MSc thesis MKE, Bioinformatics track

Efficient approximate leave-one-out cross-validation for ridge and lasso

Rosa Meijer

Delft University of Technology

Supervisors

Dr. J.J. Goeman
Prof.dr.ir M.J.T. Reinders

Other members of the thesis committee

Dr. S. le Cessie
Dr. M. Loog
Ir. J. de Ridder

October, 2010 Delft

Preface

When I started this project, all that was well defined was the starting point but the exact direction that should be taken from there was still unclear. After familiarizing myself with an already existing method that can be used to speed up leave-one-out cross-validation for a linear model, the first step was to add a ridge penalty to this linear model and find out if the method would still work. It turned out that the method still worked fine and resulted in exact answers in the linear ridge model.

A rather natural way to proceed from here was to apply a slightly altered version of the method to a different model, namely the logistic (ridge) model. In this way an efficient procedure to approximate leave-one-out cross-validation results was obtained. Although the same procedure was already suggested by others, little attention had been paid to examining its usefulness in practical situations.

After reaching this point, some choices had to be made. Since I wrote my literature survey on survival analysis, an obvious choice would be to extend the approximation method to work for survival models. However, there were other possibilities. The first one was to change the penalty parameter from the ridge into the lasso penalty and try to find out if the method could be adjusted in order to work with this new penalty. The second option was to turn to k-fold cross-validation instead of leave-one-out cross-validation and try to find out if a similar approximation method would work there and would possibly even be exact in the linear case.

Looking back, I can say that I in fact explored all paths mentioned. However, not all my findings made it into the final article. Fortunately, this report consists of two parts: a work document (the first part) and an article. In the final article, the focus lies on an approximation method that works for survival models with either a ridge or a lasso penalty. However, the chapter on k-fold cross-validation that can be found in the work document is also worth reading. Additionally, in the first few chapters of the work document I tried to find optimal values of the penalty parameters by means of derivatives, so a more theoretical approach. Since the methods failed to produce reliable answers (or did not produce answers at all), I did not refer to them in the final article but the interested reader might still want to look into them briefly. Furthermore, in the chapter on lasso regression, one can find an extension of the method that is used in the article. Since I am still doubting its usefulness, it is not included in the final article.

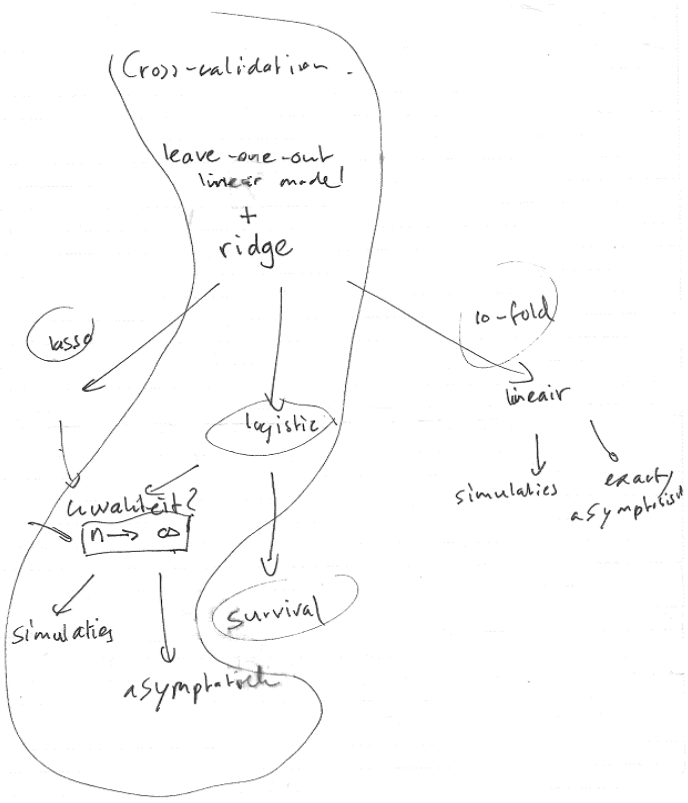
Although nothing is ever finished, this is as close as it gets (for now).

Acknowledgements

First of all, I would like to thank my supervisor Jelle Goeman. Because of his enthusiasm, patience, at times his perfectionism and his ability to explain even theoretical subjects by means of understandable examples I really enjoyed working with him. Moreover, he made me question my scientific abilities less by asking me to present our work at the ISCB conference in Montpellier last summer and even offering me a PhD position. Since I can not think of anyone I would rather work with I accepted this offer and I'm really looking forward to the next four years in which we will be working together. The next person I would like to thank is Marcel Reinders. First of all for being one of the inventors of the bioinformatics master. The combination of statistics, informatics and biology was exactly the mix I was looking for. Secondly, he showed me that quality is not the only important aspect of doing research. Although we never fully agreed on this point, for me it was important to finally understand that seeking for perfection will not always lead to the desired result. I would like to thank the remaining members of my thesis committee in advance for making time to read all this. Of course, there are some friends I would like to thank. Thijmen, Marleen, Lotte and Johannes, thanks for telling me that if I couldn't do it, no one could. I know it is not true, but it was definitely nice to know that you believed in me. Willian, thanks for pushing me at times when I needed it. Oksana and Alina, thanks for letting me know that making mistakes will always be part of life and last but not least, I would like to thank my parents and sisters who still think I will be teaching at the university someday. I really hope so.

Work Document

Essentially based on the following picture:



Contents

1 Introduction

In many areas, statistical models are used. Regardless of the precise model, the final step in the model building sequence will always be a model validation procedure. In this validation step, one tries to find out whether the created prediction model really predicts well, not only on the training data set, but also on an independent test set. Using an independent test set is very important, since the results obtained from training and validating the model on the same data set will be too optimistic.

When independent validation data is absent (which is usually the case) and an estimate of the predictive accuracy is needed, resampling the original data to create an independent test set is a commonly used approach. There are many ways in which this can be done, but one of the most frequently used methods is called **cross-validation**. The most important cross-validation methods are *leave-one-out cross-validation* and *k-fold cross-valdiation*.
In leave-one-out cross-validation (LOOCV), each observation is chosen once to be the test set. The model is fitted n times and is validated every time on one single test object. The results can then be averaged. An advantage of this method is that the n training sets contain only 1 element less than the potential training set (i.e. the full data set) and therefore the obtained predictive performance will closely resemble the real performance. A disadvantage of this approach is that it can be very time-consuming.
A more general version of LOOCV is **k-fold cross-validation**. The data is divided in k (nearly) equal parts. Subsequently the model is created by using $k-1$ parts and validated using the only part of the data that was not involved in the model building. For small values of k, a lot of potential training data is not used in the model building step and the obtained predictive performance will for that reason probably be worse than the real performance. When k gets larger, this problem diminishes.
Another problem of k-fold cross-validation is that its outcomes are not directly reproducible. Where leave-one-out cross-validation is purely deterministic, k-fold cross-validation depends on the actual partition. To reduce this problem, multiple paritions can be used and the results can again be averaged, but in this way the method gets more time-consuming and in order to average over all possible partitions, one will need a tremendous amount of calculations.

Although leave-one-out cross-validation can be very time consuming, the method usually works fine as long as all there is to estimate is a vector of regression coefficients. Unfortunatly, when the number of covariates comes close to or even exceeds the number of samples, a different problem arises since the regression model should be adapted to deal with this new situation. In regression methods suitable for high dimensional covariate spaces (ridge or lasso regression for example), not only the regression coefficient has to be estimated, but also the value of some tuning parameter. The optimal value of this penalty parameter has to be established by cross-validation, which can take much time (many dif-

ferent values of the tuning parameter has to be examined in order to find the one most suitable for the final model) and in order to maintain an independent test set even double LOOCV should be used.

The objective of this study is to come up with a method that produces similar output to ordinary cross-validation, but is less time consuming. Estimating the optimal values of penalty parameters in ridge and lasso regression will become less time-consuming and carrying out (an approximated version of) double LOOCV will become practically feasible in this way. We derive a method in which approximations of the real maximum likelihood leave-one-out regression coefficients are used to find optimal values of the tuning parameters in ridge and lasso regression, without actually having to refit the model. In linear ridge regression, our approximation is exact. In more complex models (generalized linear models and Cox' proportional hazards model) the approximations are based on a first order Taylor approximation of the gradient of the log-likelihood around the maximum penalized likelihood estimator of the full model. When the number of observations increases, the real cross-validated estimates will be closer to the estimate of the full model and the error term in the Taylor approximation will diminish. Therefore, the approximation method is especially suitable for large datasets, for which ordinary cross-validation takes much time.

In the upcoming chapters the approximation method will be described in detail. First it will be shown that the method gives exact answers in linear ridge regression and then the method will be extended to generalized linear models and Cox' proportional hazards model. The ridge and lasso penalties will be discussed seperately.

To compare the results of this method to the results that would have been obtained by using ordinary LOOCV, both methods are applied to several microarray datasets. The usefulness of the method is based on two criteria:
- are the actual values of the approximated leave-one-out estimates on average close to the real values, and
- can the optimal tuning parameter be estimated accurately based on the approximations?

In addiation to the leave-one-out cross-validation approximation procedure, a k-fold cross-validation approximation procedure will be discussed in chapter 6.

2 Linear regression

Since we will need the Sherman-Morrison-Woodbury theorem (as given in [9]) in almost all future derivations, let's start this chapter with writing down this theorem and its proof:

Theorem 2.0.1. *Sherman-Morrison-Woodbury theorem*
Let $\boldsymbol{A}$ be a nonsingular $p \times p$ matrix, and $\boldsymbol{u}$ and $\boldsymbol{v}$ be two p-dimensional column vectors. Then

$$\left(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T\right)^{-1} = \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}}.$$

To proof this theorem, one must show that $\left(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T\right) Y = Y \left(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T\right) = \boldsymbol{I}$, where Y equals the right-hand side of the Sherman-Morrison-Woodbury equation. In the proof, we use the fact that $\boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}$ is just a scalar.

Proof.

$$\begin{aligned}
\left(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T\right)\left(\boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}}\right) &= \boldsymbol{A}\boldsymbol{A}^{-1} + \boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1} - \frac{\boldsymbol{A}\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1} + \boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}} \\
&= \boldsymbol{I} + \boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1} - \frac{\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1} + \boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}} \\
&= \boldsymbol{I} + \boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1} - \frac{\left(1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}\right)\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}} \\
&= \boldsymbol{I} + \boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1} - \boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1} \\
&= \boldsymbol{I}
\end{aligned}$$

and

$$\begin{aligned}
\left(\boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}}\right)\left(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T\right) &= \boldsymbol{A}^{-1}\boldsymbol{A} + \boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T - \frac{\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{A} + \boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}} \\
&= \boldsymbol{I} + \boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T - \frac{\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T + \boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}} \\
&= \boldsymbol{I} + \boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T - \frac{\left(1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}\right)\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}} \\
&= \boldsymbol{I} + \boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T - \boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T \\
&= \boldsymbol{I}
\end{aligned}$$

□

A general form of a linear model is:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\boldsymbol{y}$ (the response variable) and $\boldsymbol{\epsilon}$ are n-dimensional random vectors, $\boldsymbol{X}$ is an $n \times p$ regression-matrix of known constants (often also called the design-matrix) and $\boldsymbol{\beta}$ is a p-dimensional parameter vector.
The question is how to find a good estimate for $\boldsymbol{\beta}$ and after we found this estimate, the next question is how good this estimate really is.

The optimal $\boldsymbol{\beta}$ should approximate the values of the response variable as good as possible. In other words, the residuals

$$e_i(\boldsymbol{\beta}) = y_i - (\boldsymbol{X}\boldsymbol{\beta})_i$$

should be as small as possible.
Therefore, we look for the $\boldsymbol{\beta}$ that minimizes the *residual sum of squares* (RSS), given by

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} e_i(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - (\boldsymbol{X}\boldsymbol{\beta})_i)^2 = ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||^2.$$

To minimize this equation, we differentiate $RSS(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$ and find the value of $\boldsymbol{\beta}$ for which this equation equals 0, which gives us

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{1}$$

This $\hat{\boldsymbol{\beta}}$ is the so called "least squares estimator", but it can easily be shown that this estimator equals the maximum likelihood estimator, when the errors $\boldsymbol{\epsilon}$ have a multivariate normal distribution with mean 0 and variance matrix $\sigma^2\boldsymbol{I}$.

The likelihood for the linear model will in this case be given by:

$$L(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y}) = \frac{1}{(2\pi)^{n/2}} \frac{1}{\sigma^n} \exp\left(-\frac{1}{2}\frac{(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})}{\sigma^2}\right).$$

Maximizing this equation will give the maximum likelihood estimator. Since the log-likelihood is easier to maximize and will result in the same estimator, we work with

$$l(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y}) = -\frac{n}{2}\ln(2\pi) - n\ln(\sigma) - \frac{1}{2}\frac{(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})}{\sigma^2}.$$

Differentiating this expression with respect to $\boldsymbol{\beta}$ will give the maximum likelihood estimator (MLE) for $\boldsymbol{\beta}$:

$$\frac{\partial l(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y})}{\partial \boldsymbol{\beta}} = -\frac{1}{2}\frac{\left(-2\boldsymbol{X}^T\boldsymbol{y} + 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}\right)}{\sigma^2} = 0 \quad \Rightarrow \quad \hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}.$$

It is clear that the MLE and the least squares estimator are completely identical.

One assumption still has to be made. In order to make sure that the inverse of $\boldsymbol{X}^T\boldsymbol{X}$ exists, we assume that the columns of the design matrix $\boldsymbol{X}$ are linearly independent and the number of covariates p is smaller or equal to the number of individuals n, so X has rank p. We know that the following holds:

$$rank(\boldsymbol{X}^T\boldsymbol{X}) = rank(\boldsymbol{X}).$$

Proof. Since $\boldsymbol{X}^T\boldsymbol{X}$ is a $p \times p$ matrix and therefore has the same number of columns as $\boldsymbol{X}$, we know:

$$rank(\boldsymbol{X}) + nullity(\boldsymbol{X}) = p = rank(\boldsymbol{X}^T\boldsymbol{X}) + nullity(\boldsymbol{X}^T\boldsymbol{X}),$$

so it is enough to show that the null space of $\boldsymbol{X}^T\boldsymbol{X}$ equals the null space of $\boldsymbol{X}$. Let $\boldsymbol{b}$ be in the null space of $\boldsymbol{X}$, so $\boldsymbol{X}\boldsymbol{b} = \boldsymbol{0}$. Then we know that $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{b} = \boldsymbol{X}^T\boldsymbol{0} = \boldsymbol{0}$, so $\boldsymbol{b}$ is in the null space of $\boldsymbol{X}^T\boldsymbol{X}$. In addition, for all vectors $\boldsymbol{b}$ for which $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{b}$=0, we also have $\boldsymbol{b}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{b} = \boldsymbol{0} = ||\boldsymbol{X}\boldsymbol{b}||^2$, thus $\boldsymbol{b}$ is in the null space of $\boldsymbol{X}$. □

Since $rank(\boldsymbol{X}) = p$ we know that $rank(\boldsymbol{X}^T\boldsymbol{X}) = p$ which implies that $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ exists. Note that as soon as p gets larger than n, this does no longer hold. (That is the moment where ridge regression can become useful.)

