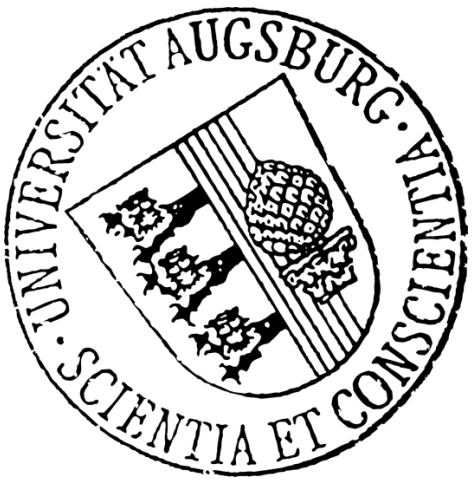
Master Thesis

University of Augsburg

Department of Mathematics

Chair for Computational Statistics
and Data Analysis



Principles of Deep Learning for Survival Analysis

Anna Sophie Rubeck

October 2021

Contents

1. Introduction

Survival analysis is a pillar of statistical analysis especially for medical data. Such data could for example originate from case studies conducted over some period of time. It might very well happen that in the limited time period of such a study, some event of interest does not occur. For instance a patient might neither recover nor die from a disease during the study. Hence a fundamental issue is that if we do not observe the event we are presented with missing data. As a consequence, analysis of survival data requires special methods.

These methods often rely on assumptions about the data that might be too restrictive for complex data. An alternative approach is to apply deep learning techniques. To account for missing data, these techniques require the specification of a suitable loss function governing the training of feedforward networks. However deep learning algorithms have the key advantage that they are less dependent on the validity of specific assumptions.

In this thesis we describe how survival data can be analyzed with deep learning algorithms as deep feedforward networks. We start by presenting the main ideas of survival analysis in Section 2. The concept of censoring, the layout of survival data, as well as parametric and semiparametric approaches to survival analysis are described. Then we briefly outline deep learning focusing on feedforward networks. In Section 4 we illustrate how deep learning can be used to analyze survival data in the case where no observations are censored. The main part of this thesis is Section 5 where we consider censored data and how deep learning algorithms can be extended to account for this setting. For this goal we introduce censoring unbiased loss functions and how they can be used to estimate survival probabilities and restricted mean survival. Then we show that deep feedforward networks that use the censoring unbiased loss functions are equivalent to feedforward networks that use the L_2 loss with transformed data. The resulting algorithms are analyzed in a simulation study and tested using the Mayo clinic primary biliary cholangitis data.

2. Principles of Survival Analysis

In this section, following Kleinbaum and Klein (2020) and Klein and Moeschberger (2003), we briefly describe the main ideas of survival analysis, the concept of censoring, and parametric and semiparametric approaches to the problem of estimating survival times.

Survival analysis is used in various fields such as medicine, insurance and finance to predict a certain risk. The main goal is to estimate and analyze the time until a specified event occurs. A medical application could be a study about the time to death after the contraction of a disease. In this case the patient enters the study as soon as the disease is diagnosed. After a certain time the patient dies. As the event of interest is death we will refer to the time period between study entry and death as the *true survival time*. In some cases we cannot observe the patient's true survival time due to censoring which will be explained in greater detail in Section 2.1.

In this thesis we consider a single event of interest that can either occur or not. It is also possible to consider competing risks meaning that multiple different events that are taken into account. A detailed explanation of this case is given in Kalbfleisch and Prentice (2002, Chapter 8). We assume that the event of interest can only take place once during the observed period. It is possible to study recurrent events where events can happen to a subject multiple times. For further details refer to Kalbfleisch and Prentice (2002, Chapter 9).

2.1. Censoring

This section follows Kleinbaum and Klein (2020) and Klein and Moeschberger (2003).

Survival data is often incomplete meaning that for some subjects the particular event is not observed within the study period. We refer to this data as *censored*. In this case we cannot observe the true survival time but instead observe the *censoring time*, i.e. the time at which the censored subject is no longer examined. The *observed survival time* is then either given by the true survival time (if we observe the event) or the censoring time (if the data is censored).

Even though censored data is not observed until the event, it still contains valuable information about the time between the entry of the study and the event. Consider for example data obtained from a medical study. There are three possible reasons why censoring occurs:

- The study ends before a participant experiences the event.
- A participant is lost to follow-up during the study period.
- A participant withdraws from the study or experiences a different event that makes further follow-up impossible.

The reasons mentioned above eintail *right censored* data. We call censored data right censored, if the true survival time is equal to or greater than the censoring time.

There are two other types of censoring that may occur. If the true survival time is less than or equal to the observed censoring time, the data is *left censored*. This could be due to subjects that already experienced the event of interest before entering the study making the true event time unobservable. Consider analysis of the Corona disease where patients enter the study as soon as they experience symptoms. It is not possible to determine the exact time when the patient contracted the disease, thus leading to left censored data.

A more general case is *interval censoring*. It occurs if we only know that the true survival time lies within a specified time interval, but we cannot observe it directly. The last case incorporates right censoring and left censoring as special cases.

Even though censored data is not fully observed, we cannot treat them the same as uncensored data or even leave them out completely. This both leads to biased estimators. For example, if we treat censored subjects as if they experienced the event at their censoring time, we potentially ignore the time period between the drop out of the study and the event. This would lead to an artificially lowered survival curve.

We consider the data to be possibly right censored, but not left or interval censored. Further information about left and interval censoring can for example be found in Kalbfleisch and Prentice (2002).

There are three main assumptions about censoring that are often required for further analysis, which are independent, random and non-informative censoring. To achieve random censoring, we assume that the failure rate for censored subjects is the same as the failure rate for the uncensored individuals that remain in the risk set. Independent censoring is the assumption that censoring occurs randomly within any subgroup of interest. The third assumption is that censoring is non-informative. This means that given the explanatory variables, the distribution of survival times is conditionally independent of the distribution of censoring times. These assumptions are covered in greater detail in Kalbfleisch and Prentice (2002).

2.2. Survival Analysis Data

This section is based on Kleinbaum and Klein (2020).

We now describe how the survival data for each of the patients $i = 1, \ldots, n$ can be represented.

We denote by $T_i \geq 0$ the random variable for the true survival time of subject i and by $C_i \geq 0$ the random variable for the censoring time of the respective individual. The variables $T_1, \ldots, T_n$ and $C_1, \ldots, C_n$ are each identically and independently

distributed. As we consider right censored data, the observed time is given by $\tilde{T}_i = \min\{T_i, C_i\}$.

We also observe p specific characteristics for each subject that might play a role in the prediction of the survival time. These explanatory (or predictor) variables are known for each individual. They are given by the vector

$$X_i = (X_{i1}, \ldots, X_{ip}) \in \delta \subset \mathbb{R}^p,$$

where δ denotes the bounded space consisting of all possible values that the explanatory variables can take. The event indicator $\Delta_i = I(T_i \leq C_i)$ specifies whether subject i has experienced the event of interest ($\Delta_i = 1$) or the data is censored ($\Delta_i = 0$). Then the basic data layout has the form $(\tilde{T}_i, \Delta_i, X_i)$, $i = 1, \ldots, n$, where $n \in \mathbb{N}$ is the number of patients that participate in the study. As it is common in the literature we will not further distinguish notationally between a random variable and its realization.

Two important functions are frequently used for analysis and prediction of survival times: the survival function and the hazard function.

The survival function $S(t) = P(T > t)$ specifies the distribution of the survival time given by $F(t) = 1 - S(t)$. It can for example be used to analyze and compare the time to event of two different groups. The survival function has three important properties: $S(\cdot)$ is a non-increasing function, $S(0) = 1$, and $\lim_{t\to\infty} S(t) = 0$. These properties follow directly from the relation $F(t) = 1 - S(t)$ and that F is a distribution function.

The hazard function $h(\cdot)$, also called conditional failure rate, is given by

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t}.$$

It is a measure of instantaneous potential to experience the event and can be used to identify a specific model form.

The hazard function has two important properties: it is a non-negative function, i.e. $h(t) \geq 0 \; \forall t$, and it has no upper bound. In contrast to the survival function $S(\cdot)$, the hazard function does not necessarily have any monotonicity properties.

There exists a clearly defined relationship between the survival function $S(t)$ and the hazard function $h(t)$:

$$\begin{aligned} S(t) &= \exp\left(-\int_0^t h(u)du\right) \\ h(t) &= -\left[\frac{dS(t)/dt}{S(t)}\right]. \end{aligned}$$

As a consequence, specifying one of the two functions directly yields the other one.

2.3. (Semi-)Parametric Approaches to Estimation of Survival Time

This section relies on Kleinbaum and Klein (2020).

An approach to the estimation of survival time are *parametric survival models*. Here, we assume a specific family of distributions for the survival time and then use the given data to estimate the parameters of the distribution. The parameters can be estimated using maximum likelihood. Examples for parametric models are linear, logistic, or Poisson regression. Common choices for the family of distributions when analyzing survival times are the Weibull, exponential or lognormal distribution. The parametric approach is discussed in further detail in Kleinbaum and Klein (2020).

Three main advantages of the parametric survival model are its simplicity, that it is consistent with the theoretical survival function $S(\cdot)$ and that it specifies the whole survival function.

A big disadvantage is that the quality of the estimation using parametric models strongly depends on a suitable choice of the family of distributions. As a consequence, a detailed analysis of the data and a precise understanding of the families of distributions are required. The Cox proportional hazard model tries to overcome these limitations by relaxing the parametric approach and is described in the following.

The *Cox proportional hazards model* (Cox PH model) is a popular approach to analyzing survival data. The model is built upon the assumption that the hazard function has the following form:

$$h(t, x) = h_0(t) \exp\left(\sum_{i=1}^{p} \beta_i x_i\right),$$

where $x = (x_1, \ldots, x_p)$ are the explanatory variables and $\beta = (\beta_1, \ldots, \beta_p)$ is a vector of weights. This includes the central assumption of the Cox PH model, which is the proportional hazards (PH) assumption. It states that the *baseline hazard* h_0 does not involve the explanatory variables and is thus independent of them.

Typically, maximum likelihood is used to estimate the weights β_i, $i = 1, \ldots, p$. As the baseline hazard h_0 is not specified, the Cox PH model is a semiparametric model.

In contrast to parametric models, the Cox PH model does not require an assumption about the distribution of the outcome. Another important advantage is its robustness, meaning that it closely approximates the correct parametric model of the survival function.

If we are sure which family of distributions models the survival times adequately,

parametric models give the best estimate. But if we are in doubt that the model is a good fit, the Cox PH model provides reliable results.

As it is necessary for the performance of the Cox PH model that the PH assumption holds, there are several tests that can be performed, e.g. graphical techniques or goodness-of-fit-tests. Further details can for example be found in Kleinbaum and Klein (2020, Chapter 3).

There exist various modifications of the Cox PH model. If the PH assumption is not satisfied, we can for example use a stratified Cox procedure instead, a modified version of the Cox PH model. The general idea is to categorize the data and divide it into subgroups in a way that within each subgroup the PH assumption is met. Then for each subgroup the Cox PH model can be applied to obtain partial likelihood functions. Those can then be combined to an overall likelihood. This approach is explained in greater detail in Kleinbaum and Klein (2020, Chapter 5).

In many cases it may be too simplistic to assume that the effect of the explanatory variables is linear, as it is the case for the Cox PH model. Thus, we need a richer family of survival models to properly fit survival data with nonlinear risk functions.

One possible approach to compensate for this disadvantage is to use deep learning procedures. In this thesis, we take a closer look at this approach. We start by introducing the main ideas of deep learning and explain the idea of feedforward networks.

3. Principles of Deep Learning

This section relies on Goodfellow, Bengio, and Courville (2016).

