Variable selection and shrinkage in the Cox proportional hazards model

by

Ruth Leonie Koole

in partial fulfillment of the requirements for the degree of

Bachelor of Science
in Applied Mathematics

at the Delft University of Technology,
to be defended publicly on 20-06-2017

Thesis committee: Prof. dr. ir. G. Jongbloed (supervisor)
Dr. D. C. Gijswijt
Dr. ir. M. Keijzer

This thesis is confidential and cannot be made public until 20-06-2017

TU Delft Delft University of Technology

Contents

1

Introduction

In the medical world, specialists and patients are often interested in the time it takes to a certain event to happen. For example, when a certain fatal disease is diagnosed, the specialists want to know how long this patient will live. Or maybe another specialist wants to know how long it takes to cure a patient when he uses a specific treatment for his patient.

The time until the certain events happens may be strongly dependent of characteristics of the patient. One can imagine that for a specific disease a child will probably be cured faster than an elderly woman with the same type of disease. This can be influenced by many different variables and specialists would want to know what variables have the biggest influence on the survival time of a patient.

Survival analysis is a branch of statistics for analyzing the time until a certain event happens, which is used in plenty of applied fields, such as medicine, economics, biology and engineering. In this thesis, there will be focused on an application in medicine.
In survival analysis, the term survival time is commonly used for the time a certain event happens, where in medical applications the event is often seen as death. One important function in survival analysis is the survival function, which is defined as the probability that an object (or in the medical approach this is a patient) will survive until time t. In medical research survival functions are often plotted to give an idea about the probability of surviving beyond a specific time. An example of a survival plot can be found below in figure 1.1. In this figure there can be seen that as time is increasing the probability of surviving beyond that time is decreasing.
Another important function is the hazard rate function, which is proportional to the the probability that an event will happen in a small interval after time t, given that the event has not yet happened at time t. This is often referred to as the rate of mortality or the rate of an event.

One well known model in survival analysis is the Cox proportional hazards model, which was invented by Sir David Cox in 1972. Nowadays, this is one of the most important models in survival analysis [1]. In such a model the time to a certain event is related to many explanatory variabels which may or may not have an influence on the time to the event.

To analyse what explanatory variables are most related to the distribution of on the time to an event, Robert Tibshirani proposed a method for variable selection and shrinkage in the Cox proportional hazards model [2]. This method, which is called the Least Absolute Shrinkage and Selection Operator (LASSO) results in a subset of explanatory variables from which the parameters are shrunken towards zero. The parameters related to the other explanatory variables are all equal to zero.

Patients with breast cancer at the same stage of disease can have different treatment responses and different outcome. Besides analyzing standard clinical risk factors such as diagnosis age and the diameter of the tumor, L. van 't Veer proposed to look at DNA measurements of the patients [3]. In the period between 1984 and 1995, 144 patients with breast cancer were monitored at the Netherlands Cancer Institute [4]. Those patients were selected among all patients that were diagnosed with breast

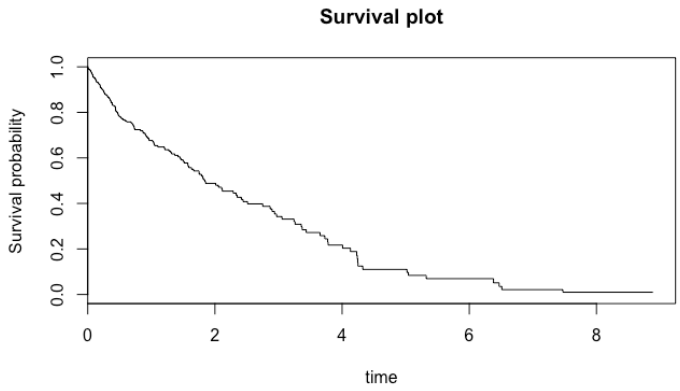

Figure 1.1

cancer according to several criteria: the diameter of the tumor at the moment of diagnose was less than 5 centimeters, the age of the patients at moment of diagnose was less than 53 years old and there was a lymph-node-positive disease diagnosed (which means that the cancer cells have spread throughout the whole body which results in a higher risk). For all monitored patients there were 5 clinical risk variables and 70 genetical measurements monitored. In this thesis, the Cox Proportional Hazards model and the method of LASSO are used to look at the relation of the gene expression measurements and the standard clinical and histological variables on the distribution of survival time of the patients.

The main aim of this thesis is to use Tibshirani's method of LASSO on the Cox proportional hazards model. First there will be focus on the stochastic model of the proportional hazards model. Then in the second part we will focus on algorithms for solving the optimization problems that need to be solved in order to fit the Cox proportional hazards model to data. Afterwards this model is applied to the data of breast cancer patients of the Netherlands Cancer Institute in order to see what variables are most related to te distribution of survival time of the patients.

2

Stochastic model for survival data

The main aim of survival analysis is to analyze the distribution of the time until a certain event happens. For the application in medicine, this event is often seen as death, but it can also be defined differently, such as the time until the patient is cured. The time until the event if often referred to as survival time. In this thesis, there will be spoken about time to death and therefore the event is death. In this section, first some basic concepts of survival analysis will be introduced that will be used for the introduction of the Cox proportional hazards model later this section.

2.1. The basic concepts of survival analysis

In survival analysis a few basic functions are used to describe certain characteristics of survival data. Those functions are introduced in this section.

Let T be the random variable which denotes the time to death of a specific patient. Often the aim is to make a good prediction about the time until death, based on the characteristics of the patient. This can be done using the survival function, which is one of the most used functions in survival analysis. Define $S(t)$ as the survival function, which is the probability that the patient will survive at least until time t. Then $S(t)$ is given by:

$$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t) \tag{2.1}$$

where $F(t) = P(T \leq t)$ is the cumulative distribution function of T, which is the probability that the event time is smaller than or equal to time t. Let $f(t) = \frac{d}{dt}F(t)$ be the probability density function of t when T is continuous.

Another main function in survival analysis is the hazard rate function. The hazard rate function evaluated at time t is, for $\epsilon > 0$ sufficiently small, proportional to the probability that the patient will die within the small interval $[t, t+\epsilon]$ given that the patient survives beyond time t. In a mathematical way, the hazard rate function $\lambda(t)$ is defined as:

$$\lambda(t) = \lim_{\epsilon \downarrow 0} \frac{P(T \in [t, t+\epsilon] \mid T \geq t)}{\epsilon} \tag{2.2}$$

The hazard rate function is often referred to as the rate of mortality or the rate of an event. The hazard rate function can also be expressed in terms of the survival function, which can be proved using the definition of the survival function to rewrite the definition of the hazard rate function:

$$\lambda(t) = \lim_{\epsilon \downarrow 0} \frac{P(T \in [t, t+\epsilon] \mid T \geq t)}{\epsilon} = \lim_{\epsilon \downarrow 0} \frac{1}{\epsilon} \frac{P(T \in [t, t+\epsilon])}{P(T \geq t)} = \frac{f(t)}{S(t)} \tag{2.3}$$

Also, using that $S(t) = 1 - F(t)$, another expression for the hazard rate function is:

$$\lambda(t) = \frac{f(t)}{S(t)} = \frac{\frac{-dS(t)}{dt}}{S(t)} = -\frac{d}{dt} \ln(S(t))$$

When integrating the hazard rate until time t, one obtains the cumulative hazard rate, which is represented as $\Lambda(t)$. $\Lambda(t)$ is an increasing function with $\Lambda(0) = 0$. Rewriting the previous relation gives an expression of the survival function in terms of the cumulative hazard function $\Lambda(t)$:

$$\ln(S(t)) = -\int_0^t \lambda(s)ds + C = -\Lambda(t)$$

where C is zero since $S(0) = 1$ and $\Lambda(0) = 0$.

2.2. The Cox proportional hazards model

In the previous section, the basic tools for survival analysis were introduced for a general continuous random variable. One can imagine that this hazard rate function or survival function can be strongly influenced by certain characteristics of the patient. Characteristics of a patient can for example be age, sex, blood pressure or the type of treatment a patient was given.

The Cox proportional hazards model was introduced by Sir David Cox in 1972. Nowadays, it is one of the most used models in survival analysis and it is also known as the Cox model [1]. This model introduces a definition for the hazard rate function which is dependent on the explanatory variables.

Suppose there are n patients observed over a period of time, and suppose p explanatory variables are known. Let X be a $n \times p$-matrix containing the variables for all objects. Then the i^{th} row of this matrix, which contains the p explanatory variables of the i^{th} patient, is denoted by X_i and the j^{th} column, which contains information about a certain variable for all n patients, is notated by X^j. Suppose the measured event time of the event, depending on X_i is t_i. Then according to the Cox proportional hazards model the conditional hazard rate function of the i^{th} object is defined as:

$$\lambda(t|X_i) = \lambda_0(t)e^{\beta^T \mathrm{X_i}} \tag{2.4}$$

where $\lambda_0(t)$ is called the baseline hazard function and $\beta = (\beta_1, \dots, \beta_p)$ are the regression parameters of the model. The baseline hazard rate function describes the risk at time t for a patient with $X_i = 0$, where 0 is a p-dimensional vector. This patient with explanatory variables equal to zero can be seen as a reference cell or pivot. The factor $e^{\beta^T X_i}$ is the relative risk, which increases or reduces the risk in proportion to the explanatory variables X_i. Not that the increase or reduce in risk is independent of the time t.

Given that the hazard rate function is dependent on the explanatory variables, the cumulative hazard function will also be dependent on the explanatory variables. Therefore, in the Cox proportional hazards model, one can define the cumulative hazard function as:

$$\Lambda(t|X_i) = \int_0^t \lambda(s|X_i)\ ds$$

Therefore, the survival function is also dependent on the explanatory variables and can be defined as:

$$S(t|X_i) = e^{-\Lambda(t|X_i)} = e^{-\int_0^t \lambda(s|X_i)\ ds} \tag{2.5}$$

In the Cox model it is assumed that the hazard rate of a patient is dependent on the explanatory variables of the patient in a specific mathematical way. Knowing the explanatory variables, which are stored in the matrix X, one can estimate the regression parameters $\beta = (\beta_1, \dots, \beta_p)$ by maximum likelihood estimation. Once the regression parameters are known, for example the expected lifetime of a patient can be estimated. The maximum likelihood estimation can be done by maximizing two different types of likelihoods, the likelihood function and the partial likelihood function. Both functions will be introduced in the following sections and the differences will also be discussed

2.2.1. Likelihood function of the Cox model

Suppose there are n patients observed over a period of time, which are denoted by $1, \dots, n$. Let $t_1, \dots, t_n$ be the measured event times of the patients and let X again be the matrix of all p explanatory variables of all patients, where X_i denotes the explanatory variables for the i^{th} patient.

The likelihood can be derived by computing the joint density function for the event times. Then using independence, the hazard rate function in terms of the survival function (as was introduced in equation 2.3) and the survival function as defined in equation 2.5, the likelihood function is given by:

$$\begin{aligned} L(\beta) &= f(t_1, \dots, t_n|\beta) = \prod_{i=1}^{n} f(t_i|\beta) = \prod_{i=1}^{n} \lambda(t_i|\beta) \cdot S(t_i|\beta) \\ &= \prod_{i=1}^{n} \lambda_0(t_i) e^{\beta^T X_i} \cdot e^{-\Lambda(t_i|\beta)} = \prod_{i=1}^{n} \lambda_0(t_i) e^{\beta^T X_i} \cdot \exp\left(-\int_0^{t_i} \lambda_0(s) e^{\beta^T X_i} ds\right) \\ &= \prod_{i=1}^{n} \lambda_0(t_i) e^{\beta^T X_i} \cdot \exp\left(-\int_0^{t_i} \lambda_0(s) ds \; e^{\beta^T X_i}\right) \end{aligned}$$

Rewriting this a little gives that the likelihood function equals:

$$L(\beta) = \prod_{i=1}^{n} \frac{\lambda_0(t_i) e^{\beta^T X_i}}{\exp\left(\int_0^{t_i} \lambda_0(s) ds \; e^{\beta^T X_i}\right)} \tag{2.6}$$

Then according to the method of maximum likelihood estimation, one should maximize $L(\beta)$ in order to estimate the regression parameters $\beta = (\beta_1, \dots, \beta_p)$.

Define $\ell(\beta) = -\ln(L(\beta))$ as the minus-log-likelihood then instead of maximizing $L(\beta)$, it is equivalent to minimize the minus-log-likelihood which equals:

$$\begin{aligned} \ell(\beta) &= -\ln(L(\beta)) = -\ln\left(\prod_{i=1}^{n} \frac{\lambda_0(t_i) e^{\beta^T X_i}}{\exp\left(\int_0^{t_i} \lambda_0(s) ds \; e^{\beta^T X_i}\right)}\right) \\ &= -\sum_{i=1}^{n} \ln\left(\frac{\lambda_0(t_i) e^{\beta^T X_i}}{\exp\left(\int_0^{t_i} \lambda_0(s) ds \; e^{\beta^T X_i}\right)}\right) \\ &= -\sum_{i=1}^{n} \left(\ln\left(\lambda_0(t_i) e^{\beta^T X_i}\right) - \ln\left(\exp\left(\int_0^{t_i} \lambda_0(s) ds \; e^{\beta^T X_i}\right)\right)\right) \\ &= -\sum_{i=1}^{n} \left(\ln(\lambda_0(t_i)) + \beta^T X_i - \int_0^{t_i} \lambda_0(s) ds \; e^{\beta^T X_i}\right) \end{aligned} \tag{2.7}$$

The estimation of the regression parameters according to the minus-log-likelihood function can be found in section 5.1.

Note that the likelihood function (and the minus-log-likelihood function) are dependent of the baseline hazard function. Since this baseline hazard function should be integrated in order to compute the minus-log-likelihood of the model, this may cause problems in case the baseline hazard function is unbounded. That will result in an unbounded minus-log-likelihood, which causes difficulties in estimating the regression parameters.

Because of this disadvantage of the likelihood function, Sir David Cox proposed to look a likelihood type of function, which is called the partial likelihood function. This partial likelihood function leaves the baseline hazard function completely unspecified.

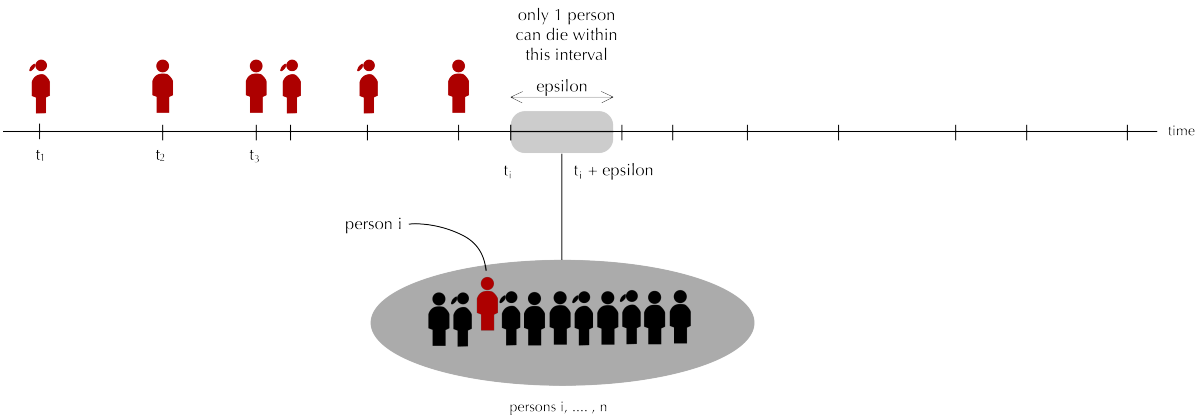

Figure 2.1: Visualization of the situation

2.2.2. Partial likelihood function of the Cox model

Again, suppose that there are n patients observed over a period of time. Let $t_1, \dots, t_n$ be the measured event times, but now also assume that they are untied and ordered such that $t_1 < \cdots < t_n$. Let $1, \dots, n$ denote the patients corresponding to event time t_i and the patient's explanatory variables X_i.

Since the event times are ordered, one can derive a likelihood type of function by looking at the data at an alternative view. Because of the ordering, it is known that at time t_i patients $\{1, \dots, i-1\}$ have already experienced the event and that persons $\{i, \dots, n\}$ are at risk to experience the event. Knowing that there is exactly one patient that will experience the event at time t_i, what is the probability that this will be exactly patient i, with explanatory variables X_i? The situation is sketched in figure 2.1.

