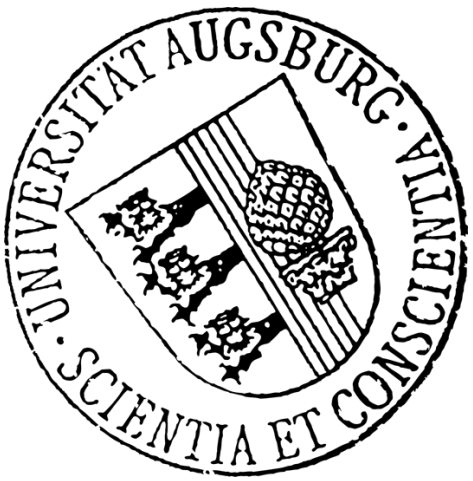
# Todo list

Master Thesis

University 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

- growing importance of machine learning in many areas, as medicine - many different approaches - problem: treatment of missing data - need specialized techniques as in contrast to standard regression or classification tasks the outcome can for example often not be fully observed

# 2 Principles of Survival Analysis

This section is based on Kleinbaum and Klein (2020) and Klein and Moeschberger (2003).

In this section 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. An application inside the medical field could be a study about the time to death after the outbreak of a disease.

add more about the general setting (time series of data, consider medical cases as examples, ...)

add explanation of true/observed survival time

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

## 2.1 Censoring

This section relies on 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 which is the time when the event occurs, but instead we observe the censoring time which is the time from which on the censored subject is no longer examined.

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. Assuming data arising 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 all lead to *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 arise due to subjects that already experienced the event of interest before entering the study. 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 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, as this would both lead 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.

In this thesis 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 random within any subgroup of interest. The third assumption, namely that censoring is non-informative, 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).

passende Stelle für censoring assumptions finden

## 2.2 Survival Analysis Data

This section mainly relies 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 $T_i \geq 0$ one realization of the random variable for the true survival time of subject $i$ and by $C_i \geq 0$ one realization of the random variable for the censoring time of individual $i$. 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 and given by the vector $X_i = (X_{i1}, \ldots, X_{ip}) \in \delta \subset \mathbb{R}^p$, where $\delta$ 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 $n$ is the number of patients included in the study.

For analysis and prediction we often use two important functions: the survivor function and the hazard function. The survivor function $S(t) = P(T > t)$ specifies the distribution of the survival time $F(t) = 1 - S(t)$. It can for example be used to analyze and compare the time to event of two different groups. The survivor function has three important properties: $S(t)$ is a non-increasing function, $S(0) = 1$ and $S(\infty) = 0$. These properties follow directly from the relation $F(t) = 1 - S(t)$ and that $F$ is a proper distribution function.

The hazard function $h(t)$, 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 survivor function $S(\cdot)$ the hazard function does not necessarily have any monotonic properties.

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

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

As a consequence, specifying one of the two function directly gives us the remaining.

## 2.3 Parametric and Semiparametic 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. These parameters

can for examples be estimated using 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 of the main advantages of the parametric survival model are its simplicity, that it is consistent with the theoretical survival function $S(t)$ 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 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 following assumption about the form of the hazard function:

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

where $x = (x_1, \ldots, x_p)$ are the explanatory variables. This includes the central assumption of the Cox PH model, which is the proportional hazards (PH) assumption. It states that there exists a function $h_0(t)$, the baseline hazard, that does not involve the predictor variables and is thus independent of the observations.

As the baseline hazard $h_0(t)$ is not specified, the Cox PH model is a semiparametric model. Typically, maximum likelihood is used to estimate the parameters $\beta_i$, $i = 1, \ldots, n$.

Sätze besser verbinden bzw mehr Kontext

In contrast to parametric models, the Cox PH model does not require an assumption about the distribution of the outcome. Another important advantages 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 correctly, parametric models give the best estimate. But if we are in doubt that the model is correct, the Cox PH model gives 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).

