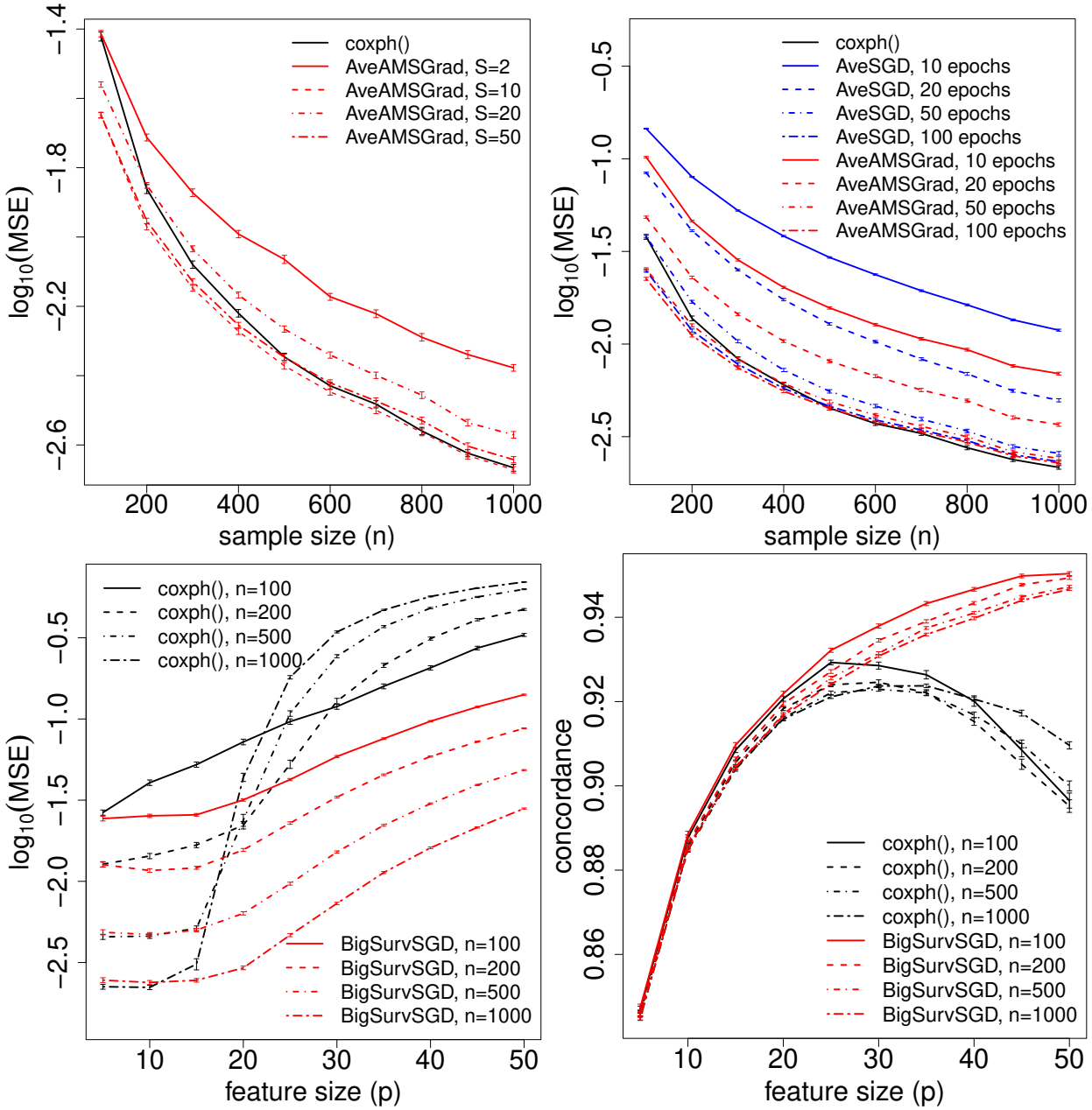

Figure 2.1: (top left) $\log_{10}$(MSE) of estimates with `coxph()` and `AveAMSGrad` for $p = 10$ features, 100 epochs, and different strata sizes (S); (top right) $\log_{10}$(MSE) of estimates with averaged SGD (`AveSGD`), averaged `AMSGrad` (`AveAMSGrad`), and `coxph()` for strata size $s = 20$ and $p = 10$ features; (bottom left) $\log_{10}$(MSE)Figure and (bottom right) concordance index with `BigSurvSGD` (`AveAMSGrad`) and `coxph()` for strata size $s = 20$, different feature sizes p, and different sample sizes n. For all results, we choose mini-batch size $K = 1$, probability of censoring $p_c = 0.2$, and the optimal value C (0.12 for `AveAMSGrad` and 1.5 for `AveSGD`) for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

of observations, i.e. larger n. See Appendix A.14 for a discussion on how to deal with the high-dimensional data, i.e., larger p.

Inference: We next examine how the coverage of the 95% CIs generated from `coxph()`, and our bootstrap and plugin approaches behave as a function of sample size (n) and feature size (p). We choose strata size $s = 20$, $\beta^* = 1$, and we allow our algorithm to run for 100 epochs to estimate $\tilde{\boldsymbol{\beta}}$. For the bootstrap approach, we consider $B = 1000$ bootstrap resamples and we allow each resample to run for 100 epochs to get the bootstrapped estimates $\tilde{\boldsymbol{\beta}}^b$, $b = 1, 2, \ldots, B = 1000$. For the plugin method, we choose $n_o = 1000$ sample strata per observation to calculate the standard error of our estimate $\tilde{\boldsymbol{\beta}}$. The left panel in Figure 2.3 presents the coverage from different approaches for $p = 10$ features and varying sample sizes over 100 simulated datasets. We observe all methods perform well for the small-to-medium sample sizes but `coxph()` fails for the large sample sizes (e.g., $n = 10^5$) due to its computational instability. The right panel in Figure 2.3 presents the coverage from different approaches for more features $p = 20$ (higher signal) and varying sample sizes over 100 simulated datasets. We observe both our proposed plug-in and bootstrap approaches give coverage very close to the nominal 95%. `coxph()` fails for all ranges of sample size; it behaves worse for larger sample sizes.

Rare events: We examine the effect of censoring rates, particularly when they are high (i.e., rare events). We choose $p = 10$ uniformly-distributed covariates with $\boldsymbol{\beta}^* = \beta^* \times (1, 1, 0, ..., 0)$ where we separately consider $\beta^* = 0.1$, 0.5, 1. For our algorithm we choose strata size $s = 20$, and 100 epochs. For the bootstrap approach, we construct confidence intervals with standard errors estimated using $B = 100$ resamples and we allow each resample to run for 10 epochs (starting at our original estimate) to get the bootstrapped estimates $\tilde{\boldsymbol{\beta}}^b$, $b = 1, 2, \ldots, B = 100$. For the plugin method, we choose $n_o = 100$ sample strata per observation to calculate the standard error of our estimate $\tilde{\boldsymbol{\beta}}$. Figure 2.4 presents the MSE and coverage from different approaches for sample size $n = 1000$ and varying rates of censoring p_c over 100 simulated datasets. We observe that proposed approaches and `coxph()` perform closely. All approaches perform better when there is more signal, and worse when

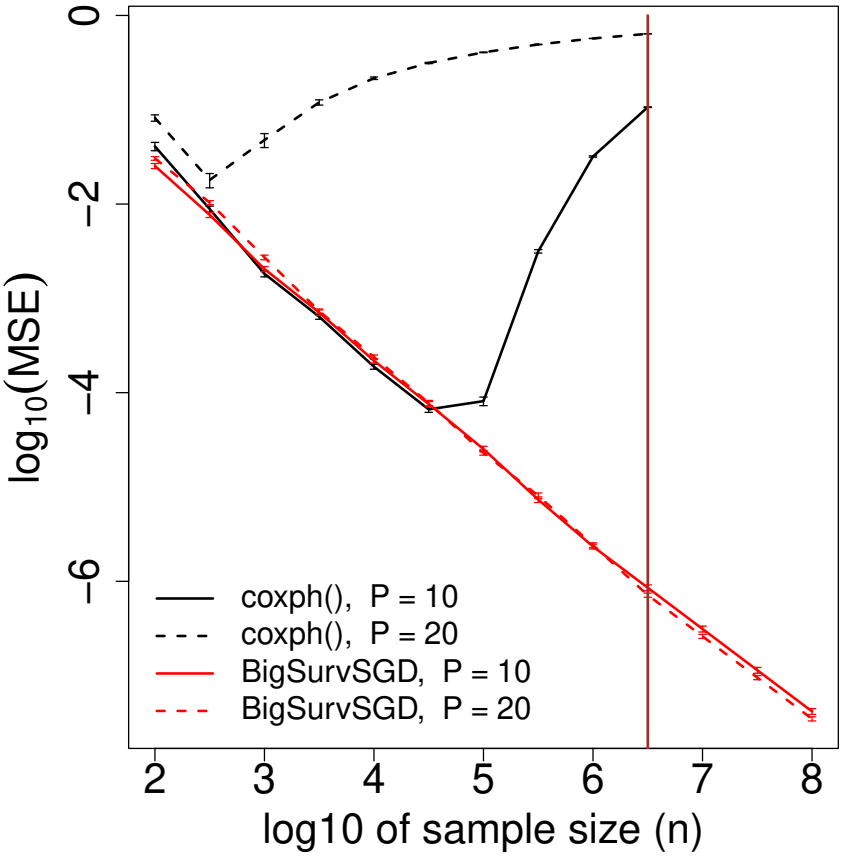

Figure 2.2:]

$\log_{10}$(MSE) with `BigSurvSD` (`AveAMSGrad`) and `coxph()` for different sample sizes (n) and feature sizes (p). We choose mini-batch size $K = 1$, number of epochs 100, probability of censoring $p_c = 0.2$, and the optimal value $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$. The grey area illustrates regions where the code returns out-of-memory error.

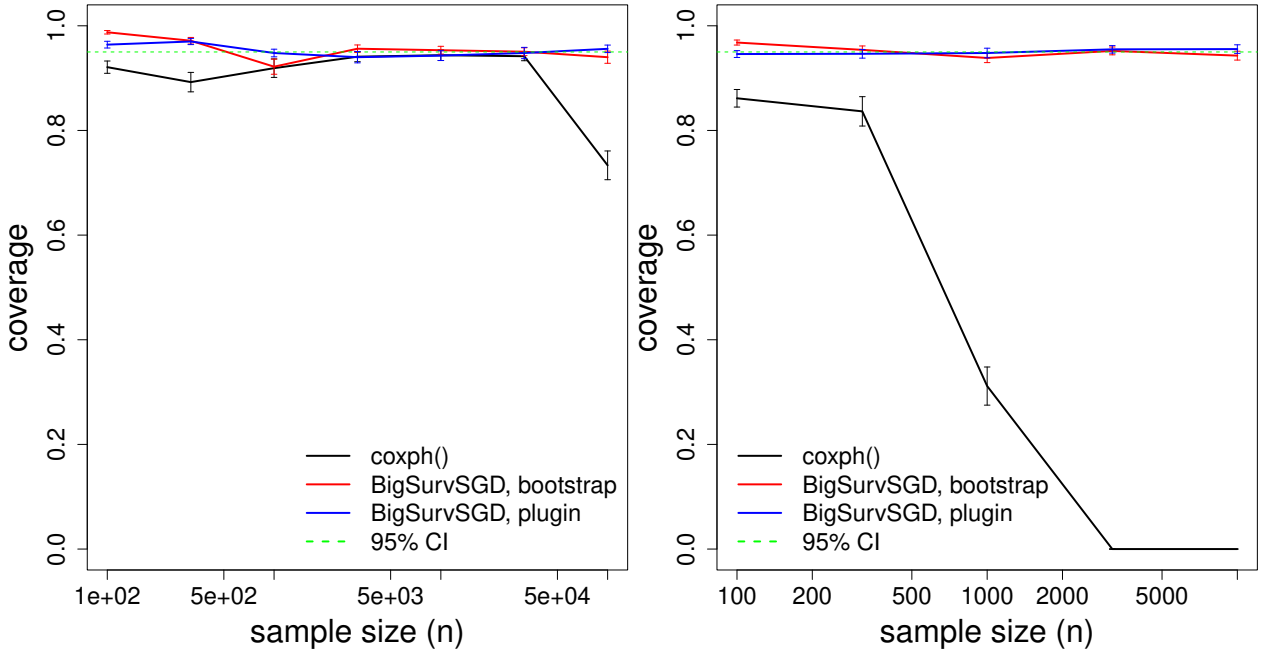

Figure 2.3: (left) coverage for $p = 10$ features; (right) coverage for $p = 20$ features. For both results, we choose mini-batch size $K = 1$, number of epochs 100, strata size $s = 20$, probability of censoring $p_c = 0.2$, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$. We use $B = 1000$ bootstrap resamples for the bootstrap approach and $n_o = 1000$ sample strata per observation for the plugin approach.

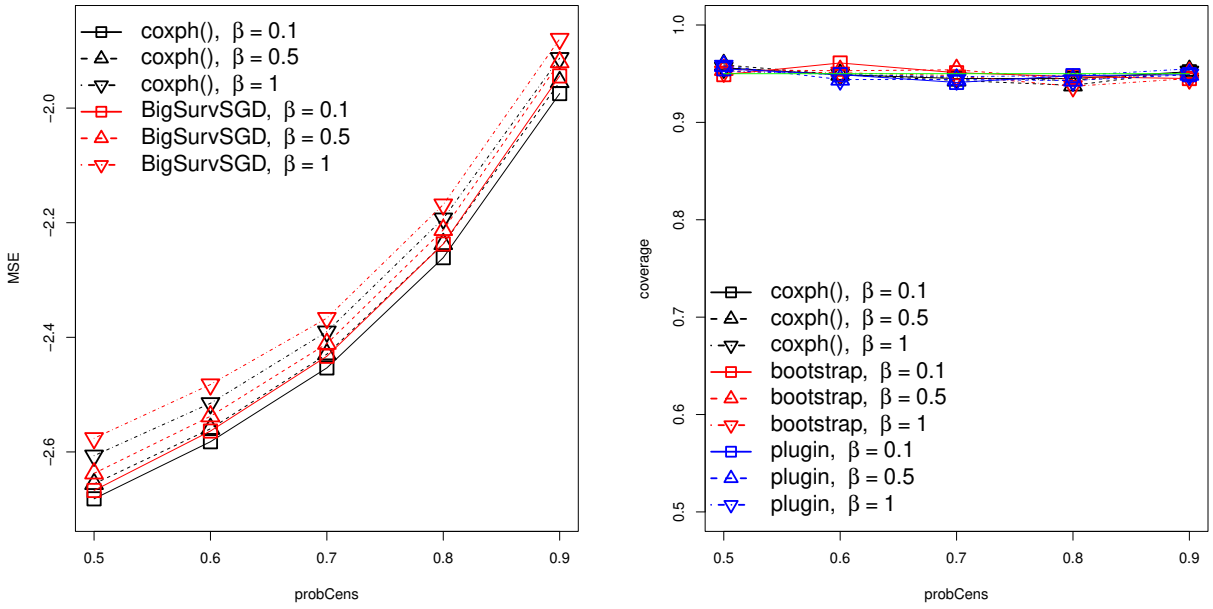

Figure 2.4: (left) $\log_{10}$(MSE); (right) coverage with $p = 10$ uniformly distributed and normalized features for varying values of probability of censoring p_c and levels of signal β with $\boldsymbol{\beta} = \beta \times (1, 1, 0, ..., 0)$. (only two covariates carry the signal). We choose mini-batch size $K = 1$, number of epochs 100, strata size $s = 20$, sample size $n = 1000$, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$. We use $B = 10$ bootstrap resamples for the bootstrap-t approach [57] and $n_o = 100$ sample strata per observation for the plugin approach.

there is more censoring. Appendix A.11.7 presents results with a larger strata size, $s = 50$. In practice, higher strata size improves the efficiency in scenarios with rare event. One benefit of our approach is that it scales well with a large number of observations, particularly when the event of interest is very infrequent.

Categorical and skewed covariates: We examine the effect of categorical and skewed covariates on the performance of `coxph()` and our proposed approach. We choose $p = 10$, five balanced binary covariates and five right-skewed covariates generated with a log-normal distribution with parameters $\mu = 0$ and $\sigma = 1$. We choose $\boldsymbol{\beta}^* = \beta \times (1, 0, 0, 0, 0, 1, 0, 0, 0, 0)$ and separately consider $\beta = 0.1, 0.5, 1$. For our algorithms estimates and CIs we use the

same parameters as in the *rare events* simulations. Figure 2.5 presents the MSE and coverage from different approaches for varying sample size n and level of signal β. We observe that all methods perform similarly. However, `coxph()` performs worse when sample size and the level of signal grow. The reason is that skewed covariates have large outliers, and `coxph()` becomes computationally unstable as a result of its recursive updating mechanism (specially when sample size and levels of signal are high). In contrast, our proposed framework performs does not have this issue. We also observe the performance of `coxph()` degrades more when the covariates become more correlated. Appendix A.11 presents additional results on different types of covariates, signal levels, and correlations among covariates. Note that here we focused on small-to-moderate datasets with two independent features carrying the signal in order to compare `coxph()` with our proposed framework. For larger problems (with more signal features) analogous results hold: See Appendix A.11 for more comparisons. An additional advantage of our proposed framework is that it supports significantly bigger datasets (as seen in Figure 2.2) and is easily extensible to massive datasets.

Model mis-specification: So far, we assumed that the Cox proportional hazards model holds. Now we consider a scenario for which the model is misspecified by missing to in include one of two variables affecting the event time. We compare the the estimate of parameter calculated by (2.5) for different values of strata size s. Suppose that our event time T and censoring time C are generated based on hazards functions $h_T(t, x)$ and $h_C(t, x)$ as following

$$\begin{aligned} h_T(t, x) &= h_0(t)e^{\gamma_1 x_1 + \gamma_2 x_2}, \\ h_C(t, x) &= h_0(t)e^{\alpha x_1}, \end{aligned} \tag{2.13}$$

where γ_1 and γ_2 formulate the relationship between the event time T, and features x_1 and x_2; α formulates the relationship between the censoring time C and feature x_1. The observation time, i.e., time to event or censoring whichever happens first and the event status are given

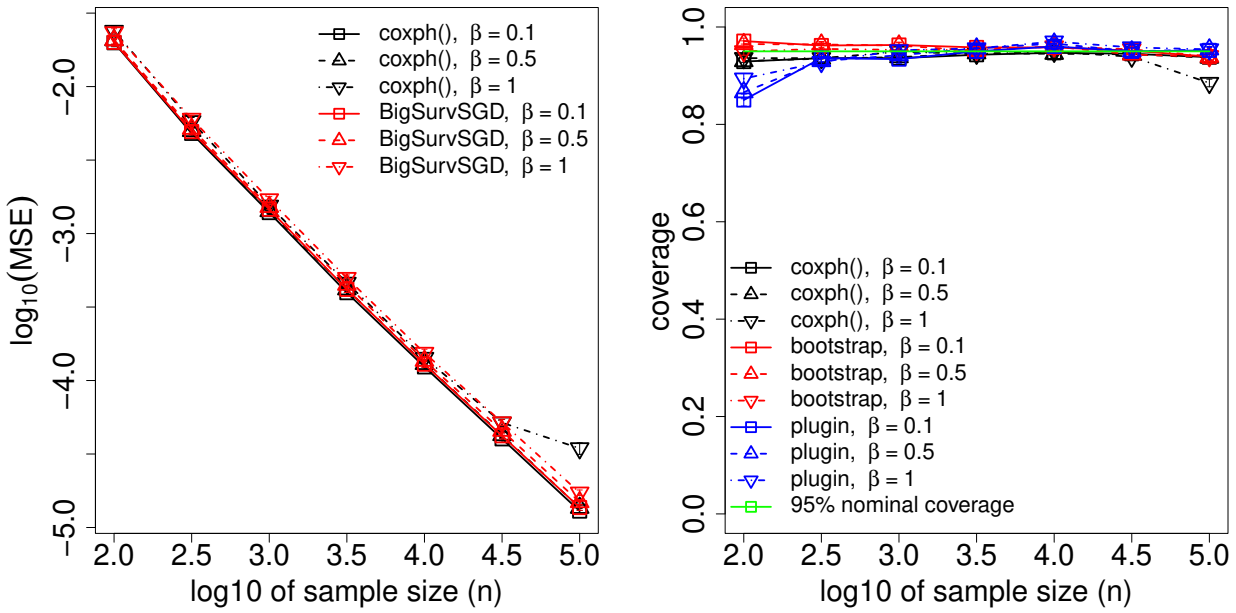

Figure 2.5: (left) $\log_{10}$(MSE); (right) coverage for varying values of sample size n and levels of signal with $\boldsymbol{\beta}^* = \beta \times (1, 0, 0, 0, 0, 1, 0, 0, 0, 0)$. We consider for $p = 10$ covariates: the first 5 covariates are binary variables generated from the Bernoulli distribution with probability of success 0.5 and the second five covariares are right-skewed variables generated from the log-normal distribution with parameters $\mu = 0$ and $\sigma = 1$. We choose mini-batch size $K = 1$, number of epochs 100, strata size $s = 20$, probability of censoring $p_c = 0.2$, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$. We use $B = 10$ bootstrap resamples for the bootstrap-t approach [57] and $n_o = 100$ sample strata per observation for the plugin approach.

by

$$Y = min(T, C),$$
$$\Delta = I(T \leq C). \tag{2.14}$$

The censoring rate (i.e., $p_c = Pr(C < T)$) depends on parameters γ_1, γ_2, and α. Here we only observe x_1 and we aim to study the effect of covariate x_1 through $\beta^{(s)}$ calculated by the minimization problem given by (2.5). We are interested in knowing that how different values of γ_1, γ_2, and α affect the estimate of $\beta^{(s)}$ from (2.5). For simulations, we independently

generate x_1 and x_2 from a uniform distribution, i.e., $x_1, x_2 \sim U(0,1)$ and then we use (A.47) and (2.12) to generate survival data with sample size $n = 10^5$. All of the following results are based on averaging over 100 randomly generated survival datasets. Figure 2.6 illustrates the average censoring rate for different values of γ_1, γ_2, and α. By noting that $x_1, x_2 > 0$, by following survival data generation mechanism in (2.12), the censoring rates increases by α and decreases by γ_1 and γ_2. Figure 2.7 compares the average estimate of parameter $\beta^{(s)}$ for different values of γ_1, γ_2, and α. With $\gamma_1 = 0$ (the first row in Figure 2.7), i.e., there is no effect from x_1, the estimate of $\beta^{(s)}$ are very close to γ_1. With $\gamma_2 = 0$ (the first column in Figure 2.7), i.e., no model misspecification, the estimate of the parameter $\beta^{(s)}$ is about the same as γ_1 for all models except for higher values of α where the optimization procedure with strata size $s = 2$ suffers from very low event rate. For other scenarios, when the model is misspecified (i.e., $\gamma_2 \neq 0$ and $\gamma_1 \neq 0$), when event rate is not very low, optimization with strata size $s = 2$ outperforms others. This is because in one hand, when $s \to \infty$, model misspecification affect the entire optimization procedure. On the other hand, when $s = 2$, model misspecification only affect some pairs (i.e., iterations) of the optimization procedure given by (2.5), not all of them.

Note that here we assume that x_1 and x_2 are independent covariates. Also, condition on observing x_1, the event and censoring times are independent. In Appendix A.11.8, we present two other scenarios, one with $x_2 = x_1^2$ (i.e., x_1 and x_2 are correlated in an quadratic form) and the other one for dependent censoring.

2.6.2 Survival neural networks

Our proposed framework facilitates the use of neural network-based predictive models. This section investigates it.

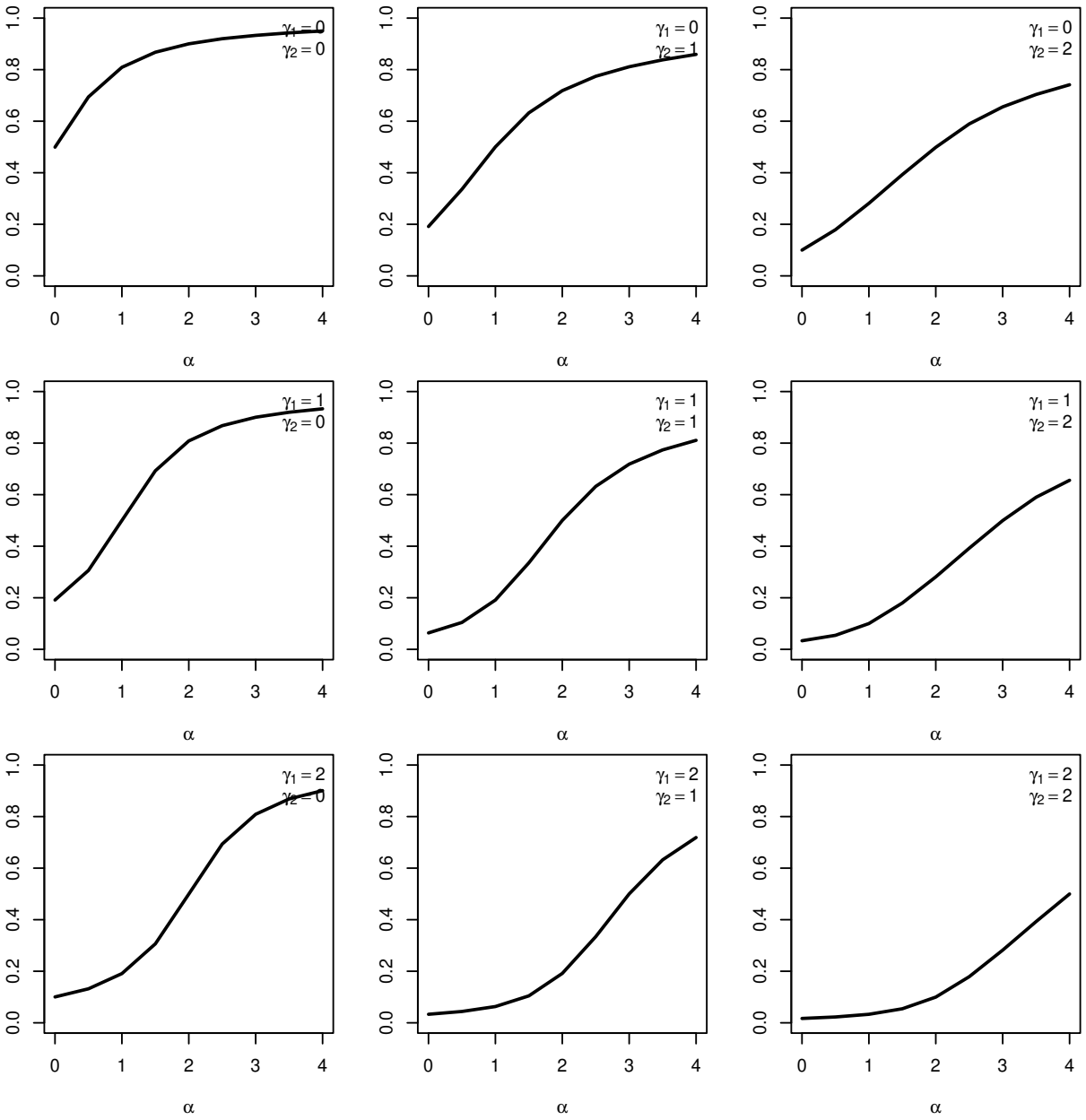

Figure 2.6: Average censoring rate over 100 randomly generated survival datasets with sample size $n = 10^5$ for different values of γ_1, γ_2, and α. Event and censoring times are generated based on hazards models in (2.13) with sample size sample size $n = 10^5$ where censoring rate is calculated by $Pr(C < T)$.

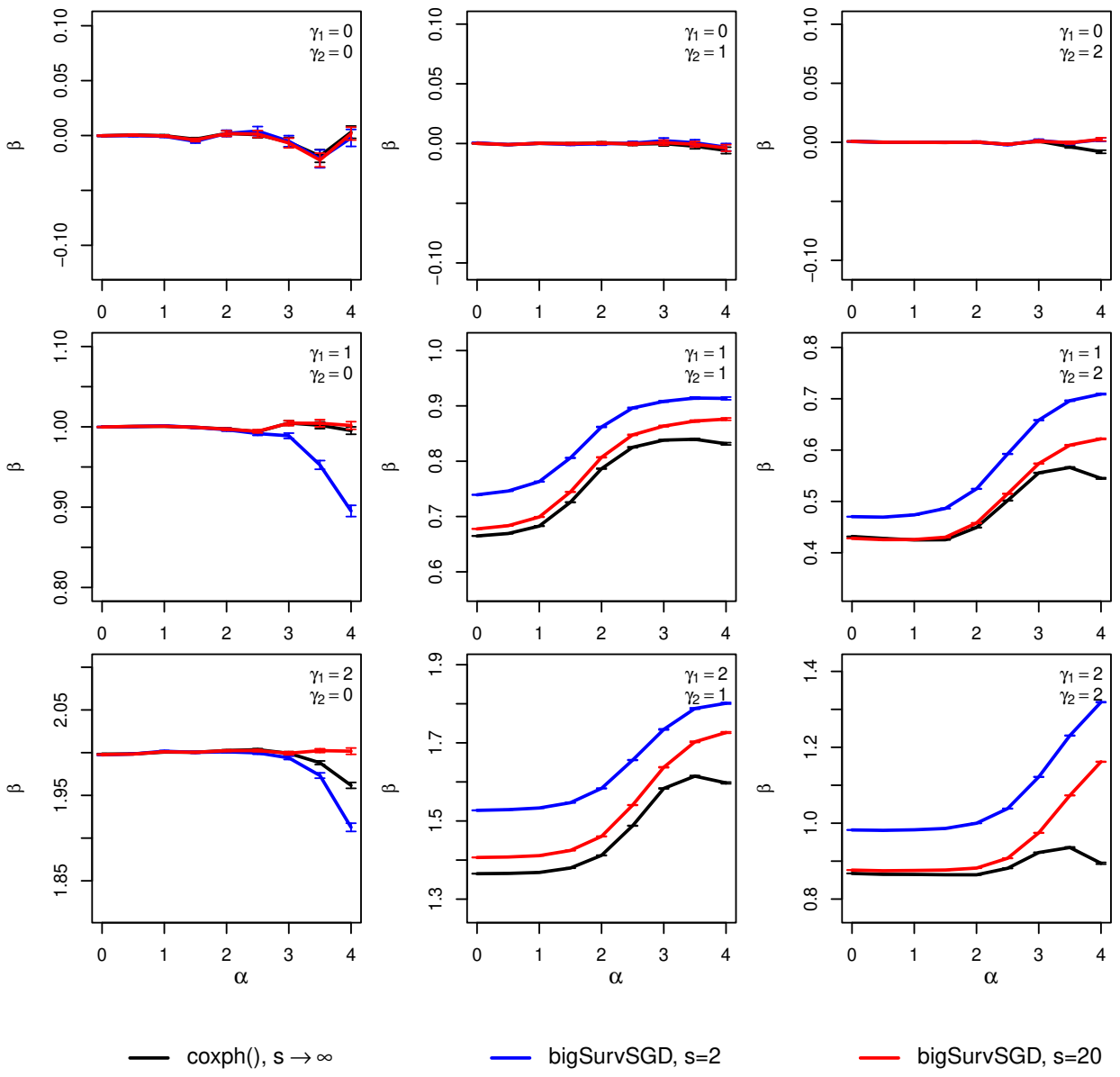

Figure 2.7: Average estimate of $\beta^{(s)}$ (y-axis) over 100 randomly generated survival datasets with sample size $n = 10^5$ from `coxph()` and `bigSurvSGD` with strata sizes $s = 2$ and $s = 20$. We considered hazards models in (2.13) to generate event and censoring with different values of γ_1, γ_2, α which result in different censoring rate (see Figure 2.6). We choose sample size $n = 10^5$. For our proposed framework, we choose mini-batch size $K = 1$, number of epochs 100, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

Proposed Neural Network Architecture

Figure 2.8 illustrates a generic network architecture based on our proposed framework in Section 2.3.3. There are s input lines, each going through a neural network `Net`. The network `Net` can take any custom architecture, e.g., multi-layer perceptron (MLP) for tabular data dand convolutional neural network (CNN) for imaging data. The network `Net` is exactly the same for all s lines, i.e., it shares the same network parameters $\boldsymbol{\beta}$ among all s lines. At each update of the network parameters, each line receives features/image(s) from a patient, goes through sub-network `Net`, and outputs the risk score $f_{\boldsymbol{\beta}}(.)$. Given a random stratum of s patients with data $\mathcal{D}^{(s)}$, we use back-propagation to update the network parameters $\boldsymbol{\beta}$ toward maximizing the stratum-average log-partial likelihood.

Tabular Data Examples

This section presents results for different methods using a variety of tabular datasets summarized in Table 2.1 (see Appendix A.11.3 for more details on these datasets). In addition to our proposed framework `BigSurvSGD` with linear $f_\beta(\boldsymbol{x})$ with varying strata size s, we also consider a version that uses a a multi-layer perceptron (MLP)-based neural network for $f_\beta(\boldsymbol{x})$, which we term `BigSurvMLP`. We compare our proposed frameworks `BigSurvSGD` and `bigSurvMLP` with `coxph()`; `CoxCC` and `CoxTime` [37]; `DeepSurv` [32]; `PCHazard` and `PMF` [35]; `DeepHit` [34]; `MTLR` [36]; and random survival forest `RSF` [58]. Some of these methods require tuning of hyper-parameters: See Appendix A.10.1 and A.10.2 for more details on the network architecture and the procedure used to tune hyper-parameters. Table 2.2 presents the average concordance index over 100 random training/testing splits of data. Our proposed framework `bigSurvMLP` is competitive on all datasets (it is always among the top two methods).

While strata-size, s, has at most a minor effect, our method `BigSurvSGD` with linear f_β seems to potentially perform slightly better (w.r.t. average concordance index) as we decrease the strata size: This is interesting as it is somewhat counter to our empirical

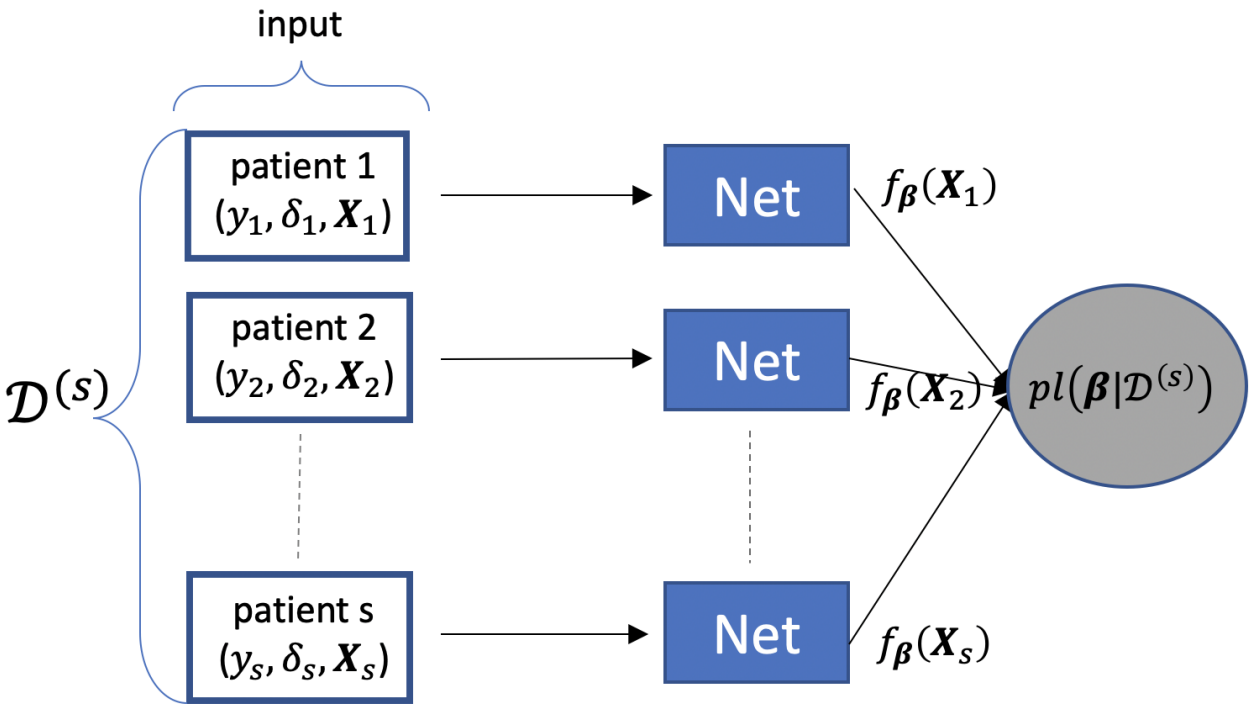

Figure 2.8: A generic network architecture for training the survival prediction model based on our proposed framework in Section 2.3.3. There are s lines of the sub-network `Net` which share exactly the same parameters $\boldsymbol{\beta}$. In line i, the sub-network `Net` extracts the features from its input $(y_i, \delta_i, \boldsymbol{X}_i)$ and calculates its risk score $f_{\boldsymbol{\beta}}(\boldsymbol{X}_i)$. Then the network parameters $\boldsymbol{\beta}$ are updated iteratively toward maximizing the stratum-averaged log-partial likelihood $pl(\boldsymbol{\beta}|\mathcal{D}^{(s)})$ following (2.9). $\boldsymbol{X}$ can have any structure, e.g., tabular data or imaging data.

efficiency results for a correctly specified model (as discussed in Sections 2.3.3 and 2.6.1). We believe this occurs because proportional hazards will never *exactly* hold, and in the case of this minor misspecification, optimizing a pairwise smoothed concordance on training data is closest to the pairwise concordance that we evaluate on test data. We additionally note, as expected, that `BigSurvSGD` and `coxph()` have increasingly similar performance as s increases. Appendix A.11.3 presents additional results for 95% CIs for the FLCHAIN and GBSG datasets.

Table 2.1: Summary of available real-world datasets.

	n	p	p_c	*source*
FLCHAIN	6,524	8	0.70	[59]
GBSG	2,232	7	0.43	[60]
METABRIC	1,904	9	0.42	[61]
NWTCO	4,028	6	0.86	[62]
SUPPORT	8,873	14	0.32	[63]

n : sample size, p : number of clinical variables, p_c : probability of censoring, and *source* : source of dataset.

Simulated imaging data

Our proposed framework can also engage with imaging data by using a convolutional neural network. To illustrate this possibility, we use the MNIST dataset [64] to simulate time-to-event outcome. The MNIST dataset is a standard benchmark dataset that has been used for multi-class classification purposes. It contains $n_{train} = 60,000$ and $n_{test} = 10,000$ greyscale images with size 28×28 for training and testing. These images correspond to digit numbers between 0 and 9. We simulate time-to-event outcome y_i such that the risk score (i.e., $f_{\boldsymbol{\beta}^*}(\boldsymbol{X}_i)$ defined in (2.1)) for individual i is proportional to its corresponding digit value d_i:

$$\begin{aligned} &y_i \sim \exp(\mu = \exp(-\eta d_i)) \text{ (time to event/censoring)}, \\ &\delta_i \sim Bernoulli(p = 1 - p_c),\ p_c = Pr(t_i > c_i), \end{aligned} \tag{2.15}$$

where η controls the amount of signal in our dataset: Higher η corresponds to better separation of event times for different digits. Note that (2.15) is exactly the same as (2.12) except we replace $\boldsymbol{X}_i^T\boldsymbol{\beta}^*$ with ηd_i. Now we create dataset $\mathcal{D}^{(n_{train})} = \{(y_i, \delta_i, \boldsymbol{X}_i), i = 1, 2, \ldots, n_{train}\}$ for training where $\boldsymbol{X}_i$ is the handwritten image encoding the digit value d_i. Note that, with known digit values (d_i's), the problem is as easy as fitting a univariate Cox model. Here we assume that the digit values are unknown, and we only have the coded digits through their greyscale images. Therefore, the task of the neural network is to learn event times

Table 2.2: Average testing concordance index over 100 random training/testing splits for different tabular datasets. For methods with hyper-parameters, we used a 5-fold cross-validation procedure for tuning. See Appendix A.10 for details about the network architecture and the procedure for tuning the hyper-parameters.