Let $P_i(\epsilon)$ be the probability that patient i with explanatory variables X_i experiences the event at time t_i, with $\epsilon > 0$ sufficiently small. This probability $P_i(\epsilon)$ can then be expressed as the following conditional probability [5]:

$$P_i(\epsilon) = P\left(T_i \in [t_i, t_i + \epsilon] \mid \{T_k \in [t_i, t_i + \epsilon] \text{ for exactly one } k = i, \dots, n\} \cap \{T_k \geq t_i \forall\ k = i, \dots, n\}\right) \quad (2.8)$$

Let $A = \{T_i \in [t_i, t_i + \epsilon]\}$, $B = \{T_k \in [t_i, t_i + \epsilon]$ for exactly one $k = i, \dots, n\}$ and $C = \{T_k \geq t_i\ \forall\ k = i, \dots, n\}$. Note that the conditional probability in equation 2.8 is of the form $P(A|B \cap C)$, which can easily be rewritten as:

$$P(A|B \cap C) = \frac{P(A \cap B \cap C)}{P(B \cap C)} = \frac{P(A \cap B|C)P(C)}{P(B|C)P(C)} = \frac{P(A \cap B|C)}{P(B|C)}$$

Using this, equation 2.8 can be rewritten as:

$$P_i(\epsilon) = \frac{P(\{T_i \in [t_i, t_i + \epsilon]\} \cap \{T_k \in [t_i, t_i + \epsilon] \text{ for exactly one } k = i, \dots, n\} \mid T_k \geq t_i\ \forall\ k = i, \dots, n)}{P(T_k \in [t_i, t_i + \epsilon] \text{ for exactly one } k = i, \dots, n \mid T_k \geq t_i\ \forall\ k = i, \dots, n)} \quad (2.9)$$

When taking a closer look at the numerator of equation 2.9, note that since there can only be one event time in the interval $[t_i, t_i + \epsilon]$ and T_i has to be in that same interval, all other event times can not be within the interval $[t_i, t_i + \epsilon]$. Given that all remaining event times are bigger than t_i, all event times T_k with $k = i + 1, \dots, n$ must be bigger than $t_i + \epsilon$.

Then using the definition of a conditional probability and the previous finding, the numerator of equation 2.9 can be rewritten as:

$$
\begin{aligned}
&P(\{T_i \in [t_i, t_i+\epsilon]\} \cap \{T_k > t_i + \epsilon \ \forall\ k = i+1, \dots, n\} \mid T_k \geq t_i \ \forall\ k = i, \dots, n) \\
=&\frac{P(\{T_i \in [t_i, t_i+\epsilon]\} \cap \{T_k > t_i + \epsilon \ \forall\ k = i+1, \dots, n\} \cap \{T_k \geq t_i \forall\ k = i, \dots, n\})}{P(T_k \geq t_i \ \forall\ k = i, \dots, n)} \\
=&\frac{P(\{T_i \in [t_i, t_i+\epsilon]\} \cap \{T_i \geq t_i\} \cap \{\cap_{k=i+1}^{n} \{\{T_k > t_i + \epsilon\} \cap \{T_k \geq t_i\}\}\})}{P(T_k \geq t_i \forall\ k = i, \dots, n)}
\end{aligned}
$$

Using independence of the event times this becomes:

$$
\begin{aligned}
&\frac{P(\{T_i \in [t_i, t_i+\epsilon]\} \cap \{T_i \geq t_i\})}{P(T_i \geq t_i)} \cdot \frac{P(\cap_{k=i+1}^{n} \{\{T_k > t_i + \epsilon\} \cap \{T_k \geq t_i\}\})}{P(\cap_{k=i+1}^{n} \{T_k \geq t_i\})} \\
&= P(T_i \in [t_i, t_i+\epsilon] \mid T_i \geq t_i) \cdot \prod_{k=i+1}^{n} P(T_k > t_i + \epsilon \mid T_k \geq t_i)
\end{aligned} \tag{2.10}
$$

Having another representation of the numerator of equation 2.9, it is instructive to have a closer look at the denominator of equation 2.9 too.

Note that at event time t_i, all patients $i, i+1, \dots, n$ are still at risk to experience the event. Suppose that patient $k = i, i+1, \dots, n$ has the event within the interval $[t_i, t_i+\epsilon]$, then since there can be only one event in that interval all remaining event times must be bigger than $t_i + \epsilon$. So let k be a patient with event time bigger than t_i, then the probability that object k is the only object with event time in the interval $[t_i, t_i+\epsilon]$ equals: $P(\{T_k \in [t_i, t_i+\epsilon]\} \cap \{T_j > t_i + \epsilon \ \forall\ j \neq k\} \mid T_j \geq t_i \ \forall\ j = i, \dots, n)$. Knowing that there is one patient that experiences the event within the interval $[t_i, t_i+\epsilon]$, gives that one of the patients $i, i+1, \dots, n$ should experience the event.

Using this and independence of the event times, the denominator of equation 2.9 can be rewritten as:

$$
\begin{aligned}
&P(T_k \in [t_i, t_i+\epsilon] \text{ for exactly one } k = i, \dots, n \mid T_k \geq t_i \ \forall\ k = i, \dots, n) \\
&= \sum_{k=i}^{n} P\left(\{T_k \in [t_i, t_i+\epsilon]\} \cap \{T_j > t_i + \epsilon \ \forall\ j \neq k\} \mid T_j \geq t_i \ \forall\ j = i, \dots, n\right) \\
&= \sum_{k=i}^{n} \frac{P\left(\{T_k \in [t_i, t_i+\epsilon]\} \cap \{T_j > t_i + \epsilon \ \forall\ j \neq k\} \cap \{T_j \geq t_i \ \forall\ j = i, \dots, n\}\right)}{P(T_j \geq t_i \ \forall\ j = i, \dots, n)} \\
&= \sum_{k=i}^{n} \left(\frac{P\left(\{T_k \in [t_i, t_i+\epsilon]\} \cap \{T_k \geq t_i\}\right)}{P\left(T_k \geq t_i\right)} \cdot \frac{P\left(\{T_j > t_i + \epsilon \ \forall\ j \neq k\} \cap \{T_j \geq t_i \ \forall\ j \neq k\}\right)}{P\left(T_j \geq t_i \ \forall\ j \neq k\right)} \right) \\
&= \sum_{k=i}^{n} \left(P\left(T_k \in [t_i, t_i+\epsilon] \mid T_k \geq t_i\right) \cdot \prod_{j \neq k} P\left(T_j > t_i + \epsilon \mid T_j \geq t_i\right) \right)
\end{aligned} \tag{2.11}
$$

Then substituting equations 2.10 and 2.11 into equation 2.9 gives that:

$$
\begin{aligned}
P_i(\epsilon) &= \frac{P(T_i \in [t_i, t_i+\epsilon] \mid T_i \geq t_i) \cdot \prod_{j=i+1}^{n} P(T_j > t_i + \epsilon \mid T_j \geq t_i)}{\sum_{k=i}^{n} \left(P\left(T_k \in [t_i, t_i+\epsilon] \mid T_k \geq t_i\right) \cdot \prod_{j \neq k} P\left(T_j > t_i + \epsilon \mid T_j \geq t_i\right) \right)} \\
&= \frac{P(T_i \in [t_i, t_i+\epsilon] \mid T_i \geq t_i) \cdot P(T_k > t_i + \epsilon \mid T_k \geq t_i)}{\sum_{k=i}^{n} P\left(T_k \in [t_i, t_i+\epsilon] \mid T_k \geq t_i\right) \cdot P(T_i > t_i + \epsilon \mid T_i \geq t_i)} \\
&= \frac{P(T_i \in [t_i, t_i+\epsilon] \mid T_i \geq t_i) \ / \ P(T_i > t_i + \epsilon \mid T_i \geq t_i)}{\sum_{k=i}^{n} P\left(T_k \in [t_i, t_i+\epsilon] \mid T_k \geq t_i\right) \ / \ P(T_k > t_i + \epsilon \mid T_k \geq t_i)} \\
&= \frac{P(T_i \in [t_i, t_i+\epsilon] \mid T_i \geq t_i)}{\sum_{k=i}^{n} P\left(T_k \in [t_i, t_i+\epsilon] \mid T_k \geq t_i\right)}
\end{aligned}
$$

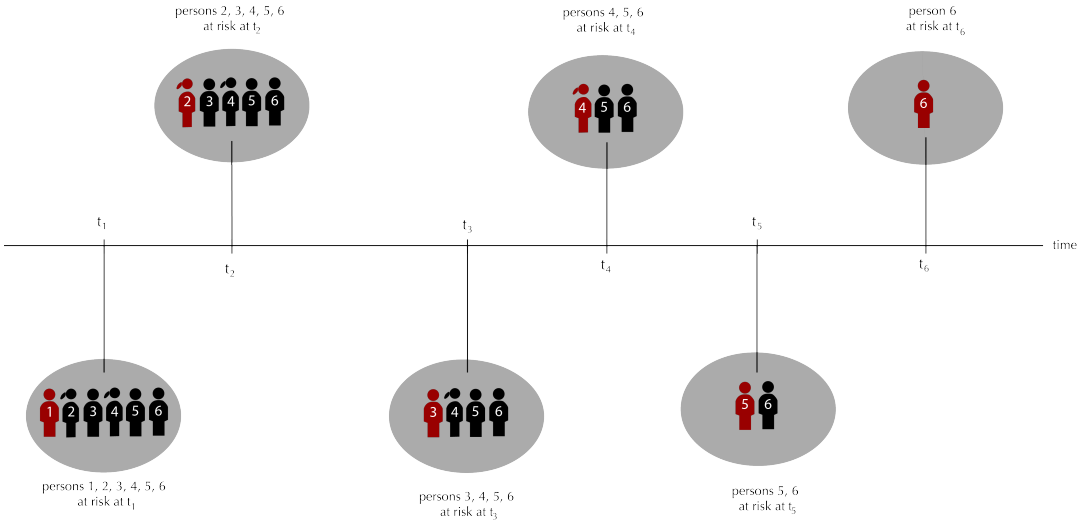

Figure 2.2: Visualization of the situation

Recall from section 2.1 that the hazard rate function was introduced as:

$$\lambda(t) = \lim_{\epsilon \downarrow 0} \frac{P(T \in [t, t+\epsilon] \mid T \geq t)}{\epsilon}$$

Now let $P_i = \lim_{\epsilon \downarrow 0} P_i(\epsilon)$ be the probability $P_i(\epsilon)$, with $\epsilon > 0$ very small. Then using the definition for the hazard rate function, will result in a representation of the probability P_i in terms of the hazard rate function:

$$\begin{aligned} P_i &= \lim_{\epsilon \downarrow 0} P_i(\epsilon) \\ &= \frac{\lim_{\epsilon \downarrow 0} P(T_i \in [t_i, t_i+\epsilon] \mid T_i \geq t_i)}{\lim_{\epsilon \downarrow 0} \sum_{k=i}^{n} P(T_k \in [t_i, t_i+\epsilon] \mid T_k \geq t_i)} \\ &= \frac{\lim_{\epsilon \downarrow 0} \epsilon \cdot \lambda_i(t_i)}{\sum_{k=i}^{n} \lim_{\epsilon \downarrow 0} \epsilon \cdot \lambda_k(t_i)} \\ &= \frac{\lambda_i(t_i)}{\sum_{k=i}^{n} \lambda_k(t_i)} \end{aligned} \tag{2.12}$$

Then according to the Cox model, the hazard rate function equals $\lambda_i(t_i|X_i) = \lambda_0(t_i)e^{\beta^T X_i}$, which gives that the probability P_i in terms of the regression parameters β and the variables X_i equals:

$$P_i = \frac{\lambda_i(t_i|X_i)}{\sum_{k=i}^{n} \lambda_k(t_i|X_k)} = \frac{\lambda_0(t_i)e^{\beta^T X_i}}{\sum_{k=i}^{n} \lambda_0(t_i)e^{\beta^T X_k}} = \frac{e^{\beta^T X_i}}{\sum_{k=i}^{n} e^{\beta^T X_k}} \tag{2.13}$$

So when the event times are ordered and untied, it is possible to express the probability P_i that exactly patient i with explanatory variables X_i experiences the event at time t_i in terms of the hazard rate function.

Now assume that all event times $t_1, \ldots, t_n$ are known. The so called partial likelihood, which was introduced by Sir David Cox, can be found by taking the probability that for every $i = 1, \ldots, n$ patient i experiences the event at event time t_i. The so called partial likelihood describes the probability that at time t_1 patient 1 experiences the event and at time t_2 patient 2 experiences the event and at time t_3 patient 3 experiences the event, and so on until time t_n. The main idea can be found in figure 2.2.

The partial likelihood $L_p(\beta)$ can be found by computing the probability:

$$\begin{aligned} L_p(\beta) &= \lim_{\epsilon \downarrow 0} P(\{\text{only } T_1 \in [t_1, t_i + \epsilon]\} \cap \cdots \cap \{\text{only } T_n \in [t_n, t_n + \epsilon]\} \mid T_1 \geq t_1, \ldots, T_n \geq t_n) \\ &= \lim_{\epsilon \downarrow 0} P(\text{only } T_1 \in [t_1, t_i + \epsilon] \mid T_1, T_2, \ldots, T_n \geq t_1) \cdot \cdots \cdot P(\text{only } T_n \in [t_n, t_n + \epsilon] \mid T_n \geq t_n) \\ &= \lim_{\epsilon \downarrow 0} P_1(\epsilon) \cdot \cdots \cdot P_n(\epsilon) \\ &= P_1 \cdot \cdots \cdot P_n \end{aligned}$$

Then using the expression for the probability P_i from equation 2.13, gives that the partial likelihood equals:

$$L_p(\beta) = \prod_{i=1}^{n} \frac{e^{\beta^T X_i}}{\sum_{k=i}^{n} e^{\beta^T X_k}} \tag{2.14}$$

Define $\ell_p(\beta) = -\ln L_p(\beta)$ as the minus-log-partial-likelihood, then this equals:

$$\begin{aligned} \ell_p(\beta) &= -\ln(L_p(\beta)) \\ &= -\ln\left(\prod_{i=1}^{n} \frac{e^{\beta^T X_i}}{\sum_{k=i}^{n} e^{\beta^T X_k}}\right) \\ &= -\sum_{i=1}^{n} \ln\left(\frac{e^{\beta^T X_i}}{\sum_{k=i}^{n} e^{\beta^T X_k}}\right) \\ &= -\sum_{i=1}^{n} \left(\ln\left(e^{\beta^T X_i}\right) - \ln\left(\sum_{k=i}^{n} e^{\beta^T X_k}\right)\right) \\ &= \sum_{i=1}^{n} \left(\ln\left(\sum_{k=i}^{n} e^{\beta^T X_k}\right) - \beta^T X_i\right) \end{aligned} \tag{2.15}$$

Note that in the computations in equation 2.13 the baseline hazard rate function cancels out. The probability that patient i with explanatory variables X_i has event time t_i is therefore independent on the baseline hazard rate function. This results in a so called partial likelihood that is also independent on the baseline hazard rate function, which is in contrast to the likelihood function of equation 2.6, which is dependent on the baseline hazard rate function.

2.2.3. Differences likelihood and partial likelihood

In the previous sections, the likelihood (equation 2.6) and the partial likelihood (equation 2.14) for the Cox model have been derived. Both functions can be used for the maximum likelihood estimation of the regression parameters, see section 5.1.

The most important difference between those functions is the appearance of the baseline hazard rate function. In the computation of the probability P_i the baseline hazard rate function cancels out, see equation 2.13. The partial likelihood is therefore independent on the baseline hazard rate function, in contrast to the likelihood function which is dependent on the baseline hazard rate function.

One disadvantage of using the baseline hazard function is in the case that the baseline hazard rate function is unbounded. Since the baseline hazard rate function should be integrated in order to compute the likelihood function, this would result in an unbounded likelihood function, which causes difficulties in estimating the regression parameters when using maximum likelihood estimation.

Another disadvantage of the appearance of the baseline hazard rate function is the difficulty to determine the baseline hazard rate function. For determining this function, specific assumptions for

the distribution of baseline hazard rate function should be made.

Due to the absence of the baseline hazard function in the partial likelihood function, the partial likelihood function is preferred over the likelihood function. So when applying maximum likelihood estimation for estimating the regression parameters, the partial likelihood function is used instead of the likelihood function. In this thesis the partial likelihood function is used for further computations, but in section 5.1 both functions are used to estimate the regression parameters.

2.2.4. Censored data

In survival analysis, one common problem which has to be dealt with is the one of censored data. Censoring occurs when event times are known to occur within a certain interval. There are several types of censoring, such as right-censoring, left-censoring and interval censoring.

In medical applications, event times are typically right-censored. Therefore, in this thesis only right-censoring is taken in consideration. In the case of right-censored data, an event time T is observed if $T \leq C$, where $C > 0$ is called the censoring time. So if the event time of an patient is not observed, the only thing known is that $T > C$. Right-censoring occurs in case patients disappear before the event has actually happened. This can have several reasons, such as death due to another cause, or the choice to discontinue their participation in the study or the patient moving away.

Suppose there are n patients with event times $T_1, \ldots T_n$, define $C_1, \ldots C_n$ as the censoring times. Then one way to model right-censored data in a mathematical is by defining:

$$\Delta_i = \begin{cases} 1 & \text{if } T_i \leq C_i \\ 0 & \text{if } T_i > C_i \end{cases}$$

and $Y_i = \min(T_i, C_i)$.

Let the data be $(y_i, \delta_i), i = 1, \ldots, n$, where y_i is the realization of Y_i and δ_i the realization of Δ_i. When censoring is taken in consideration, the only way a patient could have an event time at t_i, is when the event is actually noticed so when $\delta_i = 1$. In the case that $\delta_i = 0$, the only thing that was measured, is that patient i was still alive at time t_i, but the specific event was not noticed.