If the PH assumption is not fulfilled, we can for example use a stratified Cox procedure instead, which is 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, that can then be combined to an overall likelihood. This approach is explained in greater detail in Kleinbaum and Klein (2020).

There exist various other modifications of the Cox PH model.

add penalized Cox model

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. But first, we introduce the main ideas of deep learning and explain the idea of feed forward networks.

# 3 Principles of Deep Learning

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

allgemeines zu ML

Machine Learning algorithms consist of a task and a performance measure.

- classified according to what kind of experience they are allowed to have during the learning process

- learn/extract information from a given sample

- build prediction model according to the training sample

The central challenge for a machine learning algorithm is that it must perform well on new and previously unseen inputs, i.e. 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.

add that mostly overfitting is a real problem + Quelle

A common approach to reduce the possibility of overfitting is regularization. In principal, regularization is any modification of a learning algorithm, that aims to reduce its generalization error but not its training error. In practice this is often achieved by adding a penalty to the cost function.

add: difference of ML and DL

A commonly used deep learning model are deep feedforward networks that 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 $\beta$ that result in the best function approximation.

The term 'network' suggests that the mapping $f$ is a composition of many different functions, i.e. $f(x) = f^{(K)} \circ \cdots \circ f^{(1)}(x)$. The functions $f^{(i)}$ are called layers and the number of composed functions $K$ gives the depth of the model.

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 we do not specify what output they generate. We are only interested in the prediction, the output of the final layer. The hidden layers are vector valued functions whose dimensions define

the width of the model.

The networks 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.

Deep feedforward networks can extend linear models to overcome its limitations. The idea is to apply the linear model not directly to the data $x$ but to a transformed input $\phi(x)$, where $\phi$ is a nonlinear transformation. The remaining question is, how to choose this mapping $\phi$. The deep learning approach to this problem is to define the mapping $y = \phi(x, \beta)^T w$ and learn the parameter $\beta$. This is an example for a deep feedforward network with one hidden layer, namely $\phi$.

When using deep feedforward networks overfitting can occur. A common approach to counteract this effect is a regularization method called dropout. For this method, a dropout rate $p$ must be chosen, that specifies, that a unit is included in the model with probability $1 - p$. In each step of the learning algorithm it is reevaluated which units are part of the model. As a consequence, each of the hidden units must be able to perform well regardless of the other units included in the model. Further details about the dropout regularization can be found in Goodfellow, Bengio, and Courville (2016, chapter 7.12).

soll rein:

- model is associated with a directed acyclic graph
- what are units
- most loss functions non convex for neural networks -> use iterative, gradient based optimizers. Important: initialize all weights to small random values
- important aspect of designing deep feedforward model: choice of cost function
- most neural networks are trained using maximum likelihood -> cost function = negative log likelihood (gradient of cost function must be large and predictable enough to serve as a good guide for the learning algorithm -> use sigmoid output unit(for binomial), use softmax for multinoulli); most popular cost function: cross entropy cost function
- when using feedforward networks: need to determine how many units we need and how they are connected

# 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 $X_i = (X_{i1}, \ldots, X_{ip}) \in \delta \subset \mathbb{R}^p$. The set $\delta$ describes the bounded space consisting of all values that the explanatory variables can take. 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 $h(T_i)$ are the identity function $h(T_i) = T_i$ or the logarithmic transformation $h(T_i) = \ln(T_i)$.

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 weight vector.
3. Use cross-validation to select the penalization parameter $\eta$ 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 $f^{(k)}_{\beta_k}(X)$ 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 which will be estimated next.

In the second step, the estimation of the weight vector, 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 $h(T)$. We also use a penalization parameter $\eta$, that tries to keep the weights 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 vector of weights $\beta$ by minimizing the empirical penalized loss function $\hat{L}(\beta)$ 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 $\beta$, given by $\|\beta\|^2 = \sum_{i=1}^{q} |\beta_i|^2$.