Method	Outcome	NN	FLCHAIN	GBSG	METABRIC	NWTCO	SUPPORT
`coxph()`	Cont.	No	0.792	0.661	0.634	0.713	0.569
bigSurvSGD, s=2	Cont.	No	0.793	0.667	0.639	0.716	0.573
bigSurvSGD, s=5	Cont.	No	0.793	0.665	0.639	0.715	0.573
bigSurvSGD, s=10	Cont.	No	0.793	0.664	0.637	0.714	0.572
bigSurvSGD, s=20	Cont.	No	0.793	0.663	0.636	0.714	0.571
bigSurvSGD, s=50	Cont.	No	0.792	0.662	0.634	0.713	0.570
RSF	Cont.	No	0.787	0.672	0.646	0.708	**0.616**
bigSurvMLP	Cont.	Yes	0.923	**0.675**	**0.651**	**0.719**	**0.616**
CoxCC	Cont.	Yes	0.922	0.664	0.642	0.571	0.606
CoxTime	Cont.	Yes	0.923	0.666	0.634	0.564	0.609
DeepSurv	Cont.	Yes	**0.924**	0.670	0.643	0.569	0.609
DeepHit	Disc.	Yes	0.923	0.660	0.577	0.670	0.587
MTLR	Disc.	Yes	0.922	0.663	0.626	0.575	0.604
PCHazard	Disc.	Yes	0.923	0.663	0.627	0.690	0.602
PMF	Disc.	Yes	0.923	0.665	0.628	0.681	0.608

Cont. : uses continuous time-to-event outcome, *Disc.* : uses discrete time-to-event outcome, and NN : uses neural network (multi-layer perceptron) for predicting risk score.

from handwritten images. Only three methods `MTLR`, `PMF`, and our proposed CNN-based framework `BigSurvCNN` facilitate the use of CNNs. To compare these different methods, we keep 10,000 testing images as the fixed hold-out testing dataset. To explore the effects of data constraints (e.g., sample size), we sample a fraction of the original 60,000 training samples as the training dataset. We consider 100 of such randomly sampled training datasets with size N_{sample}. We train our models using the sampled training dataset and report the

concordance index over the held-out testing dataset. We use the same minimalist network architecture `Net` for all three methods. These methods also require tuning of additional hyper-parameters: See Appendix A.10 for more details on the network architecture and the procedure for tuning hyper-parameters. Figure 2.5 compares average test concordance index for varying η and N_{sample} over 100 randomly sampled training datasets. All three methodologies compared perform quite similarly. In very small samples (e.g., $N_{sample} = 100$) and low-to-moderate signals (i.e., η), it appears that `bigSurvCNN` may have some improvement (though training CNNs with very small datasets is generally not ideal). We believe that these results indicate that `bigSurvCNN` is a strong and flexible candidate method for connecting neural network architecture to survival analysis. `bigSurvCNN` could be particularly effective with slightly more complex study designs, e.g., when a stratified analysis is required (when there are measured stratification features that are prognostic, but could not be used in future predictions, e.g. site in a multi-site study).

2.7 Discussion

We propose a simple framework for conducting large scale survival analysis using a Cox model. Our framework leverages a modified population optimization problem which allows us to apply iterative methods over only a subset of our observations at a time. In particular, it allows us to leverage the tools of stochastic gradient descent (and its extensions). This results in an algorithm that is more computationally efficient and stable than the current state of the art for problems with a larger number of observations (and features). We also introduced methods to construct confidence intervals for the parameters estimated by our framework. We showed that our framework is as effective as the standard Cox model. However, our proposed is more effective when the standard Cox model suffers from round-off error or the dataset is too large to fit in memory. In addition, we illustrated how to employ our framework with a neural-network-based hazard estimator. Using both real-world and simulated survival datasets, we showed that employing neural networks in our proposed framework is competitive with other state-of-the-art methods.

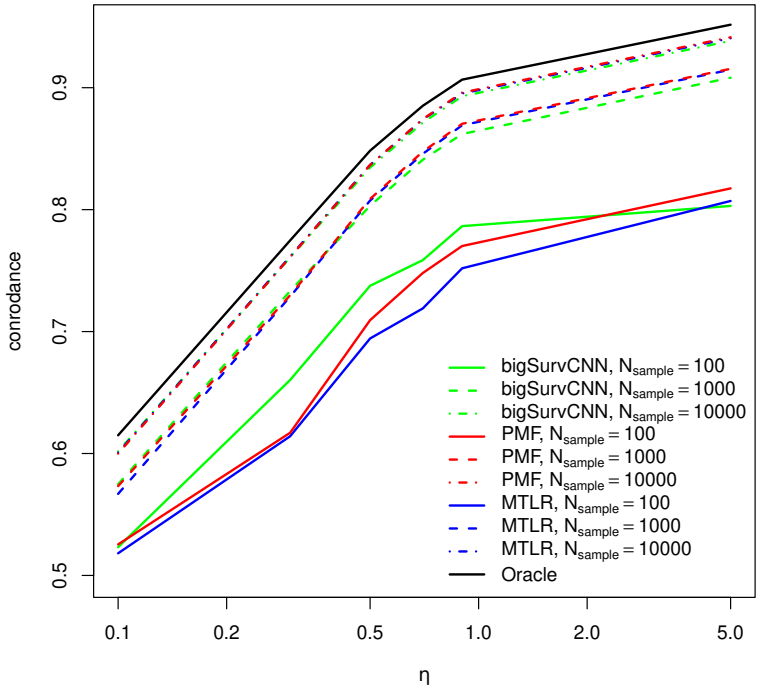

Figure 2.9: Average testing concordance index over 100 random sampled training data generated by the MNIST dataset. We compare our proposed method `bigSurvCNN`, `MTLR`, `PMF`, and Oracle for varying sizes of sampled training dataset N_{sample}. See Appendix A.9 for details about how we calculate the Oracle concordance. See Appendix A.10 for more details on the network architecture and the procedure for tuning the hyper-parameters.

2.8 Software

Our implementation for fitting the Cox model using the stochastic gradient descent (SGD) algorithm is part of our prepared R package `bigSurvSGD`. This package is publicly available on CRAN [65]. The Python codes for implementing our proposed neural network architectures `bigSurvMLP` and `bigSurvCNN` and comparing them with other existing methods are available in the Github repository `https://github.com/atarkhan/bigSurvNN` [66].

Chapter 3

A PENALIZED EXTENSION FOR HIGH DIMENSIONAL DATA

3.1 Introduction

Advances in high-throughput technologies have increasingly allowed us to collect massive datasets with many observations n and features p [4]. In Chapter 2, we discussed that the Cox model faces computational instability with many observations, even with a few features, especially if the datasets do not fit into memory. We then proposed a re-framed Cox model that is directly amenable to stochastic gradient-based optimization algorithms. The proposed framework scales well in the number of observations because data can be read off the hard-drive in chunks and the estimator can be updated iteratively. However, in many scenarios, the number of features is greater than the number of observations (i.e., $p > n$) (e.g., gene expression). The standard Cox model is known to behave poorly (the estimated coefficients may go to infinity) when the number of features is greater than the number of observations or even when the number of observations is greater than but close to the number of features.

There have been efforts in the literature to tackle the challenge of fitting a model with many features. Authors in [67] added a lasso (ℓ_1) penalty to linear regression to combat degenerate behavior. Authors in [8] extended the lasso penalty to the Cox model. The lasso penalized model not only resolves degenerate behavior (gives a well-defined solution), it also provides sparse solutions with many coefficients shrinking toward zero. Authors in [9] proposed using an elastic net penalty, which is a convex combination of ℓ_1 and ℓ_2 norms, in linear regression. Authors in [10] extended the use of the elastic net penalty to the Cox model. They used the Newton-Raphson algorithm to fit the penalized model along a path of penalty parameter values. Authors in [68, 69] fitted the regularized Cox models using

gradient descent. Although these models work well for small-to-medium datasets, they do not scale up well to large datasets. Authors in [70] studied $L_{1/2+2}$ method for gene selection in the Cox proportional hazards model where $L_{1/2+2}$ stands for a convex combination of $l_{1/2}$ and l_2 penalties. Authors in [21] presented tools for fitting regularized Cox survival analysis models on high-dimensional, massive sample-size data using a variant of the cyclic coordinate descent optimization technique. However, this method does not scale up well with many observations because of the use of gradient descent that needs the whole cohort on the memory. Authors in [28] proposed fitting Cox proportional hazards model with large data. However, they assumed that $n \gg p$ that might not be the case for the datasets with many covariates. Authors in [71] developed an algorithm to fit a regularized Cox model with l_1 penalty based on the Batch Screening Iterative Lasso method [72]. Their algorithm leverages the screen-solve-check substructure inspired by [73] to screen covariates subset-by-subset through an ordered regularization path. As the authors mentioned, their algorithm is slow due to many iterations and cross-language communication implemented in R. Their algorithm is based on gradient descent that needs the whole cohort to be on the memory.

Authors in [5], inspired by [74], introduced a path-wise algorithm for the penalized Cox model with the elastic net penalty. They employed a cyclical coordinate descent algorithm and used warm starts to fit the penalized Cox model. They implemented their work in an R package, `glmnet` [75], which efficiently fits the elastic-net regularized Cox model for moderate n and potentially large p. Unfortunately, this optimization method (and more generally those that engage with the full partial likelihood) require that the entire dataset be read into memory before the model is fit. This creates problems when engaging with very large datasets, including many observations.

In this chapter, we propose a framework for fitting the penalized Cox model that scales well with p and n, and allows one to fit a penalized Cox model even with data that does not fit into memory in a single machine. We consider a modification of the log partial likelihood of Chapter 2 with the addition of a convex combination of l_1 and l_2 penalties (*elastic net*). In particular, this new and modified objective function decouples across subsets (strata) of

observations. It enables us to use stochastic gradient-based algorithms that engage with a portion of observations at a time. We use stochastic proximal gradient descent (SPGD) to iteratively fit our re-framed penalized Cox model. We show that the m-th iterate of our proposed algorithm based on SPGD can achieve the non-asymptotic bound of $O(m^{-1})$ in expectation. We fit the model for a sequence of regularization parameters, borrowing the idea from [5, 74]. We empirically demonstrate that our proposed framework works when both the number of observations and the number of features grow (in particular with $p \gg n$). I implemented our proposed framework as part of our publicly available R package `bigSurvSGD` [65].

3.2 Penalized Cox model

We consider the standard survival analysis framework presented in Chapter 2. We assume that the number of features is greater than the number of observations or the number of observations is greater than but close to the number of features. In such settings, the solutions in (2.3) are not well behaved and the penalized Cox model is usually used to obtain well-behaved solutions. In our work, we focus on the penalized Cox model with the elastic net penalty studied by [74, 5]. The elastic-net penalized full partial likelihood estimator is given by

$$\widehat{\boldsymbol{\beta}}^{(n)} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\Big\{ -\frac{2}{n} pl^{(n)}(\boldsymbol{\beta}|\mathcal{D}^{(n)}) + P_{\alpha,\lambda}(\boldsymbol{\beta})\Big\}, \quad P_{\alpha,\lambda}(\boldsymbol{\beta}) = \lambda\Big(\alpha||\boldsymbol{\beta}||_1 + \frac{1}{2}(1-\alpha)||\boldsymbol{\beta}||_2^2\Big), \tag{3.1}$$

where $P_{\alpha,\lambda}(\boldsymbol{\beta})$ is known as the elastic net penalty. If we choose $\lambda = 0$, the problem in (3.1) reduces to the one in (2.3). By changing α from 0 to 1 and $\lambda \neq 0$, we move from ridge-like to lasso-like solutions. By adjusting α, we may get solutions that have the benefits of both penalties. With large-scale and ultra-high dimensional survival datasets, the data is often too large to be stored in memory, which leads to severe issues when trying to fit (3.1).

3.3 Re-framed large scale penalized Cox models

We propose to re-frame the penalized model in (3.1) to handle large-scale and ultra-high dimensional survival datasets. We consider a population parameter $\boldsymbol{\beta}^{(s)}$, defined as the minimization of the penalized expected negative partial likelihood of s random patients

$$\boldsymbol{\beta}^{(s)} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\Big\{\mathbb{E}_s\Big[l^{(s)}(\boldsymbol{\beta})\Big]\Big\} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\Big\{\mathbb{E}_s\Big[\frac{-1}{s}pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)}) + P_{\alpha,\lambda}(\boldsymbol{\beta})\Big]\Big\}, \tag{3.2}$$

where $\mathcal{D}^{(s)}$ is a draw of s random patients. The classical optimization methods generally assume that the objective function is smooth and strongly convex [76, 77]. The objective function in (3.2) is strongly convex with respect to $\boldsymbol{\beta}$: it is the sum of the convex function $-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})$ [6] and the strongly convex elastic net penalty function $P_{\alpha,\lambda}(\boldsymbol{\beta})$ [9, 78]. Therefore, $\boldsymbol{\beta}^{(s)}$ is the unique minimizer of the population optimization problem in (3.2). However, the objective function in (3.2) is not differentiable for $\alpha > 0$ because the l_1 norm in $P_{\alpha,\lambda}(\boldsymbol{\beta})$ is non-differentiable. Given this, the stochastic gradient descent cannot be directly employed. The stochastic sub-gradient descent generally also results in very poor performance and we aim to avoid it. However, the problem in (3.2) can be treated as a composite problem which has been extensively addressed in the literature [79, 80, 81]. In particular, there has been growing popularity of optimization methods that separate the contribution of smooth and non-smooth components [82, 83]. To optimize our re-framed problem in (3.2), we write it in the form of a composite problem as

$$\boldsymbol{\beta}^{(s)} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\Big\{\mathbb{E}_s\Big[f^{(s)}(\boldsymbol{\beta})\Big] + R(\boldsymbol{\beta})\Big\} \tag{3.3}$$

where $f^{(s)}(\boldsymbol{\beta}) = \frac{-1}{s}pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})$ is a convex and smooth function over randomly selected stratum of data $\mathcal{D}^{(s)}$, and $R(\boldsymbol{\beta}) = P_{\alpha,\lambda}(\boldsymbol{\beta}) = \lambda\Big(\frac{1}{2}(1-\alpha)||\boldsymbol{\beta}||_2^2 + \alpha||\boldsymbol{\beta}||_1\Big)$ is a simple non-differentiable strongly convex function independent of s. Alternatively, we can consider a U-statistic-based version of our population loss in (3.2), given by

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\Big\{-\frac{1}{sn_s}\sum_{\mathcal{D}^{(s)}\subset\mathcal{D}^{(n)}} pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)}) + P_{\alpha,\lambda}(\boldsymbol{\beta})\Big]\Big\}, \tag{3.4}$$

where $n_s = \binom{n}{s}$ is total number of distinct strata of size s. The minimization problem in (3.4) is similar to what is known as minimization of (penalized) generalized U-statistics [84, 85] of degree s with kernel $h = -pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)}) + P_{\alpha,\lambda}(\boldsymbol{\beta})$. One benefit of our empirical minimization problem in (3.4) compared to (3.1) is that the objective function in (3.4) can decouple over strata of size s. Therefore, we can use a stochastic-based optimization to update our estimator $\widehat{\boldsymbol{\beta}}$ over iterations using (mini-batches of) strata of size s instead of considering all observations. Additionally, this enables managing extremely large datasets by reading data off the hard drive in chunks as small as s observations at a time.

We apply stochastic proximal gradient descent (SPGD) [86, 87] to iteratively update our estimator $\widehat{\boldsymbol{\beta}}$ using a single or mini-batches of random strata of s observations. Suppose that we have n_s independent strata, $D_1^{(s)}, \ldots, D_{n_s}^{(s)}$ each with s independent patients drawn from our population (with $s \geq 2$). The $m+1$ iterate of our SPGD algorithm is given by:

$$\widehat{\boldsymbol{\beta}}_{m+1} = \frac{S\Big(\widehat{\boldsymbol{\beta}}_m + \frac{\gamma_m}{s}\nabla p\ell^{(s)}(\widehat{\boldsymbol{\beta}}_m|\mathcal{D}_m^{(s)}), \gamma_m\alpha\lambda\Big)}{1+\gamma_m(1-\alpha)\lambda}, \tag{3.5}$$

where $S(z, a)$ is the soft thresholding operator. See See Appendix B.1 for more details on how we derive updating rule in (3.5). The computation time for SPGD updating rule in (3.5) is dominated by calculating the gradient $\nabla p\ell^{(s)}(\widehat{\boldsymbol{\beta}}_m|\mathcal{D}_m^{(s)})$. Therefore, the computation time for each iteration of our proposed updating rule in (3.5) is of order $O(sp)$ where s remains fixed. When sample size n grows (massive datasets), the computation time for each iteration of our proposed method remains fixed. In addition, we can choose s as small as 2 to reduce the amount of data needed to be stored in memory when the dataset is ultra-high dimensional (i.e., very large p). In practice, we may add some relaxation steps [87] to (3.5) to give a smoother solution path. Algorithm 1 summarizes the implementation of our updating procedure with a relaxation step. Note that the definition of *epoch* in our algorithm is slightly different. In our algorithm, we define an *epoch* as iterating over one random partition of the data, which includes $\sim n/s$ non-overlapped strata. In practice, we usually iterate over more than one epoch (random partition) to improve performance. γ_m is a strictly positive sequence and τ_m is a sequence in $[0, 1]$. Expression (3.6) in Algorithm 1 is a relaxation step proposed

by [87] to smooth the solutions. This step relaxes the current update of our estimator by taking a weighted average of the update from the proximal operator and the update from the previous iterate. When $\tau_m = 1$, the algorithm reduces to the one proposed by [88]. In Appendix 3.4, we derive the non-asymptotic convergence bound for the m-th iterate in expectation, i.e., $\mathbb{E}[||\widehat{\boldsymbol{\beta}}_m - \boldsymbol{\beta}_0||^2]$ and show that we can attain bound $O(m^{-1})$ by choosing appropriate γ_m and τ_m.

Algorithm 1: Stochastic proximal gradient descent algorithm for updating our estimator

---

Result: $\widehat{\boldsymbol{\beta}}$

Initialization:

Choose strata size s

Choose number of epochs n_E

$m = 0$

$\widehat{\boldsymbol{\beta}}_0 = \mathbf{0}$

Choose γ_m and τ_m

for *($n_e = 1, 2, ..., n_E$)* **do**

Divide original cohort data $\mathcal{D}^{(n)}$ into n_s disjoint strata $\mathcal{D}_1^{(s)}$, ..., $\mathcal{D}_{n_s}^{(s)}$ with size s.

for *($b = 1, 2, ..., n_s$)* **do**

$m = m + 1$

Compute

$$y_m = \frac{S\left(\widehat{\boldsymbol{\beta}}_m + \frac{\gamma_m}{s}\nabla_{\boldsymbol{\beta}} p\ell^{(s)}(\widehat{\boldsymbol{\beta}}_m | \mathcal{D}_b^{(s)}), \gamma_m \alpha\lambda\right)}{1 + \gamma_m(1-\alpha)\lambda}$$

Update $\widehat{\boldsymbol{\beta}}_{m+1}$ as (relaxation step)

$$\widehat{\boldsymbol{\beta}}_{m+1} = (1 - \tau_m)\widehat{\boldsymbol{\beta}}_m + \tau_m y_m \tag{3.6}$$

end

end

3.4 Convergence of Algorithm 1

We re-framed the penalized Cox model and proposed an iterative Algorithm 1 including a relaxation step based on SPGD to solve it. Our proposed algorithm iterates over strata of size s to update the estimator. With an original cohort of n patients, there will be in total $n_s = \binom{n}{s}$, i.e., in order of n^s distinct strata of size s. However, one benefit of our proposed re-framed framework is that its m-th iterate can achieve the non-asymptotic convergence bound of $O(m^{-1})$ in expectation, i.e., $\mathbb{E}[||\widehat{\boldsymbol{\beta}}_m - \boldsymbol{\beta}_0||^2] = O(m^{-1})$ where $\boldsymbol{\beta}_0$ is the unique minimizer of the population optimization problem given by (3.3). Here, we prove that such a convergence bound is achievable by choosing appropriate γ_m and τ_m. Recall our composite optimization problem in (3.4). We consider the following conditions throughout this section.

(M1) Gradient of $f^{(s)}(\boldsymbol{\beta})$ is an unbiased estimate of the gradient $F^{(s)}(\boldsymbol{\beta})$,

(M2) Gradient of $f^{(s)}(\boldsymbol{\beta})$ is D-Lipschitz-continuous, i.e., $\forall\ \boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathbb{R}^p$, there exists $D \geq 0$ such that,

$$||\nabla_{\boldsymbol{\beta}} f^{(s)}(\boldsymbol{\beta}_1) - \nabla_{\boldsymbol{\beta}} f^{(s)}(\boldsymbol{\beta}_2)|| \leq D||\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2||, \tag{3.7}$$

(M3) Variance of the gradient of $f^{(s)}(\boldsymbol{\beta})$ is bounded, i.e., there exists $\sigma^2 \in \mathbb{R}_+$ such that

$$\mathbb{E}(||\nabla_{\boldsymbol{\beta}} f^{(s)}(\boldsymbol{\beta}_0)||^2) \leq \sigma^2, w.p.1, \tag{3.8}$$

(M4) Function $F^{(s)}(\boldsymbol{\beta})$ is μ-strongly convex and $R(\boldsymbol{\beta})$ is ν-strongly convex for some $\mu \in [0, +\infty)$ and $\nu \in [0, +\infty)$ such that $\mu + \nu > 0$ (i.e., at least one of them is greater than zero). This guarantees that the problem in (3.2) has a unique solution $\boldsymbol{\beta}_0$.

(M5) There exist τ_m and γ_m such that

$$\sum_{m\geq 1} \tau_m \gamma_m = +\infty \quad \text{and} \quad \sum_{m\geq 1} \tau_m \gamma_m^2 < +\infty, \tag{3.9}$$

Corollary 1: For our composite objective function in (3.4) and under the conditions $\boldsymbol{M}_1 - \boldsymbol{M}_5$ if (1) there exists $\underline{\tau} \in (0, +\infty)$ such that $\inf_{m \geq 1} \tau_m = \underline{\tau}$, and (2) we choose $\gamma_m = c_1 m^{-\theta}$ with $c_1 \in (0, +\infty)$ and $\theta \in (0, 1]$; the following holds

$$\mathbb{E}[||\widehat{\boldsymbol{\beta}}_m - \boldsymbol{\beta}_0||^2] = \begin{cases} O(m^{-\theta}) & \text{if } \theta \in (0,1) \\ O(m^{-c}) + O(m^{-1}) & \text{if } \theta = 0 \end{cases}, \tag{3.10}$$

where $c = \frac{2c_1\underline{\tau}(\nu+\mu\epsilon)}{(1+\nu)^2}$. There are many choices for (γ_m, τ_m) such that condition $\boldsymbol{M}_5$ holds. For instance, $\boldsymbol{M}_5$ holds if we choose $\gamma_m = \frac{c_1}{m^a}$ and $\tau_m = \frac{c_2}{m^b}$ such that $a + b \leq 1$ and $2a + b > 1$. In the following, we show that conditions **M1-M4** hold for our composite objective function in (3.4).

Condition **M1** automatically holds for our composite objective function in (3.4) based on the definition of $F^{(s)}(\boldsymbol{\beta}) = \mathbb{E}_s[f^{(s)}(\boldsymbol{\beta})]$ in (3.2) and that $\nabla_{\boldsymbol{\beta}} F^{(s)}(\boldsymbol{\beta}) = \nabla_{\boldsymbol{\beta}} \mathbb{E}_s[f^{(s)}(\boldsymbol{\beta})] = \mathbb{E}_s[\nabla_{\boldsymbol{\beta}} f^{(s)}(\boldsymbol{\beta})]$.

The loss function $f^{(s)}(\boldsymbol{\beta})$ belongs to C^∞ continuous function family and condition (3.7) is equivalent to

$$\exists\, D \geq 0,\ s.t.,\ \forall \nu \in S_\nu = \{\nu : ||\nu||_2 = 1\}, \quad \nu^T \nabla^2_{\boldsymbol{\beta}} f^{(s)}(\boldsymbol{\beta}) \nu \leq D. \tag{3.11}$$

In Appendix A.4, we showed that the gradient of negative log-partial likelihood, i.e., $f^{(s)}(\boldsymbol{\beta})$ is D-Libschitz with $D = 0.5 \times \max_i ||\boldsymbol{x}^{(i)}||_2^2$. Thus, our composite function in (3.4) satisfies condition **M2**.

In Appendix A.4, we showed that the gradient of negative log-partial likelihood, i.e., $f^{(s)}(\boldsymbol{\beta})$ is bounded. Thus, our composite objective function satisfies condition **M3**.

In addition, $R(\boldsymbol{\beta}) = P_{\alpha,\lambda}(\boldsymbol{\beta}) = \frac{1}{2}(1-\alpha)||\boldsymbol{\beta}||_2^2 + \alpha||\boldsymbol{\beta}||_1$ is strongly convex w.r.t. β for $\alpha < 1$. As a result, both μ and ν are greater than zero. This satisfies condition **M4** for our

composite objective function in (3.4).

The proof of statement in Proposition 1 follows Theorem 3.2 and Corollary 3.3 in [87]. As a special case of Proposition 1, if we choose $\theta = 1$, $\tau_m = 1 = \underline{\tau}$ for every $m \geq 1$, and choose c_1 such that $c > 1$, we can achieve $\mathbb{E}[||\widehat{\boldsymbol{\beta}}_m - \boldsymbol{\beta}_0||^2] = O(m^{-1})$.

3.5 Simulation study

3.5.1 Data generation mechanism

We use a Gaussian copula model to generate three types of covariates (features) under various correlation schemes. We first generate p covariates from a multivariate normal distribution with mean zero and an exchangeable correlation structure with coefficient ρ. From here, we transform the first third of our features to marginally generate uniformly distributed the data between 0 and 1. In the second third, we transform the data to be binary by evaluating whether they each exceed 0. For the last third of covariates, we use transformation $Y = e^X$ to create right-skewed covariates. From here, we scale all the covariates to have variance 1. We only allow p_{sig} covariates of each of the 3 types to carry non-zero signal level β^*. We generate our survival outcome following [56], using proportional hazards (2.1): We use an exponentially distributed baseline hazard $h_0(t)$ with a scaling parameter 1 and generate independent event and censoring time as

$$\begin{aligned} &y_i \sim \exp(\mu = \exp\left(-\boldsymbol{x}_i^T \boldsymbol{\beta}^*\right)) \text{ (time to event/censoring)}, \\ &\delta_i \sim Bernoulli(p = 1 - p_c),\ p_c = Pr(t_i > c_i) \end{aligned} \tag{3.12}$$

where $y_i = \min(t_i, c_i)$, i.e., time to event or censoring, whichever comes first. Here p_c, the probability of censoring, is a parameter we can tune. Note that $\mu = \exp\left(-\boldsymbol{x}^T \boldsymbol{\beta}^*\right)$ is the scale parameter of the exponential distribution.

3.5.2 Simulation methods

We compare our proposed framework (called `penSurvSGD`) with the state-of-the-art method implemented in the package `glmnet`. We use a Linux-based SONY VAIO laptop with `2.90GHz 4-Core Intel Core i7-3520M CPU`, memory `RAM 12GB`, graphics card `GeForce GT 640M LE` to run the simulations and gather the results.

We fix $\alpha = 0.95$ and consider a regularization path of 10 different values for λ which are equally spaced on the log scale between $\lambda_{\max}$ and $\lambda_{\min}$. For our proposed method, we choose mini-batches of size 1 (i.e., a single stratum per each iterate), $n_e = 100$ epochs, 1000 randomly selected strata for estimating $\lambda_{\max}$, and $\gamma_m = \frac{C}{\sqrt{m}}$ with $C = 0.12$.

We tune hyper-parameters λ and α to achieve the best model. We fix $\alpha = 0.95$ and consider 10 different values of λ on *logarithmic* scale starting from $\lambda_{\max}$ calculated by (B.11) and ending to λ_{min} which is selected as 0.01 for $p > n$ and 0.0001 for $p \leq n$. To tune λ, we split our data into 60%, 20%, and 20% for training, validation, and testing. We train our model over training data for all values of λ along the regularization path. We use the validation data to determine the best model (i.e., the best value of λ) that maximizes the validation concordance index. Finally, we select the best-trained model and report the testing concordance index.

3.5.3 Results

We compare the average testing concordance index (averaged over 50 random splits of data) of our proposed framework `penCoxSPGD` with `glmnet` for varying sample size n, feature size p, level of signal β^*, number of features carrying the signal p_{sig}, and level of correlation ρ. Tables 3.1 and 3.2 show results for $p = 10^3$ with $p_{sig} = 1$ and $p_{sig} = 3$; Tables 3.3 and 3.4 show results for $p = 10^4$ with $p_{sig} = 1$ and $p_{sig} = 3$; Tables 3.5 and 3.6 show results for $p = 10^5$ with $p_{sig} = 1$ and $p_{sig} = 3$. We observe that both methods perform very similarly. As expected, increasing β^*, p_{sig}, and n improve the performance. In addition, a higher correlation among features improves the performance. We believe this occurs because,

with a higher correlation, those selected covariates with no signal (i.e., with $\beta = 0$) better represent covariates carrying the signal (i.e., with $\beta \neq 0$). We also observe that our proposed framework with the strata size $s = 2$ performs slightly better than $s = 20$. We presume this happens because optimizing a penalized smoothed pairwise concordance on training data is closest to the pairwise concordance that we evaluate on test data. One benefit of our proposed framework is that it can be scaled to datasets too large to fit into memory. This is important for large sample sizes or extremely high dimensional features. For instance, with $p = 10^5$ features, increasing sample size from $n = 10^2$ to $n = 10^3$ significantly improves the performance by using our proposed framework (see the second column in Tables 3.5 and 3.6). However, such an improvement is not possible with `glmnet` on our machine because the data does not fit in memory (see the last rows in Tables 3.1-3.6 with "$-$" for `glmnet`).

3.6 Discussion

We propose a simple framework for fitting the penalized Cox model using large and ultra-high-dimensional datasets. We employ the stochastic proximal gradient descent (SPGD) algorithm in our approach to fit a penalized Cox model with the elastic-net penalty. Our framework needs a subset of s observations ($s \ll n$) per each iterate of the SPGD algorithm. It handles large datasets by reading data in chunks off the hard drive. However, there are other algorithms [21, 28, 71] which handles large datasets by considering subsets of covariates per iteration. We can easily combine our proposed framework with these algorithms to propose even more scalable algorithms for large and ultra-high-dimensional survival datasets. It is known that the Lasso penalty results in biased estimates [89]. Therefore, another extension is to de-bias the estimates given by our proposed framework for high-dimensional inference [90].

3.7 Software

Our implementation for fitting the penalized Cox model using the stochastic proximal gradient descent (SPGD) algorithm is part of our prepared R package `bigSurvSGD`. This package

		$\rho = 0.0$		$\rho = 0.5$	
		$\beta = 0.1$	$\beta = 1.0$	$\beta = 0.1$	$\beta = 1.0$
	Oracle	0.55	0.80	0.56	0.84
$n = 10^2$	glmnet	0.50 (0.01)	0.69 (0.01)	0.51 (0.01)	0.78 (0.01)
	penCoxSPGD, s=2	0.49 (0.01)	0.66 (0.01)	0.51 (0.01)	0.77 (0.01)
	penCoxSPGD, s=20	0.51 (0.01)	0.66 (0.01)	0.51 (0.01)	0.75 (0.01)
$n = 10^3$	glmnet	0.51 (0.00)	0.80 (0.00)	0.53 (0.00)	0.83 (0.00)
	penCoxSPGD, s=2	0.50 (0.00)	0.80 (0.00)	0.55 (0.00)	0.83 (0.00)
	penCoxSPGD, s=20	0.51 (0.00)	0.79 (0.00)	0.54 (0.00)	0.81 (0.00)
$n = 10^4$	glmnet	0.54 (0.00)	0.80 (0.00)	0.56 (0.00)	0.84 (0.00)
	penCoxSPGD, s=2	0.53 (0.00)	0.80 (0.00)	0.55 (0.00)	0.84 (0.00)
	penCoxSPGD, s=20	0.51 (0.00)	0.80 (0.00)	0.55 (0.00)	0.84 (0.00)
$n = 10^5$	glmnet	—	—	—	—
	penCoxSPGD, s=2	0.54 (0.00)	0.80 (0.00)	0.56 (0.00)	0.84 (0.00)
	penCoxSPGD, s=20	0.53 (0.00)	0.80 (0.00)	0.55 (0.00)	0.84 (0.00)

Table 3.1: Comparing concordance index between `glmnet` and `penCoxSPGD` with $p = 10^3$ covariates among which $p_{sig} = 1$ covariate per each type carries the signal. — indicates that there is no result due to lack of memory error. We present the results as `mean (SE)` for 50 random splits of data.

is publicly available on CRAN [65].

		$\rho = 0.0$		$\rho = 0.5$	
		$\beta = 0.1$	$\beta = 1.0$	$\beta = 0.1$	$\beta = 1.0$
	Oracle	0.58	0.87	0.65	0.93
$n = 10^2$	glmnet	0.50 (0.01)	0.58 (0.01)	0.61 (0.01)	0.86 (0.01)
	penCoxSPGD, s=2	0.51 (0.01)	0.63 (0.01)	0.61 (0.01)	0.86 (0.01)
	penCoxSPGD, s=20	0.51 (0.01)	0.62 (0.01)	0.62 (0.01)	0.86 (0.01)
$n = 10^3$	glmnet	0.52 (0.00)	0.87 (0.00)	0.63 (0.00)	0.93 (0.00)
	penCoxSPGD, s=2	0.51 (0.00)	0.86 (0.00)	0.64 (0.00)	0.92 (0.00)
	penCoxSPGD, s=20	0.51 (0.00)	0.85 (0.00)	0.64 (0.00)	0.92 (0.00)
$n = 10^4$	glmnet	0.58 (0.00)	0.87 (0.00)	0.65 (0.00)	0.93 (0.00)
	penCoxSPGD, s=2	0.56 (0.00)	0.87 (0.00)	0.64 (0.00)	0.93 (0.00)
	penCoxSPGD, s=20	0.54 (0.00)	0.87 (0.00)	0.64 (0.00)	0.93 (0.00)
$n = 10^5$	glmnet	—	—	—	—
	penCoxSPGD, s=2	0.58 (0.00)	0.87 (0.00)	0.65 (0.00)	0.92 (0.00)
	penCoxSPGD, s=20	0.57 (0.00)	0.87 (0.00)	0.64 (0.00)	0.92 (0.00)

Table 3.2: Comparing concordance index between `glmnet` and `penCoxSPGD` with $p = 10^3$ covariates among which $p_{sig} = 3$ covariates per each type carry the signal. — indicates that there is no result due to lack of memory error. We present the results as `mean (SE)` for 50 random splits of data.

		$\rho = 0.0$		$\rho = 0.5$	
		$\beta = 0.1$	$\beta = 1.0$	$\beta = 0.1$	$\beta = 1.0$
	Oracle	0.54	0.80	0.57	0.84
$n = 10^2$	glmnet	0.48 (0.01)	0.63 (0.01)	0.48 (0.01)	0.76 (0.01)
	penCoxSPGD, s=2	0.47 (0.01)	0.56 (0.01)	0.51 (0.01)	0.76 (0.01)
	penCoxSPGD, s=20	0.50 (0.01)	0.53 (0.01)	0.52 (0.01)	0.75 (0.01)
$n = 10^3$	glmnet	0.50 (0.00)	0.80 (0.00)	0.55 (0.00)	0.84 (0.00)
	penCoxSPGD, s=2	0.50 (0.00)	0.79 (0.00)	0.55 (0.00)	0.82 (0.00)
	penCoxSPGD, s=20	0.50 (0.00)	0.75 (0.00)	0.54 (0.00)	0.77 (0.00)
$n = 10^4$	glmnet	—	—	—	—
	penCoxSPGD, s=2	0.51 (0.00)	0.80 (0.00)	0.55 (0.00)	0.84 (0.00)
	penCoxSPGD, s=20	0.51 (0.00)	0.79 (0.00)	0.55 (0.00)	0.81 (0.00)

Table 3.3: Comparing concordance index between `glmnet` and `penCoxSPGD` with $p = 10^4$ covariates among which $p_{sig} = 1$ covariate per each type carries the signal. — indicates that there is no result due to lack of memory error. We present the results as `mean (SE)` for 50 random splits of data.

		$\rho = 0.0$		$\rho = 0.5$	
		$\beta = 0.1$	$\beta = 1.0$	$\beta = 0.1$	$\beta = 1.0$
	Oracle	0.58	0.87	0.65	0.93
$n = 10^2$	glmnet	0.49 (0.01)	0.54 (0.01)	0.60 (0.01)	0.85 (0.01)
	penCoxSPGD, s=2	0.50 (0.01)	0.52 (0.01)	0.62 (0.01)	0.86 (0.01)
	penCoxSPGD, s=20	0.52 (0.01)	0.52 (0.01)	0.63 (0.01)	0.86 (0.01)
$n = 10^3$	glmnet	0.51 (0.00)	0.87 (0.00)	0.64 (0.00)	0.92 (0.00)
	penCoxSPGD, s=2	0.50 (0.00)	0.86 (0.00)	0.64 (0.00)	0.91 (0.00)
	penCoxSPGD, s=20	0.52 (0.00)	0.82 (0.00)	0.64 (0.00)	0.89 (0.00)
$n = 10^4$	glmnet	—	—	—	—
	penCoxSPGD, s=2	0.53 (0.00)	0.87 (0.00)	0.64 (0.00)	0.93 (0.00)
	penCoxSPGD, s=20	0.51 (0.00)	0.86 (0.00)	0.64 (0.00)	0.92 (0.00)

Table 3.4: Comparing concordance index between `glmnet` and `penCoxSPGD` with $p = 10^4$ covariates among which $p_{sig} = 3$ covariates per each type carry the signal. — indicates that there is no result due to lack of memory error. We present the results as `mean (SE)` for 50 random splits of data.

		$\rho = 0.0$		$\rho = 0.5$	
		$\beta = 0.1$	$\beta = 1.0$	$\beta = 0.1$	$\beta = 1.0$
	Oracle	0.54	0.81	0.55	0.85
$n = 10^2$	glmnet	0.51 0.01)	0.59 (0.01)	0.50 (0.01)	0.75 (0.01)
	penCoxSPGD, s=2	0.49 (0.01)	0.52 (0.01)	0.48 (0.01)	0.69 (0.01)
	penCoxSPGD, s=20	0.50 (0.01)	0.49 (0.01)	0.50 (0.01)	0.63 (0.02)
$n = 10^3$	glmnet	—	—	—	—
	penCoxSPGD, s=2	0.50 (0.00)	0.75 (0.00)	0.52 (0.00)	0.74 (0.00)
	penCoxSPGD, s=20	0.50 (0.00)	0.51 (0.00)	0.51 (0.00)	0.67 (0.01)

Table 3.5: Comparing concordance index between `glmnet` and `penCoxSPGD` with $p = 10^5$ covariates among which $p_{sig} = 1$ covariate per each type carries the signal. — indicates that there is no result due to lack of memory error. We present the results as `mean (SE)` for 50 random splits of data.

		$\rho = 0.0$		$\rho = 0.5$	
		$\beta = 0.1$	$\beta = 1.0$	$\beta = 0.1$	$\beta = 1.0$
	Oracle	0.57	0.87	0.65	0.93
$n = 10^2$	glmnet	0.51 (0.01)	0.50 (0.01)	0.58 (0.01)	0.84 (0.01)
	penCoxSPGD, s=2	0.50 (0.01)	0.49 (0.01)	0.57 (0.01)	0.81 (0.01)
	penCoxSPGD, s=20	0.50 (0.01)	0.49 (0.01)	0.52 (0.02)	0.76 (0.01)
$n = 10^3$	glmnet	—	—	—	—
	penCoxSPGD, s=2	0.50 (0.00)	0.78 (0.00)	0.61 (0.00)	0.85 (0.00)
	penCoxSPGD, s=20	0.50 (0.00)	0.50 (0.00)	0.57 (0.00)	0.78 (0.00)

Table 3.6: Comparing concordance index between `glmnet` and `penCoxSPGD` with $p = 10^5$ covariates among which $p_{sig} = 3$ covariates per each type carry the signal. — indicates that there is no result due to lack of memory error. We present the results as `mean (SE)` for 50 random splits of data.