So at event time t_i the patients $i, i+1, \ldots, n$ are still at risk, whether their event times are censored or not. In case the event time of patient i is censored, it is known that the event does not happen at time t_i, but somewhere in the future. Then the probability that patient i actually experiences the event time within the interval $[t_i, t_i + \epsilon]$ is zero, since it is already known the event takes place further in the future. At the next event time t_{i+1} patient i is not taken in consideration anymore, since the patient disappeared in our study. The main idea is sketched in figure 2.3. Note that if patient i has a censored event time, this patient can not be left out in the whole study, since this patient is still at risk for experiencing the event until time t_i.

Define the index $I = \{i \in [1, n] : \delta_i = 1\}$. Then I only consists of the patients whose events were actually noticed. All patients that are not in I are still taken in consideration, since they are still at risk for experiencing the event until their censoring time. After their censoring time, those patients are not taken in consideration.

Then due to the censoring, the partial likelihood as was introduced in section 2.2.2 is adapted to:

$$L_{pc}(\beta) = \prod_{i \in I} \left(\frac{e^{\beta^T X_i}}{\sum_{k=i}^{n} e^{\beta^T X_k}} \right)$$

And the minus-log-partial likelihood equals:

$$\ell_{pc}(\beta) = \sum_{i \in I} \left(\ln \left(\sum_{k=i}^{n} e^{\beta^T X_k} \right) - \beta^T X_i \right) \tag{2.16}$$

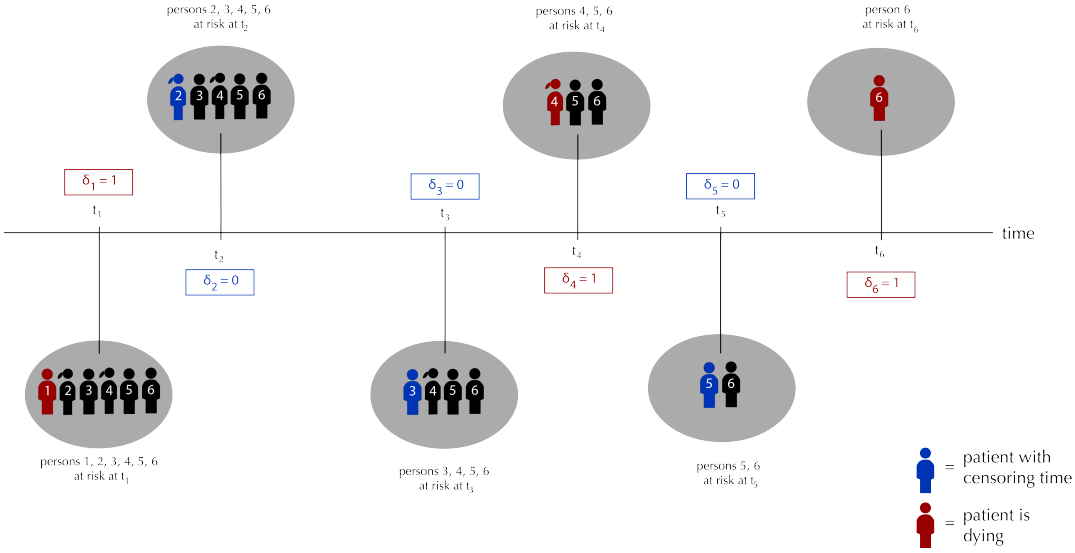

Figure 2.3: Visualization of the situation

3

Variable selection and shrinkage

In the medical application of survival analysis, there are often many potential explanatory variables involved. There may be so many variables involved, that the model might become over parameterized or difficult to interpret. In such a case, there are several regression methods that can be used to limit the number of used explanatory variables.

One method which can be used is variable selection, which is a discrete process where variables are either retained or discarded. When there are a large number of explanatory variables, it is often instructive to determine a smaller subset of the variables which exhibit the strongest effects. One disadvantage of variable selection is that unless the produced model possibly has a smaller prediction error than the full model, the produced model often exhibits large variance [6].

Another type of method is shrinkage, which are more continuous type of methods and hence do not suffer as much from variability. However, shrinkage does not discard certain variables and hence does not give an easily interpretable model. [6]

Robert Tibshirani proposed to use a method called The Least Absolute Shrinkage and Selection Operator for variable selection and shrinkage in the Cox model [2]. This method is also known as the LASSO, and produces coefficients that are shrunken towards zero but also coefficients that are exactly zero.

The original LASSO minimizes the residual sum of squares subject to the sum of the absolute values of the coefficients being less than a certain constant. Because of this constraint, this method shrinks some coefficients and set others to be exactly zero and hence uses the good features of both subset selection and shrinkage.

The LASSO is attractive as a regularization method because it simultaneously performs variable selection and shrinkage. It shrinks all regression coefficients towards zero and automatically sets many of them exactly to zero, depending on the amount of regularization employed. This can be especially useful in high-dimensional data, in which there are more regression coefficients than observations. In this case strong variable selection is desirable in order to obtain an interpretable prediction rule, and shrinkage is desirable to prevent overfit.

Consider the data $(Y_i, x_{ij}), i = 1, \ldots, n$, where x_{ij} is the explanatory variable j for patient i and Y_i is the event time for patient i. Then for a linear regression model of the form $\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j$ the LASSO estimate for β is defined by:

$$\hat{\beta} = \min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq s \tag{3.1}$$

where $s \geq 0$ is a tuning parameter. s is also known as the shrinkage parameter. Note that the function

to minimize is equal to the residual sum of squares.

This method is similar to the method of ridge regression, since only the penalty function differs. In the case that the penalty function equals $\sum_{j=1}^{p} \beta_j^2 \leq s$, the method is known as ridge regression.

Tibshirani proposed to use the method of LASSO on the Cox proportional hazards model by implementing the constraints of equation 3.1 to the partial likelihood function [2]. This results in the following optimization problem:

$$\hat{\beta} = \min_{\beta} \ell_{pc}(\beta) \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq s \tag{3.2}$$

where $\ell_{pc}(\beta)$ is the log-partial-likelihood for right-censored data as in equation 2.16. Another representation of the optimization problem in 3.2 is the Lagrangian form:

$$\hat{\beta} = \min_{\beta} \ell_{pc}(\beta) + \alpha \sum_{j=1}^{p} |\beta_j| \tag{3.3}$$

where $\ell_{pc}(\beta) + \alpha \sum_{j=1}^{p} |\beta_j|$ is called the penalized partial likelihood of the model, and will therefore be notated as $\ell_{pen}(\beta)$. Note that the second part of the penalized likelihood function is equal to the L_1-norm. In this function $\alpha > 0$ is the tuning parameter, also known as the shrinkage parameter. This parameter is strongly related to the shrinkage parameter s.

Computing the LASSO estimate requires solving a quadratic programming problem with linear inequality constraints. Solving this problem requires the use of certain algorithms, see section 4. Before those algorithms can be used, it is worthwhile to have a closer look at the penalized likelihood function $\ell_{pen}(\beta)$ to see what characteristics it has.

3.1. Characteristics of the penalized partial likelihood function

To minimize the penalized likelihood function, it is important to know more about characteristics of the function to see if it meets the conditions needed to apply certain algorithms for minimizing functions. In section 4 those algorithms will be discussed.

Note that the penalized likelihood function is the sum of two terms. The first term is the minus-log-partial-likelihood function from equation 2.15 and the second term is called the penalty function, which is equal to the L_1-norm.

This minus-log-partial-likelihood-function is well behaved, since it is a convex function and it is at least twice differentiable. In definition 1, a convex function is defined.

Definition 1. *A function $f : X \to \mathbb{R}$ is called **convex** if $\forall\ x, y \in X$ and $t \in [0, 1]$ holds that:*
$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$

Theorem 1. *The minus-log-partial-likelihood function is a convex function*

Proof. Let $\beta, \gamma \in \mathbb{R}^p$ and $t \in [0,1]$. Then:

$$\begin{aligned}
\ell_{pc}(t\beta + (1-t)\gamma) &= \sum_{i \in I} \left(\ln \left(\sum_{k=i}^{n} e^{(t\beta + (1-t)\gamma)^T X_k} \right) - (t\beta + (1-t)\gamma)^T X_i \right) \\
&= \sum_{i \in I} \left(\ln \left(\sum_{k=i}^{n} e^{(t\beta)^T X_k} \cdot e^{((1-t)\gamma)^T X_k} \right) - t\beta^T X_i - (1-t)\gamma^T X_i \right) \\
&= \sum_{i \in I} \left(\ln \left(\sum_{k=i}^{n} |e^{(t\beta)^T X_k} \cdot e^{((1-t)\gamma)^T X_k}| \right) - t\beta^T X_i - (1-t)\gamma^T X_i \right)
\end{aligned}$$

For proving the inequality, Hölder's inequality is used. Hölder's inequality states that for $p, q > 1$ with $\frac{1}{p} + \frac{1}{q} = 1$ and $a_k, b_k \in \mathbb{R}$ there holds that:

$$\sum_{k=1}^{n} |a_k b_k| \leq \left(\sum_{k=1}^{n} |a_k|^p \right)^{\frac{1}{p}} \left(\sum_{k=1}^{n} |b_k|^q \right)^{\frac{1}{q}}$$

Since $t \in [0,1]$, Hölder's inequality can be used with $\frac{1}{p} = t$ and $\frac{1}{q} = 1 - t$, which gives that:

$$\begin{aligned}
\ell_{pc}(t\beta + (1-t)\gamma) &\leq \sum_{i \in I} \left(\ln \left(\left(\sum_{k=i}^{n} |e^{t\beta^T X_k}|^{\frac{1}{t}} \right)^t \cdot \left(\sum_{k=i}^{n} |e^{(1-t)\gamma^T X_k}|^{\frac{1}{1-t}} \right)^{1-t} \right) - t\beta^T X_i - (1-t)\gamma^T X_i \right) \\
&= \sum_{i \in I} \left(\ln \left(\left(\sum_{k=i}^{n} |e^{\beta^T X_k}| \right)^t \cdot \left(\sum_{k=i}^{n} |e^{\gamma^T X_k}| \right)^{1-t} \right) - t\beta^T X_i - (1-t)\gamma^T X_i \right) \\
&= \sum_{i \in I} \left(\ln \left(\sum_{k=i}^{n} |e^{\beta^T X_k}| \right)^t + \ln \left(\sum_{k=i}^{n} |e^{\gamma^T X_k}| \right)^{1-t} - t\beta^T X_i - (1-t)\gamma^T X_i \right) \\
&= \sum_{i \in I} \left(t \ln \left(\sum_{k=i}^{n} e^{\beta^T X_k} \right) - t\beta^T X_i + (1-t) \ln \left(\sum_{k=i}^{n} e^{\gamma^T X_k} \right) - (1-t)\gamma^T X_i \right) \\
&= t \sum_{i \in I} \left(\ln \left(\sum_{k=i}^{n} e^{\beta^T X_k} \right) - \beta^T X_i \right) + (1-t) \sum_{i \in I} \left(\ln \left(\sum_{k=i}^{n} e^{\gamma^T X_k} \right) - \gamma^T X_i \right) \\
&= t\ell_{pc}(\beta) + (1-t)\ell_{pc}(\gamma)
\end{aligned}$$

There follows that indeed, the log-partial-likelihood function is a convex function. □

The second term, which is the penalty function $P(\beta) = \alpha \sum_{j=1}^{p} |\beta_j|$, is less well behaved than $\ell_{pc}(\beta)$. Note that this penalty function is the L_1-norm. A plot of this function can be seen in figure 3.1. As can be seen in this figure, the penalty function is only differentiable in the case that $\beta_j \neq 0$. Besides that, the penalty function is a convex function.

Theorem 2. *The penalty function is a convex function*

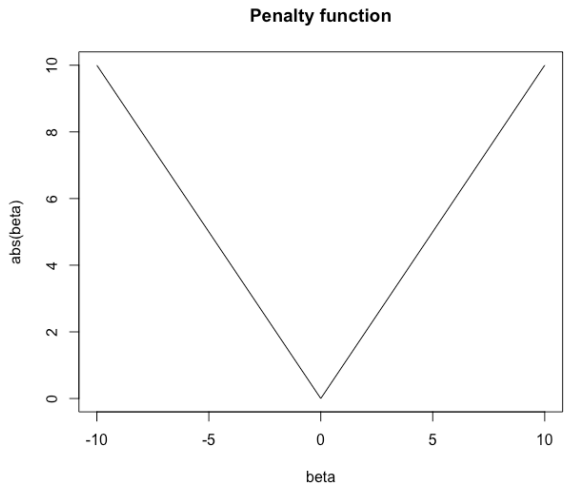

Figure 3.1

Proof. Let $\beta, \gamma \in \mathbb{R}^p$ and $t \in [0,1]$. Then using the triangle inequality there holds that:

$$P(t\beta + (1-t)\gamma) = \alpha \sum_{j=1}^{p} |t\beta_j + (1-t)\gamma_j| \tag{3.4}$$

$$\leq \alpha \sum_{j=1}^{p} \left(t|\beta_j| + (1-t)|\gamma_j|\right) \text{ (triangle inequality)} \tag{3.5}$$

$$= \alpha t \sum_{j=1}^{p} |\beta_j| + \alpha(1-t) \sum_{j=1}^{p} |\gamma_j| \tag{3.6}$$

$$= tP(\beta) + (1-t)P(\gamma) \tag{3.7}$$

And therefore, the penalty function is a convex function. □

Now since both $\ell_{pc}(\beta)$ and the penalty function $P(\beta)$ are convex functions and the penalized likelihood function is the sum of those functions, the penalized likelihood function is a convex function as well.

Also note that since the penalty function $P(\beta)$ is not differentiable everywhere, the penalized likelihood function $\ell_{pen}(\beta)$ is also not differentiable everywhere. This is something that has to be dealt with when searching for the solution of optimization problem 3.3.

3.2. The LASSO-plot

When the LASSO-method is applied to data, a way to visualize the results is in a plot where the regression coefficients are plotted against the shrinkage parameter α. An example of such a plot is found in figure 3.2. This figure is made using the `penalized`-package [7] in the statistical software package `R`. In his package J. Goeman uses the notation λ_1 for the shrinkage parameter α [7]. For the implementation that resulted in this plot, see section 5.1 and Appendix A.

There are a few things that are important to notice about this plot. To begin with, the coefficients where $\alpha = 0$ equal the coefficients for the optimization problem from equation 3.3 where $\alpha = 0$. These coefficients are the solution to the unpenalized optimization problem where no variable selection or shrinkage is involved.

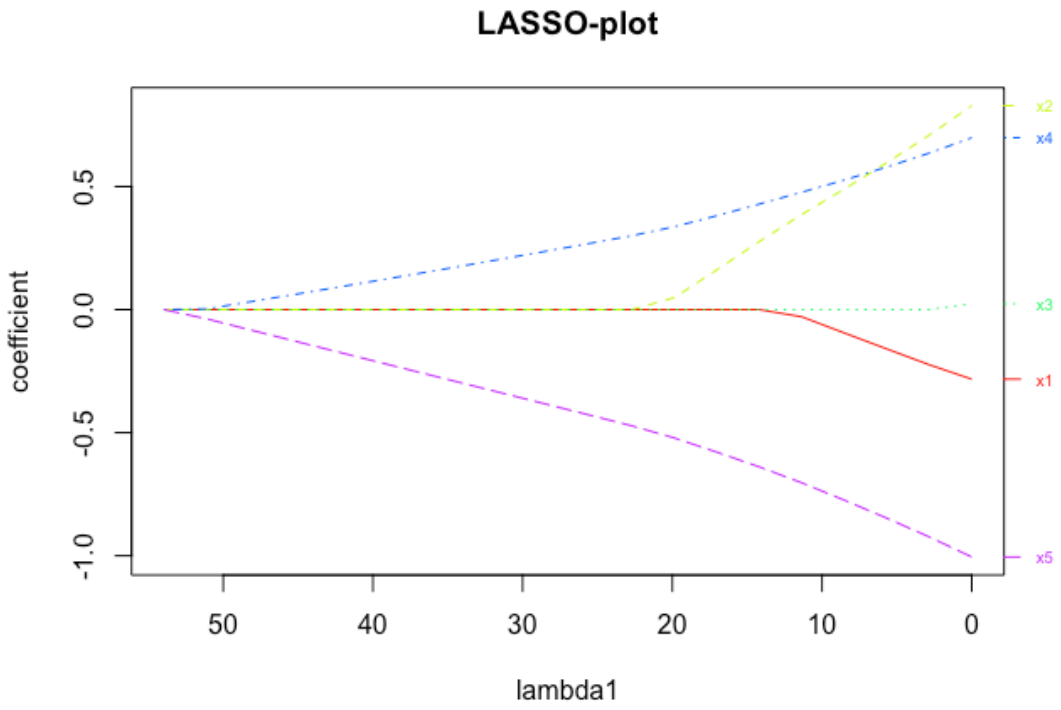

Figure 3.2: A LASSO-plot

Secondly, it is important to notice that as the shrinkage parameter increases, all coefficients are shrunken towards zero. Eventually the coefficients are exactly zero, which is a result of the variable selection in the method of LASSO. So for a certain value of α one can see in this plot which coefficients are set equal to zero and which coefficients are shrunken. For example, for $\alpha = 30$, one can see that the variables $x1, x2$ and $x3$ are already set equal to zero. So for that value of α only the variables $x4$ and $x5$ should be taken in consideration.

