2019-07-01

Sequential Survival Analysis with Deep Learning

Seth William Glazier
Brigham Young University

Sequential Survival Analysis with Deep Learning

Seth William Glazier

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Jeffrey Humpherys, Chair
Tyler Jarvis
Jared Whitehead

Department of Mathematics

Brigham Young University

ABSTRACT

Sequential Survival Analysis with Deep Learning

Seth William Glazier
Department of Mathematics, BYU
Master of Science

Survival Analysis is the collection of statistical techniques used to model the time of occurrence, i.e. survival time, of an event of interest such as death, marriage, the lifespan of a consumer product or the onset of a disease. Traditional survival analysis methods rely on assumptions that make it difficult, if not impossible to learn complex non-linear relationships between the covariates and survival time that is inherent in many real world applications. We first demonstrate that a recurrent neural network (RNN) is better suited to model problems with non-linear dependencies in synthetic time-dependent and non-time-dependent experiments.

Keywords: survival analysis, deep learning

Acknowledgments

I would like to thank Dr. Jeffrey Humpherys for his belief in me; the opportunities and encouragement he's continually provided me has laid the groundwork for a lifetime of fulfilling research. I thank my wife Elise, whose temporal and emotional support has made my desire to pursue a graduate education possible. I thank the BYU Department of Mathematics for providing the knowledge and resources needed for my intellectual edification.

Contents

List of Tables

List of Figures

Chapter 1. Introduction

Survival analysis is a branch of statistics concerned with the analysis of the duration of time until the occurrence of an event of interest. This duration, denoted by the random variable T, is defined as the amount of time elapsed between some *time of origin*, when a subject is first observed, and an *end point*, which marks either the occurrence of the event of interest, or the end of the subject's participation in the study, referred to as censoring. It is common to refer to the occurrence of the event of interest as *death*, the time until death as *survival time*, if death has not occurred yet, a subject is said to be "alive", etc. We adopt this terminology and repeatedly use it throughout this text. The goal of survival analysis is to provide answers such as: What portion of a population will be alive after time t? What features are most predictive of death? What is the expected survival time of an individual?

Though survival analysis is prevalent in many fields, such as economics (time looking for employment), social science (time from adulthood to marriage), finance (time to default on a loan), and reliability engineering (time to failure of an electronic component), it's especially prevalent in medicine. In medical applications, survival analysis techniques can be used to determine the effectiveness of a new type of treatment for a disease or medical condition, such as breast cancer [2], the relationship between tumor size and post-surgery survival of lung cancer patients [3], identifying the factors most indicative of a successful ventilator weaning in chronic obstructive pulmonary disease (COPD) [4], etc. The term "5-year survival rate" is used often when giving diagnosis information to a cancer patient, or any other potentially terminal diagnosis. These survival rates, or survival probabilities are calculated using methods from survival analysis. The accuracy of these predictions is essential for giving certainty to individuals looking to maximize the quality of their remaining life and engage in end-of-life planning. Accurately predicting the onset of a preventable disease such as diabetes, COPD, or chronic kidney disease (CKD) can help medical providers identify high-risk patients with whom preventative measures can be taken. Having an accurate assessment

of their risk of disease onset can also motivate patients to comply with advice from their medical provider. The results of survival analysis studies guides the deployment of new treatments and surgical techniques, identifies patient features that are most predictive of medical outcomes, and informs both patients and doctors of the risk of disease onset or the development of a medical condition. An improvement in the quality of survival analysis model predictions can help doctors, as they strive to provide the best possible care and guidance.

1.1 Motivating Example: Prostate Cancer

To illustrate the potential use of survival analysis in the field of health care, lets consider the example of prostate cancer. Prostate cancer is one of the most common types of cancer among men. According to the American Cancer Society, [5] in 2019 there are expected to be 174,650 new diagnosis of prostate cancer among men living in the United States. Older people are disproportionately affected, with most cases occurring in men older than 40, and the average age of diagnosis being 66. About 90% of all prostate cancer cases are diagnosed when the cancer is still confined to the prostate and nearby tissue. If the cancer is contained within the local region of the prostate, five year survival rates are nearly 100%, however, if the cancer spreads to other parts of the body, five year survival rates drop to around 30%. Given that local prostate cancer is relatively harmless, the goal of prostate cancer treatments is to prevent the spread of cancerous cells from the prostate region to other parts of the body. Available treatment options can have undesirable side effects that can seriously impact a patient's quality of life, even if successful. There are four main treatment options for men diagnosed with prostate cancer:

- **Active surveillance**: Most treatments for prostate cancer can have side affects such as erectile dysfunction, which is the inability to hold an erection and incontinence, which is the loss of urine flow control and/or bowel function. These side effects significantly impact the quality of a man's life, enough so that in cases where the prostate cancer

is expected to grow very slowly, the best course of action may be to delay cancer treatment. This option is known as active surveillance. During active surveillance, the cancer is closely monitored, and if/when the cancer is found to be worsening, other treatments are considered. [6]

- **Prostatectomy**: A prostatectomy is the surgical removal of the prostate and can eliminate prostate cancer from the source, hopefully removing the tumor from the body before it becomes too large or metastatic, which occurs when cancer cells from the original tumor begin to spread throughout the body. There are several methods for removing the prostate, some more invasive than others, but all of them have a high risk of affecting sexual function and urinary incontinence. [7]

- **Radiation therapy**: The use of radiation to destroy cancer cells. There are several forms of radiation therapy, but the most common type consists of several scheduled sessions in which the area of the cancer is exposed to a focused beam of x-rays. Though the x-rays are focused on the prostate cancer, there is potential for damage in the surrounding areas, which can cause gastrointestinal side effects such as diarrhea, rectal discomfort or rectal bleeding, in addition to potentially affecting sexual function. [8]

- **Androgen deprivation therapy**: Prostate cancer growth is driven by male sex hormones called androgens. Lowering androgen levels in the body can slow the growth of the cancer. The most prevalent androgen is testosterone, so androgen deprivation therapy (ADT) typically consists of lowering testosterone levels in the body, which can be done via surgical removal of the testicles, known as surgical castration, or via drugs designed to reduce testicular function, known as chemical castration. Side effects include erectile dysfunction, loss of sexual desire, depression, cognitive dysfunction, weight gain, loss of muscle mass, and osteoporosis. Additionally, ADT has been linked to developing metabolic syndrome, which is a set of conditions such as high cholesterol,

obesity, and high blood pressure that increase the risk of heart disease, strokes, and diabetes. [9]

All of these treatments have success rates and drawbacks that are affected by a variety of factors including the patient's age, medical history, physical condition, and the stage of their prostate cancer. The choice of treatment affects not only a patient's probability of survival, but also their quality of life. A patient's decision of what treatment to engage in is very personal, and the factors considered include the overall effectiveness and duration of the different treatment options, and the type of risks the patient is willing to take. It's imperative that a patient have access to the most accurate information to guide them in making one of the most important decisions of their lives. One of the goals of a survival analysis model is to provide individualized projections of the survival probabilities for each treatment, providing both the doctor and the patient with the information they need to make an optimal decision.

1.2 Standard Survival Analysis Methods

When performing a survival analysis, two functions are of fundamental interest: the *survival function* $S(t)$ and the *hazard function* $\lambda(t)$. The survival function gives the probability that the subject will be alive at time t and thus their survival time T is greater than t. The hazard function is the probability that a subject will die during a small interval of time, given that at the beginning of the time interval the subject is still alive. Estimating these functions is among the primary goals of survival analysis and the two most common methods for doing so are the Kaplan-Meier [10] and the Cox Proportional Hazards [11] models. Kaplan-Meier is a non-parametric model that uses the gathered survival data to generate an estimate of the survival function $S(t)$ known as a Kaplan-Meier (KM) curve. KM curves are ubiquitous in survival analysis and are used to create an initial, assumption-free baseline to which other models are compared.

Cox proportional hazards (Cox) models are a form of regression analysis that model the relationship between the covariates $\mathbf{x}$ of a dataset and the hazard function $\lambda(t)$. Cox proportional hazards assumes a hazard function of the form $\lambda(t|\mathbf{x_i}) = \lambda_0(t)\exp(\boldsymbol{\beta}\mathbf{x_i})$, where $\mathbf{x}_i$ denotes the covariates of subject i and $\boldsymbol{\beta}$ is a vector of learned weights. The Cox model is semi-parametric in that it makes no assumptions on the form of $\lambda_0(t)$ (the baseline hazard function), but does make the assumption that each of the covariates have a multiplicative effect on the hazard function (hence the term *proportional hazards*). Like the KM curve, Cox models are ubiquitous in survival analysis, especially in medicine, where the lack of assumption on the baseline hazard function, and the computed feature weights $\boldsymbol{\beta}$ are both desirable properties. The learned weights $\boldsymbol{\beta}$ are used to determine which covariates are most predictive of survival (if β_i is larger than β_j, then according to the model, feature x_i has a greater impact on survival than x_j). To illustrate how common Cox proportional hazard models are, note that according to Google Scholar, the paper in which the Cox proportional hazards mode was first put forth is one of the 100 most cited scientific papers [12], despite the fact that survival analysis is a relatively small sub-field of statistics. In fact, it's not uncommon for textbooks or articles on medical applications on survival analysis and medicine to exclusively teach KM and Cox models [13].

1.3 Purpose of Research

It is sometimes the case that a survival analysis problem can be re-framed as a sequence modeling problem. Gathering time-to-event data usually requires observing patients at certain times (intervals between observations are not necessarily regular) and recording their health state and other relevant variables. These types of data gathering processes provide a sequence of observed variables ending in the observation of the event of interest. To use sequential data in a traditional survival analysis modeling technique such as Cox proportional hazards, the sequence is reduced to a vector representation that reduces the temporal

dimension of the sequence, failing to capture temporal relationships in the data that may be useful for predicting survival outcomes.

One example of time-to-event data that also contains sequential information is electronic health record (EHR) data. When a patient visits a health care provider, the results of that visit are recorded as a set of international classification of diseases version 10 (ICD-10) medical codes [14]. Codes exist for diagnoses, prescriptions, surgical procedures, etc. Each visit is represented as a set of codes for all diagnosis/procedures performed during the visit. The sequence of all these visits creates a record of a person's medical history. Thus, EHR of an individual consists of two types of data: static data, consisting of demographic variables that for most analysis is considered to be fixed, and sequential data, consisting of the sequence of medical codes that describe a person's medical history. Standard survival analysis models such as Cox proportional hazards make predictions from feature vectors, making them well equipped to operate on a vector of a patient's static features. Unfortunately, because Cox models operate on feature vectors, they are not a natural fit for sequential data.

In order to include sequential data in a Cox model, one must find a vector representation of the sequence. One popular way to do this is the *bag-of-words* method, where a sequence is represented as a count vector. In the EHR case, there are approximately 87,000 ICD-10 codes, so each possible code would be uniquely mapped to a number between 1 and 87,000. The count vector for the medical history of an individual would consist of the number of times each code occurs in their EHR, so if code 50 corresponds w/ a strep throat, and a person has had strep throat 3 different times, the value of the count vector at index 50 would be 3. Unfortunately, this representation loses all temporal information in the sequence, which could be very useful for survival analysis problems such as the onset of a disease. For example, if a woman has had 6 bouts of symptoms resembling a urinary tract infection (UTI) at semi-regular intervals over the last 20 years, they should not be considered at risk for interstitial cystitis (IC) [15], an autoimmune disease that causes chronic bladder pain that is often mis-diagnosed as a series of UTIs, however, if a patient has had 6 bouts of

UTI-like symptoms in the last 6 months, they should be considered very high-risk for IC or some other urinary tract condition. A bag-of-words representation of a sequence of ICD codes would not be able to distinguish between the two cases.

Given that

- Large portions of EHR data is sequential,
- There are lots of survival analysis applications involving EHR, and
- Current standard survival analysis modeling techniques are not designed to operate on sequential data,

we experiment with using a recurrent neural network (RNN) as the base machine learning algorithm to perform survival analysis on data that has both static and sequential features. RNNs are designed to work on both vector and sequential data and have performed well on sequential natural language processing tasks such as machine translation [16], text classification [17], sequence labeling [18], and text generation [19].