Machine learning describes a broad class of algorithms having the goal to learn or extract information from a given sample without the need of explicit programming. To achieve this goal machine learning algorithms require a task and a performance measure. Such a task could be to recognize whether a cat or a dog is on a picture, given examples of already classified pictures. The algorithm then builds a prediction model based on a given training set. In the example above the training set consists of pictures of cats and dogs and corresponding labels specifying what is visible on the respective picture. The performance measure then describes how well this task was achieved by calculating some sort of distance between the predicted label and the real label that is given in the training set. It is a key requirement to specify a suitable performance measure when applying machine learning algorithms.

The central challenge for a machine learning algorithm is that it must perform well on new and previously unseen inputs meaning that the error measure on the test set should be as small as possible and at the same time the error measure on the training set should also be reduced. This means that there are two factors that determine how well a machine learning algorithm will perform: make the training error small and make the gap between the training and the test error small.

If the first task is not fullfilled, underfitting occurs. This means that the error value on the training set is not sufficiently low for our model. If the second task is not fullfilled, overfitting occurs. In this case, the gap between training and test error is too large.

A common approach to reduce the possibility of overfitting is regularization. In principle, regularization is any modification of a learning algorithm that aims to reduce its generalization error. In practice this is often achieved by adding a penalty to the cost function.

Deep learning models are machine learning models that are based on artificial neural networks. A common class of deep learning models are deep feedforward networks. Analogous to other estimators they aim to approximate some function f^*. More precisely, given the data x and the desired output y, the goal is to find f^*, such that $f^*(x) = y$. This is achieved by defining a mapping $y = f(x; \beta)$ and learning the parameters β that result in the best function approximation. In this setting loss functions are used as a performance measure. Possible choices include the L_2 loss or the Brier loss which are both further described in Section 5.

The term "network" suggests that the mapping f is a composition of many different functions. A directed acyclic graph is used to describe how these functions are composed together in the model. For example, the network could consist of a chain of K functions $f = f^{(K)} \circ \cdots \circ f^{(1)}$. In this case each function $f^{(k)}$, $k = 1, \ldots, K$ forms a layer and the number of composed functions K gives the

depth of the model. The dimension of the output of each layer $f^{(k)}$ is denoted by n_k and determines the width of the layer. Typically these functions are vector-valued and instead of thinking of the layer as a vector-to-vector function, we can interpret it as many *units* that each represent a vector-to-scalar function. Considering $f^{(1)}(x, y) = (x^2 + y, 2y)$ we can represent the function by two units $f_1^{(1)}(x, y) = x^2 + y$ and $f_2^{(1)}(x, y) = 2y$.

The final layer is called the output layer. We want to achieve that this layer produces a value that is close to the desired output y, i.e. for each data x and corresponding label y: $y \approx f^*(x)$.

The remaining layers are called hidden layers because their results and their inner workings do not constitute the observed output of the network. We are only interested in the prediction, the output of the final layer. Before creating a deep feedforward network it is necessary to determine the structure of the associated directed acyclic graph. This is achieved by setting the number of units and specifying how these units are connected.

In the model each of the units $f_i^{(k)}$, $k = 1, \dots, K$, $i = 1, \dots, n_k$ is associated with a vector of weights $a_i^{(k)} \in \mathbb{R}^{n_{k-1}}$, a bias term $b_i^{(k)} \in \mathbb{R}$ and an activation function $\phi : \mathbb{R} \to \mathbb{R}$. The output of each unit $f_i^{(k)}$, $k = 1, \dots, K$ is then computed as

$$f_i^{(k)}(y) = \phi\left(\left(a_i^{(k)}\right)^T y + b_i^{(k)}\right),$$

where y is the output of layer $f^{(k-1)}$. These units are then combined to form the output of the whole layer which is given by $f^{(k)} = \left(f_1^{(k)}, \dots, f_{n_k}^{(k)}\right)^T$. The output of the first layer $f^{(1)}$ is computed using the data x as input and n_0 is defined as the dimension of x. Popular choices for the activation function are the ReLU activation function given by $\phi(x) = \max(0, x)$ or the sigmoid activation function given by $\phi(x) = (1 + e^{-x})^{-1}$. The weights and the biases make up the learning parameter β.

The above described procedure can be used to iteratively compute the output of the final layer.

A central question in dealing with feedforward networks is how to determine appropriate weights and biases. There are several techniques for this training procedure for instance stochastic variants of gradient descent. An overview of such algorithms can be found in Goodfellow, Bengio, and Courville (2016, Chapter 5).

The networks described above are called feedforward networks because information only travels in one direction from input to output, i.e. there are no feedback connections between the layers. This is the reason why we require the associated graph to be acyclic.

When building deep feedforward networks, overfitting can occur. A common approach to counteract this phenomenon is a regularization method called dropout.

According to Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014) the idea is to randomly drop units from the neural network during each step of the learning algorithm. For this method, a dropout rate p must be chosen which specifies that a unit is included in the current iteration with probability $1 - p$. As a consequence, each of the hidden units must be able to perform well regardless of the other units included in the model and the co-dependence is minimized. Further details about dropout regularization can be found in Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014).

In the following section we describe how survival data can be analyzed using feedforward networks

4. Deep Learning for Fully Observed Data

This section follows the ideas of Steingrimsson and Morrison (2020) and insights described in Goodfellow, Bengio, and Courville (2016).

As described in Section 2.2, we assume that the dataset consists for each subject $i = 1, \ldots, n$ of a positive continuous failure time $T_i \in \mathbb{R}^+$ and a vector of explanatory variables $X_i = (X_{i1}, \ldots, X_{ip}) \in \delta \subset \mathbb{R}^p$. In the context of a medical study these explanatory variables could for example be age, in which treatment group the patient was or the oxygen concentration in blood at the beginning of the study. The set δ describes the bounded space consisting of all values that the explanatory variables can take in the specific context. In the example described above, a possible choice for δ could be $\delta = [0, 120] \times \{0, 1, 2\} \times [50, 200]$. We assume that the outcome T_i is possibly transformed via a monotone function $h : \mathbb{R}^+ \to \mathbb{R}$. Possible choices for this transformation are the identity function $h(t) = t$ or the logarithmic transformation $h(t) = \ln(t)$.

In this section we assume that the data is not censored. As a consequence, the event indicator is not required and the data set reduces to the fully observed data given by

$$\mathcal{F} = \{(h(T_i), X_i)\,; i = 1, \ldots, n\}.$$

We now describe the structure of a deep learning algorithm based on feedforward networks for fully observed data, as presented in Steingrimsson and Morrison (2020).

Algorithm 4.1.

1. Setup the structure of the neural network.
2. Estimate the learning parameter.
3. Use cross-validation to select the penalization parameter η from a predetermined sequence $\eta_1, \ldots, \eta_M$.
4. Create the final prediction model.

We now further analyze these steps.

First we define the layout of the neural network that we want to use for our prediction. For this, we need to set the number of layers K and define the functions $f^{(k)}_{\beta_k}$ for each layer $k = 1, \ldots, K$. The output of the hidden layer architecture is then given by $f_\beta(X) = f^{(K)}_{\beta_K} \circ f^{(K-1)}_{\beta_{K-1}} \circ \cdots \circ f^{(1)}_{\beta_1}(X)$, where $\beta = (\beta_1^T, \ldots, \beta_K^T)^T$ is a vector of unknown weights and biases which will be estimated next.

In the second step, the estimation of the learning parameter, we need a prespecified loss function $L(h(T), f(X))$ that describes the difference between the

prediction of our model $f(X)$ and the desired outcome $h(T)$. As we described in Section 3 the learning parameter β consists of weights and biases. We also use a penalization parameter η to try to keep the learning parameter as small as possible. Large weights can be a sign of a complex network and thus overfitting of the training data. To counteract this effect, we penalize such large weights. We estimate the learning parameter β by minimizing the empirical penalized loss function $\hat{L}$ which is defined as:

$$\hat{L}(\beta) = \frac{1}{n} \sum_{i=1}^{n} L(h(T_i), f_\beta(X_i)) + \eta \|\beta\|^2, \tag{4.1}$$

where $\|\beta\|^2$ is the L_2 norm of β.

The third step ensures that we make a suitable choice for the penalization parameter η. To perform cross-validation, we first split the given dataset randomly into D disjoint sets $K_1, \ldots, K_D$ and choose a sequence of M possible penalization parameters. Suitable choices for η strongly depend on the model architecture. A detailed analysis for neural networks with one hidden layer is given in Heiss, Teichmann, and Wutte (2019). Then, for a fixed subset K_ℓ, $\ell \in \{1, \ldots, D\}$ and penalization parameter η_m, $m \in \{1, \ldots, M\}$, we set $\hat{\beta}^{(\ell)}_{\eta_m}$ as the learning parameter estimated in equation (4.1) using the penalization parameter η_m and the data $\mathcal{F} \setminus K_\ell$. Let $a_{i,\ell}$ be the indicator whether observation i is included in the dataset K_ℓ ($a_{i,\ell} = 1$) or not ($a_{i,\ell} = 0$). The cross-validation error corresponding to the penalization parameter η_m is then defined as

$$\alpha(m) = \sum_{l=1}^{D} \sum_{i=1}^{n} a_{i,l} L(h(T_i), f_{\hat{\beta}^{(l)}_{\eta_m}}(X_i)).$$

Then we choose the penalization parameter that minimizes the cross-validation error for our model, i.e. let $\eta = \eta_{m^*}$, where $m^* = \arg\min_{m \in \{1,\ldots,M\}} \alpha(m)$.

In step four we use the optimal penalization parameter η_{m^*} to estimate the learning parameter $\hat{\beta}^{(l)}_{\eta_{m^*}}$ and then build the final prediction model $f_{\hat{\beta}^{(l)}_{\eta_{m^*}}}$.

The learning parameter $\beta(X)$ that is estimated using the explanatory variables X minimizes the expected loss used in the algorithm. Thus, the parameter that we want to estimate determines the choice of loss function. If we use for example the L_2 loss, the population parameter estimated by the full data deep learning algorithm is the conditional mean $E[h(T)|X]$. If we instead want to estimate survival probabilities $P(T \geq t|X)$ for a fixed time point t, we can use the Brier loss which is given by $E[(I(T \geq t) - \beta(X))^2]$.

In the next section we describe how Algorithm 4.1 can be extended to censored data.

5. Deep Learning for Censored Data

This section relies on Steingrimsson and Morrison (2020) as well as Goodfellow, Bengio, and Courville (2016).

In this section we will now allow for the dataset to contain censored data meaning that in some cases we cannot observe the true event time of an individual.

The observed dataset consists of n independent and identically distributed observations and is given by

$$\mathcal{O} = \{(\tilde{T}_i = \min\{T_i, C_i\}, \Delta_i = I(T_i \leq C_i), X_i); i = 1, \ldots, n\},$$

where $\tilde{T}_i$ denotes the observed time, which is either the event time T_i or the censoring time C_i, and Δ_i indicates whether subject i experienced the event of interest or was censored. Again, the explanatory variables are given by X_i.

We define the true conditional survival functions for the event time T and the censoring time C as $S_0(u|X) = P(T > u|X)$ and $G_0(u|X) = P(C > u|X)$ respectively. In contrast to Section 2.2 the survival function now depends on the explanatory variables X. We assume that the censoring time C is continuous and that the corresponding censoring mechanism is non-informative, i.e. that C is independent of $T|X$.

For further computations we also assume that $G_0(T|X) \geq \varepsilon > 0$ for some $\varepsilon > 0$, to ensure positivity of the conditional censoring function. We refer to this as the positivity assumption.

The main difficulty in extending Algorithm 4.1 to allow for censored data is to define a suitable loss function that can be computed in the presence of censoring and at the same time reduces to the full data loss $L(h(T), f(X))$ if the data is not censored. To close the gap between what is done in the presence and absence of censoring, we will now define *censoring unbiased transformations*.