The third step ensures, that we make a suitable choice for the penalization parameter $\eta$. 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 $\eta$ 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 $K_l$, $l \in \{1, \ldots, D\}$ and penalization parameter $\eta_m$, $m \in \{1, \ldots, M\}$, we set $\hat{\beta}^{(l)}_{\eta_m}$ as the vector of weights estimated in equation (4.1) using the penalization parameter $\eta_m$ and the data $\mathcal{F} \setminus K_l$. Let $a_{i,l}$ be an indicator whether observation $i$ is included in the dataset $K_l$ ($a_{i,l} = 1$) or not ($a_{i,l} = 0$). The cross-validation error corresponding to the penalization parameter $\eta_m$ 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 $\eta_{m^*}$ to estimate the vector of weights $\hat{\beta}^{(l)}_{\eta_{m^*}}$ and then build the final prediction model $f_{\hat{\beta}^{(l)}_{\eta_{m^*}}}(X)$.

The estimated vector of weights $\beta$ 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 $E[(I(T \geq t) - \beta(X))^2]$.

# 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, which means 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 and $\Delta_i$ indicates, whether subject $i$ experienced the event of interest or was censored.

We define the conditional survivor functions for $T$ and $C$ as $S_0(u|X) = P(T > u|X)$ and $G_0(u|X) = P(T > u|X)$ respectively. In contrast to section 2.2 the survivor 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 calculations we also assume that $G_0(\tilde{T}|X) \geq \varepsilon > 0$ for some $\varepsilon > 0$, to ensure positivity of the conditional censoring function.

The main difficulty in extending algorithm 4.1 to possibly censored data is to define a suitable loss function, that can be calculated 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*.

evtl zugehörige Verteilungsfunktion hier mit rein nehmen

## 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 responses with surrogate values. This can be achieved by using an appropriate mapping $Y^*(\cdot)$. The transformed values are then inserted in standard smoothing algorithms.

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

general case with Y response variable, could for example be event time

**Definition 5.1.** Let $Y$ be a scalar function of the full data $\mathcal{F}$ and $Y^*(\mathcal{O})$ a scalar function of the observed data $\mathcal{O}$.

Then we call $Y^*(\mathcal{O})$ a *censoring unbiased transformation (CUT)* for $Y$ if for every point $x$ in $\delta$, i.e. $x \in \delta \subset \mathbb{R}^p$:

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

In other words, a censoring unbiased transformation maps the response $Y$ 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

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

where

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

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.

show that BJ Trafo is a CUT

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

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

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

check benefits of doubly robust trafo in contrast to Buckley-James OR downsides of BJ

In contrast to the Buckley-James transformation, which depends on the survivor 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, the censoring and the survival distributions, are taken into account. The IPCW mapping is defined as follows:

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

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 the IPCW transformation is a CUT.

add proof/reference that IPCW trafo is CUT for G=G0

The Buckley-James estimator, as well as the IPCW mapping, depend on the nuisance 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, that requires a well approximation of either $S(\cdot|X)$ or $G(\cdot|X)$, but not necessarily both, is the so called doubly robust mapping. It is defined as follows:

$$Y^*_{DR}(\mathcal{O}; G, S) = \frac{\Delta Y}{G(Y|X)} + \frac{(1-\Delta)m(Y, X; S)}{G(Y|X)} - \int_0^Y \frac{m(c, X; S)}{G(c|X)} d\Lambda_G(c|X), \quad (5.3)$$

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 (5.3) is equal to the IPCW transformation. Similar to the Buckley-James transformation, a term that considers the censored data as well 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).

The double robustness property is shown in theorem 3.1 in Steingrimsson, Diao, and Strawderman (2019) and is reproduced below.