When we found the value for $\hat{\boldsymbol{\beta}}$ based on the training data set, we would like to test our model on an independent test set, but unfortunately, independent test data is usually absent. To get an indication of the true performance, we can use leave-one-out cross-validation and train the model n times, using only $n-1$ observations and testing this model on the one observation that was left out. The corresponding cross-validated residual sum of squares is given by:

$$RSS_{cv} = \sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^T\hat{\boldsymbol{\beta}}_{-i})^2, \tag{2}$$

where $\hat{\boldsymbol{\beta}}_{-i}$ is given by (1), only this time $\boldsymbol{X}$ is a $(n-1) \times p$ matrix (from now on denoted by $\boldsymbol{X}_{-i}$) and the i^{th} element of $\boldsymbol{y}$ is left out ($\boldsymbol{y}_{-i}$).
To find the value of RSS_{cv}, n inverses have to be calculated, namely all $(\boldsymbol{X}_{-i}^T\boldsymbol{X}_{-i})^{-1}$'s, where i ranges from 1 to n. To save time, we can use the fact that all those inverses are very similar, which can be seen from the Sherman-Morrison-Woodbury theorem.

Let $\boldsymbol{A}$ (in theorem 2.0.1) equal $\boldsymbol{X}^T\boldsymbol{X}$, let $\boldsymbol{u}^T$ be the ith row of the $\boldsymbol{X}$-matrix (denoted by $\boldsymbol{x}_i$) and take $\boldsymbol{v} = -\boldsymbol{u}$. The equation now becomes

$$(\boldsymbol{X}'_{-i}\boldsymbol{X}_{-i})^{-1} = (\boldsymbol{X}'\boldsymbol{X} - \boldsymbol{x}_i\boldsymbol{x}_i')^{-1} = (\boldsymbol{X}'\boldsymbol{X})^{-1} + \frac{(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}_i\boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{X})^{-1}}{1 - \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}_i}, \quad (3)$$

where $\boldsymbol{X}'_{-i}\boldsymbol{X}_{-i}$ is the $\boldsymbol{X}'\boldsymbol{X}$ matrix with the ith individual left out. To see that $(\boldsymbol{X}'\boldsymbol{X})_{(-i)}$ is indeed given by $(\boldsymbol{X}'\boldsymbol{X} - \boldsymbol{x}_i\boldsymbol{x}_i')$ we should use the following equality:

$$\boldsymbol{X}'\boldsymbol{X} = \sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{x}_i'.$$

Multiplying equation (3) by $\boldsymbol{X}'\boldsymbol{y} - \boldsymbol{x}_i y_i$ gives:

$$\begin{aligned}
(X'_{-i}X_{-i})^{-1}(X'y - \boldsymbol{x}_i y_i) &= (X'X)^{-1}(X'y - \boldsymbol{x}_i y_i) + \frac{(X'X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i'(X'X)^{-1}(X'y - \boldsymbol{x}_i y_i)}{1 - \boldsymbol{x}_i'(X'X)^{-1}\boldsymbol{x}_i} \\
\hat{\boldsymbol{\beta}}_{-i} &= \hat{\boldsymbol{\beta}} - (X'X)^{-1}\boldsymbol{x}_i y_i + \frac{(X'X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i'(X'X)^{-1}(X'y - \boldsymbol{x}_i y_i)}{1 - \boldsymbol{x}_i'(X'X)^{-1}\boldsymbol{x}_i}
\end{aligned}$$

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &- (X'X)^{-1}\boldsymbol{x}_i y_i + \frac{(X'X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i'(X'X)^{-1}(X'y - \boldsymbol{x}_i y_i)}{1 - \boldsymbol{x}_i'(X'X)^{-1}\boldsymbol{x}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(X'X)^{-1}\boldsymbol{x}_i y_i(1 - \boldsymbol{x}_i'(X'X)^{-1}\boldsymbol{x}_i) - (X'X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i'(X'X)^{-1}(X'y - \boldsymbol{x}_i y_i)}{1 - \boldsymbol{x}_i'(X'X)^{-1}\boldsymbol{x}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(X'X)^{-1}\boldsymbol{x}_i y_i - (X'X)^{-1}\boldsymbol{x}_i y_i\boldsymbol{x}_i'(X'X)^{-1}\boldsymbol{x}_i - (X'X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i'(X'X)^{-1}(X'y - \boldsymbol{x}_i y_i)}{1 - \boldsymbol{x}_i'(X'X)^{-1}\boldsymbol{x}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(X'X)^{-1}\boldsymbol{x}_i y_i - (X'X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i'(X'X)^{-1}(\boldsymbol{x}_i y_i + X'y - \boldsymbol{x}_i y_i)}{1 - \boldsymbol{x}_i'(X'X)^{-1}\boldsymbol{x}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(X'X)^{-1}\boldsymbol{x}_i y_i - (X'X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i'(X'X)^{-1}(X'y)}{1 - \boldsymbol{x}_i'(X'X)^{-1}\boldsymbol{x}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(X'X)^{-1}\boldsymbol{x}_i y_i - (X'X)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i'\hat{\boldsymbol{\beta}}}{1 - \boldsymbol{x}_i'(X'X)^{-1}\boldsymbol{x}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(X'X)^{-1}\boldsymbol{x}_i(y_i - \boldsymbol{x}_i'\hat{\boldsymbol{\beta}})}{1 - \boldsymbol{x}_i'(X'X)^{-1}\boldsymbol{x}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(X'X)^{-1}\boldsymbol{x}_i e_i}{1 - h_{ii}}
\end{aligned}$$

As a result, we have now:

$$\hat{\boldsymbol{\beta}}_{-i} = \hat{\boldsymbol{\beta}} - \frac{(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}_i e_i}{1 - h_{ii}}, \tag{4}$$

where $e_i = y_i - \boldsymbol{x}_i'\hat{\boldsymbol{\beta}}$ and h_{ii} is the i^{th} diagonal element of the hat-matrix $\boldsymbol{H}$, given by $\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$. This is an already know result and can be found for example in [2].

This expression can now be substituted in the cross-validated residual sum of squares, as given in equation (2):

$$\sum_{i=1}^{n}\left(y_i - \boldsymbol{x}_i'\hat{\boldsymbol{\beta}}_{-i}\right)^2 = \sum_{i=1}^{n}\left(y_i - \boldsymbol{x}_i'\hat{\boldsymbol{\beta}} + \frac{\boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}_i e_i}{1 - h_{ii}}\right)^2 = \sum_{i=1}^{n}\left(e_i + \frac{h_{ii}e_i}{1 - h_{ii}}\right)^2,$$

and eventually we get:

$$\sum_{i=1}^{n}\left(y_i - \boldsymbol{x}_i'\hat{\boldsymbol{\beta}}_{-i}\right)^2 = \sum_{i=1}^{n}\left(\frac{e_i}{1 - h_{ii}}\right)^2. \tag{5}$$

The next question is if there's an easy way to compute this sum by means of matrix multiplication.

Let's denote the vector with predicted y-values, based on the cross-validated $\hat{\boldsymbol{\beta}}$ by $\hat{\boldsymbol{y}}_{cv}$. Then, by using equation (5), we can write

$$\boldsymbol{y} - \hat{\boldsymbol{y}}_{cv} = (diag(I_n - H))^{-1}(\boldsymbol{y} - \hat{\boldsymbol{y}}).$$

If we use the fact that $\hat{\boldsymbol{y}}$ equals $H\boldsymbol{y}$, we get

$$\boldsymbol{y} - \hat{\boldsymbol{y}}_{cv} = (diag(I_n - H))^{-1}(I_n - H)\boldsymbol{y}.$$

Eventually, we want to have an equation for the residual sum of squares, which means that we have to take the inner product of the just derived vector with itself:

$$\begin{aligned} RSS_{cv} &= ((diag(I_n - H))^{-1}(I_n - H)\boldsymbol{y})'((diag(I_n - H))^{-1}(I_n - H)\boldsymbol{y}) \\ &= \boldsymbol{y}'(I_n - H)(diag(I_n - H))^{-2}(I_n - H)\boldsymbol{y} \end{aligned} \tag{6}$$

The last equality hold because $(diag(I_n - H))^{-1}$ and $(I_n - H)$ are symmetric matrices. (The first is a diagonal matrix and $(I_n - H)$ is symmetric because H is symmetric [1].)

Now we have one formula with which we can immediately find the cross-validated residual sum of squares, once we have our data set (and thereby the hat-matrix), without having to fit the model over and over again.

[1] $(\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}')' = (\boldsymbol{X}')'((\boldsymbol{X}'\boldsymbol{X})^{-1})'\boldsymbol{X}' = \boldsymbol{X}((\boldsymbol{X}'\boldsymbol{X})')^{-1}\boldsymbol{X}' = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$

A problem occurs when the denominator in equation (4) becomes 0, so when (one of) the diagonal elements of the hat-matrix equal 1. One situation where this occurs is when the design-matrix $\boldsymbol{X}$ is an invertible square matrix, so when the number of covariates equals the number of individuals. In that case, the hat-matrix $\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$ is exactly the identity matrix. This can be seen by proving that $(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$ equals $\boldsymbol{X}^{-1}$. Since $\boldsymbol{X}$ is invertible, so is $\boldsymbol{X}'$ and thus $(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}' = \boldsymbol{X}^{-1}(\boldsymbol{X}')^{-1}\boldsymbol{X}' = \boldsymbol{X}^{-1}$. The geometric interpretation is that the predicted values for $\boldsymbol{y}$ are exactly equal to the real values. This makes sense because the original equation $\boldsymbol{y} = \boldsymbol{X}\beta$ can be solved exactly since $\boldsymbol{X}$ is invertible.

It is important to note that the expression for $\hat{\boldsymbol{\beta}}_{-i}$ is based on the fact that there exists a closed formula for $\hat{\boldsymbol{\beta}}$. In other models (for example logistic or survival models), $\hat{\boldsymbol{\beta}}$ can only be found in an iterative way and the derivation of an approximated version of $\hat{\boldsymbol{\beta}}_{-i}$ will be different from the one described in this first section and will not result in an exact solution. The approximation will be based on a taylor expansion (described in detail in section 5) and will look like

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} - (l''_{(-i)}(\hat{\boldsymbol{\beta}}))^{-1} l'_{(-i)}(\hat{\boldsymbol{\beta}}), \tag{7}$$

with $l_{(-i)}(\boldsymbol{\beta})$ the log-likelihood based on $n-1$ observations. The error we make in this approximation is:

$$\frac{l'''_{(-i)}(\boldsymbol{\beta}^*)}{2!}(\hat{\boldsymbol{\beta}}_{(-i)} - \hat{\boldsymbol{\beta}})^2,$$

for some $\boldsymbol{\beta}^* \in (\hat{\boldsymbol{\beta}}_{(-i)}, \hat{\boldsymbol{\beta}})$.

It can easily be seen that this approximation method will in the linear model result in the same approximation for $\hat{\boldsymbol{\beta}}_{(-i)}$ as the direct formula and will again be exact. The exactness results from the fact that the log-likelihood is a quadratic function in $\boldsymbol{\beta}$ and therefore, its third derivative equals 0.
If we substitute the expressions for the first and second derivative of the log-likelihood in equation (7), we get:

$$\begin{aligned}
\hat{\boldsymbol{\beta}}_{(-i)} &= \hat{\boldsymbol{\beta}} + (\frac{1}{\sigma^2}\boldsymbol{X}'_{-i}\boldsymbol{X}_{-i})^{-1}\frac{\boldsymbol{X}'_{-i}\boldsymbol{y}_{-i} + \boldsymbol{X}'_{-i}\boldsymbol{X}_{-i}\hat{\boldsymbol{\beta}}}{\sigma^2} \\
&= \hat{\boldsymbol{\beta}} + (\boldsymbol{X}'_{-i}\boldsymbol{X}_{-i})^{-1}\boldsymbol{X}'_{-i}\boldsymbol{y}_{-i} - (\boldsymbol{X}'_{-i}\boldsymbol{X}_{-i})^{-1}\boldsymbol{X}'_{-i}\boldsymbol{X}_{-i}\hat{\boldsymbol{\beta}} \\
&= \hat{\boldsymbol{\beta}} + (\boldsymbol{X}'_{-i}\boldsymbol{X}_{-i})^{-1}\boldsymbol{X}'_{-i}\boldsymbol{y}_{-i} - \hat{\boldsymbol{\beta}} \\
&= (\boldsymbol{X}'_{-i}\boldsymbol{X}_{-i})^{-1}\boldsymbol{X}'_{-i}\boldsymbol{y}_{-i}.
\end{aligned}$$

Applying the Sherman-Morrison-Woodbury theorem would again result in (4).

3 Linear ridge regression

The important difference is that the hat-matrix is not longer $\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$ but is now given by $\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_p)\boldsymbol{X}'$. Let's denote this new hat-matrix by $\boldsymbol{H}_\lambda$.

It's important to note that the inverse of $(\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_p)$ will always exist, also when p is bigger than n and the matrix $\boldsymbol{X}'\boldsymbol{X}$ is for that reason not invertible anymore.
The inverse always exists because $\boldsymbol{X}'\boldsymbol{X}$ is symmetric and is therefore orthogonally diagonalizable. This matrix can thus be written as $\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}'$, where $\boldsymbol{D}$ is the diagonal matrix with the eigenvalues of $\boldsymbol{X}'\boldsymbol{X}$ on the diagonal. Since $\boldsymbol{X}'\boldsymbol{X}$ is positive semi definite[2], we know that all diagonal-elements will be bigger or equal to zero. If we now write $\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_p$ as $\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}' + \boldsymbol{V}\lambda \boldsymbol{I}_p\boldsymbol{V}'$, we see that $\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_p$ is also orthogonally diagonalizable: $\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_p = \boldsymbol{V}(\boldsymbol{D} + \lambda \boldsymbol{I}_p)\boldsymbol{V}'$ and that all eigenvalues of this matrix are bigger than zero (given that λ is bigger than zero). When all eigenvalues differ from zero, the corresponding matrix is invertible.
Since $\boldsymbol{X}'\boldsymbol{X}$ is symmetric, $(\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I}_p)$ is symmetric and therefore $\boldsymbol{H}_\lambda$ is a symmetric matrix too.