Chapter 4

TRAINING DEEP MULTIPLE-INSTANCE LEARNING NETWORKS USING INSTANCE SAMPLING

4.1 Introduction

Thanks to the advancements in digital pathology, especially slide scanners, visual inspection of sampled tissues through high-resolution Whole-Slide Images (WSIs) from biopsies has become the gold standard for diagnosing various diseases in oncology, such as prostate cancer [13, 14, 91]. However, the manual inspection of the entire WSI (with a typical size $10^5 \times 10^5$ pixels) is costly and time-consuming to be done by an expert. Also, the diagnosis might differ from one expert to another, known as the observer variability [92].

Computational pathology aims to develop automated machine learning and artificial intelligence tools to analyze the giga-pixel WSIs [93, 94, 95]. Such tools save cost and time; they have showed great accuracy and provided high-quality health care to patients with different diseases [11, 95, 94]. However, developing such automated tools for WSIs comes with some new challenges, especially when using complex models such as deep neural networks. WSIs are giga-pixel images, and they are too big to be fed into a deep neural network because of the memory constraint. One immediate solution is to divide a WSI into many (typically hundreds of thousands) smaller regions (with the typical size of 256×256 or 512×512), also known as patches or tiles. One can train a deep neural network by feeding a single or very few numbers of these small images. However, the main challenge is the lack of pixel-level annotation and that the labels (i.e., indicating the status of disease) are only available at the slide (patient) level. A solution might be annotating those smaller image regions (or patches). Labeling such images by an expert at the pixel level (or in smaller image patches) is costly (labor and time) [96].

Multiple instance learning (MIL), as a typical weakly supervised learning method, has been proposed to tackle this challenge [15, 16]. In a MIL problem, the aim is to train a model with bags of instances where the algorithm can only access the labels at the bag level. Such a scenario often happens in pathology, where one usually divides a gigapixel WSI image into many smaller image regions, known as tiles or patches. For prostate cancer, for example, each image tile can be partially related to a sub-type (the bag label), but it may not represent it by itself. Therefore, the upcoming challenge with the MIL problem is that not all instances (image tiles) are equally predictive of the bag label (class), and some of them may even relate to the other classes [97].

Some works considered combining the instance-level responses from a classifier to alleviate this challenge [98, 99, 100, 17]. Among them, [17] proposed an attention-based deep MIL framework to handle this challenge. Their proposed framework includes two networks : (1) attention network and (2) classification network. These two networks are trained simultaneously. The attention network has parameters for updating the attention (importance) weights of different instances, while the classification network has parameters for the classification task. Although this approach increases flexibility and interpretability of MIL problems, it still has a challenge: it uses all instances per bag across all iterates when training the combined network. A WSI has hundreds of thousands of image tiles (e.g., with size 256×256). Feeding all of these instances, regardless of their predictive information for the class label, is time-consuming and computationally expensive. An attention MIL network may not need to be trained by instances that are just noises or have little information for the class label.

This chapter investigates different sampling strategies for the attention-based deep MIL framework. We consider four sampling strategies: (1) no sampling, (2) random sampling, (3) adaptive sampling, and (4) top-k sampling. We show how the sampling strategies substantially reduces computation time. Among them, we also show that random sampling strategy can improve performance compared to no sampling (i.e., using whole instances in the original work [17]) if we choose an enough number of selected instances. We use the Can-

cer Genome Atlas (TCGA) repository of prostate adenocarcinoma (TCGA-PRAD) dataset [101] and Camelyon16 [102] to compare different strategies and support our discoveries.

4.2 MIL and Attention-based MIL Networks

4.2.1 MIL problem formulation

Suppose there are N subjects (or patients) with bags of images $\boldsymbol{\mathcal{X}}^{(1)}, \boldsymbol{\mathcal{X}}^{(2)}, \ldots, \boldsymbol{\mathcal{X}}^{(N)}$ and bag-level binary labels $y^{(1)}, y^{(2)}, \ldots, y^{(N)} \in \{0, 1, \ldots, M-1\}$ where M is number of classes. The bag for n-th patient (i.e., $\boldsymbol{\mathcal{X}}^{(n)}$) contains K_n instance images $\boldsymbol{X}_1^{(n)}, \boldsymbol{X}_2^{(n)}, \ldots, \boldsymbol{X}_{K_n}^{(n)}$. For instance, in computational pathology, we can obtain such K_n instance images by sampling from different regions of a WSI (i.e., $\boldsymbol{\mathcal{X}}^{(n)}$). In the classical supervised learning, we have $K_n = 1$, i.e., there is one image per subject with corresponding label $y^{(n)}$. Note that the number of instances inside the bag can vary among different subjects. To decrease computing time and cost, it is common to use a state-of-the-art pre-trained network such as *ResNet50* [103] to extract a low-dimensional embedding feature $\boldsymbol{h}_k^{(n)}$ from k^{th} instance image of n-th subject, $\boldsymbol{X}_k^{(n)}$. After that, we have dataset $\{(\boldsymbol{h}_k^{(n)}, y^{(n)},\ \text{for}\ n = 1, 2, \ldots N\ \text{and}\ k = 1, 2, \ldots, K_n\}$. The task of the neural network is to predict the label of n-th subject, $y^{(n)}$, through extracting features from its K_n embedding $\boldsymbol{h}_k^{(n)}$, $k = 1, 2, \ldots, K_n$. In computational pathology applications (e.g., prostate cancer [91]), the instance-level labels $y_k^{(n)}, k = 1, 2, \ldots, K_n$ are unknown and we only have the bag-level label $y^{(n)}$. The bag-level images (e.g., WSIs) are too big to feed into the neural networks due to the memory constraint. Multiple-instance learning (MIL) is a weakly supervised learning approach to train the neural networks using instances while only bag labels are available [96]. For binary classification task (i.e., $M = 2$), the basic assumption of a MIL problem is:

$$y^{(n)} = \begin{cases} 0, & \text{iff } \sum_{k=1}^{K_n} y_k^{(n)} = 0 \\ 1, & \text{otherwise.} \end{cases} \quad \text{or} \quad y^{(n)} = \max_k \{y_k^{(n)}\}, \tag{4.1}$$

conveying that a bag is labeled positive if it contains at least a positive instance. The above two expressions are not appealing from the optimization perspective. The other possible

solutions are to use element-wise maximum and mean operators. But these two operators are pre-calculated and are non-trainable.

4.2.2 Attention-based MIL

Authors in [17] proposed an attention-based MIL pooling approach that is trainable. They proposed a combined architecture of two trainable networks: attention network and classification network. The attention network is trained so that the weighted average of embedding features by their trainable attention weights represents the class the most. The classification network is trained to minimize the prediction error given the pooled embedding feature as its input. These two networks are trained simultaneously. To allow for the element-wise non-linearity, (dis)similarities discovery, and a better expressiveness, the authors in [17] proposed to use a gated attention mechanism for MIL pooling. The MIL pooled (aggregated) feature is given as

$$\boldsymbol{h}_{bag}^{(n)} = \sum_{k=1}^{K_n} a_k^{(n)} \boldsymbol{h}_k^{(n)}, \tag{4.2}$$

with

$$a_k^{(n)} = \frac{\exp\left\{\boldsymbol{w}^T\left(\tanh(\boldsymbol{V}\boldsymbol{h}_k^{(n)}) \odot \text{sigm}(\boldsymbol{U}\boldsymbol{h}_k^{(n)})\right)\right\}}{\sum_{k'=1}^{K_n} \exp\left\{\boldsymbol{w}\left(\tanh(\boldsymbol{V}\boldsymbol{h}_{k'}^{(n)}) \odot \text{sigm}(\boldsymbol{U}\boldsymbol{h}_{k'}^{(n)})\right)\right)\right\}}, \tag{4.3}$$

where $\boldsymbol{w} \in \mathbb{R}^{L_1 \times 1}$, $\boldsymbol{U} \in \mathbb{R}^{L_1 \times L_2}$, and $\boldsymbol{V} \in \mathbb{R}^{L_1 \times L_2}$ are trainable parameters included in the attention network; tanh(.) and sigm(.) are the element-wise hyperbolic tangent and sigmoid functions; $\odot$ is an element-wise multiplication. Such an MIL pooling mechanism preserves flexibility and interpretability (see Section 2.4 in [17]). Finally, the bag-level aggregated representation $\boldsymbol{h}_{bag}^{(n)}$ is fed into the classification network that includes M individual classification branches. Each classification branch estimates the predicted score of the corresponding class. The predicted $M \times 1$ score vector is given as

$$\boldsymbol{s}_{bag}^{(n)} = \boldsymbol{W}_c^T \boldsymbol{h}_{bag}^{(n)}, \tag{4.4}$$

where $\boldsymbol{W}_c \in \mathbb{R}^{L_2 \times M}$ is the trainable classifier with M columns corresponding to the M branches predicting the score of M classes for the bag. Finally, one can estimate the bag label by

$$\widehat{y}^{(n)} = \arg\max_c \{\boldsymbol{s}_{bag}^{(n)}\}. \tag{4.5}$$

In many pathology applications, there might be many instances within each WSI which would increase computing time and cost. In the next section, we present different sampling strategies to overcome these possible shortcomings.

4.3 Sampling Strategies for Attention-based MIL

4.3.1 Random sampling

With random sampling strategy, we randomly draw a few instances (or images) to train the deep neural network. The main reason to use this strategy is because of memory constraint: it is not possible to bring all instances/images of a patient (bag) or a batch of patients into memory to train the deep neural network. This strategy has been used in the literature [104, 105, 106] demonstrated great success in reducing computing resources and time. However, there is a lack of investigation on the computing time and performance of random sampling in the deep attention-based MIL network. On the one hand, different random subsets of instances for a patient (bag) for training the network over different iterates may increase generalizability and handle over-fitting better [107]. On the other hand, using a limited number of instances per iterate may not capture the entire information in predicting the outcome of the patient. Therefore, it might be worth investigating such a trade-off, which is one aim of this chapter.

4.3.2 Adaptive sampling

In practical applications (e.g., prostate cancer [91]), there are many instance images that may not contribute to the bag (patient) class. There have been some works in the literature handling this issue [108, 109, 110], but they all used whole instances. We propose to

adaptively draw G well-predictive instances per subject (bag) from an empirical sampling distribution [18, 19]. For nth patient, we estimate the sampling distribution as a multinomial distribution with a corresponding vector of probabilities $\mathcal{P}^{(n)} = (p_1^{(n)}, p_2^{(n)}, \ldots, p_{K_n}^{(n)})$, with $0 \leq p_k^{(n)} \leq 1$, $\sum_{k=1}^{K_n} p_k^{(n)} = 1$; we choose $p_k^{(n)} = a_k^{(n)}$ (the attention weight extracted from the forward attention network). We propose to draw a subset of G indices from distribution $\mathcal{P}^{(n)}$ as

$$(I_1^{(n)}, I_2^{(n)}, \ldots, I_G^{(n)}) \sim \mathcal{P}^{(n)}. \tag{4.6}$$

With (4.6), instances that have higher attention weights (i.e., higher $a_k^{(n)}$ that are well-predictive of the outcome) will be chosen more often during training. After adaptively drawing the G instances over each iterate, we train the attention-based neural network by following (4.2) - (4.5) by replacing K_n with G. Since the estimates of the network parameters and consequently the attention weights $a_k^{(n)}$ are noisier over a couple of initial iterates (epochs), we propose to consider a few initial iterates as warm-up iterates where we use all instances to train the network. Although the estimation of instance sampling distribution using the forward attention network is faster than training the whole network, it may add overload if we do it on every epoch. Therefore, one might decide to estimate $\mathcal{P}^{(n)}$ only after every a pre-specified number of epochs, e_{update}.

Note that authors in [111] compared uniform and adaptive instance sampling with other networks without sampling. But they fixed the attention network for the uniform sampling. We take a more fair approach and consider the same network architecture for all strategies we aim to compare in this chapter.

4.3.3 Top-k sampling

An alternative to the adaptive sampling is top-k sampling strategy, which has been used in the computational pathology literature [112, 113]. In this sampling strategy, top k instances with the highest instance-level score are selected to train the network. In our comparison, we select top $k = G$ instances with the highest attention weights given by (4.3). Throughout

Table 4.1: Grade Group, Gleason score, and their association with the risk level

Grade Group	Gleason score	Combined Gleason Score	Risk level
1	3+3	6	Low risk
2	3+4	7	Favorable intermediate
3	4+3	7	Unfavorable intermediate
4	4+4, 3+5, 5+3	8	High risk
5	4+5, 5+4, 5+5	9 and 10	Very high risk

this chapter, we call this instance sampling method as top-k.

Figure 4.1 illustrates the different instance sampling strategies.

4.4 Dataset, Network Architecures, and Tuning Hyper-parameters

4.4.1 Datasets

TCGA-PRAD (prostate cancer) dataset

We use The Cancer Genome Atlas (TCGA) repository of prostate adenocarcinoma (TCGA-PRAD) dataset [101] to evaluate our proposed approach. The Gleason score (GS) from the biopsied tissue is the common method for measuring the cancer status [114]. The GS is the sum of primary and secondary scores, each ranging from 3 to 5. Therefore, the GS ranges from 6 (3+3) to 10 (5+5). Another alternative and commonly-used scoring system is Grade Group (GG) which divides the prostates cancer patients into five groups based on pathological patterns. Table 4.1 summarizes GS, GG, and corresponding risk levels based on *NCCN Clinical Practice Guidelines in Oncology* [115]. Both GG and GS have been widely used in prostates cancer studies [116].

We follow the same procedure for sampling (with 20× magnification) image tiles (with size 256×256) from WSIs and the same procedure for pre-processing image tiles as explained and used in [108]. The bag (patient) size varies among patients, with a minimum of 1,308,

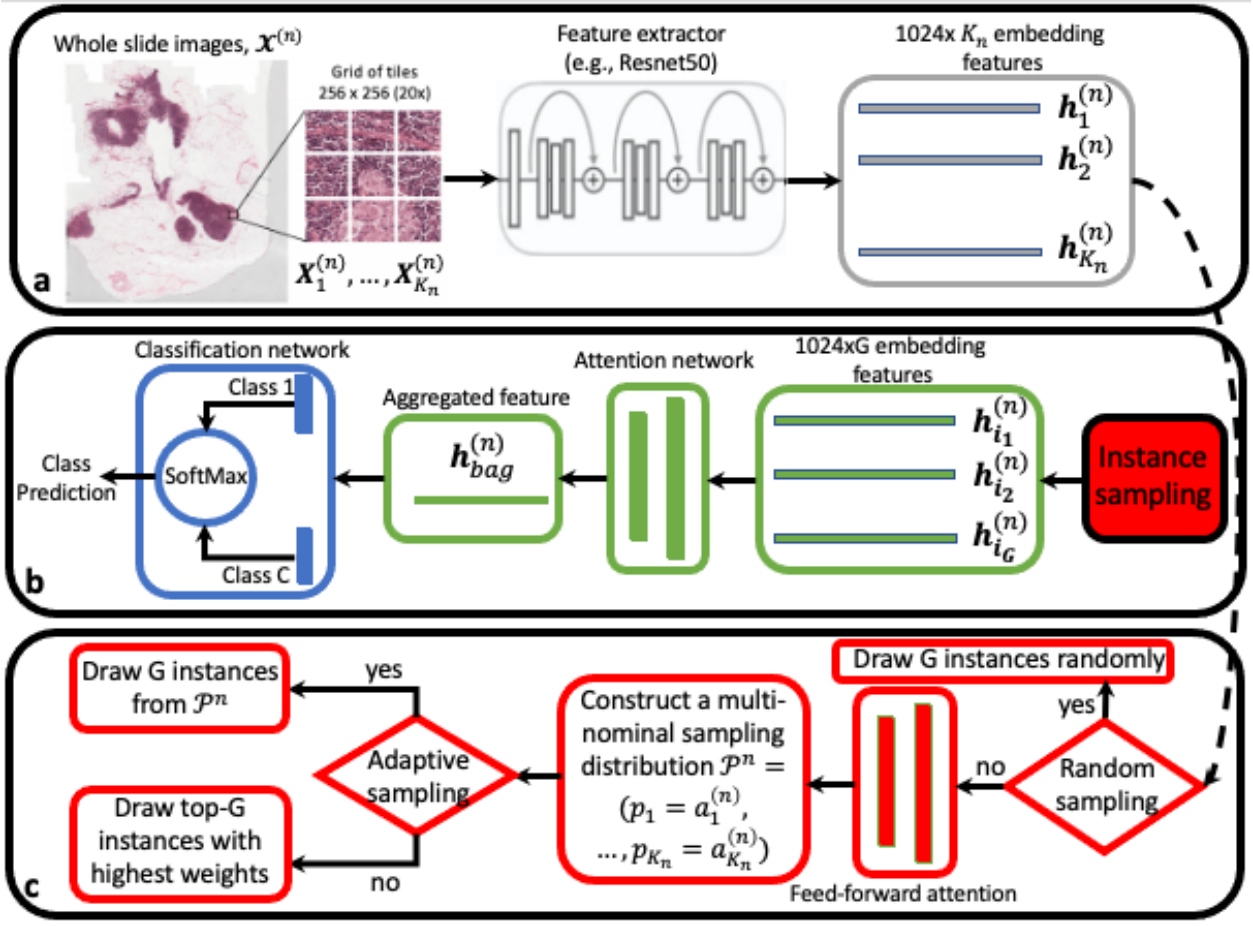

Figure 4.1: Different instance sampling strategies. **(a): pre-processing** We sample patches from the WSI $\boldsymbol{X}^{(n)}$ and use pre-trained network (e.g., ResNet50) to extract lower-dimensional features $\boldsymbol{h}_1^{(n)}$, ..., $\boldsymbol{h}_{K_n}^{(n)}$. **(b): training procedure** We use a subset of G instances (selected by strategies in panel **(c)**) and obtain aggregated feature $\boldsymbol{h}_{bag}^{(n)}$ using the attention network. Then, we predict the class label using the classification network. **(c): instance sampling strategies** We use the trained (fixed) feed-forward attention network to estimate the sampling distribution $\mathcal{P}^{(n)}$ and then draw G instances out of K_n instances using different sampling strategies.

a maximum of 130,752, and an average of 49,811 image tiles. To reduce computing time and cost, we use pre-trained Resnet50 network [103] to extract features from image tiles (instances) into one-dimensional embedding features with size 1,024 (this procedure is known as transfer learning [117]). Here, our focus is the binary classification where we divide patients into two classes: class 0 includes *low risk* (grade group 1) and *favorable intermediate* (grade group 2); and class 1 includes *unfavorable intermediate risk* (grade group 3), *high risk* (grade group 4), and *very high risk* (grade group 5). The resulted dataset has 318 patients with 129 patients with class 0 and 189 patients with class 1.

Camelyon16 (breast cancer) dataset

The Camelyon16 dataset engages with breast cancer imaging [102]. The aim of this dataset was to detect lymph node metastases among breast cancer patients using hematoxylin and eosin (H&E) stained whole-slide images. It is difficult and time-consuming to detect lymph node metastases with the giga-pixel sized images. It is of interest to develop a procedure for automated detection of breast cancer metastases in lymph node pictures. We use the same pre-processing as we used for the TCGA-PRAD dataset. We also used the pre-trained Resnet50 network [103] to extract features from image tiles (instances) into one-dimensional embedding features of size 1,024. After pre-processing and feature extraction, we are left with 80 patients (bags) with cancerous tissue and 123 patients with normal tissue.

PANDA (prostate cancer) dataset

The aim of the Prostate cANcer graDe Assessment (PANDA) dataset was to use automated tools for diagnosing the prostate cancer [118]. The slide images are from microscopy scans of prostate biopsy samples, which are quite large. We use the same pre-processing as the one used for TCGA-PRAD and Camylyon16 datasets. We also used the pre-trained Resnet50 network [103] to extract features from image tiles (instances) into one-dimensional embedding features with size 1,024. Similar to the TCGA-PRAD dataset, we consider binary classification where we divide the patients into two classes: class 0 includes *very low risk*

(grade group 0) *low risk* (grade group 1) and *favorable intermediate* (grade group 2); and class 1 includes *unfavorable intermediate risk* (grade group 3), *high risk* (grade group 4), and *very high risk* (grade group 5). The resulted dataset has 4643 patients composed of 2916 patients with class 0 and 1727 patients with class 1. To examine the effect of sample size, we randomly sub-sample this dataset to create datasets with sample sizes 200, 500, 1000, and 4643 (i.e., the whole samples).

MNIST dataset

We use the MNIST dataset [64] to simulate multiple-instance learning datasets. The MNIST dataset is a standard benchmark dataset that has been used for multi-class classification purposes. It contains $n_{train} = 60,000$ and $n_{test} = 10,000$ grey-scale images with size 28×28 for training and testing. These images correspond to digit numbers between 0 and 9. The reason to choose MNIST dataset is that one can get an accuracy of over 99% for a simple and non-MIL classification problem even with simple network architectures [119]. We consider digit 1 as the positive instance and all other digits (i.e., 0 and 2-9) as the negative instance. We generate two different datasets using the MNIST dataset as follows.

Dataset 1: The first dataset contains no pure noisy instances. This might be the case where one uses a pre-processing step to exclude pure noisy images (e.g., background noise) from the bags. This dataset only contains positive instances, negative instances, and partial instances cropped from both positive and negative instances. We generate one sample (bag) from this dataset as follows:

(a) With a probability of 0.5, we decide whether the bag has a positive label, i.e., contains positive instances (i.e., images from digit 1),

(a.1) if the bag has a positive label, we randomly select $10 - 50$ images from digit 1 (i.e., positive instances) and $10 - 50$ images from other digits (negative instances),

(a.1) if the bag has a negative label, we randomly select $20-100$ images from other digits (i.e., negative instances).

(b) We randomly select $1000-5000$ partial images from all digits 0-9. Such partial images contain a maximum 25% of the randomly cropped regions of digits. We assign a value of 0 to the pixels in non-cropped regions.

Dataset 2: The aim of this dataset is to increase the bag size by adding pure noisy images. This might be the case where a pre-processing step is not available or used. For generating this dataset, we add the following step:

(c) We randomly generate $5000-45000$ pure noisy images whose pixels get a random value between 0 and 1 from the uniform distribution, $U(0,1)$.

To examine the effect of sample size, we consider generating datasets with 200, 500, and 1000 samples.

4.4.2 Network architecture

First, we consider a fully connected layer $\boldsymbol{W}_d \in \mathbb{R}^{d\times 512}$ with ReLU activation function to reduce the dimension of feature embedding space from d ($d = 784$ for MNIST-based dataset and $d = 1024$ for other three datasets) to 512. For the attention network, we consider the gated attention with $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{256\times 512}$, each followed by a single shared branch $\boldsymbol{w} \in \mathbb{R}^{256\times 1}$. For the classification network, we choose a fully connected layer $\boldsymbol{W}_c \in \mathbb{R}^{512\times C}$ where we choose $C = 2$ for binary classification. We use the Adam algorithm [120] to optimize the parameters of the deep neural network for all methods.

4.4.3 Tuning hyper-parameters

To find the best possible model for classification, we consider different hyper-parameters for all methods evaluated in this chapter: initial learning rate values $(10^{-4}, 10^{-3})$, regularization rate $(10^{-5}, 10^{-3})$, and dropout rate $(0.2, 0.5)$. We randomly split the data into

training/validation/testing datasets (80% training, 10% validation, and 10% training). For each combination of hyper-parameters, we train a model on the training dataset until there is no improvement on the validation AUC. We use a stopping criterion [121] with *patience*=10 epochs (after which there will be no later improvement in AUC on the validation dataset) to determine when to stop training. We use the best model maximizing the validation AUC and report testing AUC.

4.5 Results

We consider the binary classification task and compare four sampling strategies: (1) no sampling where we use whole instances over iterates (this is the standard attention MIL in [17] and referred to as CLAM-MIL in [108]); (2) random sampling where we randomly draw G instances, (3) adaptive sampling where we adaptively select G instances; and (4) top-k sampling where we choose $k = G$ instances with the highest attention weights. We evaluate different strategies with both TCGA-PRAD and Camelyon16 datasets. We use the same network architecture, hyper-parameters, and tuning procedure for all methods. For all methods, we choose the minimum number of epochs as $e_{min} = 50$, maximum number of epochs as $e_{max} = 300$, and patience $e_{patience} = 20$ for early stopping. For random and adaptive sampling methods, we consider ten warm-up epochs ($e_{warm} = 10$) to initially train the model using the whole instances. After that, we pick $G = 10$ ($\sim 0.2\%$ of all available instances), 30, 100, 300, and 1000. We conducted all experiments for TCGA-PRAD and Camelyon16 on AWS nodes with one NVIDIA Tesla T4 GPU node, 32 CPUs, and 235 GB memory. For all datasets, we consider ten repetitions of Monte Carlo simulations for splitting data into training/validation/testing and we report *mean* $\pm$ *Standard error (SE).*

Figure 4.2 compares the testing AUC, training time, and the number of training epochs stopping after training by the early stopping algorithm (see Appendix A.10.3 for more details) for TCGA-PRAD (left panel) and Camelyon16 (right panel).

We observe that all instance sampling strategies reduce the computational complexity as expected. Also, we observe that random instance sampling with enough selected instances

(e.g., around $G = 100$ or more) outperforms no sampling (i.e., using whole instances) strategy. Adaptive sampling might do better than random instance sampling when the number of instances per patient (bag) is minimal (around $G = 10$ or less) due to, e.g., memory constraints. Top-k sampling strategy performs the worst. From the results, we discover an important fact about using sampling strategies for the attention-based MIL networks: instance sampling strategies (versus using all instances) not only save computing time and resources, but can also improve the performance of the patients' disease status with WSI's.

Both TCGA-PRAD and Camelyon16 datasets have small sample sizes (318 and 203 patients). We consider the PANDA dataset (with a total size of 4643) to examine another real dataset and also to explore the effect of sample size. Figures 4.3 and 4.4 illustrate the testing AUC, training time, and the number of training epochs (with the early stopping algorithm; see Appendix A.10.3 for more details) for the PANDA dataset with different sample sizes 200, 500, 1000, and 4643. We observe the same patterns here as well: the random instance sampling with enough selected instances (e.g., around $G = 100$ or more) outperforms other strategies including no sampling (i.e., using whole instances). Adaptive sampling outperforms random instance sampling when the number of instances per patient is limited (around $G = 10$ or less) due to, e.g., memory constraints. Top-k sampling strategy performs the worst. As a natural observation, increasing sample size improves the performance of all sampling strategies.

Now we consider two synthesized datasets *Dataset 1* (with no additional pure noisy images) and *Dataset 2* (with additional pure noisy images), both generated from the MNIST dataset. Figures 4.5, 4.6, and 4.7 illustrates the testing AUC, training time, and the number of training epochs (with the early stopping algorithm; see Appendix A.10.3 for more details) for MNIST-based generated datasets with sizes 200, 500, and 1000. The left panels present the results for *Dataset 1* and the right panels present the results for *Dataset 2*. For *Dataset 1*, all strategies perform similarly to each other except for the random sampling strategy for which the network has a harder task when it only selects one instance (i.e., $G = 1$). Adaptive sampling strategies perform very well: they correctly predict the bag label even

with one selected instance (i.e., $G = 1$) while reducing the training time. For *Dataset 2* (with added pure noisy images), all sampling strategies train the neural network almost four times faster than no sampling (i.e., using all samples). The adaptive sampling strategies perform very well, even with one selected instance (i.e., $G = 1$). However, with additional pure noisy images, the random sampling strategy needs more selected instances ($G \approx 30$ or more) to predict the bag labels correctly. The random sampling strategy requires fewer selected samples for the correct classification if sample size increases.

4.6 Discussion

We proposed an adaptive sampling strategy for training the attention-based MIL networks. We also compared our proposed adaptive sampling strategy with other sampling strategies using both real and synthesized (MNIST-based) MIL datasets. The adaptive sampling strategy significantly reduces computing time (and hence resources) while having a negligible performance loss, especially when we can only select fewer instances (in order of 10 or less). We discovered that, if we select more instances (in order of 100 or more), the random sampling outperforms other sampling strategies, including no sampling (i.e., using all instances) strategy. We can justify this observation as follows. With random sampling, the neural network sees almost different subsets of instances for each bag (patient) over different iterates (epochs). This behaves similarly to regularization or augmentation to avoid over-fitting, especially for small datasets.

We used the pre-processing to exclude noisy (e.g., background) tiles or less-informative tiles beforehand. Then we used a pre-trained network (e.g., Resnet50) to extract low dimensional features from remaining image tiles after pre-processing. The reason to use a pre-trained network was to go to lower-dimensional feature space where comparison with no sampling (i.e., using all instances) becomes feasible. One issue with using a pre-trained (e.g., Resnet50) is that the domain of knowledge might be different. Resnet50 used ImageNet classification dataset [103] to train the model, which differs from WSIs. It is worth exploring more complicated scenarios, e.g., fine-tuning some layers of the pre-trained network to ex-

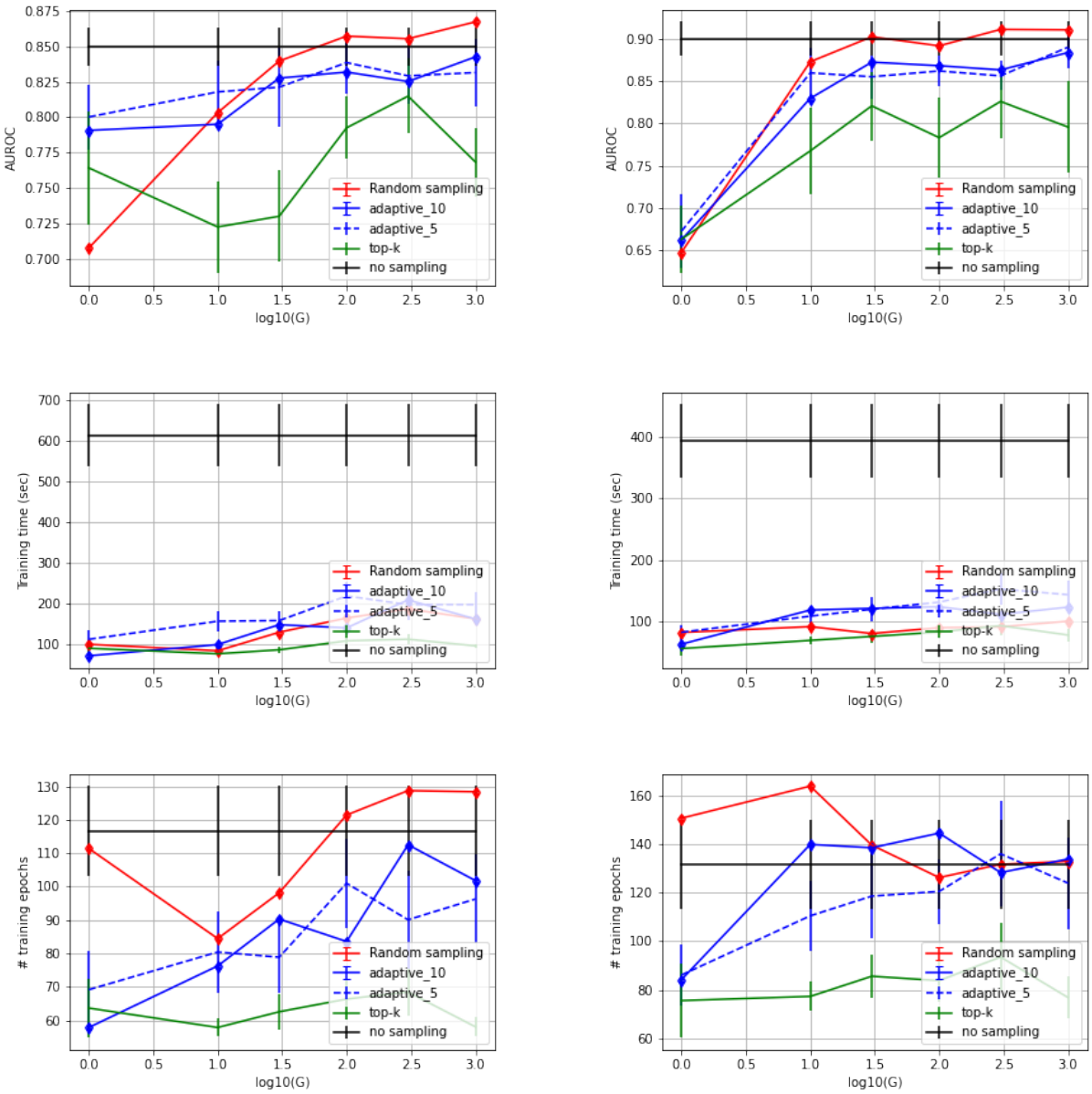

Figure 4.2: (left column) TCGA-PRAD (right column) Camelyon16; (top) Area under ROC curve, (middle) training time, and (bottom) number of training epochs; We compare different sampling strategies: no sampling, random sampling, adaptive sampling, and top-k sampling.

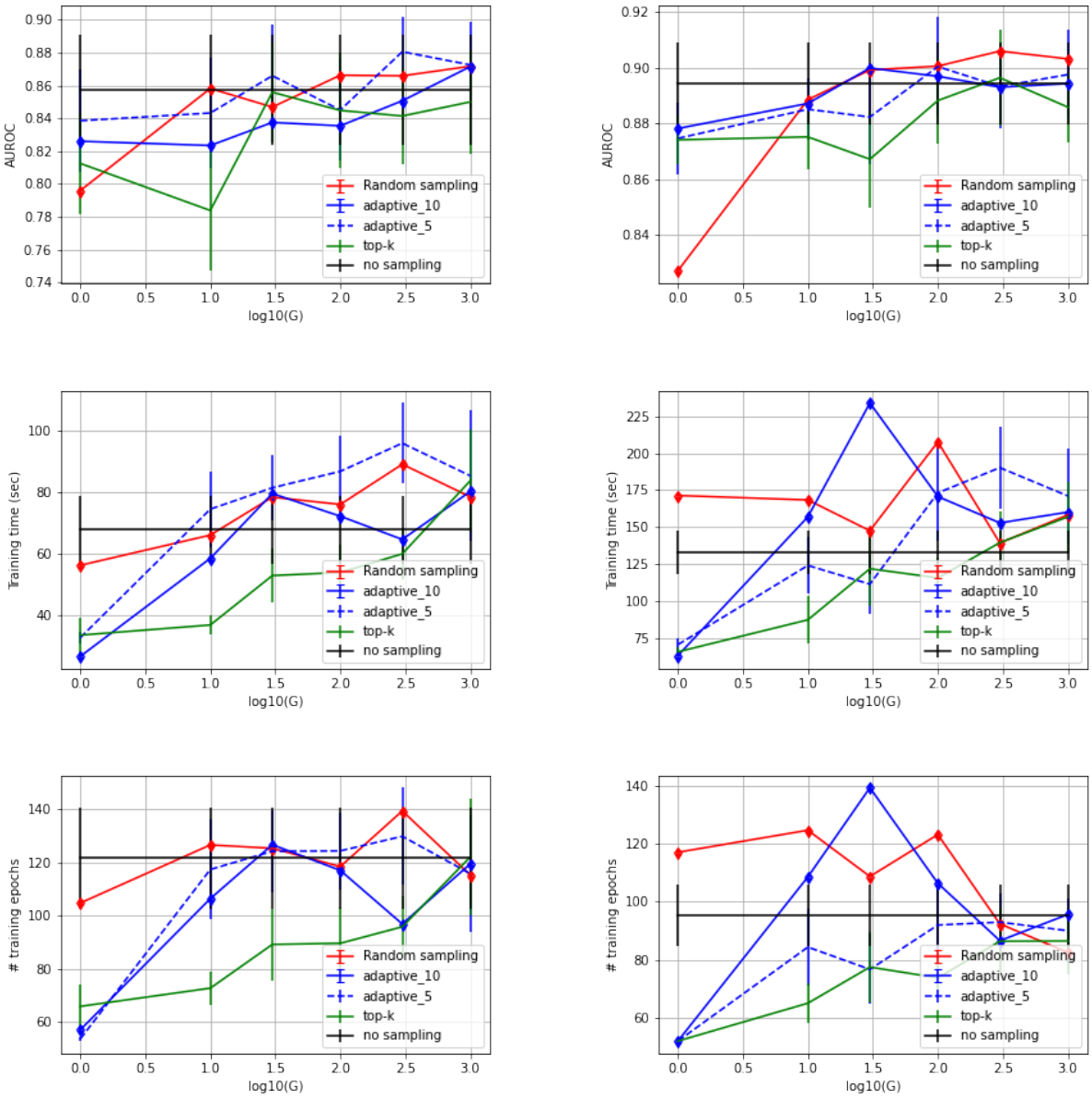

Figure 4.3: (left column) PANDA with $N = 200$ (right column) PANDA with $N = 500$; (top) Area under ROC curve, (middle) training time, and (bottom) number of training epochs; We compare different sampling strategies: no sampling, random sampling, adaptive sampling, and top-k sampling.

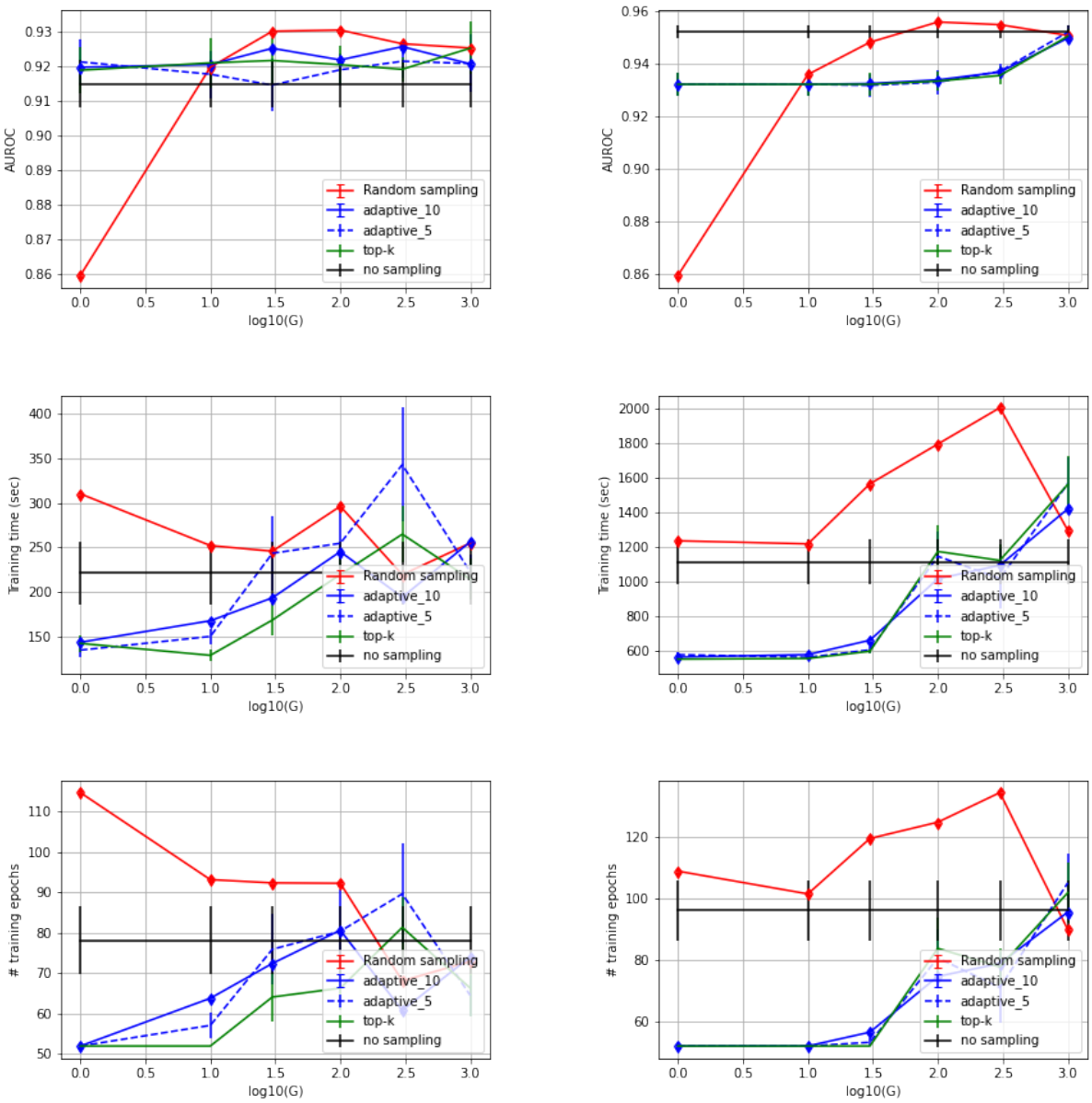

Figure 4.4: (left column) PANDA with $N = 1000$ (right column) PANDA with $N = 4643$; (top) Area under ROC curve, (middle) training time, and (bottom) number of training epochs; We compare different sampling strategies: no sampling, random sampling, adaptive sampling, and top-k sampling.