3.3. Choosing the shrinkage parameter

As can be seen in figure 3.2 the shrinkage parameter α determines what variables should be taken in consideration. Robert Tibshirani proposed to determine this parameter using cross validation [2]. Cross-validation is one of the simplest and most widely used methods for estimating prediction errors to determine a parameter [8]. In cross-validation it is typical to test the performance of the predicted model on another part of the data that is not yet used for the estimation.
In k-fold cross validation the data is split up into k parts so that $k-1$ parts can be used to fit the model and the other part can be used to test the model. The last part is then used to calculate the prediction error of the fitted model. This prediction error is then dependent of the parameter to be estimated. The parameter can then be found by minimizing the prediction error.

Next to Robert Tibshirani [2], J. Goeman also uses the method of cross validation to select the shrinkage parameter [7]. He also implemented this in the `penalized`-package [7]. In the case of the data which was used to produce figure 3.2, this results in a value of $\alpha = 1,814474$ and variable $x3$ that is already shrunken to zero, see Appendix A.

4

Algorithms for computing the parameter estimates

In section 2.2 it was proposed to estimate the regression parameters of the Cox proportional hazards model by maximum likelihood estimation. The likelihood function and the partial likelihood function were derived. Since it is often easier to find the optimal points for the log-likelihood function instead of finding the optima for the likelihood function, the minus-log likelihoods were computed. It was also argued that the partial likelihood is preferred over the normal likelihood and also censoring was introduced for the partial likelihood. Those likelihoods have not yet been used to estimate the regression parameters, since there are algorithms needed to solve the optimization problem. The optimization problem that needs to be solved is given by:

$$\min_{\beta} \ell_{pc}(\beta) = \min_{\beta} \sum_{i \in J} \left(\ln \left(\sum_{k=i}^{n} e^{\beta^T X_k} \right) - \beta^T X_i \right) \tag{4.1}$$

In section 3 it was proposed to add constraints to the optimization problem in 4.1 in order to apply variable selection and shrinkage. For applying the LASSO method, the following quadratic programming problem with linear constraints should be solved:

$$\min_{\beta} \ell_{pc}(\beta) \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq s \tag{4.2}$$

where $s \geq 0$ is the tuning parameter.

In order to solve the optimization problem in equation 4.1 Robert Tibshirani proposed to use Newton's method [2]. In order to solve the last optimization problem J. Goeman proposed to use a combination of Newton's method and the Gradient Descent algorithm [7].

4.1. Newton's Method

In order to solve the unconstrained optimization problems, Robert Tibshirani proposed to use Newton's Method [2]. Newton's method is an iterative algorithm for estimating the minimum of an at least twice differentiable and convex function.

Let β^0 be an initial guess to the solution of optimization problem 4.1. Then Newton's method fits a paraboloid at β^0 and searches for the minimum point of that paraboloid. That minimum is then used to find β^1, then again a parabola is fitted at β^1. This is done until the sequence of $\beta^0, \beta^1, \ldots$ converges to a vector $\hat{\beta}$. The main idea of Newton's method in the 1-dimensional case can be found in figure 4.1. It is clear that in the case this paraboloid should be an approximation of the function to be minimized,

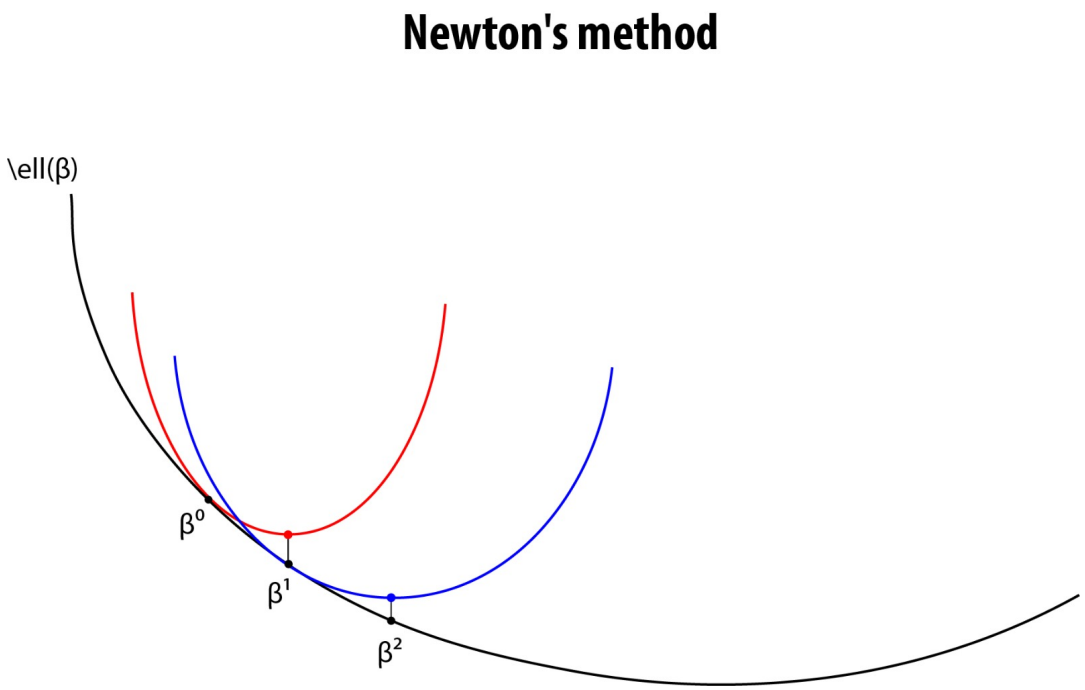

Figure 4.1: Visualization of Newton's method in 1-dimensional case

this function should be a convex function and should be at least twice differentiable.

Suppose β^k is known, then in order to find the next point in the sequence β^{k+1}, the second order Taylor approximation is computed to fit a parabola at β^k.
Let

$$u_j(\beta) = \frac{\partial}{\partial \beta_j}\ell(\beta),\ u(\beta) = \nabla\ell(\beta)$$

and

$$A_{ij}(\beta) = \frac{\partial^2}{\partial \beta_i \partial \beta_j}\ell(\beta),\ A(\beta) = \Delta\ell(\beta)$$

Then the second order Taylor approximation of $\ell(\beta)$ at β^k equals:

$$\ell_q(\beta^{k+1}) = \ell(\beta^k) + (\beta^{k+1} - \beta^k)^T u(\beta^k) + \frac{1}{2}(\beta^{k+1} - \beta^k)^T A(\beta^k)(\beta^{k+1} - \beta^k) \tag{4.3}$$

where $u(\beta^k)$ is the gradient of $\ell(\beta)$ at β^k and $A(\beta^k)$ is the hessian matrix of $\ell(\beta)$ at β^k.

Then in order to find the minimum of this parabola, which is defined as β^{k+1}, the derivative of $\ell_q(\beta^{k+1})$ must be equal to zero. This derivative equals:

$$\begin{aligned}\nabla\ell_q(\beta^{k+1}) &= \nabla\left(\ell(\beta^k) + (\beta^{k+1} - \beta^k)^T u(\beta^k) + \frac{1}{2}(\beta^{k+1} - \beta^k)^T A(\beta^k)(\beta^{k+1} - \beta^k)\right) \\ &= u(\beta^k) + \frac{1}{2}\cdot 2A(\beta^k)(\beta^{k+1} - \beta^k) \\ &= u(\beta^k) + A(\beta^k)(\beta^{k+1} - \beta^k)\end{aligned}$$

By computing the roots of this derivative there follows that β^{k+1} can be found by solving:

$$A(\beta^k)\beta^{k+1} = A(\beta^k)\beta^k - u(\beta^k)$$

Assuming that matrix $A(\beta^k)$ is nonsingular, there follows that:

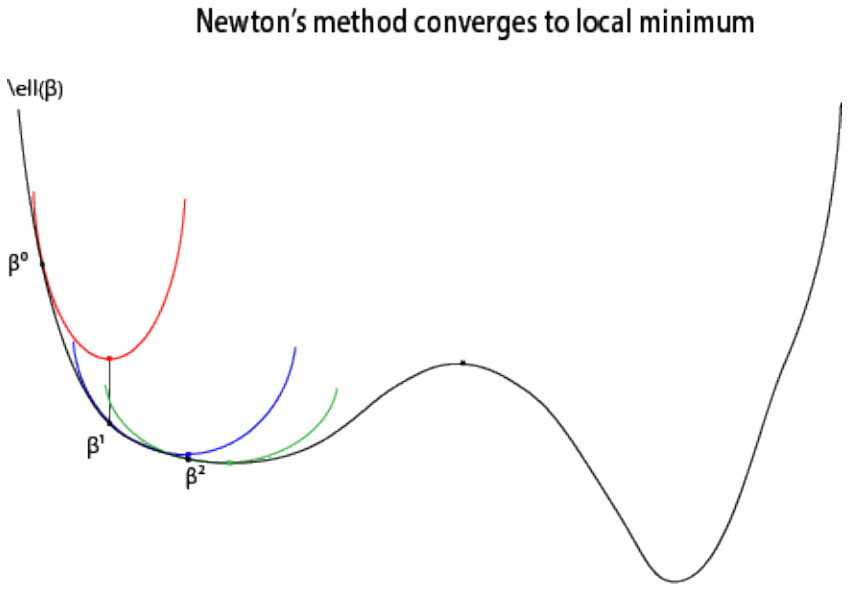

Figure 4.2: Newton's method converges to local minimum

$$\beta^{k+1} = \beta^k - A^{-1}(\beta^k)u(\beta^k)$$

Summarizing, the estimations of the regression parameters β can be found by using Newton's method, which is given by:

Newton's method:

1. Set β^0 equal to 0 for $k = 0$
2. Compute the the gradient $u(\beta^k)$ and the hessian $A(\beta^k)$
3. Let β^{k+1} be the solution of $\min \ell_q(\beta^k)$
4. Repeat steps 2 and 3 until β^k does not change

One thing that should be noticed is that Newton's method is not guaranteed to converge, since the efficiency of the method is strongly dependent on the initial guess. For example, in case that the function to be minimized contains several local optima, it is possible that Newton's method converges to an optimum that is not the global optimum, see figure 4.2. In this case the method converges, but not to the solution that is wanted.

Another issue that could occur is when the initial guess has a derivative equal to zero, for example when the initial guess is equal to a local minimum or maximum. In such a case, the method does not converge.

4.2. Gradient Descent algorithm

In order to solve the constrained optimization problem in equation 4.2, J. Goeman proposed to use a combination of the Gradient Descent algorithm and Newton's method [7]. In his article, J. Goeman first applies the Gradient Descent algorithm and then afterwards the Gradient Descent algorithm is combined with Newton's method. In this thesis, the approach will be similar.

The gradient descent is a first-order iterative optimization algorithm to find the minimum of a function using the gradient of the function. To find a local minimum, the gradient descent algorithm makes a step proportional to the negative of the gradient of the function in a current point. The gradient descent is also known as the method of steepest descent.

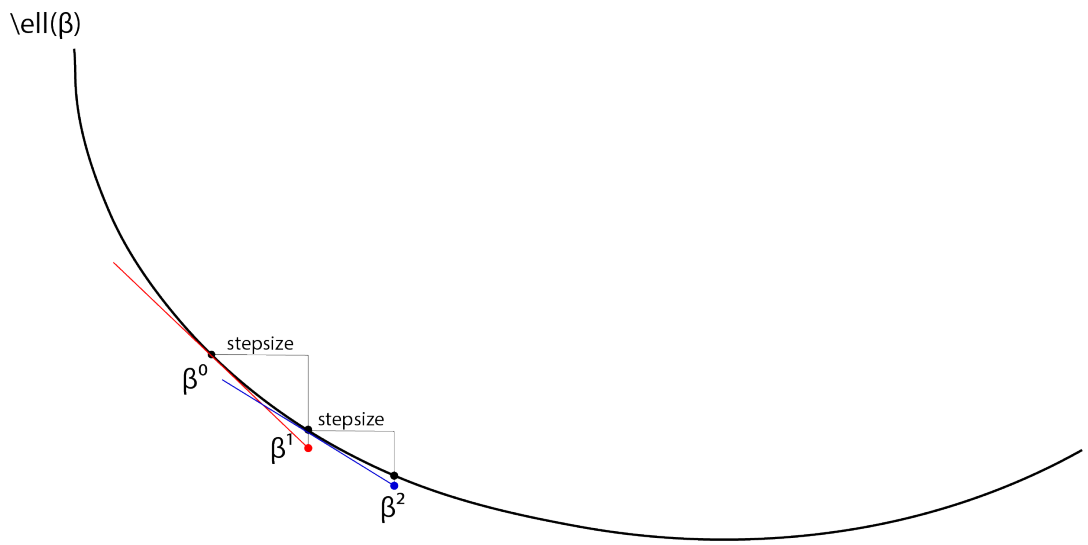

Figure 4.3

Let β^0 be an initial guess to the optimization problem in 4.2. Then the Gradient Descent algorithm computes the gradient of $\ell_{pen}(\beta)$ at β^0. Since the penalized likelihood function should be minimized, one is interested in the direction where the gradient of the function is going downhill. Then in order to find β^1 there is moved in the direction of the gradient that is going downhill with a stepsize t. In the one dimensional example, one moves in the direction of the derivative in case the derivative is negative and one moves in the negative direction of the gradient in the case the derivative is positive. Then the gradient is determined at β^1 and then there will be moved in the downhill direction again in order to find β^2. This is done until the sequence $\beta^0, \beta^1, \ldots$ converges to a vector $\hat{\beta}$ which satisfies $v(\hat{\beta}) = 0$. The point in the sequence $\beta^0, \beta^1, \ldots$ can then be found by:

$$\beta^{k+1} = \beta^k - t(\beta^k) \cdot v(\beta^k)$$

where $t(\beta^k)$ is the stepsize, which can differ at each step and $v(\beta^k)$ is the gradient of $\ell_{pen}(\beta)$ in β^k. The main idea of the gradient descent algorithm can be found in figure 4.3.

Note that for applying this algorithm the function to be minimized should be at least once differentiable and should be convex. In section 3.1 there was proved that indeed the penalized likelihood function is a convex function. But in that section there was also noted that the penalty function is not everywhere differentiable. So when applying this algorithm the lack of differentiability is a thing that has to be dealt with. For doing this, the original gradient descent algorithm should be adapted a little.

Since the penalized likelihood function is only differentiable in the case that $\beta_j \neq 0 \ \forall \ j = 1, \ldots, p$, the gradient can in that case be easily determined by:

$$\begin{aligned} v(\beta) &= \lim_{t \downarrow 0} \frac{\ell_{pen}(\beta + t) - \ell_{pen}(\beta)}{t} = \lim_{t \downarrow 0} \frac{\ell_{pc}(\beta + t) - \ell_{p}(\beta)}{t} + \lim_{t \downarrow 0} \frac{P(\beta + t) - P(\beta)}{t} \\ &= u(\beta) + \lim_{t \downarrow 0} \frac{P(\beta + t) - P(\beta)}{t} \end{aligned}$$

where $u(\beta)$ is the gradient of the unpenalized likelihood function.

Then for $\beta_j > 0$, there holds that:

$$v_j(\beta) = u_j(\beta) + \lim_{t \downarrow 0} \frac{P(\beta_j + t) - P(\beta_j)}{t} = u_j(\beta) + \lim_{\epsilon \downarrow 0} \alpha \frac{\beta_j + t - \beta_j}{t} = u_j(\beta) + \alpha$$

And for $\beta_j < 0$, there holds that:

$$v_j(\beta) = u_j(\beta) + \lim_{t \downarrow 0} \frac{P(\beta_j + t) - P(\beta_j)}{t} = u_j(\beta) + \lim_{t \downarrow 0} \alpha \frac{-(\beta_j + t) + \beta_j}{t} = u_j(\beta) - \alpha$$

Since the penalized likelihood function is not differentiable for any $\beta_j = 0$, it is only possible to define a directional derivative for every β in every direction $w \in \mathbb{R}^p$ by:

$$\ell'_{pen}(\beta, w) = \lim_{t \downarrow 0} \frac{\ell_{pen}(\beta + tw) - \ell_{pen}(\beta)}{t}$$

Since the penalized likelihood function consists of two functions, of which one is differentiable and the other is not differentiable for $\beta_j = 0$, the direction will be based on the direction of the differentiable function, which is in this case the unpenalized likelihood function. Therefore, we choose to define the gradient the penalized likelihood function in the case that $\beta_j = 0$ as:

$$v_j(\beta) = \alpha + \text{sign}(u_j(\beta)) \text{ if } |u_j(\beta)| \geq \alpha$$

Summarizing, the gradient of the penalized likelihood can be calculated by:

$$v_j(\beta) = \begin{cases} u_j(\beta) + \alpha & \text{if } \beta_j > 0 \\ u_j(\beta) - \alpha & \text{if } \beta_j < 0 \\ u_j(\beta) + \alpha \cdot \text{sign}(u_j(\beta)) & \text{if } \beta_j = 0 \text{ and } |u_j(\beta)| \geq \alpha \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where

$$\text{sign}(u_j) = \begin{cases} 1 & \text{if } u_j(\beta) > 0 \\ 0 & \text{if } u_j(\beta) = 0 \\ -1 & \text{if } u_j(\beta) < 0 \end{cases}$$

The discontinuities of the gradient divide the domain of $\ell_{pen}(\beta)$ into 3^p subdomains in which the gradient is continuous. In the Gradient Descent algorithm, this gradient can only be used within a certain subdomain, because of this discontinuities. To stay in such a subdomain non of the β_j^{k+1}'s can equal zero, which gives that:

$$\beta^{k+1} = \beta^k - t(\beta^k)v(\beta^k) \neq 0 \ \forall \ j = 1, \ldots, p$$

This results in a constraint for the step size $t(\beta^k)$, since $t(\beta^k)$ can not be bigger than the value for $t(\beta^k)$ where one of the β_j^k equals zero. Therefore this constraint equals $0 < t(\beta^k) < t_{edge}(\beta^k)$ with:

$$t_{edge}(\beta^k) = \min_j \left\{ \frac{\beta_j^k}{v(\beta_j^k)} : \text{sign}(\beta_j^k) = -\text{sign}(\beta_j^k) \neq 0 \right\}$$

In order to choose the optimal step size for a step in the gradient descent algorithm, second order Taylor approximations for $\ell_{pen}(\beta^k)$ are used to approximate $\ell_{pen}(\beta^{k+1})$.