5.1. Censoring Unbiased Transformations

This section is based on Steingrimsson, Diao, and Strawderman (2019) and Rubin and van der Laan (2007).

For prediction with right censored data, it is a popular approach to replace the possibly censored observations $\tilde{T}$ with surrogate values. This can be achieved by using an appropriate mapping

$$Y^*(\cdot) : \mathbb{R}^+ \to \mathbb{R}.$$

In our case the key requirement is that the mapping $Y^*(\cdot)$ is a censoring unbiased transformation, which is defined as follows.

Definition 5.1. Let $\tilde{T}$ be a scalar function of the full data $\mathcal{F}$ and $Y^*(\mathcal{O})$ a scalar function of the observed data $\mathcal{O}$. Then $Y^*(\mathcal{O})$ is a *censoring unbiased transformation (CUT)* for $\tilde{T}$ if for every point X in δ:

$$E[Y^*(\mathcal{O})|X] = E[\tilde{T}|X].$$

In other words, a censoring unbiased transformation maps the observed time $\tilde{T}$ in a way that it keeps its conditional mean structure.

We will now take a closer look at three specific CUTs: the Buckley–James transformation, the inverse probability of censoring weighted mapping and a doubly robust censoring unbiased transformation.

The Buckley–James transformation is given by

$$Y^*_{BJ}(\mathcal{O}; S) = \Delta\tilde{T} + (1 - \Delta)m(C, X; S), \tag{5.1}$$

where

$$m(t, X; S) = E_S[\tilde{T}|\tilde{T} > t, X] = -\frac{1}{S(t|X)} \int_t^\infty u dS(u|X) \tag{5.2}$$

and $S(\cdot|X)$ denotes the conditional survival function.

This transformation is rather intuitive. For an uncensored subject, i.e. $\Delta = 1$, the response remains unchanged. If the response is censored, i.e. $\Delta = 0$, it is mapped to its conditional expectation.

Theorem 5.1. *Under the mild regularity conditions given in Appendix A it holds that $Y^*_{BJ}(\cdot; S_0)$ is a CUT.*

The proof is given in Appendix B.1 and follows the ideas in Rubin and van der Laan (2007) with $G(t|X) = 1$ for all t.

The Buckley–James Transformation has an important property. Among all CUTs $Y^*(\cdot)$ the mapping $Y^*_{BJ}(\cdot)$ results in the best predictor of the original response, i.e.

$$Y^*_{BJ}(\mathcal{O}) = \underset{Y^*(\cdot) \text{ is a CUT}}{\arg\min} E[|Y^*(\mathcal{O}) - \tilde{T}|^2].$$

On the other hand, it requires the correct specification of the survival function $S(\cdot|X)$.

In contrast to the Buckley–James transformation, which depends on the survival function $S(\cdot|X)$, it is also possible to build mappings that instead depend only on the censoring mechanism. One example for this is the inverse probability of censoring weighted (IPCW) mapping. Here, the basic idea is to weight the contribution of each uncensored observation by the inverse probability of being

censored. This way both distributions, the censoring and the survival distributions, are taken into account. The IPCW mapping is defined as follows:

$$Y^*_{IPCW}(\mathcal{O}; G) = \frac{\Delta \tilde{T}}{G(\tilde{T}|X)}.$$

For the IPCW transformation we need to estimate the conditional censoring function given the explanatory variables $G(\cdot|X)$. If this function is modeled correctly, meaning that $G(\cdot|X) = G_0(\cdot|X)$, the IPCW transformation $Y^*_{IPCW}(\cdot; G_0)$ is a CUT. This claim is shown in the following theorem.

Theorem 5.2. *Under the regularity conditions given in Appendix A the IPCW transformation $Y^*_{IPCW}(\cdot; G_0)$ is a CUT.*

Proof. The desired property can be proven by using the tower rule

$$\begin{aligned} E\left[Y^*_{IPCW}(\mathcal{O}; G_0)|X\right] &= E\left[\frac{\Delta \tilde{T}}{G_0(\tilde{T}|X)}\middle| X\right] = E\left[E\left[\frac{\Delta \tilde{T}}{G_0(\tilde{T}|X)}\middle| X, \tilde{T}\right]\middle| X\right] \\ &= E\left[\frac{\tilde{T}}{G_0(\tilde{T}|X)} P(\Delta = 1|X, \tilde{T})\middle| X\right] \end{aligned}$$

Now we can use that $P(\Delta = 1|X, \tilde{T}) = 1 - P(\Delta = 0|X, \tilde{T})$ to obtain

$$E\left[Y^*_{IPCW}(\mathcal{O}; G_0)|X\right] = E\left[\frac{\tilde{T}}{G_0(\tilde{T}|X)} G_0(\tilde{T}|X)\middle| X\right] = E[\tilde{T}|X]. \qquad \square$$

The IPCW transformed data is a weighted version of the observed time with weights

$$\omega_{IPCW}(\mathcal{O}; G) = \frac{\Delta}{G(\tilde{T}|X)}. \tag{5.3}$$

The Buckley–James estimator, as well as the IPCW mapping, depend on the parameters $S(\cdot|X)$ and $G(\cdot|X)$ respectively. As a consequence, poor estimators of these two conditional functions can significantly degrade the performance of regression applied to the transformed data.

A third mapping is the doubly robust mapping. In this context "doubly robust" implies that the mapping requires a good approximation of either $S(\cdot|X)$ or $G(\cdot|X)$, but not necessarily both. It is defined as follows:

$$Y^*_{DR}(\mathcal{O}; G, S) = \frac{\Delta \tilde{T}}{G(\tilde{T}|X)} + \frac{(1-\Delta) m(C, X; S)}{G(C|X)} - \int_0^{\tilde{T}} \frac{m(c, X; S)}{G(c|X)} d\Lambda_G(c|X), \tag{5.4}$$

with $m(c, X; S)$ as defined in Equation (5.2) and the cumulative hazard function $\Lambda_G(t|X) = -\int_0^t 1/G(u|X)dG(u|X)$.

The first term in Equation (5.4) is as in the IPCW transformation. Similar to the Buckley–James transformation, a term that considers the censored data is added. The last part can be seen as an augmentation term. In total, the doubly robust transformation can be interpreted as an augmented inverse probability weighted mapping. For further details about the augmentation refer to Kalbfleisch and Prentice (2002).

We can show the double robustness property using Theorem 3.1 in Rubin and van der Laan (2007) which is reproduced below.

Theorem 5.3. *Assume that the regularity conditions given in Appendix A are met. Then, if either $S_0(\cdot|X)$ or $G_0(\cdot|X)$ are specified correctly, i.e. $\tilde{S}(\cdot|X) = S_0(\cdot|X)$ or $\tilde{G}(\cdot|X) = G_0(\cdot|X)$, it holds that*

$$E[Y_d^*(\mathcal{O}; \tilde{G}, \tilde{S})|X] = E[\tilde{T}|X],$$

with

$$Y_d^*(\mathcal{O}; G, S) = \frac{\Delta\tilde{T}}{G(\tilde{T}|X)} + \frac{(1-\Delta)m(C, X; S)}{G(C|X)} - \int_{-\infty}^{Y} \frac{m(c, X; S)}{G^2(c|X)} dF(c|X) \quad (5.5)$$

In the following lemma we show that the transformations Y_d^* and Y_{DR}^* are equivalent and thus Theorem 5.1 can be applied to show that $Y_{DR}^*(\cdot; G_0, \tilde{S})$, $Y_{DR}^*(\cdot; \tilde{G}, S_0)$ and $Y_{DR}^*(\cdot; G_0, S_0)$ are each CUTs.

Lemma 5.1. *Assuming the regularity conditions given in Appendix A it holds that $Y_{DR}^*(\mathcal{O}; G_0, \tilde{S})$, $Y_{DR}^*(\mathcal{O}; \tilde{G}, S_0)$ and $Y_{DR}^*(\mathcal{O}; G_0, S_0)$ are each CUTs.*

Proof. We verify the lemma by proving that Y_d^* and Y_{DR}^* both perform the same transformation and as a consequence Theorem 5.3 can be applied to Y_{DR}^*. The first two terms of the transformation Y_d^* are equal to those in the transformation Y_{DR}^*. Thus it is sufficient to show that

$$\int_{-\infty}^{Y} \frac{m(c, X; S)}{G^2(c|X)} dF(c|X) = \int_{0}^{Y} \frac{m(c, X; S)}{G(c|X)} d\Lambda_G(c|X).$$

As we only consider positive event times the censoring distribution is only defined for nonnegative values. Thus the lower bound of the left hand integral can be set to 0.

$$\begin{aligned} \int_{-\infty}^{Y} \frac{m(c, X; S)}{G^2(c|X)} dF(c|X) &= \int_{0}^{Y} \frac{m(c, X; S)}{G^2(c|X)} dF(c|X) \\ &= \int_{0}^{Y} \frac{m(c, X; S)}{G^2(c|X)} d(-G(c|X)) \end{aligned}$$

For the second step we use the relation $F(\cdot|X) = 1 - G(\cdot|X)$ and that constant terms in the integrator vanish. Next, we use the definition of the cumulative hazard function and that according to the regularity conditions $G(\cdot|X)$ has a density function $g(\cdot|X)$

$$\Lambda_G(c|X) = -\int_0^c \frac{1}{G(u|X)} dG(u|X) = -\int_0^c \frac{g(u|X)}{G(u|X)} du.$$

The fundamental theorem of calculus then gives

$$\frac{d\Lambda_G(c|X)}{dc} = -\frac{g(c|X)}{G(c|X)}.$$

Using this identity and the density of G yields

$$\begin{aligned}
\int_{-\infty}^{Y} \frac{m(c, X; S)}{G^2(c|X)} dF(c|X) &= \int_0^Y \frac{m(c, X; S)}{G^2(c|X)} d(-G(c|X)) \\
&= \int_0^Y \frac{m(c, X; S)}{G(c|X)} \frac{(-g(c|X))}{G(c|X)} dc \\
&= \int_0^Y \frac{m(c, X; S)}{G(c|X)} \frac{d\Lambda_G(c|X)}{dc} dc \\
&= \int_0^Y \frac{m(c, X; S)}{G(c|X)} d\Lambda_G(c|X).
\end{aligned}$$

As a consequence Theorem 5.3 can also be applied for Y^*_{DR} and we obtain that the doubly robust transformation is a CUT if the survival function $S(\cdot|X)$ or the censoring function $G(\cdot|X)$ are specified correctly. □

5.2. Censoring Unbiased Loss Functions

This section follows the ideas presented in Steingrimsson, Diao, and Strawderman (2019) and Steingrimsson and Morrison (2020).

The theory of censoring unbiased transformations can be directly applied to loss functions and results in a class of censoring unbiased loss functions. These loss functions can be computed in the presence and absence of censoring, and reduce to the full data loss if the survival time is not censored.

In this thesis we focus on estimation of the expected loss $\mathcal{R}(\beta) = E[L(h(T), \beta(X))]$, where $\beta(X)$ contains the learning parameters and is estimated using the explanatory variables X. For this we consider two censoring unbiased estimators that both reduce to the full data loss in the absence of censoring: the doubly robust loss function and the Buckley–James estimator. We will not consider the IPCW transformed loss as censored observations are not taken into account apart from

potentially through computation of the censoring function $\hat{G}$. According to Steingrimsson and Morrison (2019) this leads to an inefficient estimator for the full data loss.

The Buckley–James estimator for $\mathcal{R}(\beta)$ is given by

$$L_{BJ}(\mathcal{O}, \beta; S_0) = \frac{1}{n} \sum_{i=1}^{n} \left[\Delta_i L(h(T_i), \beta(X_i)) + (1 - \Delta_i) m_L(C_i, X_i, \beta; S_0) \right]. \quad (5.6)$$

This estimated loss function is the average of the Buckley–James transformed full data loss.

