Continuous and Discrete-Time Survival Prediction with Neural Networks

Håvard Kvamme HAAVAKVA@MATH.UIO.NO
Ørnulf Borgan BORGAN@MATH.UIO.NO
Department of Mathematics
University of Oslo
P.O. Box 1053 Blindern
0316 Oslo, Norway

Abstract

Application of discrete-time survival methods for continuous-time survival prediction is considered. For this purpose, a scheme for discretization of continuous-time data is proposed by considering the quantiles of the estimated event-time distribution, and, for smaller data sets, it is found to be preferable over the commonly used equidistant scheme. Furthermore, two interpolation schemes for continuous-time survival estimates are explored, both of which are shown to yield improved performance compared to the discrete-time estimates. The survival methods considered are based on the likelihood for right-censored survival data, and parameterize either the probability mass function (PMF) or the discrete-time hazard rate, both with neural networks. Through simulations and study of real-world data, the hazard rate parametrization is found to perform slightly better than the parametrization of the PMF. Inspired by these investigations, a continuous-time method is proposed by assuming that the continuous-time hazard rate is piecewise constant. The method, named PC-Hazard, is found to be highly competitive with the aforementioned methods in addition to other methods for survival prediction found in the literature.

Keywords: survival analysis, neural networks, time-to-event prediction, interpolation, discretization

1. Introduction

Survival analysis, or time-to-event analysis, considers the problem of modeling the time of a future event. A plethora of statistical methods for analyzing right-censored time-to-event data have been developed over the last fifty years or so. Most of these methods, like Cox regression, assume continuous-time models, but methods based on discrete-time models are sometimes used as well. For a review see, e.g., Klein and Moeschberger (2003) for statistical methods based on continuous-time models and Tutz and Schmid (2016) for discrete-time models and methods.

As a result of the rapid development in machine learning and, in particular, neural networks, a number of new methods for time-to-event predictions have been developed in the last few years. This development has benefited from the excellent frameworks for neural network development, such as TensorFlow, PyTorch, Theano, Keras, and CNTK, which have simplified the application of neural networks to existing likelihood-based methodology. Thus, novel methods for time-to-event predictions have been developed based on Cox partial

likelihood (e.g., Katzman et al., 2018; Luck et al., 2017; Yousefi et al., 2017; Kvamme et al., 2019) and the discrete-time survival likelihood (e.g., Lee et al., 2018; Fotso, 2018; Gensheimer and Narasimhan, 2019).

To the best of our knowledge, Lee et al. (2018) were the first to apply neural networks to the discrete-time likelihood for right-censored time-to-event data. Their approach was to parameterize the probability mass function (PMF) of the event times. In statistical survival analysis, it is, however, more common to express the likelihood by the discrete-time hazard rate (see, e.g., Tutz and Schmid, 2016). Gensheimer and Narasimhan (2019) used this form of the likelihood and parameterized the hazard rates with a neural network. In this paper we perform a systematic study of the use of neural nets in conjunction with discrete-time likelihoods for right-censored time-to-event data; in particular, we perform a comparison of methods that parameterize the PMF and the discrete hazard rate.

It is common to apply discrete-time methods as an approximation for continuous-time survival data. To this end one has to perform a discretization of the continuous time scale; a subject that has received little attention in the literature. We consider two discretization schemes, corresponding to equidistant times or equidistant survival probabilities, and conduct a simulation study to better understand the effect of the discretization scheme and the number of time-points used for the discrete-time methods.

Closely related to the discretization of a continuous time scale is the subject of interpolation. A coarse discretization grid has the benefit of reducing the number of parameters in a neural network. But the approximation error that incurs when a discrete-time method is used as an approximation for continuous-time data, becomes smaller with a denser grid. By interpolating the discrete-time survival estimates, it is possible to use a coarser discretization grid without increasing the approximation error. For this reason, two simple interpolation schemes are investigated in this paper; the first assumes constant density functions between the time-points in the discretization grid, and the second assumes constant hazard rates between the grid points. As a modification of the latter method, we also propose a continuous-time method obtained by assuming that the continuous-time hazard rate is piecewise constant, and we compare this method with the aforementioned discrete-time methods with and without interpolation.

The paper is organized as follows. First, in Section 2, we present a summary of related methods for time-to-event predictions. Then, in Section 3, we consider the discrete-time likelihood for right-censored event-times and discuss how the likelihood may be parameterized with neural networks. In Section 4, continuous-time models for time-to-event data are considered, and we discuss how discretization of the continuous time scale enables the application of discrete-time survival methods for continuous-time data. Here we also present the two schemes for interpolating discrete survival functions, and we consider our continuous-time method obtained by assuming piecewise constant hazards. In Section 5, a simulation study is conducted to understand the impact the discretization and interpolation schemes have on the methods, and in Section 6, we compare the methods with existing methods for time-to-event predictions using five real-world data sets. Finally, we summarize and discuss our findings in Section 7. The code for all methods, data sets, and simulations presented in this paper is available at `https://github.com/havakv/pycox`.

2. Related Works

Numerous researchers have used neural networks to parameterize the likelihood for discrete-time survival data. If fact, the two discrete-time survival methods explored in this paper were first proposed by Lee et al. (2018) and Gensheimer and Narasimhan (2019).

DeepHit (Lee et al., 2018) parameterizes the probability mass function (PMF) of the survival distribution and combines the log-likelihood for right-censored data with a ranking loss for improved discriminative performance. The method has been extended to allow for competing risks data. Lee et al. (2018) only used the time-dependent concordance of Antolini et al. (2005) for performance evaluation, and they did not discuss discretization of a continuous time scale. Kvamme et al. (2019) showed that, by only considering concordance, DeepHit has excellent discriminative performance at the cost of poorly calibrated survival estimates.

It is well known in the survival analysis literature that the log-likelihood of discrete-time survival data can be expressed as a Bernoulli log-likelihood of the hazard rates. This enables the use of generalized linear models (GLM) software for fitting survival models parameterized by the hazard rate; for an overview see Tutz and Schmid (2016). Gensheimer and Narasimhan (2019) extended this methodology by parameterizing the hazard rates with a neural network. They showed that their method performs well, both in terms of discrimination and calibration of the survival predictions. However, they did not compare their methodology with methods that parameterize the PMF.

Yu et al. (2011) proposed the multi-task logistic regression, which is a generalization of the binomial log-likelihood, to jointly model a sequence of binary labels representing event indicators. Fotso (2018) later applied this framework to neural networks. We show in Section 3.2.1 that the multi-task logistic regression is, in fact, a PMF parametrization.

Another approach to time-to-event prediction is to consider time as continuous rather than discrete. As a result, the obtained methodology is often not fully parametric. Many of the proposed continuous-time methods are based on Cox proportional hazards model, also called Cox regression model. Estimation in this semi-parametric model is commonly based on Cox partial likelihood (see, e.g., Klein and Moeschberger, 2003). Faraggi and Simon (1995) were the first to parameterize a Cox regression model with a neural network. They were, however, unsuccessful in achieving any improvements over regular Cox regression.

Later extensions of the Cox proportional hazards methodology include new network architectures, larger data sets, and better optimization schemes (Katzman et al., 2018; Ching et al., 2018; Yousefi et al., 2017). As a result, the predictive performance has been improved, in addition to enabling covariates in the form of images (Zhu et al., 2016; Zhu et al., 2017). Luck et al. (2017) combined the negative Cox partial log-likelihood with an isotonic regression loss in an attempt to obtain better discriminative performance. Regardless, their method is still limited by the proportional hazards assumption.

The proportionality assumption of Cox regression model is quite restrictive. Unlike the discrete methods discussed above, none of the aforementioned Cox extensions can estimate survival curves that cross each other. Kvamme et al. (2019) alleviated this restriction by proposing a non-proportional extension of the Cox methodology. This was achieved by approximating the partial log-likelihood with a loss based on case-control sampling.

Random Survival Forest (RSF) by Ishwaran et al. (2008) is a fully non-parametric continuous-time method for right-censored survival data. RSF computes decision trees based on the log-rank test and estimates the cumulative hazard rate with the Nelson-Aalen estimator. The RSF method has become a staple in the predictive survival literature, and it is used as a benchmark in the majority of the work listed in this section.

3. Discrete-Time Models

In this section, we will restrict ourselves to models in discrete time. Then, in Section 4, we will discuss how discrete-time models may be used as approximations of models in continuous time. In the following, we start by a brief introduction to terms in the field of survival analysis, followed by the derivation of the likelihood for right-censored survival data, which is the basis for all methods presented in this paper (and much of survival analysis in general). We will then show how we can parameterize the likelihood with neural networks to obtain the methods proposed by Lee et al. (2018), Gensheimer and Narasimhan (2019), Yu et al. (2011), and Fotso (2018).

3.1 The Discrete-Time Survival Likelihood

Assume that time is discrete with values $0 = \tau_0 < \tau_1 < \dots$, and let $\mathbb{T} = \{\tau_1, \tau_2, \dots\}$ denote the set of positive τ_j's. The time of an event is denoted $T^* \in \mathbb{T}$, and our goal is to model the distribution of such event times, or durations. The probability mass function (PMF) and the survival function for the event times are defined as

$$\begin{aligned} f(\tau_j) &= \mathrm{P}(T^* = \tau_j), \\ S(\tau_j) &= \mathrm{P}(T^* > \tau_j) = \sum_{k>j} f(\tau_k). \end{aligned} \tag{1}$$

In survival analysis, models are often expressed in terms of the hazard function rather than the PMF. For discrete time, the hazard is defined as

$$h(\tau_j) = \mathrm{P}(T^* = \tau_j \,|\, T^* > \tau_{j-1}) = \frac{f(\tau_j)}{S(\tau_{j-1})} = \frac{S(\tau_{j-1}) - S(\tau_j)}{S(\tau_{j-1})},$$

and it follows that

$$f(\tau_j) = h(\tau_j)\, S(\tau_{j-1}), \tag{2}$$

$$S(\tau_j) = [1 - h(\tau_j)]\, S(\tau_{j-1}). \tag{3}$$

Note further, that from (3) it follows that the survival function can be expressed as

$$S(\tau_j) = \prod_{k=1}^{j} [1 - h(\tau_k)]. \tag{4}$$

In most studies, we do not observe all event times. For some individuals, we will only observe a right-censored duration. So to allow for censoring, we let $C^* \in \mathbb{T}_C =$

$\{\tau_1, \tau_2, \ldots, \tau_m\}$ be a right-censoring time. In the same manner as for the event time, the censoring-time has the PMF and survival function

$$f_{C^*}(\tau_j) = \mathrm{P}(C^* = \tau_j),$$
$$S_{C^*}(\tau_j) = \mathrm{P}(C^* > \tau_j).$$

T^* and C^* are typically not observed directly, but instead, we observe a potentially right-censored duration T and an event indicator D given by

$$T = \min\{T^*, C^*\},$$
$$D = \mathbb{1}\{T^* \leq C^*\}.$$

We here follow the common convention in survival analysis that when an event and censoring time coincide, we observe the occurrence of the event. Note that, as $C^* \leq \tau_m$, we are not able to observe event times T^* larger than τ_m. Hence, we are restricted to model the distribution of the event times in $\mathbb{T}_C$.