We give a more in-depth explanation of current survival analysis techniques in Chapter 2, followed by an RNN-centric explanation on sequence modeling in Chapter 3, followed by a detailed description of our experiments, a breakdown of our results, and some proposals for future work.

Chapter 2. Survival Analysis

Survival analysis is a specialized branch of statistics concerned with modeling and predicting the occurrence of one or more events of interest, such as death, birth, marriage, the onset of a disease, or even the mechanical failure of a consumer product. This type of data is known as time-to-event data, where each data-point, be it a person, patient, or mechanical device, consists of a survival time T, denoting the time between first observation, and the occurrence of the event E. This requires that occurrence of the event be well-defined and

observable, and for a time of origin, such as birth, the beginning of a medical study, the date of manufacturing, etc. to be specified for each subject. Historically the event of interest in survival analysis has been biological death, and it's common to refer to the event of interest as *death*, which we will frequently do throughout the rest of this paper.

In practical applications it's common that an event may never be observed. In the case of a medical study where the event of interest is usually the diagnosis of a medical condition, there will be some patients who leave the study before its conclusion and others who stay but aren't diagnosed before the conclusion of the study. In both cases it's possible that the condition may develop for these people, but it will not be observed in the data. This lack of observation is known as *censoring* and is a characteristic unique to time-to-event data that survival analysis methods must account for, and is part of what distinguishes a survival analysis problem from other statistical problems.

The goal of survival analysis is to provide answers to the questions that arise from time-to-event data, such as: What is the probability a subject will survive past a specified time? At what rate will the studied population die? What effect do certain characteristics of a subject have on its survival probability?

2.1 Censoring

Censoring occurs when the death of a subject is unobservable. There are many different types of censoring, with the most common being *right censoring*. A subject is said to be right censored when their last time of observation, t_c, passes and the event of interest has not yet occurred. This could happen, for example, if a patient drops out of a study early, and death is never observed. Most studies have some kind of last observation time, at which point the study is concluded. It's very common for the study to come to an end and for there to remain many individuals who have not yet "died".

Two similar types of censoring are $left\ censoring$ and *interval censoring*. Left censoring is similar to right censoring except that rather than death occurring after a study, it is known

to occur at some time before a study, but that time is unspecified. Interval censoring is when an event is known to happen during an interval of time, but again the exact time is unknown. This is very common in medical studies, where a patient will be diagnosed with some disease. It is known that the onset of the disease happened at some point between the time of diagnosis and the time of the patient's previous appointment, but the exact time of onset is impossible to know.

For most survival analysis methods it is assumed that censoring is *non-informative*, meaning that the censoring mechanism is independent of the process that drives event occurrence. The censoring of an individual should not be related in any way to the event of interest. For example, if patients in a medical study who recover from an illness and become healthy are more likely to leave the study (a very plausible scenario), the censoring is informative. Another unfortunately plausible scenario occurs when studying insurance data. In insurance data it's common to identify someone as "censored" when they leave the insurance provider, however, it's very possible that an individual whose health worsens may no longer be well enough to go to work, and thus become unemployed. If their health insurance is provided through their employer, their unemployment would coincide with a change of insurance provider, in which case the censoring is definitely informative. For a given study, it's often impossible to tell if censoring is informative or not.

Time-to-event must contain at least two columns: the event column and the time column, often (but not always) respectively denoted E and T. The event column is a binary indication of whether or not the event was observed, with 1 being an observation and 0 meaning that the event was not observed. The time column represents the amount of time between initial observation and either death or censorship, depending on the corresponding value of E. The vast majority of the time, a time-to-event dataset will contain other characteristics of interest about each subject, such as the age and sex of patients in a medical study, but the time and event column are required in order to apply any survival analysis techniques. As an example

	week	arrest	fin	age	race	wexp	mar	paro	prio
0	20	1	0	27	1	0	0	1	3
1	17	1	0	18	1	0	0	1	8
2	25	1	0	19	0	1	0	1	13
3	52	0	1	23	1	1	1	1	1
4	52	0	0	19	0	1	0	1	3
5	52	0	0	24	1	1	0	0	2
6	23	1	0	25	1	1	1	1	0
7	52	0	1	21	1	1	0	1	4
8	52	0	0	22	1	0	0	0	6
9	52	0	0	20	1	1	0	0	0

Table 2.1: In this example dataset, the week column is the time (often denoted T) and the arrest column is the event indicator (often denoted E). The rest of the columns represent various characteristics, such as age, race, and sex, that could provide useful insights on the relationship between these covariates and survival.

See Table 4.1 for a snippet of the Rossi [1] recidivism dataset provided by the Python survival analysis package *lifelines*. [20]

2.2 Definitions

2.2.1 Survival Function. The primary object of interest in survival analysis is the survival function $S(t)$, defined

$$S(t) = \Pr(T > t) \tag{2.1}$$

where t is an arbitrary time and T is a random variable representing the time of death. Thus for a given time t, the survival function returns the probability of surviving past time t, or equivalently that death will occur after time t. Usually it is assumed that $S(0) = 1$ though that is not always the case. Another assumption is that $S(t)$ is non-increasing, meaning $S(a) \leq S(t)\ \forall\ a \leq t$. This makes intuitive sense, your probability of going 10 months without a heart attack is going to be less than or equal to surviving 9 months, because the occurrence of the former necessitates the occurrence of the latter.

In most cases, such as that of physical death, it is assumed that $S(t) \to 0$ as $t \to \infty$, since everyone eventually dies, but there are examples where this is not the case. For example, if our event is marriage, there will be some portion of the population that never gets married.

2.2.2 Lifetime Distribution and Density Functions. Where the survival function $S(t)$ gives the probability of surviving past time t, the lifetime distribution function is the compliment of the survival function, is defined

$$F(t) = 1 - S(t) \tag{2.2}$$

and gives the probability that death has already occurred for a given t. Assuming that F is differentiable, its derivative f is called the density function of the lifetime distribution,

$$f(t) = F'(t) \tag{2.3}$$

or is simply referred to as the event density, and is the rate of death per unit of time. Thus the survival, lifetime distribution, and event density functions can all be derived from each other

$$S(t) = \Pr(T > t) \ = \ \int_t^\infty f(u)du = 1 - F(t). \tag{2.4}$$

2.2.3 Hazard Function. The hazard function λ is defined as the rate of events at time t conditional on $T > t$ (i.e. that death occurs after t). Given that a subject has survived up to time t, $\lambda(t)$ gives us the probability that death will occur in an additional instant of time Δt:

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{\Pr(t \leq T < t + \Delta t)}{\Delta t \cdot S(t)} = \frac{f(t)}{S(t)} = -\frac{S'(t)}{S(t)} \tag{2.5}$$

Note that $f(t)$ is the event density function, which outputs the rate of events per unit of time, and that $S(t)$ gives the probability of survival up to time t, so the hazard function formulation $\lambda(t) = \frac{f(t)}{S(t)}$ is the rate of events scaled by the proportion of the population

expected to be alive and can thus be interpreted as giving the instantaneous rate of death *for individuals* who have survived up to time t, and is the probability density function of the distribution of death.

The hazard function can provide valuable insights on the underlying causes of mortality, often more so than the other functions used to describe the lifetime of a survival population. For example, a hazard function may have a bathtub curve, which is really high early on, then decreases to some minimum, before rising sharply again at some later time. This is the case with many populations, such as mechanical systems, where a lot fail early on, because of some issue with its manufacturing, after which the remaining units tend to last a long time, until beginning to break down because of aging. Human mortality follows a similar pattern where the risk of death is much higher for elderly people and young children than for the middle aged.

The cumulative hazard function is defined:

$$\Lambda(t) = \int_0^t \lambda(t)dt \tag{2.6}$$

and is the sum of all risks on the time interval 0 to t. Using these equations we can make a more direct connection between the hazard and survival functions:

$$\lambda(t) = \frac{f(t)}{S(t)} \tag{2.7}$$

$$\lambda(t) = \frac{f(t)}{(1 - F(t))} \tag{2.8}$$

$$\lambda(t) = -\frac{\partial}{\partial t} \ln\left(1 - F(t)\right) \tag{2.9}$$

integrating both sides,

$$\Lambda(t) = -\ln(1 - F(t)) \tag{2.10}$$

$$\Lambda(t) = -\ln(S(t)) \tag{2.11}$$

$$-\Lambda(t) = \ln S(t) \tag{2.12}$$

taking the exponent, we end up with

$$S(t) = \exp(-\Lambda(t)) \tag{2.13}$$

These equations show how the hazard and survival functions are just different ways of representing the same distribution of survival time T. From one of them, all of the others can be derived. Given the survival function $S(t)$, we can take the derivative of the compliment $F(t)$ to get $f(t)$ and from there we can derive the hazard function via $\frac{f(t)}{S(t)}$. As we will see, all of these characterizations of T provide a distinct perspective on survival that can make an analysis more interpretable and useful.

2.3 Survival Models

Nearly all survival populations are non-homogeneous, which is to say that there exists covariates or features for each subject that may have an effect on survival time. Due to the presence of these covariates, it would be unwise to model the entire population using a single survival function $S(t)$, and often it is the goal of survival analysis to derive and compare survival functions of different sub-populations in the data to detect and measure the effect the features of a subject have on its survival time. There are several ways to do this and we will review a few of the most prominent techniques.

2.3.1 Kaplan-Meier Curves. The Kaplan-Meier (KM) curve is a commonly used non-parametric method for estimating the survival function of populations in a time-to-event

dataset. Non-parametric methods are straightforward and make no assumptions on the distribution of $S(t)$, but instead draw their estimates directly from the data. Non-parametric methods are easy to use, and useful to get an initial read on the form of the dataset and make basic comparisons of covariate effects, but may struggle with more complex scenarios.

Let $n = n_0$ be the sample size, $t_1 < t_2 < \cdots t_k$ be the observed event times, d_i be the number of subjects who had a non-censored event occur at time t_i, where $i \in 1, 2, \cdots k$, and c_i is the number of subjects who were censored between t_{i-1} and t_i. Then we have $n_i = (d_i + c_i) + (d_{i+1} + c_{i+1}) + \cdots + (d_k + c_k)$, were n_i is the number of non-censored individuals who are still alive at time t_i-1.