Therefore, the second derivative of the penalized likelihood function should be determined. Since the penalty function is not differentiable, the second derivate can only be determined within a certain subdomain of gradient continuity. Within such a subdomain, the second derivative of the penalty function is zero, so the second derivative of the penalized likelihood function equals the second derivative of the unpenalized likelihood function, which is:

$$\ell''_{pen}(\beta, w) = w^T A(\beta) w \tag{4.5}$$

where $A(\beta)$ is as described in section 4.1.

In the gradient descent algorithm, the second order Taylor approximations are used at each step to approximate $\ell_{pen}(\beta^{k+1})$ locally form β^k in the direction of the gradient. Then using the defined gradient as in equation 4.4 and the hessian defined as in equation 4.5, the second order Taylor approximation for $\ell_{pen}(\beta)$ in β^k is given by:

$$\ell_{\tilde{q}}(\beta^{k+1}) = \ell_{pen}(\beta^k) + (\beta^{k+1} - \beta^k)^T v(\beta^k) + \frac{1}{2}(\beta^{k+1} - \beta^k)^T A(\beta^k)(\beta^{k+1} - \beta^k)$$

Using that $\beta^{k+1} = \beta^k - t(\beta^k)v(\beta^k)$ according to the gradient descent algorithm, gives that $\beta^{k+1} - \beta^k = -t(\beta^k)v(\beta^k)$, which gives that this second order Taylor approximation equals:

$$\ell_{\tilde{q}}(\beta^{k+1}) = \ell_{pen}(\beta^k) - t(\beta^k)v(\beta^k)^T v(\beta^k) + \frac{1}{2}t(\beta^k)^2 v(\beta^k)^T A(\beta^k)v(\beta^k) \tag{4.6}$$

Then the optimum of the Taylor approximation can be found by taking the derivative of 4.6 with respect to $t(\beta^k)$ and equal that to 0. The derivative of the Taylor approximation equals:

$$\begin{aligned}\nabla\left(\ell_{\tilde{q}}(\beta^{k+1})\right) &= \nabla\left(\ell_{pen}(\beta^k) - t(\beta^k)v(\beta^k)^T v(\beta^k) + \frac{1}{2}t(\beta^k)^2 v(\beta^k)^T A(\beta^k)v(\beta^k)\right) \\ &= -v(\beta^k)^T v(\beta^k) + t(\beta^k)v(\beta^k)^T A(\beta^k)v(\beta^k)\end{aligned} \tag{4.7}$$

Equal this derivative to 0 gives that the optimum of the Taylor approximation of $\ell_{pen}(\beta^k)$ can be found at a value of $t_{opt}(\beta^k)$ with

$$t_{opt}(\beta^k) = \frac{v(\beta^k)^T v(\beta^k)}{v(\beta^k)^T A(\beta^k)v(\beta^k)}$$

Then for finding the minimum value of the penalized likelihood function, the step size $t_{opt}(\beta^k)$ should be chosen in order to find the optimum. But if this step size is greater than $t_{edge}(\beta^k)$, one will not stay within a subdomain of gradient continuity, which is not allowed. Therefore, the next point in the sequence $\beta^0, \beta^1, \ldots$ can be found by:

$$\beta^{k+1} = \beta^k - \min\left\{t_{opt}(\beta^k), t_{edge}(\beta^k)\right\} \cdot v(\beta^k) \tag{4.8}$$

Summarizing, the the Gradient Descent algorithm can be found below:

Gradient descent algorithm

1. Set β^0 equal to an initial guess for $k = 0$
2. Compute the gradient $v(\beta^k)$ for β^k
3. Let $\beta^{k+1} = \beta^k - \min\left\{t_{opt}(\beta^k), t_{edge}(\beta^k)\right\} \cdot v(\beta^k)$
4. Repeat steps 2 and 3 until β^k does not change

Similar to Newton's Method, convergence of the Gradient Descent algorithm is not guaranteed. Again, the convergence is strongly dependent on the initial guess of the solution. If the initial guess is near a local minimum, it is likely the algorithm does not find the global minimum of the function. And also the algorithm will not converge in the case the initial guess is equal a point with zero gradient.

But for the Gradient Descent algorithm, the chosen step size is of importance too. When a step size is chosen that is too large, algorithm may jump over the minimum and will continue 'zigzagging' to the solution, see figure 4.4. By determining $t_opt(\beta)$ this problem does not arise in this application.

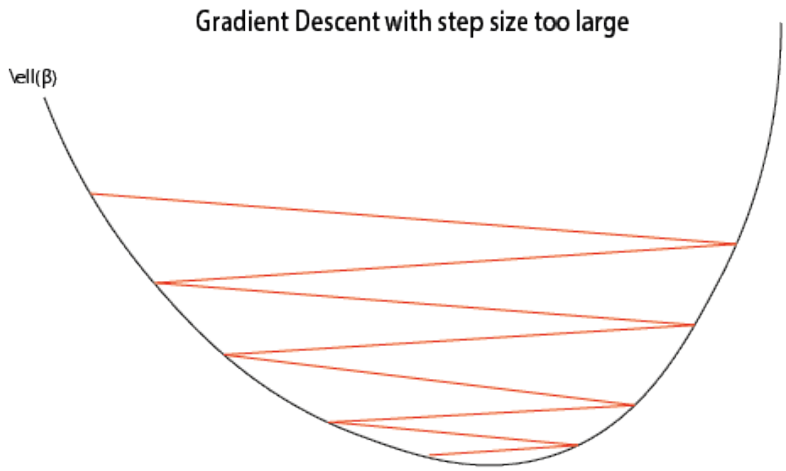

Figure 4.4

4.2.1. Combined Gradient Descent and Newton's method

In the previous section, the Gradient Descent algorithm is introduced to solve the optimization problem in equation 4.2. One main disadvantage of the Gradient descent algorithm, is that it requires a large number of steps to converge [7]. This can be avoided by giving the algorithm the option to switch to Newton's method, which is known to converge much faster than the Gradient Descent algorithm [7].

As was derived in section 4.1, a step according to Newton's method is given by:

$$\beta^{k+1} = \beta^k - A(\beta^k)^{-1} v(\beta^k)$$

where $A(\beta^k)$ is the hessian of the penalized likelihood function function. In order to use Newton's method, the function to be minimized should be at least twice differentiable and a concave function [7]. In section 3.1 there was proved that indeed the penalized likelihood function is convex, but it is not everywhere twice differentiable, but as we have seen in section 4.2, the penalized likelihood function is at least twice differentiable in a subdomain of the gradient continuity. Therefore, within a certain subdomain, Newton's method can be used.

So suppose the vector β^k is known and suppose that $t_{opt}(\beta^k) < t_{edge}(\beta^k)$, then while computing β^{k+1} one will stay in the subdomain of gradient continuity. Then for this step, Newton's method can be used, instead of the Gradient Descent algorithm. Note that this can only be a valid step in the case that for every value of $j = 1, \dots, p$ holds that $\text{sign}(\beta_j^k) = \text{sign}(\beta_j^{k+1})$. If that would not be the case, the Gradient Descent algorithm should be used. In the case that $t_{opt}(\beta^k) > t_{edge}(\beta^k)$ the Gradient Descent algorithm should be used to jump to the next subdomain of gradient continuity.

One problem that may arise is that β_j^k set equal to zero for one or more $j = 1, \dots, p$. In such a case, it is not possible to make a step according to Newton's method, since the Hessian matrix would not be invertible. This problem can be avoided by dismissing the β_j^k's that equal zero.
Let $J = \{j : \beta_j \neq 0\}$ be the index set of the active variables. Then constrained to a single subdomain, the target function $\ell_{pen}(\beta)$ can be viewed as an m-dimensional function, with $m = \#J \leq p$.

For this m-dimensional function the gradient and the hessian can be computed. Let $\tilde{\beta^k} = (\beta^k_{j_1}, \ldots, \beta^k_{j_m})$ the vector of β^k_j's unequal to 0. Then the gradient of the m-dimensional equals $v(\tilde{\beta^k})$ and the Hessian $A(\tilde{\beta^k})$ can be computed by:

$$A_{ij}(\tilde{\beta^k}) = \frac{\delta^2}{\delta\hat{\beta}^k_i \delta\hat{\beta}^k_j}\ell_{pen}(\hat{\beta}^k) = \frac{\delta^2}{\delta\hat{\beta}^k_i \delta\hat{\beta}^k_j}\ell_p(\hat{\beta}^k)$$

The next step is then computed by: $\hat{\beta}^{k+1} = \hat{\beta}^k - A^{-1}(\hat{\beta^k})v(\tilde{\beta^k})$. This m-dimensional vector can then be mapped back to a p-dimensional vector by augmenting $\hat{\beta}^{k+1}$ with zero's for all non-active variables.

Summarizing, the combined algorithm based on the Gradient Descent algorithm and Newton's method is:

The combined Gradient Descent and Newton's method algorithm:

1. Set β^0 equal to an initial guess for $k = 0$
2. Compute the gradient $v(\beta^k)$ and the Hessian $A(\beta^k)$ for β^k
3. Let

$$\beta^{k+1} = \begin{cases} \beta^k - t_{edge}v(\beta^k) & \text{if } t_{opt} \geq t_{edge} \\ \beta^k - A(\beta^k)^{-1}v(\beta^k) & \text{if } t_{opt} < t_{edge} \text{ and } \text{sign}(\beta^{k+1}_{NR}) = \text{sign}(\beta^k) \\ \beta^k - t_{opt}v(\beta^k) & \text{otherwise} \end{cases} \quad (4.9)$$

4. Repeat steps 2 and 3 until β^k does not change

5

Simulation

In the previous sections, the Cox model was introduced together with methods to fit this model to data. This was done for the unpenalized likelihood function where no LASSO method is applied and for the penalized likelihood function, where the LASSO is applied.

Now that it is known how to fit the Cox model to data, data is needed to implement this. In this section the Cox model will be fitted to a real dataset of breast cancer patients and to data which is generated randomly according to the Cox model.

5.1. Generating data according to the Cox model

Generating some random data according to the Cox model is very instructive to create a feeling with the model. This can be done using the basic concepts survival analysis, which were introduced in section 2.1, and the basic concepts of the Cox model, which were introduced in section 2.2. The implementation of this section can be found in Appendix A.

To begin with, the explanatory variables are randomly generated and stored in matrix X. Then values for β are chosen between -1 and 1.

Now that the variables and the regression parameters are known, the event times can be generated according to the Cox model. Let T denote the random variable for the event time, then by using the characteristics of the Cox model, there can be derived that the cumulative hazard function is standard exponentially distributed, by:

$$\begin{aligned} P(\Lambda(T) \leq v) &= P(T \leq \Lambda^{-1}(v)) = 1 - P(T > \Lambda^{-1}(v)) \\ &= 1 - S(\Lambda^{-1}(v)) = 1 - e^{\Lambda(\Lambda^{-1}(v))} = 1 - e^{-v} \end{aligned}$$

Given that the cumulative hazard function evaluated at T is exponentially distributed, which means that $\Lambda(T) \sim \exp(1)$, gives that the event time T could be generated by the inverse cumulative hazard function. For simplicity, there is assumed that the baseline hazard rate function is constant in this case, which gives that the cumulative hazard rate function equals: $\Lambda(t|X_i) = \lambda_0 e^{\beta^T X_i} t$. Then the event times are distributed according to the inverse of the cumulative hazard rate function, which equals:

$$\Lambda^{-1}(v) = \frac{1}{\lambda_0} e^{-\beta^T X} \cdot v \tag{5.1}$$

Now the data is generated according to the Cox model, and can be used to estimate the regression parameters β. This will be done for either the partial likelihood as the likelihood. This results in different estimations of β, which can be seen in figure 5.1. Note that both methods give good estimations of the chosen value of β.

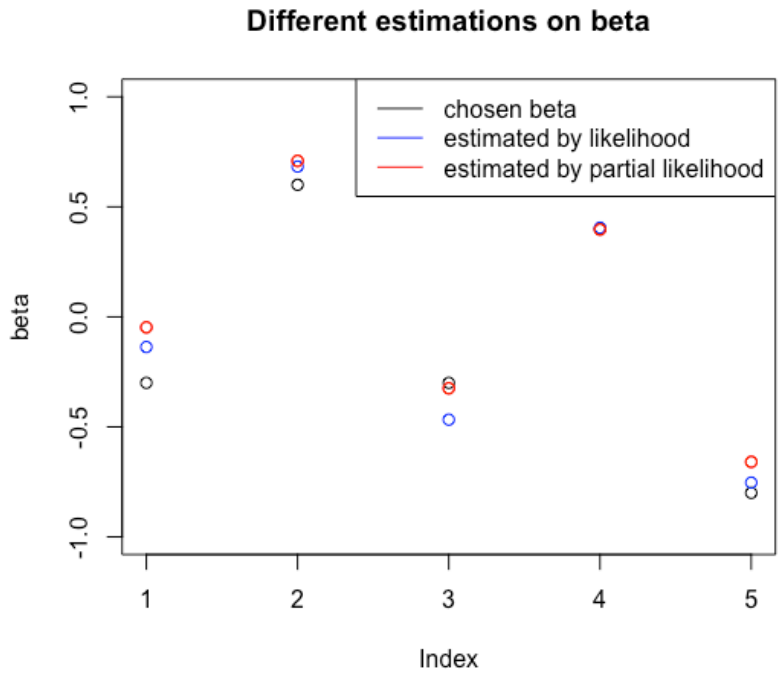

Figure 5.1: Different estimations for beta

	chosen value	likelihood	partial likelihood	coxph
β_1	-0.3	-0.1370292	-0.04789277	-0.04789277
β_2	0.6	0.6828860	0.70859639	0.70859639
β_3	-0.3	-0.4671500	-0.32474223	-0.32474223
β_4	0.4	0.4050273	0.39661637	0.39661637
β_5	-0.8	-0.7532476	-0.65904476	-0.65904476

Table 5.1: Different estimations for betar

In the software package `R`, there is a so called `survival`-package, which can be used to fit the Cox model to data using the `coxph`-function. This function is also used to estimate the regression parameters and as can be seen in table 5.1, it gives exactly the same regression coefficients as the ones estimated using the partial likelihood.

In section 2.2.4 the problem of censored data was introduced. Taking censoring in consideration, means that the estimations of the regression parameters will change. The censoring times are randomly chosen, according to the Cox model. This is similar to choosing the event times. Then Y and δ are determined as was defined in section 2.2.4. Censoring can have a big influence fitting the model to data, which results in different regression parameters, which results in different expectations of survival times. In this example, the censoring causes that the expected lifetime of a patient is longer than when there is no censoring involved, as can be seen in figure 5.2.