Exactly by the same reasoning as in the previous chapter, the cross-validated residual sum of squares can be defined in the following formula:

$$RSS_{cv}(\lambda) = \boldsymbol{y}'(\boldsymbol{I}_n - \boldsymbol{H}_\lambda)(diag(\boldsymbol{I}_n - \boldsymbol{H}_\lambda))^{-2}(\boldsymbol{I}_n - \boldsymbol{H}_\lambda)\boldsymbol{y} \tag{8}$$

One remark regarding the just introduced hat-matrix should be made. The extra term given by $\lambda \boldsymbol{I}_p$ implies that all covariates are equally penalized. However, one could think of situations where some covariates should not get an extra penalty while others do. In that case, the matrix $\boldsymbol{I}_p$ could be replaced by a matrix $\boldsymbol{A}$, with zeros as well as ones on the diagonal. Of course, one should be careful not to incorporate to many unpenalized covariates in the model since the hat-matrix still has to be invertible.

3.1 finding the optimal λ

Since we want to minimize the RSS_{cv}, we would like to find the λ for which $\frac{dRSS_{cv}(\lambda)}{d\lambda}$ equals zero (or comes close to zero).

The first step is to compute this derivative. Since $d(UV) = (dU)V + U(dV)$ for matrices U and V and this can be extended to $d(UVW) = (dU)VW + U(dV)W + UV(dW)$ and so on, the first step is to use this product rule.

[2]To proof that the matrix $\boldsymbol{X}'\boldsymbol{X}$ is positive semi definite, we have to show that $\boldsymbol{z}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{z}$ is bigger than or equal to zero for all vectors $\boldsymbol{z}$. This can be easily shown by rewriting the expression:

$$\boldsymbol{z}'X'X\boldsymbol{z} = (X\boldsymbol{z})'(X\boldsymbol{z}) = ||X\boldsymbol{z}||^2 \geq 0.$$

$$\begin{aligned}\frac{dRSS_{cv}(\lambda)}{d\lambda} &= \frac{d(\boldsymbol{y}')}{d\lambda}(I_n - H_\lambda)(diag(I_n - H_\lambda))^{-2}(I_n - H_\lambda)\boldsymbol{y} \\ &+ \boldsymbol{y}'\frac{d(I_n - H_\lambda)}{d\lambda}(diag(I_n - H_\lambda))^{-2}(I_n - H_\lambda)\boldsymbol{y} \\ &+ \boldsymbol{y}'(I_n - H_\lambda)\frac{d((diag(I_n - H_\lambda))^{-2})}{d\lambda}(I_n - H_\lambda)\boldsymbol{y} \\ &+ \boldsymbol{y}'(I_n - H_\lambda)(diag(I_n - H_\lambda))^{-2}\frac{d(I_n - H_\lambda)}{d\lambda}\boldsymbol{y} \\ &+ \boldsymbol{y}'(I_n - H_\lambda)(diag(I_n - H_\lambda))^{-2}(I_n - H_\lambda)\frac{d(\boldsymbol{y})}{d\lambda}.\end{aligned} \tag{9}$$

Since $\boldsymbol{y}$ is just a vector of constants that do not depend on λ, $\frac{d(\boldsymbol{y}')}{d\lambda}$ and $\frac{d(\boldsymbol{y})}{d\lambda}$ are both zero. To find an expression for $\frac{d(I_n - H_\lambda)}{d\lambda}$ we use the rule that $d(F^{-1}) = -F^{-1}(dF)F^{-1}$. From now on, I will usually write $d(.)$ instead of $\frac{d(.)}{d\lambda}$.

$$\begin{aligned}d(I_n - H_\lambda) &= d(I_n) - d(H_\lambda) \\ &= -d(X(X'X + \lambda I_p)^{-1}X') \\ &= -Xd((X'X + \lambda I_p)^{-1})X' \\ &= -X \cdot -(X'X + \lambda I_p)^{-1}d(X'X + \lambda I_p)(X'X + \lambda I_p)^{-1}X' \\ &= X(X'X + \lambda I_p)^{-1}I_p(X'X + \lambda I_p)^{-1}X' \\ &= X(X'X + \lambda I_p)^{-2}X'\end{aligned}$$

The last expression we look for is the one belonging to $d((diag(I_n - H_\lambda))^{-2})$. To derive this expression, we again use the rule for differentiating the inverse of a matrix and furthermore we use the rule that says $d(diag(X)) = diag(dX)$. We start by finding $d((diag(I_n - H_\lambda))^{-1})$ since $d((diag(I_n - H_\lambda))^{-2}) = d((diag(I_n - H_\lambda))^{-1}(diag(I_n - H_\lambda))^{-1})$

$$\begin{aligned}d((diag(I_n - H_\lambda))^{-1}) &= -(diag(I_n - H_\lambda))^{-1}d(diag(I_n - H_\lambda))(diag(I_n - H_\lambda))^{-1} \\ &= -(diag(I_n - H_\lambda))^{-1}diag(d(I_n - H_\lambda))(diag(I_n - H_\lambda))^{-1} \\ &= -(diag(I_n - H_\lambda))^{-1}diag(X(X'X + \lambda I_p)^{-2}X')(diag(I_n - H_\lambda))^{-1}\end{aligned}$$

Using the product rule gives know:

$$\begin{aligned}&d((diag(I_n - H_\lambda))^{-1}(diag(I_n - H_\lambda))^{-1}) \\ &= -(diag(I_n - H_\lambda))^{-1}diag(X(X'X + \lambda I_p)^{-2}X')(diag(I_n - H_\lambda))^{-1} \times (diag(I_n - H_\lambda))^{-1} \\ &+ (diag(I_n - H_\lambda))^{-1} \times -(diag(I_n - H_\lambda))^{-1}diag(X(X'X + \lambda I_p)^{-2}X')(diag(I_n - H_\lambda))^{-1} \\ &= -2(diag(I_n - H_\lambda))^{-3}diag(X(X'X + \lambda I_p)^{-2}X')\end{aligned}$$

Since all matrices in the equation are diagonal matrices, it's allowed to change the specific order the matrices are in.

It's nice to note that using the chain rule would have given the same expression:

$$\begin{aligned} d((diag(I_n - H_\lambda))^{-2}) &= -2(diag(I_n - H_\lambda))^{-3} d(diag(I_n - H_\lambda)) \\ &= -2(diag(I_n - H_\lambda))^{-3} diag(X(X'X + \lambda I_p)^{-2} X') \end{aligned}$$

Now we have found these derivatives, we can plug them into equation (9):

$$\begin{aligned} \frac{dRSS_{cv}(\lambda)}{d\lambda} &= \boldsymbol{y}' X(X'X + \lambda I_p)^{-2} X'(diag(I_n - H_\lambda))^{-2}(I_n - H_\lambda)\boldsymbol{y} \\ &+ \boldsymbol{y}'(I_n - H_\lambda) \cdot -2(diag(I_n - H_\lambda))^{-3} diag(X(X'X + \lambda I_p)^{-2} X')(I_n - H_\lambda)\boldsymbol{y} \\ &+ \boldsymbol{y}'(I_n - H_\lambda)(diag(I_n - H_\lambda))^{-2} X(X'X + \lambda I_p)^{-2} X'\boldsymbol{y} \end{aligned} \tag{10}$$

The first and third term can be added, since $\boldsymbol{y}' X(X'X + \lambda I_p)^{-2} X'(diag(I_n - H_\lambda))^{-2}(I_n - H_\lambda)\boldsymbol{y}$ is a scalar and is therefore equal to its transpose $\boldsymbol{y}'(I_n - H_\lambda)(diag(I_n - H_\lambda))^{-2} X(X'X + \lambda I_p)^{-2} X'\boldsymbol{y}$. This results in:

$$\begin{aligned} \frac{dRSS_{cv}(\lambda)}{d\lambda} &= \boldsymbol{y}'(I_n - H_\lambda) \cdot -2(diag(I_n - H_\lambda))^{-3} diag(X(X'X + \lambda I_p)^{-2} X')(I_n - H_\lambda)\boldsymbol{y} \\ &+ 2\boldsymbol{y}'(I_n - H_\lambda)(diag(I_n - H_\lambda))^{-2} X(X'X + \lambda I_p)^{-2} X'\boldsymbol{y} \end{aligned} \tag{11}$$

Which can also be written as:

$$\begin{aligned} &\frac{dRSS_{cv}(\lambda)}{d\lambda} = 2\boldsymbol{y}'(I_n - H_\lambda)(diag(I_n - H_\lambda))^{-2} \\ &\left(-(diag(I_n - H_\lambda))^{-1} diag(X(X'X + \lambda I_p)^{-2} X')(I_n - H_\lambda) + X(X'X + \lambda I_p)^{-2} X'\right)\boldsymbol{y} \end{aligned} \tag{12}$$

Unfortunately, finding the root of this equation is far from trivial and therefore, I did not make use of this derivative in the end. I did compute the second derivative (although the derivation is not shown here) because I wanted to use the Newton algorithm to find the root of the first derivative, but the expression for the second derivative was, even after simplification, so difficult that I did not implement the Newton algorithm.
Still, it would be nice to examine whether a root-finding algorithm (such as the secant method) could be used in combination with the algebraic expression for the first derivative.

4 Binary logistic regression

In binary logistic regression, we assume that the binary response variables are independent and Bernoulli distributed: $y_i \sim Ber(p_i)$. The aim is to find an estimate for p_i based on covariates $x_{i1} \cdots x_{ip}$. If we still want to use the linear predictor $\eta_i = \sum_{j=1}^{p} \beta_j x_{ij}$ a link-function has to be chosen in order to map the interval $[0, 1]$ to the whole real line (since η_i can take on any value). In logistic regression, this link function is the *logit* and the model becomes:

$$logit\left(p_i(\boldsymbol{x}_i)\right) = log\left(\frac{p_i(\boldsymbol{x}_i)}{1 - p_i(\boldsymbol{x}_i)}\right) = \sum_{j=1}^{p} \beta_j x_{ij},$$

which is equivalent to

$$p_i(\boldsymbol{x}_i) = exp\left(\sum_{j=1}^{p} \beta_j x_{ij}\right) / \left(1 + exp\left(\sum_{j=1}^{p} \beta_j x_{ij}\right)\right).$$

The likelihood function corresponding to this model is given by

$$L(\boldsymbol{\beta}, \boldsymbol{y}) = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{(1-y_i)}$$

from which the log-likelihood can be derived:

$$l(\boldsymbol{\beta}, \boldsymbol{y}) = \sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i) \log(1 - p_i).$$

As always, we are looking for a vector $\boldsymbol{\beta}$ that predicts the values of the y_i's as well as possible. Since y_i can only have two values (0 and 1) and $p_i(\boldsymbol{x}_i)$ gives the probability that $y_i = 1$, given the value of $\boldsymbol{x}_i = (x_{i1}, \cdots, x_{ip})'$, it is not completely straightforward to define the prediction error. Three frequently used measures (see also [1]) are the following:

(1) *classification error*

$$\begin{aligned} CE = 1 \quad & \text{if } y_{new} = 1 \text{ and } \hat{p} < \frac{1}{2} \text{ or } y_{new} = 0 \text{ and } \hat{p} > \frac{1}{2} \\ = \frac{1}{2} \quad & \text{if } \hat{p} = \frac{1}{2} \\ = 0 \quad & \text{otherwise} \end{aligned}$$

(2) *squared error*

$$SE = (y_{new} - \hat{p})^2$$

(3) *minus log-likelihood error*

$$ML = -\left(y_{new} \log(\hat{p}) + (1 - y_{new}) \log(1 - \hat{p})\right)$$

Normally, the optimal $\boldsymbol{\beta}$ is chosen to be the maximum likelihood estimator (MLE), denoted by $\hat{\boldsymbol{\beta}}$. Since this estimator maximizes the log-likelihood, it minimizes the *minus log-likelihood error* and is therefore especially useful in combination with this measure.

The question becomes how to find this $\hat{\boldsymbol{\beta}}$, and since we usually don't have an independent validation set, the next question is if there is a way to find (an approximation of) the cross-validated MLE's $\hat{\boldsymbol{\beta}}_{(-i)}$, without having to fit the model n times. To find $\hat{\boldsymbol{\beta}}$, corresponding to the whole dataset, the Newton-Raphson maximization procedure can be used. For this, all we need is the first and second derivative of $l(\boldsymbol{\beta}, \boldsymbol{y})$ and some initial point $\boldsymbol{\beta_0}$.
To find the derivatives, first the log-likelihood has to be rewritten:

$$\begin{aligned}
l(\beta, \boldsymbol{y}) &= \sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \\
&= \sum_{i=1}^{n} y_i \log\left(\frac{\exp(\boldsymbol{x}_i'\beta)}{1 + \exp(\boldsymbol{x}_i'\beta)}\right) + (1 - y_i) \log\left(1 - \frac{\exp(\boldsymbol{x}_i'\beta)}{1 + \exp(\boldsymbol{x}_i'\beta)}\right) \\
&= \sum_{i=1}^{n} y_i(\boldsymbol{x}_i'\beta) - y_i \log\left(1 + \exp(\boldsymbol{x}_i'\beta)\right) + (1 - y_i) \log\left(\frac{1}{1 + \exp(\boldsymbol{x}_i'\beta)}\right) \\
&= \sum_{i=1}^{n} y_i(\boldsymbol{x}_i'\beta) - y_i \log\left(1 + \exp(\boldsymbol{x}_i'\beta)\right) - (1 - y_i) \log\left(1 + \exp(\boldsymbol{x}_i'\beta)\right) \\
&= \sum_{i=1}^{n} y_i(\boldsymbol{x}_i'\beta) - \log\left(1 + \exp(\boldsymbol{x}_i'\beta)\right)
\end{aligned}$$

The first derivative (the gradient) now becomes:

$$\begin{aligned}
l'(\beta, \boldsymbol{y}) &= \sum_{i=1}^{n} \boldsymbol{x}_i y_i - \frac{\boldsymbol{x}_i \exp(\boldsymbol{x}_i'\beta)}{1 + \exp(\boldsymbol{x}_i'\beta)} \\
&= \sum_{i=1}^{n} \boldsymbol{x}_i(y_i - p_i(\boldsymbol{x}_i)) \\
&= X'(\boldsymbol{y} - \boldsymbol{p}).
\end{aligned}$$

The second derivative (the hessian) becomes:

$$\begin{aligned}
l''(\beta, \boldsymbol{y}) &= \sum_{i=1}^{n} -\left(\frac{(\boldsymbol{x}_i')^2 \exp(\boldsymbol{x}_i'\beta)(1 + \exp(\boldsymbol{x}_i'\beta))}{(1 + \exp(\boldsymbol{x}_i'\beta))^2} - \frac{(\boldsymbol{x}_i')^2 \exp(\boldsymbol{x}_i'\beta)^2}{(1 + \exp(\boldsymbol{x}_i'\beta))^2}\right) \\
&= \sum_{i=1}^{n} -((\boldsymbol{x}_i')^2 p_i(\boldsymbol{x}_i) - (\boldsymbol{x}_i')^2 p_i(\boldsymbol{x}_i)^2) \\
&= -\boldsymbol{X}'\boldsymbol{W}\boldsymbol{X},
\end{aligned}$$

where $\boldsymbol{W}$ is a diagonal matrix with $w_{kk} = p_k(1 - p_k)$.
The Newton-Raphson method will converge since the hessian matrix is negative semi definite ($\boldsymbol{X}'\boldsymbol{X}$ is positive semi definite and the elements of $\boldsymbol{W}$ can only be non-negative) and therefore the log-likelihood function is concave, but still, it can take much time to find $\hat{\boldsymbol{\beta}}$. If we want to find all the $\hat{\boldsymbol{\beta}}_{(-i)}$'s in this way, this can be very time consuming.