The Kaplan-Meier estimation of the survival function is defined:

$$\hat{S}(t) = \prod_{i:t_i \leq t} \frac{n_i - d_i}{n_i}, \tag{2.14}$$

Looking at the equation above, the quantity $\frac{n_i - d_i}{n_i}$ is the percentage of at-risk individuals who died at time t_j, thus the estimated survival at time t, denoted $\hat{S}(t)$, is the product of the estimated survival probabilities up to time t. Notice that individuals who are censored are still included in the population of at-risk individuals (denoted n_i for time i) until their time of censorship, and thus still have an effect on the survival function estimation, even though their death is never observed. Even if an individual drops out of a medical study prior to its completion, we still know that the individual had survived at least up until the time of their censorship, and given the prevalence of censorship in virtually all survival datasets, it is important that a survival model can leverage this information to improve its estimates. Figure 2.1 is an example of a KM curve, where the standard errors are computed using Greenwood's formula [21]:

$$\hat{var}(\hat{S}(t)) = \hat{S}(t)^2 \prod_{i:t_i \leq t} \frac{d_i}{n_i(n_i - d_i)}. \tag{2.15}$$

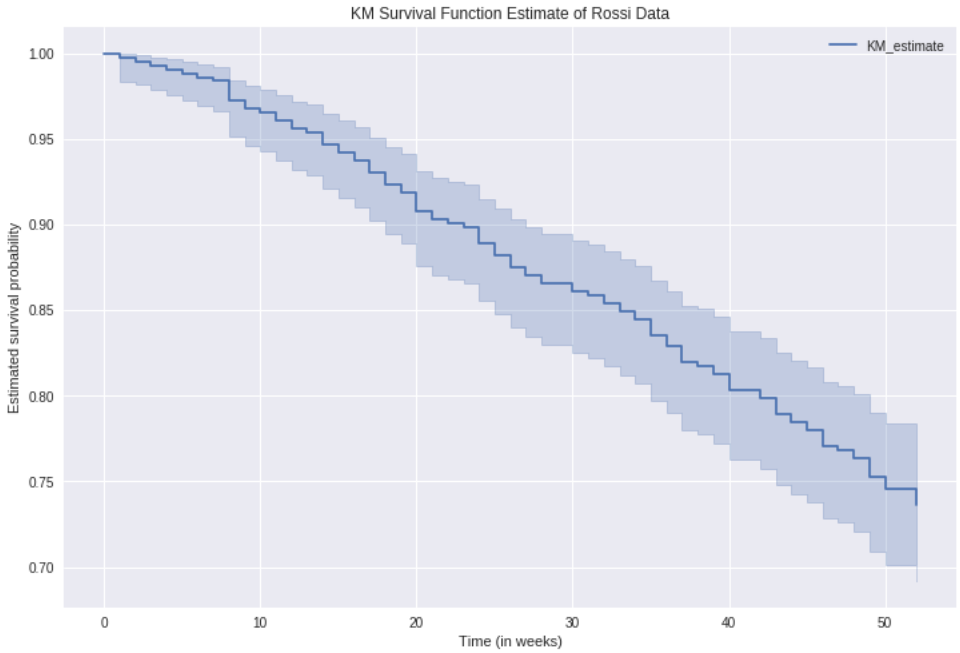

Figure 2.1: Kaplan-Meier curve for Rossi recidivism dataset [1]

A common use of Kaplan-Meier curves is for simple analysis of which the goal is to compare the survival functions of two different sub-populations. In the example of a medical trial, researchers my want to compare the effects of an experimental drug A vs drug B on the survival time of a diseased population. In a cohort study, researchers may want to compare the rate of lung cancer among smokers compared to non-smokers. In the Rossi [1] dataset mentioned earlier, one may wish to compare recidivism rates of those in different socioeconomic classes. In Figure 2.2 we compare the KM curves between two populations of recently released prison inmates, where one group received financial aid and the other did not.

2.3.2 Cox Proportional Hazards. A Cox proportional hazards model [11] is of the form

$$\lambda(t|\mathbf{x_i}) = \lambda_0(t) \exp(\boldsymbol{\beta}\mathbf{x_i}), \tag{2.16}$$

where i indexes the subjects in the study, $\mathbf{x_i}$ denotes the covariate vector of subject i, $\lambda_0(t)$ is the baseline hazard function, and $\boldsymbol{\beta}$ is a vector of learned scalars that quantify the effect

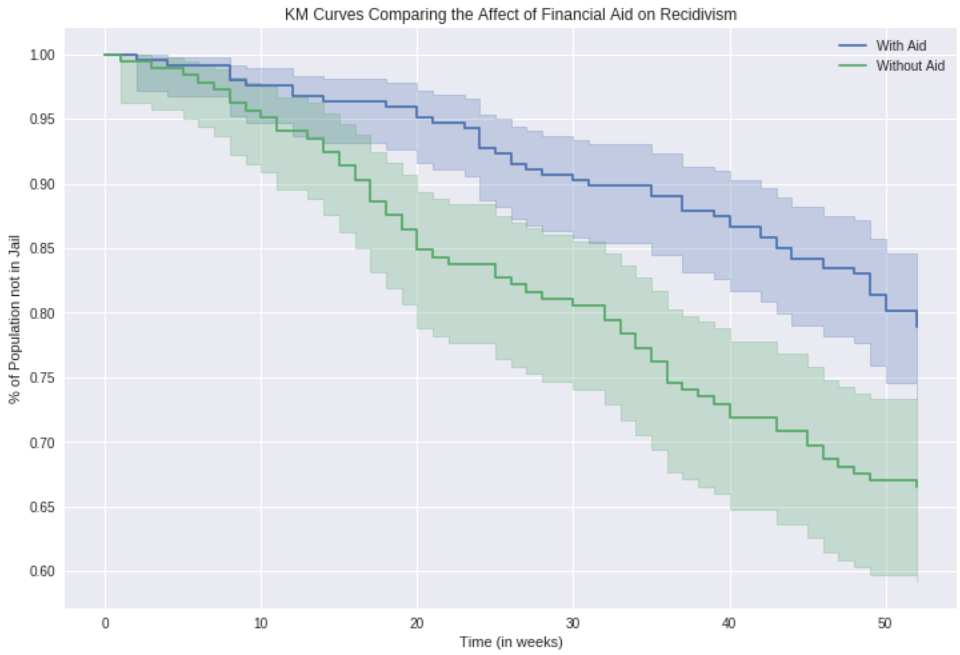

Figure 2.2: A comparison between the KM estimated survival curves. The With-Aid group consists of former inmates of a Maryland correctional facility who, upon release, were given financial aid. The Without-Aid group consists of those individuals who did not receive aid. It's apparent from the two plots that the aided inmates were less likely to relapse into criminal behavior than the unaided population.

each covariate has on the risk of event. The baseline hazard function gives the hazard for a subject whose covariates are all zero. The assumption made by the proportional hazard model is the aptly named *proportional hazards assumption*, which assumes that the ratio between the hazard function $\lambda(t)$ and the baseline hazard function $\lambda_0(t)$ (this is known as the *hazard ratio*) is constant over time. Re-writing (2.16) as

$$\lambda(t|\mathbf{x_i}) = \lambda_0(t) \exp(\beta_1 x_{i,1}) \exp(\beta_2 x_{i,2}) ... \exp(\beta_n x_{i,n}) \tag{2.17}$$

makes clear that for each unit increase of covariate $x_{i,k}$, the hazard is multiplied by $\exp(\beta_k)$, and that, as mentioned earlier, the hazard function $\lambda(t|\mathbf{x_i})$ is a multiplicative constant of $\lambda_0(t)$, where that constant is determined by the covariates $\mathbf{x_i}$. It is also assumed that the effect of each covariate on the hazard is the same for all times t. Consider a simple example where we have a dummy variable x which indicates whether or not a patient is receiving an experimental treatment. Then the model is

$$\lambda(t|x) = \begin{cases} \lambda_0(t) & \text{if } n = 1 \\ \lambda_0(t)e^{\beta} & \text{if } n = 0. \end{cases}$$

Thus the risk of patients who are not receiving the treatment can be modeled with $\lambda_0(t)$ and e^{β} is the the proportionate affect of the treatment on risk. If $e^{\beta} = .5$, (or $\beta \approx -.6931$) the risk of a patient receiving the treatment is half the risk of someone not receiving the treatment at any given time. It's important to note that the Cox proportional hazards model separates the effects time and the covariates have on survival.

We can integrate both sides of 2.16 with respect to t to get the cumulative hazard function

$$\Lambda(t|\mathbf{x_i}) = \Lambda_0(t) \exp(\boldsymbol{\beta}\mathbf{x_i}), \tag{2.18}$$

which is also proportional. Combining this with (2.13) gives us

$$S(t|\mathbf{x_i}) = \exp\left(-\Lambda(t)\right)^{\exp\left(\boldsymbol{\beta}^T\mathbf{x_i}\right)} = S_0(t)^{\exp\left(\boldsymbol{\beta}^T\mathbf{x_i}\right)}, \tag{2.19}$$

where $S_0(t) = \exp\left(-\Lambda(t)\right)$. Examining (2.19), we see that the covariate values (x_i) have the effect of raising the baseline survival function $S_0(t)$ to the power $\exp\left(\boldsymbol{\beta}^T\mathbf{x_i}\right)$.

The likelihood of subject j dying at time t_j with covariate values $\mathbf{x_j}$ can be written as:

$$L_i(\boldsymbol{\beta}) = \frac{\lambda_0(t_j)\exp\left(\boldsymbol{\beta}^T\mathbf{x_j}\right)}{\sum_{i:T_i>t_j}\lambda_0(t_j)\exp\left(\boldsymbol{\beta}^T\mathbf{x_i}\right)}, \tag{2.20}$$

Where the denominator is the sum of the hazards at time t_j of all subjects i whose survival time T_i is greater than t_j (i.e. all subjects who are still alive at time t_j). Note that the baseline hazard function $\lambda_0(t_j)$ cancels out and (2.20) simplifies to

$$L_i(\boldsymbol{\beta}) = \frac{\exp\left(\boldsymbol{\beta}^T\mathbf{x_j}\right)}{\sum_{i:T_i>t_j}\exp\left(\boldsymbol{\beta}^T\mathbf{x_i}\right)}. \tag{2.21}$$

The probabilities that comprise the likelihood are the probability that the subject who dies at time T_i out of all possible subjects is indeed subject i. These likelihoods are treated as independent, meaning that the joint probability over all observed events is the *partial likelihood*

$$L(\boldsymbol{\beta}) = \prod_{i:E_i=1} L_i(\boldsymbol{\beta}), \tag{2.22}$$

where $E_i = 1$ indicates the occurrence of the event of interest. The goal is to estimate $\boldsymbol{\beta}$ by maximizing $\boldsymbol{\beta}$ over the partial likelihood or, equivalently, the log partial likelihood

$$l(\boldsymbol{\beta}) = \sum_{i:E_i=1}\left(\mathbf{x_i}\cdot\boldsymbol{\beta} - \sum_{j:T_j>T_i}\exp\left(\boldsymbol{\beta}^T\mathbf{x_j}\right)\right), \tag{2.23}$$

which can be done using Newton's Method.

It's worth pointing out that this formulation assumes that all events happen at distinct times, which is not necessarily true. There are several approaches to handle situations in which multiple events happen at the same time, but one of the most commonly used methods was presented by [22], who suggests that we sum all the covariate-related parts for all subjects whose event occurs at the same time, and then raise that sum to a power equal to the number of events that occurred at t_i. This turns (2.21) into

$$L_i(\boldsymbol{\beta}) = \frac{\prod_{j \in D_i} \exp(\boldsymbol{\beta}^T \mathbf{x_j})}{\left[\sum_{j \in R(t_i)} \exp(\boldsymbol{\beta}^T \mathbf{x_j})\right]^{d_i}},$$

where D_i is the set of subjects dying at t_i, $R(t_i)$ is the set of all patients who have not died or been censored before t_i, and d_i is the number of subjects in D_i.

Looking at (2.21) we can see that maximizing (2.22) means choosing $\boldsymbol{\beta}$ such that $\exp(\boldsymbol{\beta}^T \mathbf{x_j})$ is high when death time T_j is small and low when T_j is large. This relationship is incentivized by both the numerator and denominator of (2.21). A remarkable property of the Cox proportional hazards approach is that it doesn't need the baseline hazard function $\lambda_0(t)$ to estimate $\boldsymbol{\beta}$. Each β_k determines the multiplicative affect of covariate x_k on the hazard, thus the Cox model allows a researcher to measure the effect of each covariate without having to make any additional assumptions about the mechanism of survival.

2.4 Survival Metrics

2.4.1 Log-rank Test. Quantifying the difference between two survival curves is a common task of a survival analysis. Though the difference in survival functions between the two populations in Figure 2.2 are visually obvious, more quantitative means are required to draw any conclusions. The log-rank test [23] is a simple way to do this.