Again, the structure of this estimator is rather intuitive. For uncensored observations we add the full data loss $L(h(T_i), \beta(X_i))$ and for censored observations we use the expectation of the full data loss $m_L(C_i, X_i, \beta; S_0)$ given by equation (5.8).

In order to well-estimate $\mathcal{R}(\beta)$, this loss function requires to find a suitable estimator for the conditional survival function $S_0(\cdot|X)$.

The doubly robust loss function is defined as

$$\begin{aligned} L_{DR}(\mathcal{O}, \beta; G_0, S_0) = & \frac{1}{n} \sum_{i=1}^{n} \frac{\Delta_i L(h(T_i), \beta(X_i))}{G_0(\tilde{T}_i|X_i)} \\ & + \frac{1}{n} \sum_{i=1}^{n} \left(\frac{(1 - \Delta_i) m_L(\tilde{T}_i, X_i, \beta; S_0)}{G_0(\tilde{T}_i|X_i)} \right. \\ & \left. - \int_0^{\tilde{T}_i} \frac{m_L(u, X_i, \beta; S_0)}{G_0(u|X_i)} d\Lambda_{G_0}(u|X_i) \right), \end{aligned} \quad (5.7)$$

where for a survival curve S

$$\begin{aligned} m_L(u, x, \beta; S) &= E_S[L(h(T), \beta(X))|T \geq u, X = x] \\ &= -\int_u^{\infty} \frac{L(h(t), \beta(x))}{S(u|w)} dS(t|w) \end{aligned} \quad (5.8)$$

and $\Lambda_G(u|X) = -\int_0^u 1/G(t|X) dG(t|X)$ is the cumulative hazard function.

Again, we obtain this loss function by computing the average of the full data loss that is now transformed via equation (5.4).

Theorem 5.3 also holds for the doubly robust loss function. As a consequence we only need to model either $S(\cdot|X)$ or $G(\cdot|X)$ correctly in order to obtain a suitable estimator for $\mathcal{R}(\beta)$.

As a next step we can plug in estimators $\hat{S}$ and $\hat{G}$ for the conditional survival functions S_0 and G_0 to obtain empirical estimators for the full data loss.

For the empirical Buckley–James loss we obtain

$$L_{BJ}(\mathcal{O}, \beta; \hat{S}) = \frac{1}{n} \sum_{i=1}^{n} \left(\Delta_i L(h(T_i), \beta(X_i)) + (1 - \Delta_i) m_L(C_i, X_i, \beta; \hat{S}) \right).$$

The empirical doubly robust loss function is given by

$$L_{DR}(\mathcal{O}, \beta; \tilde{G}, \tilde{S}) = \frac{1}{n}\sum_{i=1}^{n} \frac{\Delta_i L(h(T_i), \beta(X_i))}{\tilde{G}(\tilde{T}_i|X_i)} + \frac{1}{n}\sum_{i=1}^{n} \left(\frac{(1-\Delta_i) m_L(\tilde{T}_i, X_i, \beta; \tilde{S})}{\tilde{G}(\tilde{T}_i|X_i)} - \int_0^{\tilde{T}_i} \frac{m_L(u, X_i, \beta; \tilde{S})}{\tilde{G}(u|X_i)} d\Lambda_{\tilde{G}}(u|X_i) \right).$$

If we use the incorrect model specification that $\hat{G}(t|X) = 1$ for all t, the empirical Buckley–James loss is equivalent to the empirical doubly robust loss, i.e. $L_{BJ}(\mathcal{O}, \beta; \hat{S}) = L_{DR}(\mathcal{O}, \beta; \hat{S}, \hat{G} = 1)$.

We can now replace the full data loss function $L(h(T), f(X))$ in Algorithm 4.1 with either the empirical Buckley–James loss L_{BJ} or the empirical doubly robust loss L_{DR} and obtain a prediction model that can be computed with the observed data $\mathcal{O}$. We refer to the resulting algorithms as the Buckley–James or the doubly robust deep learning algorithms respectively. As both loss functions are censoring unbiased loss functions, we refer to the resulting algorithms as censoring unbiased deep learning (CUDL).

5.3. Estimating Survival Probabilities $P(T \geq t|X)$

This section is based on Steingrimsson and Morrison (2020).

In this section we specify how the survival probabilities $P(T \geq t|X)$ can be estimated for a specified time point t. As mentioned before, the target parameter determines the choice of loss function. For the estimation of survival probabilities we can use the Brier loss $L_{t,2}(T, \beta(X)) = (I(T \geq t) - \beta(X))^2$ as our full data loss.

As a first step we have to define a modified dataset with respect to the fixed time point t:

$$\mathcal{O}(t) = \{(\tilde{T}_i(t) = \min(T_i, C_i, t), \Delta_i(t) = I(\min(T_i, t) \leq C_i), X_i); i = 1, \ldots, n\}.$$

Using the Brier loss as the full data loss in Equation (5.7) results in the doubly

robust Brier loss, which is given by

$$\begin{aligned} L_{DR,t}(\mathcal{O}(t), \beta; \hat{G}, \hat{S}) = & \frac{1}{n} \sum_{i=1}^{n} \frac{\Delta_i(t)(I(\tilde{T}_i \geq t) - \beta(X_i))^2}{\hat{G}(\tilde{T}_i|X_i)} \\ & + \frac{1}{n} \sum_{i=1}^{n} \frac{(1 - \Delta_i(t)) m_{L_2,t}(\tilde{T}_i(t), X_i, \beta; \hat{S})}{\hat{G}(\tilde{T}_i(t)|X_i)} \\ & - \int_0^{\hat{T}_i(t)} \frac{m_{L_2,t}(u, X_i, \beta; \hat{S}) d\hat{\Lambda}_G(u|X_i)}{\hat{G}(u|X_i)}, \end{aligned}$$

with

$$m_{L_2,t}(u, X, \beta; S) = E_S[(I(\tilde{T} \geq t) - \beta(X))^2 | T \geq u, X].$$

Analogously, by plugging in the Brier risk to the Buckley–James estimators we obtain the empirical Buckley–James Brier loss given by

$$L_{BJ,t}(\mathcal{O}, \beta; \hat{S}) = \frac{1}{n} \sum_{i=1}^{n} \Big(\Delta_i(t)(I(\tilde{T}_i \geq t) - \beta(X_i))^2 + (1 - \Delta_i(t)) m_{L_2,t}(\tilde{T}_i(t), X_i; \hat{S}) \Big)$$

Both loss functions can now be incorporated in the CUDL algorithm and result in a prediction model for $P(T \geq t|X)$.

5.4. Estimating Mean Survival

This section relies on Steingrimsson and Morrison (2020) and Strawderman (2000).

Another popular goal is to estimate the mean survival $E[T|X]$. For this target parameter we can use the L_2 loss as our full data loss and set $h(t) = t$.

A direct estimation of the mean survival $E[T|X]$ requires strong assumptions. This is why we alternatively estimate the restricted mean survival $E[\min(T, \tau)|X]$ for a fixed constant τ.

Analogously to the estimation of survival probabilities, we need to incorporate the constant τ in the data set. This leads us to the modified version

$$\mathcal{O}(\tau) = \{(\tilde{T}_i(\tau) = \min\{T_i, C_i, \tau\}, \Delta_i(\tau) = I(\min\{T_i, \tau\} \leq C_i), X_i); i = 1, \ldots, n\}.$$

Selecting the L_2 loss as the full data loss, choosing $h(t) = \min t, \tau$ and applying the CUDL algorithms to the modified data set gives us an estimator for the restricted mean survival.

For a more detailed analysis of the estimation of restricted mean survival refer to Strawderman (2000).

5.5. Implementation Based on a Doubly Robust Transformation

This section follows the ideas presented in Steingrimsson and Morrison (2020).

We now describe how we can implement the doubly robust and Buckley–James CUDL algorithm using a specific response transformation.

We use the following response transformation for each subject i with observations $\mathcal{O}_i = (h(\tilde{T}_i), \Delta_i, X_i) \in \mathcal{O}$:

$$D(\mathcal{O}_i; G, S) = A_{1i} + B_{1i} - C_{1i},$$

where for $k = 0, 1, 2$

$$A_{ki} = \frac{\Delta_i h(\tilde{T}_i)^k}{G(\tilde{T}_i|X_i)},$$

$$B_{ki} = \frac{(1-\Delta_i) m_k(\tilde{T}_i, X_i; S)}{G(\tilde{T}_i|X_i)}$$

and

$$C_{ki} = \int_0^{\tilde{T}_i} \frac{m_k(u, X_i; S)}{G(u|X_i)} d\Lambda_G(u|X_i).$$

For $k = 1, 2$

$$m_k(t, X; S) = E_S[h^k(T)|T \geq t, X = x] = -[S(t|x)]^{-1} \int_t^{\infty} [h(u)^k] dS(u|x) \quad (5.9)$$

and $m_0(t, X; S) = 1$ for all t.

The transformation described above is equal to the doubly robust CUT described in Section 5.1 and thus requires that at least one of the models $T|X$ or $C|X$ is correctly specified.

We can now define the response transformed L_2 loss, that uses the censoring unbiased outcome transformation $D(O_i; G, S)$ as the outcome:

$$L_2^*(\mathcal{O}, \beta; G, S) = \frac{1}{n} \sum_{i=1}^{n} (D(O_i; G, S) - \beta(X_i))^2.$$

Our main goal is to show that a deep learning algorithm, that uses the transformed data and the response transformed L_2 loss, is equivalent to the CUDL algorithms presented in Section 5.2. As a result, the deep learning algorithm that uses the transformed data inherits the desired doubly robustness property from the CUDL algorithm.

This equivalence is stated in Theorem 1 in Steingrimsson and Morrison (2020) and reproduced below.

Theorem 5.4. *Under mild regularity conditions, the prediction model created using the CUDL algorithm with the loss function $L_{DR}^{(2)}(\mathcal{O}, \beta; G, S)$ is identical to the prediction model built using the full data deep learning algorithm implemented using the loss function $L_2^*(\mathcal{O}, \beta; G, S)$.*

Proof. Using the notation we introduced earlier in this section, we can rewrite the doubly robust loss function with the L_2 loss as the full data loss as:

$$L_{DR}^{(2)}(\mathcal{O}, \beta; G, S) = \frac{1}{n} \sum_{i=1}^{n} ((A_{2i} + B_{2i} - C_{2i}) - 2(A_{1i} + B_{1i} - C_{1i})\beta(X_i) + (A_{0i} + B_{0i} - C_{0i})\beta(X_i)^2) . \tag{5.10}$$

This equality can be shown by reordering the terms in the sum above in alphabetical order.

Lemma B.1 provided in the appendix states that $A_{0i} + B_{0i} - C_{0i} = 1$. This observation leads us to

$$L_{DR}^{(2)}(\mathcal{O}, \beta; G, S) = \frac{1}{n} \sum_{i=1}^{n} \left((A_{2i} + B_{2i} - C_{2i}) - 2(A_{1i} + B_{1i} - C_{1i})\beta(X_i) + \beta(X_i)^2\right) .$$

We can also reformulate the response transformed L_2 loss by expanding the square and obtain

$$L_2^*(\mathcal{O}, \beta; G, S) = \frac{1}{n} \sum_{i=1}^{n} \left((A_{1i} + B_{1i} - C_{1i})^2 - 2(A_{1i} + B_{1i} - C_{1i})\beta(X_i) + \beta(X_i)^2\right) . \tag{5.11}$$

We can see that the loss functions in Equation (5.10) and Equation (5.11) are actually very similar. Only the first term in the sum, which is independent of β, is different for each loss function.