A different idea is to start with $\hat{\boldsymbol{\beta}}$ and adjust this value a little bit. In [11], this idea is discussed in more detail. First some notation has to be introduced. The contribution of observation i to the log-likelihood is defined as

$$l_i(\boldsymbol{\beta}) = l(\boldsymbol{\beta}) - l_{(-i)}(\boldsymbol{\beta}),$$

where $l_{(-i)}(\boldsymbol{\beta})$ is the log-likelihood when observation i is left out. The value of $\boldsymbol{\beta}$ that maximizes $l_{(-i)}(\boldsymbol{\beta})$ is the earlier introduced $\hat{\boldsymbol{\beta}}_{(-i)}$.

We can make a first order Taylor approximation of $l'_{(-i)}(\boldsymbol{\beta})$ at $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}$ which gives:

$$l'_{(-i)}(\boldsymbol{\beta}) = l'_{(-i)}(\hat{\boldsymbol{\beta}}) + (\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})l''_{(-i)}(\hat{\boldsymbol{\beta}}).$$

Now, we can use the fact that $\hat{\boldsymbol{\beta}}_{(-i)}$ maximizes $l_{(-i)}(\boldsymbol{\beta})$ and therefore $l'_{(-i)}(\hat{\boldsymbol{\beta}}_{(-i)})$ equals 0:

$$0 = l'_{(-i)}(\hat{\boldsymbol{\beta}}) + (\hat{\boldsymbol{\beta}}_{(-i)} - \hat{\boldsymbol{\beta}})l''_{(-i)}(\hat{\boldsymbol{\beta}}),$$

from which

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} - (l''_{(-i)}(\hat{\boldsymbol{\beta}}))^{-1}l'_{(-i)}(\hat{\boldsymbol{\beta}})$$

follows. (Which equals 1 Newton-Raphson step with $\hat{\boldsymbol{\beta}}$ as initial point.)

If we put in the formulas for the hessian and the gradient we get:

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} + (\boldsymbol{X}'_{(-i)}\hat{\boldsymbol{W}}_{(-i)}\boldsymbol{X}_{(-i)})^{-1}\boldsymbol{X}'_{(-i)}(\boldsymbol{y}_{(-i)} - \hat{\boldsymbol{p}}_{(-i)}),$$

where the values of the $\hat{p}_i$'s (and thus also the $\hat{w}_{ii}$'s) are calculated based on the value for $\hat{\boldsymbol{\beta}}$.

The only term in this formula that is difficult to calculate is the inverse of the hessian-matrix. However, the Sherman-Morrison-Woodbury theorem as discussed in section 2 can be used to simplify this term. Because $\hat{W}$ is a diagonal matrix

$$\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X} = \sum_{i=1}^{n} \hat{w}_{ii}\boldsymbol{x}_i\boldsymbol{x}'_i.$$

and therefore we can rewrite $(\boldsymbol{X}'_{(-i)}\hat{\boldsymbol{W}}_{(-i)}\boldsymbol{X}_{(-i)})^{-1}$ in the desired form:

$$(\boldsymbol{X}'_{(-i)}\hat{\boldsymbol{W}}_{(-i)}\boldsymbol{X}_{(-i)})^{-1} = (\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X} - \hat{w}_{ii}\boldsymbol{x}_i\boldsymbol{x}'_i)^{-1}.$$

Using the Sherman-Morrison-Woodbury theorem this results in:

$$(\boldsymbol{X}'_{(-i)}\hat{\boldsymbol{W}}_{(-i)}\boldsymbol{X}_{(-i)})^{-1} = (\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X})^{-1} + \frac{(\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X})^{-1}\hat{w}_{ii}\boldsymbol{x}_i\boldsymbol{x}'_i(\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X})^{-1}}{1-\hat{w}_{ii}\boldsymbol{x}'_i(\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X})^{-1}\boldsymbol{x}_i}$$

which eventually gives

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} + \left((\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X})^{-1} + \frac{(\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X})^{-1}\hat{w}_{ii}\boldsymbol{x}_i\boldsymbol{x}'_i(\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X})^{-1}}{1-\hat{w}_{ii}\boldsymbol{x}'_i(\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X})^{-1}\boldsymbol{x}_i}\right)\boldsymbol{X}'_{(-i)}(\boldsymbol{y}_{(-i)} - \hat{\boldsymbol{p}}_{(-i)}). \tag{13}$$

We can introduce new variables $\boldsymbol{Z} = \hat{\boldsymbol{W}}^{\frac{1}{2}}\boldsymbol{X}$ and $\boldsymbol{v} = \hat{\boldsymbol{W}}^{-\frac{1}{2}}(\boldsymbol{y} - \hat{\boldsymbol{p}})$ and rewrite (13) into

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} + \left((\boldsymbol{Z}'\boldsymbol{Z})^{-1} + \frac{(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}}{1-\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i}\right)(\boldsymbol{Z}'\boldsymbol{v} - \boldsymbol{z}_i v_i). \tag{14}$$

Now we want to simplify this expression. (In this simplification process we can use the fact that the Newton-Raphson algorithm converged to $\hat{\boldsymbol{\beta}}$ so $\hat{\boldsymbol{\beta}} + (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{v}$ equals $\hat{\boldsymbol{\beta}}$, which implies that $(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{v}$ equals 0.)

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &+ (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{v} - (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i v_i + \frac{(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}(\boldsymbol{Z}'\boldsymbol{v} - \boldsymbol{z}_i v_i)}{1-\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i v_i(1-\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i) - (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}(\boldsymbol{Z}'\boldsymbol{v} - \boldsymbol{z}_i v_i)}{1-\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i v_i - (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i v_i\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i - (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}(\boldsymbol{Z}'\boldsymbol{v} - \boldsymbol{z}_i v_i)}{1-\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i v_i - (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}(v_i\boldsymbol{z}_i + \boldsymbol{Z}'\boldsymbol{v} - \boldsymbol{z}_i v_i)}{1-\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i v_i - (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{v}}{1-\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i v_i}{1-\boldsymbol{z}'_i(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{z}_i}
\end{aligned}$$

So eventually, we get:

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} - \frac{(\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X})^{-1}\boldsymbol{x}_i(y_i - p_i)}{1-v_{ii}}, \tag{15}$$

where v_{ii} is the i^{th} diagonal element from the matrix which is given by $\hat{\boldsymbol{W}}^{\frac{1}{2}}\boldsymbol{X}(\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X})^{-1}\boldsymbol{X}'\hat{\boldsymbol{W}}^{\frac{1}{2}}$.

Now we want to use this approximated $\hat{\boldsymbol{\beta}}_{(-i)}$ in the cross-validated version of the *mean minus log-likelihood* which is given by

$$MML_{CV} = -n^{-1} \sum_{i=1}^{n} \left(y_i \log(\hat{p}_{(-i)}(\boldsymbol{x}_i)) + (1 - y_i) \log(1 - \hat{p}_{(-i)}(\boldsymbol{x}_i)) \right), \quad (16)$$

where $\hat{p}_{(-i)}(\boldsymbol{x}_i)$ is just $\dfrac{\exp(\boldsymbol{x}_i' \hat{\boldsymbol{\beta}}_{(-i)})}{1 + \exp(\boldsymbol{x}_i' \hat{\boldsymbol{\beta}}_{(-i)})}$.

The first option would be to substitute the approximated $\hat{\boldsymbol{\beta}}_{(-i)}$'s in the actual formula for MML_{CV}. Because all approximated $\hat{\boldsymbol{\beta}}_{(-i)}$'s only depend on one inverse calculation (the inverse of the hessian matrix in the model where all observations are still included) and some matrix multiplications, this will definitely be a faster approach than calculating MML_{CV} based on the real values of the leave-one-out MLE's. However, we wanted to see if we could make the calculations even faster by not only using a Taylor approximation of the $\hat{\boldsymbol{\beta}}_{(-i)}$'s but also of the MML_{CV}-formula itself.

Let's first denote $\left(y_i \log(\hat{p}_{(-i)}(\boldsymbol{x}_i)) + (1 - y_i) \log(1 - \hat{p}_{(-i)}(\boldsymbol{x}_i)) \right)$ by $f(\hat{\boldsymbol{\beta}}_{\boldsymbol{(-i)}})$. A first order Taylor approximation around $\hat{\boldsymbol{\beta}}$ is now given by:

$$f(\hat{\boldsymbol{\beta}}_{\boldsymbol{(-i)}}) = f(\hat{\boldsymbol{\beta}}) + f'(\hat{\boldsymbol{\beta}})(\hat{\boldsymbol{\beta}}_{\boldsymbol{(-i)}} - \hat{\boldsymbol{\beta}}),$$

where the error term is given by $O\left((\hat{\boldsymbol{\beta}}_{(-i)} - \hat{\boldsymbol{\beta}})^2\right)$.

Note that a higher order Taylor approximation would not give more precise results, since the expression $\hat{\boldsymbol{\beta}}_{\boldsymbol{(-i)}} - \hat{\boldsymbol{\beta}}$ (where we take the expression for $\hat{\boldsymbol{\beta}}_{(-i)}$ as given in formula(12)) has itself an error term of $O\left((\hat{\boldsymbol{\beta}}_{(-i)} - \hat{\boldsymbol{\beta}})^2\right)$ because it is derived using again a first order Taylor approximation.

If we plug in the expressions for the first and second derivative (as derived earlier) we get:

$$f(\hat{\boldsymbol{\beta}}_{\boldsymbol{(-i)}}) = (y_i \log(\hat{p}(\boldsymbol{x}_i)) + (1 - y_i) \log(1 - \hat{p}(\boldsymbol{x}_i))) + \boldsymbol{x}_i'(y_i - \hat{p}(\boldsymbol{x}_i))(\hat{\boldsymbol{\beta}}_{\boldsymbol{(-i)}} - \hat{\boldsymbol{\beta}}),$$

where $\hat{p}(\boldsymbol{x}_i)$ is given by $\dfrac{\exp(\boldsymbol{x}_i' \hat{\boldsymbol{\beta}})}{1 + \exp(\boldsymbol{x}_i' \hat{\boldsymbol{\beta}})}$.

When the expression from formula (12) is used, this can be rewritten as:

$$f(\hat{\boldsymbol{\beta}}_{\boldsymbol{(-i)}}) = (y_i \log(\hat{p}(\boldsymbol{x}_i)) + (1 - y_i) \log(1 - \hat{p}(\boldsymbol{x}_i))) - \boldsymbol{x}_i'(y_i - \hat{p}(\boldsymbol{x}_i)) \frac{(X' \hat{W} X)^{-1} \boldsymbol{x}_i (y_i - p_i)}{1 - v_{ii}} \quad (17)$$

This equation can now be plugged in equation (16) to find an approximation of the cross-validated mean minus log-likelihood:

$$MML_{CV} \approx$$
$$-n^{-1}\sum_{i=1}^{n}\left((y_i\log(\hat{p}(\boldsymbol{x}_i))+(1-y_i)\log(1-\hat{p}(\boldsymbol{x}_i)))-\boldsymbol{x}_i'(y_i-\hat{p}(\boldsymbol{x}_i))\frac{(X'\hat{W}X)^{-1}\boldsymbol{x}_i(y_i-\hat{p}_i)}{1-v_{ii}}\right).$$

The degree of the error term stays quadratic.

This formula can still be simplified a little, if we realize that $\boldsymbol{x}_i'(X'\hat{W}X)^{-1}\boldsymbol{x}_i$ is just the i^{th} diagonal element of the matrix $X(X'\hat{W}X)^{-1}X'$ which is almost equal to the earlier defined matrix V. Actually $\boldsymbol{x}_i'(X'\hat{W}X)^{-1}\boldsymbol{x}_i$ is just v_{ii} divided by w_{ii}. The formula now becomes:

$$MML_{CV} \approx$$
$$-n^{-1}\sum_{i=1}^{n}\left((y_i\log(\hat{p}(\boldsymbol{x}_i))+(1-y_i)\log(1-\hat{p}(\boldsymbol{x}_i)))-(y_i-\hat{p}(\boldsymbol{x}_i))\frac{\frac{v_{ii}}{w_{ii}}(y_i-\hat{p}_i)}{1-v_{ii}}\right).$$

In matrix notation, we get:

$$MML_{CV} \approx$$
$$-n^{-1}\left(\boldsymbol{y}'\log(\hat{\boldsymbol{p}})+(\boldsymbol{1}-\boldsymbol{y})'\log(\boldsymbol{1}-\hat{\boldsymbol{p}})-(\boldsymbol{y}-\hat{\boldsymbol{p}})'(diag(I_n-V))^{-1}(diag(V))W^{-1}(\boldsymbol{y}-\hat{\boldsymbol{p}})\right)$$

Although this expression would be very convenient to work with, since there is a direct matrix multiplication to calculate an approximated version of the mean-minus log-likelihood, simulations showed that the approximation was a little too inaccurate.

5 Binary logistic ridge regression

5.1 Finding an expression for $\hat{\boldsymbol{\beta}}_{(-i)}$

Just as in the case of linear regression, the standard logistic regression model is not suited for data sets with a large number of covariates. That's where logistic ridge regression comes in.

The log-likelihood function looks almost the same as the log-likelihood function in standard ridge regression, only a quadratic penalty term is added:

$$l(\beta, \boldsymbol{y}) = \sum_{i=1}^{n} \left(y_i \log(p_i) + (1 - y_i) \log(1 - p_i)\right) - \lambda \beta' \beta.$$

The first and second derivative can be calculated in exactly the same way as done in the previous section and become:

$$l'(\beta, \boldsymbol{y}) = X'(\boldsymbol{y} - \boldsymbol{p}) - 2\lambda\beta$$

and

$$l''(\beta, \boldsymbol{y}) = -X'WX - 2\lambda I_p,$$

where W is still the diagonal matrix with $w_{kk} = p_k(1 - p_k)$.
For a given λ, the way of finding $\hat{\boldsymbol{\beta}}$ is exactly the same as before: just apply the Newton-Raphson algorithm.