The log-rank test is used to test the null hypothesis that all of the sub-populations have the same survival probability. The test statistic is computed by comparing the rate of death of the different sub-populations to the death rate of the entire population. If the null

hypothesis is true, then the death rates of the sub-populations should be more or less the same. For example, in the Rossi dataset [1] plotted in Figure 2.1 and 2.2, we have that the first death occurs in the no-aid group at time t_1=1. At the beginning of the week there were 432 at-risk individuals (i.e. not dead and non-censored), and 1 person died at t_1=1, so the expected rate of death is $\frac{1}{432} \approx 0.0023$, which means that the expected number of deaths in the no-aid group is $\frac{1}{432} \times 185 \approx 0.428$ and $\frac{1}{432} \times 247 \approx 0.571$ for the aided group. The second event time is t_2=2, where 1 person in the aided group died. Likewise, we have that, according to the null hypothesis, the expected number of deaths at t_2 is $\frac{1}{431} \times 184 \approx 0.427$ and $\frac{1}{432} \times 247 \approx 0.572$ for the non-aid and aided populations (note that the survival population sizes have been updated to reflect the death at t_1). We continue this process, adding up the expected number of deaths for each event time t_i. We then use a χ^2 test [24], where the test statistic is the sum of $\frac{(O-E)^2}{E}$ for each group, where O is the observed number of deaths in the group and E is the expected number of deaths. In this case, the test statistic is 9.91 with 2-1=1 degree of freedom for $P \leq 0.005$, giving us good reason to reject the null hypothesis.

The logrank test makes the same assumptions as the Kaplan-Meier model, specifically that censoring is non-informative, that the survival function is the same for those who entered the study at different times. The log rank test has a difficult time distinguishing survival curves that cross, since the crossing can cause the expected number of deaths for each population to be similar, even though the curves themselves are very different.

2.4.2 Concordance Index. In survival analysis, two patients are said to be concordant if the patient with the higher predicted survival time has an earlier time of death T than the patient with the lower predicted risk. The concordance index (CI), or c-index of a model is the fraction of all pairwise survival time predictions that are concordant. It can be written as

$$c = \frac{1}{|\mathcal{E}|} \sum_{T_i \text{ uncensored}} \sum_{T_j > T_i} \mathbf{1}_{f(x_i) < f(x_j)}, \tag{2.24}$$

where T_i denotes the time of death of subject i, $\mathcal{E}$ is the total number of possible pairwise comparisons, $f(x_i)$ is the predicted survival time of subject i by model f, and $\mathbf{1}_{a<b} = 1$ if $a < b$. A few properties of the concordance index are:

- CI scores are between 0 and 1, with 1 being a perfect score
- A CI score of .5 is equivalent to random guessing
- A score of 0 is perfect anti-concordance
- Well trained models typically have CI scores between 0.6 and 0.75, as there is usually too much noise in a problem for perfect concordance to be possible

Because a subjects predicted survival time is proportional to their risk, a model that predicts a subject's risk, such as the proportional hazards model, can also use the concordance metric to measure the quality of its predictions, and the c-index score can be compared to models that predict survival time directly.

The concordance index is a ranking metric, it doesn't care about the actual predicted survival times, just that the survival times are ordered correctly, however, a model with good survival time predictions will have a good concordance index. The converse is not true, and though it's unlikely that a model trained using standard survival analysis techniques would produce poor survival predictions while having a good concordance index, it's good practice to do a sanity check on survival time predictions to ensure they are inline with what one would expect. For this and other reasons, it's one of the most commonly used metrics for evaluating performance of survival analysis models.

Chapter 3. Sequence Modeling

Sequential data is defined as data to which a natural ordering exists, and that maintaining this natural ordering can provide additional information to a machine learning model. A few examples of sequential data are:

- **Text data**: Written text is a sequence of words, where the natural ordering is the order in which the words appear in the text, separated by spaces. For example, the sentence "The quick brown fox." is the ordered sequence of words ['The' 'quick', 'brown' 'fox']. More granular models break text down to its individual characters, which are also sequenced in the order in which the characters appear in each individual word in a sentence, including spaces, so in a character model "The quick brown fox." would become the ordered sequence ['T', 'h', 'e', ' ', 'q', 'u', 'i', 'c', 'k', ' ', 'b', 'r', 'o', 'w', 'n', ' ', 'f', 'o', 'x']. The ordering of words contains important information about the meaning or interpretation of the sentence. For example, "John dislikes Sally" has a different meaning than "Sally dislikes John" even though they are composed of the same words. Even simple phrases can depend on order, such as "party hat", which specifies a type of hat (for parties), compared to "hat party", which is a party where everyone wears an extravagant hat.

- **Medical data**: Medical data is electronically recorded as a sequence of ICD medical codes, in which every code corresponds to a doctor's diagnosis. Each diagnosis occurs during some contact with a medical provider, such as a routine doctor's visit or a trip to the emergency room. An intuitive ordering of these codes is chronologically by the date in which the diagnosis was given. Ordering the codes by time can reveal the trends of a person's overall health that can inform predictions of future diagnosis, such as the IC example from Chapter 1.

- **Stock Market**: Stock market data consists of regular, time-stamped price updates of different stocks, or perhaps a single stock. This information is critical in the field of mathematical finance, and for Wall Street investors. Because the stock price is often measured at uniform intervals of time, these data can be referred to as time series data.

- **Demographic Data**(**unexample**): Demographic data consists of the age, race, sex, income, level of education, and other variables that describe a person's socioeconomic

profile. Such information is often considered when applying for a loan, or for admission to a university. There is no important ordering to any of these data points, thus this data is not considered sequential data.

3.0.1 Representation of Sequential Data. To be machine readable, a sequential data must be represented as a sequence of numerical objects, usually vectors. The way in which a sequence is numerically represented can vary depending on the problem. For example, a sequence of daily stock prices of all Fortune 500 companies has a straightforward numerical form: one must first create a unique mapping between each company and the set $\{i \in (N) | 0 \leq i < 500\}$, referred to as an enumeration, then each element in the sequence can be represented as vector $v_t \in \mathbb{R}^{500}$, where index i has the stock price of the ith stock, as determined by the enumeration, and t indexes the time. Thus, if S represents our sequence, we have

$$S = (v_1, v_2, ..., v_{T-1}, v_T)$$

Not all sequential datasets have such a straightforward numerical format. For example, consider text data. Text data is represented as a sequence of symbols, usually either words or characters. The most common way to represent symbolic data is to get an enumeration of your symbol vocabulary. For a word sequence, the vocabulary is the set of all English words that can plausibly be encountered. For a character sequence, the vocabulary could be the set of all ASCII printable characters. Symbols are mapped to an integer using the enumeration of the symbol vocabulary. Using the case above, if our sequence is

[‘t’, ‘h’, ‘e’, ‘ ’, ‘q’, ‘u’, ‘i’, ‘c’, ‘k’, ‘ ’, ‘b’, ‘r’, ‘o’, ‘w’, ‘n’, ‘ ’, ‘f’, ‘o’, ‘x’]

and we have the enumeration

b:1	c:2	e:3
f:4	h:6	i:7
k:8	n:9	o:10
q:11	r:12	t:13
u:14	w:15	x:16
' ':17	SOS:0	EOS:19

Then our mapped sequence would be

[13, 6, 3, 17, 11, 14, 7, 2, 8, 17, 1, 12, 10, 15, 9, 17, 4, 10, 16].

While the sequence now has a numerical representation, the form above has a potential problem. Machine learning algorithms often use numerical distance as a similarity metric. Thus, the representation above has inadvertently introduced a notion of similarity between symbols that does not make a lot of sense (are 'b' and 'e' equally close to 'c'?). This introduces false signal that can impede a model's ability to learn meaningful relationships in the data. It is important, therefore, to represent the data in such a way that there are no false-similarities. The most common way to do this is via one-hot encoding. One-hot encoding is mapping integer values to a sparse vector. Specifically, if there are 20 symbols in your vocabulary, then each symbol can be mapped to a vector v, where every element of v is zero except for a 1 at index i, where i is determined by the enumeration. Using the example above, the first element of the sequence would be represented by a vector v_1 with zeros everywhere except for at index 13. Vectors of this type are known as one-hot vectors, and they are frequently used to represent symbolic data. It addresses the similarity issue by making all of the vector representations orthogonal and equidistant.

A last note of interest regarding sequential data representation. Note that the enumeration has mappings for SOS and EOS symbols. These symbols stand for start-of-sequence and end-of-sequence, and there are times, especially in natural language processing with text, when concatenating SOS and EOS symbols on the beginning and end of a sequence is useful. For example, in text generation, a model will continuously generate text until eventually it

outputs the SOS/EOS symbol to indicate a natural endpoint, such as the beginning/end of a sentence or paragraph.

3.1 Sequence Models

Virtually all state-of-the-art machine learning algorithms for sequential modeling are variations of neural networks, such as recurrent neural networks [25], sequential convolutional neural networks [26], and transformer models [27] [28]. All of our experiments were performed with recurrent neural networks (RNNs). The same concepts and practices that are used in dense neural networks (DNNs), also known as feed-forward or vanilla neural networks, are also used in RNNs, so we start with a primer on DNNs.

3.1.1 Neural Networks. In its simplest form, a neural network is a type of machine learning algorithm that uses a series of learned weight matrices $(W_1, W_2, ...W_\ell)$ and a non-linear function σ to map an input vector x to an output $\hat{y}$:

$$\hat{y} = W_\ell(\sigma(W_{\ell-1}(\sigma(W_{\ell-2}(....\sigma(W_1(x))))))).$$

The user-defined nonlinear function σ is applied element-wise after each transformation (aka matrix multiplication). The purpose of σ is to introduce a non-linearity into the model, allowing it to learn non-linear relationships between the features of x. An intuitive way to think of this is that the network is learning a more efficient vector representation of x, denoted $\tilde{x}$ via matrix multiplication whose features are better suited for the task the model is being trained on:

$$\tilde{x} = \sigma(W_{\ell-1}(\sigma(W_{\ell-2}(....\sigma(W_1(x)))))).$$

This new vector representation can now be linearly transformed into the prediction $\hat{y}$:

$$\hat{y} = W_\ell\tilde{x}.$$

In neural network terminology, the vector-valued function composition $\sigma(W_i(x))$ is called the ith "layer" of the network, with ℓ being the total number of layers. A model with lots of layers can learn complex relationships between the input features of x, this ability to learn a new representation specific to its task is why neural networks excel in data modalities that are difficult to efficiently represent, such as images and text. The weight matrices can differ in size, so long as the dimensions align enough to make matrix multiplication possible, i.e. if $W_i \in \mathbb{M}^{m_i \times n_i}$, then we must have that $n_i = m_{i-1}\ \forall i\ s.t.\ 1 < i \leq \ell$. The values $((n_1, m_1), .., (n_i, m_i), .., (n_\ell, m_\ell))$ are model hyperparameters chosen by the user.

Training Neural Networks. The entries of each weight matrix (also known as "weights") are initialized randomly [29], and then updated to minimize a loss function. The loss function is a differentiable function whose purpose is to evaluate how well a network models a dataset. If the predictions of the network are not good, then the loss function output will be high, if the predictions are good, then the loss function output will be low. For example, if we were trying to predict the price of rent of a New York City apartment based on features such as geographic location, average rent of the apartments nearby, what floor the apartment is on, etc., then the dataset would be the set $\{(x_i, y_i), |i \in \mathbb{N}, i \leq N\}$, where N is the number of (x,y) pairs in the dataset, x_i is the feature vector and y_i is the price of rent of apartment i. There are several loss functions that would work for this task, one of them being

$$L(\hat{y}, y) = |\hat{y} - y|$$

because it matches the criteria that we defined earlier: it's differentiable (we'll define the derivative to be 0 in the $\hat{y} = y$ case), and it is a good metric for how well the network models the dataset. If an apartment has a monthly rent of \$4000, and the network predicts \$500, the loss for that example would be $L(500, 4000) = |500 - 4000| = 3500$, but if the network predicts a rent of \$5600 for an apartment with a monthly rent of \$5000, the loss for this $(\hat{y},y)$ pair would be $L(5600, 5000) = |5600 - 5000| = 600$. Note that the better the prediction is, the lower the loss function will be.

Backpropagation. While loss functions provide a static representation of how the model is performing, it is also the means by which we fit a model to the data in the first place. This is done using an optimization method known as backpropagation [30] to iteratively change the weights in the network to improve the predictions of the network w.r.t. the loss function. Backpropagation uses the chain rule to compute the derivative of the loss w.r.t. to each weight in the entire network, the partial derivatives are then used to update the weights in the network via gradient descent.