As a consequence for a fixed penalization parameter η, minimizing

$$L_2^*(\mathcal{O}, \beta; G, S) + \eta \|\beta\|_p^2$$

and

$$L_{DR}^{(2)}(\mathcal{O}, \beta; G, S) + \eta \|\beta\|_p^2$$

results in the same learning parameter β.

If we can show that the penalization parameter selected by minimizing the cross-validated loss with the loss $L_2^*(\mathcal{O}, \beta; G, S)$ is the same if we replace the L_2^* loss by $L_{DR}^{(2)}(\mathcal{O}, \beta; G, S)$, both prediction models produce the identical output.

To show this, we use the notation from Section 3. Recall that $a_{i,\ell}$ indicates whether observation i is an element of the subset K_ℓ.

As $L_2^*(\mathcal{O}, \beta; G, S)$ and $L_{DR}^{(2)}(\mathcal{O}, \beta; G, S)$ are equal up to a term that does not depend on β, it holds that

$$\sum_{\ell=1}^{D} \sum_{i=1}^{n} a_{i,\ell} L_2^*(\mathcal{O}, \beta; G, S) = \sum_{\ell=1}^{D} \sum_{i=1}^{n} a_{i,\ell} L_{DR}^{(2)}(\mathcal{O}, \beta; G, S) + const(\mathcal{O}; G, S).$$

This shows that

$$\begin{aligned} & \underset{m \in \{1,\ldots,M\}}{\arg\min} \sum_{\ell=1}^{D} \sum_{i=1}^{n} a_{i,\ell} L_2^*(\mathcal{O}, \beta; G, S) \\ = & \underset{m \in \{1,\ldots,M\}}{\arg\min} \sum_{\ell=1}^{D} \sum_{i=1}^{n} a_{i,\ell} L_{DR}^{(2)}(\mathcal{O}, \beta; G, S), \end{aligned}$$

which completes the proof. □

The assumptions on the conditional censoring function $G(\cdot|X)$ are flexible enough to allow for $G(t|X) = 1$ for all t. And as it holds that $L_{BJ}(\mathcal{O}, \beta; \hat{S}) = L_{DR}(\mathcal{O}, \beta; \hat{S}, \hat{G} = 1)$, Theorem 5.4 also applies to the CUDL using the Buckley–James loss.

Theorem 5.4 allows us to implement the CUDL algorithm using techniques for fully observed outcomes. This leads us to the following algorithm, as described in Steingrimsson and Morrison (2020):

Algorithm 5.1.

1. Use the observed data $\mathcal{O}$ to compute estimators for the survival curves $\hat{S}(\cdot|\cdot)$ and $\hat{G}(\cdot|\cdot)$
2. Perform the response transformation $D(O_i; \hat{G}, \hat{S})$, $i = 1, \ldots, n$
3. Perform an L_2 full data deep learning algorithm, e.g. Algorithm 4.1, on the dataset $\{(D(O_i; \hat{G}, \hat{S}), X_i); i = 1, \ldots, n\}$.

In the following section we analyze the performance of the algorithm presented above in a simulation study.

6. Simulation Study

In this section we compare the performance of the CUDL algorithms to a Cox PH model, a penalized Cox PH model and a survival tree method using simulated survival data.

For comparison we estimate the conditional survival probability $P(T > t|X)$ for a prespecified time point t.

6.1. Simulation Settings

The different simulation settings follow those presented in Steingrimsson, Diao, and Strawderman (2019).

Setting 1. In our first setting we simulate the dataset such that the proportional hazards assumption is met. To achieve this we create 300 independent observations with 25 explanatory variables $X = (X_1, \ldots, X_{25})$. This vector is multivariate normal with mean zero and a covariance matrix Σ with elements $\sigma_{ij} = 0.9^{|i-j|}$. The survival times are then simulated from an exponential distribution with mean $\mu = \exp\left(0.1 \sum_{i=9}^{18} X_i\right)$. As a consequence, only the explanatory variables $X_9, \ldots, X_{18}$ are relevant for the survival time. The censoring distribution is also simulated from an exponential distribution with mean 0.5 which results in approximately 35% censored data.

Setting 2. In our second setting the proportional hazards assumption is mildly violated. Again, we simulate 300 independent observations with 25 explanatory variables $X = (X_1, \ldots, X_{25})$. This vector now consists of 25 iid uniform random variables on the interval $[0, 1]$. The survival time follows an exponential distribution with mean $\mu = \sin(X_1\pi) + 2|X_2 - 0.5| + X_3^3$. This implies that only the explanatory variables X_1, X_2, X_3 matter for the survival time, the remaining explanatory variables $X_4, \ldots, X_{25}$ are irrelevant. The censoring times are uniformly distributed on the interval $[0, 6]$. This results in approximately 23% censoring.

Setting 3. In this setting, the proportional hazards assumption is strongly violated. Our dataset consists of 300 independent observations and the explanatory variables $X = (X_1, \ldots, X_{25})$ are each multivariate normal with mean zero and covariance matrix Σ with $\sigma_{ij} = 0.75^{|i-j|}$. Now, the survival times are simulated using the gamma distribution with mean $\mu = 0.5 + 0.3|\sum_{i=11}^{15} X_i|$ and scale parameter $\beta = 2$, which results in the shape parameter $\alpha = 2/\mu$. Here we consider that only the explanatory variables $X_{11}, \ldots, X_{15}$ are relevant for the survival time. The censoring times are uniformly distributed on the interval $[0, 15]$. This leads to approximately 30% censoring.

Setting 4. In the last setting we consider the case where the censoring distribution depends on the explanatory variables. Again, we simulate 300 independent observations where the explanatory variables $X = (X_1, \ldots, X_{25})$ are multivariate normal with mean zero and the covariance matrix Σ with elements $\sigma_{ij} = 0.75^{|i-j|}$. The survival distribution is now a log-normal distribution with mean $\mu = 0.1|\sum_{i=1}^{5} X_i| + 0.1|\sum_{i=16}^{20} X_i$. As a consequence the explanatory variables $X_6, \ldots, X_{15}$ and $X_{21}, \ldots, X_{25}$ are irrelevant for the survival time. The censoring times are also simulated according to a log-normal distribution with mean $\tilde{\mu} = \mu + 0.5$ and scale parameter $\lambda = 1$. In this setting approximately 36% of our data are censored.

6.2. Implementation of the CUDL Algorithm

For the implementation we adapted the `R` code of Steingrimsson and Morrison (2020) which is publicly available in one of the authors GitHub repository `https://www.github.com/jonsteingrimsson/CensoringDL`.

6.2.1. Estimating the Survival Curves $S_0(\cdot|\cdot)$ and $G_0(\cdot|\cdot)$

This section follows the ideas presented in Rubin and van der Laan (2007) and Steingrimsson, Diao, Molinaro, and Strawderman (2016).

If the censoring time C is completely independent of both, the event time T and the explanatory variables X, the conditional censoring function $\tilde{G}(\cdot|X) = \tilde{G}(\cdot)$ can be efficiently estimated with the Kaplan–Meier (KM) estimator. The KM estimator for the censoring function is given by

$$\hat{G}(t) = \prod_{i:\, \tilde{T}_{(i)} \leq t} \left(1 - \frac{1}{\#\{j : \tilde{T}_{(j)} \geq \tilde{T}_{(i)}\}}\right)^{1-\Delta_i},$$

where $\tilde{T}_{(1)} < \cdots < \tilde{T}_{(n)}$ are the increasingly ordered observed times.

For example if we consider that censoring is caused by the end of the study, the independence assumption stated above is met and the conditional censoring time can be computed using the KM estimator.

To minimize the required number of assumptions and to cover previously excluded cases, we chose to estimate the censoring survival curve $G_0(t|X)$ using the survival tree method. This tree-based method extends the PH regression to tree-structured relative risk functions. The estimate is implemented in `R` in the following way. First, we fit a survival tree using the function `rpart` with all tuning parameters set to their default values. The function is provided in the package `rpart` and explained in greater detail in Therneau and Atkinson (2019). The tree method classifies the observations according to the explanatory variables in different subgroups. Within

each subgroup a KM estimator is computed for the related data and then the estimators are connected to an overall estimator for the observations. Further details can be found in LeBlanc and Crowley (1992).

The survival curve $S_0(t|X)$ should not be estimated using the KM estimator, as we presume that the observed time $\tilde{T}$ depends on the explanatory variables X. Instead, the survival curve is estimated via a random survival forest procedure. In `R` this estimator can be implemented using the function `rfscr` from the package `randomForestSRC`.

It is possible to choose different estimation techniques for $S_0(\cdot|X)$ and $G_0(\cdot|X)$ respectively.

When choosing the estimators it is important to avoid using an estimator for one of the functions that relies on the estimator of the other. This is because it would negatively impact the doubly robustness property, if we specify one of the estimators incorrectly and thus affect the consistency of the other. Further details can be found in Steingrimsson, Diao, Molinaro, and Strawderman (2016).

6.2.2. Estimating the Conditional Expectation $\mathbf{m_k}$

This section is based on Steingrimsson, Diao, Molinaro, and Strawderman (2016).

For the estimation of the conditional expectation m_k given in equation (5.9), we assume that the observed time is transformed with the function $h(T) = I(T > t)$. Using this and plugging in the estimator of the survival curve $\hat{S}(\cdot|X)$ results in

$$\begin{aligned}\hat{m}_{k,t}(u, X_i; \hat{S}) &= E_{\hat{S}}[h^k(T)|T \geq u, X = X_i] \\ &= -\frac{1}{\hat{S}(u|X_i)} \int_u^\infty I(r > t)^k d\hat{S}(r|X_i) \\ &= -\frac{1}{\hat{S}(u|X_i)} \int_u^\infty I(r > t) d\hat{S}(r|X_i),\end{aligned}$$

for $k = 1, 2$. Because $h(T)$ is an indicator function, the conditional expectation does no longer depend on k. In this thesis, when estimating the survival curve we always use discrete estimators which have jumps at the finite observed event times $0 = T_{(0)} < T_{(1)} < \cdots < T_{(n)}$. As a consequence the integral is in fact a finite sum, given by

$$\hat{m}_t(u, X_i; \hat{S}) = -\frac{1}{\hat{S}(u|X_i)} \sum_{i=1}^{n} I(T_{(i)} > t)(T_{(i)} - T_{(i-1)}),$$

which can be implemented in `R`.

6.2.3. Selecting Truncation Time to Ensure Positivity

This section relies on Steingrimsson, Diao, Molinaro, and Strawderman (2016).

For the methods described in Section 5 it is necessary to ensure that an empirical version of the positivity assumption holds meaning that for some $\varepsilon > 0$ it holds for the empirical censoring function that $\hat{G}(t|X) \geq \varepsilon$. When computing the IPCW and the doubly robust mapping, having a denominator too close to zero can result in unstable finite sample performance. This is even the case if the empirical positivity assumption holds.

To ensure that the empirical positivity assumption holds and that the empirical censoring function is not too small a truncation time $\tau > 0$ is introduced. The general idea of truncation is to cut off the data after the truncation time τ meaning that for each observed survival time we set $\tilde{T}_i(\tau) = \min(\tilde{T}_i, \tau)$. There are different methods of truncation depending on how the event indicator Δ_i is changed and how the truncation time is selected. One possible approach is to set

$$\Delta_i(\tau) = \Delta_i I(\tilde{T}_i \leq \tau)$$

for $i = 1, \ldots, n$ and choose the truncation time as the 95% quantile of the observed times. In this setting we interpret each observed time that has been cut off as censored and keep the event indicator for data that has not been changed. The choice of truncation time leads to 5% truncated data.