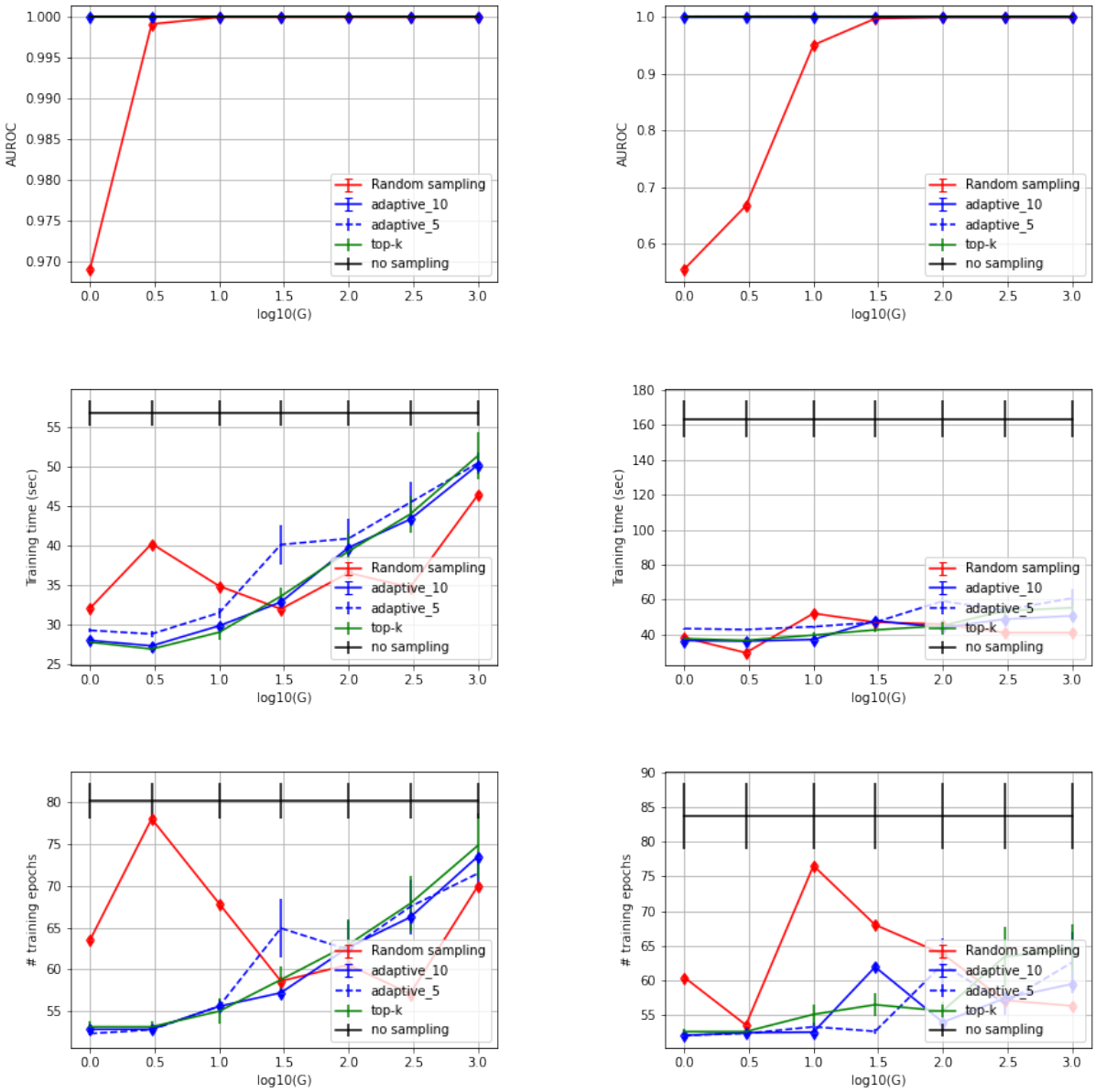

Figure 4.5: (left column) MNIST, without additional pure noisy images, $N = 200$ (right column) MNIST, with additional pure noisy images, $N = 200$; (top) Area under ROC curve, (middle) training time, and (bottom) number of training epochs; We compare different sampling strategies: no sampling, random sampling, adaptive sampling, and top-k sampling.

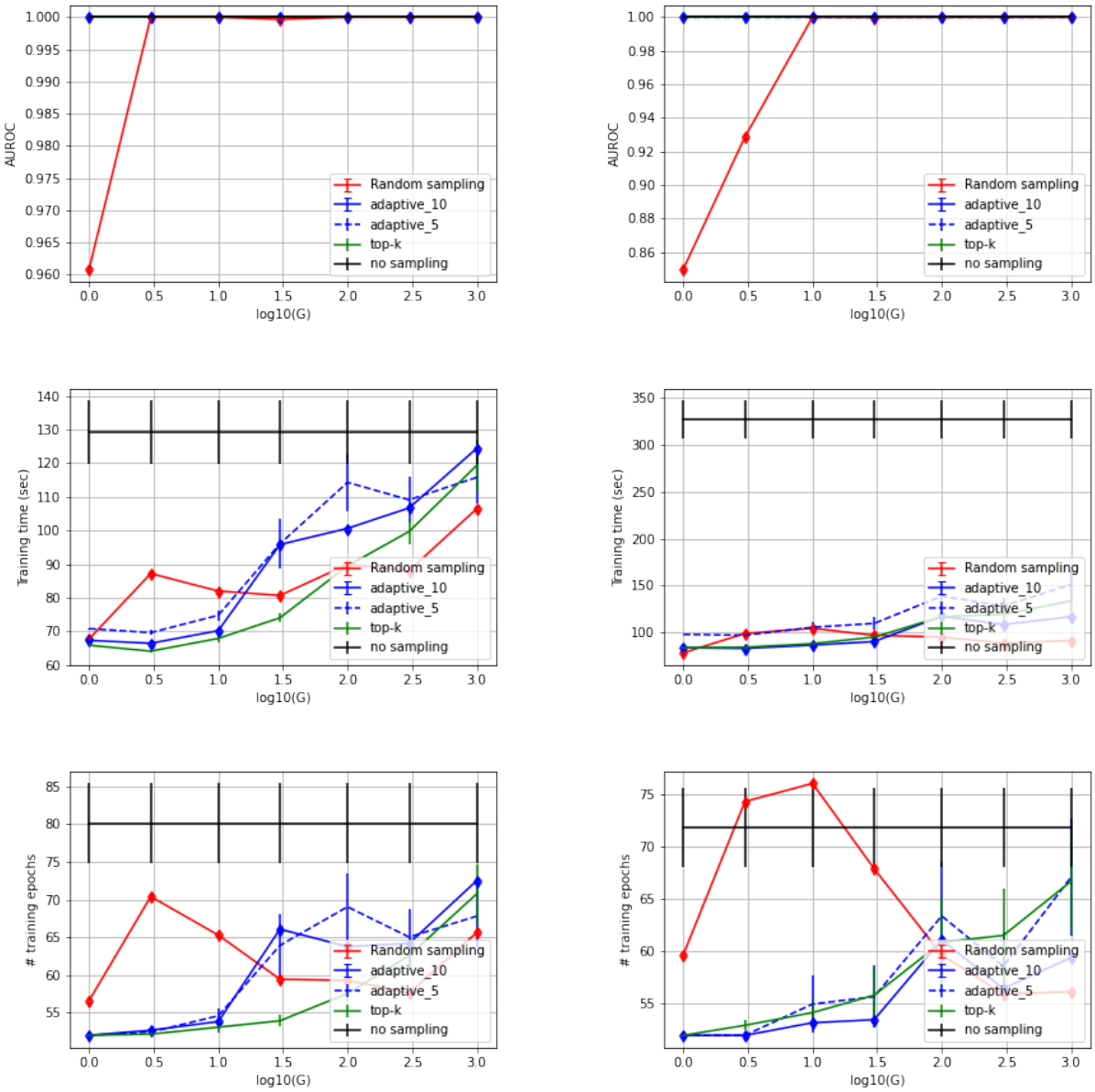

Figure 4.6: (left column) MNIST, without additional pure noisy images, $N = 500$ (right column) MNIST, with additional pure noisy images, $N = 500$; (top) Area under ROC curve, (middle) training time, and (bottom) number of training epochs; We compare different sampling strategies: no sampling, random sampling, adaptive sampling, and top-k sampling.

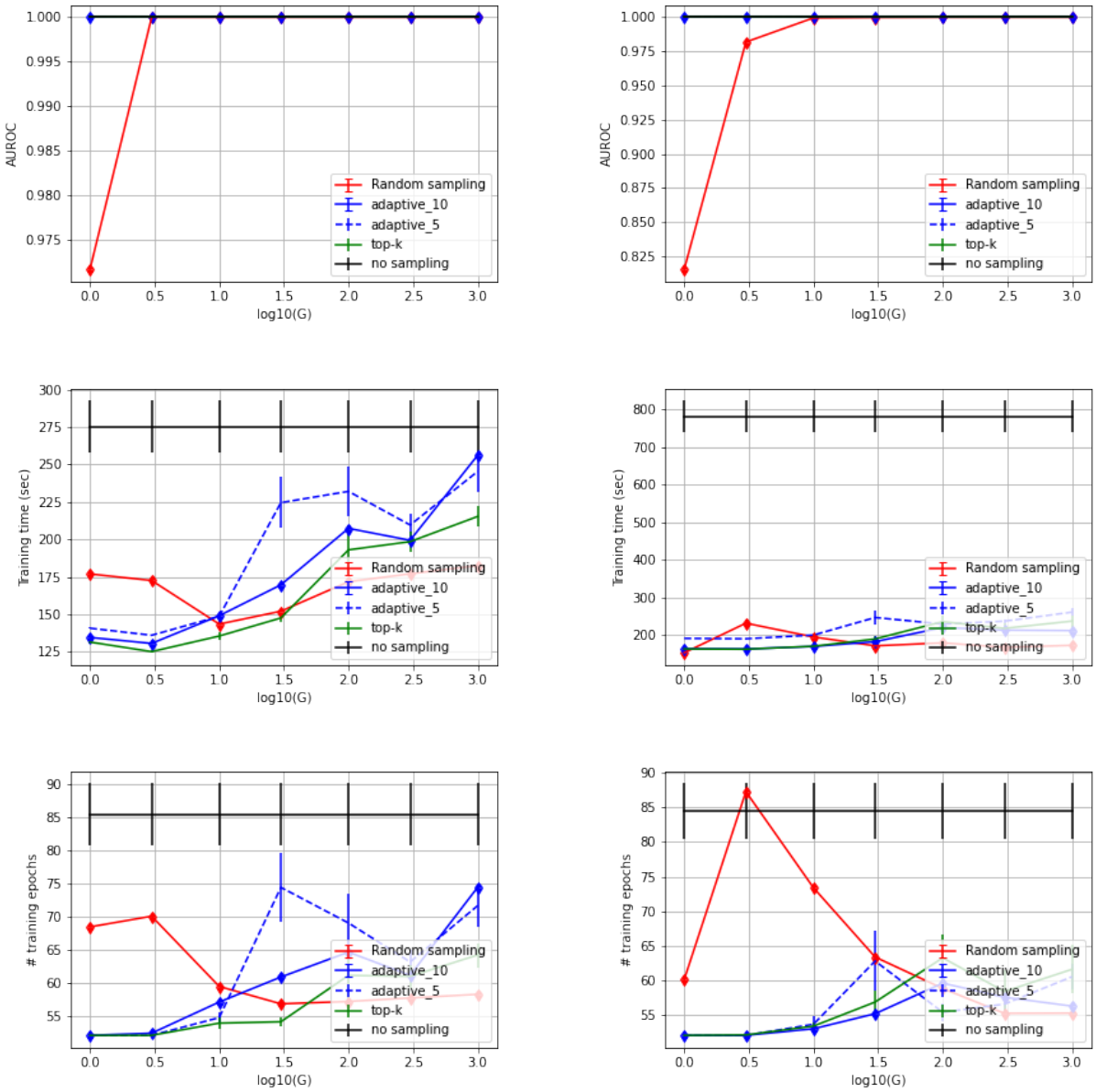

Figure 4.7: (left column) MNIST, without additional pure noisy images, $N = 1000$ (right column) MNIST, with additional pure noisy images, $N = 1000$; (top) Area under ROC curve, (middle) training time, and (bottom) number of training epochs; We compare different sampling strategies: no sampling, random sampling, adaptive sampling, and top-k sampling.

tract more distinct and predictive features out of sampled tiles. By noting that the random sampling strategy can outperform other strategies with enough number of selected instances, one can select the number of instances (e.g., $G = 100$), and use mini-batch stochastic gradient descent with the original images instead of extracted ones from a pre-trained network.

The idea of adaptive sampling strategies is quite general: it aims to reduce computing time with bags with tens of thousands instances. We considered a binary classification problem to show how it performs under different scenarios. However, it is worth extending training the attention-based MIL network using different sampling strategies to other tasks such as multi-class classification (e.g., the framework presented in [108]) or survival analysis [12, 11, 122].

Chapter 5

CONCLUSION

In this dissertation, we proposed a simple framework for conducting large-scale survival analysis using a Cox model. Our framework leverages a modified population optimization problem, which allows us to leverage the tools of stochastic algorithms such as gradient descent, proximal stochastic gradient descent and their extensions. Our proposed framework results in more computationally efficient and stable algorithms than the current state-of-the-art Cox model when the dataset is big (either in terms of the number of observations or the number of features) or the Cox model suffers from round-off errors. We developed both plug-in and bootstrap-based techniques to construct confidence intervals for the parameters estimated by our framework. In addition, we proposed a neural network-based hazard estimator for conducting survival prediction for both imaging and non-imaging survival data.

We extended our proposed framework for fitting the penalized Cox model using large and ultrahigh-dimensional datasets where we employed the proximal stochastic gradient descent algorithm. We showed that our proposed framework performs well when the number of observations and features grows, particularly when the datasets do not fit into the memory. We proposed an adaptive instance sampling strategy for attention-based multiple-instance learning (MIL) networks. The proposed strategy reduces computing time while resulting in a negligible performance loss. We showed that our suggested adaptive framework outperforms alternative sampling strategies when the number of sub-selected instances is modest. However, if we sub-select more instances, the random sampling outperforms other sampling strategies, including no sampling strategy (i.e., when all instances are used).

BIBLIOGRAPHY

[1] E. T. Lee and O. T. Go. Survival analysis in public health research. *Annual Reviews in Public Health*, 18:105–134, 1997.

[2] P. Schober and T. R. Vetter. Survival analysis and interpretation of time-to-event data: The tortoise and the hare. *Anesthesia & Analgesia*, 127(3):792–798, 2018.

[3] D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220, 1972.

[4] W. Raghupathi and V. Raghupathi. Big data analytics in healthcare: promise and potential. *Journal of Health Information Science and Systems*, 2(3):1–10, 2014.

[5] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for cox's proportional hazards model via coordinate descent. *Journal of Statistical Software*, 39(5):1–13, 2011.

[6] A. Tarkhan and N. Simon. Big survival data analysis via stochastic gradient descent. `https://github.com/atarkhan/bigSurvSGD`, 2020.

[7] S. Ruder. An overview of gradient descent optimization algorithms. eprint arXiv:1609.04747, 2016.

[8] R. Tibshirani. The lasso method for variable selection in the cox model. *Statistics in Medicine*, 16(4):385–395, 1997.

[9] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[10] M. Y. Park and T. Hastie. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):659–677, 2007.

[11] Aliasghar Tarkhan, Noah Simon, Thomas Bengtsson, Kien Nguyen, and Jian Dai. Survival prediction using deep learning. In *Proc. of AAAI Spring Symposium on Survival Prediction - Algorithms, Challenges, and Applications 2021*, volume 146 of *Proc. of Machine Learning Research*, pages 207–214. PMLR, 22–24 Mar 2021.

[12] Aliasghar Tarkhan and Noah Simon. An online framework for survival analysis: reframing Cox proportional hazards model for large data sets and neural networks. *Biostatistics*, 10 2022. kxac039.

[13] Filippo Fraggetta, Salvatore Garozzo, Gian F. Zannoni, Liron Pantanowitz, and Esther D. Rossi. Routine digital pathology workflow: the catania experience. *J Pathol Inform.*, 8(51):1–6, Dec 2017.

[14] Jonathan I. Epstein. An update of the gleason grading system. *J. Urol*, 183(2):433–440, 2010.

[15] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1):31–71, 1997.

[16] Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998.

[17] Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based deep multiple instance learning, 2018.

[18] Aliasghar Tarkhan, Trung Kien Nguyen, Noah Simon, Thomas Bengtsson, Paolo Ocampo, and Jian Dai. Attention-based deep multiple instance learning with adaptive instance sampling. In *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, pages 1–5, 2022.

[19] Aliasghar Tarkhan, Trung Kien Nguyen, Noah Simon, and Jian Dai. Investigation of training multiple instance learning networks with instance sampling. In Xinxing Xu, Xiaomeng Li, Dwarikanath Mahapatra, Li Cheng, Caroline Petitjean, and Huazhu Fu, editors, *Resource-Efficient Medical Image Analysis*, pages 95–104, Cham, 2022. Springer Nature Switzerland.

[20] B. Zethelius, L. Berglund, J. Sundstrom, , E. Ingelsson, S. Basu, A. Larsson, P. Venge, , and J. Arnlov. Use of multiple biomarkers to improve the prediction of death from cardiovascular causes. *The New England Journal of Medicine*, 358(20):2107–2116, 2008.

[21] S. Mittal and D. Madigan. High-dimensional, massive sample-size cox proportional hazards regression for survival analysis. *Biostatistics*, 15(2):207–221, 2014.

[22] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

[23] C. C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer, 2018.

[24] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: A review. *ACM Digital Library*, 34(2):18–26, 2005.

[25] L. J. Wei. The accelerated failure time model: a useful alternative to the cox regression model in survival analysis. *Statistics in Medicine*, 11:1871–1879, 1992.

[26] P. Chapfuwa, C. Tao, C. Li, C. Page, B. Goldstein, L. Carin, and R. Henao. Adversarial time-to-event modeling. *In ICML*, 2018.

[27] H. Wang, M. Yang, and J. Stufken. Information-based optimal subdata selection for big data linear regression. *J. of the American Statistical Association*, 525(114):393–405, 2019.

[28] Y. Wang, N. Palmer, Q. Di, J. Schwartz, I. Kohane, and T. Cai. A fast divide-and conquer sparse cox regression. *Biostatistics*, pages 1–21, 2019.

[29] P. Toulis and E. M. Airoldi. Asymptotic and finite-sample properties of estimators based on stochastic gradients. *The Annals of Statistics*, 45(4):1694–1727, 2017.

[30] V. C. Raykar, H. Steck, B. Krishnapuram, C. Dehing-Oberije, and P. Lambin. On ranking in survival analysis: Bounds on the concordance index. *In NeurIPS*, 2008.

[31] P. C. Austin and E. W. Steyerberg. Interpreting the concordance statistic of a logistic regression model: relation to the variance and odds ratio of a continuous explanatory variable. *BMC Medical Research Methodology*, 12(82):1–8, 2012.

[32] J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger. Deepsurv: Personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18(4), 2018.

[33] T. Ching, X. Zhu, and L. X. Garmire. Cox-nnet: An artificial neural network method for prognosis prediction of high-throughput omics data. *Plus Comp. Bio.*, 14(4):1–18, 2018.

[34] C. Lee, W. R. Zame, J. Yoon, and M. van der Schaar. Deephit: A deep learning approach to survival analysis with competing risks. *32nd AAAI Conf. on Artificial Intelligence*, 2018.

[35] H. Kvamme and O. Borgan. Continuous and discrete-time survival prediction with neural networks. eprint arXiv:1910.06724, 2019.

[36] S. Fotso. Deep neural networks for survival analysis based on a multi-task framework. eprint arXiv:1801.05512, 2018.

[37] H. Kvamme, O. Borgan, and I. Scheel. Time-to-event prediction with neural networks and cox regression. *Journal of Machine Learning Research*, 20(129):1–30, 2019.

[38] J. Fan, Y. Guo, and K. Wang. Communication-efficient accurate statistical estimation. eprint arXiv:1906.04870, 2021.

[39] J. P. Klein and M. L. Moeschberger. *Survival Analysis: Techniques for Censored and Truncated Data*. Springer, New York, USA, 2003.

[40] T. M. Therneau and T. Lumley. coxph: Core survival analysis routines, 2019. R package version 2.44-1.1.

[41] A. W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, UK, 2000.

[42] N. E. Breslow and N. E. Day. Statistical methods in cancer research.vol 1. the analysis of case-control studies. *International Agency for Research on Cancer*, 1(32):5–338, 1980.

[43] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. *Proceedings of the 32nd International Conference on Machine Learning*, 37, 2015.

[44] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[45] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[46] D. Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, School of Operations Research and Industrial Eng., Cornell Uni., NY, USA, 1988.

[47] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[48] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Paris, France, 2010. Springer.

[49] F. R. Bach and E. Moulines. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. *In NeurIPS*, pages 451—-459, 2011.

[50] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. eprint arXiv:1904.09237, 2019.

[51] Aad W. Vaart and Jon A. Wellner. *Weak Convergence and Empirical Processes With Applications to Statistics*. Springer Series in Statistics (SSS), USA, 2000.

[52] G. Blom. Some properties of incomplete u-statistics. *Biometrika*, 66:495–505, 1976.

[53] B. E. Honore and J. L. Powell. Pairwise difference estimators of censored and truncated regression models. *Journal of Econometrics*, 64(1):241–278, 1994.

[54] R. P. Sherman. Maximal inequalities for degenerate u-processes with applications to optimization estimators. *The Annals of Statistics*, 22(1):439–459, 1994.

[55] Qiyang Han. Multiplier u-processes: Sharp bounds and applications. *Bernoulli*, 2022.

[56] R. Bender, T. Augustin, and M. Blettner. Generating survival times to simulate cox proportional hazards models. *Statistics in medicine*, 24(11):1713–1723, 2005.

[57] Rand Wilcox. *Introduction to robust estimation and hypothesis testing, 4th Edition*. Academic Press, San Diego, CA, 2017.

[58] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. Random survival forests. *Annals of Applied Statistics*, 2(3):841–860, 2008.

[59] R. Kyle, T. Therneau, S. V. Rajkumar, D. Larson, M. Plevak, J. Offord, A. Dispenzieri, J. Katzmann, and L. J. Melton. Prevalence of monoclonal gammopathy of undetermined significance. *New England Journal of Medicine*, 354:1362–1369, 2006.

[60] M. Schumacher, G. Bastert, H. Bojar H, K. Hubner, M. Olschewski, W. Sauerbrei, C. Schmoor, C. Beyerle, R. L. Neumann, and H. F. Rauschecker. Randomized 2 x 2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. german breast cancer study group. *Journal of Clinical Oncology*, 12(10):2086–2093, 1994.

[61] B. Pereira and et al. The somatic mutation profiles of 2,433 breast cancers refines their genomic and transcriptomic landscapes. *Nature Communication*, 7:1–15, 2016.

[62] N. E. Breslow and N Chatterjee. Design and analysis of two-phase studies with binary outcome applied to wilms tumour prognosis. *Applied Statistics*, 48:457–68, 1999.

[63] W. A. Knaus and et al. The support prognostic model: Objective estimates of survival for seriously ill hospitalized adults. *Annals of Internal Medicine*, 122:191–203, 1995.

[64] L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[65] Aliasghar Tarkhan and Noah Simon. *bigSurvSGD: Big Survival Analysis Using Stochastic Gradient Descent*, 2020. `https://cran.r-project.org/web/packages/bigSurvSGD/index.html`).

[66] A. Tarkhan and N. Simon. Survival prediction using neural network. `https://github.com/atarkhan/bigSurvNN`, 2021.

[67] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series*, 58(1):267–288, 1996.

[68] I. Sohn, J. Kim, S.-H. Jung, and C. Park. Gradient lasso for cox proportional hazards model. *Bioinformatics*, 25(14):1775–1781, 2010.

[69] J. J. Goeman. L1 penalized estimation in the cox proportional hazards model. *Bioinformatics*, 52(1):70–84, 2010.

[70] H. H. Huang and Y. Liang. Hybrid l1/2+2 method for gene selection in the cox proportional hazards model. *The Annals of Statistics*, 164:65–73, 2018.

[71] Ruilin Li, Christopher Chang, Johanne M Justesen, Yosuke Tanigawa, Junyang Qiang, Trevor Hastie, Manuel A Rivas, and Robert Tibshirani. Fast Lasso method for large-scale and ultrahigh-dimensional Cox model with applications to UK Biobank. *Biostatistics*, 09 2020.

[72] J. Qian, Y. Tanigawa, W. Du, M. Aguirre, C. Chang, R. Tibshirani, M. A. Rivas, and T. Hastie. A fast and scalable framework for large-scale and ultrahigh-dimensional sparse regression with application to the uk biobank. *PLOS Genetics*, 6(10):1–30, 2020.

[73] R. Tibshirani, J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor, and R. J. Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, 74(2):245–266, 2012.

[74] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[75] Jerome Friedman, Trevor Hastie, Rob Tibshirani, Balasubramanian Narasimhan, Kenneth Tay, Noah Simon, Junyang Qian, and James Yang. *glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models*, 2021. `https://cran.r-project.org/web/packages/glmnet/index.html`).

[76] Albert Benveniste, Michel Metivier, and Pierre Priouret. *Adaptive Algorithms and Stochastic Approximations*. Part of the Applications of Mathematics book series (SMAP, volume 22), Springer-Verlag, Berlin, 1990.

[77] Olivier Devolder. Stochastic first order methods in smooth convex optimization. Technical report, Center for Operations Research and Econometrics, Belgium, 2011.

[78] Christine De Mol, Ernesto De Vito, and Lorenzo Rosasco. Elastic-net regularization in learning theory. *Journal of Complexity*, 25(2):201–230, 2009.

[79] Yu. Nesterov. Gradient methods for minimizing composite objective function. Technical report, CORE DISCUSSION PAPER 2007/76, Catholic University of Louvain, Belgium, 2007.

[80] Saeed Ghadimi and Guanghui Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework. *SIAM Journal on Optimization*, 22(4):1469–1492, 2012.

[81] Chonghai Hu, Weike Pan, and James Kwok. Accelerated gradient methods for stochastic optimization and online learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[82] Patrick L. Combettes and Valérie R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[83] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Part of the CMS Books in Mathematics book series (CMSBM), New York, USA, 1990.

[84] J. Lee. *U-statistics: Theory and Practice*. Marcel Dekker, Inc., New York, FL, USA, 1990.

[85] Xiao Song and Shuangge Ma. Penalised variable selection with u-estimates. *Journal of Nonparametric Statistics*, 22(4):499–515, 2010.

[86] Yu Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, may 2005.

[87] Lorenzo Rosasco, Silvia Villa, and Bang Công Vũ. Convergence of stochastic proximal gradient algorithm, 2014.

[88] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10(99):2899–2934, 2009.

[89] Cun-Hui Zhang and Jian Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *The Annals of Statistics*, 36(4), 2008.

[90] Adel Javanmard and Andrea Montanari. De-biasing the lasso: Optimal sample size for gaussian designs, 2015.

[91] Sebastian Otálora, Niccolò Marini, Henning Müller, and Manfredo Atzori. Combining weakly and strongly supervised learning improves strong supervision in gleason pattern classification. *BMC Med Imaging*, 21(77):1–14, 2021.

[92] Tad T. Brunyé, Ezgi Mercan, Donald L. Weaver, and Joann G. Elmore. Accuracy is in the eyes of the pathologist: The visual interpretive process and diagnostic accuracy with digital whole slide images. *J. of Biomed. Info.*, 66:171–179, 2010.

[93] Miao Cui and David Y. Zhang. Artificial intelligence and computational pathology. *Laboratory Investigation*, 101:412–422, 2021.

[94] Sevda Molani, Mahboubeh Madadi, and Wesley Wilkes. A partially observable markov chain framework to estimate overdiagnosis risk in breast cancer screening: Incorporating uncertainty in patients adherence behaviors. *Omega*, 89:40–53, 2019.

[95] Shima Nofallah, Sachin Mehta, Ezgi Mercan, Stevan Knezevich, Caitlin J. May, Donald Weaver, Daniela Witten, Joann G. Elmore, and Linda Shapiro. Machine learning techniques for mitoses classification. *Computerized Medical Imaging and Graphics*, 87:101832, 2021.

[96] Gwenole Quellec, Guy Cazuguel, Beatrice Cochener, and Mathieu Lamard. Multiple-instance learning for medical image and video analysis. *IEEE Reviews in Biomedical Engineering*, 10:213–234, 2017.

[97] Guoqing Liu, Jianxin Wu, and Zhi-Hua Zhou. Key instance detection in multi-instance learning. In *Proc. of the Asian Conference on Machine Learning*, volume 25 of *Proc. of Machine Learning Research*, pages 253–268. PMLR, 2012.

[98] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[99] Colin Raffel and Daniel P. W. Ellis. Feed-forward networks with attention can solve some long-term memory problems, 2016.

[100] Jan Ramon and Luc D. Raedt. In *Multi instance neural networks*, ICML Workshop on Attribute-value and Relational Learning, pages 53–60, 2000.

[101] Margarita L. Zuley, Rose Jarosz, Bettina F. Drake, Danielle Rancilio, Aleksandra Klim, Kimberly Rieger-Christ, and John Lemmerman. Radiology data from the cancer genome atlas prostate adenocarcinoma [tcga-prad] collection. *Cancer Imaging Arch*, 2016.

[102] Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes van Diest, Bram van Ginneken, Nico Karssemeijer, Geert Litjens, Jeroen A. W. M. van der Laak, and the CAMELYON16 Consortium. Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer. *JAMA*, 318(22):2199–2210, 12 2017.

[103] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[104] Xinliang Zhu, Jiawen Yao, and Junzhou Huang. Deep convolutional neural network for survival analysis with pathological images. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 544–547, 2016.

[105] Ellery Wulczyn, David F. Steiner, Zhaoyang Xu, Apaar Sadhwani, Hongwu Wang, Isabelle Flament-Auvigne, Craig H. Mermel, Po-Hsuan Cameron Chen, Yun Liu, and Martin C. Stumpe. Deep learning-based survival prediction for multiple cancer types using histopathology images. *PLOS ONE*, 15(6):e0233678, Jun 2020.

[106] Ruoyu Li, Jiawen Yao, Xinliang Zhu, Yeqing Li, and Junzhou Huang. Graph cnn for survival analysis on whole slide pathological images. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, pages 174–182, Cham, 2018. Springer International Publishing.

[107] Chris M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995.

[108] Ming Y. Luand Drew F. K. Williamson, Tiffany Y. Chen, Richard J. Chen, Matteo Barbieri, and Faisal Mahmood. Data-efficient and weakly supervised computational pathology on whole-slide images. *Nat Biomed Eng*, 5:555—-570, 2021.

[109] Ming Y. Lu, Richard J. Chen, Jingwen Wang, Debora Dillon, and Faisal Mahmood. Semi-supervised histology classification using deep multiple instance learning and contrastive predictive coding, 2019.

[110] Olivier Dehaene, Axel Camara, Olivier Moindrot, Axel de Lavergne, and Pierre Courtiol. Self-supervision closes the gap between weak and strong supervision in histology, 2020.

[111] Angelos Katharopoulos and François Fleuret. Processing megapixel images with deep attention-sampling models, 2019.

[112] Gabriele Campanella, Matthew G. Hanna, Luke Geneslaw, Allen P. Miraflor, Vitor Werneck Krauss Silva, Klaus J. Busam, Edi Brogi, Victor E. Reuter, David S. Klimstra, and Thomas J. Fuchs. Clinical-grade computational pathology using weakly supervised deep learning on whole slide images. *Nature Medicine*, pages 1–9, 2019.

[113] Yash Sharmay, Lubaina Ehsany, Sana Syed, and Donald E. Brown. Histotransfer: Understanding transfer learning for histopathology. In *2021 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*, pages 1–4, 2021.

[114] Donald F. Gleason and George T. Mellinger. Prediction of prognosis for prostatic adenocarcinoma by combined histological grading and clinical staging. *The Journal of Urology*, 111(1):58–64, 1974.

[115] NCCN. Nccn guidelines: Prostate cancer (version 4.2018). `https://www2.tri-kobe.org/nccn/guideline/archive/urological2018/english/prostate.pdf`, 2018. Accessed: 2021-11-11.

[116] Pegah Khosravi, Maria Lysandrou, Mahmoud Eljalby, Qianzi Li, Ehsan Kazemi, Pantelis Zisimopoulos, Alexandros Sigaras, Matthew Brendel, Josue Barnes, Camir Ricketts, Dmitry Meleshko, Andy Yat, Timothy D. McClure, Brian D. Robinson, Andrea Sboner, Olivier Elemento, Bilal Chughtai, and Iman Hajirasouliha. A deep learning approach to diagnostic classification of prostate cancer using pathology-radiology fusion. *J Magn Reson Imaging*, 54(2):462–471, 2021.

[117] Sinno J. Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[118] Wouter Bulten, Kimmo Kartasalo, Po-Hsuan Cameron Chen, and et al. Artificial intelligence for diagnosis and Gleason grading of prostate cancer: the PANDA challenge. *Nature Medicine*, 28:154–163, 2022.

[119] Dabal Pedamonti. Comparison of non-linear activation functions for deep neural networks on mnist classification task, 2018.

[120] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[121] Lutz Prechelt. *Early Stopping — But When?*, pages 53–67. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[122] J. Yao, X. Zhu, J. Jonnagaddala, N. Hawkins, and Junzhou Huang. Whole slide images based cancer survival prediction using attention guided deep multiple instance learning networks. *Medical Image Analysis*, 65:101789, 2020.

[123] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, jan 2011.

[124] P. K. Andersen, O. Borgan, R. D. Gill, and N. Keiding. *Introduction to Empirical Processes and Semiparametric InferenceWeak*. Springer, NC, USA, 1993.

[125] O. Aalen, O. Borgan, and H. Gjessing. *Survival and Event History Analysis: A Process Point of View*. Springer, New York, USA, 2008.

[126] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, UK, 2004.

[127] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Researc*, 12(7):2121–2159, 2011.

[128] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. eprint arXiv:1412.6980, 2014.

[129] B. Efron. Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[130] P. J. Bickel and D. A. Freedman. Some asymptotic theory for the bootstrap. *The Annals of Statistics*, 9(6):1196–1217, 1981.

[131] Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes (Classics in Mathematics), 1st edition*. Springer-Verlag, Berlin, 2011.

[132] Shahar Mendelson. Upper bounds on product and multiplier empirical processes. *Stochastic Processes and their Applications*, 126(12):3652–3680, 2016. In Memoriam: Evarist Giné.

[133] Jens Praestgaard and Jon A. Wellner. Exchangeably weighted bootstraps of the general empirical process. *Ann. Probab.*, 21(4):2053–2086, 1993.

[134] V. de la Pena and E. Gine. *Decoupling: from Dependence to Independence*. Springer, New York, 1999.

[135] M. R. Kosorok. *Introduction to Empirical Processes and Semiparametric Inference*. Springer, NC, USA, 2007.

[136] Y. Fang, J. Xu, and L. Yang. Online bootstrap confidence intervals for the stochastic gradient descent estimator. *Journal of Machine Learning Research*, 1(19):3053–3073, 2018.

[137] G. Ambler, A. Benner, and S. Luecke. *mfp: an R package for Multivariable Fractional Polynomials*, 2015. R package version 1.5.2).

[138] T. M. Therneau, T. Lumley, A. Elizabeth, and C. Cynthia. *survival: An R package for Survival Analysis*, 2020. R package version 3.1-12).

[139] D. V. Klaveren, M. Gonen, E. W. Steyerberg, and Y. Vergouwe. A new concordance measure for risk prediction models in external validation settings. *Statistics in Medicine*, 35(23):4136–4152, 2016.

[140] C. Caraiscos and Bede Liu. A roundoff error analysis of the lms adaptive algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(1):34–41, 1984.

[141] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appl. Math.*, 57(11):1413–1457, 2004.

Appendix A

ADDITIONAL MATERIALS FOR CHAPTER 2

A.1 Connection with distributed computing

Authors in [38] proposed an alternating direction method of multipliers (ADMM) [123] type of estimation algorithm where the optimization problem is broken into smaller pieces, each easier to handle. Their proposed algorithm is for distributed computing, which is useful with a large amount of data. These methodologies cannot be easily applied to optimizing the standard (whole-cohort) partial likelihood because the corresponding objective function cannot be decoupled. In our work, we proposed another way of formulating the optimization approach for the Cox model using random strata instead of the whole cohort. Therefore, one can consider using distributed optimization tools based on our way of formulating the optimization for the Cox model.

A.2 Extensions for left truncation and right Censoring

In practice, it is common for participants in a study to leave the study before experiencing an event. This is known as right censoring. In particular, it is often assumed that a patient has some random "time until censoring" C, and what we observe is $\min(C, T)$, whichever happens first (the event or censoring), along with $\delta = I(T < C)$ an indicator that the patient experienced an event (rather than censoring). Censored patients still contribute some information to the estimation of $\boldsymbol{\beta}^*$ — in particular, if a patient is censored quite late, then there is a long period of time during which we know they did not experience an event (so likely we should estimate them to be low risk). In some studies, it is common to consider the event time T as the age at which a patient had an event (rather than the calendar on the study). This means that patients do not enroll in the study at $T = 0$ (and patients can

only be observed to fail once they are enrolled). This phenomenon, wherein patients enroll in a study at times other than $T = 0$, is known as left truncation.

When i) the assumptions of the Cox model hold; and ii) censoring and truncation times are independent of event times conditional on covariates $\boldsymbol{x}$, it is relatively straightforward to adapt the Cox model to accommodate these missingness mechanisms. The partial likelihood is modified in 2 minor ways: 1) The outer summation is only taken over indices for which an event occurred (i.e. that were not censored); and 2) The risk sets, R_i, are modified to include only patients currently at risk (who have already been enrolled, and have not yet been censored, or had an event) [124]. We can similarly modify our objective function (5) in Section 2.3.3, and apply our algorithm with only minor modification (a slight change in the gradient).

A.3 Proof of consistency for parameter $\beta^{(s)}$

In this appendix, for the sake of completeness, we prove the Fisher consistency of parameter $\boldsymbol{\beta}^{(s)}$. We treat the Cox model as a counting process [125]. We assume that censoring and survival times are independent given the covariate vector of interest $\boldsymbol{x}$ and they follow model (1) with true parameter $\boldsymbol{\beta}^*$. We first define some terminology before proceeding with the proof.