Activation Functions. The non-linear function σ is called the model's activation function. Historically, the activation function of choice was the sigmoid function, however the sigmoid function has a derivative with some undesirable properties that hamper a network's ability to learn (see Figure 3.1). More recently networks have used the rectified linear unit (ReLU) function. ReLU's are nonlinear, fast to compute, and don't dampen the gradient signal in the same way as the sigmoid function (see description of Figure 3.1).

Batch Training. When training a neural network, finding the loss, computing the gradient, and updating the weights for each training example one at a time is both slow and unstable. It's unstable because there is a lot of variance between the loss of all the examples, and training one-by-one can allow outliers to derail the model. If this occurs early on in the training process, it can make it difficult for the model to eventually converge. A better strategy is to compute the gradient for a small random subset of the data and update the weights based on the average gradient. This has two main advantages, first, it's easy to parallelize the training process for groups of training examples, so computing the gradients is computationally much faster, and second, because the gradient is an average of multiple training examples, it has a much lower variance and is more robust to outliers, making the network more likely to converge.

3.1.2 Recurrent Neural Networks.

A recurrent neural network (RNN) is a specific class of neural networks that were designed to work on sequential data. RNNs are like feed-forward neural networks, but with a few augmentations that make them adept for operating

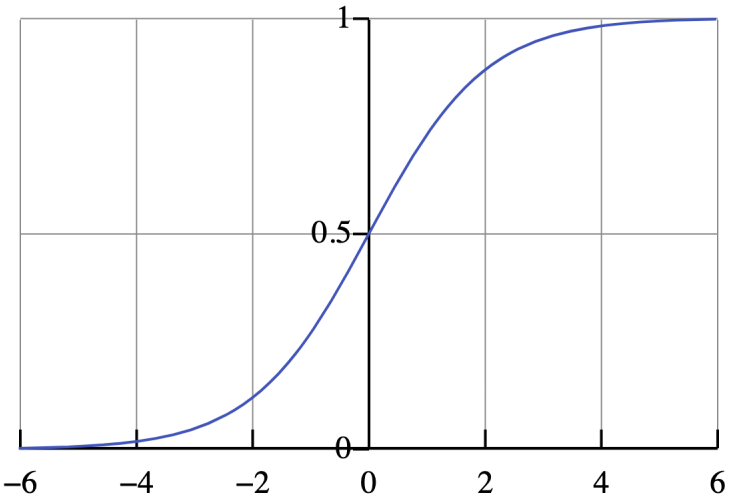

Figure 3.1: The sigmoid function was used in early neural networks because of it can intuitively be thought of as having neuron-like behavior. Neurons in the brain are either passing or inhibiting signal, the sigmoid function maps most values either close to 0 or close to 1, and thus can be seen as being either on (close to 1) or off (close to 0), like a neuron. Unfortunately the sigmoid function has a derivative with undesirable properties. If a weight in a neural network gets too extreme, the gradient gets close to zero. With a near-zero gradient, the weight never updates and gets stuck. Even without an extreme value, the derivative of the sigmoid function has a maximum value of .25. With multiple layers, the chain rule multiplies these small values together until the gradient gets close to zero, an issue known as the vanishing gradient problem. Image source: Wikipedia

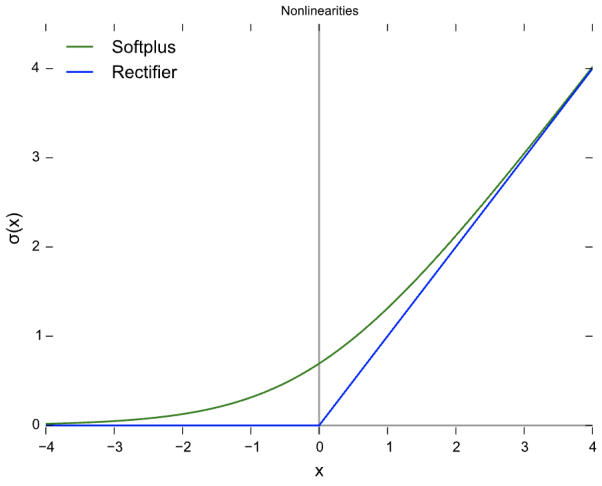

Figure 3.2: The ReLU function, graphed next to it's non-piecewise (and expensive) analog, the softplus function, is fast to compute, non-linear, and doesn't dampen the gradient in the same way as the sigmoid function. While negative values do have a zero gradient, in a sufficiently large network this will sparseify the gradient channels, but not eliminate them. Image source: Wikipedia

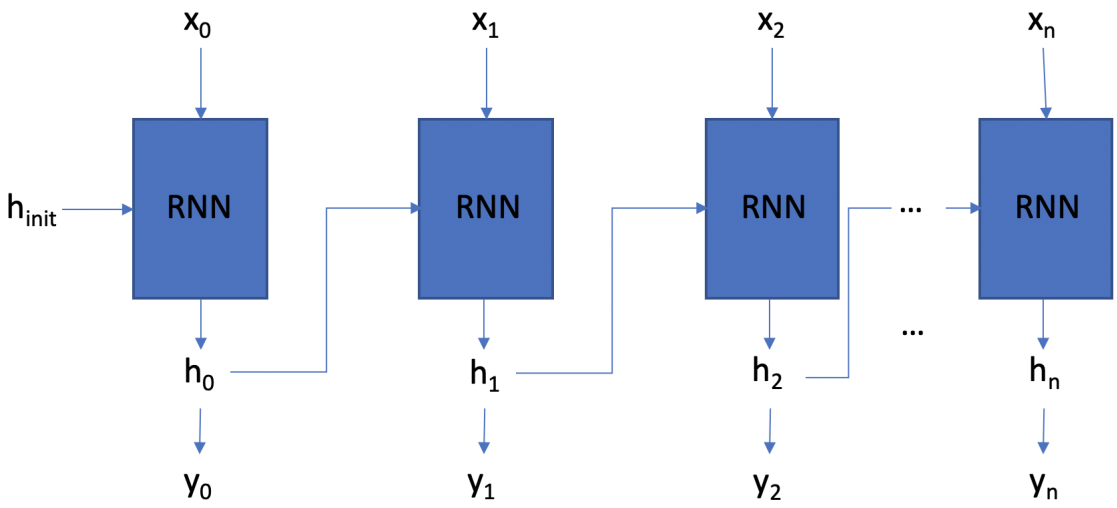

Figure 3.3: A diagram of an unrolled RNN. At each time t, the RNN accepts x_t and h_{t-1} as inputs and uses them to compute h_t, which in turn is used to compute the output for that time step, y_t. Note that the same operations are being applied at each time step, with the same weights, but with different inputs.

on sequences, the primary difference being that RNNs have two inputs: x_t and a vector h_{t-1} known as the state vector (see Figure3.3). In the canonical "tanh RNN", h_t and output y_t are computed using the equations

$$h_t = \tanh(W_x x_t + W_h h_{t-1}) \tag{3.1}$$

$$y_t = W_y h_t, \tag{3.2}$$

The intuition is that the state vector h_t acts as an internal memory of everything that has happened in the first t elements of the sequence. This memory is what makes RNNs so effective at sequence modeling, especially compared to markov models, which are stochastic state models whose next state transition probabilities only depend on the current state (they have no memory), which is fine for some applications, but in others, such as natural language processing, having a memory of what words have been said provides context that can improve

task performance. For example, if I'm trying to predict the last word of the sentence "It looks cloudy outside, I think it's going to *rain*", the context provided by the words "It looks cloudy outside, I think it's going to..." makes "rain" a reasonable guess. However, if a sequential model doesn't have the capacity to remember any of the previous words in the sequence, such as a markov model, trying to guess the next word from "to..." can be almost anything.

Gated Recurent Units. While the tanh RNN is a good model, there are a few features that are lacking:

- The entire hidden state is re-computed at each time step (see equation 3.1). This can make storing long-term dependencies difficult. For example, if the first element of a sequence was important for the task the model is being trained for, and the sequence is 20 elements long, it's unlikely that the process of rebuilding the state at each time step is going to preserve the signal all the way to the end.

- There might be times when it is useful to "reset" the hidden state, due to some natural break in the sequence. For example, if a person is sick with a non-chronic disease, but then becomes healthy again, memory of that event is no longer useful for predicting future events because the person is no longer sick, and it would be useful to have a mechanism that allows the model to forget irrelevant parts of the hidden state.

- Skipping over useless parts of the sequence. In the case of medical codes, a doctors visit for a sunburn is not likely to be something worth incorporating into the memory. Excluding these type of events makes it easier to preserve signal that is actually relevant. In many applications, such as disease prediction, there is a relatively small percentage of elements in the sequence that are relevant, the tanh RNN does not have any direct mechanism for addressing this.

Gated Recurrent Units [31] (GRUs) and Long Short-Term Memory networks [32] (LSTMs) are two widely used RNN models that were designed to address these shortcomings, and do

so by incorporating gating of the hidden state, meaning that GRUs/LSTMs have dedicated mechanisms that determine when and how the hidden state should be updated and/or reset. These mechanisms give GRUs and LSTMs better empirical performance over non-gated RNN models, such as the tanh RNN [33].

The equations that define a GRU are as follows:

$$r_t = \sigma\left(W_{rx}x_t + W_{rh}h_{t-1}\right) \tag{3.3}$$

$$z_t = \sigma\left(W_{zx}x_t + W_{zh}h_{t-1}\right) \tag{3.4}$$

$$\tilde{h}_t = \tanh(W_{hx}x_t + W_{hh}\left(r_t \odot h_{t-1}\right)) \tag{3.5}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \tag{3.6}$$

Lets break down what is going on in each gate:

$$r_t = \sigma\left(W_{rx}x_t + W_{rh}h_{t-1}\right)$$

r_t is called the reset gate, and is multiplied element-wise with previous state h_{t-1} in equation (3.5). For both (3.3) and (3.4) σ is the sigmoid function $\sigma(x) = \frac{1}{1-e-x}$, which has a range of $[0, 1]$. Since all entries of r_t will be between 0 and 1, the hadamard (i.e. element-wise) product $r_t \odot h_{t-1}$, r_t is acting as a gate, determining what parts of the old hidden state h_{t-1} should be passed on and incorporated in to the proposed state $\tilde{h}_t$. The closer an entry of r_t is to zero, the more the model believes that the corresponding feature in h_{t-1} should be reset. Note that r_t is a function of x_t and h_{t-1}.

$$z_t = \sigma\left(W_{zx}x_t + W_{zh}h_{t-1}\right)$$

z_t is known as the update gate. It has the same structure as the reset gate, where all entries are between 0 and 1 and it's a function of x_t and h_{t-1}, but it plays a different role. Looking

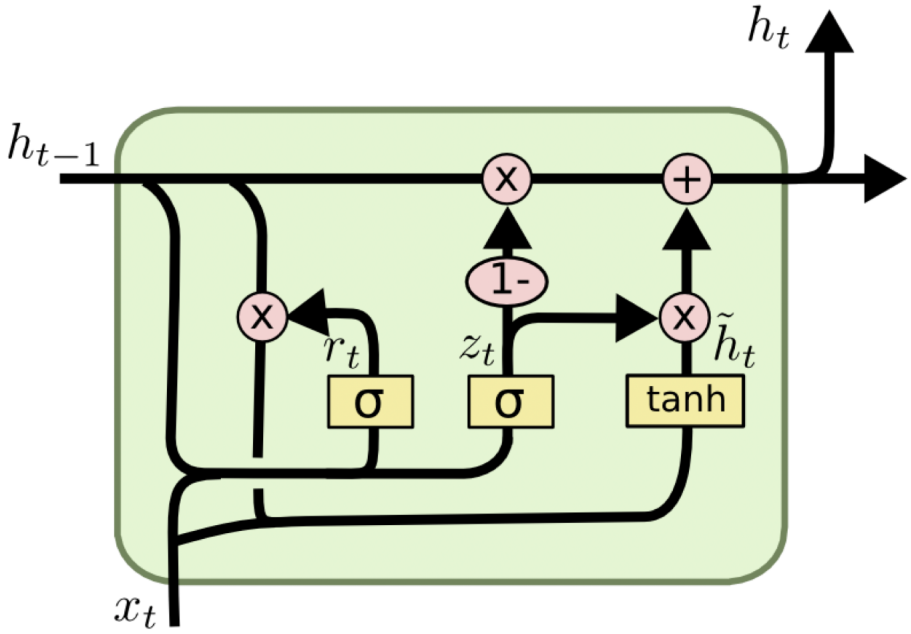

Figure 3.4: A diagram of a Gated Recurrent Unit (GRU), which is a type of RNN that uses two gate mechanisms to determine what parts of the old hidden state h_{t-1} should be incorporated into the the new state h_t and which elements of the sequence should be ignored. These additions help the GRU preserve and prioritize information in the sequence that is useful. Image from Colah's blog http://colah.github.io/posts/2015-08-Understanding-LSTMs

at (3.6), we see that z_t determines what portion of the new hidden state h_t will be comprised of the old state h_{t-1} and the proposed state $\tilde{h_t}$.