**Theorem 5.1.** *Let $S(\cdot|\cdot)$ and $G(\cdot|\cdot)$ be functions on $(t, x) \in \mathbb{R} \times \delta$ satisfying the regularity conditions given in appendix S.5 in Steingrimsson, Diao, and Strawderman (2019). Then, the transformations $Y^*_{DR}(\mathcal{O}; G, S_0)$, $Y^*_{DR}(\mathcal{O}; G_0, S)$ and $Y^*_{DR}(\mathcal{O}; G_0, S_0)$ are each CUTs for $Y$.*

wieder zu anderem Theorem ändern und Gleichheit zeigen

## 5.2 Censoring unbiased loss functions

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 calculated 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 two censoring unbiased estimators for the estimated loss $\mathcal{R}(\beta) = E[L(h(T), \beta(X))]$ that both reduce to the full data loss in the absence of censoring: the doubly robust loss function and the Buckley-James estimator.

add from basearticle: why IPCW is no longer considered/no suitable choice for estimator of full data loss

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

$$L_{BJ}(\mathcal{O}, \beta; S_0) = \frac{1}{n} \sum_{i=1}^{n} \left[ \Delta_i L(h(T_i), \beta(X_i)) + (1 - \Delta_i) m_L(C_i, X_i, \beta; S_0) \right]. \tag{5.4}$$

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 (5.6).

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

The doubly robust loss function is defined as

$$\begin{aligned} L_{DR}(\mathcal{O}, \beta; G_0, S_0) =& \frac{1}{n} \sum_{i=1}^{n} \frac{\Delta_i L(h(T_i), \beta(X_i))}{G_0(\tilde{T}_i|X_i)} \\ &+ \frac{1}{n} \sum_{i=1}^{n} \left( \frac{(1 - \Delta_i) m_L(\tilde{T}_i, X_i, \beta; S_0)}{G_0(\tilde{T}_i|X_i)} \right. \\ &\left. - \int_0^{\tilde{T}_i} \frac{m_L(u, X_i, \beta; S_0)}{G_0(u|X_i)} d\Lambda_{G_0}(u|X_i) \right), \end{aligned} \tag{5.5}$$

where for a survival curve $S$

$$\begin{aligned} m_l(u, x, \beta; S) &= E_S[L(h(T), \beta(X))|T \geq u, X = x] \\ &= -\int_u^{\infty} \frac{L(h(t), \beta(x))}{S(u|w)} dS(t|w) \end{aligned} \tag{5.6}$$

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 calculating the average of the full data loss that is now transformed via equation (5.3).

Theorem 5.1 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, to obtain a suitable estimator.

As a next step we can plug in estimators $\hat{S}$ and $\hat{G}$ for the conditional survival functions $S_0$ and $G_0$ respectively 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

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

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

evtl kurz zeigen?

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 calculated 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 (5.5) results the doubly robust Brier loss, which is given by

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

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 follows the ideas presented in Steingrimsson and Morrison (2020) and Strawderman (2000).

An other popular goal is to estimate the mean survival $E[T|X]$. For this target parameter we use the $L_2$ loss as our full data loss.

add reason why we need tau from Strawderman 2000

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

Analogously to the estimation of survival probabilities, we need to incorporate the constant $\tau$ 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 and applying the CUDL algorithms to the modified data set gives us an estimator for the restricted mean survival.

## 5.5 Implementation Based on the Doubly Robust Transformation

This section relies on Steingrimsson and Morrison (2020).

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

add paragraph that choice of h gives estimators for both, L2 and Brier loss

We use the following response transformation:

$$D(O_i; G, S) = A_{1i}(G) + B_{1i}(G, S) - C_{1i}(G, S),$$

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

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

For $k = 1, 2$

$$m_k(t, x; S) = E_S[h^k(T)|T \geq t, X = x] = -[S(t|x)]^{-1} \int_t^{\infty} [h(u)^k] dS(u|x) \quad (5.7)$$

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

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 CUDL algorithms inherits the desired doubly robustness property to the transformed data algorithm.