5.2. Breast cancer data

In the Netherlands, one in seven women suffers from breast cancer. Women with breast cancer at the same stage - which can be measured by standard measurements such as the diameter of the tumor and the diagnosis age - can have different treatment response and different outcome [3]. So apparently, the stage of disease might not be a good predictor of the outcome of the disease. Instead of looking at the standard clinical and histologic criteria of a patient L. van 't Veer proposed to look at gene expression measurements [3]. Using DNA microarray analysis on primary breast tumors, there

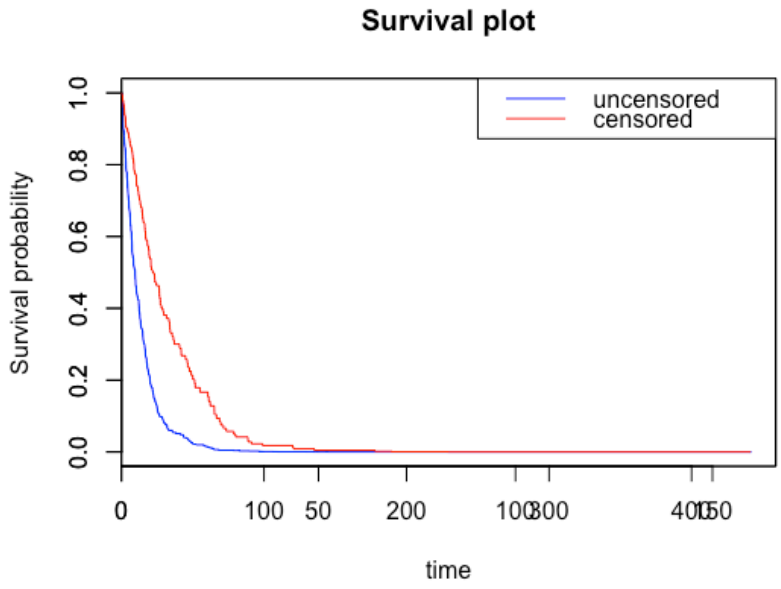

Figure 5.2

was stated that the gene expression measurements are a more powerful predictor of the outcome than the standard clinical and histologic criteria [3].

In the period between 1984 and 1995, 295 patients with breast cancer were monitored at the Netherlands Cancer Institute. Those patients were selected among all patients that were diagnosed with breast cancer according to some criteria: the diameter of the tumor at the moment of diagnose was less than 5 centimeters, the lymph nodes were tumor negative and the age of the patients at moment of diagnose was less than 53 years old. Among all patients 144 of them had a lymph-node-positive disease, which means that lymph nodes in the whole body contain cancer cells, which increases the risk of the cancer spreading. For all patients there are five standard clinical and histologic risk variables and gene expression measurements of 70 genes monitored [4].

In this thesis the data for the patients with a lymph-node-positive disease are analyzed to verify wether the gene expression measurements are a more powerful predictor of the outcome than the standard clinical and histologic criteria [4].

The data consist of 144 patients with a survival time, a variable which denotes whether the data is censored or not, five standard explanatory variables and 70 gene expression measurement variables. The data consists of 48 events and 96 censored observations. The median of the survival time is 7 years and the survival times vary from 0.05 years to 17.7 years. An overview of the survival times can be found in figure 5.3. In the case that the event actually happened within the time the patient was diagnosed, the median is 3.17 years and in the case of censoring the median is 7.8 years.

To compare the impact of the explanatory variables on the survival time, first an analysis will be made of the five standard clinical and histologic variables, and afterwards this will be compared to the analysis of the 70-gene expression measurements.

5.2.1. The standard clinical and histological variables

For the 144 patients diagnosed with a lymph-node-positive disease, five standard clinical and histological variables measured:

- Diam: The diameter of the tumor, which is measured in two levels: Diam ≤ 2 and Diam > 2.
- N: number of affected lymph nodes, which is measured in two levels: $1 \leq N \leq 3$ and $N > 3$

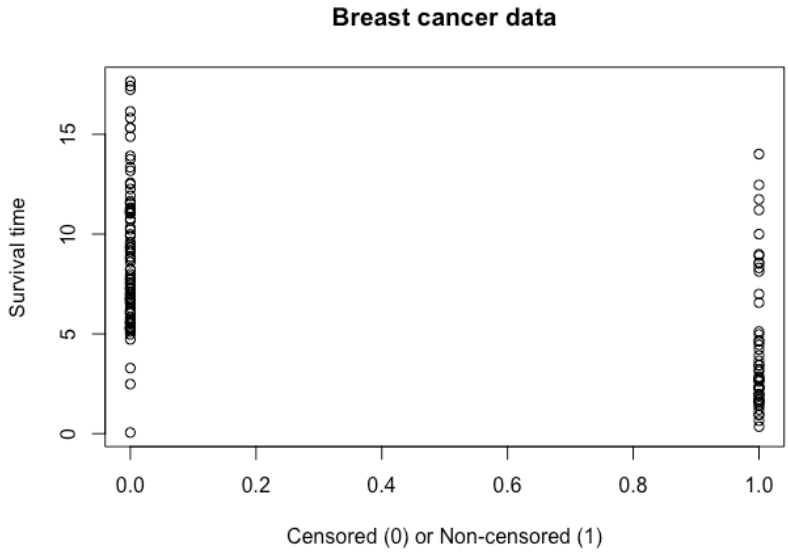

Figure 5.3: Visualization of the data

Variable	Regression parameter
Diam	0.04368370
N	0.05662764
ER	-0.27530076
Grade	-0.19700911
Age	-0.06935407

Table 5.2: Estimated regression parameters for the unpenalized optimization problem

- ER: Estrogen receptor status, which can either be positive or negative
- Grade: the grade of the tumor, which is measured in three levels: Poorly diff < Intermediate < Well diff
- Age: the diagnosis age, which is measured in years

The survival function for this data, which is only based on the events that actually happened and those five variables, can be found in figure 5.4.

Then according to Newton's method, that was explained in section 4.1, the solution to the unpenalized optimization problem can be found in table 5.2

For solving the constrained optimization problem, where the method of LASSO is applied, the combination of the Gradient Descent algorithm and Newton's method is used, as was explained in section 4.2.1. The solution to this optimization problem strongly depends on the value of the shrinkage parameter α. The value of this parameter is determined by cross-validation, as was explained in section 3.3. This method gives that the optimal value for α equals 9.652105. Using this value for estimating the regression parameters, gives that shrinkage and variable selection is applied until only one variable should be taken in consideration. This variable is the diagnosis age and for this value of α the regression parameter of the diagnosis age equals -0.06999444. And therefore the solution the the constrained optimization problem can be found in table 5.3.

The `penalized`-package that was written by J. Goeman [7] is used to make the LASSO-plot, which introduced in section 3.2. This plot can be found in figure 5.5. The grey dotted line represents the value for α that was determined by cross-validation. Note that the diagnosis age is the variable that has most influence on the survival time.

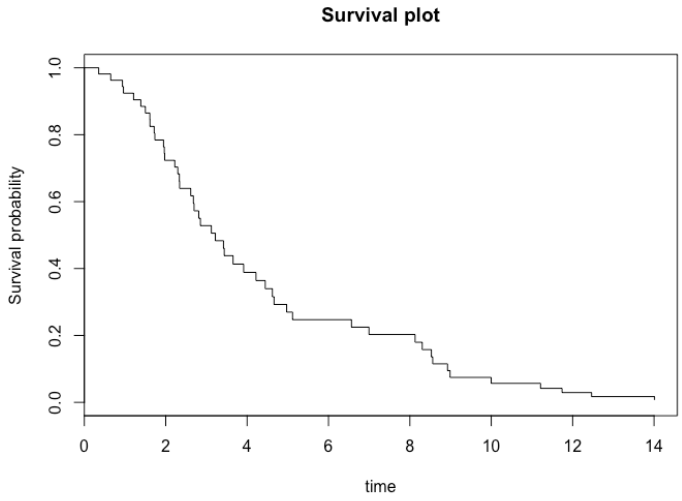

Figure 5.4: Survival plot based on the standard clinical and histological variables

Variable	Regression parameter
Diam	0
N	0
ER	0
Grade	0
Age	-0.06999444

Table 5.3: Estimated regression parameters for the constrained optimization problem

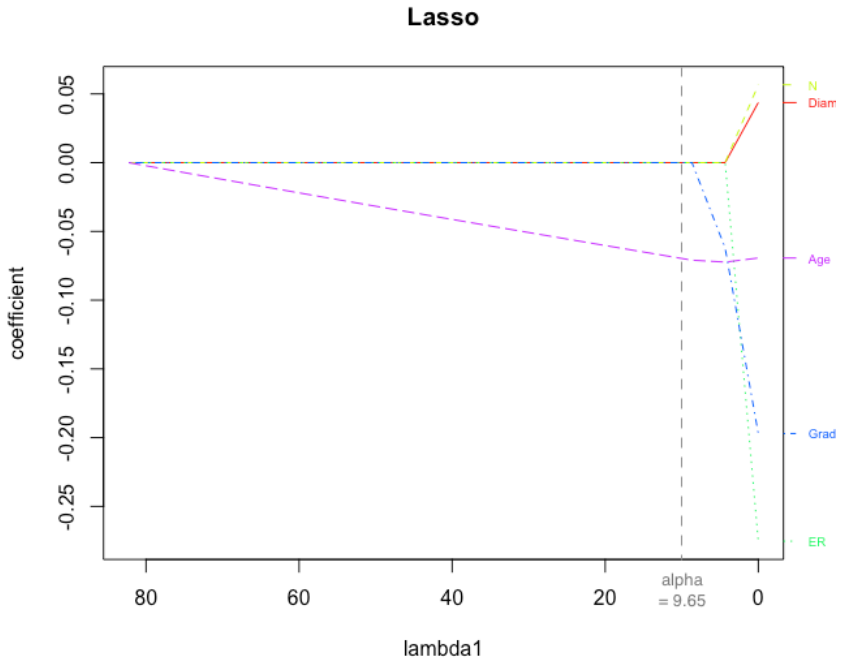

Figure 5.5: LASSO-plot

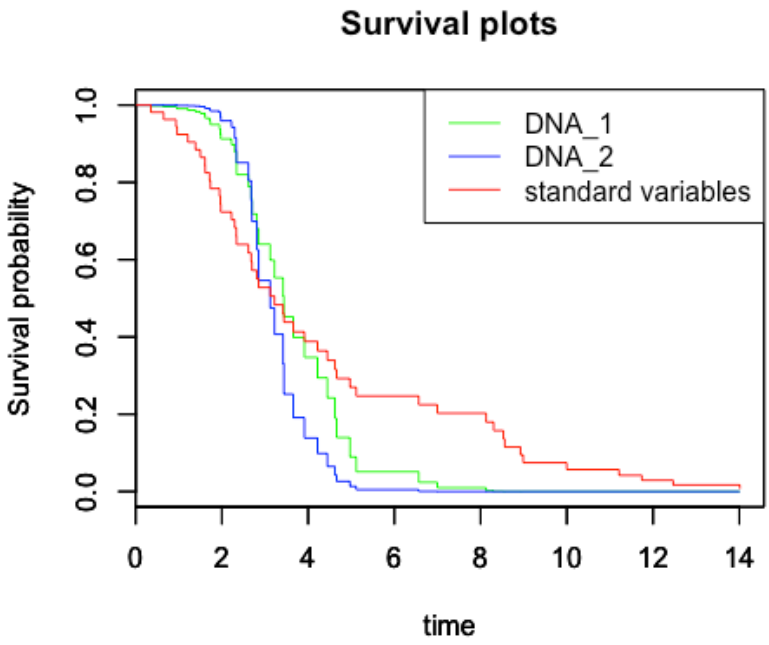

Figure 5.6: Different survival plots

DNA type	Regression parameter
SCUBE2	-0.3793645

Table 5.4: Estimated regression coefficient

5.2.2. The variables based on DNA

Next to the five clinical and histological variables, there are 70 variables based on DNA microarray analysis in the database. Those variables can be found in Appendix B.

Since there are so many variables involved that for solving the unpenalized optimization problem the algorithm does not seem to converge, the data was split into two parts. The first 35 variables are stored in the matrix X_{DNA1} and the other 35 are stored in the matrix X_{DNA2}. Then for each part of the data the regression parameters can be estimated and the survival plots can be plotted. In figure 5.6 there are three survival curves plotted, one for the first part of the data based on DNA, on for the second part of the data based on DNA and the survival curve based on the standard clinical and histological data. Note that the survival curves for the data based on DNA have approximately the same shape.

One noticeable thing in figure 5.6 is that the different types of variables indeed have a different influence on the survival time of the patient. Based on the DNA data, it is less plausible that a patient survives more than 5 years, while based on the standard clinical variables that would be more plausible.

The influence of the DNA based data will become more clear when the method of LASSO is applied to the data. This can be done for the whole dataset, so the data does not have to be split up anymore. Before the LASSO is applied, cross-validation is used to determine the optimal value for the shrinkage parameter α. According to cross-validation this value is $\alpha = 4.601118$. For that shrinkage parameter, the algorithm states that there is only one variable taken in consideration, which can be found in table 5.4 together with the regression parameter.

Since there are so many variables involved a LASSO-plot is made where the regression parameter α does not reach the value of 0. Because the regression parameter does not reach 0, not all variables are shown, since some of them already shrunk to zero. This makes that the plot stays insightful. The

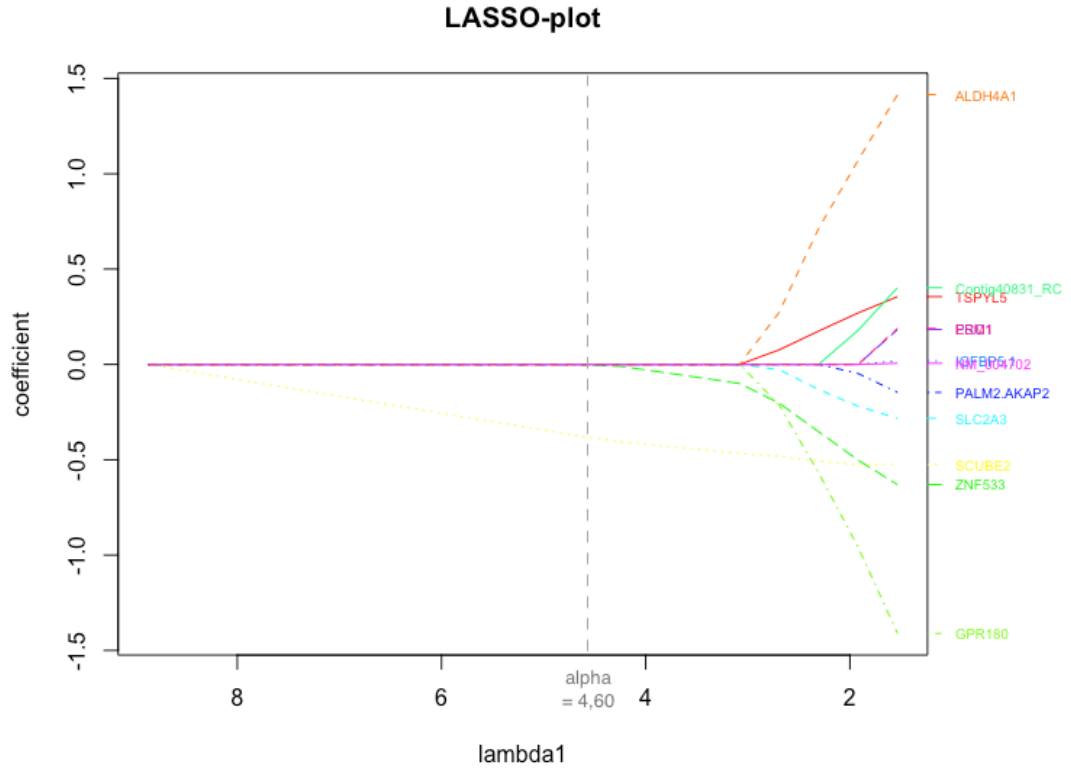

Figure 5.7: LASSO plot DNA variables

plot is shown in figure 5.7. Again, the grey dotted line shows the value for the regression parameter that was determined by cross-validation.

Note that the variable SCUBE2 is least sensitive to the LASSO and therefore SCUBE2 is the variable which has most impact on the survival time of the patients. Furthermore the variables ZNF533, ALD4A, GPR150, TSPYL5 and SLC2A3 seem to have a pretty big influence on the survival time of the patients as well.

5.2.3. All variables

Now that the two type of variables have been analyzed separately, one can have a look at the combined data. Applying the method of LASSO to this data, gives us the LASSO-plot in figure 5.8. Using cross-validation it is possible to determine the optimal value for the shrinkage parameter for the combined data. Using the 11 variables that have the most impact on the survival time according to the method of LASSO, a new survival plot can be plotted, see figure 5.9. Here, the black curve is based on the 11 variables which are also visible in the LASSO-plot in figure 5.8. Note that this curve is approximately the average of the curves based on the standard clinical variables and the curves based on the DNA data.

So based on the method of LASSO, the conclusion of M. van de Vijver [4] that the DNA variables are a more powerful predictor than the standard clinical and histologic variables is doubtable. Using cross-validation gives that the diagnosis age is the most powerful predictor, followed by the DNA variable SCUBE2. It is true that within the 11 variables with most impact 9 of them are genetic variables against two clinical variables.

The two variables that have biggest impact on the survival time are the diagnosis age and SCUBE2. To see how big the impact of both variables are the median of both variables are determined. For age the median is 43 years, then the data is split up into two parts: one for all patients younger than 43 and one for all patients older than 43. The survival curves for both cases can be seen in figure 5.10.