Definition 1: $dN_i(u)$. For patient i with time to event t_i define the counting process $dN_i(u)$ by

$$\int_a^b g(u)dN_i(u) = \begin{cases} 0 & if \quad t_i \notin [a,b] \\ g(t_i) & if \quad t_i \in [a,b] \end{cases}.$$

For instance, if we define $g(u) = 1$, the above expression is an indicator representing whether patient i failed in interval $[a, b]$ (i.e., 1 represents failure and 0 otherwise). We further define $dN^{(s)}(u) = \sum_{i=1}^{s} dN_i(u)$ which is a counting process for failure times over all s patients. We assume that the failure time process is absolutely continuous w.r.t. Lebesgue measure on time so that there is at most one failure at any time u (i.e., no ties).

Definition 2: $M_i(u)$. We define $M_i(u)$ to be an indicator representing whether patient i is at risk at time u, i.e., $t_i \geq u$. By this definition, $M(u) = \sum_{i=1}^{s} M_i(u)$ indicates number of patients who are at risk at time u. Note that the independent censoring assumption implies that those $M(u)$ patients at risk at time u (who have not yet failed or been censored) represent a random sample of the sub-population of patients who will survive until time u.

Definition 3: $F(u)$. Let $F(u)$ denote the filtration that includes all information up to time u, i.e.,

$$F(u) = \{(dN_i(t), M_i(t), x^{(i)}), i = 1, \ldots, s \quad \text{for } t < u \text{ and } dN^{(s)}(u)\}$$

Note that given $F(u)$, we know whether patient i failed or was censored (i.e., δ_i), when they failed/were censored (i.e., y_i), and their covariate vectors $\boldsymbol{x}^{(i)}$.

Using the above definitions, one can write the log-partial-likelihood as

$$pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)}) = \sum_{i=1}^{s} \int_0^{\tau} \Big\{ f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)}) - log\Big(\sum_{l=1}^{s} M_l(u) exp(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}))\Big)\Big\} dN_i(u) \tag{A.1}$$

where τ is the duration of the study. Note that we keep $f_{\boldsymbol{\beta}}(\boldsymbol{x})$ in the most general form and we only assume that it is differentiable in $\boldsymbol{\beta}$ for all $\boldsymbol{x}$. Then the score function may be written as

$$\begin{aligned}
U^s(\boldsymbol{\beta}) &= \frac{\partial pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})}{\partial \boldsymbol{\beta}} \\
&= \sum_{i=1}^{s} \int_0^{\tau} \Big(\dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)}) - \frac{\sum_{l=1}^{s} M_l(u) \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}) exp(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}))}{\sum_{l=1}^{s} M_l(u) exp(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}))} \Big) dN_i(u) \\
&= \sum_{i=1}^{s} \int_0^{\tau} \Big(\dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)}) - \sum_{l=1}^{s} w_l \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}) \Big) dN_i(u) \\
&= \sum_{i=1}^{s} \int_0^{\tau} \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)}) dN_i(u) - \sum_{l=1}^{s} \int_0^{\tau} w_l \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}) dN^{(s)}(u).
\end{aligned} \tag{A.2}$$

where $w_l = \frac{M_l(u) exp(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}))}{\sum_{l=1}^{s} M_l(u) exp(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}))}$ is a weight proportional to the hazard of failure of patient l; $dN^{(s)}(u) = \sum_{i=1}^{s} dN_i(u)$.

Now we show that the parameter $\boldsymbol{\beta^{(s)}}$ is Fisher consistent, i.e., $E[U^s(\boldsymbol{\beta^*})] = 0$.

$$
\begin{aligned}
E_{\boldsymbol{\beta}^*}[U^s(\boldsymbol{\beta^*})] &= E_{\boldsymbol{\beta}^*}\Big[\sum_{i=1}^{s}\int_0^{\tau} \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)})dN_i(u) - \sum_{l=1}^{s}\int_0^{\tau} w_l \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)})dN^{(s)}(u)\Big] \\
&= \sum_{i=1}^{s}\int_0^{\tau} E_{\boldsymbol{\beta}^*}\Big[\dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)})dN_i(u)\Big] - \sum_{l=1}^{s}\int_0^{\tau} E_{\boldsymbol{\beta}^*}\Big[w_l \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)})dN^{(s)}(u)\Big] \\
&\overset{(a)}{=} \sum_{i=1}^{s}\int_0^{\tau} E_{F(u)}\Big[E_{\boldsymbol{\beta}^*|F(u)}[\dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)})dN_i(u)]\Big] \\
&\quad - \sum_{l=1}^{s}\int_0^{\tau} E_{F(u)}\Big[E_{\boldsymbol{\beta}^*|F(u)}[w_l \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)})]dN^{(s)}(u)\Big] \\
&\overset{(b)}{=} \sum_{i=1}^{s}\int_0^{\tau} E_{F(u)}\Big[E_{\boldsymbol{\beta}^*|F(u)}[\dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)})dN_i(u)]\Big] \\
&\quad - \sum_{l=1}^{s}\int_0^{\tau} E_{F(u)}\Big[w_l \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)})dN^{(s)}(u)\Big] \\
&\overset{(c)}{=} \sum_{i=1}^{s}\int_0^{\tau} E_{F(u)}\Big[w_i \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)})dN^{(s)}(u)\Big] \\
&\quad - \sum_{l=1}^{s}\int_0^{\tau} E_{F(u)}\Big[w_l \dot{f}_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)})dN^{(s)}(u)\Big] \\
&= 0
\end{aligned} \tag{A.3}
$$

where (a) follows from the conditional expectation given the filtration $F(u)$; (b) follows from the fact that w_l and $\dot{f}_{\boldsymbol{\beta}^*}(\boldsymbol{x}^{(l)})$ are known given $F(u)$; (c) follows from the fact that $E_{\boldsymbol{\beta}^*|F(u)}[dN_i(u))] = w_i dN(u)$. Since $E_{\boldsymbol{\beta}^*}[U^s(\boldsymbol{\beta^*})] = 0$, the parameter $\boldsymbol{\beta}^{(s)}$ is Fisher consistent. In the following, we present a sufficient condition under which such a parameter is a global minmizer of $E[-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})]$.

Corollary 1: $\boldsymbol{\beta^{(s)}}$ is a global minimizer of $E[-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})]$ if $f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)})$ is an affine function of $\boldsymbol{\beta}$.

Proof. Suppose that we choose $f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)})$ a convex function of $\boldsymbol{\beta}$. Then, the first term inside summation of $-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})$ (i.e., $-f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)})$) is a concave function. We show that the second term inside summation of $-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})$, i.e., $log\Big(\sum_{l=1}^{s} Y_l(u) exp(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}))\Big)$ is a convex

function through the following steps:

Step 1: $exp(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}))$ is a convex function because $exp(.)$ is a non-decreasing convex function and $f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)})$ is convex [126, sec. 3.2.4].

Step 2: $\sum_{l=1}^{s} Y_l(u) exp(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}))$ is a convex function because a non-negative weighted sum of convex functions is convex [126, sec. 3.2.1].

Step 3: $log\Big(\sum_{l=1}^{s} Y_l(u) exp(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}))\Big)$ is a convex function because $log(.)$ is non-decreasing concave function and $\sum_{l=1}^{s} Y_l(u) exp(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)})$ is a convex function [126, sec. 3.2.4].

Therefore, expression $-f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)})Y_i(u) + log\Big(\sum_{l=1}^{s} Y_l(u) exp(f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(l)}))\Big)$ is the sum of a concave function and a convex function. A sufficient condition for convexity of this expression is convexity of $-f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)})$ or equivalently concavity of $f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)})$. This means that $f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)})$ needs to be both convex and concave at the same time. The only functions that satisfy this condition are affine functions [126, sec. 3.1.1]. By choosing $f_{\boldsymbol{\beta}}(\boldsymbol{x}^{(i)})$ as an affine function, $-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})$ and hence $E[-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})]$ become convex functions. Therefore, the parameter $\boldsymbol{\beta^{(s)}}$ becomes a global minimizer of $E[-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})]$. □

Having the loss function $E[-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})]$ a convex function motivates us to explore the convergence rate of SGD-based minimization algorithms in the next section.

A.4 Convergence rate of SGD-based estimate

In this appendix, for the sake of completeness, we prove that our SGD-based estimate can achieve the convergence rate of $O(n^{-1})$ for $f_{\boldsymbol{\beta}}(\boldsymbol{x}) = \boldsymbol{\beta}^T\boldsymbol{x}$, $s = 2$, and no ties. We also assume that our covariates are bounded, i.e., there exists some $C < \infty$ such that $||\boldsymbol{x}|| < C$ with probability 1 and that we consider a domain of optimization $\boldsymbol{B}$ such that $\max_{\boldsymbol{\beta}\in\boldsymbol{B}}\Big\{||\boldsymbol{\beta}^T\boldsymbol{x}||\Big\} < \infty$. We believe these results would hold for $s > 2$, however the calculation becomes messier

For the sake of simplicity of proof, we rewrite our objective function as

$$\begin{aligned}\boldsymbol{\beta}^{(s)} &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\Big\{\mathbb{E}_s[-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})]\Big\} \\ &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\Big\{\mathbb{E}_s[L^{(s)}(\boldsymbol{\beta})]\Big\} \\ &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\Big\{L(\boldsymbol{\beta})\Big\}, \end{aligned} \tag{A.4}$$

and consider estimating $\boldsymbol{\beta}^{(s)}$ iteratively using SGD from strata of size s through

$$\widehat{\boldsymbol{\beta}}(m) = \widehat{\boldsymbol{\beta}}(m-1) - \gamma_m \times \nabla_{\boldsymbol{\beta}} L^{(s)}(\widehat{\boldsymbol{\beta}}(m-1)|\mathcal{D}_m^{(s)}). \tag{A.5}$$

Authors in [49] showed that if the loss function $L^{(s)}(\boldsymbol{\beta})$ satisfies the 4 following assumptions, then, we can achieve the optimal convergence rate of $O(m^{-1})$ by choosing $\gamma_m = Cm^{-1}$ for the single SGD-based iterates $\widehat{\boldsymbol{\beta}}(m)$ and $\gamma_m = Cm^{-r}$, $r \in [0.5, 1)$ for averaging over iterates $\widetilde{\boldsymbol{\beta}}(m)$ (i.e., Polyak-Ruppert average).

(A1) Gradient of $L^{(s)}(\boldsymbol{\beta})$ is an unbiased estimate of the gradient $L(\boldsymbol{\beta})$,

(A2) Gradient of $L^{(s)}(\boldsymbol{\beta})$ is D-Lipschitz-continuous, i.e., $\forall\ \boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathbb{R}^p$, there exists $D \geq 0$ such that,

$$||\nabla_{\boldsymbol{\beta}} L^{(s)}(\boldsymbol{\beta}_1) - \nabla_{\boldsymbol{\beta}} L^{(s)}(\boldsymbol{\beta}_2)|| \leq D||\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2||, \tag{A.6}$$

(A3) $L(\boldsymbol{\beta}) = \mathbb{E}_s[L^{(s)}(\boldsymbol{\beta})]$ is μ-strongly convex, i.e., $\forall\ \boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathbb{R}^p$, there exists $\mu > 0$ such that,

$$L(\boldsymbol{\beta}_1) \geq L(\boldsymbol{\beta}_2) + \nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}_2)^T(\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2) + \frac{\mu}{2}||\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2||^2, \tag{A.7}$$

(A4) Variance of the gradient of $L^{(s)}(\boldsymbol{\beta})$ is bounded, i.e., there exists $\sigma^2 \in \mathbb{R}_+$ such that

$$\mathbb{E}(||\nabla_{\boldsymbol{\beta}} L^{(s)}(\boldsymbol{\beta}^*)||^2) \leq \sigma^2, w.p.1, \tag{A.8}$$

In the following, we show that the loss function in our framework, $L^{(s)}(\boldsymbol{\beta})$ satisfies all four assumptions above.

Proof of A1: This assumption is automatically satisfied based on the definition of $L(\boldsymbol{\beta}) = \mathbb{E}_s[L^{(s)}(\boldsymbol{\beta})]$ in (A.4) and that $\nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) = \nabla_{\boldsymbol{\beta}} \mathbb{E}_s[L^{(s)}(\boldsymbol{\beta})] = \mathbb{E}_s[\nabla_{\boldsymbol{\beta}} L^{(s)}(\boldsymbol{\beta})]$.

Proof of A2: The loss function $L^{(s)}(\boldsymbol{\beta})$ belongs to C^∞ continuous function family and proving (A.6) is equivalent to proving

$$\exists\, D \geq 0,\ s.t.,\ \forall \nu \in S_\nu = \{\nu : ||\nu||_2 = 1\}, \quad \nu^T \nabla^2_{\boldsymbol{\beta}} L^{(s)}(\boldsymbol{\beta}) \nu \leq D. \tag{A.9}$$

For $f_{\boldsymbol{\beta}}(\boldsymbol{x}) = \boldsymbol{\beta}^T \boldsymbol{x}$, $s = 2$ and assuming no ties, $\nabla^2_{\boldsymbol{\beta}} L^{(s)}(\boldsymbol{\beta})$ can be simplified as

$$\begin{aligned}
\nabla^2_\beta \Big\{ L^{(s)}(\boldsymbol{\beta}) \Big\} &= w(1-w)\mathbf{X} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \mathbf{X}^T \\
&\quad \times \Big(1(\delta_1 = 1) 1(y_1 < y_2) + 1(\delta_2 = 1) 1(y_2 < y_1) \Big) \\
&= w(1-w)(\mathbf{x}^{(1)} - \mathbf{x}^{(2)})(\mathbf{x}^{(1)} - \mathbf{x}^{(2)})^T \\
&\quad \times \Big(1(\delta_1 = 1) 1(y_1 < y_2) + 1(\delta_2 = 1) 1(y_2 < y_1) \Big)
\end{aligned} \tag{A.10}$$

where $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}]$; $w = w_1(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \boldsymbol{\beta}) = 1 - w_2(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \boldsymbol{\beta}) = \frac{\exp(\boldsymbol{\beta}^T \boldsymbol{x}^{(1)})}{\exp(\boldsymbol{\beta}^T \boldsymbol{x}^{(1)}) + \exp(\boldsymbol{\beta}^T \boldsymbol{x}^{(2)})}$. Note that $1(\delta_1 = 1)1(y_1 < y_2) + 1(\delta_2 = 1)1(y_2 < y_1) \leq 1$ and that $w(1-w) \leq 0.25$ because $0 \leq w \leq 1$. Therefore we have

$$\begin{aligned}
\nu^T \nabla^2_\beta \Big\{ L^{(s)}(\boldsymbol{\beta}) \Big\} \nu &\leq 0.25 \times \nu^T (\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)})(\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)})^T \nu \\
&= 0.25 \times ||\nu^T (\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)})||_2^2 \\
&\leq 0.25 \times ||(\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)})||_2^2 \times ||\nu||_2^2 \\
&\overset{(a)}{\leq} 0.25 \times (||\boldsymbol{x}^{(1)}||_2^2 + ||\boldsymbol{x}^{(2)}||_2^2) \\
&\leq 0.25 \times 2 \times max(||\boldsymbol{x}^{(1)}||_2^2, ||\boldsymbol{x}^{(2)}||_2^2) \\
&\leq 0.5 \times \max_i ||\boldsymbol{x}^{(i)}||_2^2,
\end{aligned} \tag{A.11}$$

where (a) follows the triangle inequality. Therefore, by assuming our covariates $\boldsymbol{x}$ are bounded, the gradient of $L^{(s)}(\boldsymbol{\beta})$ is $D-$Lipschitz-continuous with $D = 0.5 \times \max_i ||\boldsymbol{x}^{(i)}||^2$. This completes the proof of **A2**. $\square$

Proof of A3: The loss function $L(\boldsymbol{\beta})$ belongs to the C^∞ continuous function family and proving (A.7) is equivalent to proving

$$\exists\, \mu > 0,\ s.t. \forall \nu \in S_\nu = \{\nu : ||\nu||_2 = 1\}, \quad \nu^T I(\boldsymbol{\beta})\nu \geq \mu. \tag{A.12}$$

where $I(\boldsymbol{\beta}) = \mathbb{E}_s[\nabla^2_{\boldsymbol{\beta}} L^{(s)}(\boldsymbol{\beta})]$ is the expected Hessian matrix. Starting from (A.10), $I(\boldsymbol{\beta})$ can be written as

$$\begin{aligned}
I(\boldsymbol{\beta}) &= E_s\Big[\nabla^2_\beta\Big\{L^{(s)}(\boldsymbol{\beta})\Big\}\Big] \\
&= E_{X,Y,\Delta}\Big[\nabla^2_\beta\Big\{L^{(s)}(\boldsymbol{\beta})\Big\}\Big] \\
&\overset{(a)}{=} E_X\left[E_{Y|X}\left[E_{\Delta|X,Y}[\nabla^2_\beta\Big\{L^{(s)}(\boldsymbol{\beta})\Big\}]\right]\right] \\
&\overset{(b)}{=} (1-p_c)E_X\Big[w(1-w)(\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)})^T(\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)})\Big] \\
&= (1-p_c)\int_{X^{(1)},X^{(2)}} \Big[w(1-w)(\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)})^T(\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)})\Big] P_X(x) \\
&\overset{(c)}{=} (1-p_c)\int_{\boldsymbol{Z}} w(1-w)\boldsymbol{Z}^T\boldsymbol{Z}P_{\boldsymbol{Z}}(\boldsymbol{z})
\end{aligned} \tag{A.13}$$

where (a) follows the expansion of the intersection using conditional probabilities; (b) follow from the fact that $p_c = E_\Delta[1(\delta_i = 0))] = P_\Delta(\delta_i = 0)$ is the probability of censoring and that we have $E_Y[1(y_1 < y_2)) + 1(y_2 < y_1)] = Pr(y_1 < y_2) + Pr(y_2 < y_1) = 1$; (c) follows from the change of variable $\boldsymbol{Z} = \boldsymbol{X}^{(1)} - \boldsymbol{X}^{(2)}$. Note that since random variables $\boldsymbol{X}^{(1)}$ and $\boldsymbol{X}^{(2)}$ have density with respect to the Lebesgue measure, random variable $\boldsymbol{Z}$ also has a density with respect to the Lebesgue measure. Then we can write $\nu^T I(\boldsymbol{\beta})\nu$ as

$$\begin{aligned}
\nu^T I(\boldsymbol{\beta})\nu &= (1-p_c)\int_{\boldsymbol{Z}} w(1-w)\nu^T\boldsymbol{Z}^T\boldsymbol{Z}\nu P_{\boldsymbol{Z}}(\boldsymbol{z}) \\
&= (1-p_c)\int_{\boldsymbol{Z}} w(1-w)||\nu^T\boldsymbol{z}||_2^2 P_{\boldsymbol{Z}}(\boldsymbol{z})
\end{aligned} \tag{A.14}$$

Now we prove strong convexity of $L(\boldsymbol{\beta})$ by contradiction. The negation of statement (A.12) is:

$$\forall \mu > 0,\, \exists \nu_\mu \in S_\nu = \{\nu : ||\nu||_2 = 1\},\, s.t. \quad \nu_\mu^T I(\boldsymbol{\beta}) \nu_\mu < \mu. \tag{A.15}$$

Claim 1: Suppose the statement in A.15 holds (or equivalently (A.12) does not hold), then there exists a $\nu^* \in S_\nu = \{\nu : ||\nu||_2 = 1\}$ such that we have $P_{\boldsymbol{z}}(||\nu^{*T}\boldsymbol{z}||_2 = 0) = 1$.

Proof: Since we assumed $\boldsymbol{\beta}^T\boldsymbol{x}$ is bounded, there exists a constant $0 < C_w < 1$ such that $C_w < w < 1 - C_w$, and hence $w(1-w) > C_w^2$. Therefore, using (A.14), the statement (A.15) implies that for any $\mu > 0$ there exists $\nu_\mu \in S_\nu$ such that

$$\begin{aligned}
\mu > \nu_\mu^T I(\boldsymbol{\beta}) \nu_\mu &= (1-p_c) \int_{\boldsymbol{Z}} w(1-w) ||\nu_\mu^T \boldsymbol{z}||_2^2 P_{\boldsymbol{Z}}(\boldsymbol{z}) \\
&\overset{(a)}{>} (1-p_c) C_w^2 \int_{\boldsymbol{Z}} ||\nu_\mu^T \boldsymbol{z}||_2^2 P_{\boldsymbol{Z}}(\boldsymbol{z}) \\
&= (1-p_c) C_w^2 \int_{\boldsymbol{Z}} \Big(1(||\nu_\mu^T \boldsymbol{z}||_2^2 < \epsilon)) + 1(||\nu_\mu^T \boldsymbol{z}||_2^2 > \epsilon) \Big) ||\nu_\mu^T \boldsymbol{z}||_2^2 P_{\boldsymbol{Z}}(\boldsymbol{z}) \\
&\geq (1-p_c) C_w^2 \int_{\boldsymbol{Z}} 1(||\nu_\mu^T \boldsymbol{z}||_2^2 > \epsilon) ||\nu_\mu^T \boldsymbol{z}||_2^2 P_{\boldsymbol{Z}}(\boldsymbol{z}) \\
&> (1-p_c) C_w^2 \epsilon P_{\boldsymbol{Z}}(||\nu_\mu^T \boldsymbol{z}||_2^2 > \epsilon).
\end{aligned} \tag{A.16}$$

Note that ϵ is an arbitrary positive value. Therefore, for such a ν_μ, expression (A.16) is equivalent to

$$P_{\boldsymbol{Z}}(||\nu_\mu^T \boldsymbol{z}||_2^2 > \epsilon) < \frac{\mu}{(1-p_c) C_w^2 \epsilon} \quad \forall \mu,\, \epsilon > 0 \tag{A.17}$$

or equivalently,

$$P_{\boldsymbol{Z}}(||\nu_\mu^T \boldsymbol{z}||_2^2 < \epsilon) > 1 - \frac{\mu}{(1-p_c) C_w^2 \epsilon} \quad \forall \mu,\, \epsilon > 0. \tag{A.18}$$

Thus, there exists an infinite sequence $\nu_1, \ldots, \nu_k, \ldots$ such that for the choices of $\mu_k(\delta) = \delta(1-p_c)C_w^2\epsilon$ and $\epsilon_k = \frac{1}{k}$ we have

$$\forall\, \delta > 0\,, \exists\, K > 0\,, s.t.\,, \forall\, k > K : Pr(||\nu_k^T \boldsymbol{z}||_2 \leq \frac{1}{k}) > 1 - \delta. \tag{A.19}$$

Since S_ν is a compact space, this sequence has an infinite subsequence converging to a point $\nu^* \in S_\nu$. For such a converging point ν^*, we can write, for all k,

$$\begin{aligned} ||\nu^{*T}\boldsymbol{z}||_2 &= ||(\nu^* - \nu_k + \nu_k)^T\boldsymbol{z}||_2 \\ &\overset{(a)}{\leq} ||(\nu^* - \nu_k)^T\boldsymbol{z}||_2 + ||\nu_k^T\boldsymbol{z}||_2 \\ &\leq ||\nu^* - \nu_k||_2||\boldsymbol{z}||_2 + ||\nu_k^T\boldsymbol{z}||_2 \\ &\overset{(b)}{\leq} ||\nu^* - \nu_k||_2 C_z + ||\nu_k^T\boldsymbol{z}||_2, \end{aligned} \tag{A.20}$$

where (a) follows the triangle inequality and (b) follows the boundedness of variable $\boldsymbol{Z}$ (i.e., $||\boldsymbol{z}||_2 \leq C_z$) due to boundedness of variable $\boldsymbol{X}$. From (A.19), we may write

$$\forall\, \delta > 0\,, \epsilon > 0\,, \exists\, K_1 > 0\,, s.t.\,, \forall\, k > K_1 : Pr(||\nu_k^T\boldsymbol{z}||_2 \leq \frac{\epsilon}{2}) > 1 - \delta, \tag{A.21}$$

and since ν_k converges to ν^*, we may write

$$\forall\, \delta > 0\,, \epsilon > 0\,, \exists\, K_2 > 0\,, s.t.\,, \forall\, k > K_2 : Pr(||\nu^* - \nu_k||_2 \leq \frac{\epsilon}{2C_z}) > 1 - \delta. \tag{A.22}$$

Then $\forall\, \delta > 0$ and $\forall\, \epsilon > 0$, for $K_{max} = max(K_1, K_2)$ we have that for all $k > K_{max}$

$$\begin{aligned} Pr(||\nu^{*T}\boldsymbol{z}||_2 &\leq ||\nu^* - \nu_k||_2 C_z + ||\nu_k^T\boldsymbol{z}||_2 \\ &\leq \frac{\epsilon}{2C_z} \times C_z + \frac{\epsilon}{2} = \epsilon) > 1 - \delta, \end{aligned} \tag{A.23}$$

taking $\epsilon, \delta \to 0$, we have that $P_{\boldsymbol{Z}}(||\nu^{*T}\boldsymbol{z}||_2 = 0) = 1$. This completes the proof of *Claim 1.* □

Claim 2: If random variable $\boldsymbol{X}^{(1)}$ and $\boldsymbol{X}^{(2)}$ have density with respect to the Lebesgue measure, then we have $P_{\boldsymbol{Z}}(||\nu^T\boldsymbol{z}||_2 = 0) < 1$ for any $\nu \in S_\nu = \{\nu : ||\nu||_2 = 1\}$.

Proof: Random variables $\boldsymbol{X}^{(1)}$ and $\boldsymbol{X}^{(2)}$ have density with respect to the Lebesgue measure. Therefore, variable $\boldsymbol{Z}$ also has a density with respect to the Lebesgue measure. Therefore, for any $\nu \in S_\nu = \{\nu : ||\nu||_2 = 1\}$, $\boldsymbol{Z}$ cannot only be on a plane orthogonal to ν. In

other words,

$$\forall \nu \in S_\nu = \{\nu : ||\nu||_2 = 1\}, \exists \epsilon_\nu > 0 \,\text{and}\, \exists \delta_\nu > 0 \, s.t. \quad P_{\boldsymbol{Z}}(||\nu^{*T}\boldsymbol{z}||_2^2 > \epsilon_\nu) > \delta_\nu. \tag{A.24}$$

Then for any $\nu \in S_\nu = \{\nu : ||\nu||_2 = 1\}$, the integral $\int_{\boldsymbol{Z}} ||\nu^T \boldsymbol{z}||_2^2 dP_{\boldsymbol{Z}}(\boldsymbol{z})$ can be written as

$$\begin{aligned}
\int_{\boldsymbol{Z}} ||\nu^T \boldsymbol{z}||_2^2 dP_{\boldsymbol{Z}}(\boldsymbol{z}) &= \int_{\boldsymbol{Z}} \Big(1(||\nu^T \boldsymbol{z}||_2^2 < \epsilon_\nu)) + 1(||\nu^T \boldsymbol{z}||_2^2 > \epsilon_\nu)\Big) ||\nu^T \boldsymbol{z}||_2^2 dP_{\boldsymbol{Z}}(\boldsymbol{z}) \\
&\geq \int_{\boldsymbol{Z}} 1(||\nu^T \boldsymbol{z}||_2^2 > \epsilon_\nu) ||\nu^T \boldsymbol{z}||_2^2 dP_{\boldsymbol{Z}}(\boldsymbol{z}) \\
&> \epsilon_\nu \int_{\boldsymbol{Z}} 1(||\nu^T \boldsymbol{z}||_2^2 > \epsilon_\nu) dP_{\boldsymbol{Z}}(\boldsymbol{z}) \\
&= \epsilon_\nu P_{\boldsymbol{Z}}(||\nu^{*T} \boldsymbol{z}||_2^2 > \epsilon_\nu) \\
&> \epsilon_\nu \delta_\nu > 0,
\end{aligned} \tag{A.25}$$

indicating that $P_{\boldsymbol{Z}}(||\nu^T \boldsymbol{z}||_2 = 0) < 1$ (or $P_{\boldsymbol{Z}}(||\nu^T \boldsymbol{z}||_2 = 0) \neq 1$). This completes the proof of *Claim 2*. □

Claim 1 indicated that if the strong convexity statement (A.12) does not hold, we must have $P_{\boldsymbol{z}}(||\nu^{*T}\boldsymbol{z}||_2 = 0) = 1$. This is in contradiction with the result of *Claim 2* indicating that for random variable $\boldsymbol{Z}$ with Lebesgue density we have $P_{\boldsymbol{z}}(||\nu^{*T}\boldsymbol{z}||_2 = 0) \neq 1$. Therefore, (A.15) is not true and the statement (A.12) holds, i.e., $L(\boldsymbol{\beta})$ is strongly convex. This completes the proof of **A3**. □

Proof of A4: The gradient of $L^{(s)}(\boldsymbol{\beta})$ may be written as

$$\nabla_{\boldsymbol{\beta}} L^{(s)}(\boldsymbol{\beta}) = \sum_{i=1}^{s} 1(\delta_i = 1)(\boldsymbol{x}^{(i)} - \sum_{j \in R_i} w_j \boldsymbol{x}^{(j)}). \tag{A.26}$$

Then we can write

$$
\begin{aligned}
||\nabla_{\boldsymbol{\beta}} L^{(s)}(\boldsymbol{\beta})||^2 &= ||\sum_{i=1}^{s} 1(\delta_i = 1)(\boldsymbol{x}^{(i)} - \sum_{j \in R_i} w_j \boldsymbol{x}^{(j)})||^2 \\
&\overset{(a)}{\leq} \Big(\sum_{i=1}^{s} (||1(\delta_i = 1)|| \times ||\boldsymbol{x}^{(i)} - \sum_{j \in R_i} w_j \boldsymbol{x}^{(j)})|| \Big)^2 \\
&\overset{(b)}{\leq} \Big(\sum_{i=1}^{s} (||\boldsymbol{x}^{(i)}|| + ||\sum_{j \in R_i} w_j \boldsymbol{x}^{(j)}||) \Big)^2 \\
&\overset{(c)}{\leq} \Big(\sum_{i=1}^{s} (||\boldsymbol{x}^{(i)}|| + \max_{j \in R_i} ||\boldsymbol{x}^{(j)}||) \Big)^2 \\
&\leq 4S^2 \max_i ||\boldsymbol{x}^{(i)}||^2
\end{aligned} \tag{A.27}
$$

where (a) follows from the triangle inequality; (b) and (c) follow from the triangle inequality and that $w_j \leq 1$. Therefore, given boundedness of covariates $\boldsymbol{x}$, we can choose $\sigma = 2S \max_i ||\boldsymbol{x}^{(i)}||$. This completes the proof of **A4**. □

We showed that all four assumptions are satisfied and thus the results in [49] give us our claimed convergence rates for both single SGD-based estimates and averaging over estimates (Polyak-Ruppert average).

A.5 Mini-batches, moment-based-learning-rate, and multiple epochs

For small strata size s, the stochastic gradients given in (8) are potentially quite noisy. In these cases, rather than using only a single stratum to calculate the stochastic gradient, it may be preferable to average gradients over multiple strata. This is known as using a mini-batch [7] in the SGD literature. In this survival context, we use batches of strata, so a larger strata size s can eliminate the need for mini-batch sizes greater than 1.

Choosing a reasonable value for the hyper-parameter γ_m (the learning rate) is critical for good practical performance of the algorithm (see Section A.11.1 for related simulation results). A number of publications have developed methods for adaptively selecting the learning rate using moments, including `Adagrad` [127], and `ADAM` [128]. However, it was shown

that `ADAM` may not converge in some settings and an updated version called `AMSGrad` has been proposed [50]. We found substantially improved performance on simulated data with `AMSGrad` over the simple non-adaptive updating rule given in (9), especially in combination with averaging over iterates as discussed in Section 2.3.3.

In practice, taking only a single pass over the data leads to poor empirical performance. We generally use multiple passes (or "epochs"). In particular, for each epoch, we (1) randomly partition our data into strata; then (2) use the last updated iterate of $\widehat{\boldsymbol{\beta}}$ from the previous epoch as the initial iterate of $\widehat{\boldsymbol{\beta}}$ for the new epoch; and finally (3) apply a full pass of stochastic gradient descent (with e.g., averaging and momentum) over the partitioned data. In practice, we found that ~ 100 epochs was more than enough for very robust convergence (see the top right panel of Figure 2.1 in Section 2.6.1).

A.6 Implementation of streaming and non-streaming algorithms

In this section, we present the implementations of both streaming and non-streaming mini-batch stochastic gradient descent algorithms using our proposed framework BigSurvSGD. Without loss of generality, we only present algorithms without the moment-based step-size adaptation.

A.6.1 Implementation of streaming algorithm

As we discussed in Chapter 2, our proposed framework facilitates the implementation of an algorithm using SGD that handles the streaming data. This implementation is very straightforward: We update the estimate (i.e., $\widehat{\boldsymbol{\beta}}$) in a streaming fashion using each new coming mini-batch of data. Therefore, our algorithm does not run into memory issues because we do not need to collect all the data before the estimation (as would be required by `coxph()`). Algorithm 2 summarizes the implementation of a streaming algorithm using our proposed framework where we use a single stratum for each step (i.e., mini-batches of size 1). Extension to mini-batches of size greater than 1 is very straightforward (see next section).

Algorithm 2: Implementation of streaming algorithm using BigSurvSGD

Result: $\widehat{\boldsymbol{\beta}}$

Initialization:

Choose strata size s

$\widehat{\boldsymbol{\beta}}(0) = \mathbf{0}$

Choose C for $\gamma_m = \frac{C}{\sqrt{m}}$

for *($m = 1, 2, ...$)* **do**

Draw data $\mathcal{D}_m^{(s)}$ including s patients from patient $s \times (m-1) + 1$ to patient $s \times m$

Update the estimate $\widehat{\boldsymbol{\beta}}(m)$ using

$$\widehat{\boldsymbol{\beta}}(m) = \widehat{\boldsymbol{\beta}}(m-1) + \gamma_m \times \dot{pl}^{(s)}(\widehat{\boldsymbol{\beta}}(m-1)|\mathcal{D}_m^{(s)})$$

end

Output:

Calculate averaged over iterate estimates as $\widetilde{\boldsymbol{\beta}}(m) = \frac{1}{m}\sum_{l=1}^{m}\widehat{\boldsymbol{\beta}}(l)$, $l = 1, 2, \ldots, m$

A.6.2 Implementation of a non-streaming mini-batch stochastic gradient descent algorithm

In many cases, it is of interest to use each datum more than once (this can empirically improve performance). In these cases, we still use a mini-batch stochastic gradient descent algorithm. We split our data evenly into mini-batches, each with K strata of size s. Then we iteratively update the estimate using the batches of strata. Using multiple, rather than single strata per batch results in more stable (less noisy) gradients. We use learning rate $\frac{\gamma_m}{K}$ instead of γ_m since the gradient in each step of mini-batch gradient descent is the sum of gradients over K strata with size s. One strength of this implementation compared to `coxph()` is we can read the batches of strata chunk-by-chunk from the hard drive. Therefore, this implementation handles large amounts of data without running into the memory issues. Algorithm 3 summarizes the implementation of a non-streaming mini-batch stochastic gradient descent algorithm using our proposed framework.

Algorithm 3: Non-streaming mini-batch gradient descent algorithm using BigSurvSGD

Result: $\widehat{\boldsymbol{\beta}}$

Initialization:

Choose strata size s

Choose batch size K

Choose number of epochs n_E

$m = 0$

$\widehat{\boldsymbol{\beta}}(0) = \mathbf{0}$

Choose C for $\gamma_m = \frac{C}{\sqrt{m}}$

for *($n_e = 1, 2, ..., n_E$)* **do**

Divide data randomly into n_B disjoint batches, $b = 1, 2, \ldots, n_B$. Each batch includes K disjoint strata of patients, $\mathcal{D}_{b,1}^{(s)}$, ..., $\mathcal{D}_{b,K}^{(s)}$ with size s.

for *($b = 1, 2, ..., n_B$)* **do**

$m = m + 1$

Update the estimate of $\widehat{\boldsymbol{\beta}}(m)$ using

$$\widehat{\boldsymbol{\beta}}(m) = \widehat{\boldsymbol{\beta}}(m-1) + \frac{\gamma_m}{K} \times \sum_{k=1}^{K} \dot{pl}^{(s)}(\widehat{\boldsymbol{\beta}}(m-1)|\mathcal{D}_{b,k}^{(s)})$$

end

end

Output:

Calculate averaged over iterate estimates as $\widetilde{\boldsymbol{\beta}}(m) = \frac{1}{m}\sum_{l=1}^{m}\widehat{\boldsymbol{\beta}}(l)$, $l = 1, 2, \ldots, m$

A.7 Details of implementations for inference

In this section, we present details of implementations for both the plug-in and bootstrap approaches for constructing a confidence interval.

A.7.1 Plug-in based approach for inference

We note, in the optimization problem in expression (2.5), the kernel $-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})$ is differentiable for all values of $\boldsymbol{\beta}$, and that $\boldsymbol{\beta}^{(s)}$ is the unique minimizer of $E[-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})]$. Authors in [53] studied the asymptotic properties of approximate minimizers of problems (2.5) that satisfy [a generalization of] the approximate first-order term condition

$$\binom{n}{s}^{-1} \sum_{\mathcal{D}^{(s)} \subset \mathcal{D}^{(n)}} \frac{\partial}{\partial \boldsymbol{\beta}} \{-pl^{(s)}(\widehat{\boldsymbol{\beta}}|\mathcal{D}^{(s)})\} = o_p(n^{-1/2}). \tag{A.28}$$

Under weak regularity conditions, their work shows that the asymptotic behaviour of $\widehat{\boldsymbol{\beta}}$ is given by

$$\sqrt{n}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \to N(\mathbf{0}, \boldsymbol{H}^{-1}\boldsymbol{V}\boldsymbol{H}^{-1}) \tag{A.29}$$

where matrices $\boldsymbol{V}$ and $\boldsymbol{H}$ are given by

$$\begin{aligned} \boldsymbol{V} &= s^2 \, \mathrm{E}[r(\mathcal{D}, \boldsymbol{\beta}) r(\mathcal{D}, \boldsymbol{\beta})^T] \\ \boldsymbol{H} &= \frac{\partial}{\partial \boldsymbol{\beta}} \mathrm{E}[r(\mathcal{D}, \boldsymbol{\beta})] \end{aligned} \tag{A.30}$$

with $r(\mathcal{D}, \boldsymbol{\beta})$ in our problem is defined as

$$r(\mathcal{D}, \boldsymbol{\beta}) = \mathrm{E}\left[\frac{\partial}{\partial \boldsymbol{\beta}} \{-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)} = \{\mathcal{D}, \mathcal{D}_{i_2}, \ldots, \mathcal{D}_{i_s}\})\} \middle| \mathcal{D} \right], \tag{A.31}$$

the expectation of the first derivative of the kernel with respect to $\boldsymbol{\beta}$ when one observation (i.e., $\mathcal{D}$) out of s observations is kept fixed (for more details, see [53]). Finally, authors in [53] proposed consistent estimators for matrices $\boldsymbol{V}$ and $\boldsymbol{H}$ as

$$\begin{aligned} \widehat{\boldsymbol{V}}_n &= \frac{s^2}{n} \sum_{i=1}^{n} \widehat{r}(\mathcal{D}_i, \boldsymbol{\beta}) \widehat{r}(\mathcal{D}_i, \boldsymbol{\beta})^T \\ \widehat{\boldsymbol{H}}_n &= \frac{\partial}{\partial \boldsymbol{\beta}} \left(\frac{1}{n} \sum_{i=1}^{n} \widehat{r}(\mathcal{D}_i, \boldsymbol{\beta}) \right) \end{aligned} \tag{A.32}$$

with $\widehat{r}(\mathcal{D}_i, \boldsymbol{\beta})$ given by

$$\widehat{r}(\mathcal{D}_i, \boldsymbol{\beta}) = \binom{n-1}{s-1}^{-1} \sum_{\mathcal{D}_{i_2}, \ldots, \mathcal{D}_{i_s} \subset \{\mathcal{D}^{(n)} \setminus \mathcal{D}_i\}} \frac{\partial}{\partial \boldsymbol{\beta}} \{-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)} = \{\mathcal{D}_i, \mathcal{D}_{i_2}, \ldots, \mathcal{D}_{i_s}\})\}, \tag{A.33}$$

an empirical estimate of (A.31) where we fix $\mathcal{D} = \mathcal{D}_i$ and we include all possible $\binom{n-1}{s-1}$ combinations of $s-1$ observations $\{i_2, i_3, \ldots, i_s\}$ from $\{1, 2, \ldots, n\}\backslash\{i\}$.