GRUs have achieved state-of-the-art performance on several sequential modeling tasks, and are designed to remember important elements in a sequence and discard the elements that are not important. This is especially important in potential medical applications, where sequences can be long and most elements in the sequence are from routine visits.

Chapter 4. Experiments

4.1 Introduction

In this chapter, we formulate several different survival analysis tasks and compare the performance between deep learning and standard survival methods. As stated in Chapter 2, most survival analysis models require the researcher to make assumptions on the underlying mechanism of survival and the relationship between covariates and survival time, such as the proportional hazards assumption. While these assumptions may be helpful, we especially seek to find scenarios in which the flexibility of deep learning, and its ability to learn complex non-linear relationships between covariates and survival time, allow it to perform better than its survival model counterpart.

The structure of this chapter is as follows. First, we will highlight related work, and then we will describe the general structure of the data used for our synthetic experiments. Next we will review in detail the setup of each experiment and present the results. We follow with a discussion of results and draw some conclusions.

4.2 Related Work

There have been several research papers on applying deep learning to survival analysis, the most prominent of which is DeepSurv [34]. DeepSurv is a feed-forward neural network that

optimizes the Cox partial likelihood, essentially becoming a Cox model that can learn non-linear relationships between the covariates. In [35] a feed-forward neural network is trained on a joint loss function of the Cox partial likelihood and a ranking loss that borrows some ideas from isotonic regression, and the model outputs both the risk and survival functions. Finally [36] uses an RNN to get a risk score, which is then optimized using a Cox partial likelihood loss. The model creates a sequence by appending a time stamp onto the end of a fixed feature vector, to allow the model to learn how time can impact covariate relationships. As far as I can tell, this is the only research that uses a recurrent neural network as a way to encode the sequential data that is present in some survival analysis problems, such as with EHI data, and then use that encoding to estimate a survival function.

4.3 Synthetic Data

Many time-to-event datasets, upon which all survival analysis is performed, can also be structured as sequential data, especially in the medical field, where survival studies are common. For example, an insurance company may be interested in projecting the probability of its members contracting a number of high-cost diseases, with the hope of identifying high-risk individuals with whom they can engage with preventative medicine, both cutting costs and improving patient well-being. This can be approached as a traditional survival analysis problem, using Cox proportional hazards to predict the onset of a medical condition based on certain covariates, such as age, sex, race, and any others that the researcher may wish to include. It can also be viewed as a sequential modeling problem, where each contact with a medical service provider is part of a time-ordered sequence representation of an individual's medical history. The synthetic data created for our experiments can likewise be seen as either a survival, or sequential modeling problem.

Though our data varies for each experiment, the process for generating it is the same. First, we generate a covariate vector for each "subject" in the dataset. This vector is usually sampled from a multivariate uniform distribution. The covariates are then used to define a

transition matrix $M \in \mathbb{R}^{n \times n}$, where n is the number of states and where each row of M sums to 1. With transition matrix M, sequential data is generated in a markov chain fashion, where for the first step $t_0 = 0$, we have an initial state $s_0 \in [0, 1, ..., n-1]$. The next state, s_1 is sampled from the multinomial distribution defined by the row of M indexed by s_0. Then s_2 is sampled from the multinomial distribution defined by the row of M indexed by s_1, and so on, until we sample s_T, where T is a hyperparameter.

After we generate the sequence $(s_1, s_2, ...s_T)$, $s_i \in [0, 1, ..., n-1]$, we retroactively apply early stopping conditions in the sequences. For example, let's say we have the following sequences (n=3):

$$[0, 1, 2, 2, 1, 0, 0, 1, 2, 0, 1, 1, 1, 0, 2, 2, 0, 1, 2]$$

$$[0, 2, 1, 2, 1, 0, 1, 0, 0, 2, 1, 0, 2, 1, 0, 1, 0, 1, 1]$$

$$[1, 2, 1, 2, 2, 2, 1, 0, 1, 0, 0, 1, 0, 2, 1, 1, 1, 2, 0],$$

and lets say that our early stopping conditions are

$$[0, 0, 0],\ [1, 1, 1],\ [2, 2, 2].$$

Looking at our generated sequences, we see that the first and last sequences contain the early stopping conditions $[1, 1, 1]$ and $[2, 2, 2]$ respectively. We apply these early stopping conditions by "stopping" the sequence at the earliest time when one of the stopping conditions is met. In this case, the three sequences above would be changed to:

$$[0, 1, 2, 2, 1, 0, 0, 1, 2, 0, 1, 1, 3]$$

$$[0, 2, 1, 2, 1, 0, 1, 0, 0, 2, 1, 0, 2, 1, 0, 1, 0, 1, 1]$$

$$[1, 2, 1, 2, 2, 3],$$

where we use 3 to mark the point where the stopping condition was met.

In addition to stopping criteria, we also have a non-informative censoring mechanism that can be used to shorten the sequences. At each step in the generative process there is a p=.1 probability of censorship.

4.4 Experiment

4.4.1 Data. For our first experiment we generate data using a sinusoidal function. Each subject in the dataset is characterized by as single covariate $c \sim U(0, 5)$, that is then used to define the subject's transition matrix:

$$\begin{bmatrix} \sin(c) & \frac{1-\sin(c)}{2} & \frac{1-\sin(c)}{2} \\ \frac{\sin(c)}{2} & 1-\sin(c) & \frac{\sin(c)}{2} \\ \frac{2}{3}\sin(c) & 1-\sin(c) & \frac{\sin(c)}{3} \end{bmatrix} \tag{4.1}$$

with early stopping conditions

$$[0, 0, 0], [1, 1, 1], [2, 2, 2].$$

For censoring, at each time step each subject was given a probability p of being censored. This was implemented by adding a *censored* absorbing state to the transition matrix and adjusting the transition probabilities accordingly:

$$M_{c,p} = \begin{bmatrix} \sin(c) - \frac{p}{3} & \frac{1-\sin(c)}{2} - \frac{p}{3} & \frac{1-\sin(c)}{2} - \frac{p}{3} & p \\ \frac{\sin(c)}{2} - \frac{p}{3} & 1-\sin(c) - \frac{p}{3} & \frac{\sin(c)}{2} - \frac{p}{3} & p \\ \frac{2}{3}\sin(c) - \frac{p}{3} & 1-\sin(c) - \frac{p}{3} & \frac{\sin(c)}{3} - \frac{p}{3} & p \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.2}$$

This was the transition matrix used to generate state sequences for each subject in the dataset, with a maximum sequence length of 20, meaning that after 20 time steps we stop generating a new state in the sequence, regardless of whether or not death has occurred.

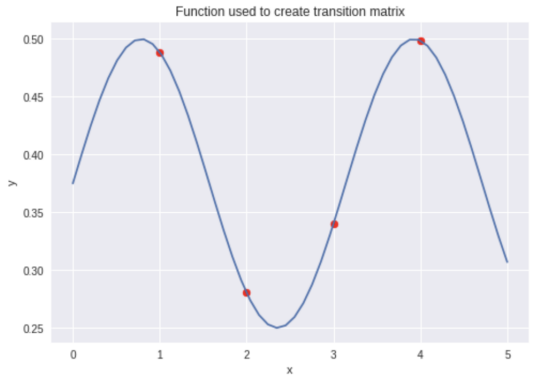

Figure 4.1: We chose to base our transition matrix on the sine function its non-linearity provides a good example for neural network models to demonstrate their ability to learn non-linear relationships. The red dots denote the four points where we evaluate the survival function estimates $\hat{S}(t)$ of both the RNN and a Cox proportional hazards models.

This maximum length threshold is analogous to the end-of-observation thresholds that are common in real-world survival analysis datasets, such as the end of a medical study.

In total, the generated dataset consisted of 400,000 datapoints, where each datapoint d_i consists of a covariate feature c_i, a transition matrix $M_{c_i,p}$, and a sequence (generated via $M_{c_i,p}$) $S_i = (s_0, s_1, ...s_n)$ where $n < 20$ and $s_0 = 0$ for the initial state.

4.4.2 Model. The three models used in this experiment are a recurrent neural network with cross entropy loss (RNN CE), an RNN with a Cox partial likelihood loss (RNN Cox), and a Cox Proportional Hazards model. The RNN CE uses a 2-layer GRU cell, an internal state size of 100 units, an initial learning rate of $lr_0 = .0005$ with $lr_i = .7^{lr_{i-1}}$ where i indexes the number of training epochs. The output of the RNN CE is input into a final linear layer to get the outputs for each time step. These outputs are C-dimensional vectors with C denoting the number of possible states. For each time step, the output is input into the negative log-cross-entropy (CE) loss function

$$\begin{aligned} \text{loss}(x, correct\_state) &= -\log\left(\frac{\exp(x[correct\_state])}{\sum_j \exp(x[j])}\right) \\ &= -x[correct\_state] + \log\left(\sum_j \exp(x[j])\right), \end{aligned} \tag{4.3}$$

where the total loss for each sequence is the sum of all the negative CE losses from each time step, the batch loss is the sum of all of the sequence losses, and the RNN CE model is optimized to minimize this loss. The CE loss penalizes the network for making wrong predictions, so using it to optimize the network will train the network to make correct predictions based on past elements in the sequence, in other words, we are encouraging the network to learn the conditional probability $p(s_i|s_0, s_1, ..., s_{i-1})$.

For training, each sequence is appended with a start-of-sequence (SOS) token used to signal to the network the beginning of a new sequence, and for batch-training, each sequence such that $length(S_i) < 20$ is padded to length 20 (not including the SOS token) with the appropriate end-of-sequence (EOS). There are two EOS tokens, one for death and one for censorship. The three non-terminal states are represented with 0,1,2, death is represented as state 3, censored is state 4, and the SOS token is 5. For example, the set of sequences

$$[0, 1, 2, 2, 1, 0, 0, 1, 2, 0, 1, 1, 3]$$
$$[0, 2, 1, 2, 1, 0, 1, 0, 0, 2, 1, 0, 2, 1, 4]$$
$$[1, 2, 1, 2, 2, 3]$$
$$[0, 1, 2, 2, 1, 2, 0, 0, 1, 0, 2, 0, 1, 0, 2, 1, 1, 0, 1, 2]$$

would be padded to:

$$[5, 0, 1, 2, 2, 1, 0, 0, 1, 2, 0, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3]$$
$$[5, 0, 2, 1, 2, 1, 0, 1, 0, 0, 2, 1, 0, 2, 1, 4, 4, 4, 4, 4, 4]$$
$$[5, 1, 2, 1, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]$$
$$[5, 0, 1, 2, 2, 1, 2, 0, 0, 1, 0, 2, 0, 1, 0, 2, 1, 1, 0, 1, 2],$$

The model was trained using the Adam [37] optimizer on the loss function detailed in equation (4.3) until convergence, which for neural networks is loosely defined as the point when the loss function stops decreasing with each training iteration.

The RNN Cox model has the exact same hyperparameters as the RNN CE model except for the output is a risk score that is optimized using the Cox log partial likelihood (see equation 2.23).