Now, assuming that T^* and C^* are independent, we can derive the likelihood function for right-censored survival data. To this end, note that, for $t \in \mathbb{T}_{\mathbb{C}}$ and $d \in \{0, 1\}$, we have that

$$\begin{aligned}
\mathrm{P}(T = t, D = d) &= \mathrm{P}(T^* = t,\, C^* \geq t)^d \, \mathrm{P}(T^* > t,\, C^* = t)^{1-d} \\
&= \left[\mathrm{P}(T^* = t)\, \mathrm{P}(C^* \geq t)\right]^d \left[\mathrm{P}(T^* > t)\, \mathrm{P}(C^* = t)\right]^{1-d} \\
&= \left[f(t)\, (S_{C^*}(t) + f_{C^*}(t))\right]^d \left[S(t) f_{C^*}(t)\right]^{1-d} \\
&= \left[f(t)^d \, S(t)^{1-d}\right] \left[f_{C^*}(t)^{1-d} \, (S_{C^*}(t) + f_{C^*}(t))^d\right].
\end{aligned}$$

Now, it is common to assume that $f(t)$ and $f_{C^*}(t)$ have no parameters in common. Then we can consider, separately, the contribution to the likelihood of the event time distribution and the censoring distribution. We are typically only interested in modeling the distribution of the event times, in which case, for individual i, we obtain the likelihood contribution

$$L_i = f(t_i)^{d_i} S(t_i)^{1-d_i}. \tag{5}$$

If we have data for n independent individuals, each with covariates $\mathbf{x}_i$, observed time t_i, and event indicator d_i, we can fit models by minimizing the mean negative log-likelihood

$$\text{loss} = -\frac{1}{n} \sum_{i=1}^{n} \left(d_i \log[f(t_i \,|\, \mathbf{x}_i)] + (1 - d_i) \log[S(t_i \,|\, \mathbf{x}_i)]\right). \tag{6}$$

A useful alternative to the loss function (6) is obtained by rewriting it in terms of the discrete hazards. To this end, let $\kappa(t) \in \{0, \ldots, m\}$ define the index of the discrete time t, meaning $t = \tau_{\kappa(t)}$. Using (2), (3), and (4), we can then rewrite the likelihood contribution (5)

as

$$\begin{aligned} L_i &= f(t_i)^{d_i}\, S(t_i)^{1-d_i} \\ &= [h(t_i)\, S(\tau_{\kappa(t_i)-1})]^{d_i}\, [(1-h(t_i))\, S(\tau_{\kappa(t_i)-1})]^{1-d_i} \\ &= h(t_i)^{d_i}\, [1-h(t_i)]^{1-d_i}\, S(\tau_{\kappa(t_i)-1}) \\ &= h(t_i)^{d_i}\, [1-h(t_i)]^{1-d_i} \prod_{j=1}^{\kappa(t_i)-1} [1-h(\tau_j)]. \end{aligned}$$

With this formulation, the mean negative log-likelihood in (6) can be rewritten as

$$\begin{aligned} \text{loss} &= -\frac{1}{n}\sum_{i=1}^{n}\left(d_i \log[h(t_i \,|\, \mathbf{x}_i)] + (1-d_i)\log[1-h(t_i \,|\, \mathbf{x}_i)] + \sum_{j=1}^{\kappa(t_i)-1} \log[1-h(\tau_j \,|\, \mathbf{x}_i)] \right) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{\kappa(t_i)} \left(y_{ij}\log[h(\tau_j \,|\, \mathbf{x}_i)] + (1-y_{ij})\log[1-h(\tau_j \,|\, \mathbf{x}_i)] \right). \end{aligned} \tag{7}$$

Here, $y_{ij} = \mathbb{1}\{t_i = \tau_j,\, d_i = 1\}$, meaning $\mathbf{y}_i = (y_{i1}, \ldots, y_{im})$ is a vector of zeros with a single 1 at the event index $\kappa(t_i)$ when t_i corresponds to an observed event ($d_i = 1$). We recognize this as the negative log-likelihood for Bernoulli data, or binary cross-entropy, a useful discovery first noted by Brown (1975).

With the two loss functions (6) and (7), we can now make survival models by parameterizing the PMF or the hazard function and minimizing the corresponding loss. For classical statistical models, these approaches are equivalent and have been used to obtain maximum likelihood estimates for the parameters in the PMF/hazard function; see, e.g., Tutz and Schmid (2016) for a review. We will, however, not consider classical maximum likelihood estimates, but focus on the part of the literature that fit neural networks for the purpose of time-to-event prediction, in which case the two loss functions may give different results.

3.2 Parameterization with Neural Networks

In the previous section, we saw that the survival likelihood can be expressed in terms of the PMF or the hazard function. In the following, we will describe how to use this to create survival methods by parameterizing the PMF or hazard with neural networks. In theory, as both approaches minimize the same negative log-likelihood, the methods should yield similar results. But as neural networks are quite complex, this might not be the case in practice.

First, considering the hazard parametrization of the likelihood, let $\phi(\mathbf{x}) \in \mathbb{R}^m$ represent a neural network that takes the covariates $\mathbf{x}$ as input and gives m outputs, each corresponding to a discrete time-point τ_j, i.e., $\phi(\mathbf{x}) = \{\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x})\}$. As the discrete hazards are (conditional) probabilities, we require $h(\tau_j \,|\, \mathbf{x}) \in [0, 1]$. This can be achieved by applying the logistic function (sigmoid function) to the neural network

$$h(\tau_j \,|\, \mathbf{x}) = \frac{1}{1+\exp[-\phi_j(\mathbf{x})]}.$$

We can estimate the hazard function by minimizing the loss (7), and survival estimates can be obtained from (4). To the best of our knowledge, this method was first proposed by

Gensheimer and Narasimhan (2019). However, if one considers $\phi_j(\mathbf{x})$ an arbitrary parametric function of $\mathbf{x}$, the approach is well known in the survival literature and seems to have been first addressed by Cox (1972) and Brown (1975); see also Allison (1982). The book by Tutz and Schmid (2016) gives a review of the approach.

The implementation we use in the experiments in Sections 5 and 6 differs slightly from that of Gensheimer and Narasimhan (2019), as it was found to be more numerically stable (see Appendix C). In this paper, we will refer to the method as *Logistic-Hazard*, as coined by Brown (1975) (one can also find the term Logistic Discrete Hazard used in the statistical literature). Gensheimer and Narasimhan (2019), on the other hand, referred to it as *Nnet-survival*, but to be better able to contrast this method to the other methods presented in this paper, we will instead use the more descriptive *Logistic-Hazard*.

We can obtain a survival model by parameterizing the PMF in a similar manner to the Logistic-Hazard method. As for the hazards, the PMF represents probabilities $f(\tau_j \,|\, \mathbf{x}) \in [0, 1]$, but, contrary to the conditional probabilities that define the hazard, we now require the PMF to sum to 1. As we only observe event times in $\mathbb{T}_C$, we fulfill this requirement indirectly through the probability of surviving past τ_m, i.e.,

$$\sum_{k=1}^{m} f(\tau_k \,|\, \mathbf{x}) + S(\tau_m \,|\, x) = 1. \tag{8}$$

Now, again with $\phi(\mathbf{x}) \in \mathbb{R}^m$ denoting a neural network, the PMF can be expressed as

$$f(\tau_j \,|\, \mathbf{x}) = \frac{\exp[\phi_j(\mathbf{x})]}{1 + \sum_{k=1}^{m} \exp[\phi_k(\mathbf{x})]}, \qquad \text{for } j = 1, \ldots, m. \tag{9}$$

Note that (9) is equivalent to the softmax function with a fixed $\phi_{m+1}(\mathbf{x}) = 0$. Alternatively, one could let $\phi_{m+1}(\mathbf{x})$ vary freely, something that is quite common in machine learning, but we chose to follow the typical conventions in statistics. By combining (1) and (8), we can express the survival function as

$$\begin{aligned} S(\tau_j \,|\, \mathbf{x}) &= \sum_{k=j+1}^{m} f(\tau_k \,|\, \mathbf{x}) + S(\tau_m \,|\, \mathbf{x}), \qquad \text{for } j = 1, \ldots, m-1, \\ S(\tau_m \,|\, \mathbf{x}) &= \frac{1}{1 + \sum_{k=1}^{m} \exp[\phi_k(\mathbf{x})]}. \end{aligned} \tag{10}$$

Now, let $\sigma_j[\phi(\mathbf{x})]$, for $j = 1, \ldots, m+1$, denote the softmax in (9), meaning $\sigma_{m+1}[\phi(\mathbf{x})] = S(\tau_m \,|\, \mathbf{x})$. Notice the similarities to classification with $m+1$ classes, as we are essentially classifying whether the event is happening at either time $\tau_1, \ldots, \tau_m$ or later than τ_m. However, due to censoring, the likelihood is *not* the cross-entropy. Instead, by inserting (9) and (10) into (6), we get the mean negative log-likelihood

$$\text{loss} = -\frac{1}{n} \sum_{i=1}^{n} \left(d_i \log[\sigma_{\kappa(t_i)}(\phi(\mathbf{x}_i))] + (1 - d_i) \log \left[\sum_{k=\kappa(t_i)+1}^{m+1} \sigma_k(\phi(\mathbf{x})) \right] \right), \tag{11}$$

where $\kappa(t_i)$ still denotes the duration index of individual i's event time, i.e., $t_i = \tau_{\kappa(t_i)}$. This is essentially the same negative log-likelihood as presented by Lee et al. (2018), but with

only one type of event. Also, note that contrary to the work by Lee et al. (2018) the negative log-likelihood in (11) allows for survival past time τ_m. Some numerical improvements of the implementation are addressed in Appendix C. We will refer to this method simply by *PMF* as this term is unambiguously discrete, contrary to the term *hazard* which is used both for discrete and continuous time.