This plug-in estimator provides an explicit form for our standard error, however, it is prohibitively computationally expensive because we need to go through $\binom{n-1}{s-1}$ distinct combinations for each of our n observations. Our solution is to only consider a random small fraction of $\binom{n-1}{s-1}$ combinations (e.g., $n_o = 1000 \ll \binom{n-1}{s-1}$ distinct combinations) for each observation to estimate the standard error. Algorithm 4 summarizes our implementation of this plugin approach to estimate a $(1-\alpha) \times 100\%$ CI. In practice, sampling $100 \sim 1000$ strata per observation suffice to get near nominal coverage (see Section A.11.2 for related results). After estimating the standard error, we construct the $(1-\alpha)100\%$ confidence interval following the asymptotic normality of estimator $\widetilde{\boldsymbol{\beta}}$.

A.7.2 Bootstrap-based approach for inference

The Bootstrap [129, 130] is a popular approach for constructing asymptotically valid confidence intervals that require no analytical work. For i.i.d. observations $\mathcal{D}^{(n)} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$, in simpler examples [130] suggest using a bootstrap sample (sampling with replacement) $\mathcal{D}^{b,(n)} = \{\mathcal{D}_1^b, \mathcal{D}_2^b, \ldots, \mathcal{D}_n^b\}$ and calculating the bootstrapped U-statistic as

$$U_n^b(\boldsymbol{\beta}) = \binom{n}{s}^{-1} \sum_{\mathcal{D}^{(s)} \subset \mathcal{D}^{b,(n)}} \{-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})\}. \tag{A.34}$$

Multiplier empirical processes have been widely studied in literature [51, 131, 132]. Authors in [133, 55] developed the theory of the multiplier (weighted) bootstrap for empirical processes. The weighted version of bootstrapped U-statistics is given by

$$\bar{U}_n^b(\boldsymbol{\beta}) = \binom{n}{s}^{-1} \sum_{1 \le i_1 < \cdots < i_s \le n} \omega_{i_1} \ldots \omega_{i_s} h_{\boldsymbol{\beta}}(D^{i_1}, \ldots, D^{i_s}), \tag{A.35}$$

where $\omega_1 \ldots \omega_n$ are random variables independent of data $\mathcal{D}^{(n)}$; $h_{\boldsymbol{\beta}}(\mathcal{D}^{(s)})$ with $\mathcal{D}^{(s)} = \{D^{i_1} \ldots D^{i_s}\}$ is the s-order kernel function which is $-pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})$ for our U-statistics defined

Algorithm 4: Construction of a $(1-\alpha)100\%$ CI using plug-in method

Result: $\widetilde{\boldsymbol{\beta}}$ and $(1-\alpha)100\%$

Initialization:

Choose strata size s

Choose C for $\gamma_m = \frac{C}{\sqrt{m}}$

Specify significance level α

Calculate $\widetilde{\boldsymbol{\beta}}$ using Algorithm 3

Specify n_o ($n_o \ll \binom{n-1}{s-1}$), number of strata samples per observation to estimate the standard error

for *($i = 1, 2, ...n$)* **do**

Randomly choose $\mathcal{M}$, a set of M distinct subsets of $s-1$ elements from $\{1, 2, \ldots, n\}\backslash\{i\}$. Then estimate $\widehat{r}(\mathcal{D}_i, \widetilde{\boldsymbol{\beta}})$ using

$$\widehat{r}(\mathcal{D}_i, \widetilde{\boldsymbol{\beta}}) = \frac{1}{M} \sum_{\{i_2, i_3, \ldots, i_s\} \in \mathcal{M}} \frac{\partial}{\partial \boldsymbol{\beta}} \{-pl^{(s)}(\boldsymbol{\beta} | \mathcal{D}^{(s)} = \{\mathcal{D}_i, \mathcal{D}_{i_2}, \ldots, \mathcal{D}_{i_s}\})\}|_{\boldsymbol{\beta}=\widetilde{\boldsymbol{\beta}}}$$

end

Estimate $\widehat{\boldsymbol{V}}_n$ and $\widehat{\boldsymbol{H}}_n$ using

$$\widehat{\boldsymbol{V}}_n = \frac{s^2}{n} \sum_{i=1}^{n} \widehat{r}(\mathcal{D}_i, \widetilde{\boldsymbol{\beta}}) \widehat{r}(\mathcal{D}_i, \widetilde{\boldsymbol{\beta}})^T$$

$$\widehat{\boldsymbol{H}}_n = \left[\frac{\partial}{\partial \boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} \widehat{r}(\mathcal{D}_i, \boldsymbol{\beta}) \right]_{\boldsymbol{\beta}=\widetilde{\boldsymbol{\beta}}}$$

Output:

Estimate standard error of $\widetilde{\boldsymbol{\beta}}$ using $\widehat{\widetilde{SE}} = \sqrt{diag(\widehat{\boldsymbol{H}}_n^{-1} \widehat{\boldsymbol{V}}_n \widehat{\boldsymbol{H}}_n^{-1})/n}$. Report the $(1-\alpha)100\%$ CI as $(\widetilde{\boldsymbol{\beta}} - z_{\frac{\alpha}{2}} \times \widehat{\widetilde{SE}}, \widetilde{\boldsymbol{\beta}} + z_{\frac{\alpha}{2}} \times \widehat{\widetilde{SE}})$ where $z_{\frac{\alpha}{2}}$ is the upper $\frac{\alpha}{2}$ critical value for the standard normal distribution.

in (A.34). While one could directly use this multiplier-based estimator, we do something very slightly different: We resample observations with replacement and then engage with the resulting U-process (this means we can, in theory, have replicated observations in a single term). Nevertheless, our bootstrapped U-process in (A.34) can be approximated using the multiplier U-process in (A.35) by drawing weights from a multi-nominal distribution, i.e., $(\omega_1, \ldots, \omega_n) \sim Multinominal(n, \frac{1}{n}, \ldots, \frac{1}{n})$.

Because of the possibility of replicates in our process, this approximation is off by a sum that is in total $O(n^{-1})$: This is because the probability that a term will contain a pair of the same observation is bounded by $s * (s-1)/n$ (and for x and β with bounded support our U-process is bounded and has a bounded derivative). As we will show, the multiplier U-process-based estimate converges at a rate of $||\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*|| = O(n^{-1/2})$. Thus, one could show that the estimate from (A.34) and its approximation by (A.35) asymptotically converge to the same limiting distribution (via a standard Taylor-series expansion argument). Below, we focus on showing the asymptotic behavior of the optimizer of the multiplier bootstrapped U-process approximated by (A.35) with weights drawn from the multinomial distribution.

Let $\widehat{\boldsymbol{\beta}}$ denote the estimate given by minimizing the original U-statistic in expression (2.11) (using the original sample) and $\widehat{\boldsymbol{\beta}}^b$ denote the estimate given by minimizing the multiplier bootstrapped U-statistic with the b-th bootstrapped sample. We define $\widetilde{\boldsymbol{\beta}} = \frac{1}{n}\sum_{i=1}^n \widehat{\boldsymbol{\beta}}(i)$ and $\widetilde{\boldsymbol{\beta}}^b = \frac{1}{n}\sum_{i=1}^n \widehat{\boldsymbol{\beta}}^b(i)$ as the Ruppert-Polyak averaged estimates of our original sample and the b-th bootstrap resample, respectively. Authors in [55] showed that if *Assumption A* and *Condition M* (see below) hold, $\sqrt{n}(\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta}^*)$ and $\sqrt{n}(\widetilde{\boldsymbol{\beta}}^b - \widetilde{\boldsymbol{\beta}})|\mathcal{D}^{(n)}$ converge to the same limiting distribution. In the following, we briefly explain notation when presenting *Assumption A* and *Condition M*. For more extensive explanations, we refer readers to [55, 134].

***Assumption A*:** The following assumptions hold for weights $(\omega_1 \ldots \omega_n)$:

W1: Weights $(\omega_1 \ldots \omega_n)$ are exchangeable (i.e., $(\omega_1 \ldots \omega_n) =_d (\omega_{\pi(1)} \ldots \omega_{\pi(n)})$ for any

permutation π of $\{1,\ldots,n\}$), non-negative, and $\sum_{i=1}^{n}\omega_i = n$,

W2: $sup_n \int_0^\infty \mathbb{P}(|\omega| > t)^{\frac{1}{2s}} dt < \infty$ and there exists $c > 0$ such that

$$\frac{1}{n}\sum_{i=1}^{n}(\omega_i - 1)^2 \to_{P_\omega} c^2 \tag{A.36}$$

***Condition M*:** Suppose the bootstrap weights $(\omega_1,\ldots,\omega_n)$ satisfy *Assumption A*, and the following conditions hold.

M1: $U_n(\boldsymbol{\beta})$ has a unique minimizer at $\boldsymbol{\beta} = \boldsymbol{\beta}^*$ and there exists some positive definite matrix V such that for any $\boldsymbol{\beta}$ close to $\boldsymbol{\beta}^*$ we have

$$U_n(\boldsymbol{\beta}) - U_n(\boldsymbol{\beta}^*) = \frac{-1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}^*)^T V(\boldsymbol{\beta} - \boldsymbol{\beta}^*) + o(||\boldsymbol{\beta} - \boldsymbol{\beta}^*||^2) \tag{A.37}$$

M2: $\boldsymbol{\mathcal{F}} = \{h_{\boldsymbol{\beta}} : \boldsymbol{\beta} \in B \subset \mathbb{R}^p\}$ admits P^s-square integrable envelope F such that

$$\int_0^1 (sup_Q log\mathcal{N}(\epsilon||F||_{L_2(Q)}, \mathcal{F}, L_2(Q)))^{s/2} d\epsilon < \infty, \tag{A.38}$$

where $\mathcal{N}(\epsilon, \mathcal{F}, L_2(Q))$ is the ϵ-covering number of $\mathcal{F}$ relative to norm $L_2(Q)$ for probability measure Q (see [135, 55] for more details).

M3: Suppose $\widetilde{\boldsymbol{\beta}}$ be an estimator of $\boldsymbol{\beta}^*$ that minimizes the original U-statistics and $\widetilde{\boldsymbol{\beta}}^b$ be an estimator of $\boldsymbol{\beta}^*$ that minimizes the bootstrapped U-statistics. There exists a measurable map $\Delta : \mathcal{X} \to \mathbb{R}^p$ such that $P\Delta(X) = 0$ and $P||\Delta||^2 < \infty$, and for $r_n(.,\boldsymbol{\beta})$ defined as

$$r_n(x,\theta) = \frac{\pi_1(h_{\boldsymbol{\beta}} - h_{\boldsymbol{\beta}^*})(x) - (\boldsymbol{\beta} - \boldsymbol{\beta}^*)^T\Delta(x)}{||\boldsymbol{\beta} - \boldsymbol{\beta}^*|| \vee n^{-1/2}}, \tag{A.39}$$

the following is satisfied for any $\delta_n \to 0$,

$$sup_{\boldsymbol{\beta}:||\boldsymbol{\beta}-\boldsymbol{\beta}^*||\le\delta_n}|\mathbb{G}_n r_n(.,\boldsymbol{\beta})| = o_P(1). \tag{A.40}$$

Note that $\pi_1 h(.)$ is a P-degenerate kernel of order 2 using the Hoeffding decomposition of U-statistics [51]. If $||\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta}^*|| = o_P(1)$ and $||\widetilde{\boldsymbol{\beta}}^b - \boldsymbol{\beta}^*|| = o_P(1)$ in P_D-probability, then $\sqrt{n}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*) \to_d \mathbb{N}(0, V^{-1}cov(\Delta)(V^{-1})^T$ and

$$sup_{t\in\mathbb{R}^p}|\mathbb{P}_{W|D}(\sqrt{n}(\widetilde{\boldsymbol{\beta}}^b - \widetilde{\boldsymbol{\beta}}) \le t) - \mathbb{P}(c.\sqrt{n}(\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta}^*) \le t)| \to_{P_X} 0, \tag{A.41}$$

where c is the constant in $(W2)$.

Proof of *Assumption W*:

Proof of W1 and W2: These assumptions are directly related to assumptions A1-A5 in Section 2 of [133]. Example 3.1 of that manuscript gives proof for "Efron's Bootstrap" which exactly coincides with our choice of multinomial weights.

Proof of *condition M*:

Proof of M1: In Appendix A.4, we demonstrated that our objective function is μ-strongly convex, indicating that $\boldsymbol{\beta}^*$ is the unique minimizer. Also, our kernel function $h_{\boldsymbol{\beta}}(\boldsymbol{x})$ is infinitely differentiable with respect to both of its arguments $\boldsymbol{\beta}$ and $\boldsymbol{x}$ and is indeed an analytic function. Therefore, a second-order Taylor approximation is valid at any $\boldsymbol{\beta}$.

Proof of M2: Any finite-dimensional vector space $\mathcal{F}$ of measurable functions is a Vapnik-Chervonenkis (VC) subgraph of dimension smaller than or equal to $dim(\mathcal{F}) + 1$ [51]. Our kernel function $h_{\boldsymbol{\beta}}(.)$ is defined based on a stratum of s patients each contributing by their risk function. Therefore, our kernel function is a VC class of dimension at most $s + 1$. Therefore, the integral in (A.38) is finite [51].

Proof of M3: Our kernel function is infinitely differentiable with respect to its arguments and its gradient is D-Lipschitz-continuous (see Appendix A.4). In Appendix A.4, we showed that by choosing the learning rate as $\gamma_m = \frac{C}{\sqrt{m}}$, we achieve $||\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta}^*|| = O_P(m^{-1/2})$ after m iterations of stochastic gradient descent. If the number of stochastic iterations m is asymptotically at least slightly larger than sample size n, we can achieve $\sqrt{n}||\tilde{\beta} - \beta^*|| = O(1)$. Also, our bootstrapped estimator based on stochastic optimization is asymptotically close to the U-process-minimizer-bootstrapped-estimator in (A.35). Therefore, one can use proof of Claim 1 (as part of proof for Theorem 4.2) in [55] to show that $\sqrt{n}||\widetilde{\boldsymbol{\beta}}^b - \boldsymbol{\beta}^*|| = O_P(1)$. By choosing $\Delta(x)$ as the derivative of $\pi_1 h_{\boldsymbol{\beta}}(\boldsymbol{x})$ at $\boldsymbol{\beta} = \boldsymbol{\beta}_0$, it is straightforward to prove that this condition holds.

Now that we have verified *Assumption W* and *Condition M*, we have that $\sqrt{n}(\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta}^*)$ and $\sqrt{n}(\widetilde{\boldsymbol{\beta}}^b - \widetilde{\boldsymbol{\beta}})|\mathcal{D}^{(n)}$ converge to the same limiting distribution. We can construct a confidence interval for $\widetilde{\boldsymbol{\beta}}$ using quantiles of $\sqrt{n}(\widetilde{\boldsymbol{\beta}}^b - \widetilde{\boldsymbol{\beta}})|\mathcal{D}^{(n)}$. In our implementation, we use the estimate

$\widetilde{\boldsymbol{\beta}}$ from the original sample as the initial iterate of the SGD algorithm to calculate the coefficients $\widehat{\boldsymbol{\beta}}^b$ for each bootstrap resample.

Algorithm 5 presents details for constructing a $(1-\alpha)100\%$ confidence interval using the bootstrap approach. The first step of this algorithm is to get an initial estimate $\widetilde{\boldsymbol{\beta}}$ using our proposed Algorithm 3 in Section A.7.2. The bootstrap steps for getting the bootstrapped estimates $\widetilde{\boldsymbol{\beta}}^b$, $b = 1, 2, \ldots, B$ are exactly the same as what we presented in Algorithm 3 except we use the bootstrap resamples with initialization $\widehat{\boldsymbol{\beta}}^b(0) = \widetilde{\boldsymbol{\beta}}, b = 1, 2, \ldots, B$. Then, following the fact that $\sqrt{n}(\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta})$ and $\sqrt{n}(\widetilde{\boldsymbol{\beta}}^b - \widetilde{\boldsymbol{\beta}})$ converge in distribution to the same limiting distribution (which is gaussian), we construct the $(1-\alpha)100\%$ for $\widetilde{\boldsymbol{\beta}}$. To save computing time one can just use the bootstrap to get at the standard error of the coefficient estimates. In this procedure, we use knowledge that distribution of the coefficient estimator asymptotically converges to a normal distribution to construct a symmetric confidence interval (and try to further improve this using a t-distribution). Suppose that $\widetilde{\beta}_j$ is j-th ($j = 1, 2, \ldots, P$) element of our coefficient estimate calculated by Algorithm 3 and $\widetilde{\beta}_j^b$ is corresponding j-th element of estimated parameters from the b-th ($b = 1, 2, \ldots, B$) bootstrap resample. Then, one can construct an asymptotic $(1-\alpha)100\%$ confidence interval for $\widetilde{\beta}_j^b$ as

$$(1-\alpha)100\%\, CI : \left(\widetilde{\beta}_j - qt(1-\alpha/2, df = B) * \widetilde{\sigma}_j^B,\ \widetilde{\beta}_j + qt(1-\alpha/2, df = B) * \widetilde{\sigma}_j^B\right), \quad \text{(A.42)}$$

where $qt(1-\alpha/2, df = B)$ is the $(1-\alpha/2)$th quantile of the t-distribution with degrees of freedom $df = B$; and $\widetilde{\sigma}_j^B$ is the sample standard deviation of estimates obtained from those bootstrap resamples. Details of this procedure are exactly the same as Algorithm 5 except the last two lines should be modified based on (A.42).

A single-pass bootstrap approach (online bootstrap)

In the scenario wherein we have an enormous amount of data, one may want to consider using only one pass over the data to calculate our parameter estimates and confidence intervals. One way to do this is to use the multiplier bootstrap idea proposed by [136]. For one pass of data, we multiply the gradient of the partial likelihood for each stratum (m) by a weight W_n

drawn independently from a *Poisson(1)* distribution with $\mathbb{E}(W_m) = var(W_m) = 1$. Authors in [136] showed that $\sqrt{n}(\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta}^*)$ and [the random measure] $\sqrt{n}(\widetilde{\boldsymbol{\beta}}^b - \widetilde{\boldsymbol{\beta}})|\mathcal{D}^{(n)}$ converge to the same limiting distribution if we choose $\gamma_m = Cm^{-r}$, $r \in [0.5, 1)$ (for Polyak-Ruppert average) and that assumptions A1-A4 in Appendix A.4 and the following assumption (**A5**) is satisfied:

(A5) $\mathbb{E}||\nabla L^{(s)}(\boldsymbol{\beta})||^2 \leq c(1 + ||\boldsymbol{\beta}||^2)$ for some c and $\mathbb{E}||\nabla L^{(s)}(\boldsymbol{\beta} - \nabla L^{(s)}(\boldsymbol{\beta}_0||^2 \leq \zeta(||\boldsymbol{\beta} - \boldsymbol{\beta}_0||)$ for some $\zeta(.)$ with $\zeta(x) \to 0$ as $x \to 0$.

We proved that assumptions **A1-A4** hold in Appendix A.4 and we only have left to show that assumption **A5** holds too.

Proof of A5: The first expression in **A5** holds by following our proof for assumption **A4** in (A.27). The proof of the second expression is similar to the proof of assumption **A4** (the same definition for w_j, $\boldsymbol{x}^{(i)}$, δ_i, and R_i are used here). The gradient of $L^{(s)}(\boldsymbol{\beta})$ may be written as

$$\nabla_{\boldsymbol{\beta}} L^{(s)}(\boldsymbol{\beta}) = \sum_{i=1}^{s} 1(\delta_i = 1)(\boldsymbol{x}^{(i)} - \sum_{j \in R_i} w_j \boldsymbol{x}^{(j)}). \tag{A.43}$$

Then we can write

$$
\begin{aligned}
||\nabla_{\boldsymbol{\beta}} L^{(s)}(\boldsymbol{\beta}) - \nabla_{\boldsymbol{\beta}} L^{(s)}(\boldsymbol{\beta}_0)||^2 &= ||\sum_{i=1}^{s} 1(\delta_i = 1)(\boldsymbol{x}^{(i)} - \sum_{j \in R_i} w_j \boldsymbol{x}^{(j)}) \\
&\quad - \sum_{k=1}^{s} 1(\delta_k = 1)(\boldsymbol{x}^{(k)} - \sum_{l \in R_k} w_l^0 \boldsymbol{x}^{(l)})||^2 \\
&= ||\sum_{i=1}^{s} 1(\delta_i = 1)(\sum_{j \in R_i} (w_j^0 - w_j)\boldsymbol{x}^{(j)})||^2 \\
&\overset{(a)}{\leq} ||\sum_{i=1}^{s} \sum_{j \in R_i} (w_j^0 - w_j)\boldsymbol{x}^{(j)}||^2 \\
&\overset{(b)}{\leq} (\sum_{i=1}^{s} \sum_{j \in R_i} ||w_j^0 - w_j|| \times ||\boldsymbol{x}^{(j)}||)^2 \\
&\overset{(c)}{\leq} \max_i ||\boldsymbol{x}^{(i)}||^2 \times \sum_{i=1}^{s} \sum_{j \in R_i} ||w_j^0 - w_j||)^2 \\
&\overset{(d)}{\leq} \max_i ||\boldsymbol{x}^{(i)}||^2 \times \sum_{i=1}^{s} \sum_{j \in R_i} \max_j ||w_j^0 - w_j||^2 \\
&\overset{(e)}{\leq} s^2 \max_i ||\boldsymbol{x}^{(i)}||^2 \times \max_j ||w_j^0 - w_j||^2
\end{aligned} \tag{A.44}
$$

where w_j^0 follows the expression of w_j given in Appendix A.4 by replacing $\boldsymbol{\beta}$ with $\boldsymbol{\beta}_0$; (a) follows from $1(\delta_k = 1) \leq 1$ (b) follows from the triangle inequality; (c) and (d) follow from the fact that each norm is less than or equal to its maximum; (e) follows from the fact that size of set R_j is less than s. Therefore, given boundedness of covariates $\boldsymbol{x}$ and that w_j is a continuous function with respect to $\boldsymbol{\beta}$, if we choose $\boldsymbol{\beta} - \boldsymbol{\beta}_0 \to 0$ or $\boldsymbol{\beta} \to \boldsymbol{\beta}_0$, then $x = w_j^0 - w_j \to 0$, and $\zeta(x) \to 0$ as well. This completes the proof of **A5**. □

Algorithm 5: Estimate $(1-\alpha)100\%$ confidence interval using bootstrap method

Result: $(1-\alpha)100\%$ confidence interval

Initialization:

Choose strata size s

Choose C for $\gamma_m = \frac{C}{\sqrt{m}}$

Specify significance level α

Calculate $\widetilde{\boldsymbol{\beta}}$ using Algorithm 3

for *b=1, 2, . . . , B* **do**

Resample $\mathcal{D}^{(n)}$ with replacement to get the b-th bootstrap resample $\mathcal{D}^{(n),b}$

Use Algorithm 3 to calculate $\widetilde{\boldsymbol{\beta}}^b$ with resampled data $\mathcal{D}^{(n),b}$, and initialization $\widetilde{\boldsymbol{\beta}}^b(0) = \widetilde{\boldsymbol{\beta}}$

end

Output:

Calculate $q_{\frac{\alpha}{2}}$ and $q_{1-\frac{\alpha}{2}}$ as the $\frac{\alpha}{2}$ and $1-\frac{d}{2}$ quantiles of $\widetilde{\boldsymbol{\beta}}^b - \widetilde{\boldsymbol{\beta}}$, $b = 1, 2, \ldots, B$.

Report the $(1-\alpha)100\%$ CI as $(\widetilde{\boldsymbol{\beta}} - q_{1-\frac{\alpha}{2}}, \widetilde{\boldsymbol{\beta}} - q_{\frac{\alpha}{2}})$.

A.8 Details for data examples

We describe the datasets used in Section 2.6.2.

METABRIC: This dataset is from the Molecular Taxonomy of Breast Cancer International Consortium (METABRIC) study [61]. This study aimed to determine associations between mutations, driver copy number aberration profiles, clinical-pathological parameters, and survival. It is available in R package `survival` (Therneau, 2015). We dropped patients with any missing features. The resulting dataset has 1904 primary breast cancer samples with nine features: four gene indicators (MKI67, EGFR, PGR, and ERBB2) and five clinical features (hormone treatment indicator, radiotherapy indicator, chemotherapy indicator, ER-positive indicator, age at diagnosis).

FLCHAIN: The aim of Assay Of Serum Free Light Chain (FLCHAIN) [59] was to determine the association between serum free light chain and mortality. We excluded the variable

"chapter" as it is the cause of death. We also dropped patients with any missing features. The resulting dataset has 6542 individuals with eight features: age, sex, the calendar year in which they sample blood (sample year), serum-free light chain, kappa portion (kappa), serum-free light chain, lambda portion (lambda), group of free light chain (FLC group), serum creatinine, and diagnosed with monoclonal gammopathy (0=no, 1=yes).

GBSG: Dataset German Breast Cancer Study Group (GBSG) is from a randomized 2 x 2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients [60]. This dataset is available in R package `mfp` [137]. It has 2,232 individuals (women) with seven features: hormone therapy (0=no, 1=yes), age, menopausal status (0=no, 1=yes), tumor size, tumor grade ($1 < 2 < 3$), number of positive nodes, progesterone receptor, and estrogen receptor. We used dummy variables for the categorical variables (e.g., tumor grade) with more than two categories.

NWTCO: This dataset is from the National Wilm's Tumor [62]. This dataset is available in R package `survival` [138]. This dataset aimed to assess the association between multiple variables and days of tumor relapse. This dataset has 4028 observations with six features: age, disease stage, study group, histology from the local institution, histology from the central lab, and a binary variable indicating whether the observation was included in the subcohort.

SUPPORT: The aim of Study to Understand Prognoses and Preferences for Outcomes and Risks of Treatments (SUPPORT) [63] was to understand prognoses and preferences for outcomes and risks of treatment among seriously ill hospitalized adults. It is publicly available on the Vanderbilt Biostatistics website. We dropped patients with any missing features. The resulting dataset has 8873 patients and 14 features: age, sex, race, number of comorbidities, presence of diabetes, presence of dementia, presence of cancer, mean arterial blood pressure, heart rate, respiration rate, temperature, white blood cell count, serum's sodium, and serum's creatinine.

A.9 Concordance index

The concordance index is a measure of goodness of fit for survival that identifies the level of agreement between predicted risk scores and event times. Without censoring and ties, one can define the concordance index as

$$CI = \frac{\sum_{i,j:T_i<T_j} I[\delta_i = 1]\Big\{I[r_i > r_j] + 0.5I[r_i = r_j]\Big\}}{\sum_{i,j:T_i<T_j} I[\delta_i = 1]}, \tag{A.45}$$

where r_i is the predicted risk score for patient i. With censored observations, the above formula is modified so that only comparable pairs $\{(i,j) : \text{s.t. if} T_i < T_j\,, \delta_i = 1\}$ are considered: $\sum_{i,j:T_i<T_j} I[\delta_i = 1]$ counts the number of such pairs. In this definition, which has been widely used [139], we assign concordance of 1 to a pair of observations if $T_i < T_j$ and $r_i > r_j$ (i.e., the patient with a higher risk has the event first), concordance of 0 if $T_i < T_j$ and $r_i < r_j$ (i.e., the patient with lower risk has the event first), and concordance of 0.5 if $r_i = r_j$ (undecided). For more discussion on this, one can refer to Section 5.1 in [58] (though we have slightly modified things in the case of ties).

For the discrete time-based models (see table 2.2), we use the negative median predicted survival time as the risk score in (A.45).

A.9.1 Oracle concordance for the simulation experiments using the MNIST dataset ($S = 2$)

Figure 2.9 in Section 2.6.2 illustrated how the oracle concordance changes with η and compared it to results from our proposed framework. To calculate the Oracle concordance, we assume that an oracle perfectly identifies the digit values. From this, we can calculate the oracle concordance index as

$$CI_{Oracle}(\eta) = \frac{\sum_{i,j:T_i<T_j} \Big\{I[d_i > d_j] + 0.5I[d_i = d_j]\Big\}}{n_{test}(n_{test} - 1)/2}, \tag{A.46}$$

where $n_{test}(n_{test} - 1)/2$ in the denominator counts the total number of possible pairs from the test data with size n_{test} (with the assumption of no censoring). For the MNIST dataset, we can numerically calculate an upper bound on performance using the oracle concordance

index. We have n_{test} handwritten digits uniformly distributed among 10 digits. Therefore, there is a 10% chance to choose two images from the same class and a 90% chance to choose two images from different classes. The oracle correctly ranks pairs of images from different classes giving a concordance of 1. However, the oracle remains undecided for the images from the same class giving a concordance of 0.5. Averaging these (with weights) we get the oracle concordance index for our generated survival data using the MNIST dataset as $0.9 * 1 + 0.1 * 0.5 = 0.95$.

A.10 Network architectures, hyper-parameters tuning, and stopping criterion

In Section 2.6.2, we compared different methods that build a risk score using neural networks. This section explains the network architectures, the procedure for tuning hyper-parameters, and the stopping criteria.

A.10.1 Network architecture

To have a fair comparison among neural network-based models, we choose the same architecture for the sub-network `Net` in our proposed models and for the networks in other models. The AMSGrad optimizer showed slightly better convergence results for our proposed methods (i.e., `bigSurvSGD`, `bigSurvMLP`, and `bigSurvCNN`) while the Adam optimizer performed slightly better for the other methods. Therefore, we used the AMSgrad optimizer for our proposed methods and the Adam optimizer for the other methods.

Multi-layer perceptrons (MLP) for tabular data

For the non-imaging datasets, we used two fully-connected layers, each with 32 nodes. Each of those layers is followed by a ReLU activation and a dropout layer (with varying dropout rates as a hyper-parameter).

Convolutional neural networks (CNN) for imaging data

For the imaging data-based simulations (generated based on the MNIST dataset), we used three convolution layers with 32, 64, and 64 filters with kernel size (5×5) and stride 1. Each convolution layer is followed by a ReLU activation function, a (2,2) max pool layer with stride 1, and a dropout layer (with varying dropout rates as a hyper-parameter). After all of these layers, the output is flattened. Models `MTLR` and `PMF` treat the survival prediction problem as a multi-class classification problem by partitioning the time axis into non-overlapping bins. The number of nodes in their output layer (that comes after the flattening layer) is precisely the same as the number of bins (classes). For our proposed model `bigSurvCNN`, we use a fully connected layer with 128 nodes after the flattening layer and before the last single-node output layer (that gives the predicted risk score).

A.10.2 Tuning hyper-parameters

Tabular data

Section 2.6.2 compared different methods using tabular datasets. For all methods, we split data into 80% for training and 20% for testing. We train the models with the training dataset and then report the concordance index on the testing dataset. Random Survival Forest (RSF) and neural network (MLP)-based methods have hyper-parameters that need to be tuned to achieve the best performance. We use a 5-fold cross-validation (CV) procedure to tune the hyper-parameters. We hold one fold-out, at a time, as the validation dataset and train our model over the remaining four folds with all possible sets of hyper-parameters. We determine the best set of hyper-parameters that maximizes the average validation concordance index over five validation folds. After tuning the hyper-parameters, we train the model using the entire original 80% training dataset with the tuned hyper-parameters. Then we calculate the concordance index over the 20% hold-out testing dataset. Finally, we report the average of the testing concordance index over 100 random training/testing splits. We next explain the hyper-parameters involved in different models.

Linear models: For our proposed estimator with a linear scoring function (as opposed to a neural network), `bigSurvSGD`, we used parameters: batch size $K = 1$ pair, $C = 0.12$, AMSGrad optimizer, and 100 epochs. We see that these parameters result in a generally strong performance from our other simulated examples (see Figure 2.1 here and Section 2.6.1). In addition, for the full partial-likelihood-based estimator with linear link (for which we used the `R coxph()` implementation), there are no hyper-parameters to tune.

Random Survival Forest (RSF): RSF is an ensemble tree method for survival analysis. It has been shown that constructing ensembles from trees can give strong predictive performance [58]. For this method, we consider the following hyper-parameter values: the number of trees in the forest $(100, 500)$, the minimum size of terminal node $(5, 15)$, and the number of features randomly selected as candidates for each node split $(2, 3, 4, 5)$.

MLP-based methods: Methods `BigSurvMLP`, `CoxCC`, `CoxTime`, `DeepSurv`, `PCHazard`, `PMF`, `DeepHit`, and `MTLR` use the neural network (MLP) to estimate the risk function $f_{\boldsymbol{\beta}}(\boldsymbol{x})$. For these methods, we consider the common hyper-parameters: the batch size $(64, 256)$, the learning rate $(0.001, 0.0001)$, the dropout rate $(0.0, 0.2, 0.4)$, and the number of epochs (up to 200 epochs) for training. Discrete-time methods `PCHazard`, `PMF`, `DeepHit`, and `MTLR` divide time axis into non-overlapping bins. For discrete-time models, we use quantiles to divide the time axis into 100 bins (i.e., classes). When tuning the number of epochs, we observed that there is no need to iterate up to 200 epochs. To save computation time, we use a stopping criterion to stop training when it does not improve the generalization error (or concordance index, alternatively). For each training/testing (i.e., 80%/20%) split, we tune the hyper-parameters for the MLP-based models as follows:

- We randomly split the training dataset into five folds,
  - We hold one fold out for validation and consider the remaining four folds for training,
    - For each combination of hyper-parameters (except the number of epochs) we

do the following,

- · We train the model on the four training folds until we are confident that there will be no improvement of the validation concordance index by further training. For this purpose, we use a stopping criterion to specify when (at what epoch) to stop (see Section A.10.3 for more details),
- · We determine the maximum validation concordance index along with the corresponding number of epochs,

* We repeat the above two sub-steps for all combinations of the hyper-parameters,

– We repeat the above sub-steps on all five folds where each fold is considered as the validation fold at a time,

- We determine the values of the hyper-parameters that maximize the average validation concordance index over the five hold-out validation folds,

- For the optimal values of hyper-parameters identified in the previous step, we calculate the average number of selected epochs over the five hold-out validation folds.

CNN for imaging data

For imaging data, there are only three models that facilitate the use of the convolutional neural network (CNN): our proposed CNN-based model `BigSurvCNN`, `MTLR`, and `PMF`. We simulated time-to-event outcomes using the MNIST dataset with 60,000 grayscale handwritten images for training and 10,000 for testing (see Section 2.6.2 for more details). We keep the provided 10,000 testing images as the fixed hold-out testing dataset. To explore the effects of data constraints (e.g., sample size), we sample a fraction of the original 60,000 training images for training purpose. We randomly select 100 of such sampled training datasets with the size N_{sample}. We train our models over the sampled dataset and report the concordance index over the hold-out testing dataset. All three models require tuning of several hyper-

parameters. We consider the following hyper-parameters for the three models: the batch size $(64, 256)$, the learning rate $(0.01, 0.001, 0.0001)$, dropout rate $(0.0, 0.2, 0.4)$, l_2 penalty coefficient $(0.0, 10^{-2}, 10^{-1})$, and number of epochs up to 1000. For discrete models `MTLR`, and `PMF`, we consider the number of bins to partition time into as an extra hyper-parameter with possible values of $(10, 100)$. We use the following steps for tuning the hyper-parameters:

- We randomly split the sampled dataset (with size N_{sample}) into training/validation datasets (80% training 20% validation),
  - For each combination of hyper-parameter values (except the number of epochs) we do the following,
    - We train the model on the 80% training portion until we are confident that there will be no improvement of the validation concordance index by further training. For this purpose, we use a stopping criterion to specify when (at what epoch) to stop (see Section A.10.3 for more details),
    - We determine the maximum validation concordance index along with the corresponding number of epochs,
- We determine the values of the hyper-parameters (except the number of epochs) that maximizes the validation concordance index,
- For the optimal values of hyper-parameters specified in the previous step, we identify the optimal number of epochs,
- We train the model using the original sampled training dataset (with N_{sample} samples) with the tuned hyper-parameters and report concordance index on the hold-out testing dataset.

Finally, we report the average testing concordance index for the trained models over 100 sampled training datasets.

A.10.3 Early stopping criterion

It is crucial to determine the ideal training length for a neural network: While too little training gives an under-fit model, too much training over-fits and results in poor performance on the test dataset. One common approach to deal with this is to train the model on the training dataset until the performance on a validation dataset stops improving. This widely used approach to training neural networks is known as *early stopping* [121]. In practice, the validation error curve is not usually smooth, with some stochastic behavior due to our use of stochastic optimization: It might have several nearby local minima [121]. To deal with this, one can continue training for a small number of epochs past the initially identified local minimum to increase confidence that there will be no later improvement in performance on the validation set. This approach is known as *patience* [121]. In our implementation, we propose an early stopping criterion based on a weighted moving averaged validation concordance index instead of the single validation concordance index calculated at each epoch. The weighted moving averaged concordance index considers higher weight on more recent concordance indices than past indices. For instance, at the j-th epoch of training, the i-th ($i \leq j$) validation concordance index receives weight w^{j-i}, where w ($0 \leq w \leq 1$) is the moving average weight. Note that $w = 0$ returns only the latest validation concordance index (we define $0^0 \equiv 1$ in this instance), while $w = 1$ returns the simple average of validation concordance indices. In our implemented early stop criterion, we wait for e_p (*patience*) epochs before stopping early if there is no improvement on the weighted moving averaged validation concordance index. Compared with the curve of single concordance indices, the weighted moving average smooths out the curve and better deals with the stochastic behavior of the validation concordance index. Therefore, the early stopping criterion based on the weighted moving averaged validation concordance index increases our confidence that there will be no further improvement in validation performance when training is stopped. Algorithm 6 summarizes our proposed early stopping criterion used in our implementation. For all methods, we chose the extra number of epochs (patience) before early stopping as $e_p = 5$.

For our implementation of the early stopping criterion, we chose the moving average weight factor as $w = 0.8$. Although we present our stopping criterion based on the concordance index (as a measure of accuracy) rather than loss, one can easily modify it based on loss or the other evaluation criteria.

Algorithm 6: Early stopping criterion

Result: Maximum validation concordance index and corresponding number of epochs

Initialization:

Moving average weight factor: w

Maximum number of epochs: e_{max}

Extra number of epochs (patience) before early stopping: e_p

Start with $e = 1$:

while $e \leq e_{max}$ **do**

Calculate validation concordance index $CI^{valid}[e]$

Calculate the weighted moving averaged validation concordance index as

$$CI_{MA}^{valid}[e] = \frac{\sum_{j=1}^{e} w^{j-1} CI^{valid}[j]}{\sum_{j=1}^{e} w^{j-1}}.$$

if $\Big(e > e_p\Big)$ *and* $\max\Big(CI_{MA}^{valid}[e], CI_{MA}^{valid}[e-1], \ldots, CI_{MA}^{valid}[e-e_p+1]\Big) < CI_{MA}^{valid}[e-e_p]$ **then**

Stop training,

$e_{max} = e$.

else

$e = e + 1$,

Continue training

end

end

Report $CI_{opt}^{valid} = \max\limits_{e=1,\ldots,e_{max}} \big\{CI^{valid}[e]\big\}$ as the maximum validation concordance index and $e_{opt} = \operatorname{argmax}\limits_{e=1,\ldots,e_{max}} \big\{CI^{valid}[e]\big\}$ as the corresponding number of epochs.