To find approximations of the $\hat{\boldsymbol{\beta}}_{(-i)}$'s a Taylor approximation of $l'_{(-i)}(\boldsymbol{\beta})$ around $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}$ can again be made and this time this results in

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} + (X'_{(-i)}\hat{W}_{(-i)}X_{(-i)} + 2\lambda I_p)^{-1} \left(X'_{(-i)}(\boldsymbol{y}_{(-i)} - \hat{\boldsymbol{p}}_{(-i)}) - 2\lambda\hat{\boldsymbol{\beta}}\right),$$

where the values of the $\hat{p}_i$'s (and thus also the $\hat{w}_{ii}$'s) are calculated based on the value for $\hat{\boldsymbol{\beta}}$.

Since calculating the inverse of a matrix is a time-consuming operation, we would like to find a better expression for $(X'_{(-i)}\hat{W}_{(-i)}X_{(-i)} + 2\lambda I_p)^{-1}$ and this can be done by using the Sherman-Morrison-Woodbury theory:

$$\begin{aligned}(X'_{(-i)}\hat{W}_{(-i)}X_{(-i)} + 2\lambda I_p)^{-1} &= (X'\hat{W}X + 2\lambda I_p - \hat{w}_{ii}\boldsymbol{x}_i\boldsymbol{x}'_i)^{-1} \\ &= (X'\hat{W}X + 2\lambda I_p)^{-1} + \frac{(X'\hat{W}X + 2\lambda I_p)^{-1}\hat{w}_{ii}\boldsymbol{x}_i\boldsymbol{x}'_i(X'\hat{W}X + 2\lambda I_p)^{-1}}{1 - \hat{w}_{ii}\boldsymbol{x}'_i(X'\hat{W}X + 2\lambda I_p)^{-1}\boldsymbol{x}_i}.\end{aligned}$$

So we get:

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} + \left((X'\hat{W}X + 2\lambda I_p)^{-1} + \frac{(X'\hat{W}X + 2\lambda I_p)^{-1}\hat{w}_{ii}\boldsymbol{x}_i\boldsymbol{x}_i'(X'\hat{W}X + 2\lambda I_p)^{-1}}{1 - \hat{w}_{ii}\boldsymbol{x}_i'(X'\hat{W}X + 2\lambda I_p)^{-1}\boldsymbol{x}_i} \right) \left(X_{(-i)}'(\boldsymbol{y}_{(-i)} - \hat{\boldsymbol{p}}_{(-i)}) - 2\lambda\hat{\boldsymbol{\beta}} \right)$$

We can introduce the same variables ($Z = \hat{W}^{\frac{1}{2}}X$ and $\boldsymbol{v} = \hat{W}^{-\frac{1}{2}}(\boldsymbol{y} - \hat{\boldsymbol{p}})$) as in the previous section to rewrite this equation in a simpler form:

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} + \left((Z'Z + 2\lambda I_p)^{-1} + \frac{(Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i\boldsymbol{z}_i'(Z'Z + 2\lambda I_p)^{-1}}{1 - \boldsymbol{z}_i'(Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i} \right) \left(Z'v - \boldsymbol{z}_i v_i - 2\lambda\hat{\boldsymbol{\beta}} \right)$$

In the simplification step, we again use the fact that the Newton Raphson algorithm converged to $\hat{\boldsymbol{\beta}}$ and therefore $(Z'Z + 2\lambda I_p)^{-1}(Z'v - 2\lambda\hat{\boldsymbol{\beta}})$ equals 0.

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &+ \left((Z'Z + 2\lambda I_p)^{-1} + \frac{(Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i\boldsymbol{z}_i'(Z'Z + 2\lambda I_p)^{-1}}{1 - \boldsymbol{z}_i'(Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i} \right) \left(Z'v - \boldsymbol{z}_i v_i - 2\lambda\hat{\boldsymbol{\beta}} \right) \\
\hat{\boldsymbol{\beta}} &+ (Z'Z + 2\lambda I_p)^{-1} \left(Z'v - 2\lambda\hat{\boldsymbol{\beta}} \right) \\
&- (Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i v_i + \frac{(Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i\boldsymbol{z}_i'(Z'Z + 2\lambda I_p)^{-1} \left(Z'v - \boldsymbol{z}_i v_i - 2\lambda\hat{\boldsymbol{\beta}} \right)}{1 - \boldsymbol{z}_i'(Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i} \\
\hat{\boldsymbol{\beta}} &- (Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i v_i + \frac{(Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i\boldsymbol{z}_i'(Z'Z + 2\lambda I_p)^{-1} \left(Z'v - \boldsymbol{z}_i v_i - 2\lambda\hat{\boldsymbol{\beta}} \right)}{1 - \boldsymbol{z}_i'(Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i v_i - (Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i\boldsymbol{z}_i'(Z'Z + 2\lambda I_p)^{-1} \left(\boldsymbol{z}_i v_i + Z'v - \boldsymbol{z}_i v_i - 2\lambda\hat{\boldsymbol{\beta}} \right)}{1 - \boldsymbol{z}_i'(Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i} \\
\hat{\boldsymbol{\beta}} &- \frac{(Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i v_i}{1 - \boldsymbol{z}_i'(Z'Z + 2\lambda I_p)^{-1}\boldsymbol{z}_i}
\end{aligned}$$

When we go back to the original notation we have now found:

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} - \frac{(X'\hat{W}X + 2\lambda I_p)^{-1}\boldsymbol{x}_i(y_i - \hat{p}_i)}{1 - v_{ii}}, \tag{18}$$

where $v_{ii} = \hat{w}_{ii}\boldsymbol{x}_i'(X'\hat{W}X + 2\lambda I_p)^{-1}\boldsymbol{x}_i$, which is the i^{th} diagonal element of the matrix V given by $\hat{W}^{\frac{1}{2}}X(X'\hat{W}X + 2\lambda I_p)^{-1}X'\hat{W}^{\frac{1}{2}}$.

As before, we want to have an expression for the cross-validated mean minus log-likelihood, given by:

$$MML_{CV}(\lambda) = -n^{-1} \sum_{i=1}^{n} \left(y_i \log(\hat{p}_{(-i)}(\boldsymbol{x}_i)) + (1 - y_i) \log(1 - \hat{p}_{(-i)}(\boldsymbol{x}_i)) \right).$$

Note that the name is a bit misleading, since we first introduced a likelihood with an extra penalty term and this term is left out in the previous expression. However, the $\hat{p}_{(-i)}$'s are constructed with the penalized likelihood and now we are only interested in the goodness of the prediction.

This formula depends on the value of λ (chosen beforehand), because the values of the $\hat{p}_{(-i)}$'s depend on the values of the $\hat{\boldsymbol{\beta}}_{(-i)}$'s which in turn depend on the value of $\hat{\boldsymbol{\beta}}$ which of course depends on the chosen λ.

In an attempt to calculate the values of the MML_{CV}-formula faster than by just plugging in the approximated values for the $\hat{\boldsymbol{\beta}}_{(-i)}$'s, we use another Taylor approximation as before. The only thing that changes, compared to the normal logistic regression model, is the expression for $\hat{\boldsymbol{\beta}}_{\boldsymbol{(-i)}} - \hat{\boldsymbol{\beta}}$ and the cross-validated minus log-likelihood is now approximately given by:

$$\begin{aligned} &MML_{CV}(\lambda) \approx \\ &-n^{-1} \sum_{i=1}^{n} \left((y_i \log(\hat{p}(\boldsymbol{x}_i)) + (1 - y_i) \log(1 - \hat{p}(\boldsymbol{x}_i))) - \boldsymbol{x}_i'(y_i - \hat{p}(\boldsymbol{x}_i)) \frac{(X'\hat{W}X + 2\lambda I_p)^{-1} \boldsymbol{x}_i (y_i - \hat{p}_i)}{1 - v_{ii}} \right). \end{aligned}$$

Again, this expression can be simplified a little. The expression $\boldsymbol{x}_i'(X'\hat{W}X + 2\lambda I_p)^{-1}\boldsymbol{x}_i$ equals $\left(X(X'\hat{W}X + 2\lambda I_p)^{-1} X' \right)_{ii}$ which equals $v_{ii}/\hat{w}_{ii}$ where $V = \hat{W}^{\frac{1}{2}} X (X'\hat{W}X + 2\lambda I_p)^{-1} X' \hat{W}^{\frac{1}{2}}$.

This gives us:

$$\begin{aligned} &MML_{CV}(\lambda) \approx \\ &-n^{-1} \sum_{i=1}^{n} \left((y_i \log(\hat{p}(\boldsymbol{x}_i)) + (1 - y_i) \log(1 - \hat{p}(\boldsymbol{x}_i))) - (y_i - \hat{p}(\boldsymbol{x}_i)) \frac{\frac{v_{ii}}{\hat{w}_{ii}} (y_i - \hat{p}_i)}{1 - v_{ii}} \right). \end{aligned}$$

In matrix notation, we get:

$$\begin{aligned} &MML_{CV}(\lambda) \approx \\ &-n^{-1} \left(\boldsymbol{y}' \log(\hat{\boldsymbol{p}}) + (\boldsymbol{1} - \boldsymbol{y})' \log(\boldsymbol{1} - \hat{\boldsymbol{p}}) - (\boldsymbol{y} - \hat{\boldsymbol{p}})' (diag(I_n - V))^{-1} (diag(V)) \hat{W}^{-1} (\boldsymbol{y} - \hat{\boldsymbol{p}}) \right) \end{aligned}$$

It would be nice to find the value of λ that minimized the above expression and to find this value of λ it would be very convenient to have an expression for the derivative of $MML_{CV}(\lambda)$ with respect to λ. However, later on we found out

that using the formula of the MML_{CV} based on the approximated version of the leave-one-out estimates did have advantages over using the Taylor version of the MML_{CV} and we changed our focus to accurate predicting instead of solving equations analytically.

5.2 Computational problems

When we have a data set with many covariates, it may be difficult to calculate the matrix V because in this calculation the inverse of an $p \times p$ matrix has to be found and this is not only a time-consuming operation, but it also takes much computer memory. To avoid this problem, some reparametrization trick can be used.

When the Newton-Raphson algorithm converges, the gradient of the penalized likelihood will eventually become zero:

$$X'(\boldsymbol{y} - \boldsymbol{p}) - 2\lambda\beta = 0.$$

From this we can conclude that $\boldsymbol{\beta}$ lies in the column space of X'. Therefore, we can write $\boldsymbol{\beta} = X'\boldsymbol{\gamma}$. Here, $\boldsymbol{\gamma}$ is a n-dimensional vector and in the corresponding model the former $p \times p$ matrices will now be replaced by $n \times n$ matrices, as will be shown below.

This time we work with the formula:

$$\hat{\boldsymbol{\gamma}}_{(-i)} = \hat{\boldsymbol{\gamma}} - (l''_{(-i)}(\hat{\boldsymbol{\gamma}}))^{-1} l'_{(-i)}(\hat{\boldsymbol{\gamma}}),$$

where the gradient and the hessian are given by:

$$l'(\gamma, \boldsymbol{y}) = X\left(X'(\boldsymbol{y} - \boldsymbol{p}) - 2\lambda X'\boldsymbol{\gamma}\right)$$

and

$$l''(\gamma, \boldsymbol{y}) = -X(X'WX + 2\lambda I_p,)X'$$

where W is still the diagonal matrix with $w_{kk} = p_k(1 - p_k)$.

After substitution of these formulas, we get:

$$\hat{\boldsymbol{\gamma}}_{(-i)} = \hat{\boldsymbol{\gamma}} + \left(X(X'_{(-i)}W_{(-i)}X_{(-i)} + 2\lambda I_p,)X'\right)^{-1} X\left(X'_{(-i)}(\boldsymbol{y}_{(-i)} - \boldsymbol{p}_{(-i)}) - 2\lambda X'\boldsymbol{\gamma}\right).$$

As before, $X'_{(-i)}W_{(-i)}X_{(-i)}$ equals $(X'WX - w_{ii}\boldsymbol{x}_i\boldsymbol{x}'_i)$ and $X'_{(-i)}(\boldsymbol{y}_{(-i)} - \boldsymbol{p}_{(-i)})$ is just $X'(\boldsymbol{y} - \boldsymbol{p}) - \boldsymbol{x}_i(y_i - p_i)$.

If we rename XX' by A and $X\boldsymbol{x}_i$ by a_i since this is just the i^{th} row in A, we get:

$$\hat{\boldsymbol{\gamma}}_{(-i)} = \hat{\boldsymbol{\gamma}} + \left(AWA + 2\lambda A - w_{ii}a_i a_i'\right)^{-1}\left(A(\boldsymbol{y}-\boldsymbol{p}) - a_i(y_i - p_i) - 2\lambda A\boldsymbol{\gamma}\right).$$

The Sherman-Morrison-Woodbury theorem can again be applied and if we let $Z = W^{\frac{1}{2}}A$ and $\boldsymbol{v} = W^{-\frac{1}{2}}(\boldsymbol{y}-\boldsymbol{p})$, we get

$$\hat{\boldsymbol{\gamma}}_{(-i)} = \hat{\boldsymbol{\gamma}} + \left((Z'Z+2\lambda A)^{-1} + \frac{(Z'Z+2\lambda A)^{-1}\boldsymbol{z}_i\boldsymbol{z}_i'(Z'Z+2\lambda A)^{-1}}{1-\boldsymbol{z}_i'(Z'Z+2\lambda A)^{-1}\boldsymbol{z}_i}\right)\left(Z'v - \boldsymbol{z}_i v_i - 2\lambda A\hat{\boldsymbol{\gamma}}\right).$$

If we use that $(Z'Z+2\lambda X)^{-1}(Z'v - 2\lambda A\gamma)$ equals zero (due to convergence of the NR algorithm), we eventually get:

$$\hat{\boldsymbol{\gamma}}_{(-i)} = \hat{\boldsymbol{\gamma}} - \frac{(A'\hat{W}A + 2\lambda A)^{-1}a_i(y_i - \hat{p}_i)}{1-v_{ii}}, \tag{19}$$

where v_{ii} is the i^{th} diagonal element of the matrix V given by $W^{\frac{1}{2}}A(A'\hat{W}A + 2\lambda A)^{-1}A'W^{\frac{1}{2}}$.

Starting with $\hat{\boldsymbol{\gamma}}$ we have now a way to find $\hat{\boldsymbol{\gamma}}_{(-i)}$ and $\hat{\boldsymbol{\beta}}_{(-i)}$ can now be easily obtained since $\hat{\boldsymbol{\beta}}_{(-i)} = X'\hat{\boldsymbol{\gamma}}_{(-i)}$.
Sometimes, it is also useful to find $\hat{\boldsymbol{\gamma}}$ from a known $\hat{\boldsymbol{\beta}}$ (which is for example found by the R package "penalized"). This can be done in the following way: $\hat{\boldsymbol{\gamma}} = (XX')^{-1}X\hat{\boldsymbol{\beta}}$.