4.4.3 Ground Truth. The success of the RNN model is measured by comparing its survival function to the ground truth. To get the ground truth we expanded the markov model with terminal conditions into a pure markov model. Note that the current model with transition rules can be fully modeled as a markov model using an expanded transition matrix:

Expanded Transition Matrix

$$
\begin{array}{c}
 \\
0 \\
1 \\
2 \\
(0,0) \\
(1,1) \\
(2,2) \\
3 \\
4
\end{array}
\begin{array}{c}
\begin{array}{cccccccc}
0 & 1 & 2 & (0,0) & (1,1) & (2,2) & 3 & 4
\end{array} \\
\left(\begin{array}{cccccccc}
0 & \frac{1-\sin(c)}{2}-\frac{p}{3} & \frac{1-\sin(c)}{2}-\frac{p}{3} & \sin(c)-\frac{p}{3} & 0 & 0 & p & 0 \\
\frac{\sin(c)}{2}-\frac{p}{3} & 0 & \frac{\sin(c)}{2}-\frac{p}{3} & 0 & 1-\sin(c)-\frac{p}{3} & 0 & p & 0 \\
\frac{2}{3}\sin(c)-\frac{p}{3} & 1-\sin(c)-\frac{p}{3} & 0 & 0 & 0 & \frac{\sin(c)}{3}-\frac{p}{3} & p & 0 \\
0 & 1-\sin(c)-\frac{p}{3} & \frac{\sin(c)}{3}-\frac{p}{3} & 0 & 0 & 0 & p & \frac{2}{3}\sin(c)-\frac{p}{3} \\
\frac{2}{3}\sin(c)-\frac{p}{3} & 0 & \frac{\sin(c)}{3}-\frac{p}{3} & 0 & 0 & 0 & p & 1-\sin(c)-\frac{p}{3} \\
\frac{2}{3}\sin(c)-\frac{p}{3} & 1-\sin(c)-\frac{p}{3} & 0 & 0 & 0 & 0 & p & \frac{\sin(c)}{3}-\frac{p}{3} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right)
\end{array}
$$

which has 8 states and two absorbing states: state 3 for censorship and state 4 for death, and the termination conditions have been broken down into their sub-conditions, which can be interpreted as being different states in the markov model, and then incorporated into the transition matrix. With this representation of our model, we can use the following theorem from [38]:

Theorem 4.1. Let P be the transition matrix of a Markov chain, and let u be the probability vector which represents the starting distribution. Then the probability that the chain is in state s_i after n steps is the ith entry in the vector:

$$u^{(n)} = uP^n,$$

where in this case $u = (1/3, 1/3, 1/3, 0, 0, 0, 0, 0)$ because the starting state is randomly sampled to be 1, 2, or 3, and P is our expanded transition matrix. For each time step $t \in 1, ..., 20$ we get the probability vector $u^{(t)} = uP^t$. The last two values of $u^{(t)}$ are the respective proportions of the population expected to be either censored or dead. We then use

$$S(t) = \prod_{i:t_i \leq t} \frac{n_i - d_i}{n_i}, \tag{4.4}$$

where $d_i = u^{(t_i)}[7] - u^{(t_i-1)}[7]$ is the total proportion of the population expected to die at time t_i, and n_i is the proportion of the population expected to still be alive (i.e. non-censored and not dead) at time t_i, thus $\frac{d_i}{n_i}$ is the probability of dying at time t_i, and since events at each time step are independent the exact probability of surviving up to time t is the product of surviving each at each time step $t_i \leq t$. This is similar to (2.14) except we are using exact values for d_i and n_i via the transition matrix, so it is no longer an estimate, but ground truth.

4.4.4 Results. After training the RNN CE until its cross entropy loss function converged, we also trained a Cox proportional hazards model on the data (see Table 4.4.5) and the RNN Cox model for comparison. To get the RNN CE's estimate of the survival function, $\hat{S}(t)$, we take a Markov chain Monte Carlo (MCMC) approach. For each of our four evaluation points we generate 100,000 sequences in the manner described in Figure 4.6 (with c $\in (1, 2, 3, 4)$ fixed) and then use (2.14) to get $\hat{S}(t)$.

We look at the results of this experiment from two different angles. First, we wanted to see how good the models' predictions were for a wide range of covariate values. To do this,

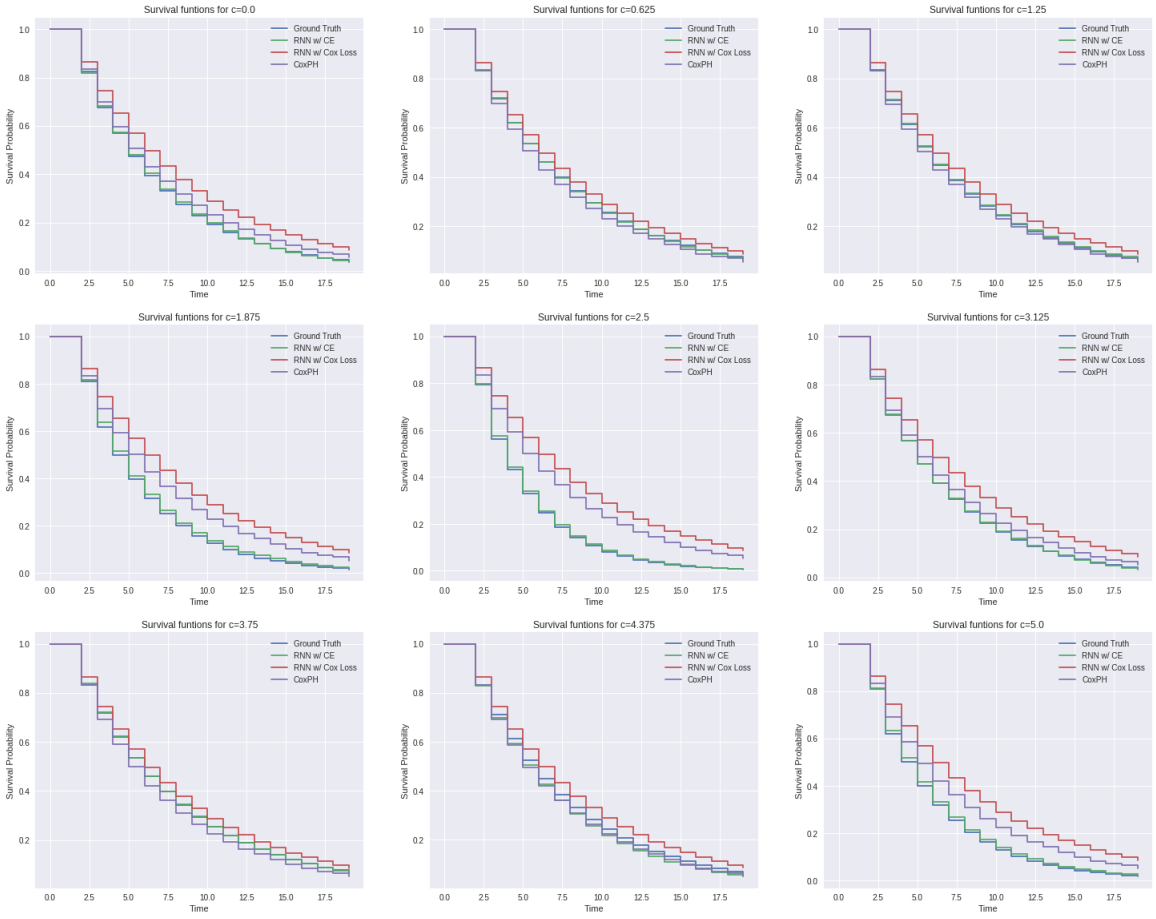

Figure 4.2: Results of Experiment. The recurrent neural network model nearly mirrors the ground-truth survival probability for all values of c, compared to the Cox proportional hazards model, which performs on par for c=1 and c=4, but is clearly bested by its RNN counterpart for c=2 and c=3.

we took 9 evenly spaced points between 0 and 5, which was the range we sampled from when we generated the dataset, and plotted their survival curves from the point t=0 to t=20, to see how well the models could predict the survival curves from just the covariate values (without sequential data). This gave a more apples-to-apples comparison between the RNN and Cox proportional hazards models. As can be seen in figure 4.2, the RNN CE does a fantastic job of predicting the ground truth survival curve, at times even being eyeball perfect with it.

For the second angle, we wanted to see how much improvement we can get from including the sequential data. We chose 3 random datapoints from our held-out test set. Each datapoint has both a covariate value and a sequence. For each time step t in the sequence,

the sequential models processed all elements of the sequence up to time t and then estimated the survival function from that point on. As an example, if a datapoint has covariate c=2.5 and sequence [0,1,2,1,1,3], then for t=2, the RNN would initialize its internal state using c and then iterate over the first two elements of the sequence, [0,1], to update the internal state. From there, the RNN would generate a survival function in the usual matter. The results of these experiments can be seen in Figures 4.3, 4.4, 4.5. In all 3 plots we see the same trend emerge: the RNN CE model does very well at estimating the survival functions. At points where the model is at very high risk of death (at states [0,0], [1,1], and [2,2]) the model is able to capture the signal present in the sequence indicating that imminent death is possible, and the survival functions generated by the RNN CE model reflect that.

4.4.5 Conclusion. There are lots of survival analysis datasets that have sequential components. The purpose of my research was to demonstrate that a survival analysis model that is able to leverage that sequential information to inform its decisions has potential to outperform traditional survival analysis models, such as Cox proportional hazards. The RNN CE model that we trained on a dataset that was designed to mimic the structure of EHR data demonstrated an ability to learn non-linear relationships between the covariates and the survival probabilities, and was also able to use sequential data to improve it's predictions, at times being eyeball-perfect with the ground truth. The natural extension of this work would be to apply these models to a real world EHR dataset, where there is potential to provide survival predictions that can improve decision-making power of both medical providers and patients.

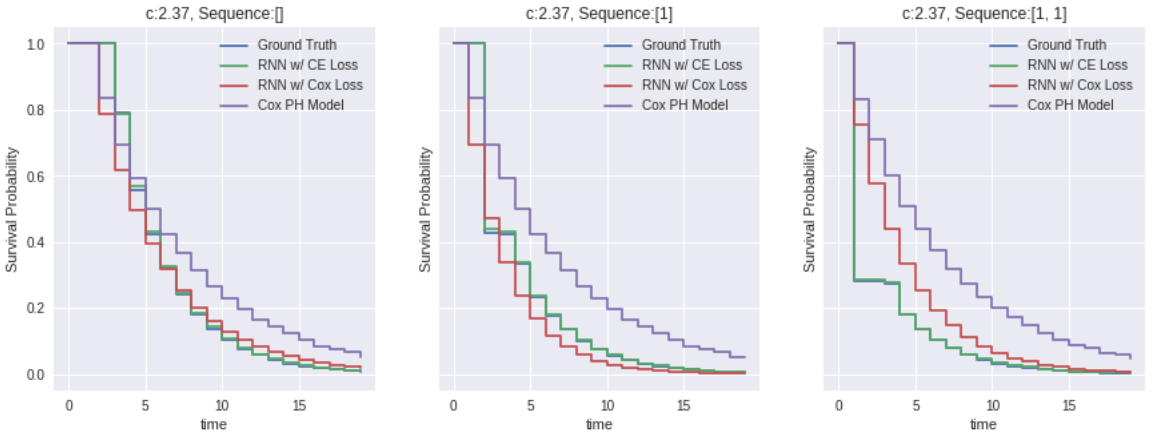

Figure 4.3: Results of Experiment. At each time step, note how the two sequential models adapt their survival functions to match the state, with the RNN CE model is nearly eyeball-perfect with the ground truth. Note how the survival probability predictions change with the state, especially how when the sequence is [1,1], which is close to one of our terminal conditions, the survival probabilities plummet as they should.