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

**Theorem 5.2.** *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:

$$L_{DR}^{(2)}(\mathcal{O}, \beta; G, S) = \frac{1}{n} \sum_{i=1}^{n} ((A_{2i} + B_{2i} - C_{2i}) - 2(A_{1i} + B_{1i} - C_{1i})\beta(X_i) + (A_{0i} + B_{0i} - C_{0i})\beta(X_i)^2) . \tag{5.8}$$

This equality can be shown by reordering the terms in the sum above in alphabetical order. Lemma A.1 provided in the appendix states that $A_{0i} + B_{0i} - C_{0i} = 1$ holds true. This 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 calculating the square and obtain

$$L_2^*(\mathcal{O}, \beta; G, S) = \frac{1}{n} \sum_{i=1}^{n} \left((A_{1i} + B_{1i} - C_{1i})^2 - 2(A_{1i} + B_{1i} - C_{1i})\beta(X_i) + \beta(X_i)^2\right) . \tag{5.9}$$

We can see that the loss functions in equation (5.8) and equation (5.9) are actually very similar. Only the first term in the sum, which is independent of $\beta$, is different for each loss function.

As a consequence for a fixed penalization parameter $\eta$ 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 weight vector $\beta$.

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 XXX. Recall that $\delta_{i,l}$ 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 $\beta$, it holds that

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

This shows that

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

which completes the proof. □

The assumptions on the conditional censoring function $G(\cdot|\cdot)$ are flexible enough to allow for $G(t|x) = 1$ for all pairs $(t, x)$. And as it holds that $L_{BJ}(\mathcal{O}, \beta; \hat{S}) = L_{DR}(\mathcal{O}, \beta; \hat{S}, \hat{G} = 1)$, theorem 5.2 also applies to the CUDL using the Buckley-James loss.

Theorem 5.2 allows us to implement the CUDL algorithm using software 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 calculate 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 $\{(D(O_i; \hat{G}, \hat{S}), X_i); i = 1, \ldots, n\}$.

# 6 Simulation Study

In this section we compare the performance of XXX

add names of algorithms for comparison

.

Our goal is to estimate the conditional survival probability $P(T > t|X)$ for a prespecified time point $t$. To achieve this, we first transform the response to account for that goal and set $Y = h(T) = I(T > t)$.

## 6.1 Simulation Settings

The different settings for our simulation 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 $M$ with elements $m_{ij} = 0.9^{|i-j|}$. The survival times are simulated from an exponential distribution with mean $\mu = \exp\left(0.1 \sum_{i=9}^{18} X_i\right)$. The censoring distribution is also simulated from an exponential distribution with mean XXX.

**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 20 iid uniform random variables on the interval $[0, 1]$. The survival time follows an exponential distribution with mean $\mu = \sin(X_1\pi) + 2|X_2 - 0.5| + X_3^3$. The censoring times are uniformly distributed on the interval $[0, 6]$. This results in approximately XXX% 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 $M$ with $m_{ij} = 0.75^{|i-j|}$. 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 $\alpha = \frac{2}{\mu}$. The censoring times are uniformly distributed on the interval $[0, 15]$. This leads to approximately XXX% 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 $M$ with $m_{ij} = 0.75^{|i-j|}$. The survival distribution is now a log-normal distribution with mean

$\mu = 0.1|\sum_{i=1}^{5} X_i| + 0.1|\sum_{i=16}^{20} X_i|$. 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 XXX% of our data are censored.