3.2.1 Multi-Task Logistic Regression

Multi-task logistic regression, proposed by Yu et al. (2011), provides a generalization of the binomial log-likelihood to jointly model the sequence of binary labels $Y_j = \mathbb{1}\{T^* \leq \tau_j\}$. This means that $Y = (y_1, \ldots, y_m)$ is a sequence with zeros for every time τ_j up to the event time, followed by one's, e.g., $(0, \ldots, 0, 1, \ldots 1)$. Then

$$\mathrm{P}(Y = (y_1, \ldots, y_m) \,|\, \mathbf{x}) = \frac{\exp\left[\sum_{k=1}^m y_k \psi_k(\mathbf{x})\right]}{1 + \sum_{k=1}^m \exp\left[\sum_{l=k}^m \psi_l(\mathbf{x})\right]}. \tag{12}$$

Yu et al. (2011) only consider linear predictors $\psi_k(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\beta}_k$, but this was extended to a neural network by Fotso (2018). The parameters of $\psi_k(\mathbf{x})$ are found by minimizing the negative log-likelihood in (6).

As $f(\tau_j \,|\, \mathbf{x}) = \mathrm{P}(Y = (y_1, \ldots, y_m) \,|\, \mathbf{x})$, where $y_k = \mathbb{1}\{k \geq j\}$, the expression in (12) can be written as

$$f(\tau_j \,|\, \mathbf{x}) = \frac{\exp\left[\sum_{k=j}^m \psi_k(\mathbf{x})\right]}{1 + \sum_{k=1}^m \exp\left[\sum_{l=k}^m \psi_l(\mathbf{x})\right]} = \frac{\exp[\phi_j(\mathbf{x})]}{1 + \sum_{k=1}^m \exp[\phi_k(\mathbf{x})]},$$

where $\phi_j(\mathbf{x}) = \sum_{k=j}^m \psi_k(\mathbf{x})$. Hence, the multi-task logistic regression is equivalent to the PMF in (9), but where $\phi_j(\mathbf{x})$ is the (reverse) cumulative sum of the output of the network $\psi(\mathbf{x}) \in \mathbb{R}^m$. To the extent of our knowledge, there are no benefits to this extra cumulative sum. Instead, it simply requires unnecessary computations, and, for large m, it can cause numerical instabilities. Hence, we will not consider this method further.

4. Continuous-Time Models

In the following, we no longer consider the time scale to be discrete, but instead consider continuous-time models, where $T^*, C^* > 0$, and we let $T = \min\{T^*, C^*\}$ and $D = \mathbb{1}\{T^* \leq C^*\}$ be as before. Let τ denote the maximum possible value of C^*, meaning $P(C^* \leq \tau) = 1$. Hence, a potentially right-censored observation T is in the interval $T \in (0, \tau]$. Instead of a PMF, we now have the density function $f(t)$ and the continuous-time survival function

$$S(t) = \mathrm{P}(T^* > t) = \int_t^\tau f(z)\, dz + S(\tau).$$

Furthermore, the continuous-time hazard rate is a non-negative function of the time (no longer restricted to $[0, 1]$),

$$h(t) = \frac{f(t)}{S(t)} = \lim_{\Delta t \to 0} \frac{\mathrm{P}(t \leq T^* < t + \Delta t \,|\, T^* \geq t)}{\Delta t}. \tag{13}$$

As a result, we can express the survival function in terms of the cumulative hazard $H(t) = \int_{\tau_0}^{t} h(z)\, dz$,

$$S(t) = \exp[-H(t)]. \tag{14}$$

This yields the continuous-time version of the likelihood contribution in (5),

$$L_i = f(t_i)^{d_i}\, S(t_i)^{1-d_i} = h(t_i)^{d_i}\, S(t_i) = h(t_i)^{d_i}\, \exp[-H(t_i)]. \tag{15}$$

The derivation of L_i follows the same steps as the derivation of the discrete likelihood contribution (5), only with density functions instead of probability mass functions.

In what follows, we will first discuss how we can apply the discrete-time methods from Section 3.2 for continuous-time data. We will here address how time can be discretized to fit the discrete-time model formulation, and how to interpolate an estimated discrete survival function for continuous-time predictions. Then, we will propose a new continuous-time method by assuming that the hazard in (13) is piecewise constant, which we call *PC-Hazard*.

4.1 Discretization of Durations

Both the PMF and Logistic-Hazard methods require time to be discrete on the form $0 = \tau_0 < \tau_1 < \cdots < \tau_m$. Hence, to apply the methods to continuous-time data, we need to perform some form of discretization of the time scale (also, for inherently discrete event times, we might want to coarsen the discrete time scale to obtain a smaller subset of τ_j's, as this will reduce the number of parameters in the neural networks). Possibly the most obvious way to discretize time is to make an equidistant grid in $[0, \tau]$ with m grid points. An alternative, that we explore in this paper, is to make a grid based on the distribution of the event times. By estimating the survival function $S(t)$ with the Kaplan-Meier estimator, we obtain a general trend of event times. With $\hat{S}(t)$ denoting the Kaplan-Meier survival estimates, we can make a grid from the quantiles of the estimates, $1 = \hat{S}(0) = \zeta_0 > \zeta_1 > \cdots > \zeta_m = \hat{S}(\tau)$. We will assume that each interval has the same decrease in the survival estimate, so that $\zeta_j - \zeta_{j+1} = (1 - \hat{S}(\tau))/m$. The corresponding duration grid, $\tau_1 < \cdots < \tau_m$, is found by solving $\hat{S}(\tau_j) = \zeta_j$. We will then obtain a more dense grid in intervals with more events, and less dense grid in intervals with fewer events. This is illustrated in Figure 1, where we can see that the grid becomes coarser as the slope of the survival curve becomes less steep.

The discrete-time methods assume that all events and censorings occur at the τ_j's, so, when performing the discretization, we move all event times in an interval to the end of that interval while censored times are moved to the end of the previous interval. This means that for $\tau_{j-1} < T_i \leq \tau_j$, we replace T_i by τ_j if $D_i = 1$, and by τ_{j-1} if $D_i = 0$. Our reason for this choice is that this is typically how event times are recorded. Consider a study where we are only able to make observations at times $\tau_1 < \tau_2 < \cdots < \tau_m$. For a censored observation, τ_{j-1} is the last point in time where the individual was recorded alive, while for an observed event, τ_j is the first duration for which the individual was recorded with the event.

As a side note, an alternative way to obtain the discrete loss in (7) is by assuming continuous event times in defined intervals $[\tau_j, \tau_{j+1})$ and censorings that only occur at the beginning or end of the intervals (see, e.g., Tutz and Schmid, 2016). This justifies the use of this loss for continuous-time data grouped in intervals.

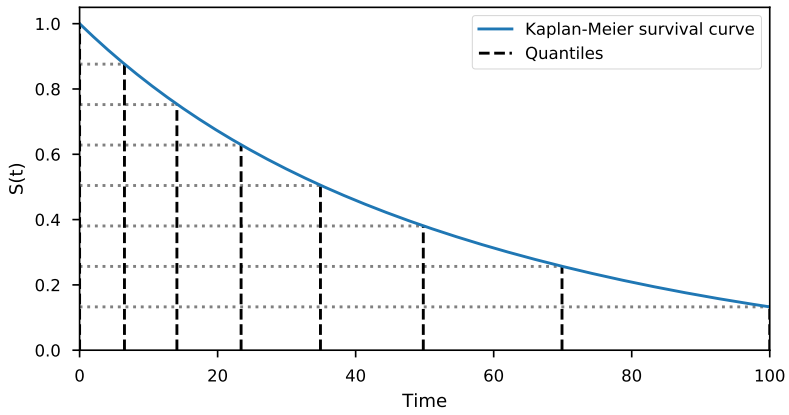

Figure 1: Illustration of the Kaplan-Meier based discretization scheme. The quantiles of the Kaplan-Meier curve are used as the grid points.

4.2 Interpolation for Continuous-Time Predictions

When discrete-time survival methods are applied to continuous-time data, as described in Section 4.1, the survival estimates become a step function with steps at the grid points (blue line in Figure 2). Consequently, for coarser grids, it might be beneficial to interpolate the discrete survival estimates. In this regard, we propose two simple interpolation schemes that fulfill the monotonicity requirement of the survival function. The first assumes that the probability density function is constant in each time interval $(\tau_{j-1}, \tau_j]$, while the second scheme assumes constant hazards in each time interval. This corresponds to piecewise linear and piecewise exponential survival estimates, respectively, and we will, therefore, refer to the schemes as *constant density interpolation* (CDI) and *constant hazard interpolation* (CHI). See Figure 2 for an illustration of the two schemes and the discrete survival estimates.