A different approach, argued by Steingrimsson, Diao, Molinaro, and Strawderman (2016) on the basis of superior empirical results, is to set

$$\Delta_i(\tau) = \Delta_i I(\tilde{T}_i \leq \tau) + I(\tilde{T}_i > \tau)$$

for $i = 1, \ldots, n$ and choose the truncation time as the 90% quantile. In this scenario we treat each subject whose observed survival time exceeds the truncation time as if they experienced the event at time τ.

For the implementation we choose the truncation method as proposed by Steingrimsson, Diao, Molinaro, and Strawderman (2016).

6.2.4. Preparation of Survival Data

We now describe how the transformation of the responses $\tilde{T}$ described in Section 5.5, namely the Buckley–James and the doubly robust transformation, can be implemented in `R`.

For the Buckley–James transformation we estimate the conditional expectation $\hat{m}$ using the random forest method with the Brier loss as full data loss. The survival function is then computed with a survival tree, as described in Section 6.2.1. In this case, the transformed response is given by

$$Y^*_{BJ}(\mathcal{O}(t)) = \Delta(t) I(\tilde{T} > t) + (1 - \Delta(t)) \hat{m}(\tilde{T}, X; \hat{S}),$$

which equals the Buckley–James transformation given in equation (5.1) with the observations $\mathcal{O}(t)$.

For the doubly robust transformation given in equation (5.4) we have to compute A_{1i}, B_{1i} and C_{1i}. Again, we can plug in $h(\tilde{T}) = I(\tilde{T} > t)$, the estimated conditional expectation $\hat{m}$ and the estimated censoring function $\hat{G}$ and obtain

$$A_{1i}(t) = \frac{\Delta_i(t) I(\tilde{T}_i > t)}{\hat{G}(\tilde{T}_i | X_i)},$$

$$B_{1i}(t) = \frac{(1 - \Delta_i(t)) \hat{m}(h(\tilde{T}_i), X_i; \hat{S})}{\hat{G}(\tilde{T}_i | X_i)}$$

and

$$C_{1i}(t) = \int_0^{\tilde{T}_i} \frac{\hat{m}(h(u), X_i; \hat{S})}{\hat{G}(u | X_i)} d\Lambda_{\hat{G}}(u | X_i).$$

The parameter $C_{1i}(t)$ can again be computed using a finite sum.

For comparison we split each dataset in a training and a test set. Analogous to Steingrimsson and Morrison (2020) we randomly chose 20% of the data as our test set. In order to preserve the ratio of censored and uncensored data we draw the 20% for the test set from the censored and uncensored data separately.

6.2.5. Competing Methods

In this section we describe how the deep learning model from Algorithm 5.1 can be implemented in `R`. Then we explain three different methods for estimation of survival time that we use to compare the performance of the CUDL algorithms.

To predict the survival probabilities with deep learning we use a feedforward network with two layers. The first layer consists of 15 units and uses the ReLU activation function $\phi(x) = \max(0, x)$. As proposed in Goodfellow, Bengio, and Courville (2016) we use the dropout rate $1 - p = 0.2$ for the input layer. The second layer, which is our output layer, consists of one unit and uses the sigmoid activation function $\phi(x) = (1 + e^{-x})^{-1}$. We choose this activation function to make sure that the resulting value can be interpreted as a probability.

For the gradient descent we use the mean squared error as loss function, the batch size is set to 32 and we use 100 epochs. In one epoch we go through the whole dataset in batches of 32, which means that we perform $(\#\text{observations})/32 \cdot 100$ steps to find the optimal parameters. Then we perform 5-fold cross validation to choose a suitable penalization term η from the sequence $(0, 0.001, 0.01, 0.1)$. As proposed by Steingrimsson and Morrison (2020) the performance of the resulting penalized feedforward network is evaluated with an IPCW weighted mean squared error. This error is computed estimating the IPCW weights which are described in Equation (5.3) and multiplying them with the mean squared error.

The procedure described above is performed twice. For the observed times that are transformed using the Buckley–James transformation and for the doubly robust transformed observed times.

We compare the deep learning algorithms that use the transformed data to three different approaches: a Cox PH model, a penalized Cox model and a random forest model. As a first step, we transform the responses $Y = h(T) = I(T > t)$, with the prespecified time point t to account for the goal of estimating survival probabilities.

The Cox PH model, which is explained in further detail in Section 2.3, can be estimated in `R` using the function `coxph` from the package `survival`. Refer to Therneau (2021) for further information. We then create the corresponding survival curve using the function `survfit`.

The second model, which is the penalized Cox model, is very similar to the Cox PH model. As a first step, a cross validated generalized linear model is evaluated to determine the explanatory variables that need to be included in the model to minimize the mean cross-validated error. These explanatory variables are then used to fit a Cox PH model in the same way as described earlier in this section.

As a third model we consider a random forest model. This can be implemented using the `R` function `rfscr` from the package `randomForestSRC`. The resulting survival curve is estimated using the function `predict`.

The performance of all of the above mentioned models is again evaluated using an IPCW weighted mean squared error.

6.3. Simulation Results

In this section we compare the different methods for survival analysis, which are described in Section 6.2.5 for each of the settings presented in Section 6.1.

First the data is generated according to the current setting. For the CUDL algorithms the observed time is using the Buckley–James and the doubly robust transformation respectively. Then the 300 observations are randomly split in a training and test set where the test set contains 20% of the data. The different methods are fit using the training data and evaluated with the IPCW weighted error. We repeat this procedure 50 times and compare the resulting errors.

For predicting the survival probability $P(T > t|X)$ we have to specify the time t. Analogous to Steingrimsson and Morrison (2020) we used the median survival time for each setting. We computed data according to the respective simulation setting 1000 times, computed the medians of the resulting survival times, and averaged them.

The resulting prediction errors are visualized using raincloud plots. These plots extend boxplots by the empirical distribution of the prediction error. Table 6.1 summarizes the 10% and 90% quantile, mean, and standard deviation for each model in each setting.

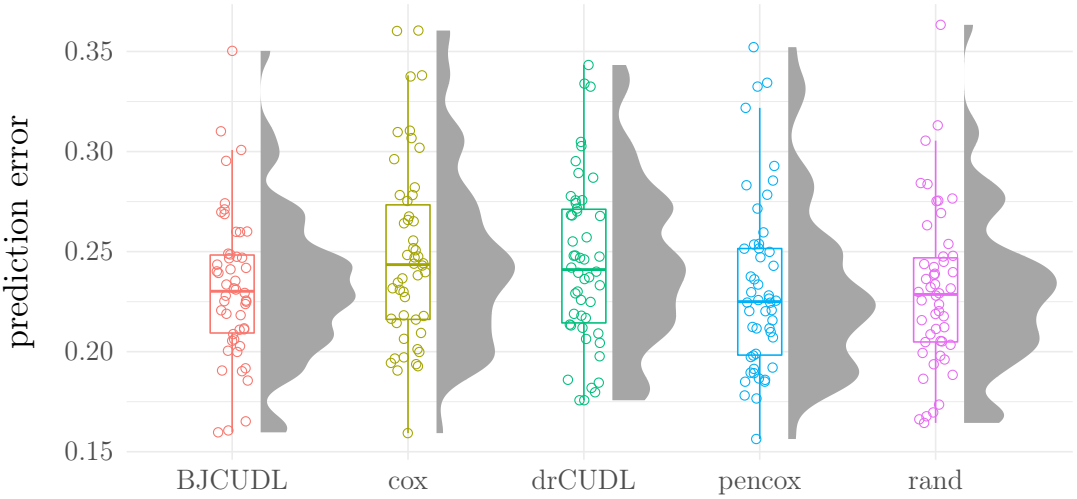


Figure 6.1: Comparison of the resulting IPCW weighted mean squared error for Setting 1. Lower values indicate better performance. BJCUDL and drCUDL are the CUDL algorithms that use the Buckley–James and the doubly robust transformation respectively. The Cox PH model and the penalized Cox PH model are denoted by cox and pencox. Rand is the random forest procedure.

In Setting 1 we simulate the survival times such that the PH assumption is met. We used the empirically computed median $t = 0.4190$ to estimate the survival probability $P(T > t|X)$. In Figure 6.1 the resulting prediction errors are illustrated. Each of the considered models gives comparably good results with mean prediction error of approximately 24% and the 10% quantile at approximately 19%. The prediction error of the CUDL algorithm that uses the Buckley–James transformation has a lower 90% quantile and standard deviation compared to the other methods.

In Setting 2 we simulate the survival times such that the PH assumption is mildly violated. We used the empirically computed median $t = 0.7046$ to estimate the survival probability. As seen in Table 6.1 the mean prediction error for each method is approximately 28%. The Buckley–James CUDL results with approximately 1.6% in the lowest standard deviation. Additionally we can see in Figure 6.2 that the prediction errors of the Buckley–James CUDL algorithm are denser distributed around the median than for the doubly robust CUDL algorithm. The IPCW weighted mean squared error of the doubly robust CUDL algorithm has a 10% quantile that lies beneath the other methods, with a value of approximately 24%. The Cox PH model performs worse than the other models. This can be attributed to the PH assumption not being met.

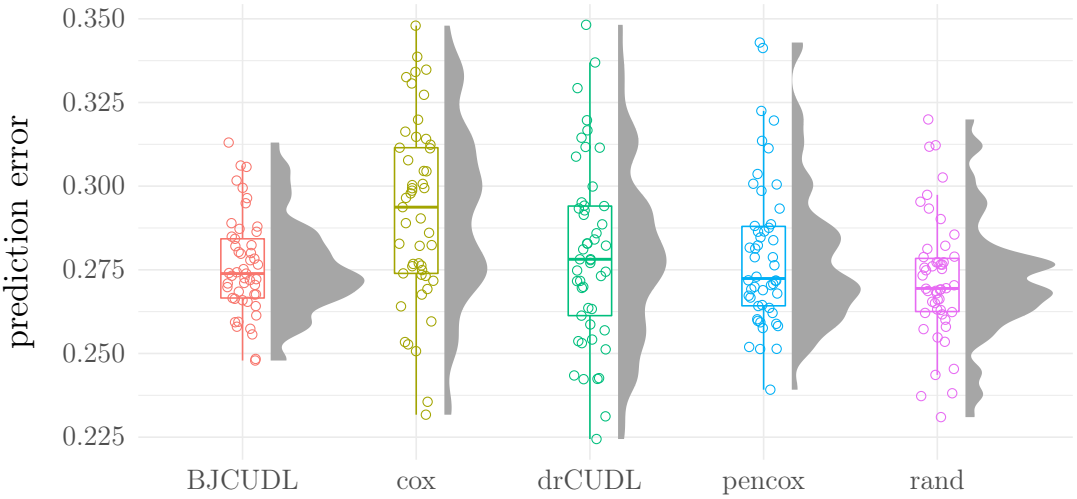


Figure 6.2: Comparison of the resulting IPCW weighted mean squared error for Setting 2. Lower values indicate better performance. The names of the different models are as in Figure 6.1.

In Setting 3 we simulate the survival times such that the PH assumption is strongly violated. We used the empirically computed median $t = 2,9072$ to estimate the survival probabilities. In Figure 6.3 we can see that all methods give similar results. The distribution of the prediction errors is dense around the median and has a nontrivial tail extending towards 0.07. According to the values in Table 6.1 the Cox PH model performs worst on average.

In Setting 4 we simulate the survival times such that the censoring distribution depends on the explanatory variables. We used the empirically computed median $t = 1,0349$ to estimate the survival probabilities. The resulting prediction errors are visualized in Figure 6.4. The penalized Cox PH model has a mean prediction error of approximately 28% and standard deviation of approximately 4%, thus performing better than the standard Cox PH model. The remaining model perform similarly to the penalized Cox PH model.

Overall the simulation study suggests that the Buckley–James and doubly robust CUDL algorithms perform similarly well as the other methods. The two algorithms showed superior performance in Setting 4 where we simulated the data such that the censoring distribution depends on the explanatory variables.