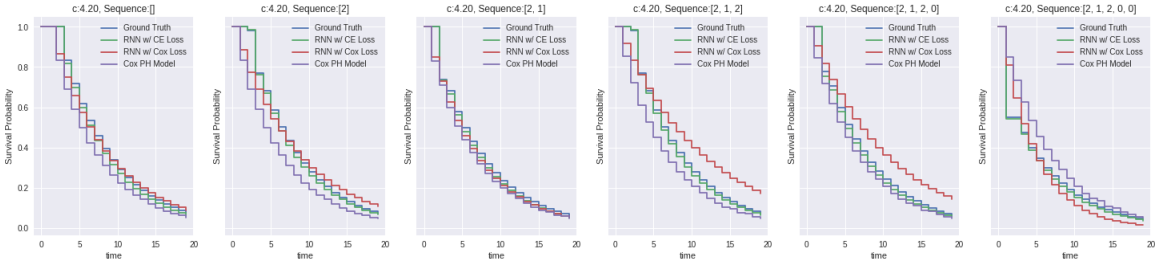

Figure 4.4: Time-step graph of experiment results

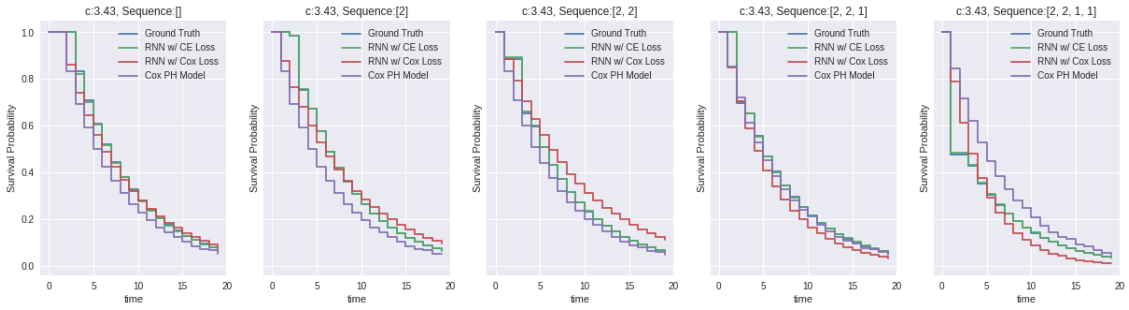

Figure 4.5: Results of Experiment. At each time step, note how the two sequential models adapt their survival functions to match the state, with the RNN CE model is nearly eyeball-perfect with the ground truth. Note how the survival probability predictions change with the state, especially how when the sequence is [1,1], which is close to one of our terminal conditions, the survival probabilities plummet as they should.

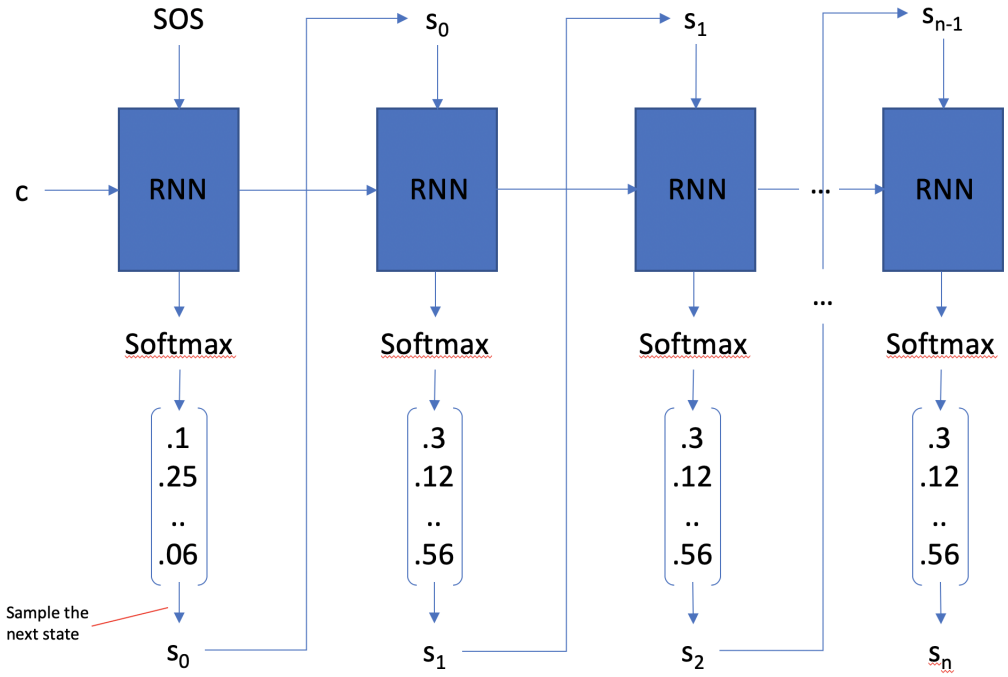

Figure 4.6: To generate a sequence, we start with a covariate value c, which is input into an embedding layer to get the initial hidden state vector. With the inital state, and the start-of-sequence (SOS) as input to the RNN, we softmax the resulting logit vector to get a discrete probability vector, from which we sample s_0. s_0, along with the new hidden state, is then used as the input into the next iteration of the RNN, which produces a new hidden state and samples the next element of the state sequence. This is repeated for n time steps resulting in $S = (s_0, s_1, ..., s_n)$.

	E	T	C
0	0.0	5.0	3.940411
1	1.0	2.0	4.695776
2	0.0	3.0	3.439005
3	0.0	2.0	2.426545
4	0.0	1.0	3.303300
5	0.0	2.0	2.732504
6	1.0	6.0	2.236948

Table 4.1: A sample of the first 6 entries of the dataset from experiment 1, processed for the Cox Proportional Hazards model. C is sampled from a uniform distribution from 0 to 5, and is used to generate the transition matrix seen in 4.2.

Bibliography

[1] Peter H. Rossi, Richard A. Berk, and Kenneth J. Lenihan. *Money, Work, and Crime*. Academic Press, 1980.

[2] Alireza Abadi, Parvin Yavari, Monireh Dehghani-Arani, Hamid Alavi-Majd, Erfan Ghasemi, Farzaneh Amanpour, and Chris Bajdik. Cox models survival analysis based on breast cancer treatments. *Iranian journal of cancer prevention*, 7(3):124–129, Summer 2014.

[3] David Ost, Judith Goldberg, Linda Rolnitzky, and William N Rom. Survival after surgery in stage ia and ib non-small cell lung cancer. *American journal of respiratory and critical care medicine*, 177(5):516–523, 03 2008.

[4] S Nava, F Rubini, E Zanotti, N Ambrosino, Claudio Bruschi, Michele Vitacca, C Fracchia, and C Rampulla. Survival and prediction of successful ventilator weaning in copd patients requiring mechanical ventilation for more than 21 days. *The European respiratory journal : official journal of the European Society for Clinical Respiratory Physiology*, 7:1645–52, 10 1994.

[5] Rebecca L. Siegel, Kimberly D. Miller, and Ahmedin Jemal. Cancer statistics, 2019. *CA: A Cancer Journal for Clinicians*, 69(1):7–34, 2019/07/15 2019.

[6] D. Andrew Loblaw Antonio Finelli Behfar Ehdaie Matthew R. Cooperberg Scott C. Morgan Scott Tyldesley John J. Haluschak Winston Tan Stewart Justman Ronald C. Chen, R. Bryan Rumble and Suneil Jain. Active surveillance for the management of localized prostate cancer endorsement.

[7] Clare Robertson, Andrew Close, Cyntthia Fraser, Tara Gurung, Xueli Jia, Pawana Sharma, Luke Vale, Craig Ramsay, and Robert Pickard. Relative effectiveness of robot-assisted and standard laparoscopic prostatectomy as alternatives to open radical prostatectomy for treatment of localised prostate cancer: A systematic review and mixed treatment comparison meta-analysis. *BJU international*, 112, 05 2013.

[8] Scott C. Morgan, Karen Hoffman, D. Andrew Loblaw, Mark K. Buyyounouski, Caroline Patton, Daniel Barocas, Soren Bentzen, Michael Chang, Jason Efstathiou, Patrick Greany, Per Halvorsen, Bridget F. Koontz, Colleen Lawton, C. Marc Leyrer, Daniel Lin, Michael Ray, and Howard Sandler. Hypofractionated radiation therapy for localized prostate cancer: An astro, asco, and aua evidence-based guideline. *Journal of Clinical Oncology*, 36(34):3411–3430, 2018.

[9] Mark A Perlmutter and Herbert Lepor. Androgen deprivation therapy in the treatment of advanced prostate cancer. *Reviews in urology*, 9 Suppl 1(Suppl 1):S3–S8, 2007.

[10] Christiana Kartsonaki. Survival analysis. *Diagnostic Histopathology*, 22(7):263 – 270, 2016. Mini-Symposium: Medical Statistics.

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

[12] Brendan Richard Van Noorden and Regina Nuzzo Maher. The top 100 papers. *Nature News*.

[13] Ritesh Singh and Keshab Mukhopadhyay. Survival analysis in clinical trials: Basics and must know areas. *Perspectives in clinical research*, 2(4):145–148, Oct-Dec 2011.

[14] World Health Organization. *ICD-10 : international statistical classification of diseases and related health problems / World Health Organization*. World Health Organization Geneva, 10th revision, 2nd ed. edition, 2004.

[15] Linda M French and Neelam Bhambore. Interstitial cystitis/painful bladder syndrome. *American family physician*, 83(10):1175, 2011.

[16] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[17] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. *CoRR*, abs/1605.05101, 2016.

[18] Xuezhe Ma and Eduard H. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *CoRR*, abs/1603.01354, 2016.

[19] Peter Potash, Alexey Romanov, and Anna Rumshisky. Ghostwriter: Using an lstm for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1924, 2015.

[20] Cameron Davidson-Pilon, Jonas Kalderstam, Paul Zivich, Ben Kuhn, Andrew Fiore-Gartland, Luis Moneda, Gabriel, Daniel WIlson, Alex Parij, Kyle Stark, Steven Anton, Lilian Besson, Jona, Harsh Gadgil, Dave Golland, Sean Hussey, Javad Noorbakhsh, Andreas Klintberg, Joanne Jordan, Jeff Rose, Isaac Slavitt, Eric Martin, Eduardo Ochoa, Dylan Albrecht, dhuynh, Denis Zgonjanin, Daniel Chen, Chris Fournier, Arturo, and Andr F. Rendeiro. Camdavidsonpilon/lifelines: v0.20.4, March 2019.

[21] Gilbert Greenwood. The construction and use of an x-ray goniometer. crystal-structure of glyoxaline compounds (with plate i.). *Mineralogical Magazine and Journal of the Mineralogical Society*, 21(112):19, 1926.

[22] N. E. Breslow. Analysis of survival data under the proportional hazards model. *International Statistical Review / Revue Internationale de Statistique*, 43(1):45–57, 1975.

[23] J Martin Bland and Douglas G Altman. The logrank test. *BMJ (Clinical research ed.)*, 328(7447):1073–1073, 05 2004.

[24] Mary L McHugh. The chi-square test of independence. *Biochemia medica*, 23(2):143–149, 06 2013.

[25] Zachary Chase Lipton. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019, 2015.

[26] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122, 2017.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[28] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019.

[29] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[30] Ronald J. Williams David E. Rumelhart, Geoffrey E. Hinton. Learning representatins by back-propagating errors.

[31] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[32] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[33] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[34] Jared L. Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18(1):24, Feb 2018.

[35] Margaux Luck, Tristan Sylvain, Héloïse Cardinal, Andrea Lodi, and Yoshua Bengio. Deep learning for patient-specific kidney graft survival analysis. *CoRR*, abs/1705.10245, 2017.

[36] Eleonora Giunchiglia, Anton Nemchenko, and Mihaela van der Schaar. Rnn-surv: A deep recurrent model for survival analysis. In Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis, editors, *Artificial Neural Networks and Machine Learning –ICANN 2018*, pages 23–32, Cham, 2018. Springer International Publishing.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, Dec 2014.

[38] Charles M. Grinstead and Laurie J. Snell. *Grinstead and Snell's Introduction to Probability*. American Mathematical Society, version dated 4 july 2006 edition, 2006.