4.2.1 Constant Density Interpolation (CDI)

For a continuous time $t \in (\tau_{j-1}, \tau_j]$, linear interpolation of the discrete survival function takes the form

$$S(t) = S(\tau_{j-1}) + [S(\tau_j) - S(\tau_{j-1})] \, \frac{t - \tau_{j-1}}{\Delta\tau_j},$$

where $\Delta\tau_j = \tau_j - \tau_{j-1}$. This means that the density function $f(t)$ is constant in this interval

$$f(t) = -S'(t) = \frac{S(\tau_{j-1}) - S(\tau_j)}{\Delta\tau_j}. \tag{16}$$

Let us now rewrite the expression of the survival function as

$$S(t) = \frac{S(\tau_{j-1})\tau_j - S(\tau_j)\tau_{j-1}}{\Delta\tau_j} - \frac{S(\tau_{j-1}) - S(\tau_j)}{\Delta\tau_j} \, t = \alpha_j - \beta_j \, t,$$

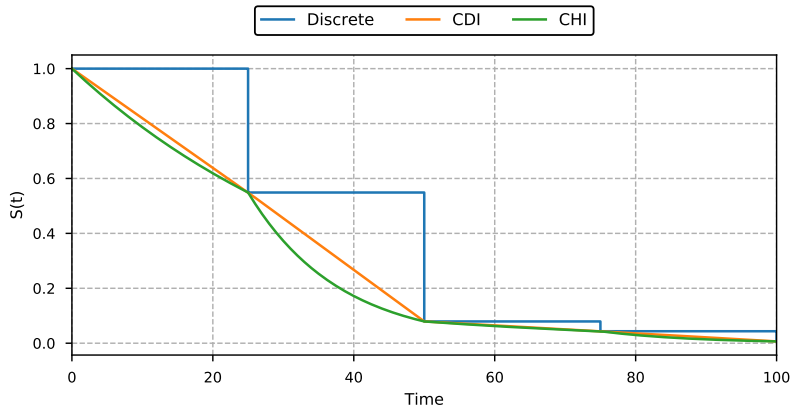

Figure 2: Survival estimates by a discrete model (e.g., PMF or Logistic-Hazard) for 5 grid points. The three lines represent the discrete survival estimates and the two interpolation schemes in Section 4.2: The constant density interpolation (CDI) and constant hazard interpolation (CHI).

where both α_j and β_j are non-negative. Using that the density is $f(t) = -S'(t) = \beta_j$, we get a simple expression for the hazard function (13)

$$h(t) = \frac{f(t)}{S(t)} = \frac{\beta_j}{\alpha_j - \beta_j\, t}.$$

Hence, we see that linear interpolation of the survival function corresponds to a constant density function and an increasing hazard rate throughout the interval.

4.2.2 Constant Hazard Interpolation (CHI)

The following scheme assumes constant hazard in each interval, which corresponds to linear interpolation of the cumulative hazard function. For a continuous time $t \in (\tau_{j-1}, \tau_j]$, the interpolated cumulative hazard is then

$$H(t) = H(\tau_{j-1}) + [H(\tau_j) - H(\tau_{j-1})]\ \frac{t - \tau_{j-1}}{\Delta\tau_j}.$$

This means that the hazard function is constant in this interval

$$h(t) = H'(t) = \frac{H(\tau_j) - H(\tau_{j-1})}{\Delta\tau_j} = \eta_j,$$

and from (14), we obtain the piecewise exponential survival function

$$S(t) = \exp[-H(t)] = \exp\left[-\eta_j\,(t - \tau_{j-1})\right]\ S(\tau_{j-1}).$$

Finally, the density is

$$f(t) = h(t)\, S(t) = \eta_j\, S(t),$$

showing that the density is decreasing throughout the interval.

Summarizing the two interpolation methods, CDI assumes that the events are spread evenly in the interval, while the CHI assumes that there are more events at the beginning of the interval. Correspondingly, the CDI assumes that the longer an individual "survives" in an interval, the higher the risk becomes of experiencing an event in the next immediate moment (increasing hazard), contrary to the CHI which assumes that this risk is constant. In fact, in the next section, we will propose a new method with the same assumptions as CHI, but contrary to the CHI, we can train it on continuous-time data.

4.3 A Piecewise Constant Continuous-Time Hazard Parametrization

We now propose a continuous-time method by parameterizing the hazards in (15). As for the CHI in Section 4.2.2, we will let the continuous-time hazard be piecewise constant. Disregarding the neural networks, this model was first proposed by Holford (1976), and further developed by Friedman (1982) who found that piecewise constant hazards yields a likelihood proportional to that of a Poisson likelihood; see Appendix B for details.

Consider a partition of the time scale $0 = \tau_0 < \tau_1 < \cdots < \tau_m = \tau$, and let $\kappa(t)$ denote the interval index of time t such that $t \in (\tau_{\kappa(t)-1}, \tau_{\kappa(t)}]$ (this is slightly different from the discrete case where we had $t = \tau_{\kappa(t)}$). If we assume that the hazard is constant within each interval, we can express the hazard as the step function

$$h(t) = \eta_{\kappa(t)},$$

for a set of non-negative constants $\{\eta_1, \ldots, \eta_m\}$. For $\Delta\tau_j = \tau_j - \tau_{j-1}$, we can now express the cumulative hazard as

$$\begin{aligned} H(t) &= \int_0^t h(z)dz \\ &= \left(\sum_{j=1}^{\kappa(t)-1} \int_{\tau_{j-1}}^{\tau_j} h(z)dz \right) + \int_{\kappa(t)-1}^{t} h(z)dz \\ &= \left(\sum_{j=1}^{\kappa(t)-1} \eta_j \, \Delta\tau_j \right) + \eta_{\kappa(t)} \, (t - \tau_{\kappa(t)-1}). \end{aligned}$$

Inserting this into (15) yields the likelihood contribution for individual i

$$L_i = h(t_i)^{d_i} \, \exp\left[-H(t_i)\right] = {\eta_{\kappa(t_i)}}^{d_i} \, \exp\left[-\eta_{\kappa(t_i)} \, (t - \tau_{\kappa(t_i)-1})\right] \prod_{j=1}^{\kappa(t_i)-1} \exp\left[-\eta_j \, \Delta\tau_j\right]. \quad (17)$$

What remains is to parameterize the hazard with a neural network. However, to avoid passing all the τ_j's to the loss function, we let the network instead parameterize the quantities

$\tilde{\eta}_j = \eta_j \, \Delta\tau_k$. This allows us to rewrite the likelihood contribution as

$$L_i = \left(\frac{\tilde{\eta}_{\kappa(t_i)}}{\Delta\tau_{\kappa(t_i)}} \right)^{d_i} \exp\left[-\tilde{\eta}_{\kappa(t_i)} \, \rho(t_i) \right] \prod_{j=1}^{\kappa(t_i)-1} \exp\left[-\tilde{\eta}_j \right]$$
$$\propto {\tilde{\eta}_{\kappa(t_i)}}^{d_i} \exp\left[-\tilde{\eta}_{\kappa(t_i)} \, \rho(t_i) \right] \prod_{j=1}^{\kappa(t_i)-1} \exp\left[-\tilde{\eta}_j \right],$$

where

$$\rho(t) = \frac{t - \tau_{\kappa(t)-1}}{\Delta\tau_{\kappa(t)}}, \tag{18}$$

is the proportion of interval $\kappa(t)$ at time t.

As before, let $\phi(\mathbf{x}) \in \mathbb{R}^m$ denote a neural network. To ensure that $\tilde{\eta}_j$ is non-negative, we use the softplus function

$$\tilde{\eta}_j = \log(1 + \exp[\phi_j(\mathbf{x})]). \tag{19}$$

Our model can now be fitted by minimizing the mean negative log-likelihood

$$\text{loss} = -\frac{1}{n} \sum_{i=1}^{n} \left(d_i \, \log \tilde{\eta}_{\kappa(t_i)}(\mathbf{x}_i) - \tilde{\eta}_{\kappa(t_i)}(\mathbf{x}_i) \, \rho(t_i) - \sum_{j=1}^{\kappa(t_i)-1} \tilde{\eta}_j(\mathbf{x}_i) \right),$$

and estimates for the survival function can be obtained by

$$S(t \,|\, \mathbf{x}) = \exp[-H(t \,|\, \mathbf{x})] = \exp[-\tilde{\eta}_{\kappa(t)}(\mathbf{x}) \, \rho(t)] \prod_{j=1}^{\kappa(t)-1} \exp[-\tilde{\eta}_j(\mathbf{x})], \tag{20}$$

where $\rho(t)$ is given by (18). We will refer to this method as the *piecewise constant hazard* method, or *PC-Hazard*. Even though this is a continuous-time method, we still need to decide the set of τ_j's that define the intervals. Therefore, the discretization techniques discussed in Section 4.1 are also relevant for this method.

Comparing the PC-Hazard to the Logistic-Hazard method with survival estimates interpolated with CHI (Section 4.2.2), we see that the only difference is in the loss function, as both PC-Hazard and CHI have piecewise constant hazards. In other words, the two methods both use (20) to obtain survival estimates, but they have different estimates for the $\tilde{\eta}_j$'s as the PC-Hazard use the observed continuous event times and censoring times, while Logistic-Hazard discretizes the times to a predefined set of τ_j's as described in Section 4.1.

5. Simulations

To get a better understanding of the methodologies discussed in Sections 3 and 4, we perform a simulation study where we vary the size of the training sets, the discretization scheme, and the number of grid points used for discretization. Gensheimer and Narasimhan (2019) performed a similar study to evaluate the effect of discretization on their Logistic-Hazard

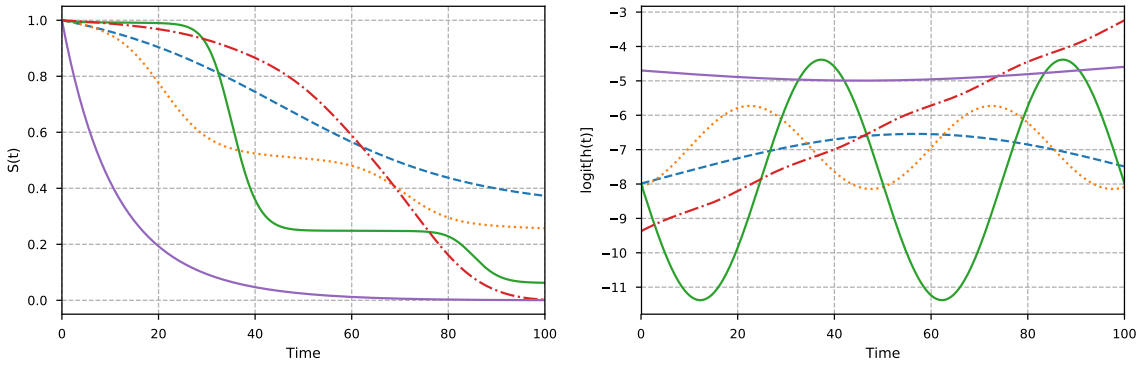

Figure 3: Examples from the simulation study in Section 5. The left figure shows examples of 5 simulated survival curves, while the right figure shows the corresponding logit hazards.

method with the conclusion that there were no differences in performance. However, their simulations were quite simple (only one binary covariate), their only performance metric was the Harrell Jr et al. (1982) concordance at 1-year survival, and they did not include any interpolation of the survival estimates. For this reason, we find that further investigation is warranted.