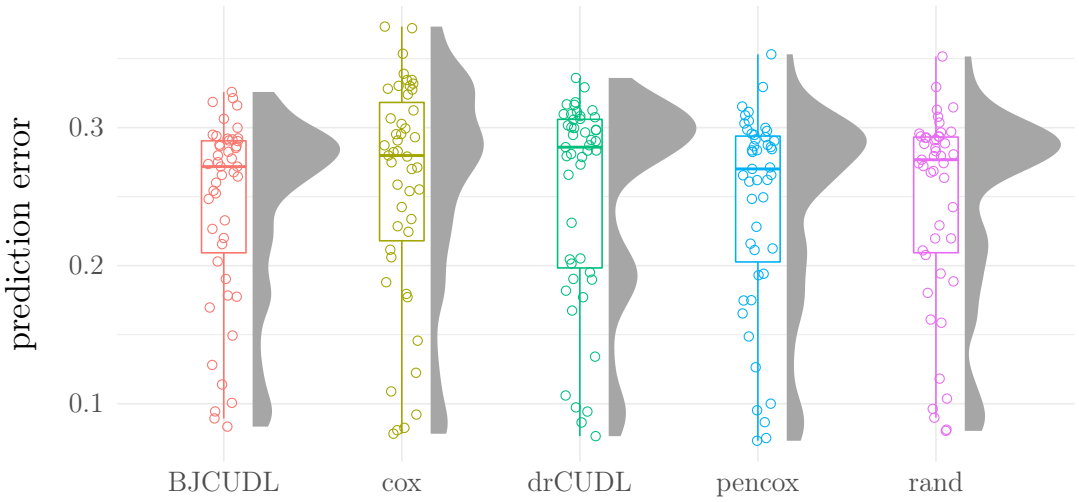


Figure 6.3: Comparison of the resulting IPCW weighted mean squared error for Setting 3. Lower values indicate better performance. The names of the different models are as in Figure 6.1.

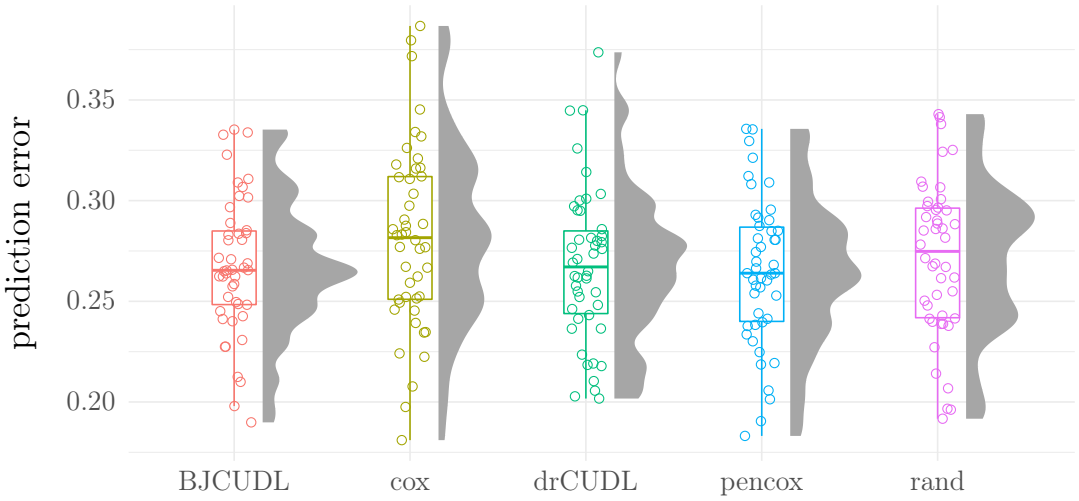


Figure 6.4: Comparison of the resulting IPCW weighted mean squared error for Setting 4. Lower values indicate better performance. The names of the different models are as in Figure 6.1.

algorithm	10% quantile	mean	90% quantile	standard deviation
		Setting 1		
BJCUDL	0.1905	0.2322	**0.2714**	**0.0372**
cox	0.1964	0.2485	0.3098	0.0458
drCUDL	0.1858	0.2444	0.2960	0.0414
pencox	0.1858	0.2319	0.2862	0.0436
rand	**0.1852**	**0.2301**	0.2773	0.0401
		Setting 2		
BJCUDL	0.2592	0.2771	0.2997	**0.0163**
cox	0.2590	0.2939	0.3327	0.0292
drCUDL	**0.2434**	0.2828	0.3170	0.0349
pencox	0.2581	0.2795	0.3115	0.0224
rand	0.2527	**0.2719**	**0.2955**	0.0183
		Setting 3		
BJCUDL	0.1000	**0.2375**	**0.3074**	**0.0810**
cox	**0.0912**	0.2507	0.3350	0.0925
drCUDL	0.0970	0.2441	0.3166	0.0868
pencox	0.0943	0.2377	0.3092	0.0867
rand	0.0956	0.2399	0.3084	0.0850
		Setting 4		
BJCUDL	0.2122	0.2683	0.3238	**0.0422**
cox	0.2210	0.2844	0.3479	0.0610
drCUDL	0.2099	0.2705	0.3278	0.0498
pencox	**0.2053**	**0.2650**	**0.3221**	0.0435
rand	0.2058	0.2716	0.3265	0.0463

Table 6.1: Comparison of 10% and 90% quantile, mean and standard deviation of the prediction error of the models for each of the settings described above. The names of the different models are chosen as in Figure 6.1.

7. Application to a Real World Scenario

In this section we analyze the performance of the models described in Section 6.2 when applied to a real world scenario.

7.1. Mayo Clinic Primary Biliary Cholangitis Data

The Mayo Clinic primary biliary cholangitis data can be found in the `R` package `survival` and is provided by Terry M. Therneau and Patricia M. Grambsch (2000).

Primary biliary cholangitis (PBC) is an autoimmune disease that leads to demolition of the small bile ducts in the liver. The disease progresses slow but is inexhortable and can lead to cirrhosis and liver decompensation. The data originates from the Mayo Clinic trial in PBC that was carried out between 1974 and 1984. In this ten year interval a total of 312 PBC patients took part in the randomized placebo controlled trial of the drug D-penicillamine. Additional 106 cases did not participate in the clinical trial but were followed for survival. This results in data of 418 individuals.

For further computations we exclude cases with missing explanatory variables. This leads to 276 observations that can be used for our models.

In the data set the survival time of each individual is given by the variable `time`. It describes the number of days between registration and either death, transplantation or the end of the study and takes values between 41 and 4795. The `status` variable is an event indicator for three possible cases, either the patient was right censored, received a transplant or experienced the event meaning that the subject died. As we only consider one event instead of two, each patient that received a transplant is treated as censored. This leads to approximately 59.78% censoring.

The data set contains 17 explanatory variables, which are age in years, sex, a treatment variable that indicates if the patient was in the placebo or treatment group or was not part of the clinical trial and various other medical features that might influence the survival time. All of the given explanatory variables are transformed into numerical values.

As a next step we added uniformly distributed noise in the range $[0, 0.1]$ to assure that each observed survival time is different. This is important for the estimation of the survival and censoring functions, as estimators like the KM estimator require the observed times to be distinct.

We estimate the survival probabilities $P(T > t|X)$ at 50 equidistant time points t on the interval $[850, 4500]$. The lower bound of this interval was chosen to ensure that at least five subjects in the dataset were censored beforehand. This prevents the estimation procedures described in Section 6.2 from using a constant estimator for the censoring function.

algorithm	10% quantile	mean	90% quantile	standard deviation
BJCUDL	0.1263	0.3257	0.5554	0.1584
cox	0.0756	0.1572	0.3011	0.0872
drCUDL	0.1812	0.3707	0.5504	0.1549
pencox	**0.0679**	0.1544	0.3022	0.0891
rand	0.0730	**0.1532**	**0.2491**	**0.0761**

Table 7.1: Comparison of 10% and 90% quantile, mean and standard deviation of the prediction error of the models for the Mayo clinic primary biliary cholangitis data. The names of the models are chosen as in Figure 7.1.

For each time point we randomly split the data into a training and test set and compute the prediction error of the five models described in Section 6.2 to compare their performance. As proposed by Steingrimsson and Morrison (2020) we use 20% of the data as the test set.

7.2. Results

In this section we compare the IPCW weighted mean squared error of the different models.

In Figure 7.1 the resulting prediction errors are visualized using raincloud plots. Table 7.1 summarizes the 10% and 90% quantile, mean, and standard deviation for each model that was used to evaluate the PBC data.

The Mayo clinic primary biliary cholangitis data satisfies the PH assumption. Which can be tested for using the `R` function `cox.zph` from the `survival` package. This function is based on Grambsch and Therneau (1994).

According to the simulations in Section 6.3, the Cox PH model, the penalized Cox PH model, and the random forest procedure should yield better results if the PH assumption is met. As can be seen in Figure 7.1, the aforementioned observation also holds for the real world scenario. While the Cox PH model, the penalized Cox PH model, and the random forest procedure result in a mean prediction error of approximately 15%, the Buckley–James and the doubly robust CUDL algorithm have a mean prediction error of approximately 33% and 37% respectively. The standard deviation of the deep learning approaches is with approximately 15% almost twice the standard deviation of the remaining models. An analysis of the highest prediction errors of the Buckley–James and doubly robust CUDL suggests that the deep learning approaches struggle with times points t that are close to the maximum and minimum values.

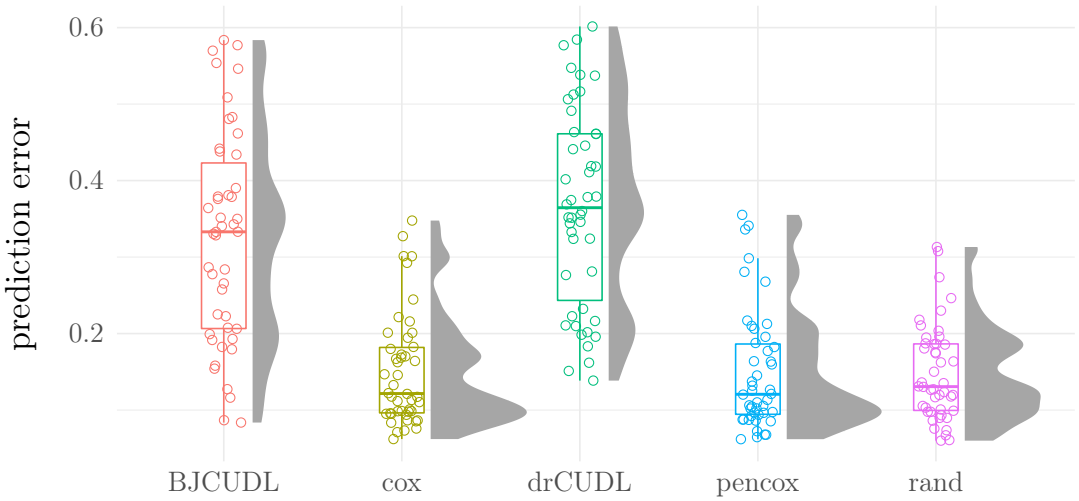


Figure 7.1: Comparison of the resulting IPCW weighted mean squared error for the PBC data. Lower values indicate better performance. BJCUDL and drCUDL are the CUDL algorithms that use the Buckley–James and the doubly robust transformation respectively. The Cox PH model and the penalized Cox PH model are denoted by cox and pencox. Rand is the random forest procedure.

8. Conclusion

In this thesis we discussed how survival data can be implemented using deep learning algorithms as feedforward networks. First we briefly described the main concepts of survival analysis and deep learning. Then we analyzed how feedforward networks can be applied to uncensored data.