The formula for the approximation of the mean minus log-likelihood can now also be given in terms of $\boldsymbol{\gamma}$:

$$MML_{CV}(\lambda) \approx$$
$$-n^{-1}\sum_{i=1}^{n}\left((y_i\log(\hat{p}(\boldsymbol{x}_i)) + (1-y_i)\log(1-\hat{p}(\boldsymbol{x}_i))) - \boldsymbol{x}_i'(y_i - \hat{p}(\boldsymbol{x}_i))X'\frac{(A'\hat{W}A+2\lambda A)^{-1}a_i(y_i-\hat{p}_i)}{1-v_{ii}}\right),$$

where V is still given by $W^{\frac{1}{2}}A(A'\hat{W}A + 2\lambda A)^{-1}A'W^{\frac{1}{2}}$.

This expression can still be simplified when we take the $\boldsymbol{x}_i'$ and the X' together ($\boldsymbol{x}_i'X' = a_i$):

$$MML_{CV}(\lambda) \approx$$
$$-n^{-1}\sum_{i=1}^{n}\left((y_i\log(\hat{p}(\boldsymbol{x}_i)) + (1-y_i)\log(1-\hat{p}(\boldsymbol{x}_i))) - (y_i - \hat{p}(\boldsymbol{x}_i))\frac{\frac{v_{ii}}{\hat{w}_{ii}}(y_i-\hat{p}_i)}{1-v_{ii}}\right).$$

5.3 Results

To test the usefulness of the approximations for the leave-one-out estimators $\hat{\boldsymbol{\beta}}_{(-i)}$, we made several comparisons between the actual cross-validated mean minus log-likelihood, the MML_{cv} based on the approximations of $\hat{\boldsymbol{\beta}}_{(-i)}$ and the approximated MML_{cv}, where the $\hat{\boldsymbol{\beta}}_{(-i)}$'s as well as the function itself were the result of a Taylor approximation. We used mostly simulated data but we also used one real data set, namely the well-known Golub data set ([5]). It showed that the approximated leave-one-out estimates were really similar to the real leave-one-out estimates (as found by the R-package "penalized") and that the mean minus log-likelihood based on these approximations was almost identical to the one based on the real values. Simulations also indicated that the larger the number of observations, the better the approximations, as could also be expected from the theory of the Taylor approximation. Unfortunately, results generated from the Taylor approximation of the MML_{CV} could be quite inaccurate.

Since the result are really similar to the ones that will later be discussed in the survival context, graphs and tables will be omitted, except for the one below, that shows that the MML_{CV} curve based on the approximated $\hat{\boldsymbol{\beta}}_{(-i)}$'s (green curve) is really similar to the one based on the real leave-one-out estimates (blue curve). Furthermore, it is clear that the first order Taylor approximation of the MML_{CV} does not give reliable answers (purple curve), but the second order Taylor approximation (yellow curve) does. The derivation of this second order Taylor approximation is not written down in this work document because it is a straight-forward algebraic exercise and the actual formula will from this point on not be used anymore.

Another remark is that, although the approximated $\hat{\boldsymbol{\beta}}_{(-i)}$'s rely on the value of $\hat{\boldsymbol{\beta}}$, the estimate based on the full model, the curve based on these approximations nicely follows the shape of the real cross-validated MML and not the curve belonging to the apparent error.

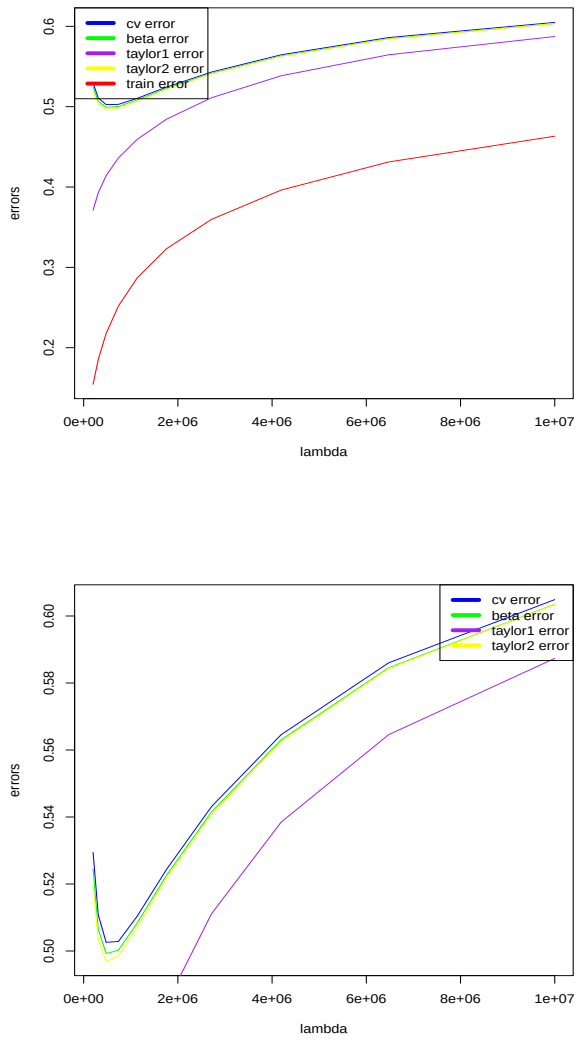


Figure 1: different approximations for the MML_{CV}, based on first 100 covariates in the Golub data set.

5.4 Mean squared error

Not only the MML is a commonly used error measure, but also the mean squared error (MSE) is often used to measure predictive performance. The cross-validated MSE is given by

$$MSE_{cv} = n^{-1} \sum_{i=1}^{n} \left(y_i - \hat{p}_{(-i)}\right)^2,$$

where $\hat{p}_{(-i)}$ is given by $\dfrac{\exp(\boldsymbol{x}_i' \hat{\boldsymbol{\beta}}_{(-i)})}{1 + \exp(\boldsymbol{x}_i' \hat{\boldsymbol{\beta}}_{(-i)})}$.

We could make a derivation that looks like the derivation of the cross-validated residual sum of squares in the linear (ridge) regression model. The derivation in the logistic model is however slightly more difficult, because the $\boldsymbol{x}_i' \hat{\boldsymbol{\beta}}_{(-i)}$-term from the linear model is now replaced by the term $\dfrac{\exp(\boldsymbol{x}_i' \hat{\boldsymbol{\beta}}_{(-i)})}{1 + \exp(\boldsymbol{x}_i' \hat{\boldsymbol{\beta}}_{(-i)}}$.

One idea is to find an expression for $\hat{p}_{(-i)}$ in terms of $\hat{p}_i$, which is just the i^{th} element of $\hat{\boldsymbol{p}}$. We can first define p_i as a function of β

$$f(\beta) = \frac{\exp(\boldsymbol{x}_i' \beta)}{1 + \exp(\boldsymbol{x}_i' \beta)}$$

and now we can approximate $f(\hat{\boldsymbol{\beta}}_{(-i)})$ (which is just $\hat{p}_{(-i)}$) with a first order Taylor expansion around $\hat{\boldsymbol{\beta}}$:

$$f(\hat{\boldsymbol{\beta}}_{(-i)}) = f(\hat{\boldsymbol{\beta}}) + \left(\frac{\partial f(\hat{\boldsymbol{\beta}})}{\partial \beta}\right)^T (\hat{\boldsymbol{\beta}}_{(-i)} - \hat{\boldsymbol{\beta}}).$$

We already found an expression for $(\hat{\boldsymbol{\beta}}_{(-i)} - \hat{\boldsymbol{\beta}})$, so we only have to find an expression for $\dfrac{\partial f(\hat{\boldsymbol{\beta}})}{\partial \beta}$.

It turns out that the derivative, evaluated for $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}$ is given by $\boldsymbol{x}_i \hat{p}_i (1 - \hat{p}_i)$, and this can also be written as $\boldsymbol{x}_i w_{ii}$, where W is the same matrix as in the previous sections.

If we fill in all known expressions in the Taylor expansion, we get

$$\hat{p}_{(-i)} = \hat{p}_i - \frac{\boldsymbol{x}_i' w_{ii} (X' \hat{W} X + 2\lambda I_p)^{-1} \boldsymbol{x}_i (y_i - \hat{p}_i)}{1 - v_{ii}} = \hat{p}_i - \frac{v_{ii}(y_i - \hat{p}_i)}{1 - v_{ii}}.$$

If we use this expression in the formula for MSE_{cv} and denote $y_i - \hat{p}_i$ by e_i (comparable to the e_i in the linear model wich was given by $y_i - \boldsymbol{x}_i' \hat{\boldsymbol{\beta}}$) we get:

$$MSE_{cv} = n^{-1} \sum_{i=1}^{n} \left(e_i + \frac{v_{ii} e_i}{1 - v_{ii}}\right)^2 = n^{-1} \sum_{i=1}^{n} \left(\frac{e_i (1 - v_{ii}) + v_{ii} e_i}{1 - v_{ii}}\right)^2 = n^{-1} \sum_{i=1}^{n} \left(\frac{e_i}{1 - v_{ii}}\right)^2.$$

This expression is also given in [1].

A different idea is to make a Taylor series expansion directly of the formula $\left(y_i - \hat{p}_{(-i)}\right)^2$. If we denote this formula by $f(\beta)$, a second order Taylor expansion would look like this:

$$f(\hat{\boldsymbol{\beta}}_{(-i)}) = f(\hat{\boldsymbol{\beta}}) + \frac{\partial f(\hat{\boldsymbol{\beta}})}{\partial \beta}(\hat{\boldsymbol{\beta}}_{(-i)} - \hat{\boldsymbol{\beta}}) + 0.5(\hat{\boldsymbol{\beta}}_{(-i)} - \hat{\boldsymbol{\beta}})' \frac{\partial^2 f(\hat{\boldsymbol{\beta}})}{\partial \beta^2}(\hat{\boldsymbol{\beta}}_{(-i)} - \hat{\boldsymbol{\beta}}).$$

Let's start with $\frac{\partial f(\hat{\boldsymbol{\beta}})}{\partial \beta}$. We already know that $\frac{\partial p_i}{\partial \beta}$ equals $\boldsymbol{x}_i \hat{p}_i(1-\hat{p}_i)$ and by using this, we get

$$\frac{\partial f(\hat{\boldsymbol{\beta}})}{\partial \beta} = -2(y_i - \hat{p}_i)\boldsymbol{x}_i \hat{p}_i(1-\hat{p}_i).$$

This expression can then be used to find the second order derivative:

$$\begin{aligned}
\frac{\partial^2 f(\hat{\boldsymbol{\beta}})}{\partial \beta^2} &= 2\boldsymbol{x}_i \boldsymbol{x}_i' \left(\hat{p}_i(1-\hat{p}_i)\right)^2 - 2(y_i - \hat{p}_i)\boldsymbol{x}_i \boldsymbol{x}_i' \left(\hat{p}_i(1-\hat{p}_i) - 2\hat{p}_i^2(1-\hat{p}_i)\right) \\
&= 2\boldsymbol{x}_i \boldsymbol{x}_i' \left(\hat{p}_i(1-\hat{p}_i)\right)^2 - 2(y_i - \hat{p}_i)\boldsymbol{x}_i \boldsymbol{x}_i' \hat{p}_i(1-\hat{p}_i)(1-2\hat{p}_i) \\
&= 2\boldsymbol{x}_i \boldsymbol{x}_i' \hat{p}_i(1-\hat{p}_i)\left(\hat{p}_i(1-\hat{p}_i) - (y_i - \hat{p}_i)(1-2\hat{p}_i)\right)
\end{aligned}$$

In the next 2 picture, the training MSE / apparent error (red), the real cross validated MSE (blue), the MSE based on approximations (green), the MSE as given in [1] (pink) and the MSE based on the just described Taylor approximation (yellow) are plotted.

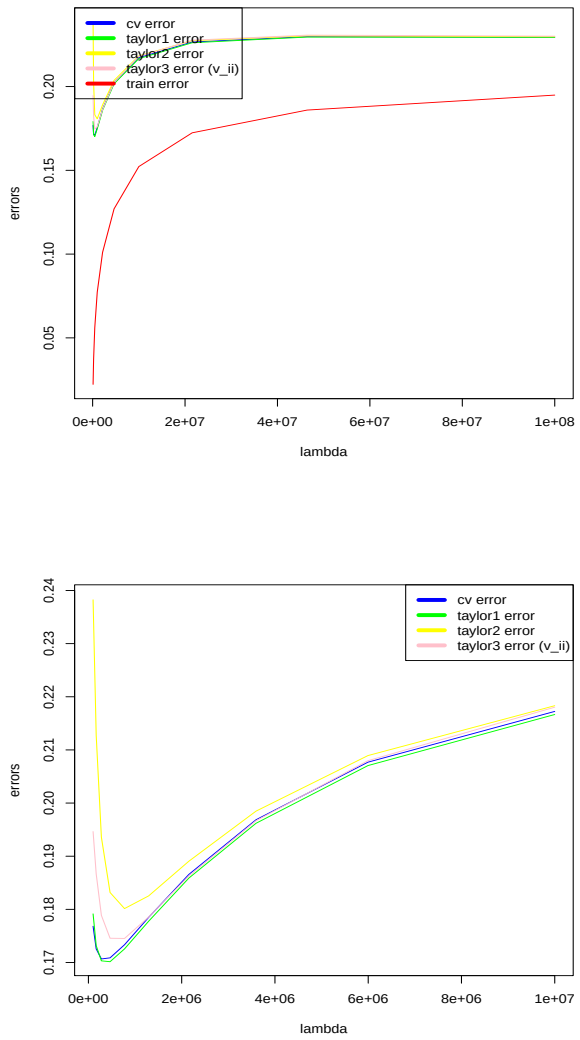


Figure 2: different approximations for the MSE_{CV}, based on first 100 covariates in the Golub data set.

It is clear that the green line (the MSE based on the approximated leave-one-out estimates) is preferable to the other ones. Note that the optimal λ found by either using the MSE_{CV} as done here, or the MML_{CV} as done in the previous section will be very similar.