We generate simulated survival times by sequentially sampling from discrete-time hazards defined on a fine grid of time points. The hazards are specified through their logit transforms, as this enables us to use functions in $\mathbb{R}$ while still obtaining hazards in $[0, 1]$. The logit hazards, $g(t) = \text{logit}[h(t)]$, are linear combinations of the three functions

$$
\begin{aligned}
g_{\text{sin}}(t \,|\, \mathbf{x}) &= \gamma_1 \sin(\gamma_2[t + \gamma_3]) + \gamma_4, \\
g_{\text{con}}(t \,|\, \mathbf{x}) &= \gamma_5, \\
g_{\text{acc}}(t \,|\, \mathbf{x}) &= \gamma_6 \cdot t - 10,
\end{aligned}
$$

with additional parameters γ_7, γ_8, and γ_9 determining the linear combination. As described in Appendix A, each γ_k is a function of five covariates, meaning we have a total of 45 covariates. We let the discrete time scale consist of 1,000 equidistant points between 0 and 100 (i.e. $\tau_0 = 0$, $\tau_1 = 0.1$, $\tau_2 = 0.2$, $\ldots$, $\tau_{1000} = 100$). Knowing the hazards, the true survival function can be obtained with (4), $S(\tau_j \,|\, \mathbf{x}) = \prod_{k=1}^{j}[1 - h(\tau_k \,|\, \mathbf{x})]$. In Figure 3 we show five examples of logit hazard functions and their corresponding survival functions. Note that even though we simulate our data using a discrete-time model, the time-grid is so fine that this mimics simulation from a continuous-time model. The full details of this simulation study are given in Appendix A.

We created three training sets of size 3,000, 10,000, and 50,000, a validation set of size 10,000 (for hyperparameter tuning) and a test set of size 100,000. For the training and validation sets, we included a censoring distribution with constant hazard resulting in 37 % censoring. The full uncensored test set is used for evaluation. For the discretization of the time scale, we applied both the equidistant scheme and the Kaplan-Meier quantiles, each with 5, 25, 100, and 250 grid points. The neural networks were all ReLU-nets with batch

normalization and dropout between each layer, with all layers consisting of the same number of nodes. We performed a hyperparameter grid search over 1, 2, 4, and 8 layers; 16, 64, and 256 nodes; and dropout of 0 and 0.5. Each net was trained with batch size of 256 and the AdamWR optimizer (Loshchilov and Hutter, 2019) with cycle length 1, where, at each restart, the cycle length was doubled and the learning rate was multiplied by 0.8. Learning rates were found using the methods proposed by Smith (2017). The hyperparameter tuning was repeated 10 times, giving 10 fitted models for each combination of method, grid size, discretization scheme, and training set size.

5.1 Comparison of Discrete-Time Methods

We start by comparing the two discrete methods from Section 3.2, that parameterize the PMF and the discrete-time hazards. We refer to them as PMF and Logistic-Hazard, respectively. For evaluation, we use the time-dependent concordance (Antolini et al., 2005), in addition to the MSE between the survival estimates and the true survival function at all 1,000 time points $\tau_1, \ldots, \tau_{1000}$

$$\text{MSE} = \frac{1}{100,000} \sum_{i=1}^{100,000} \frac{1}{1,000} \sum_{j=1}^{1,000} \Big(\hat{S}(\tau_j \,|\, \mathbf{x}_i) - S_i(\tau_j) \Big)^2. \tag{21}$$

Here $\hat{S}(\tau_j \,|\, \mathbf{x}_i)$ and $S_i(\tau_j)$ are the estimated and true survival functions, respectively, for individual i (in the test set) at time τ_j. So, in this regard, the discrete-time survival estimates are represented by step functions, as illustrated in Figure 2.

In Figure 4 we plot the median test scores of the two methods versus the grid size used for discretization. The numbers above each plot give the size of the training set used to fit the methods. The full lines represent equidistant grids, while the dotted lines are from grids obtained with quantiles from Kaplan-Meier survival estimates. We have also included the constant hazard interpolation (CHI) of the survival estimates from the Logistic-Hazard method (see Section 4.2.2).

It is evident that smaller discretization grids are better for the smaller training sets, while larger training sets allow for larger grids. This is reasonable as the smaller grids result in fewer parameters in the neural networks. Nevertheless, the smallest grid of size 5 seems to only work well for the interpolated estimates, and very poorly for the discrete estimates. We also note that the discretization grids from Kaplan-Meier quantiles seem to give slightly better scores than the equidistant grids. Comparing the discrete survival estimates from Logistic-Hazard (blue lines) with the CHI estimates (orange lines), we see that the two lines overlap for larger grid sizes. This is expected as the effect of interpolation decreases as the grids become denser.

In general, the PMF method does not perform as well as the Logistic-Hazard, though the difference is rather small. Also, while the interpolated estimates yield better results for most grid configurations, the best scores are almost identical. This means that the interpolated estimates have more stable performance, but with careful tuning of the discretization scheme, similar performance can be obtained with the discrete estimates.

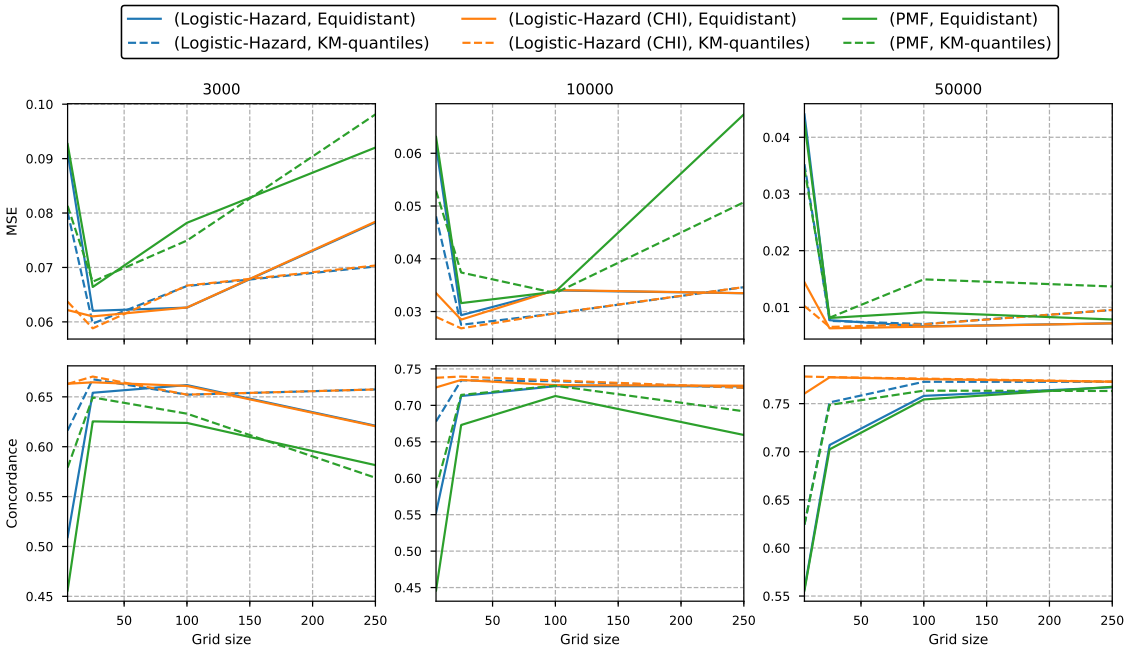

Figure 4: Median MSE and concordance for each grid size in the simulation study in Section 5.1. The number above each plot gives the size of the training set. The full lines use an equidistant grid, while the dotted lines use Kaplan-Meier quantiles for discretization. Note that the plots are not on the same scale.

5.2 Comparison of Interpolation Schemes

In the following, we compare the interpolation schemes for the discrete-time hazard method Logistic-Hazard. The experiments are not shown for the PMF method as the results are very similar.

In Section 4.2 we presented two methods for interpolation of discrete survival estimates. The first assume constant density in each interval (denoted CDI for constant density interpolation), while the second assumes constant hazard in each interval (denoted CHI for constant hazard interpolation). In our simulation study, we have four grid sizes and two discretization schemes. As the hyperparameter tuning was repeated 10 times this gives 80 fitted models for each method on each data set. In Figure 5, we plot the scores of these 80 models sorted from best to worst, as this both tells us the best performance, in addition to the stability of the methods. The figure contains results from the discrete survival estimates (Logistic-Hazard), the constant density interpolation (CDI), and the constant hazard interpolation (CHI).

Clearly, there is almost no difference in performance between the two interpolation schemes, while the discrete estimates have slightly worse best-case performance and much worse worst-case performance. In fact, the only difference between the two interpolation schemes is that that CDI estimates give slightly better MSE while the CHI estimates give slightly better concordance. In this regard, we will in the further simulations only include

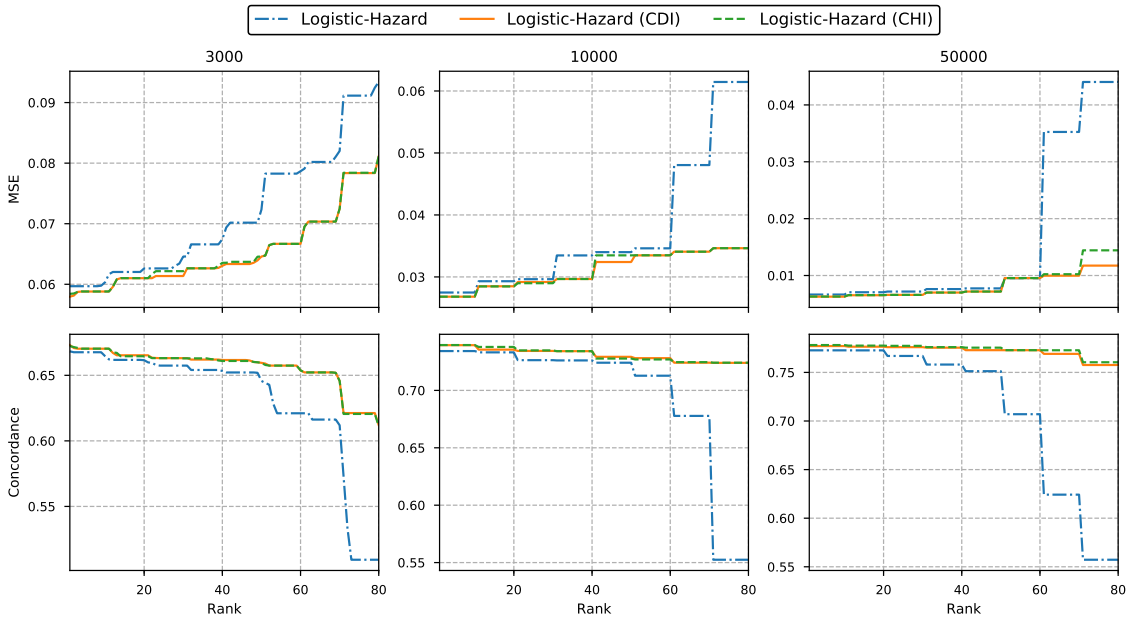

Figure 5: MSE and concordance from the simulation study in Section 5.2. The scores are plotted from best to worst. The number above each plot gives the size of the training set. Note that the plots are not on the same scale.

the CHI estimates, as they have the same assumption as the continuous-time PC-Hazard method, simplifying the comparison between the methods.