Calculate validation concordance index $CI^{valid}[e]$

A.11 Complementary results

A.11.1 Effect of misspecification of the initial learning rate

Choosing an appropriate learning rate may help improve the statistical efficiency of our estimator for a fixed number of epochs. In practice, we have found that `AMSGrad` is much more robust to misspecification of the learning rate. The left panel in Figure A.1 compares MSE with varying choices of C for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$. We use AveAMSGrad over 1000 simulated datasets (based on the data generation mechanism explained in Section 2.6.1 with the probability of censoring $p_c = 0.2$) with 100 and 1000 epochs. We observe choosing an appropriate value of C (around 0.1) gives a strong performance. Choosing smaller or larger values of C may necessitate more than 100 epochs: Smaller values of C (e.g., $C = 0.05$) result in too little "learning" in each step of the SGD algorithm, while higher values of C result in too much noise. The right panel of Figure A.1 considers the same settings as the left panel, except we increase the number of epochs from 100 to 1000. As we see, our algorithm with the higher number of epochs is relatively robust to a wider range of C around the empirically optimal value.

A.11.2 Effect of number of sample strata on coverage for plugin approach

In Section A.7, we gave a plugin approach for constructing a $(1-\alpha)\times 100\%$ CI. Although there are $\binom{n-1}{s-1}$ possible strata per each observation, in practice, we prefer to consider a random small fraction of them to estimate the standard error. Figure A.2 presents the coverage of 95% CI constructed using the plugin approach with a varying number of strata sampled per observation (i.e., n_o). We considered 100 simulated datasets (based on the data generating mechanism explained in Section 2.6.1 with a probability of censoring $p_c = 0.2$) with 100 epochs to estimate $\widetilde{\boldsymbol{\beta}}$. We observe sampling between 100 and 1000 strata per observation (n_o) gives near nominal coverage. This makes our computation tractable, especially for large datasets (e.g., for sample size $n = 10000$, we only need 100 sample strata instead of going through all $\binom{9999}{19} \approx 8.06 \times 10^{58}$ sample strata).

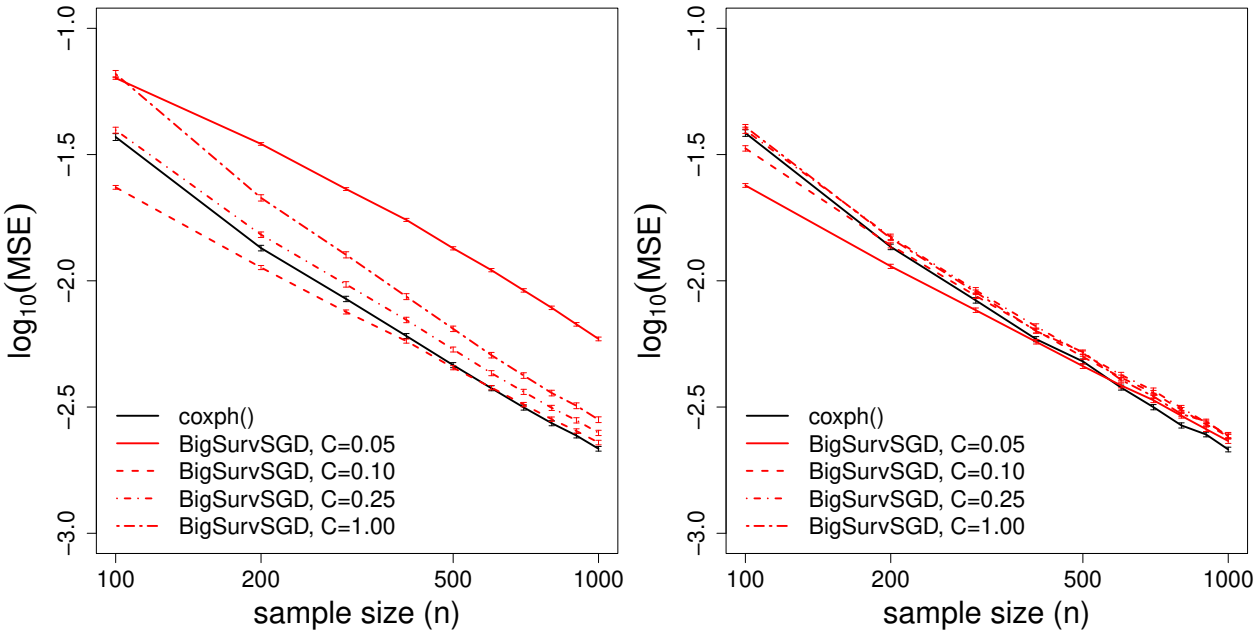

Figure A.1: (left panel) $\log_{10}$(MSE) for varying values of C with 100 epochs; (right panel) $\log_{10}$(MSE) for varying values of C with 1000 epochs. We choose mini-batch size $K = 1$, strata size $S = 20$, $P = 10$ features, and probability of censoring $p_c = 0.2$.

A.11.3 Estimation of hazards ratio with 95% confidence interval

We present results of estimates of the hazard ratio with 95% confidence intervals for the FLCHAIN and GBSG datasets. For the FLCHAIN dataset, variable *lambda* has a heavily right-skewed distribution and variable *mgus* is unbalanced (with 99% value 0 and 1% value 10). For the GBSG dataset, variables *tumor size*, *posnodal*, *prm*, and *esm* have right-skewed distributions and variable *tumor grade* is a categorical variable with three categories. We chose strata size $S = 20$, mini-batch size $K = 1$, $C = 0.12$ in the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$, number of epochs 100, and AMSGrad algorithm with an average of estimate over iterates (AveAMSGrad). For the plugin method, we used $n_o = 1000$ strata per observation to estimate the standard error. For the bootstrap method, we used $B = 1000$ bootstrap resamples with 100 epochs. Table A.1 and A.2 present estimates of hazard ratio with a 95%

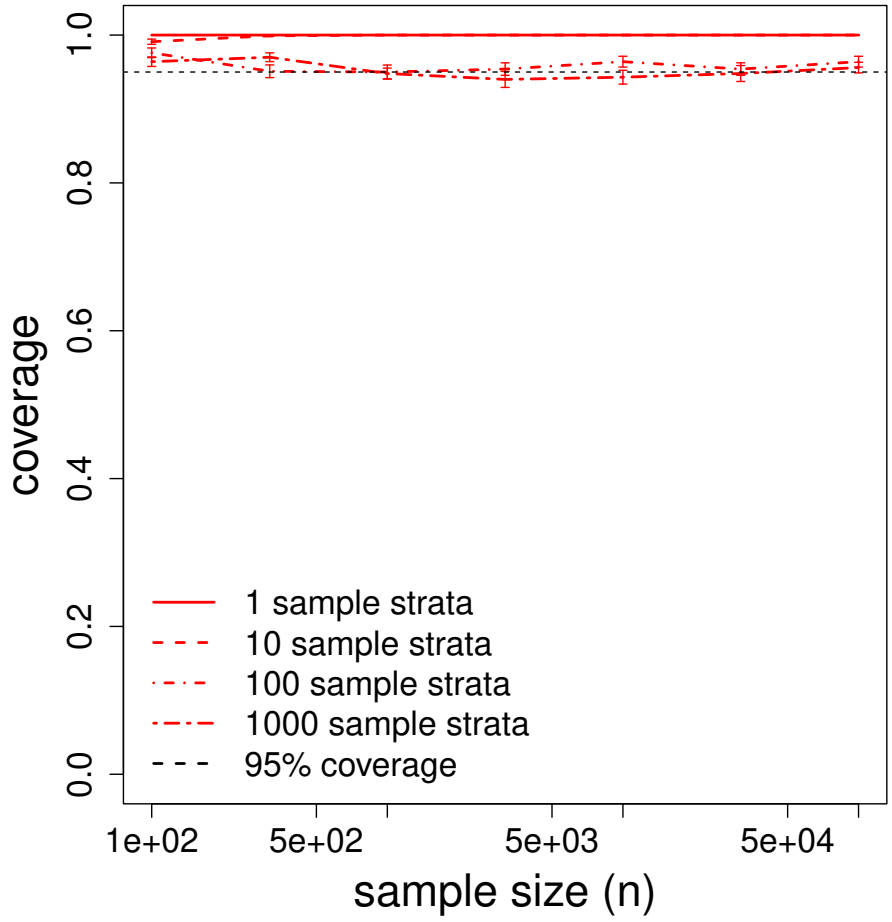

Figure A.2: Coverage versus number of samples per observation (n_o) for estimating 95% CI using plugin method. We used AveAMSGrad with empirically optimal C for learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$), batch size $K = 1$, number of epochs 100, $P = 10$ features, strata size $S = 20$, and probability of censoring $p_c = 0.2$.

confidence interval for the FLCHAIN and GBSG datasets. As we see, all three approaches give very close estimates of hazard ratio and 95% confidence intervals for all covariates' coefficients.

Table A.1: Estimates of hazards ratio and 95% confidence interval for different covariates in dataset FLCHAIN ($n = 6542$, $P = 8$, $p_c = 0.7$) with `coxph()` and BigSurvSGD (AveAMSGrad). We use strata size $S = 20$, batch size $K = 1$, number of epochs 100, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

	`coxph()`	BigSurvSGD-plugin	BigSurvSGD-bootstrap
age	1.107 (1.102, 1.113)	1.107 (1.099, 1.114)	1.107 (1.100, 1.113)
sex	0.756 (0.688, 0.831)	0.755 (0.674, 0.847)	0.756 (0.673, 0.844)
sample year	1.056 (1.017, 1.096)	1.054 (1.015, 1.095)	1.054 (1.008, 1.105)
kappa	1.018 (0.952, 1.088)	1.031 (0.939, 1.130)	1.029 (0.891, 1.149)
lambda	1.185 (1.122, 1.251)	1.181 (1.111, 1.254)	1.183 (1.097, 1.311)
FLC group	1.057 (1.035, 1.079)	1.056 (1.032, 1.080)	1.056 (1.026, 1.090)
serum creatinine	1.036 (0.941, 1.141)	1.044 (0.932, 1.170)	1.044 (0.854, 1.244)
mgus	1.303 (0.792, 2.146)	1.292 (0.817, 2.045)	1.293 (0.857, 2.976)

A.11.4 Computing time for different methods

In Section 2.6.1, we showed that our proposed methodology gives reliable estimates and inference (in-line with results from maximization of the full partial likelihood, when that method works) and it avoids memory issues. Figure A.3 compares the computing time (in seconds) of the i) proposed framework estimating only the coefficients; ii) the proposed method with bootstrap or plugin approach to estimate variance, and iii) the standard `coxph()`. We evaluated these methods for a varying number of observations (n) and covariates ($P = 10, 20$). The simulations to get Figure A.3 were conducted on a quad-core Intel Core i7-2600 with 12 GB RAM. The computing time of all frameworks increases almost linearly in the sample size, n. Figure A.3 shows that the current implementation of our methodologies is slower than `coxph()`. The bootstrap methodology with $B = 10$ re-samples (suffices to give a correct coverage) suffers less, and the computing time with and without calculating variance is close

Table A.2: Estimates of hazards ratio and 95% confidence interval for different covariates in GBSG dataset ($n = 2232$, $P = 7$, $p_c = 0.43$) with `coxph()` and BigSurvSGD (AveAMSGrad). We use strata size $S = 20$, batch size $K = 1$, number of epochs 100, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

	`coxph()`	BigSurvSGD-plugin	BigSurvSGD-bootstrap
htreat	0.707 (0.549, 0.911)	0.687 (0.533, 0.886)	0.689 (0.535, 0.909)
age	0.991 (0.973, 1.009)	0.988 (0.970, 1.008)	0.987 (0.967, 1.007)
menopausal status	1.295 (0.904, 1.855)	1.342 (0.921, 1.955)	1.370 (0.935, 1.973)
tumor size	1.008 (1.000, 1.016)	1.008 (1.000, 1.016)	1.008 (0.999, 1.017)
posnodal	1.050 (1.035, 1.065)	1.056 (1.032, 1.086)	1.057 (1.019, 1.081)
prm	0.998 (0.997, 0.999)	0.998 (0.997, 0.999)	0.998 (0.997, 0.999)
esm	1.000 (0.999, 1.001)	1.000 (0.999, 1.001)	1.000 (0.999, 1.001)
tumor grade 1	0.459 (0.271, 0.776)	0.463 (0.281, 0.763)	0.466 (0.294, 0.833)
tumor grade 2	0.866 (0.663, 1.131)	0.874 (0.661, 1.158)	0.872 (0.647, 1.143)

to each other. The reason is that it can be easily paralleled over the bootstrap re-samples. In small sample sizes it is definitely better to use the standard `coxph()`, however, as scalability becomes an issue, for massive datasets, our framework performs quite well (and `coxph()` can no longer be used).

A.11.5 Predicted risk scores for imaging data

In Section 2.6.2, we presented results for the concordance index with a minimalist architecture for sub-network `Net`. We observed that the performance improves when training sample size and η (i.e., the proportion of risk score to digit value) increases. The main reason for getting a poor performance for small values of the training sample size and η is that the network may not be able to predict the risk score (i.e., $f_\beta(\boldsymbol{x})$) well enough. Figure A.4

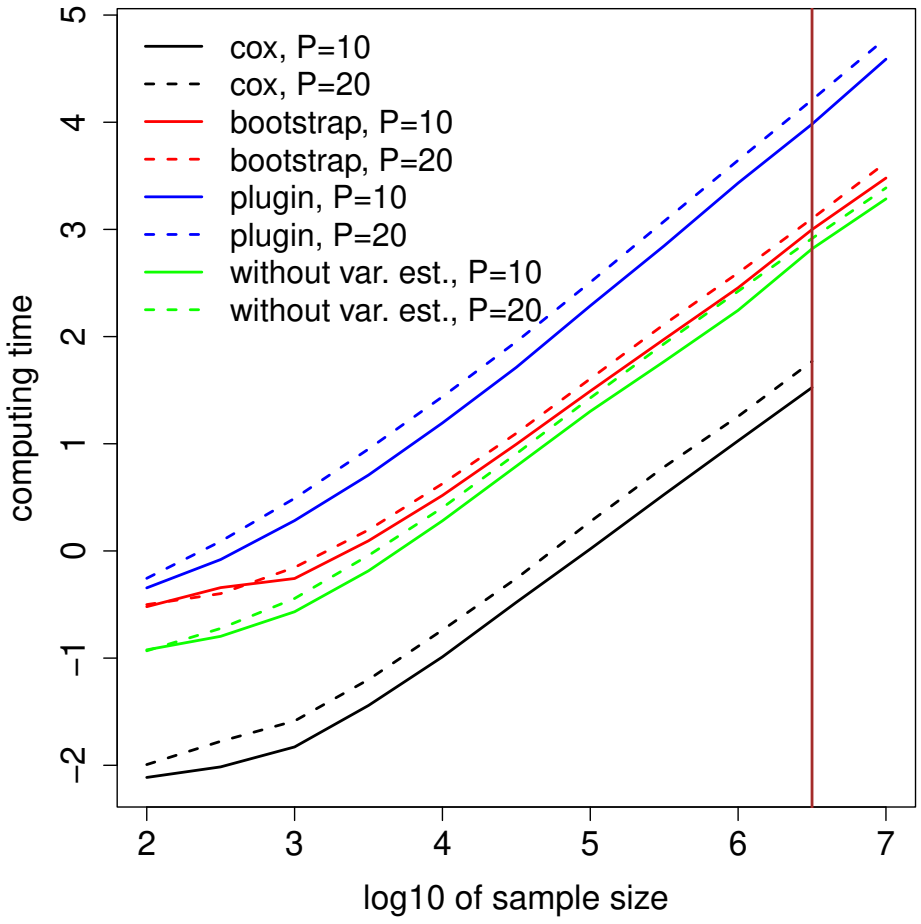

Figure A.3: Computing time with proposed bootstrap and plugin methods and `coxph()` for different sample sizes (n) and feature sizes (p). We choose mini-batch size $K = 1$, number of epochs 100, $B = 10$ bootstrap samples for the bootstrap method (we observed this number suffices to give a correct coverage), $n_o = 100$ samples per observation for the plugin method, probability of censoring $p_c = 0.2$, and the optimal value $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

presents the predicted risk scores (centered by subtracting the mean) versus true digits for $n_{test} = 10,000$ testing images using the minimalist architecture for sub-network `Net`. The network architecture, hyper-parameters tuning procedure, and optimizer are the same as

what we considered to get Figure 2.9. We observe that the predicted scores for different digits become more separated when training sample size n_{train} and η increases. More separated predicted risk scores result in a better ranking of digit pairs, consistent with what we observed in Figure 2.9. By comparing top images with bottom ones, we also observe that increasing sample size improves the prediction only when there is enough signal in the images.

A.11.6 More simulation results with different types of covariates

In this section, we compare our methodologies with the current state-of-the-art `coxph()` for more simulated scenarios. We consider scenarios including (1) fewer covariates with different signal levels, (2) categorical covariates, (3) heavily-skewed covariates, and (4) correlated covariates. For all simulated data, given the covariate matrix $\boldsymbol{x}$, we generate time-to-event/censoring y and event status δ as

$$y \sim exp\Big(\mu = \exp\left(-\boldsymbol{x}^T\boldsymbol{\beta^*}\right)\Big)$$
$$\delta \sim Bernoulli\Big(p = 1 - p_c\Big),\ p_c = Pr(t > c).$$

For all simulation results presented here, we choose mini-batch size $K = 1$, number of epochs 100, $B = 100$ bootstrap samples for the bootstrap method, $n_o = 100$ samples per observation for the plugin method, probability of censoring $p_c = 0.2$ (except for rare events), and the proportional constant value $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$. We average 100 random Monte Carlo simulations to report the results.

We consider $P = 10$ covariates in total. To examine different levels of signal, we choose $\boldsymbol{\beta^*} = \beta \times (1, \overrightarrow{0}_4, 1, \overrightarrow{0}_4)$, i.e., only $p_s = 2$ covariates among all covariates carry a signal through β. In all of our extended simulations in this section, we choose three values for β: 0.1 for (weak-level signal), 0.5 (medium-level signal), and 1.0 (high-level signal). Note that we limit our analysis to small-to-moderate datasets that allow us to compare `coxph()` to our proposed framework. However, an additional benefit of our proposed framework over the standard `coxph()` is that it supports large datasets without encountering computational issues and could scale up to nearly arbitrary numbers of observations.

Simulated survival dataset with uniform features

In this scenario, we generate $P = 10$ uniform covariates and we normalize them to have variance one. Figure A.5 compares MSE and coverage of our proposed methodologies to the current state-of-the-art `coxph()`. We observe that our proposed methodologies and `coxph`() perform quite well for varying sample size (n) and signal level β.

Simulated survival dataset with categorical features

In this scenario, we generate $P = 10$ binary covariates and we normalize them to have variance one. Figure A.6 compares MSE and coverage of our proposed methodologies to the state-of-the-art `coxph()`. We observe that our proposed methodologies and `coxph`() perform quite well for varying sample size (n) and signal level β.

Simulated survival dataset with heavily-skewed features

In this scenario, we generate right-skewed covariates using a log-normal distribution with parameters $\mu = 0$ and $\sigma = 1$. We normalize all covariates to have variance one. Figure A.7 compares MSE and coverage of our proposed methodologies with the state-of-the-art `coxph()`. We notice that when the sample size n and signal level β grow, `coxph()` performs worse, whereas our methodologies improve as n and β increase. Although we normalize the covariates to have variance 1, skewed covariates have large outliers, and `coxph()` becomes computationally unstable because of its recursive updating mechanism.

Simulated survival dataset with both categorical and heavily-skewed features

In this scenario, we investigate both categorical and skewed covariates. Additionally, we allow correlation between all covariates. To begin, we generate $P = 10$ covariates using a multivariate normal distribution with a mean zero and an exchangeable correlation structure with a coefficient of ρ. We then apply marginal transformations to obtain the specified marginal distributions.

Figures A.8, A.9, and A.10 compare MSE and coverage of our proposed methodologies with the current state-of-the-art `coxph()` for coefficients $\rho = 0.2$, $\rho = 0.5$, and $\rho = 0.8$. We observe that when the sample size n, signal level β and ρ increase, `coxph()` starts to perform worse. Our proposed methodologies improve as n and β increase and remain robust against varying values of correlation coefficient ρ.

A.11.7 Rare event with strata size $S = 50$

In this section, we examine the effect of the probability of censoring with a different (higher) strata size, i.e., S=50. Figure A.11 compares our proposed framework to the current state-of-the-art `coxph()` with sample size $n = 1000$, varying values of the censoring probability. Outside of the strata size, all other settings are the same as we presented in Section 2.6.1 for $S = 20$. We observe that our proposed methodology performs very closely to `coxph()`. By comparing Figure A.11 here to Figure 2.4, we observe that choosing a higher strata size improves efficiency. However, a choice of $S = 20$ may be good enough in practice.

A.11.8 Correlated covariates and dependent censoring

In Section 2.6.1 of Chapter 2, presented a scenario for a mis-specified model where we missed including the second covariate affecting the event time. We assumed that two covariates affecting the event time are independent. In this section, we present two other scenarios, one for dependent covariates and the other one for dependent censoring.

Correlated covariates

Suppose that we generate event time T and censoring time C based on hazards functions $h_T(t, x)$ and $h_C(t, x)$ as following

$$\begin{aligned} h_T(t, x) &= h_0(t)e^{\gamma_1 x + \gamma_2 x^2}, \\ h_C(t, x) &= h_0(t)e^{\alpha x}, \end{aligned} \tag{A.47}$$

where γ_1 and γ_2 formulate the relationship between the event time T, and feature x and its quadratic term x^2; α formulates the relationship between the censoring time C and feature x. Similar to the scenario presented in 2.6.1, the observation time and event status are given by (2.14). Here we only observe x and miss to include term x^2 (which is correlated with x), and we aim to study the effect of covariate x by the minimization problem given by (2.5). In particular, we are interested in knowing that how different values of γ_1, γ_2, and α affect the estimate of $\beta^{(s)}$. For simulations, we generate x from a uniform distribution, i.e., $x \sim U(0,1)$ and then we use (A.47) and (2.12) to generate the survival data with sample size $n = 10^5$. All the following results are based on averaging over 100 randomly generated survival datasets. Figure A.12 illustrates the average censoring rates for different values of γ_1, γ_2, and α. As we expect, the censoring rates increase by α, and decrease by γ_1 and γ_2. Figure A.13 compares the average estimate of parameter $\beta^{(s)}$ for different values of γ_1, γ_2, and α. With $\gamma_2 = 0$ (the first column in Figure A.13), i.e., no model mis-specification, the estimate of the parameter $\beta^{(s)}$ is about the same as γ_1 for all models except for higher values of α where our proposed framework with strata size $s = 2$ suffers from very low event rate. With $\gamma_1 = 0$ (the first row in Figure 2.7), i.e., there no effect from x but there is an effect from its quadratic form x^2, all methods perform poorly when the model is mis-specified (i.e., $\gamma_2 \neq 0$). For other scenarios, when there is a linear effect from covariate x and the model is mis-specified (i.e., $\gamma_2 \neq 0$ and $\gamma_1 \neq 0$), the optimization with different strata sizes gives an estimate of $\beta^{(s)}$ which is far from γ_1.

Dependent censoring

Suppose that our event time T and censoring time C are generated based on hazards functions $h_T(t,x)$ and $h_C(t,x)$ as following

$$\begin{aligned} h_T(t,x) &= h_0(t)e^{\gamma_1 x_1 + \gamma_2 x_2}, \\ h_C(t,x) &= h_0(t)e^{\alpha(\gamma_1 x_1 + \gamma_2 x_2)}, \end{aligned} \tag{A.48}$$

where γ_1 and γ_2 formulate the relationship between the event time T, and features x_1 and x_2; α, γ_1, and γ_2 formulate the relationship between the censoring time C and features x_1 and x_1. In this scenario, conditioned on observing x_1, the event and censoring times are dependent (also called residual dependent). Here, we only observe x_1 and missed to include covariate x_2. We are interested in knowing that how different values of γ_1, γ_2, and α affect the estimate of $\beta^{(s)}$ from optimization procedure (2.5) for different values of strata size s. For simulations, we independently generate x_1 and x_2 from a uniform distribution, i.e., $x_1, x_2 \sim U(0, 1)$ and then we use (A.48) and (2.12) to generate survival data with sample size $n = 10^5$. All the following results are based on averaging over 100 randomly generated survival datasets. Figure A.14 illustrates the average censoring rate for different values of γ_1, γ_2, and α. By noting that $x_1, x_2 > 0$, the censoring rates increases by α, γ_1, and γ_2. Figure A.15 compares the average estimate of parameter $\beta^{(s)}$ for different values of γ_1, γ_2, and α. With $\gamma_1 = 0$ (the first row in Figure A.15), i.e., there no effect from x_1, the estimate of $\beta^{(s)}$ is very close to γ_1. With $\gamma_2 = 0$ (the first column in Figure A.15), i.e., no model mis-specification, the estimate of the parameter $\beta^{(s)}$ is about the same as γ_1 for all models except for higher values of α where optimization procedure with $s = 2$ suffers from very low event rate. For other scenarios, when the model is mis-specified (i.e., $\gamma_2 \neq 0$ and $\gamma_1 \neq 0$), when the event rate is not very low, optimization with strata size $s = 2$ outperforms others. But the estimate of $\beta^{(s)}$ from (2.5) with $s = 2$ suffers more when the censoring rate increases.

A.12 Round-off error

In Chapter 2, we discussed that the current gold standard `coxph()` is susceptible to round-off error, resulting in computational instability. The reason for this is because of the trick used to decrease the computational complexity from $O(n^2)$ to $O(n)$. In this section, we explain how such a round-off error happens. As a reminder, the gradient of the log full-partial

likelihood is given by (see equation (2.4))

$$\nabla_\beta \Big\{ - pl^{(n)}(\boldsymbol{\beta}|\mathcal{D}^{(n)}) \Big\} = -\sum_{i=1}^{n} \Big(\boldsymbol{x}^{(i)} - \frac{\sum_{j\in\mathcal{R}_i} \boldsymbol{x}^{(j)} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(j)}\right)}{\sum_{j\in\mathcal{R}_i} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(j)}\right)} \Big). \tag{A.49}$$

Suppose we do not have censoring and ties, and all patients are ordered based on their time-to-event, where patients with smaller time-to-event come first. Because of the nested summations, a naïve calculation would have computational complexity on the order of $O(n^2)$. Since we have $R_i = R_{i+1} \cup \{i\}$, we can write

$$\begin{aligned} \sum_{j\in\mathcal{R}_i} \boldsymbol{x}^{(j)} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(j)}\right) &= \sum_{j\in\mathcal{R}_{i+1}} \boldsymbol{x}^{(j)} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(j)}\right) + \boldsymbol{x}^{(i)} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right) \\ \sum_{j\in\mathcal{R}_i} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(j)}\right) &= \sum_{j\in\mathcal{R}_{i+1}} \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(j)}\right) + \exp\left(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right). \end{aligned} \tag{A.50}$$

Therefore, one can start with the n-th observation and gradually update the numerator and denominator of (A.49) using the idea in (A.50). This trick reduces the computation complexity to $O(n)$ but comes with a cost in stability: Round-off error when you add a minimal number to a very large number or consider something like $[(small + large) - large]$ [140]. In the standard `coxph()` implementation, the chance of round-off error increases as (1) the number of high-signal features grows and (2) the number of observations grows. With more high-signal features, the chance of including two observations with significantly different risks (in different orders of magnitude) in the same risk set grows. When we use the trick in (A.50) with many observations, round-off error may accumulate when updating the numerator and the denominator in (A.49).

A.13 Choosing strata size s

In the case of a correctly specified model, one can use U-statistic theory (see Section 2.4) to derive the variance of our proposed estimator as a function of strata size s. We found that analytically evaluating the effect of s on that variance is difficult (it is unclear to us how to simplify terms). Instead, we evaluated this effect empirically as shown in Figure 2.1. We found that choosing strata size $s = 2$ (the smallest possible value) results in a small, but

potentially noticeable, loss in efficiency. For strata sizes, $s = 5$ and $s = 10$, the amount of loss is, by our determination, practically negligible and we have almost no loss for $s = 20$ or higher. In the case of model mis-specification, we found that choosing strata size $s = 2$ is very interpretable as we optimize a smoothed pairwise concordance (see Table 2.2 and Figure 2.9). We also investigate the effect of strata size when the event is rare (Figures 2.4 and A.11). We observed that when the model is correctly specified and the event is rare, choosing $S = 20$ generally suffices to achieve the correct estimate and inference (though to be safe, one might choose $S = 50$ in this case). Our recommendation is to use $S = 20$ in the case that you strongly believe in linearity and proportional hazards and to use $S = 2$ if instead, you are just using the partial-likelihood-loss for risk stratification.

A.14 Our big data problem versus high-dimensional data

In Section 2.6.1, we presented results for big data scenarios. We allow the number of observations n to grow while we keep the number of covariates fixed such that $p \ll n$. While it is well known that an unregularized model will not work well in a high-dimensional problem, we are illustrating different phenomena here. The problem we engage with is not particularly a high dimensional problem, in that $p \ll n$. Here, the issue is related to computational instability as opposed to some curse of dimensionality — this is illustrated via the simulation result in Figure 2.1 in Section 2.6.1 as our proposed method (which is also unregularized) gives good estimates (and more specifically, unlike `coxph()` the performance of the method doesn't degrade as n increases). However, we have decreased focus on this, as the full signal scenario is admittedly unrealistic, and have instead focused more on the broadened simulation experiments.

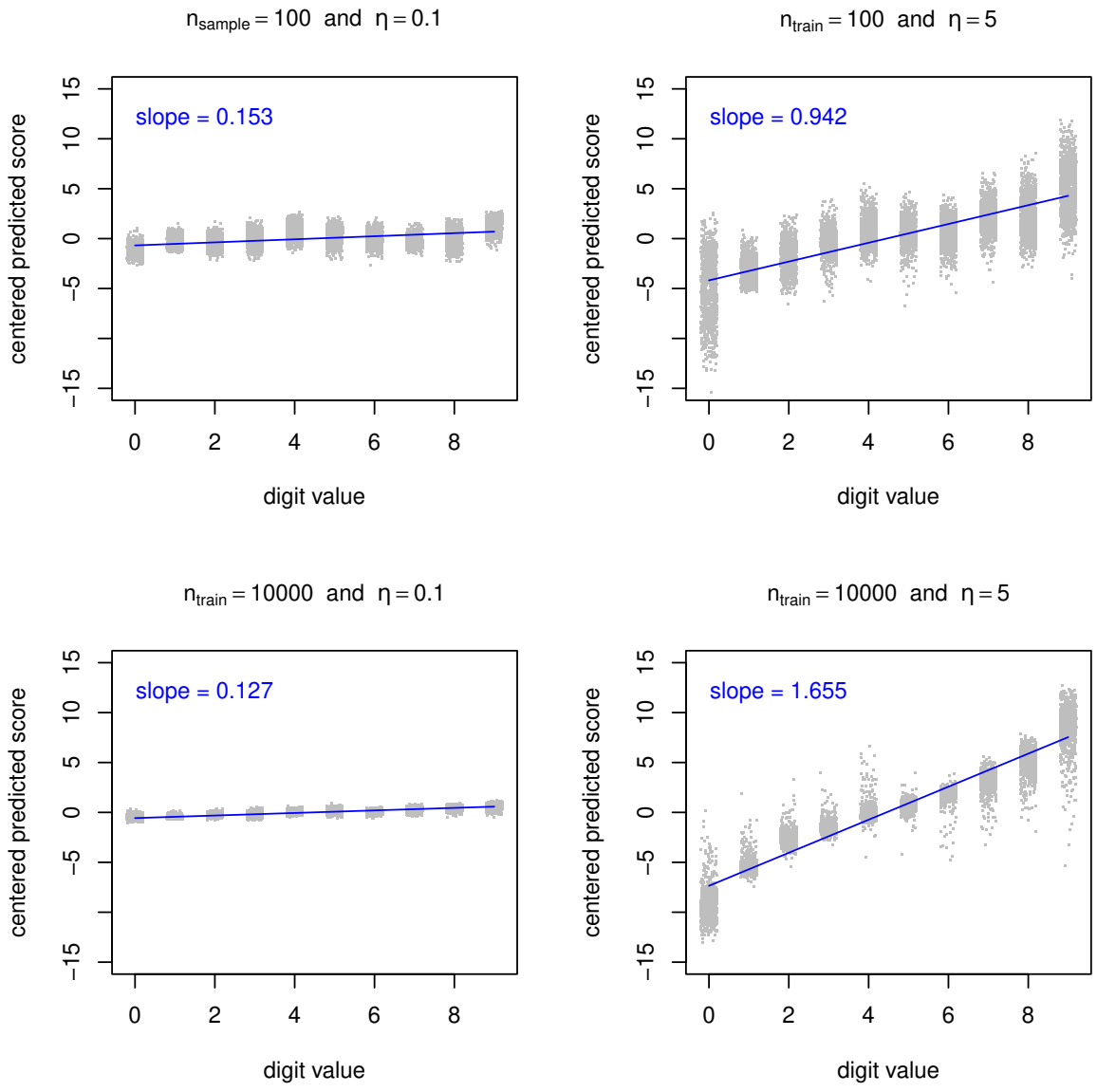

Figure A.4: Centered predicted risk scores ($f_{\boldsymbol{\beta}}(\boldsymbol{X}_i)$) versus centered true digit value (d_i) for handwritten digit i in testing data ($i = 1, 2, \ldots, n_{test} = 10000$) for our proposed methods `bigSurvCNN`. The blue line is the best linear line regressing all centered predicted scores versus their true digit values. The network architecture and parameters are exactly the same as we considered in Section 2.6.2 to get Figure 2.9 (see Appendix A.4 for more details about the network architecture and the procedure for tuning hyper-parameters).

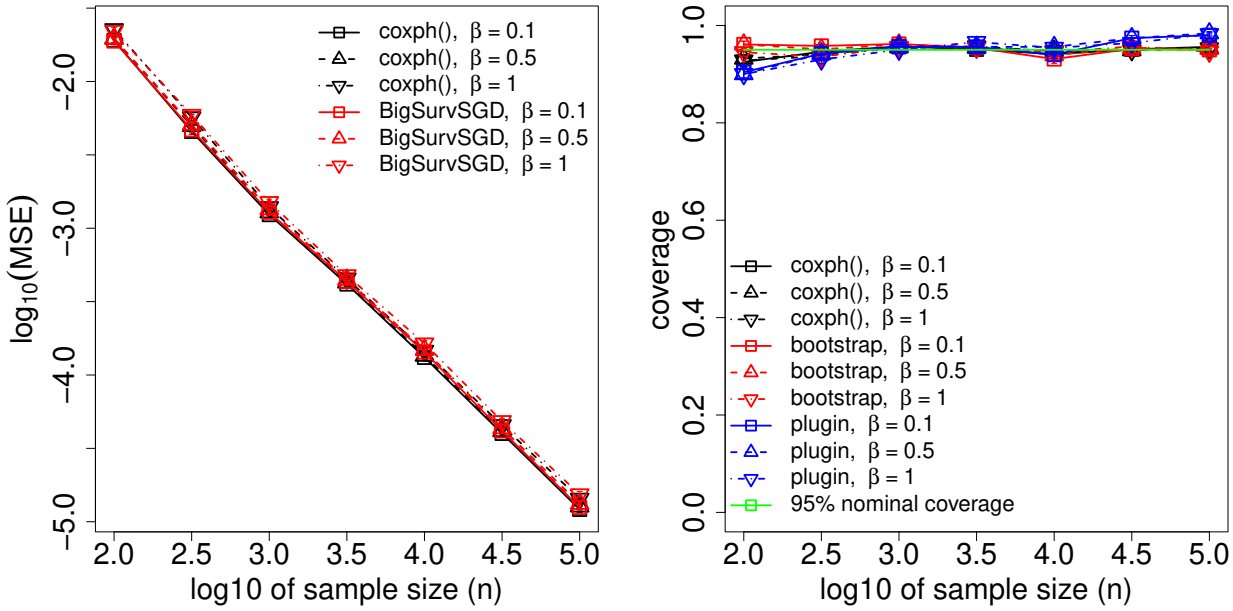

Figure A.5: Comparing $\log_{10}$(MSE) (left panel) and coverage (right panel) for varying sample size n and varying level of signal β. The total number of uniform features (with variance 1) is $P = 10$ but only $P_{sig} = 2$ among them have a signal level of β. We choose mini-batch size $K = 1$, number of epochs 100, $B = 100$ bootstrap samples for the bootstrap method, $n_o = 100$ samples per observation for the plugin method, probability of censoring $p_c = 0.2$, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

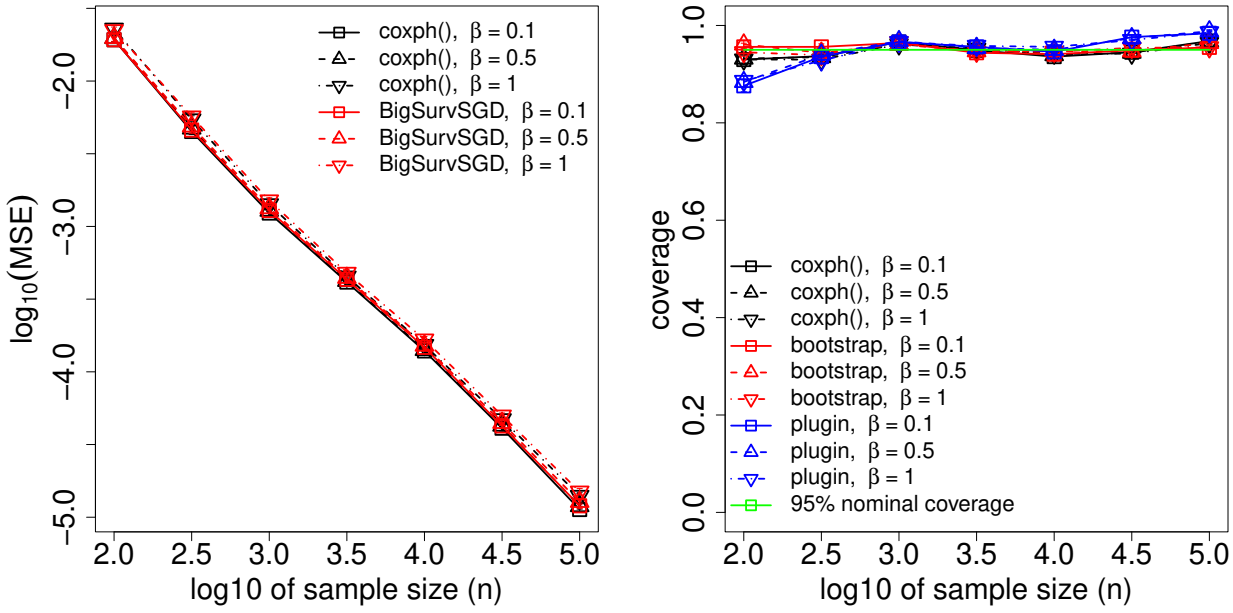

Figure A.6: Comparing $\log_{10}$(MSE) (left panel) and coverage (right panel) for varying sample size n and varying level of signal β. The total number of binary features (with variance 1) is $P = 10$ but only $P_{sig} = 2$ among them have a signal level of β. We choose mini-batch size $K = 1$, number of epochs 100, $B = 100$ bootstrap samples for the bootstrap method, $n_o = 100$ samples per observation for the plugin method, probability of censoring $p_c = 0.2$, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