add sentence that these results are generated in functions

## 6.2 Implementation of the CUDL Algorithm

### 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 is completely independent of both, the event time $T$ and the explanatory variables $X$, the conditional survival function $\tilde{G}(\cdot|X) = \tilde{G}(\cdot)$ can be efficiently estimated with the Kaplan-Meier (KM) estimator. This KM estimator is given by

$$\hat{G}(t) = \prod_{T_{(i)} \leq t} \frac{n_i - d_i}{n_i},$$

where $T_{(1)} < \cdots < T_{(n)}$ are the observed censoring times, that are increasingly ordered, $n_i$ denotes the number of subjects $i$ which are at risk at time $T_{(i)}$ and $d_i$ describes the number of subjects $i$ which are censored at time $T_{(i)}$. For example if we consider that censoring is caused by the end of the study, the above stated independence assumption is met and the conditional censoring time can be calculated 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, that is provided in the package rpart. The tree method classifies the observations according to the explanatory variables in different subgroups. Within each subgroup a KM estimator is calculated 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 Kaplan-Meier estimator, as we presume that the observed time $T$ 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 the function rfscr from the package randomForestSRC.

It is possible to choose different estimators 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 $m_k$

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

For the estimation of the conditional expectation $m_k$ given in equation (5.7), 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, that 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

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

add procedure described in code

### 6.2.3 Selecting Truncation Time to Ensure Positivity

Steingrimsson, Diao, Molinaro, and Strawderman (2016).

### 6.2.4 Transformation of Survival Data

We now describe how the transformation of the responses $T$ described in section XXX can be implemented in R.

We use the Buckley-James as well as the douby robust transformation to compare their performance.

For the Buckley-James transformation we estimate the conditional expectation $m$ using the random forest method with the Brier loss as full data loss. The

distribution of survival times is then calculated with a survival tree, as described in section XXX. In this case, the transformed response is given by

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

which equals the Buckley-James transformation given in equation (5.1) with response $Y = h(T) = I(T > t)$ and the truncated observations $\mathcal{O}(t)$.

For the doubly robust transformation given in equation (5.3) we have to calculate $A_{1i}$, $B_{1i}$ and $C_{1i}$. Again, we can plug in $h(T) = I(T > t)$, the estimated expectation $m$ and the estimated censoring function $\hat{G}$ and obtain

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

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

and

### 6.2.5 Keras Model for Transformed Data

In this section we describe how the deep learning model from algorithm 5.1 can be implemented in R.

To predict the survival probabilities 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 this layer is our input layer, we use the dropout rate $1 - p = 0.2$.

The second layer, which is our output layer, consists of one unit and uses the sigmoid activation function, which is given by $\phi(x) = (1 + e^{-t})^{-1}$. 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 the gradient descent algorithm $(number of observations)/32 * 100$ – times to find the optimal parameter. After fitting the keras model with the training data, we evaluate its performance with the test data and the resulting IPC weighted mean squared error.

As a second deep learning model we perform 5-fold cross validation to choose a suitable penalization term $\eta$ from the sequence $(0, 0.001, 0.01, 0.1)$. The performance of the resulting penalized feedforward network is also evaluated with the IPC weighted mean squared error.

The procedure described above is performed twice. First for the response that is transformed using the Buckley-James transformation and then for the doubly robust transformed response.

### 6.2.6 Algorithms Used for Comparison

think of better title

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 $t$.

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. 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 evaluated using a IPC weighted mean squared error.

## 6.3 Simulation Results

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

First the data is generated according to the current setting. Then, the IPC weighted mean squared error of each of the models is calculated and used as our performance measure. We repeat this procedure 50 times and compare the resulting errors.

# 7 Real Data

change title

describe data set

evaluate algorithms with data set

compare resulting error

# 8 Conclusion

evtl als Ausblick: Erweiterung auf competing risks, time dependent explanatory variables, recurrent events; Test different estimators for $S$ and $G$

# References


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

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.

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 (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.

# Appendices

## A Proofs

### A.1 Auxiliary Lemma for Theorem 5.2

**Lemma A.1.** *Using the notation introduced in section XXX, 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

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

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:

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

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 $\Delta^*$ 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 $M^*(s)$.

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

Now we can plug in our choice of $\Delta_i^*$ and $M^*(s)$ to equation (A.2).

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

Combining equation (A.1) and (A.3) now yields:

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

□