5.3 Comparison with PC-Hazard

Finally, we compare the previous methods with our proposed continuous-time hazard method from Section 4.3, PC-Hazard. In Figure 6 we plot the MSE and concordance for the interpolated Logistic-Hazard (CHI) method, and the continuous-time PC-Hazard method. First, we notice that PC-Hazard does better for the smallest grids with only five grid points, while Logistic-Hazard (CHI) typically performs best with 25 grid points. Also, in terms of MSE, PC-Hazard does the best for the smallest training set, while Logistic-Hazard (CHI) does better for the two larger training sets. In terms of concordance, PC-Hazard performs the best for the smallest and largest data sets. All differences are however quite small. On the other hand, the Logistic-Hazard (CHI) estimates do better for a variety of grid configurations, showing that it is less sensitive to the discretization than the PC-Hazard method. Finally, we again see that the Kaplan-Meier quantiles seem to give slightly better performance than the equidistant discretization grids.

In Figure A.1 in Appendix A, we have included a plot of the same type as Figure 5 for the Logistic-Hazard (CHI) method, the Logistic-Hazard method, the PMF method and the PC-Hazard method. The figure again shows that the PMF method performs slightly worse than the other methods, while the PC-Hazard method performs similarly to the Logistic-Hazard (CHI) estimates.

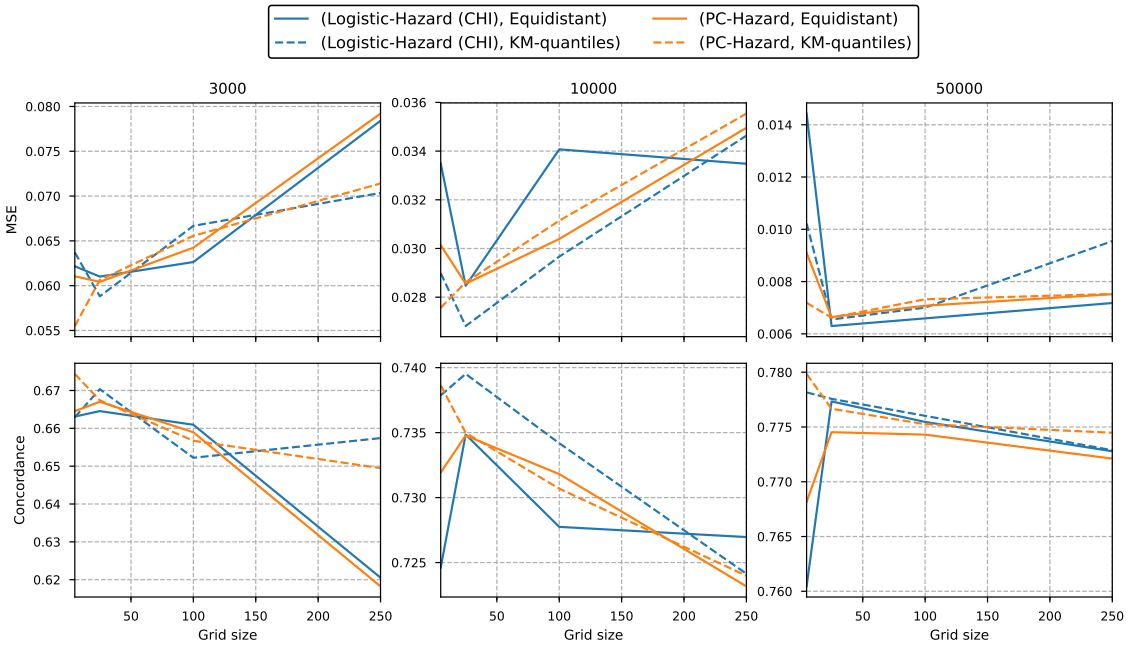

Figure 6: Median MSE and concordance for each grid size of the simulation study in Section 5.3. The number above each plot gives the size of the training set. The full lines use an equidistant grid, while the dotted lines use Kaplan-Meier quantiles for discretization. Note that the plots are not on the same scale.

5.4 Summary of Simulations

To summarize the results of the simulations, we have shown that the size of the discretization grid (number of τ_j's) has a large impact on the performance of the methods, and therefore needs to be carefully tuned. Finer grids enable the methods to reduce bias in the predictions but require more parameters in the neural networks (higher variance). By defining the discretization grid with Kaplan-Meier quantiles, the performance for the smaller grids typically improve. Furthermore, interpolation of the discrete-time survival estimates made the performance less sensitive to the number of grid points, and was generally found to improve performance of the methods for smaller grid sizes. The performance of the two proposed interpolation schemes, CHI and CDI, was more or less indistinguishable.

Comparing the three methods, we found that PMF performs slightly worse than Logistic-Hazard, both in terms of best-case performance and stability to discretization-grid configurations. PC-Hazard was found to be competitive with the interpolated Logistic-Hazard method and even performed better for the smallest training set. But the differences between all methods were small, and the size of the training sets and the grid size were shown to have a much larger impact on the performance than the choice of method.

Dataset	Size	Covariates	Prop. Censored
FLCHAIN	6,524	8	0.70
METABRIC	1,904	9	0.42
NWTCO	4,028	6	0.86
Rot. & GBSG	2,232	7	0.43
SUPPORT	8,873	14	0.32

Table 1: Datasets for comparing survival methods.

6. Experiments with Real Data

We now compare the methods discussed in this paper to other methods in the literature, in particular DeepHit (Lee et al., 2018), DeepSurv (Katzman et al., 2018), Cox-Time (Kvamme et al., 2019), CoxCC (Kvamme et al., 2019), Random Survival Forests (RSF, Ishwaran et al., 2008), and a regular Cox regression.

We conduct the comparison on five common real-world data sets: the Study to Understand Prognoses Preferences Outcomes and Risks of Treatment (SUPPORT), the Molecular Taxonomy of Breast Cancer International Consortium (METABRIC), the Rotterdam tumor bank and German Breast Cancer Study Group (Rot. & GBSG), the Assay Of Serum Free Light Chain (FLCHAIN), and the National Wilm's Tumor Study (NWTCO). Katzman et al. (2018) made the first three datasets available in their python package DeepSurv, and we have made no further preprocessing of the data. FLCHAIN and NWTCO were made available in the survival package of R (Therneau, 2015), but we use the same version of FLCHAIN as Kvamme et al. (2019). No alterations were made to the NWTCO data set. The size, the number of covariates, and the proportion of censored individuals in each data set are given in Table 1.

Hyperparameter tuning is performed with the evaluation criteria given in the paper that proposed the method. For the methods presented in this paper, however, we will use the integrated Brier score (IBS) by Graf et al. (1999) computed over 100 equidistant points between the minimum and maximum observed times in the validation set. In the simulations in Section 5, we could use the validation loss for this purpose. This is, however, no longer feasible as we now also need to tune the discretization scheme and the discretization affects the magnitude of the losses. The IBS considers both discrimination and calibration, and is a useful substitute for the MSE (21) when the true survival function is not available. Hence, we believe it is a reasonable tuning criterion.

The experiments were conducted by five-fold cross-validation. We used the same hyperparameter search and training strategy as presented in Section 6.1 of the paper by Kvamme et al. (2019), but decrease the learning rate by 0.8 at the start of each cycle, as this was found to give more stable training. The best parameter configuration for each method on each fold was fitted 10 times, and we calculated the median concordance and integrated Brier score (IBS) of the 10 repetitions and averaged that over the five folds. The results are presented in Tables 2 and 3.

In terms of concordance, we see that DeepHit and PC-Hazard perform very well. The three Logistic-Hazard methods and Cox-Time all perform close to the PC-Hazard, while the

Model	FLCHAIN	METABRIC	NWTCO	Rot. & GBSG	SUPPORT
Cox Regression	0.790	0.626	0.706	0.664	0.599
CoxCC	0.792	0.647	0.711	0.670	0.614
DeepSurv	0.792	0.640	0.709	0.674	0.615
Cox-Time	**0.793**	0.664	0.709	0.674	0.630
RSF	0.784	0.651	0.705	0.668	0.632
DeepHit	0.791	**0.675**	0.710	0.675	**0.639**
PMF	0.786	0.632	0.710	0.669	0.627
Logistic-Hazard	0.792	0.658	0.704	0.670	0.625
Logistic-Hazard (CHI)	0.790	0.656	0.714	0.673	0.628
Logistic-Hazard (CDI)	0.790	0.660	0.700	0.676	0.630
PC-Hazard	0.791	0.655	**0.716**	**0.679**	0.628

Table 2: Concordance from 5-fold cross-validation on real-world data sets.