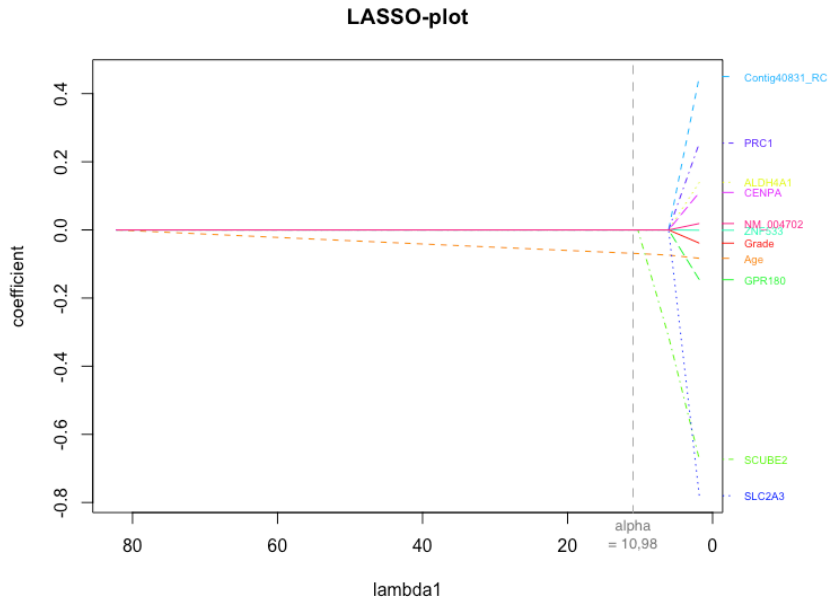

Figure 5.8: LASSO-plot all variables

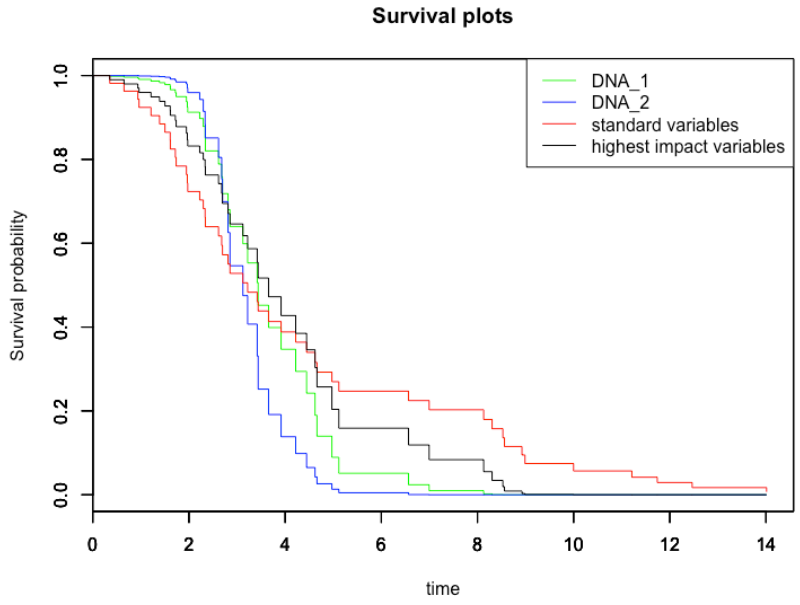

Figure 5.9: Survival plot all variables

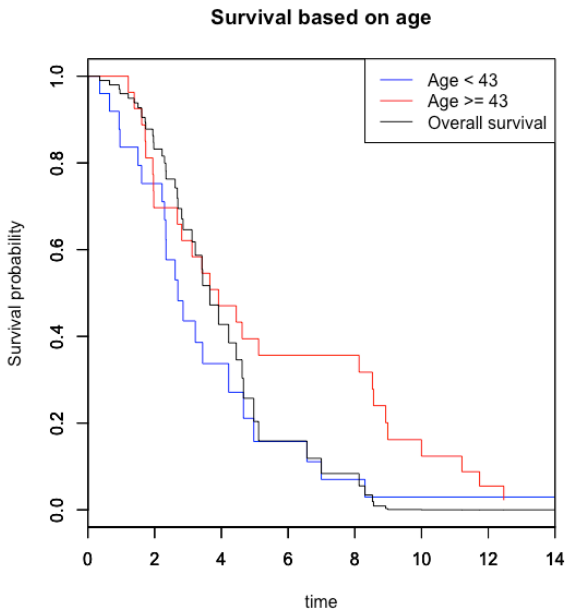

Figure 5.10: Survival plots based on different ages

Note that the younger you are the shorter your expected survival time is The same is done for the variable SCUBE2, which has a median of approximately -0.39, see figure 5.11.

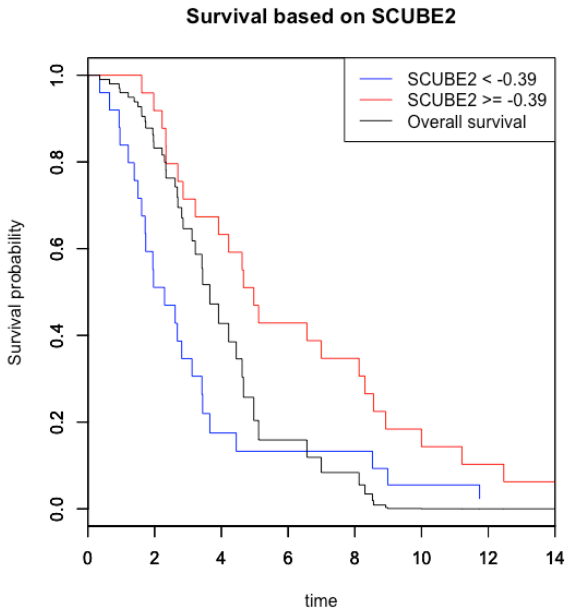

Figure 5.11: Survival plots based on SCUBE2

6

Summary

During this research, there was shown how the Cox proportional hazards model can be fitted to data. Suppose the survival data consists of n observed patients with p explanatory variables. In the Cox proportional hazards model, the hazard rate function of a patient is dependent of the p explanatory variables of a patient equals:

$$\lambda(t|X_i) = \lambda_0(t)e^{\beta^T \mathrm{X_i}}$$

where X_i contains all explanatory variables of the i^{th} patient, $\lambda_0(t)$ is the baseline hazard function and $\beta = (\beta_1, \dots, \beta_p)$ are the regression parameters of the model.

Maximum likelihood estimation is used to determine the regression parameters of the model. Therefore the likelihood and the so called partial likelihood of the Cox proportional hazards model were constructed. For the construction of the partial likelihood an alternative view on survival data where the event times are ordered is used. Then the partial likelihood is based on the conditional probabilities that patient i with explanatory variables X_i has event time t_i for every $i = 1, \dots, n$.

One complication that frequently occurs in survival analysis is the one of censored data. This occurs when the exact event time is not known, but it is only known that the event happened within a certain time interval. The constructed partial likelihood was slightly adapted to take censoring into account, which resulted in the minus-log-partial likelihood:

$$\ell_{pc}(\beta) = \sum_{i \in I} \left(\ln \left(\sum_{k=i}^{n} e^{\beta^T X_k} \right) - \beta^T X_i \right)$$

where I is the index set of the patients whose data is not censored.

Searching for the minimum of this minus-log-partial-likelihood function requires an efficient algorithm. In this research Newton's method is used to solve this optimization problem.

In applications of survival analysis, there are often so many explanatory variables involved that the model becomes difficult to interpret. Therefore Robert Tibshirani proposed to use the method of the Least Absolute Shrinkage and Selection Operator (LASSO) to apply variable selection and shrinkage to the Cox proportional hazards model [2]. Applying this method of LASSO to the Cox model, results in the following optimization problem:

$$\hat{\beta} = \min_{\beta} \ell_{pc}(\beta) + \alpha \sum_{j=1}^{p} |\beta_j|$$

where $\sum_{j=1}^{p} |\beta_j|$ is known as the L_1-norm.

Due to the lack of differentiability of the L_1-norm this optimization problem requires a more complex algorithm, so a combination of the Gradient Descent algorithm and Newton's method is used for solving, according to [7].

This Cox proportional hazards model with the Least Absolute Shrinkage and Selection Operator is then applied to data of breast cancer patients. This data contains some standard clinical and histological variables and variables based on DNA measurements. Medical research stated that the DNA based variables are a much more powerful predictor than the standard clinical variables [3]. According to this method, the most important variable is the diagnosis age, which is followed by the DNA based variable SCUBE2.

This method resulted in a subset of variables that are shrunken towards zero that made the model easier to interpret. Besides that, the model helped to determine what variables are a good predictor of survival time and what variabels are not. Therefore, there can be concluded that is useful to apply the method of LASSO to the Cox proportional hazards model.

7

Discussion

In this research several decisions were made, where one may make another decision. First of all, there was decided to assume that all explanatory variables are not dependent of time, which may often not be the case in reality. For example, one can imagine that certain explanatory variables can vary within the time that a patient is observed (which can be for several years). The Cox proportional hazards model is able to deal with time dependent variables, although this was not taken in consideration in this thesis.

One common problem in survival data is the one of censoring, which was explained in section 2.2.4. In this thesis, only right-censoring is taken in consideration, since survival data in medical applications are typically right-censored. Obviously, one can also construct the Cox proportional hazards model for left-censored or interval censored data.

When there is concluded that a regression model is needed in order to handle the large number of explanatory variables, there was chosen to use the Least Absolute Shrinkage and Selection Operator, also known as the method of LASSO. Although this method gave a good result, there are several other penalization methods available which can be applied to the Cox proportional hazards model. One example of another penalization model, which can also be applied in combination with the method of LASSO is ridge regression [7]. In order to improve the model, this regression model could be taken in consideration.

In the method of LASSO a shrinkage parameter α is involved, in order to determine the amount of shrinkage. In this research there was decided to use cross-validation to determine the optimal value for α. There are several other methods to determine the value α, which can be used instead of cross-validation.

In order to apply the maximum likelihood estimation on the different likelihood functions, Newton's method (in combination with the Gradient Descent algorithm) is used in order to make a good estimation for the regression parameters. Other algorithms could be used to find the minima of the minus-log-partial likelihood function, which would probably give slightly different results for the regression parameters. But since the penalty function is not differentiable everywhere, many popular algorithms for finding minima can not be used, which should be kept in mind.

A

Appendix A

A.1. Console

```
rm(list = ls())
setwd("~/Documents/BEP/Rcodes/Final/Likelihood_estimation_censored")

library(survival)
library(pracma)
library(matrixcalc)
library(penalized)

source("1.R")
source("lp_uncensored.R")
source("lp_censored.R")
source("fitting_cox.R")

#CREATE EXPLANATORY VARIABLES
n <- 300; #set the sample size
p <- 5; #set amount of different variables

x1 <- rnorm(n,1,0.6) #variable 1
x2 <- sample(0:1,n,replace = TRUE); #variable 2
x3 <- sample(1:2,n,replace = TRUE); #variable 3
x4 <- rnorm(n,0,1); #variable 4
x5 <- sample(1:3, n, replace = TRUE); #variable 5
df <- data.frame(x1, x2, x3, x4, x5); #make dataframe from explanatory variables
X <- data.matrix(df); #convert dataframe to matrix

#SETTINGS
beta <- 0.1 * sample(-10:10, p, replace = TRUE); #choose beta's randomly between -1 and 1
tol <- 1e-8; #set the tolerance size
beta0 <- numeric(p); #set intial value of beta

#Compute the baseline hazard, the hazard, the cumulative hazard and the survival function
lambda <- 1; #For simplicity, set lambda equal to 1

lambda_0 <- function(t) #the baseline hazard function
{
  lambda #suppose the baseline hazard is constant
}

hazard <-function(t, X, beta) #the hazard function
{
  lambda_0(t) * exp(X %*% beta)
}

cumhazard <- function(t, X, beta) #the cumulative hazard function
{
  exp(X %*% beta) * t #because lambda_0 is constant
}

S <- function(t, X, beta) #the survival function
{
  exp(-cumhazard(t, X, beta))
}

#COMPUTING THE EVENT TIMES
E <- rexp(n); #exponential distribution
F <- rexp(n); #exponential distribution
T <- E * (1/lambda) * exp(-1* (X %*% beta)); #compute event times according to cox model
```

```
C <- F * (1/lambda) * exp(-1* (X %*% beta)); #compute censoring times according to cox model
Y <- pmin(T,C); #take minimum of censoring and event times
delta <- as.numeric(T<=C); #set delta 1 if T<=C, and 0 otherwise
data <- data.frame(Y, delta, X); #create dataframe of all data
data_ordered <- data[order(data$Y),]; #order the data according to the event times
X_ordered <- data.matrix(data_ordered[1:p+2]);
data_cens <- data_ordered[data_ordered$delta ==1,];
X_cens <- data.matrix(data_cens[1:p+2]); #create matrix of ordered explanatory variables

#COMPUTATIONS UNPENALIZED OPTIMIZATION PROBLEM
beta_unpen_l <- fitting_cox(beta0,l, tol);                          #Use function fitting_cox to estimate
    beta using likelihood
beta_unpen_lp <- fitting_cox(beta0, lp_uncensored, tol);            #Use function fitting_cox to estimate
    beta using partial likelihood

fit_unpen <- coxph(Surv(T) ~ x1 + x2 + x3 + x4 + x5, data = data);  #use the coxph function to estimate
    beta
beta_unpen_coxph <- coef(fit_unpen)                                 #Coefficients of beta

#PLOT THE DIFFERENT ESTIMATIONS OF BETA
plot(beta, ylim = c(-1,1), main = "Different estimations on beta")
points(beta_unpen_l, col = "blue", ylim = c(-1,1))
points(beta_unpen_lp, col = "red", ylim = c(-1,1))
points(beta_unpen_lp, col = "red", ylim = c(-1,1))
legend("topright", c("chosen beta", "estimated by likelihood", "estimated by partial likelihood"), lty = c
    (1,1,1), col = c("black", "blue", "red"))

#COMPUTATIONS PENALIZED OPTIMIZATION PROBLEM
beta_unpen_lp_cens <- fitting_cox(beta0, lp_censored, tol);         #Use function fitting_cox to estimate
    beta using partial likelihood
fit_unpen_cens <- coxph(Surv(Y, delta == 1) ~ x1 + x2 + x3 + x4 + x5, data = data);  #use the coxph
    function to estimate beta
beta_unpen_coxph_cens <- coef(fit_unpen_cens);

#PLOT SURVIVAL FUNCTIONS
plot(survfit(fit_unpen), conf.int = FALSE, col = "blue", main = "Survival plot", ylab = "Survival
    probability", xlab = "time"); #plot the survival function
par(new = TRUE)
plot(survfit(fit_unpen_cens), conf.int = FALSE, col = "red");         #plot the survival function censored
legend("topright", c("uncensored", "censored"), lty = c(1,1), col = c("blue", "red"))


#USE CROSS_VALIDATION TO DETERMINE ALPHA
opt <- optL1(Surv(Y, delta), penalized =  X_cens, data = data_cens, lambda2 = 0);
alpha <- opt$lambda;

#USE PENALIZED PACKAGE TO MAKE LASSO-PLOT
pen_plot <- penalized(Surv(Y, delta), penalized = X_cens, data = data_cens, lambda1 = 0, lambda2 = 0,
    startbeta = beta0, steps = 20) #Make penalized opbject
plotpath(pen_plot, main = "LASSO-plot")
pen <- penalized(Surv(Y, delta), penalized = X_cens, data = data_cens, lambda1 = alpha, lambda2 = 0,
    startbeta = beta0) #Make penalized opbject
coef_pen <- coefficients(pen)
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_estimation_censored/console_censored.R

A.2. Minus-log-likelihood function

```
l <- function(beta){ # -loglikelihood function
  temp1 <- as.vector(X %*% beta);
  temp2 <- as.vector(T * exp(temp1));
  return(-n*log(lambda) - sum(temp1) + lambda * sum(temp2))
}
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_estimation_censored/l.R

A.3. Minus-log-partial-likelihood function

```
lp_uncensored <- function(beta){ # -log partial likelihood function
  temp1 <- as.vector(X_ordered %*% beta)
  som <- sum(temp1)
  e <- exp(temp1)
  temp2 <- numeric(n);
  for(i in 1:n){
    temp2[i] <- log(sum(e[i:n]))
  }
  return(sum(temp2) - som)
}
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_estimation_censored/lp_uncensored.R

A.4. Minus-log-partial-likelihood function for censoring

```
lp_censored <- function(beta){ # -log partial likelihood function
  temp1 <- as.vector(X_cens %*% beta)
  som <- sum(temp1)
  e <- exp(temp1)
  temp2 <- numeric(nrow(X_cens));
  for(i in 1:nrow(X_cens)){
    temp2[i] <- log(sum(e[i:nrow(X_cens)]))
  }
  return(sum(temp2) - som)
}
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_estimation_censored/lp_censored.R