5.5 Finding the optimal λ

It is not completely straightforward to find the derivative of $MML_{CV}(\lambda)$, since it is no explicit formula of λ. We know that $\hat{\boldsymbol{p}}$ depends on λ, since $\hat{\boldsymbol{\beta}}$ does, but this dependence is not explicitly defined because $\hat{\boldsymbol{\beta}}$ is found iteratively. Therefore, before we can find an expression for the derivative, some simplifications have to be made.
The (strong) assumption that will be made is that $\hat{\boldsymbol{\beta}}$ is only one Newton-Raphson step away from the real $\boldsymbol{\beta}$. Thus $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta} + (X'WX + 2\lambda I_p)^{-1}(X'(\boldsymbol{y} - \boldsymbol{p}) - 2\lambda\boldsymbol{\beta})$, where W and $\boldsymbol{p}$ are also based on the real $\boldsymbol{\beta}$. Since $\boldsymbol{\beta}$ does not depend on λ, we now have a created an explicit dependence between $\hat{\boldsymbol{\beta}}$ and λ. Now we have defined $\hat{\boldsymbol{\beta}}$ in this way, we can do something similar for $\hat{\boldsymbol{p}}$.

We have:

$$\hat{\boldsymbol{p}} = \frac{\exp(X\hat{\boldsymbol{\beta}})}{1 + \exp(X\hat{\boldsymbol{\beta}})}.$$

Let's denote this function by $f(\hat{\boldsymbol{\beta}})$. Making a first order Taylor expansion around the real $\boldsymbol{\beta}$ gives

$$f(\hat{\boldsymbol{\beta}}) = f(\boldsymbol{\beta}) + f'(\boldsymbol{\beta})(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) + O((\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^2)$$

The only term that still has to be calculated is $f'(\boldsymbol{\beta})$. If we introduce $\boldsymbol{\eta} = X\boldsymbol{\beta}$, we have

$$\frac{\partial f}{\partial \boldsymbol{\beta}} = \frac{\partial f}{\partial \boldsymbol{\eta}} \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\beta}}.$$

Of course, $\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\beta}}$ is just equal to X. The other derivative is given by:

$$\frac{\partial f}{\partial \boldsymbol{\eta}} = \frac{\exp(\boldsymbol{\eta})}{(1 + \exp(\boldsymbol{\eta}))^2} = \frac{\exp(\boldsymbol{\eta})}{1 + \exp(\boldsymbol{\eta})} \frac{1}{1 + \exp(\boldsymbol{\eta})} = \boldsymbol{p}(1 - \boldsymbol{p}).$$

Now we can write $\hat{\boldsymbol{p}}$ in terms of the real $\boldsymbol{p}$:

$$\hat{\boldsymbol{p}} = \boldsymbol{p} + WX(X'WX + 2\lambda I_p)^{-1}(X'(\boldsymbol{y} - \boldsymbol{p}) - 2\lambda\boldsymbol{\beta}).$$

If we use this expression for $\hat{\boldsymbol{p}}$ in the earlier derived formula for $MML_{CV}(\lambda)$ (and replace all $\hat{W}$'s by W) we get a formula that explicitly depends on λ.

The formula does now look like this:

$$\begin{aligned} &MML_{CV}(\lambda) \approx \\ &-n^{-1}\left(\boldsymbol{y}'\log(\hat{\boldsymbol{p}}) + (\boldsymbol{1} - \boldsymbol{y})'\log(\boldsymbol{1} - \hat{\boldsymbol{p}}) - (\boldsymbol{y} - \hat{\boldsymbol{p}})'(diag(I_n - V))^{-1}(diag(V))W^{-1}(\boldsymbol{y} - \hat{\boldsymbol{p}})\right), \end{aligned}$$

where all $\hat{\boldsymbol{p}}$'s are equal to $\boldsymbol{p} + WX(X'WX + 2\lambda I_p)^{-1}(X'(\boldsymbol{y} - \boldsymbol{p}) - 2\lambda\boldsymbol{\beta})$.

From this formula we can take the derivative.

First I will derive the derivatives of the separate pieces.

Since $\boldsymbol{y}$ and W do not depend on λ, $\frac{\partial \boldsymbol{y}'}{\partial \lambda}$, $\frac{\partial (1-\boldsymbol{y})'}{\partial \lambda}$ and $\frac{W^{-1}}{\partial \lambda}$ are all equal to zero.
The derivative of $\hat{\boldsymbol{p}}$ is given by:

$$\frac{\partial \hat{\boldsymbol{p}}}{\partial \lambda} = -2WX(X'WX + 2\lambda I_p)^{-2}(X'(\boldsymbol{y}-\boldsymbol{p}) - 2\lambda\boldsymbol{\beta}) - 2WX(X'WX + 2\lambda I_p)^{-1}\beta.$$

The derivative of $\log(\hat{\boldsymbol{p}})$ becomes

$$\frac{\partial \log(\hat{\boldsymbol{p}})}{\partial \lambda} = (diag(\hat{\boldsymbol{p}}))^{-1}\frac{\partial \hat{\boldsymbol{p}}}{\partial \lambda}.$$

The derivative of $\log(\mathbf{1} - \hat{\boldsymbol{p}})$ is almost the same:

$$\frac{\partial \log(\mathbf{1} - \hat{\boldsymbol{p}})}{\partial \lambda} = -(diag(\mathbf{1} - \hat{\boldsymbol{p}}))^{-1}\frac{\partial \hat{\boldsymbol{p}}}{\partial \lambda}.$$

The derivative of $(\boldsymbol{y} - \hat{\boldsymbol{p}})$ is given by:

$$\frac{\partial (\boldsymbol{y} - \hat{\boldsymbol{p}})}{\partial \lambda} = -\frac{\partial \hat{\boldsymbol{p}}}{\partial \lambda}$$

and since the derivative of the transpose is just the transpose of the derivative we have

$$\frac{\partial (\boldsymbol{y} - \hat{\boldsymbol{p}})'}{\partial \lambda} = -\left(\frac{\partial \hat{\boldsymbol{p}}}{\partial \lambda}\right)'.$$

The remaining terms depend on the derivative of V, which is given by:

$$\frac{\partial V}{\partial \lambda} = -2W^{\frac{1}{2}}X(X'\hat{W}X + 2\lambda I_p)^{-2}X'W^{\frac{1}{2}}.$$

Now the derivative of $diag(V)$ becomes

$$\frac{\partial diag(V)}{\partial \lambda} = diag(\frac{\partial V}{\partial \lambda})$$

and the derivative of $(diag(I_n - V))^{-1}$ is

$$\frac{\partial (diag(I_n - V))^{-1}}{\partial \lambda} = -(diag(I_n - V))^{-1}diag(-\frac{\partial V}{\partial \lambda})(diag(I_n - V))^{-1}$$

The derivative as a whole looks like this:

$$
\begin{aligned}
&\frac{\partial MML_{CV}(\lambda)}{\partial \lambda} = \\
- \quad & n^{-1}\Big(\boldsymbol{y}'(diag(\hat{\boldsymbol{p}}))^{-1}\frac{\partial \hat{\boldsymbol{p}}}{\partial \lambda} + (\mathbf{1}-\boldsymbol{y})' \cdot -(diag(\mathbf{1}-\hat{\boldsymbol{p}}))^{-1}\frac{\partial \hat{\boldsymbol{p}}}{\partial \lambda} \\
- \quad & \big(-(\frac{\partial \hat{\boldsymbol{p}}}{\partial \lambda})'(diag(I_n - V))^{-1}(diag(V))W^{-1}(\boldsymbol{y}-\hat{\boldsymbol{p}}) \\
+ \quad & (\boldsymbol{y}-\hat{\boldsymbol{p}})' \cdot -(diag(I_n - V))^{-1}diag(-\frac{\partial V}{\partial \lambda})(diag(I_n - V))^{-1}(diag(V))W^{-1}(\boldsymbol{y}-\hat{\boldsymbol{p}}) \\
+ \quad & (\boldsymbol{y}-\hat{\boldsymbol{p}})'(diag(I_n - V))^{-1}diag(\frac{\partial V}{\partial \lambda})W^{-1}(\boldsymbol{y}-\hat{\boldsymbol{p}}) \\
+ \quad & (\boldsymbol{y}-\hat{\boldsymbol{p}})'(diag(I_n - V))^{-1}(diag(V))W^{-1} \cdot -\frac{\partial \hat{\boldsymbol{p}}}{\partial \lambda}\big)\Big).
\end{aligned}
$$

Since we do not know the real $\boldsymbol{\beta}$ and $\boldsymbol{p}$ but we do know the values for $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{p}}$, we replace the real values by these estimates in order to get outcomes. (Note that we then get an expression like this: $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}} + (X'\hat{W}X + 2\lambda I_p)^{-1}(X'(\boldsymbol{y}-\hat{\boldsymbol{p}}) - 2\lambda\hat{\boldsymbol{\beta}})$ which is again $\hat{\boldsymbol{\beta}}$ since the Newton-Raphson algorithm converged and therefore $(X'\hat{W}X + 2\lambda I_p)^{-1}(X'(\boldsymbol{y}-\hat{\boldsymbol{p}}) - 2\lambda\hat{\boldsymbol{\beta}})$ equals zero.)

To check whether this formula looks like the real derivative of MML_{CV} I entered the actual formula in `R` (as can be seen in the "derivative"-function, given on the next page) and compared the values with those found by a different approximation formula, the "approximated-derivative"-formula. This formula is based on the values of MML_{CV} and calculates so called "Newton's difference quotients".

```
derivative <- function(X,Y,lambda)
{
    n <- nrow(X)
    m <- ncol(X)
    eye_n <- diag(rep(1,n))
    eye_m <- diag(rep(1,m))

    res <- VindBeta(X,Y,lambda)
    beta <- res$beta

    eta = as.vector(X %*% beta)
    p <- exp(eta)/(1+exp(eta))
    W <- diag(p*(1-p))
    neghessiaan <- crossprod(X, W %*% X)+2*lambda*diag(rep(1,m))
    V <- W^(1/2)%*%X%*%solve(neghessiaan)%*%t(X)%*%W^(1/2)
    invneghessiaan <- solve(neghessiaan)

    dpdl <- -2*W%*%X%*%invneghessiaan%*%invneghessiaan%*%(t(X)%*%(Y-p)
            -2*lambda*beta)-2*W%*%X%*%invneghessiaan%*%beta
    dvdl <- -2*W^(1/2)%*%X%*%invneghessiaan%*%invneghessiaan%*%t(X)%*%W^(1/2)
    div <- diag(diag(eye_n-V))
    invdiv <- solve(div)

    A <- t(Y)%*%solve(diag(p))%*%dpdl + t(1-Y)%*%-solve(diag(1-p))%*%dpdl
    B <- -t(dpdl)%*%invdiv%*%diag(diag(V))%*%solve(W)%*%(Y-p)
    C <- t(Y-p)%*%-invdiv%*%diag(diag(-dvdl))%*%invdiv%*%diag(diag(V))%*%solve(W)%*%(Y-p)
    D <- t(Y-p)%*%invdiv%*%diag(diag(dvdl))%*%solve(W)%*%(Y-p)
    E <- t(Y-p)%*%invdiv%*%diag(diag(V))%*%solve(W)%*%-dpdl

    result <- -1/n*(A-(B+C+D+E))

    return(result)
}

real_derivative = function(X,Y,lowerbound,upperbound,stepsize)
{
    lambda <- as.matrix(seq(from=lowerbound,to=upperbound,by=stepsize))
    realvalue <- as.matrix(seq(from=lowerbound,to=upperbound,by=stepsize))
    for(i in 1:nrow(lambda))
    {
      realvalue[i] <- derivative(X,Y,lambda[i])
    }

    return(list(realvalue=realvalue,lambda=lambda))
}
```

```
approximated_derivative = function(X,Y,lowerbound,upperbound,stepsize)
{
lambda <- as.matrix(seq(from=lowerbound,to=upperbound,by=stepsize))
appr <- as.matrix(seq(from=lowerbound,to=upperbound,by=stepsize))

#for every interval, calculate the approximated derivative

        for(i in 2:nrow(lambda))
        {
        MML1 <- MMLcvp(X,Y,lambda[i])   #real cross validated MMLcv
        MML2 <- MMLcvp(X,Y,lambda[i-1])
        appr[i] <- (MML1-MML2)/stepsize
        }

        return(list(appr=appr,lambda=lambda))
}
```

The graph shown here is based on a simulated data set:

```
X<-matrix(rnorm(1000,0,1),50,20)
eta <- X%*%c(1,1,3,0,0,2,0,0,0,0,0,5,7,1,1,0,0,0,0,1)
p <- exp(eta)/(1+exp(eta))
Y <- round(p)
res1 <- approximated_derivative(X,Y,0.5,5,0.05)
res2 <- real_derivative(X,Y,0.5,5,0.05)
allrange <- 1e-4 + range(res2$realvalue,res1$appr)
plot(res2$lambda, res2$realvalue, type="l", xlab = "lambda", ylab = "derivatives",
     ylim=allrange, col="green")
lines(res1$lambda[-1],res1$appr[-1], type="l",col="blue")
legend("topright", legend=c("derivative formula", "appr derivative"),
     col=c("green","blue"),lwd=5)
```

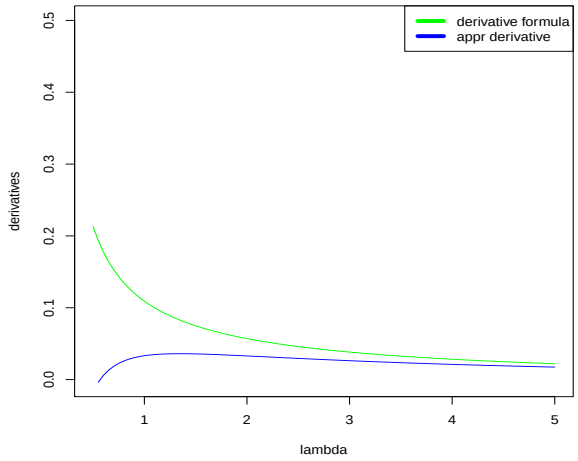


Figure 3: comparison of Newton's difference quotient and formula for the derivative

It is clear that the formula does not resemble the "actual" derivative at all. Again, this algebraic approach will not be continued.

6 k-fold cross-validation

In the previous sections, we were only considering leave-one-out cross-validation. Although the approximated version of leave-one-out cross-validation is much faster than actual LOOCV and for that reason there may be less need to use k-fold cross-validation, it is very likely that k-fold cross-validation will still be preferred by some researchers to (an approximated version of) leave-one-out cross-validation.

Despite the fact that k-fold cross-validation is less time-consuming than leave-one-out cross-validation, finding a penalty parameter by means of double k-fold cross-validation can be time consuming. (Especially when using complex statistical models in combination with large data sets.) A procedure to find an efficient approximation of $\hat{\boldsymbol{\beta}}_{-K}$ (similar to $\hat{\boldsymbol{\beta}}_{-i}$ but now based on all data points except for those in the k^{th} group, the capital K thus refers to a set of observations) can therefore be very valuable.