Model	FLCHAIN	METABRIC	NWTCO	Rot. & GBSG	SUPPORT
Cox Regression	0.0961	0.183	0.0791	0.180	0.218
CoxCC	0.0924	0.173	0.0745	0.171	0.213
DeepSurv	0.0919	0.175	0.0745	0.170	0.213
Cox-Time	0.0925	0.173	0.0753	0.170	**0.212**
RSF	0.0928	0.175	0.0749	0.170	0.213
DeepHit	0.0929	0.186	0.0758	0.184	0.227
PMF	0.0924	0.174	0.0748	**0.169**	0.213
Logistic-Hazard	0.0918	**0.172**	0.0742	0.171	0.213
Logistic-Hazard (CHI)	0.0919	0.173	**0.0738**	0.170	0.213
Logistic-Hazard (CDI)	**0.0917**	**0.172**	0.0741	0.170	**0.212**
PC-Hazard	0.0918	**0.172**	**0.0738**	**0.169**	**0.212**

Table 3: Integrated Brier score from 5-fold cross-validation on real-world data sets.

PMF, RSF and the other Cox methods perform slightly worse. The concordances of the two proposed interpolation schemes, CHI and CDI, are very similar, but the CDI method gives slightly higher scores. There does, however, not seem to be much performance gain in interpolation for the concordance.

Examining the IBS in Table 3 we again find that PC-Hazard performs very well. But now, DeepHit does quite poorly. This is expected as DeepHit is designed for discrimination rather than well-calibrated estimates (see Kvamme et al., 2019). In general, the PMF, the RSF and the three proportional Cox methods seem to have slightly higher IBS than the Hazard methods, but again the differences are quite small. Cox-Time performs quite well on all data sets except for FLCHAIN and NWTCO. Comparing the interpolation schemes of Logistic-Hazard, it seems that CDI still performs slightly better than CHI, although both are quite close to the discrete estimates of Logistic-Hazard.

Interestingly, we see that, for the NWTCO data set, PC-Hazard and Logistic-Hazard (CHI) performs the best both in terms of concordance and IBS. This likely means that the piecewise exponential survival estimates are a good way of representing this data set.

In summary, all three methods discussed in this paper are competitive with existing survival methodology. However, the interpolated Logistic-Hazard or the PC-Hazard seems to give the most stable high performance considering both discrimination and calibration.

7. Discussion

In this paper, we have explored survival methodology built on neural networks for discrete-time data, and how it can be applied for continuous-time prediction. We have compared two existing discrete-time survival methods that minimize the negative log-likelihood of right-censored event times, where the first method (Lee et al., 2018) parameterize the event-time probability mass function (PMF), while the second method (Gensheimer and Narasimhan, 2019) parameterize the discrete hazard rate (Logistic-Hazard). Furthermore, we showed that the multi-task logistic regression (Yu et al., 2011; Fotso, 2018) is, in fact, a PMF parametrization. Through empirical studies of simulated and real data sets, we found that the Logistic-Hazard method performed slightly better than the PMF parametrization.

We proposed two interpolation schemes for the discrete methods, which were found to typically improve performance for smaller data sets. This is likely caused by the fact that interpolation allows for coarser discretization of the time scale, which reduces the number of parameters in the neural networks. We found that the interpolation scheme that assumed constant density within each time interval (CDI) performed slightly better than the scheme assuming constant hazard in each time interval (CHI). Note, however, that none of the schemes affect the training procedure, meaning both can be compared at test time.

We also proposed a new continuous-time method that assumes constant hazard in predefined time-intervals (PC-Hazard). The method was found to perform very well compared to existing methods, both in terms of discrimination and calibration. Furthermore, in a simulation study, we found that the method continued to performed well for coarser discretization grids than the interpolated Logistic-Hazard method. This was particularly beneficial for the smallest data set in the simulation study.

All three methods investigated in this paper need some form of discretization or coarsening of the time-scale. In that regard, we proposed a simple scheme that use the quantiles of the event-time distribution estimated by Kaplan-Meier, and showed through simulations that the quantile-based grids typically outperformed equidistant grids for coarser grids.

Acknowledgments

This work was supported by The Norwegian Research Council 237718 through the Big Insight Center for research-driven innovation.

Appendix A. More on the Simulations

In the following, we include additional information about the simulation study in Section 5. We start by explaining in detail how the data sets were created, and in Section A.2 we give some additional results.

A.1 Discrete-Time Survival Simulations from Logit Hazards

The simulated survival data sets were generated by drawing from the discrete hazard $h(t \mid \mathbf{x})$ across times $t \in \{0.1, 0.2, \ldots, 100\}$. The discrete hazard was defined through the logit hazard $g(t \mid \mathbf{x}) \in \mathbb{R}$,

$$h(t \mid \mathbf{x}) = \frac{1}{1 + \exp[-g(t \mid \mathbf{x})]},$$

ensuring that $h(t \mid \mathbf{x}) \in (0, 1)$. We let the logit hazard be a weighted sum of three different functions, $g_{\text{sin}}(t \mid \mathbf{x})$, $g_{\text{con}}(t \mid \mathbf{x})$, and $g_{\text{acc}}(t \mid \mathbf{x})$, giving

$$\begin{aligned}
g(t \mid \mathbf{x}) &= \alpha_1 \, g_{\text{sin}}(t \mid \mathbf{x}) + \alpha_2 \, g_{\text{con}}(t \mid \mathbf{x}) + \alpha_3 \, g_{\text{acc}}(t \mid \mathbf{x}), \\
g_{\text{sin}}(t \mid \mathbf{x}) &= \gamma_1 \sin(\gamma_2[t + \gamma_3]) + \gamma_4, \\
g_{\text{con}}(t \mid \mathbf{x}) &= \gamma_5, \\
g_{\text{acc}}(t \mid \mathbf{x}) &= \gamma_6 \cdot t - 10, \\
\alpha_i &= \frac{\exp(\gamma_{i+6})}{\sum_{j=1}^{3} \exp(\gamma_{j+6})}, \quad \text{for } i = 1, 2, 3.
\end{aligned}$$

Here, we actually have covariate-dependent γ_i's, i.e., $\gamma_i(\mathbf{x})$, but we have omitted the $\mathbf{x}$ for readability. Let $\tilde{x}_j$ be a linear combination of a subset of the covariates, $\tilde{x}_j = \mathbf{x}_j^T \boldsymbol{\beta}_j$ for $j = 1, \ldots, 9$, where the subsets are non-overlapping and of equal size (if the subsets are of size m, we have $\mathbf{x} \in \mathbb{R}^{9m}$). The γ's in the study are defined as

$$\begin{aligned}
\gamma_1(\mathbf{x}) &= 5\tilde{x}_1, \\
\gamma_2(\mathbf{x}) &= \frac{2\,\pi}{100} \cdot 2^{\lfloor \frac{5}{2}(\tilde{x}_2 + 1) - 1 \rfloor}, \\
\gamma_3(\mathbf{x}) &= 15\tilde{x}_3, \\
\gamma_4(\mathbf{x}) &= 2\tilde{x}_4 - 6 - |\gamma_1(\mathbf{x})|, \\
\gamma_5(\mathbf{x}) &= \frac{5}{2}(\tilde{x}_5 + 1) - 8, \\
\gamma_6(\mathbf{x}) &= \frac{1}{1 + \exp[-\frac{6}{2}(\tilde{x}_6 + 1) + 5]}, \\
\gamma_7(\mathbf{x}) &= 5(\tilde{x}_7 + 0.6), \\
\gamma_8(\mathbf{x}) &= 5\tilde{x}_8, \\
\gamma_9(\mathbf{x}) &= 5\tilde{x}_9,
\end{aligned}$$

where $\lfloor z \rfloor$ is the floor operation. We draw $\tilde{x}_j \overset{iid}{\sim} \text{Unif}[-1, 1]$, and $\beta_k \overset{iid}{\sim} N(0, 1)$. The forms of the $\gamma_i(\mathbf{x})$'s have been chosen to obtain reasonable survival functions. In particular, $\gamma_2(\mathbf{x})$ ensures that the number of periods is a multiple of 2, as we found it more reasonable than having arbitrary periods.

Finally, we draw covariates $x_{j,k}$, while ensuring $\mathbf{x}_j^T \boldsymbol{\beta}_j = \tilde{x}_j$, through the following scheme: For known $\boldsymbol{\beta}_j \in \mathbb{R}^m$, we draw $x_{j,k}$ conditionally such that

$$\left(\tilde{x}_j - \sum_{i=1}^{k} x_{j,i} \, \beta_{j,i} \right) \mid \tilde{x}_j, x_{j,1}, \ldots, x_{j,k-1} \sim \text{Unif}[-1, 1], \quad \text{for } k = 1, \ldots, m-1.$$

Hence, we sample $u_{j,k} \overset{iid}{\sim} \text{Unif}[-1, 1]$ for $k = 1, \ldots, m-1$, and set $u_{j,k} = \tilde{x}_j - \sum_{i=1}^{k} x_{j,i}\, \beta_{j,i}$, giving the covariates

$$x_{j,k} = \begin{cases} \frac{1}{\beta_{j,1}} \left(\tilde{x}_j - u_{j,1} \right), & \text{if } k = 1 \\ \frac{1}{\beta_{j,k}} \left(u_{j,k-1} - u_{j,k} \right), & \text{if } k = 2, \ldots, m-1 \\ \frac{1}{\beta_{j,m}} \left(\tilde{x}_j - \sum_{i=1}^{m-1} x_{j,i} \beta_{j,i} \right), & \text{if } k = m. \end{cases}$$

Using this scheme, it is straightforward to change the number of covariates without affecting the hazards. The code for generating theses simulations is available at `https://github.com/havakv/pycox`.

A.2 Additional Simulation Results

We here present some additional results from the simulation study in Section 5.2. Recall that each method is fitted 80 times (4 grids $\times$ 2 discretization schemes $\times$ 10 repetitions). In the same manner as in Figure 5, we plot in Figure A.1 the MSE and concordance for the Logistic-Hazard, Logistic-Hazard (CHI), PC-Hazard, and PMF, where the scores of the 80 models are sorted from best to worst.

We again see that PC-Hazard and the Logistic-Hazard (CHI) perform better than the discrete estimates of Logistic-Hazard and PMF. Furthermore, Logistic-Hazard seems to generally perform better than the PMF method. We still find that for the best grid configurations, the differences between all models are very small. But we reiterate that, for practical purposes, it is quite desirable to have stable performance for a variety of hyperparameter configurations.