In the main part we considered censored data and studied how existing algorithms can be extended to survival data. To achieve this we first introduced the concept of censoring unbiased transformations which preserve the conditional mean structure of the data. Two important examples for censoring unbiased transformations are the Buckley–James and the doubly robust transformation. These transformations were then applied to full data loss functions which resulted in the class of censoring unbiased deep learning algorithms. Then we proved that these algorithms are equivalent to deep learning algorithms that use the full data loss with transformed data as input.

As a next step we evaluated the performance of these deep learning algorithms in four different settings. To conclude this thesis we also applied the methods to the Mayo clinic primary biliary cholangitis data. Our implementations yielded similar results as in Steingrimsson and Morrison (2020). If the PH assumption is met the Cox PH model gives reliable results. However, if the data is more complex it can be beneficial to use deep learning algorithms instead.

Exploring the space of hyperparameters for the deep learning models is a computationally demanding task. This is why both Steingrimsson and Morrison (2020) and we opted to not optimize the hyperparameter values and used default values instead. It would be interesting to analyze whether the performance of the deep learning algorithms could be substantially increased by better hyperparameters.

In this thesis we considered a single event that can occur at most once. For further research it could be interesting to study how the transformation can be extended to competing risks or recurrent events.

References

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning* (first ed.). The MIT Press.

Grambsch, P. and T. Therneau (1994). Proportional hazards tests and diagnostics based on weighted residuals. *Biometrika 81*.

Heiss, J., J. Teichmann, and H. Wutte (2019, 11). How implicit regularization of neural networks affects the learned function – part i.

Kalbfleisch, J. D. and R. L. Prentice (2002). *The statistical Analysis of Failure Time Data* (second ed.). John Wiley & Sons.

Klein, J. P. and M. L. Moeschberger (2003). *Survival Analysis: Techniques for Censored and Truncated Data* (second ed.). Springer.

Kleinbaum, D. G. and M. Klein (2020). *Survival Analysis* (third ed.). Springer Science + Business Media.

LeBlanc, M. and J. Crowley (1992). Relative risk trees for censored survival data. *International Biometric Society Vol. 48, No. 2*, 411–425.

Rubin, D. and M. J. van der Laan (2007). A doubly robust censoring unbiased transformation. *The International Journal of Biostatistics Vol. 3*, Iss. 1, Article 4.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014, 06). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15*, 1929–1958.

Steingrimsson, J. A., L. Diao, A. M. Molinaro, and R. L. Strawderman (2016). Doubly robust survival trees. *Statistics in Medicine 35*.

Steingrimsson, J. A., L. Diao, and R. L. Strawderman (2019). Censoring unbiased regression trees and ensembles. *Journal of the American Statistical Association 114,525*, 370–383.

Steingrimsson, J. A. and S. Morrison (2019). Deep learning for survival outcomes. https://arxiv.org/abs/1904.10345, Submitted on 23 Apr 2019.

Steingrimsson, J. A. and S. Morrison (2020). Deep learning for survival outcomes. *Statistics in Medicine 39 (17)*, 2339–2349.

Strawderman, R. L. (2000). Estimating the mean of an increasing stochastic process at a censored stopping time. *Journal of the American Statistical Association 95, No. 452*, 1192–1208.

Terry M. Therneau and Patricia M. Grambsch (2000). *Modeling Survival Data: Extending the Cox Model*. New York: Springer.

Therneau, T. and B. Atkinson (2019). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-15.

Therneau, T. M. (2021). *A Package for Survival Analysis in R*. R package version 3.2-13.

Appendices

A. Regularity Conditions

As proposed in Theorem 3.1 in Rubin and van der Laan (2007) we make the following assumptions. Suppose that the observed time $\tilde{T}$ and the censoring time C are continuous random variables, that they are conditionally independent given the explanatory variables X, and that the conditional distribution $F_0(\cdot|X) = 1-G_0(\cdot|X)$ has a conditional density $f_0(\cdot|X)$. Assume that $\tilde{T} \leq \tau < \infty$ for some τ and that $\tilde{S}(\tau|X) = 0$ with probability one for some conditional function $\tilde{S}(\cdot|X)$. Suppose further that $\tilde{G}(\tau|X) \geq \varepsilon > 0$ for some ε and conditional survival function $\tilde{G}(\cdot|X)$, with corresponding conditional density $\tilde{g}(\cdot|X)$. Assume that $\tilde{g}(\cdot|X)$ is absolutely continuous with respect to $g_0(\cdot|X)$ for all X. We will use the convention that $m(t, X; \tilde{S})$ is set to zero if $\tilde{S}(t|X) = 0$.

B. Proofs

B.1. Proof of Theorem 5.1

Theorem 5.1. *Under the mild regularity conditions given in Appendix A it holds that $Y^*_{BJ}(\cdot; S_0)$ is a CUT.*

Proof. The transformation $Y^*_{BJ}(\mathcal{O}; S_0)$ is a CUT if $E[Y^*_{BJ}(\mathcal{O}; S_0)|X] = E[Y|X]$ holds.

Because of the linearity of expectation we can analyze each term in $Y^*_{BJ}(\mathcal{O}; S_0)$ separately. For the first term we obtain:

$$\begin{aligned} E[\Delta Y|X] &= E[E[\Delta Y|Y, X]|X] = E[Y P(\Delta = 1|Y, X)|X] \\ &= E[Y G(Y|X)|X] = \int_0^\tau y G(y|X) d(1 - S(y|X)) \\ &= -\int_0^\tau y G(y|X) dS(y|X) \end{aligned}$$

For the second term we obtain:

$$\begin{aligned}
E[(1-\Delta)m(Y,X;S)] &= E[E[(1-\Delta)m(Y,X;S)|Y,X]|X] \\
&= E[m(Y,X;S)P(\Delta=0|Y,X)|X] \\
&= E[m(Y,X;S)S(Y|X)|X] \\
&= E\left[-\frac{S(Y|X)}{S_0(Y|X)}\int_Y^\tau y dS_0(y|X)\right] \\
&= \int_0^\tau \frac{S(c|X)}{S_0(c|X)}\int_Y^\tau y dS_0(y|X) dG(c|X) \\
&= \int_0^\infty \int_0^\infty \frac{S(c|X)}{S_0(c|X)} I(c<y<\tau) y dS_0(y|X) dG(c|X) \\
&= \int_0^\tau y \int_0^y \frac{S(c|X)}{S_0(c|X)} dG(c|X) dS_0(y|X)
\end{aligned}$$

Combining both and setting $S = S_0$ yields

$$\begin{aligned}
E[Y^*_{BJ}(\mathcal{O};S_0)|X] &= E[\Delta Y|X] + E[(1-\Delta)m(Y,X;S_0)] \\
&= -\int_0^\tau yG(y|X)dS_0(y|X) + \int_0^\tau y\int_0^y \frac{S_0(c|X)}{S_0(c|X)} dG(c|X)dS_0(y|X) \\
&= -\int_0^\tau y\left[G(y|X) - \int_0^y dG(c|X)\right] dS_0(y|X) \\
&= -\int_0^\tau y\left[G(y|X) - (G(y|X)-1)\right] dS_0(y|X) \\
&= -\int_0^\tau y dS_0(y|X) \\
&= E[Y|X].
\end{aligned}$$

□

B.2. Auxiliary Lemma for Theorem 5.4

Lemma B.1. *Using the notation introduced in Section 5.5, it holds that*

$$A_{0i} + B_{0i} - C_{0i} = 1$$

Proof. By plugging in the definitions of A_{0i}, B_{0i} and C_{0i} respectively, we obtain

$$A_{0i} + B_{0i} - C_{0i} = \frac{\Delta_i}{G(T_i|X_i)} + \frac{1-\Delta_i}{G(T_i|X_i)} - \int_0^{T_i} \frac{1}{G(u|X_i)} d\Lambda_G(u|X_i)$$

We can now combine the first two terms to obtain

$$A_{0i} + B_{0i} - C_{0i} = \frac{1}{G(T_i|X_i)} - \int_0^{T_i} \frac{1}{G(u|X_i)} d\Lambda_G(u|X_i). \tag{B.1}$$

This formulation allows us to apply Lemma 1 from Strawderman (2000). Using the notation introduced in this thesis, the aforementioned lemma gives us the following equation:

$$\frac{\Delta_i^*}{G(T_i|X_i)} = 1 - \int_0^\infty \frac{1}{G(s|X_i)} dM^*(s), \tag{B.2}$$

with

$$M^*(s) = I\{T_i \leq s, \Delta_i^* = 0\} - \int_0^s I\{T_i \geq u\} d\Lambda_G(u|X_i),$$

and Δ^* is any arbitrary indicator function.

Now we set $\Delta_i^* = 1$ for all $i = 1, \ldots, n$ and take a closer look at the definition of the function $M^*(s)$.

$$\begin{aligned} M^*(s) &= I\{T_i \leq s, \Delta_i^* = 0\} - \int_0^s I\{T_i \geq u\} d\Lambda_G(u|X_i) \\ &= -\int_0^s I\{T_i \geq u\} d\Lambda_G(u|X_i) \\ &= -\int_0^{min\{T_i, s\}} d\Lambda_G(u|X_i) \\ &= -(\Lambda_G(\min(T_i, s)|X_i) - \Lambda_G(0|X_i)) \\ &= -\Lambda_G(\min(T_i, s)|X_i) \end{aligned}$$

We plug in our choice of Δ_i^* and $M^*(s)$ to equation (B.2) and obtain

$$\begin{aligned} \frac{1}{G(T_i|X_i)} &= 1 + \int_0^\infty \frac{1}{G(s|X_i)} d\Lambda_G(\min(T_i, s)|X_i) \\ &= 1 + \int_0^{T_i} \frac{1}{G(s|X_i)} d\Lambda_G(s|X_i) + \int_{T_i}^\infty \frac{1}{G(s|X_i)} d\Lambda_G(T_i|X_i) \\ &= 1 + \int_0^{T_i} \frac{1}{G(s|X_i)} d\Lambda_G(s|X_i) \end{aligned} \tag{B.3}$$

Combining equation (B.1) and (B.3) then yields:

$$\begin{aligned} A_{0i} + B_{0i} - C_{0i} &= \frac{1}{G(T_i|X_i)} - \int_0^{T_i} \frac{1}{G(u|X_i)} d\Lambda_G(u|X_i) \\ &= 1 + \int_0^{T_i} \frac{1}{G(s|X_i)} d\Lambda_G(s|X_i) - \int_0^{T_i} \frac{1}{G(u|X_i)} d\Lambda_G(u|X_i) \\ &= 1. \end{aligned}$$

□

C. R Code

The `R` code that is used for the simulation study and the analysis of the PBC data is provided on the attached cd. It mainly relies on the `R` code of Steingrimsson and Morrison (2020) which is publicly available in one of the authors GitHub repository `https://www.github.com/jonsteingrimsson/CensoringDL`. It consists of six `R` script files that are briefly described below.

1. **create-data.R:** This script file contains the implementation of the four simulation settings described in Section 6.1 as well as the procedure that splits the data in the training and test set.

2. **transform-data.R:** In this script file the Buckley–James and the doubly robust transformations are implemented. It also contains the computation of the IPCW weights that are used for our performance measure.

3. **CUDLalgorithms.R:** This `R` script contains the implementation of the CUDL algorithms and is further described in Section 6.2.5.

4. **competing-methods.R:** In this script file the Cox PH model, the penalized Cox PH model, and the random forest procedure are implemented as described in Section 6.2.5.

5. **auxiliary-functions.R:** This script contains the implementation of the estimation procedures for the censoring function and the conditional expectation as described in Sections 6.2.1 and 6.2.2. The computation of the parameters that are needed for the transformation is also implemented in this script. It also contains a function that truncates the data as described in Section 6.2.3.

6. **runSimulation.R:** This `R` script file calls the methods described above to run the simulation.