A.5. Newton's method for fitting Cox(unpenalized)

```
fitting_cox <- function(beta0, l, tol) #this function includes the Newton-Raphson algorithm
{
  u <- grad(l, beta0); #compute the gradient of l(beta0)
  A <- hessian(l, beta0); #compute the hessian of l(beta0)
  beta_est <- as.vector(beta0 - inv(A) %*% u); #compute beta^{k+1}
  print(beta_est)
  if(max(abs(beta0-beta_est)) > tol) #check if sequence is converging
  {
    fitting_cox(beta_est, l, tol)
  }
  else
  {
    return(beta_est)
  }
}
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_estimation_censored/fitting_cox.R

B

Appendix B

DNA types	
TSPYL5	Contig63649_RC
DIAPH3	NUSAP1
AA555029_RC	ALDH4A1
QSCN6L1	FGF18
DIAPH3.1	Contig32125_RC
BBC3	DIAPH3.2
RP5.860F19.3	C16orf61
SCUBE2	EXT1
FLT1	GNAZ
OXCT1	MMP9
RUNDC1	Contig3525_RC
ECT2	GMPS
KNTC2	WISP1
CDC42BPA	SERF1A
AYTL2	GSTM3
GPR180	RAB6B
ZNF533	RTN4RL1
UCHL5	PECI
MTDH	Contig40831_RC
TGFB3	MELK
COL4A2	DTL
STK32B	DCK
FBXO31	GPR126
SLC2A3	PECI.1
ORC6L	RFC4
CDCA7	LOC643008
MS4A7	MCM6
AP2B1	C9orf30
IGFBP5	HRASLS
PITRM1	IGFBP5.1
NMU	PALM2.AKAP2
LGP2	PRC1
Contig20217_RC	CENPA
EGLN1	NM_004702
ESM1	C20orf46

Table B.1

C

Appendix C

C.1. Console

```
rm(list = ls())
setwd("~/Documents/BEP/Rcodes/Final/Likelihood_penalized")

library(survival)
library(pracma)
library(matrixcalc)
library(penalized)
library(gdata)

source("lp.R")
source("v_grad.R")
source("t_edge.R")
source("t_opt.R")
source("fitting_cox.R")
source("gradient_descent.R")
source("newton_raphson.R")

data("nki70")

#CREATE EXPLANATORY VARIABLES
n <- nrow(nki70);                                   #set the sample size
P <- ncol(nki70)-2;                                 #set amount of different variables

#SETTINGS
tol <- 1e-8; #set the tolerance size

#COMPUTING THE EVENT TIMES
data_ordered <- nki70[order(nki70$time),];                      #order the data according to the event times
data_cens <- data_ordered[data_ordered$event == 1, ];           #only use data if delta is 1
X_ordered <- data.matrix(data_ordered[3:75]);                   #create matrix of ordered explantory
    variables
X_cens <- data.matrix(data_cens[1:P+2]);                        #create matrix of explanatory variables
T_cens <- data_cens$time;                                       #create vector of event times

######COMPUTATIONS FOR STANDARD VARIABLES######
X_standard <- X_cens[,1:5];               #create matrix for only standard clinical variables
p <- ncol(X_standard);                    #set p equal to number of variables
N <- nrow(X_standard);                    #set N equal to number of patients with an event
beta0_standard <- numeric(p);             #set beta0_standard equal to a p-vector of zero's

#SOLVING THE UNPENALIZED OPTIMIZATION PROBLEM
beta_unpen_standard <- fitting_cox(beta0_standard, lp, tol)                         #solve the
    unpenalized optimization problem using Newton's method
fit_unpen_standard <- coxph(Surv(time, event == 1) ~ X_standard, data = data_cens);  #use the coxph
    function to solve the unpenalized optimazation problem
plot(survfit(fit_unpen_standard), conf.int = FALSE,  main = "Survival plot", ylab = "Survival probability"
    , xlab = "time"); #plot the survival function

#USE CROSSVALIDATION TO ESTIMATE ALPHA
opt_standard <- optL1(Surv(time, event), penalized =  X_standard, data = data_cens, lambda2 = 0);   #use
    penalized-package to make cross-validation object
alpha_standard <- opt_standard$lambda;                                                              #set
    alpha equal to optimal lambda1 of the cross-validation object

#SOLVING THE PENALIZED OPTIMIZATION PROBLEM
beta_gd_standard <- gradient_descent(beta0_standard, tol, alpha_standard)           #solve the penalized
    optimization problem using gradient descent
```

```
beta_NR_standard <- newton_raphson(beta0_standard, tol, alpha_standard)              #solve the penalized optimization problem using gradient descent & Newton's method
pen_standard <- penalized(Surv(time, event), penalized = X_standard, data = data_cens, lambda1 = alpha_standard, lambda2 = 0, startbeta = beta0_standard) #use penalized package to make penalize object
coef_pen_standard <- coefficients(pen)                                               #calculate the coefficients of the penalized objects
pen_plot_standard <- penalized(Surv(time, event), penalized = X_standard, data = data_cens, lambda1 = 0, lambda2 = 0, startbeta = beta0_standard, steps = 20) #use penalized package to make penalize object with steps
plotpath(pen_plot_standard, main = "Lasso")                                          #make LASSO-plot


######COMPUTATIONS FOR DNA VARIABLES######
X_dna <- X_cens[,6:75];                  #create matrix for only DNA variables
X_dna1 <- X_cens[,6:40];                 #create matrix for only the first 35 DNA variables
X_dna2 <- X_cens[,41:75];                #create matrix for only the second 35 DNA variables
p <- ncol(X_dna);                        #set p equal to number of variables
N <- nrow(X_dna);                        #set N equal to number of patients with an event
beta0_dna1 <- numeric(ncol(X_dna1));                 #set beta0_standard equal to a p-vector of zero's
beta0_dna2 <- numeric(ncol(X_dna2));
beta0_dna <- numeric(p);

#SOLVING THE UNPENALIZED OPTIMIZATION PROBLEM
beta_unpen_dna <- fitting_cox(beta0_dna, lp, tol)                                    #solve the unpenalized optimization problem using Newton's method --> does not converge
beta_unpen_dna1 <- fitting_cox(beta0_dna1, lp, tol)                                  #solve the unpenalized optimization problem using Newton's method
beta_unpen_dna2 <- fitting_cox(beta0_dna2, lp, tol)                                  #solve the unpenalized optimization problem using Newton's method

fit_unpen_dna1 <- coxph(Surv(time, event == 1) ~ X_dna1, data = data_cens);          #use the coxph function to solve the unpenalized optimazation problem
fit_unpen_dna2 <- coxph(Surv(time, event == 1) ~ X_dna2, data = data_cens);          #use the coxph function to solve the unpenalized optimazation problem

#USE CROSSVALIDATION TO ESTIMATE ALPHA
opt_dna <- optL1(Surv(time, event), penalized =  X_dna, data = data_cens, lambda2 = 0); #use penalized-package to make cross-validation object
alpha_dna <- opt_dna$lambda;                                                         #set alpha equal to optimal lambda1 of the cross-validation object

#SOLVING THE PENALIZED OPTIMIZATION PROBLEM
beta_gd_dna <- gradient_descent(beta0_dna, tol, alpha_dna)                           #solve the penalized optimization problem using gradient descent
beta_NR_dna <- newton_raphson(beta0_dna, tol, alpha_dna)                             #solve the penalized optimization problem using gradient descent & Newton's method
pen_dna <- penalized(Surv(time, event), penalized = X_dna, data = data_cens, lambda1 = alpha_dna, lambda2 = 0, startbeta = beta0_dna) #use penalized package to make penalize object
coef_pen_dna <- coefficients(pen_dna)                                                #calculate the coefficients of the penalized objects
pen_plot_dna <- penalized(Surv(time, event), penalized = X_dna, data = data_cens, lambda1 = alpha_dna/3, lambda2 = 0, startbeta = beta0_dna, steps = 20) #use penalized package to make penalize object with steps
plotpath(pen_plot_dna, main = "LASSO-plot")                                          #make LASSO-plot

######COMPUTATIONS FOR ALL VARIABLES######
p <- ncol(X_cens);
N <- nrow(X_cens);
beta0 <- numeric(ncol(X_cens));
opt <- optL1(Surv(time, event), penalized =  X_cens, data = data_cens, lambda2 = 0);      #use penalized-package to make cross-validation object
alpha <- opt$lambda;
pen <- penalized(Surv(time, event), penalized = X_cens, data = data_cens, lambda1 = alpha/6, lambda2 = 0, startbeta = beta0) #use penalized package to make penalize object with steps
coef_pen <- coefficients(pen);
beta_NR <- newton_raphson(beta0, tol, alpha)
pen_plot <- penalized(Surv(time, event), penalized = X_cens, data = data_cens, lambda1 = alpha/6, lambda2 = 0, startbeta = beta0, steps = 20) #use penalized package to make penalize object with steps
plotpath(pen_plot, main = "LASSO-plot")                                              #make LASSO-plot

fit <- coxph(Surv(time, event == 1) ~ Age + SCUBE2 + ALDH4A1 + GPR180 + ZNF533 + Contig40831_RC + SLC2A3 + Grade + PRC1 + CENPA + NM_004702, data = data_cens)

#PLOT THE SURVIVAL FUNCTIONS
plot(survfit(fit_unpen_dna1), col = "green", conf.int = FALSE,  main = "Survival plots", ylab = "Survival probability", xlab = "time"); #plot the survival function
par(new = TRUE)
plot(survfit(fit_unpen_dna2), col = "blue", conf.int = FALSE); #plot the survival function
par(new = TRUE)
plot(survfit(fit_unpen_standard), col = "red", conf.int = FALSE); #plot the survival function
par(new = TRUE)
plot(survfit(fit), conf.int = FALSE)
```

```
legend("topright", c("DNA_1", "DNA_2", "standard variables","highest impact variables"), lty = c(1,1,1,1),
      col=c("green", "blue","red","black")) # gives the legend lines the correct color and width)

######PLOTTING SURVIVAL CURVES FOR AGE AND SCUBE2######
median(data_cens$Age)
age1 <- data_cens[data_cens$Age < 43,];
age2 <- data_cens[data_cens$Age >= 43,];

fit1 <- coxph(Surv(time, event) ~ Age , data = age1);
fit2 <- coxph(Surv(time, event) ~ Age , data = age2);

plot(survfit(fit1), conf.int = FALSE, col = "blue", main = "Survival based on age", xlab = "time", ylab =
     "Survival probability", xlim = c(0,14))
par(new = TRUE);
plot(survfit(fit2), conf.int = FALSE, col = "red", xlim = c(0,14))
par(new = TRUE)
plot(survfit(fit), conf.int = FALSE, xlim = c(0,14))
legend("topright", c("Age < 43", "Age >= 43", "Overall survival"), lty = c(1,1,1), col=c("blue","red", "
     black")) # gives the legend lines the correct color and width)

median(data_cens$SCUBE2)
scube21 <- data_cens[data_cens$SCUBE2 < -0.39,];
scube22<- data_cens[data_cens$SCUBE2 >= -0.39,];

fit3 <- coxph(Surv(time, event) ~ SCUBE2 , data = scube21);
fit4 <- coxph(Surv(time, event) ~ SCUBE2 , data = scube22);

plot(survfit(fit3), conf.int = FALSE, col = "blue", main = "Survival based on SCUBE2", xlab = "time", ylab
      = "Survival probability", xlim = c(0,14))
par(new = TRUE);
plot(survfit(fit4), conf.int = FALSE, col = "red", xlim = c(0,14))
par(new = TRUE)
plot(survfit(fit), conf.int = FALSE, xlim = c(0,14))
legend("topright", c("SCUBE2 < -0.39", "SCUBE2 >= -0.39", "Overall survival"), lty = c(1,1,1), col=c("blue
     ","red", "black")) # gives the legend lines the correct color and width)
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_penalized/console_nki.R

C.2. Minus-log-partial-likelihood function

```
lp <- function(beta){ # -log partial likelihood function
  temp1 <- as.vector(X_cens %*% beta);
  som <- sum(temp1);
  e <- exp(temp1);
  temp2 <- numeric(N);
  for(i in 1:N){
    temp2[i] <- log(sum(e[i:N]))
  }
  return(sum(temp2) - som)
}
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_penalized/lp.R

C.3. Newton's method for fitting Cox (unpenalized)

```
fitting_cox <- function(beta0, l, tol) #this function includes the Newton-Raphson algorithm
{
  u <- grad(l, beta0); #compute the gradient of l(beta0)
  A <- hessian(l, beta0); #compute the hessian of l(beta0)
  beta_est <- as.vector(beta0 - inv(A) %*% u); #compute beta^{k+1}
  if( max(abs(beta0-beta_est)) > tol) #check if sequence is converging
  {
    fitting_cox(beta_est, l, tol)
  }
  else
  {
    return(beta_est)
  }
}
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_penalized/fitting_cox.R

C.4. Function for computing t_{edge}

```r
t_edge <- function(beta, alpha){  #Function for calculating t_edge
  temp <- numeric(p);
  v <- v_grad(beta,alpha);
  if(all(beta == 0) == TRUE){
    return(0)
  } else{
    for(j in 1:p){
      if(sign(beta[j]) == sign(v[j]) & sign(beta[j]) != 0){
        temp[j] <- beta[j]/(v[j])
      } else
        temp[j] <- 0;
    }
    if(all(temp == 0)){
      return(0)
    } else{
      return( min(temp[temp>0])) #return the mimimum of temp such that temp>0
    }
  }
}
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_penalized/t_edge.R

C.5. Function for computing t_{opt}

```r
t_opt <- function(beta, alpha){
  A <- hessian(lp, beta);
  v <- v_grad(beta, alpha);
  temp1 <- (v %*% v);
  temp2 <- (v %*% A) %*% v;
  return(as.double(temp1/temp2))
}
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_penalized/t_opt.R

C.6. Function for computing the gradient

```r
v_grad <- function(beta, alpha){
  v <- numeric(p);
  u <- grad(lp, beta)
  for(j in 1:p){
    if(beta[j] != 0){
      v[j] <- u[j] + sign(beta[j])*alpha
    } else if(beta[j] == 0 & abs(u[j]) >= alpha) {
      v[j] <- u[j] + alpha * sign(u[j])
    } else
      v[j] <- 0
  }
  return(v)
}
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_penalized/v_grad.R

C.7. Gradient Descent algorithm for applying LASSO

```r
gradient_descent <- function(beta, tol, alpha) #this function includes the Newton-Raphson algorithm
{
  t_edge <- t_edge(beta, alpha);
  t_opt <- t_opt(beta, alpha);
  t <- min(t_edge[t_edge>0], t_opt);
  v <- as.vector(v_grad(beta, alpha));
  beta_gd <- as.vector(beta - t * v);
  print(beta_gd);
  if(max(abs(beta-beta_gd)) > tol)
  {
    gradient_descent(beta_gd, tol, alpha)
  }
  else
  {
    print(beta_gd)
  }
}
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_penalized/gradient_descent.R

C.8. Combined Gradient Descent and Newton's method for applying LASSO

```
newton_raphson <- function(beta, tol, alpha)
{
  t_edge <- t_edge(beta, alpha);
  t_opt <- t_opt(beta, alpha);
  t <- min(t_edge[t_edge>0], t_opt);
  v <- v_grad(beta, alpha);

  if(t == t_edge){
    beta_NR <- as.vector(beta - t_edge * v);
  } else if(all(beta == 0) == FALSE) {
    A <- hessian(lp, beta);
    v <- v_grad(beta, alpha);

    if(any(beta == 0) == TRUE){
      index <- numeric(p);
      for(j in 1:p){
        if(beta[j] == 0){
          index[j] <- j
        }
      }
      index <- index[index != 0];
      beta_nonzero <- beta[beta !=0];
      A_nonzero <- as.matrix(A[-index, -index]);
      v_nonzero <- v[-index];

      beta_NR <- as.vector(beta_nonzero - inv(A_nonzero) %*% v_nonzero);
      for(j in 1:length(index)){
        beta_NR <- append(beta_NR, 0, after=(index[j]-1));
      }
    } else{
      beta_NR <- as.vector(beta - inv(A) %*% v);
    }

    if(all(sign(beta) == sign(beta_NR)) == TRUE){
      beta_NR <- beta_NR
    } else{
      beta_NR <- as.vector(beta - t_opt %*% v);
    }
  } else{
    beta_NR <- as.vector(beta - t_opt %*% v);
  }
  if(max(abs(beta_NR - beta)) > tol){
    newton_raphson(beta_NR, tol, alpha)
  } else{
    return(beta_NR[beta_NR !=0])
  }
}
```

/Users/ruthkoole/Documents/BEP/Rcodes/Final/Likelihood_penalized/newton_raphson.R

Bibliography

[1] M. Nikulin et al, *The Cox Model and Its Applications* (2016).

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

[3] L. J. van 't Veer et al., *Gene expression profiling predicts clinical outcome of breast cancer,* Nature **415**, 530 (2002).

[4] M. J. van de Vijver et al., *A gene-expression signature as a predictor of survival in breast cancer,* The New England Journal of Medicine **347** (2002), DOI: 10.1056/NEJMoa021967.

[5] G. Jongbloed, *Modeling and analysis of time-to-event data* (2015) pp. 64,65.

[6] R. Tibshirani, *Regression shrinkage and selection via the lasso,* Journal of the Royal Statistical Society, Series B (Methodological) **58**, 267 (1996).

[7] J. J. Goeman, *L1 penalized estimation of the cox proportional hazards model,* Biometrical Journal **52**, 70 (2010).

[8] T. Hastie et al, *The Elements of Statistical Learning* (2009).