Appendix B. PC-Hazard and Poisson Regression

The PC-Hazard method presented in Section 4.3 is essentially a neural network version of the piecewise exponential model studied by Holford (1976) and Friedman (1982). Friedman showed that the likelihood obtained with piecewise constant hazards is proportional to the Poisson likelihood. Consequently, one can use standard software to fit the model. Nevertheless, we prefer to implement the log-likelihood of PC-Hazard more directly, and do not use the Poisson likelihood. This is because we wanted to ensure numerical stability with the softplus (19) (as the inverse link function), while the Poisson likelihood available in most frameworks requires the log link function (i.e., an exponential activation function instead of the softplus).

To see how we can obtain the Poisson likelihood, we first need to define some variables. Recall that $\kappa(t)$ denotes the index of an interval, such that $t \in (\tau_{\kappa(t)-1}, \tau_{\kappa(t)}]$. If we define $y_{ij} = \mathbb{1}\{\kappa(t_i) = j, d_i = 1\}$ and let

$$\Delta\tilde{t}_{ij} = \begin{cases} \tau_j - \tau_{j-1}, & \text{if } t_i > \tau_j \\ t_i - \tau_{j-1}, & \text{if } \tau_{j-1} < t_i \leq \tau_j \\ 0, & \text{if } t_i \leq \tau_{j-1}, \end{cases}$$

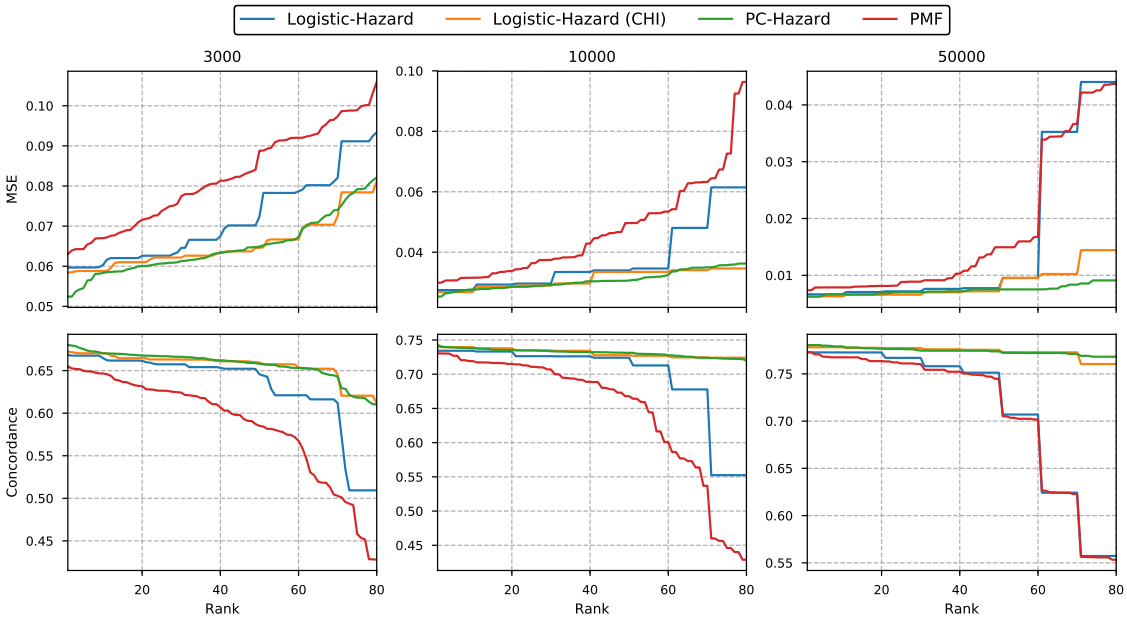

Figure A.1: MSE and concordance from the simulation study in Section 5. The scores are plotted from best to worst. The number above each plot gives the size of the training set. Note that the plots are not on the same scale.

we can rewrite the likelihood contribution in (17) as

$$L_i = \prod_{j=1}^{\kappa(t_i)} (\Delta \tilde{t}_{ij} \eta_j)^{y_{ij}} \exp\left[-\Delta \tilde{t}_{ij} \eta_j\right],$$

which is proportional to the likelihood of $\kappa(t_i)$ independent Poisson-distributed variables y_{ij} with expectations $\mu_{ij} = \Delta \tilde{t}_{ij} \eta_j$.

Appendix C. Implementation details

The implementations of the survival methods described in Sections 3 and 4 are slightly different from the mathematical notation. This is because we also need to consider numerical stability. An implementation of the methods can be found at `https://github.com/havakv/pycox`.

For the PMF parameterization, we used the log-sum-exp trick

$$\log\left(\sum_j \exp(z_j)\right) = \gamma + \log\left(\sum_j \exp(z_j - \gamma)\right),$$

where $\gamma = \max_j(z_j)$, to ensure that we only take the exponential of non-positive numbers. Hence, by rewriting the loss (11) in terms of $\phi_j(\mathbf{x})$, with $\phi_{m+1}(\mathbf{x}) = 0$ and $\gamma_i = \max_j(\phi_j(\mathbf{x}_i))$,

we obtain

$$\text{loss} = -\frac{1}{n}\sum_{i=1}^{n} d_i[\phi_{\kappa(t_i)}(\mathbf{x}_i) - \gamma_i] + \frac{1}{n}\sum_{i=1}^{n}\log\left(\sum_{j=1}^{m+1}\exp[\phi_j(\mathbf{x}_i) - \gamma_i]\right) - \frac{1}{n}\sum_{i=1}^{n}(1-d_i)\log\left(\sum_{j=\kappa(t_i)+1}^{m+1}\exp[\phi_j(\mathbf{x}_i) - \gamma_i]\right).$$

For the discrete hazard parametrization, we can simply formulate it as the negative log-likelihood for Bernoulli data, or binary cross-entropy, and use existing implementations of the loss function to ensure numerical stability. In practice, these implementations use the log-sum-exp trick on the logits $\phi_j(\mathbf{x})$.

Finally, for the continuous hazard parametrization, we use existing implementations of the softplus function which uses a linear function over a certain threshold, meaning $\log(1 + \exp[z]) \approx z$ for large values of z. However, we also note that for $z \approx 0$, we have that $\log(1 + z) \approx z$. Hence, for $\phi_{\kappa(t_i)}(\mathbf{x}_i) \ll 0$ we use that

$$\log \tilde{\eta}_{\kappa(t_i)}(\mathbf{x}_i) = \log[\log(1 + \exp[\phi_{\kappa(t_i)}(\mathbf{x}_i)])] \approx \phi_{\kappa(t_i)}(\mathbf{x}_i).$$

References

Paul D. Allison. Discrete-time methods for the analysis of event histories. *Sociological Methodology*, 13:61–98, 1982.

Laura Antolini, Patrizia Boracchi, and Elia Biganzoli. A time-dependent discrimination index for survival data. *Statistics in Medicine*, 24(24):3927–3944, 2005.

Charles C. Brown. On the use of indicator variables for studying the time-dependence of parameters in a response-time model. *Biometrics*, 31(4):863–872, 1975.

Travers Ching, Xun Zhu, and Lana X. Garmire. Cox-nnet: An artificial neural network method for prognosis prediction of high-throughput omics data. *PLoS Computational Biology*, 14(4):e1006076, 2018.

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

David Faraggi and Richard Simon. A neural network model for survival data. *Statistics in Medicine*, 14(1):73–82, 1995.

Stephane Fotso. Deep neural networks for survival analysis based on a multi-task framework. *arXiv preprint arXiv:1801.05512*, 2018.

Michael Friedman. Piecewise exponential models for survival data with covariates. *The Annals of Statistics*, 10(1):101–113, 1982.

Michael F. Gensheimer and Balasubramanian Narasimhan. A scalable discrete-time survival model for neural networks. *PeerJ*, 7:e6257, 2019.

Erika Graf, Claudia Schmoor, Willi Sauerbrei, and Martin Schumacher. Assessment and comparison of prognostic classification schemes for survival data. *Statistics in Medicine*, 18(17-18):2529–2545, 1999.

Frank E. Harrell Jr, Robert M. Califf, David B. Pryor, Kerry L. Lee, and Robert A. Rosati. Evaluating the yield of medical tests. *Journal of the American Medical Association*, 247 (18):2543–2546, 1982.

Theodore R. Holford. Life tables with concomitant information. *Biometrics*, pages 587–597, 1976.

Hemant Ishwaran, Udaya B. Kogalur, Eugene H. Blackstone, and Michael S. Lauer. Random survival forests. *Annals of Applied Statistics*, 2(3):841–860, 2008.

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), 2018.

John P. Klein and Melvin L. Moeschberger. *Survival Analysis: Techniques for Censored and Truncated Data*. Springer, New York, 2. edition, 2003.

Håvard Kvamme, Ørnulf Borgan, and Ida Scheel. Time-to-event prediction with neural networks and Cox regression. *Journal of Machine Learning Research*, 20(129):1–30, 2019.

Changhee Lee, William R Zame, Jinsung Yoon, and Mihaela van der Schaar. Deephit: A deep learning approach to survival analysis with competing risks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Margaux Luck, Tristan Sylvain, Héloïse Cardinal, Andrea Lodi, and Yoshua Bengio. Deep learning for patient-specific kidney graft survival analysis. *arXiv preprint arXiv:1705.10245*, 2017.

Leslie N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2017.

Terry M. Therneau. *A Package for Survival Analysis in S*, 2015. URL `https://CRAN.R-project.org/package=survival`. version 2.38.

Gerhard Tutz and Matthias Schmid. *Modeling discrete time-to-event data*. Springer, 2016.

Safoora Yousefi, Fatemeh Amrollahi, Mohamed Amgad, Chengliang Dong, Joshua E. Lewis, Congzheng Song, David A. Gutman, Sameer H. Halani, Jose Enrique Velazquez Vega, Daniel J. Brat, et al. Predicting clinical outcomes from large scale cancer genomic profiles with deep survival models. *Scientific reports*, 7(11707), 2017.

Chun-Nam Yu, Russell Greiner, Hsiu-Chin Lin, and Vickie Baracos. Learning patient-specific cancer survival distributions as a sequence of dependent regressors. In *Advances in Neural Information Processing Systems 24*, pages 1845–1853. Curran Associates, Inc., 2011.

X. Zhu, J. Yao, F. Zhu, and J. Huang. Wsisa: Making survival prediction from whole slide histopathological images. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6855–6863, July 2017.

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.