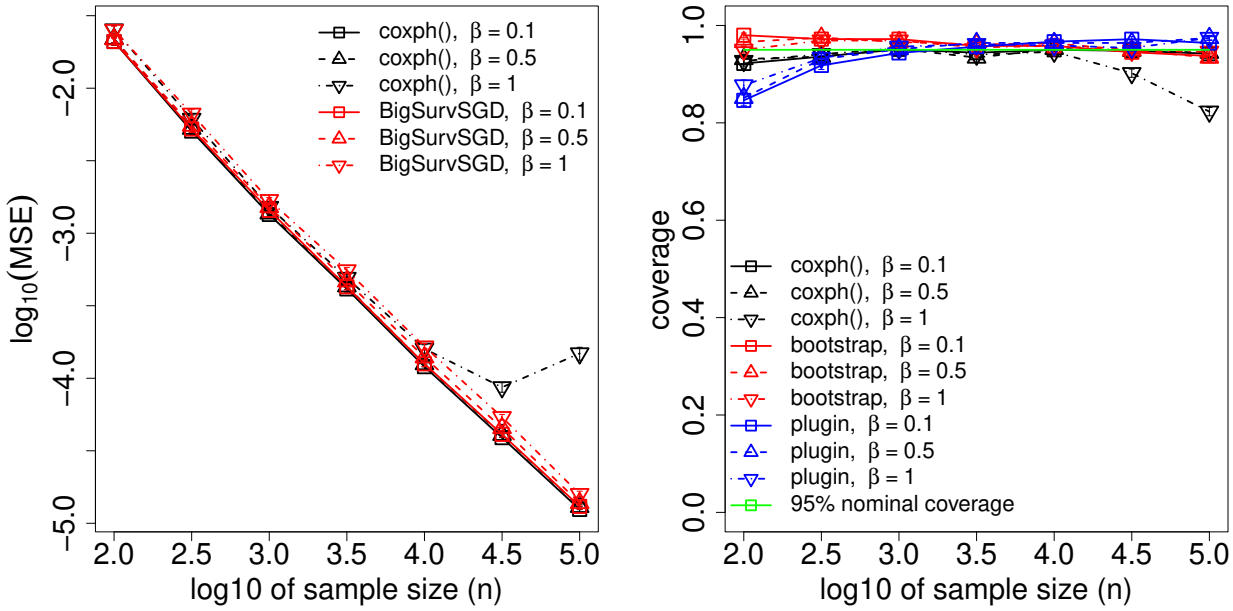

Figure A.7: Comparing $\log_{10}$(MSE) (left panel) and coverage (right panel) for varying sample size n and varying level of signal β. The total number of right-skewed features (with variance 1) is $P = 10$ but only $P_{sig} = 2$ among them have a signal level of β. We choose mini-batch size $K = 1$, number of epochs 100, $B = 100$ bootstrap samples for the bootstrap method, $n_o = 100$ samples per observation for the plugin method, probability of censoring $p_c = 0.2$, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

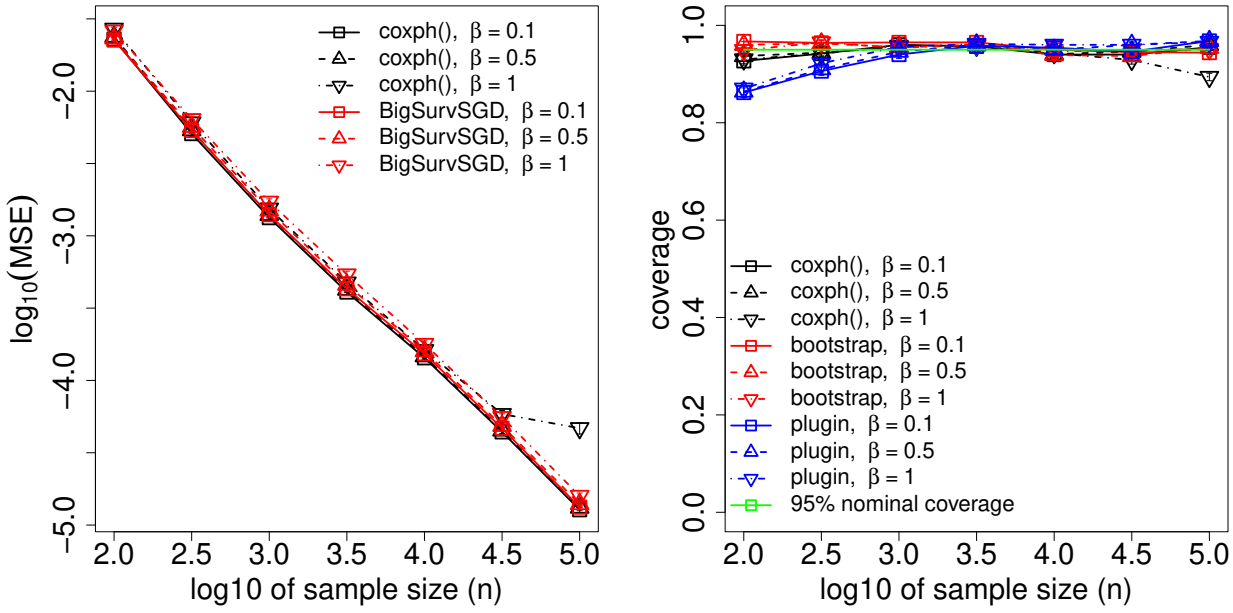

Figure A.8: Comparing $\log_{10}$(MSE) (left panel) and coverage (right panel) for varying sample size n and varying level of signal β. In total, there are five binary and five right-skewed features, generated marginally from multi-normal covariates with an exchangeable correlation structure with $\rho = 0.2$ (low level of correlation). But only one covariate per each type of covariate has a signal with the level of β. We choose mini-batch size $K = 1$, number of epochs 100, $B = 100$ bootstrap samples for the bootstrap method, $n_o = 100$ samples per observation for the plugin method, probability of censoring $p_c = 0.2$, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

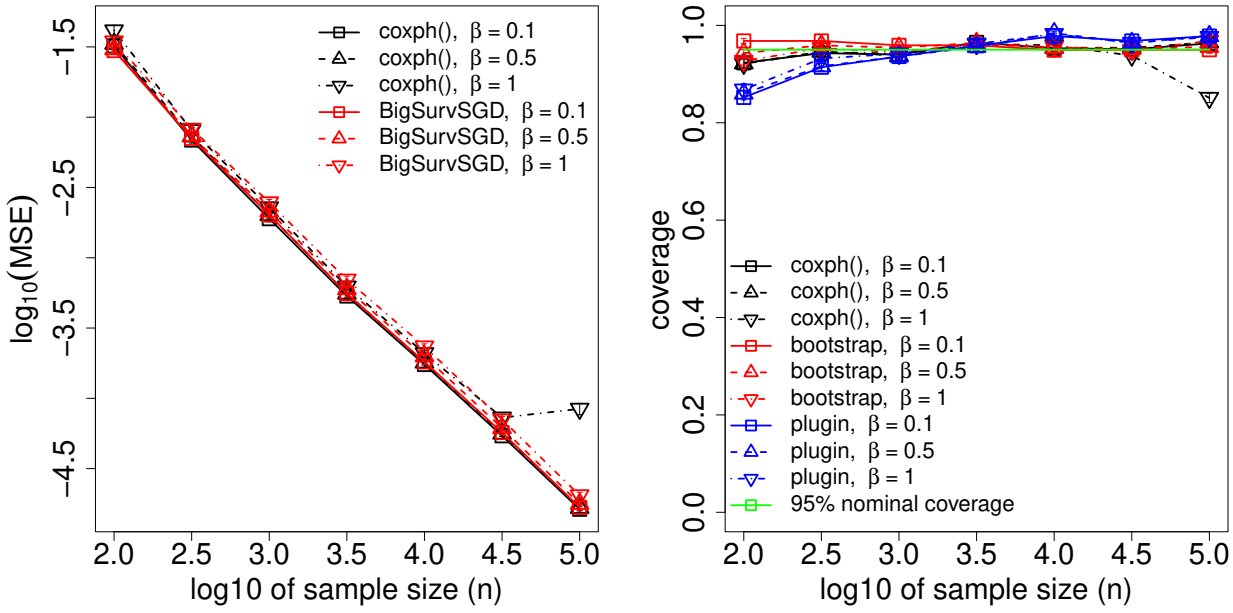

Figure A.9: Comparing $\log_{10}$(MSE) (left panel) and coverage (right panel) for varying sample size n and varying level of signal β. In total, there are five binary and five right-skewed features, generated marginally from multivariate normal covariates with an exchangeable correlation structure with $\rho = 0.5$ (medium level of correlation). But only one covariate per type has a signal with level β. We choose mini-batch size $K = 1$, number of epochs 100, $B = 10$ bootstrap samples for the bootstrap method, $n_o = 100$ samples per observation for the plugin method, probability of censoring $p_c = 0.2$, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

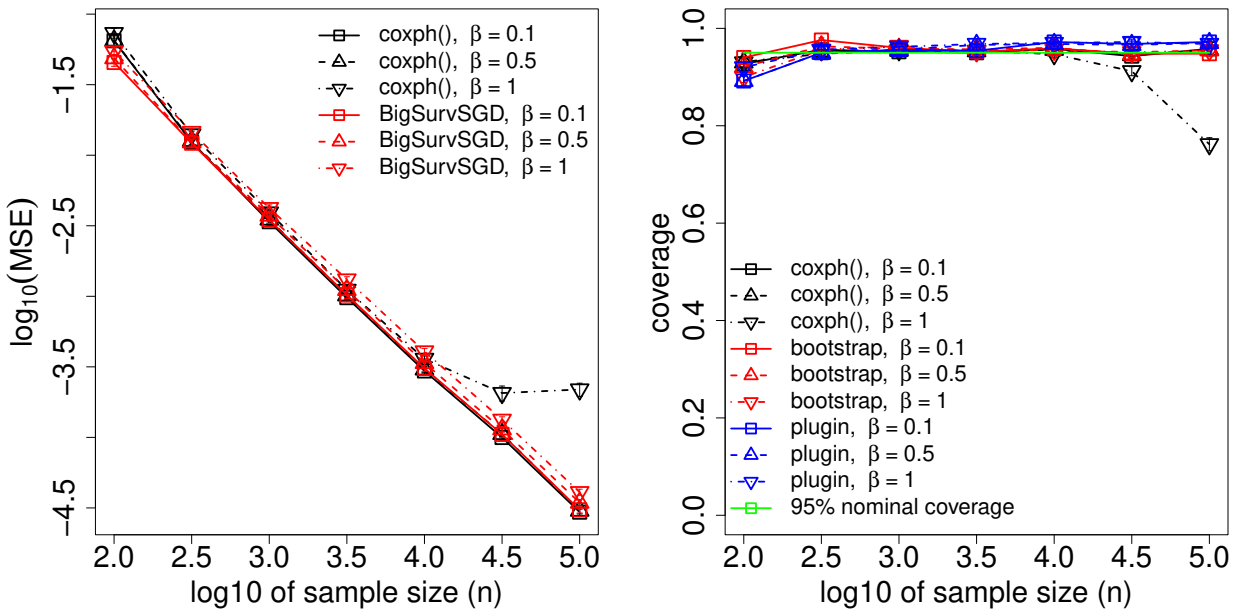

Figure A.10: Comparing $\log_{10}$(MSE) (left panel) and coverage (right panel) for varying sample size n and varying level of signal β. In total, there are five binary and five right-skewed features, generated marginally from multi-normal covariates with an exchangeable correlation structure with $\rho = 0.8$ (high level of correlation). But only one covariate per type has a signal with level β. We choose mini-batch size $K = 1$, number of epochs 100, $B = 100$ bootstrap samples for the bootstrap method, $n_o = 100$ samples per observation for the plugin method, probability of censoring $p_c = 0.2$, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

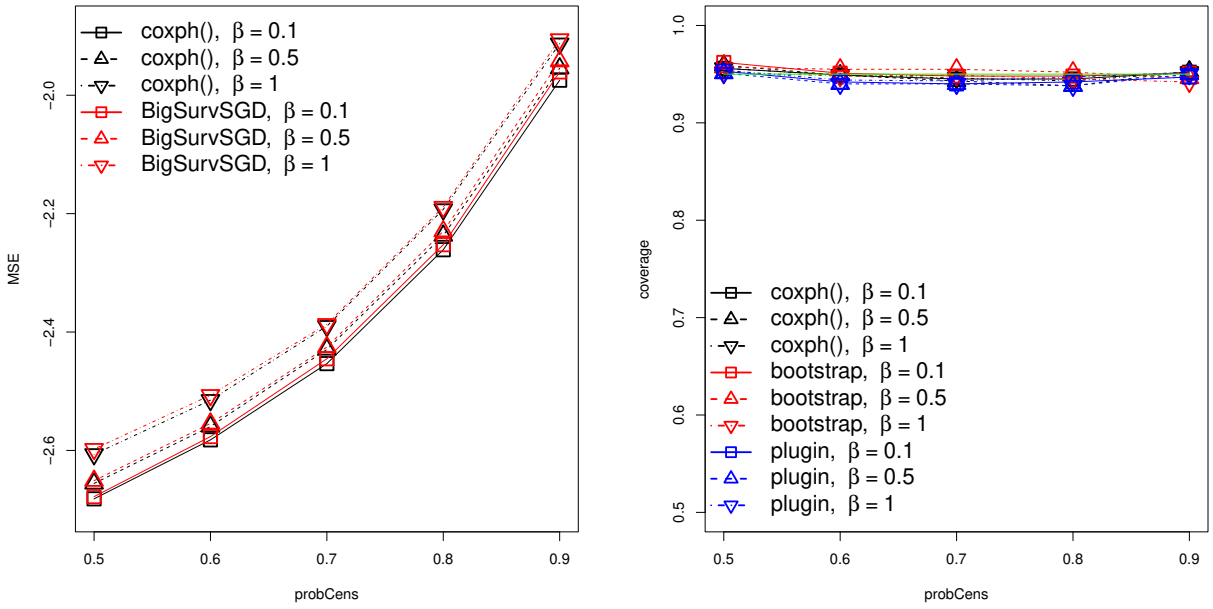

Figure A.11: Comparing $\log_{10}$(MSE) (left panel) and coverage (right panel) for varying levels of signal β. The total number of uniformly distributed features (with variance 1) is $P = 10$ but only $P_{sig} = 2$ among them have a signal level of β. We choose strata size $S = 50$, sample size $n = 1000$, mini-batch size $K = 1$, number of epochs 100, $B = 100$ bootstrap samples for the bootstrap method, $n_o = 100$ samples per observation for the plugin method, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

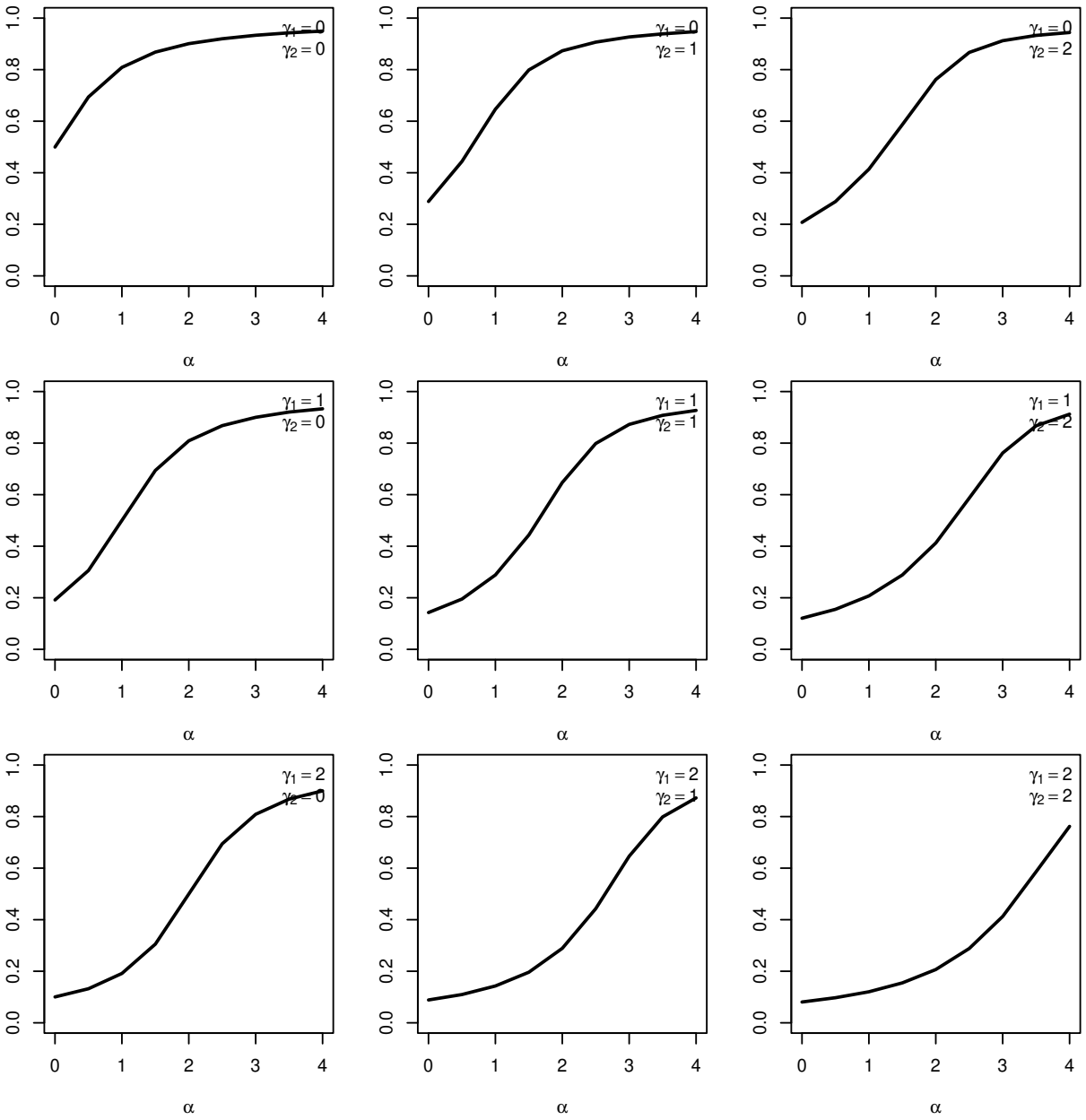

Figure A.12: Average censoring rate over 100 randomly generated survival datasets with sample size $n = 10^5$ for different values of γ_1, γ_2, and α. Event and censoring times are generated based on hazards models in (A.47) with sample size sample size $n = 10^5$ where censoring rate is calculated by $Pr(C < T)$.

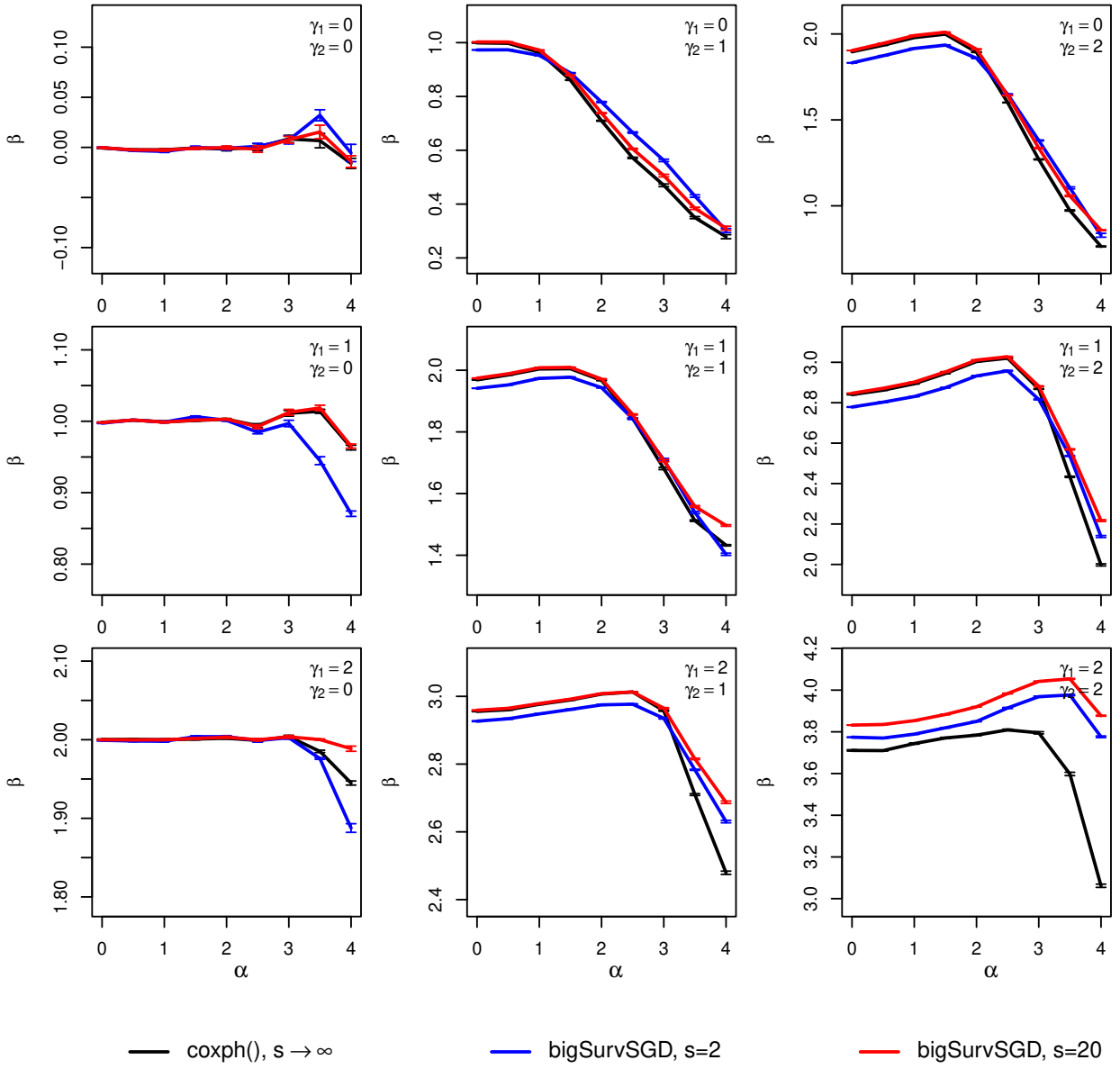

Figure A.13: Average estimate of $\beta^{(s)}$ (y-axis) over 100 randomly generated survival datasets with sample size $n = 10^5$ from `coxph()` and `bigSurvSGD` with strata sizes $s = 2$ and $s = 20$. We choose sample size $n = 10^5$. For our proposed framework, we choose mini-batch size $K = 1$, number of epochs 100, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

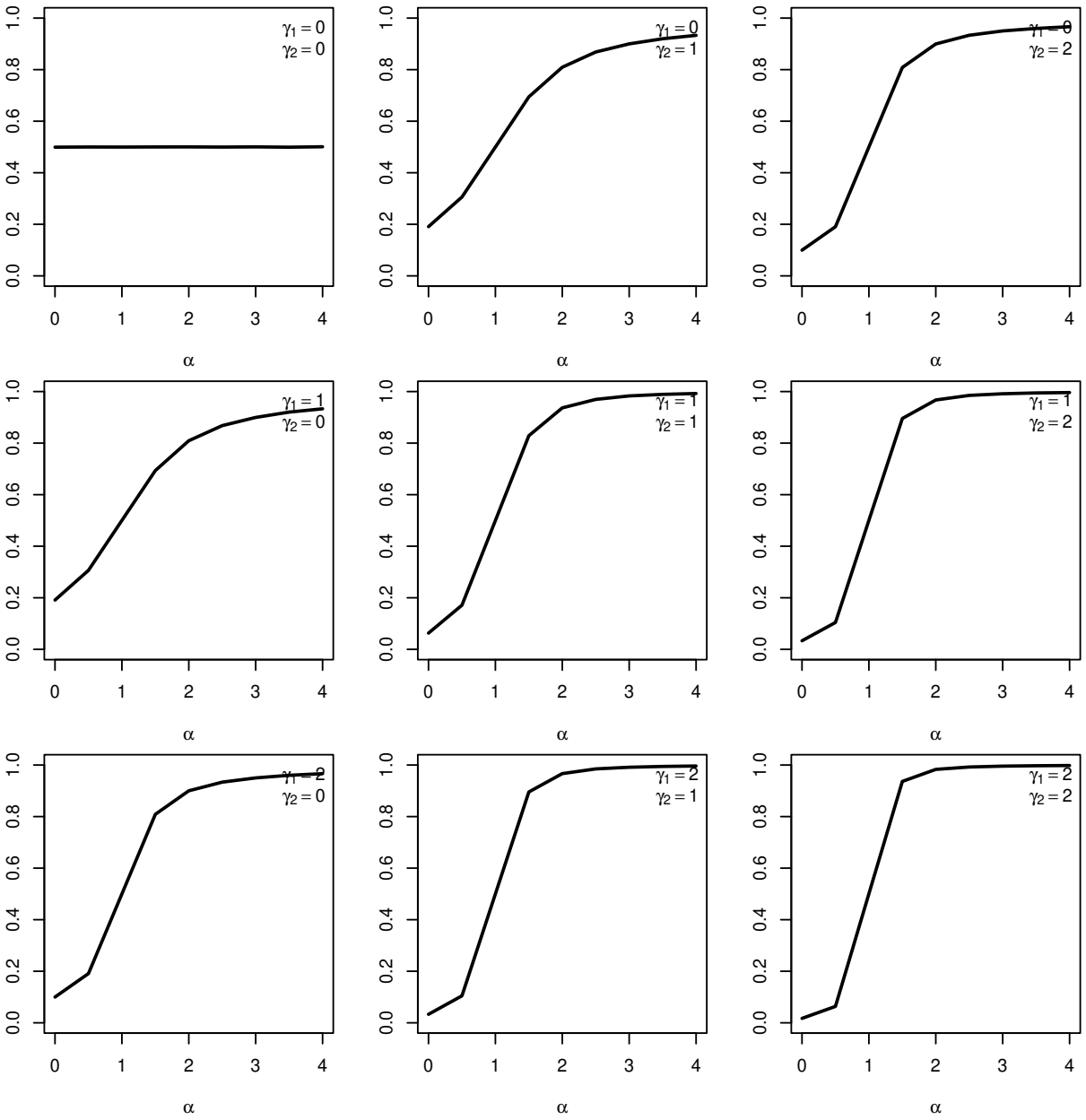

Figure A.14: Average censoring rate over 100 randomly generated survival datasets with sample size $n = 10^5$ for different values of γ_1, γ_2, and α. Event and censoring times are generated based on hazards models in (A.48) with sample size sample size $n = 10^5$ where censoring rate is calculated by $Pr(C < T)$.

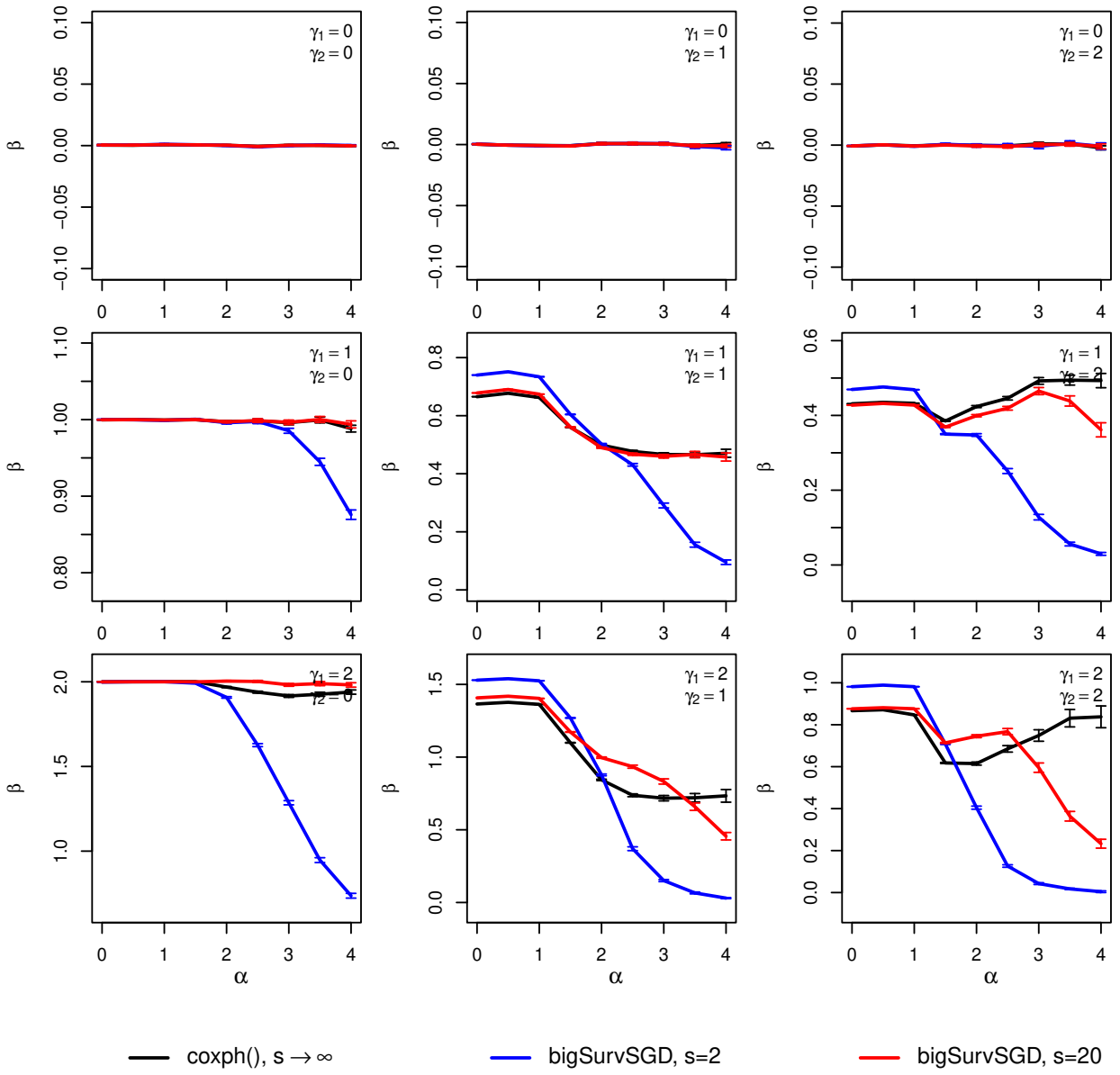

Figure A.15: Average estimate of $\beta^{(s)}$ (y-axis) over 100 randomly generated survival datasets with sample size $n = 10^5$ from `coxph()` and `bigSurvSGD` with strata sizes $s = 2$ and $s = 20$. We choose sample size $n = 10^5$. For our proposed framework, we choose mini-batch size $K = 1$, number of epochs 100, and $C = 0.12$ for the learning rate defined as $\gamma_m = \frac{C}{\sqrt{m}}$.

Appendix B

ADDITIONAL MATERIALS FOR CHAPTER 3

B.1 Derivation of the proposed updating procedure in Chapter 3

With proximal gradient descent (PGD) [141], the updating rule for our optimization problem in (3.3) is given by

$$\widehat{\boldsymbol{\beta}}_{m+1} = prox_{\gamma_m,R}\Big(\widehat{\boldsymbol{\beta}}_m - \gamma_m \nabla_{\boldsymbol{\beta}} F^{(s)}(\widehat{\boldsymbol{\beta}}_m)\Big), \tag{B.1}$$

where $F^{(s)}(\boldsymbol{\beta}) = \mathbb{E}_s\Big[f^{(s)}(\boldsymbol{\beta})\Big]$; γ_m is the learning rate (and should be specified in advance); and $prox_{\gamma,R}(x)$ is the proximity operator defined as

$$prox_{\gamma,R}(x) = \underset{y}{\operatorname{argmin}}\Big\{\frac{1}{2}||y-x||^2 + \gamma R(y)\Big\}. \tag{B.2}$$

Here, PGD applies the proximity operator (defined by the non-smooth function $R(\boldsymbol{\beta})$) to the iterations of gradient descent (calculated by the smooth function $F^{(s)}(\boldsymbol{\beta})$). By combining (B.1) and (B.2), we remain to solve

$$\widehat{\boldsymbol{\beta}}_{m+1} = \underset{y}{\operatorname{argmin}}\Big\{\frac{1}{2}||y - \Big(\widehat{\boldsymbol{\beta}}_m - \gamma_m \nabla_{\boldsymbol{\beta}} F^{(s)}(\widehat{\boldsymbol{\beta}}_m)\Big)||^2 + \gamma_m(1-\alpha)\lambda y + \gamma_m \alpha ||y||_1\Big\}, \tag{B.3}$$

which further simplifies to

$$(1+\gamma_m(1-\alpha)\lambda)\cdot\widehat{\boldsymbol{\beta}}_{m+1} = \widehat{\boldsymbol{\beta}}_m + \frac{\gamma_m}{s}\nabla_{\boldsymbol{\beta}}\mathbb{E}_s\Big[p\ell^{(s)}(\widehat{\boldsymbol{\beta}}_m|\mathcal{D}_m^{(s)})\Big] - \gamma_m\alpha\lambda * sign(\widehat{\boldsymbol{\beta}}_{m+1}), \tag{B.4}$$

where $sign(x)$ is defined as 1 if $x > 0$, -1 if $x < 0$, otherwise it is in $(-1, 1)$. After some simple calculations, the $m+1$ iterate of our PGD algorithm is given by:

$$\widehat{\boldsymbol{\beta}}_{m+1} = \frac{S\Big(\widehat{\boldsymbol{\beta}}_m + \frac{\gamma_m}{s}\nabla_{\boldsymbol{\beta}}\mathbb{E}_s\Big[p\ell^{(s)}(\widehat{\boldsymbol{\beta}}_m|\mathcal{D}_m^{(s)})\Big], \gamma_m\alpha\lambda\Big)}{1+\gamma_m(1-\alpha)\lambda} \tag{B.5}$$

where $S(z,a)$ is the soft thresholding operator defined as $S(z,a) = sign(z)(|z|-a)_+$.

$$S(z,a) = sign(z)(|z|-a)_+ = \begin{cases} z-a & \text{if } z > a \\ 0 & \text{if } -a \le z_j \le a \\ z+a & \text{if } z_i < -a, \end{cases} \tag{B.6}$$

Noting the fact that for any $\boldsymbol{\beta}$ we have

$$\nabla_{\boldsymbol{\beta}} \mathbb{E}_s \left[pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}_m^{(s)}) \right] = \mathbb{E}_s \left[\nabla_{\boldsymbol{\beta}} \left\{ pl^{(s)}\boldsymbol{\beta}|\mathcal{D}_m^{(s)}) \right\} \right], \tag{B.7}$$

provided that $\boldsymbol{x}$ are drawn from a reasonable distribution (e.g., bounded). Therefore, by using the alternative U-statistic-based loss in (3.4), we can get the updating rule given by (3.5).

B.2 A regularization path for λ

Given the objective function, we are usually interested in evaluating the model in (3.2) and its U-statistics-based version in (3.4) for different values of α and λ. We consider the pathwise procedure proposed and studied by [74, 5] for tuning λ. To do this, we fix α and evaluate solutions for a path of λ values starting from λ_{max} and ending with λ_{min}. Choosing λ_{min} depends on the structure of data but a rule of thumb is to choose $\lambda_{min} = 0.01$ for $p < n$ and $\lambda_{min} = 0.0001$ for $p \ge n$ [5, 75]. Note that $\lambda_{min} = 0$ returns the non-regularized solution that might behave poorly. We next explain how to calculate λ_{max}. The solution of (3.2) is in the form of $\nabla_{\boldsymbol{\beta}} \mathbb{E}_s \Big[\ell_s(\boldsymbol{\beta}) \Big] = \mathbb{E}_s \Big[\nabla_{\boldsymbol{\beta}} \ell_s(\boldsymbol{\beta}) \Big] = 0$ which is equivalent to

$$\frac{1}{s} \mathbb{E}_s \Big[\nabla_{\boldsymbol{\beta}} p\ell^{(s)}(\boldsymbol{\beta}) \Big] = \lambda\alpha * sign(\boldsymbol{\beta}) + \lambda(1-\alpha) * \boldsymbol{\beta}. \tag{B.8}$$

Given (B.8), a necessary condition to have $\boldsymbol{\beta} = \mathbf{0}$ (i.e., $\beta_j = 0$ for $j = 1, 2, \ldots, p$) is:

$$\Big| sign(\beta_j) \Big| = \frac{\Big[\Big| \mathbb{E}_s [\nabla_{\boldsymbol{\beta}} p\ell^{(s)}(\boldsymbol{\beta})]_{\boldsymbol{\beta}=\mathbf{0}} \Big| \Big]_j}{s\lambda\alpha} < 1 \quad \text{for} \quad j = 1, 2, \ldots, p, \tag{B.9}$$

or

$$\lambda > \frac{||\mathbb{E}_s [\nabla_{\boldsymbol{\beta}} p\ell^{(s)}(\boldsymbol{\beta})]|_{\boldsymbol{\beta}=\mathbf{0}}||_\infty}{s\alpha} \tag{B.10}$$

Remember that, by definition, $sign(x) \in (-1, 1)$ for $x = 0$. The smallest value of λ giving us all-zero solutions for $\boldsymbol{\beta}$ is $||\nabla_{\boldsymbol{\beta}} p\ell^{(s)}(\boldsymbol{\beta})_{\boldsymbol{\beta}=\mathbf{0}}||_\infty/\alpha$ which can be chosen as the maximum value on the regularization path for λ, i.e., λ_{max}. In practice, we can estimate λ_{max} for our re-framed penalized Cox model by taking an average over n_s ($n_s \ll \binom{n}{s}$) randomly selected strata as

$$\widehat{\lambda}_{max} = \frac{\left|\left|\frac{1}{n_s}\sum_{n_s\ strata}\{\nabla pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})\}\Big|_{\boldsymbol{\beta}=\overrightarrow{0}}\right|\right|_\infty}{s\alpha}. \tag{B.11}$$

After calculating λ_{max}, we can select a λ-sequence on the log scale starting from λ_{max} and ending at a pre-specified λ_{min}. We may consider different sequences of λ for different values of α to find the best pair of (α, λ)) based on the pre-specified selection criterion (e.g., concordance index).

For a specific random strata with data $\mathcal{D}^{(s)} = \{\mathcal{D}_i = (y_i, \delta_i, \boldsymbol{x}^{(i)})|i = 1, 2, \ldots, s\}$, we can simplify $\nabla_\beta\Big\{pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})\Big\}\Big|_{\boldsymbol{\beta}=\overrightarrow{0}}$ using matrix multiplications as

$$\begin{aligned}
\nabla_\beta\Big\{pl^{(s)}(\boldsymbol{\beta}|\mathcal{D}^{(s)})\Big\}\Big|_{\boldsymbol{\beta}=\overrightarrow{0}} &= \sum_{i=1}^{s}\Big(\boldsymbol{x}_i^T - \frac{\sum_{j\in\mathcal{R}_i}\boldsymbol{x}_j^T e^{\boldsymbol{x}_j^T\boldsymbol{\beta}})}{\sum_{j\in\mathcal{R}_i} e^{\boldsymbol{x}_j^T\boldsymbol{\beta}}}\Big)\Big|_{\boldsymbol{\beta}=\overrightarrow{0}} \\
&= \begin{bmatrix}\delta_1\\ \delta_2\\ \delta_3\\ \vdots\\ \delta_{s-1}\end{bmatrix}^T \begin{bmatrix} 1-\frac{1}{s} & \frac{-1}{s} & \frac{-1}{s} & \ldots & \frac{-1}{s} & \frac{-1}{s}\\ 0 & 1-\frac{1}{s-1} & \frac{-1}{s-1} & \ldots, & \frac{-1}{s-1} & \frac{-1}{s-1}\\ 0 & 0 & 1-\frac{1}{s-2} & \ldots & \frac{-1}{s-2} & \frac{-1}{s-2}\\ \vdots & \vdots & \vdots & \ldots & \vdots & \vdots\\ 0 & 0 & 0 & \ldots & \frac{1}{2} & \frac{-1}{2}\end{bmatrix} \begin{bmatrix}\boldsymbol{x}^{(1)T}\\ \boldsymbol{x}^{(2)T}\\ \boldsymbol{x}^{(3)T}\\ \vdots\\ \boldsymbol{x}^{(s-1)T}\end{bmatrix}.
\end{aligned} \tag{B.12}$$

VITA

Aliasghar (Arash) Tarkhan was born to a family of rice farmers in Rostam Rud, a village in Natel Kenar-e Sofla Rural District in the Central District of Nur County, Mazandaran Province, Iran. He received two bachelor's degrees in Electrical Engineering with a focus on Telecommunication Systems and Power Systems from the Amirkabir University of Technology in 2008. He received a master's degree in Electrical Engineering with a focus on Microwave and Optical Communications in 2011. He also received a master's degree in Electrical Engineering with a focus on Signal Processing from the University of Washington in 2016.