In finding approximations for the $\hat{\boldsymbol{\beta}}_{-i}$'s in previous sections, we always used the Sherman-Morrison-Woodbury theorem to find the inverse of the Hessian matrix belonging to the model with one observation left out, by using the Hessian matrix of the full model. If we would use the same approach for k-fold cross-validation as for leave-one-out cross-validation, we would either have to use this Sherman-Morisson-Woodbury theorem multiple times in a row or we could use the "matrix inversion theorem" which is a more general form of the Sherman-Morisson-Woodbury theorem. The matrix inversion theorem (as given in [6]) says the following:

Theorem 6.0.1. *Matrix Inversion theorem*
If both $\boldsymbol{A}$ *and* $\boldsymbol{I} - \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U}$ *are invertible, then* $\boldsymbol{A} - \boldsymbol{U}\boldsymbol{V}$ *is invertible and*

$$(\boldsymbol{A} - \boldsymbol{U}\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{I} - \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1}.$$

In the linear model, we know that $\hat{\boldsymbol{\beta}}_{-K}$ is given by $(\boldsymbol{X}'_{-K}\boldsymbol{X}_{-K})^{-1}\boldsymbol{X}'_{-K}\boldsymbol{y}_{-K}$. We would like to find an expression for $\hat{\boldsymbol{\beta}}_{-K}$ in terms of $\hat{\boldsymbol{\beta}}$.

Starting from the equality

$$\boldsymbol{X}'\boldsymbol{X} = \sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{x}'_i,$$

we derive

$$\boldsymbol{X}'_{-K}\boldsymbol{X}_{-K} = \boldsymbol{X}'\boldsymbol{X} - \boldsymbol{X}'_K\boldsymbol{X}_K$$

where the matrix $\boldsymbol{X}_K$ is the matrix which has the covariate vectors $\boldsymbol{x}_i$ from the k^{th} group as its rows. If we now substitute $\boldsymbol{A} = \boldsymbol{X}'\boldsymbol{X}$, $\boldsymbol{V} = \boldsymbol{X}_K$ and $\boldsymbol{U} = \boldsymbol{X}'_K$ in the matrix inversion theorem, we get

$$\hat{\boldsymbol{\beta}}_{-K} = \left((\boldsymbol{X}'\boldsymbol{X})^{-1} + (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K(\boldsymbol{I} - \boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K)^{-1}\boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\right)\boldsymbol{X}'_{-K}\boldsymbol{y}_{-K}$$

which equals

$$\hat{\boldsymbol{\beta}}-(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K\boldsymbol{y}_K+(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K(\boldsymbol{I}-\boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K)^{-1}\boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_{-K}\boldsymbol{y}_{-K}.$$

If we put in an extra identity matrix of the following form:
$(\boldsymbol{I}-\boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K)^{-1}(\boldsymbol{I}-\boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K)$ and rewrite the above equation we get:

$$\hat{\boldsymbol{\beta}}-(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K(\boldsymbol{I}-\boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K)^{-1}\left((\boldsymbol{I}-\boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K)\boldsymbol{y}_K-\boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_{-K}\boldsymbol{y}_{-K}\right),$$

which is just

$$\hat{\boldsymbol{\beta}}-(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K(\boldsymbol{I}-\boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K)^{-1}\left(\boldsymbol{y}_K-\boldsymbol{X}_K\hat{\boldsymbol{\beta}}\right).$$

(Note that $\boldsymbol{y}_K$ is a vector, containing the y_i's belonging to the k^{th} group and is not just the k^{th} element of $\boldsymbol{y}$.) So, just as we found an expression for $\hat{\boldsymbol{\beta}}_{-i}$ in terms of $\hat{\boldsymbol{\beta}}$ we know found $\hat{\boldsymbol{\beta}}_{-K}$ in terms of $\hat{\boldsymbol{\beta}}$:

$$\hat{\boldsymbol{\beta}}_{-K}=\hat{\boldsymbol{\beta}}-(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K(\boldsymbol{I}-\boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K)^{-1}\left(\boldsymbol{y}_K-\boldsymbol{X}_K\hat{\boldsymbol{\beta}}\right). \quad (20)$$

Usually, we have to calculate $(\boldsymbol{X}'_{-K}\boldsymbol{X}_{-K})^{-1}$ k times, so the inverse of an $p\times p$ matrix. With this new formula, the inverse that has to be calculated multiple times is $(\boldsymbol{I}-\boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K)^{-1}$, which is an $l\times l$ matrix (with l the number of individuals in the the k^{th} group). This new method will therefore be especially useful when l is a lot smaller than p.

Earlier, we found a simple expression for the leave-one-out cross-validated residual sum of squares. The question is if the same can be done for the k-fold cross-validated RSS. This time the RSS looks like

$$RSS_{kfold}=\sum_K(\boldsymbol{y}_K-\boldsymbol{X}_K\hat{\boldsymbol{\beta}}_{-K})^T(\boldsymbol{y}_K-\boldsymbol{X}_K\hat{\boldsymbol{\beta}}_{-K}). \quad (21)$$

When we substitute (20) into this formula and simplify the expression, we get

$$RSS_{kfold}=\sum_K||(\boldsymbol{I}-\boldsymbol{X}_K(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'_K)^{-1}(\boldsymbol{y}_K-\boldsymbol{X}_K\hat{\boldsymbol{\beta}})||^2. \quad (22)$$

To find an approximation for $\hat{\boldsymbol{\beta}}_{-K}$ in a logistic model, we will again use a Taylor approximation.
In approximating the $\hat{\boldsymbol{\beta}}_{-i}$'s, we made a first order Taylor approximation of the derivative of the log-likelihood which resulted in:

$$\hat{\boldsymbol{\beta}}_{-i} = \hat{\boldsymbol{\beta}} - (l''_{(-i)}(\hat{\boldsymbol{\beta}}))^{-1} l'_{(-i)}(\hat{\boldsymbol{\beta}}).$$

By assuming that $\hat{\boldsymbol{\beta}}_{-K}$ is still quite similar to $\hat{\boldsymbol{\beta}}$, we could by the same reasoning come to the following approximation:

$$\hat{\boldsymbol{\beta}}_{-K} = \hat{\boldsymbol{\beta}} - (l''_{(-k)}(\hat{\boldsymbol{\beta}}))^{-1} l'_{(-k)}(\hat{\boldsymbol{\beta}}),$$

where

$$l''_{(-k)}(\hat{\boldsymbol{\beta}})^{-1} = \boldsymbol{X}'_{-K} \boldsymbol{W}_{-K} \boldsymbol{X}_{-K}$$

and

$$l'_{(-k)}(\hat{\boldsymbol{\beta}}) = \boldsymbol{X}'_{(-k)}(\boldsymbol{y}_{-K} - \boldsymbol{p}_{-K}).$$

The values of $\boldsymbol{W}_{-K}$ and $\boldsymbol{p}_{-K}$ are based on $\hat{\boldsymbol{\beta}}$.

The hessian matrix based on all but the k^{th} group can be rewritten into

$$\boldsymbol{X}'_{-K} \boldsymbol{W}_{-K} \boldsymbol{X}_{-K} = \boldsymbol{X}' \boldsymbol{W} \boldsymbol{X} - \boldsymbol{X}'_K (\boldsymbol{W} \boldsymbol{X})_K.$$

From this expression we see that the matrix inversion theorem can again be used. This time we take $\boldsymbol{A} = \boldsymbol{X}' \boldsymbol{W} \boldsymbol{X}$, $\boldsymbol{V} = (\boldsymbol{W} \boldsymbol{X})_K$ and $\boldsymbol{U} = \boldsymbol{X}'_K$.

After some simplifications, the final formula becomes:

$$\hat{\boldsymbol{\beta}}_{-K} \approx \hat{\boldsymbol{\beta}} - (\boldsymbol{X}' \boldsymbol{W} \boldsymbol{X})^{-1} \boldsymbol{X}'_K (\boldsymbol{I} - (\boldsymbol{W} \boldsymbol{X})_K (\boldsymbol{X}' \boldsymbol{W} \boldsymbol{X})^{-1} \boldsymbol{X}'_K)^{-1} (\boldsymbol{y}_K - \boldsymbol{p}_K). \quad (23)$$

Extending this formula to logistic ridge regression and survival regression will be straightforward.

A different idea is to add multiple indicator variables to the model. In [11] the suggestion is made to use one indicator variable in order to find an approximation for the leave-one-out regression coefficients, but this idea can be extended to regression coefficients corresponding to k-fold cross-validation, by adding l indicator variables, where l is the number of samples in a group (in case of an equal distribution over the groups, l will be n/k).
For leave-one-out regression, the old model $\boldsymbol{X}\boldsymbol{\beta}$ is replaced by $\boldsymbol{X}\boldsymbol{\beta}^* + \boldsymbol{z}_i\gamma$, where $\boldsymbol{z}_i$ is an indicator variable for individual i. Because the dependent variable y_i can be estimated perfectly just based on the appropriate value of γ, the estimate for $\boldsymbol{\beta}^*$ will not depend on the i^{th} observation and the maximum likelihood estimate of $\boldsymbol{\beta}^*$ will (at least in models with independent components) be $\hat{\boldsymbol{\beta}}_{-i}$. To save time, we will again only take one Newton-Raphson step, with $\hat{\boldsymbol{\beta}}$ and 0 as initial values for $\boldsymbol{\beta}^*$ and γ respectively. Extending this model to k-fold cross-validation can be done by just adding more indicator variables. When there is an extra penalty term in the model, it is important that the new regression coefficients (the ones corresponding to the indicator variables) should be unpenalized.
In case of leave-one-out regression, this "indicator-method" will result in exactly the same approximations as we found by using a first order Taylor approximation in combination with the Sherman-Morrison-Woodbury theorem.
To proof this (in the logistic regression context), we use the following theorem on partitioned matrices, as given in [8]:

Theorem 6.0.2. *Let A be an $m \times n$ matrix of the following form:*

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

where A_{11} is $m_1 \times n_1$, A_{12} is $m_1 \times n_2$, A_{21} is $m_2 \times n_1$, A_{22} is $m_2 \times n_2$ and $m_1 + m_2 = m, n_1 + n_2 = n$.

If A is non-singular and $D = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is also non-singular, then

$$A^{-1} = \begin{pmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}D^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}D^{-1} \\ -D^{-1}A_{21}A_{11}^{-1} & D^{-1} \end{pmatrix}.$$

The new design matrix $\boldsymbol{X}^*$ is just the old design matrix with an extra column added:

$$\boldsymbol{X}^* = \begin{pmatrix} x_{11} & \dots & x_{1p} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ x_{i1} & \dots & x_{ip} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{np} & 0 \end{pmatrix}$$

The approximation is now given by the first p components of:

$$\begin{pmatrix} \hat{\boldsymbol{\beta}} \\ 0 \end{pmatrix} - (\boldsymbol{X}^{*T}\boldsymbol{W}\boldsymbol{X}^*)^{-1}\boldsymbol{X}^{*T}(\boldsymbol{y} - \boldsymbol{p}),$$

where $\boldsymbol{p}$ and $\boldsymbol{W}$ are based on $\hat{\boldsymbol{\beta}}$.
We know that

$$\boldsymbol{X}^{*T}(\boldsymbol{y}-\boldsymbol{p}) = \begin{pmatrix} \boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{p}) \\ y_i - p_i \end{pmatrix}$$

and

$$(\boldsymbol{X}^{*T}\boldsymbol{W}\boldsymbol{X}^*) = \begin{pmatrix} \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X} & w_{ii}\boldsymbol{x}_i \\ w_{ii}\boldsymbol{x}_i^T & w_{ii} \end{pmatrix}.$$

We can now use the above stated theorem to calculate the inverse of $\boldsymbol{X}^{*T}\boldsymbol{W}\boldsymbol{X}^*$:

$$(\boldsymbol{X}^{*T}\boldsymbol{W}\boldsymbol{X}^*)^{-1} = \begin{pmatrix} (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1} + (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}w_{ii}\boldsymbol{x}_i D^{-1} w_{ii}\boldsymbol{x}_i^T(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1} & -(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}w_{ii}\boldsymbol{x}_i D^{-1} \\ -D^{-1}w_{ii}\boldsymbol{x}_i^T(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1} & D^{-1} \end{pmatrix},$$

with $D = (w_{ii} - w_{ii}\boldsymbol{x}_i^T(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}w_{ii}\boldsymbol{x}_i)$. Since D is just a scalar, $D^{-1} = \frac{1}{D}$.

Since we are only interested in the first p coefficients of the new parameter vector, we can forget about the last row of this inverse and we find as an approximate for $\hat{\boldsymbol{\beta}}_{-i}$:

$$\left((\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1} + (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}w_{ii}\boldsymbol{x}_i D^{-1} w_{ii}\boldsymbol{x}_i^T(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\right)\boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{p}) - (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}w_{ii}\boldsymbol{x}_i D^{-1}(y_i - p_i).$$

Since an earlier Newton-Raphson procedure converged to $\hat{\boldsymbol{\beta}}$ we now that $\boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{p})$ equals 0 and the above expression reduces to

$$(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}w_{ii}\boldsymbol{x}_i D^{-1}(y_i - p_i)$$

So, we found:

$$\hat{\boldsymbol{\beta}}_{-i} \approx \frac{(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}w_{ii}\boldsymbol{x}_i(y_i - p_i)}{w_{ii} - w_{ii}\boldsymbol{x}_i^T(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}w_{ii}\boldsymbol{x}_i},$$

which is equal to

$$\frac{(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{x}_i(y_i - p_i)}{1 - w_{ii}\boldsymbol{x}_i^T(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{x}_i},$$

which is exactly the approximation that we found earlier (equation (15)).

For k-fold cross-validation, it should be possible to prove in the same way that adding l indicator variables and taking one Newton-Raphson step will result in the same approximation as a first order Taylor expansion in combination with the matrix inversion theorem.
I did not prove this equality formally, but by running simulations (with a logistic ridge regression model) I could see that the answers were indeed exactly the same.

To test the usefulness of the derived approximation method, on several simulated data sets the mean-minus log-likelihood is calculated based on either the real values of the k-fold estimates or the approximated ones. The graph below is based on the following dataset and 10-fold cross-validation is used.

```
X<-matrix(rnorm(15000,0,1),500,30)
eta <- X%*%c(1,1,3,0,0,2,0,0,0,0,0,5,7,1,1,0,0,0,0,1,5,6,1,0,0,0,2,4,5,1) + rnorm(500,3,2)
p <- exp(eta)/(1+exp(eta))
Y <- round(p)
```

The purple line represents the approximation based on the matrix inversion theorem. The green line represented the approximation based on the indicator-method but overlaps completely with the purple curve. The real optimum (so the one of the blue curve) lies at $\lambda = 0.76$ and the optimum of the approximated curve lies at $\lambda = 0.68$. Of course, the approximation is not perfect but it still seems to do well.

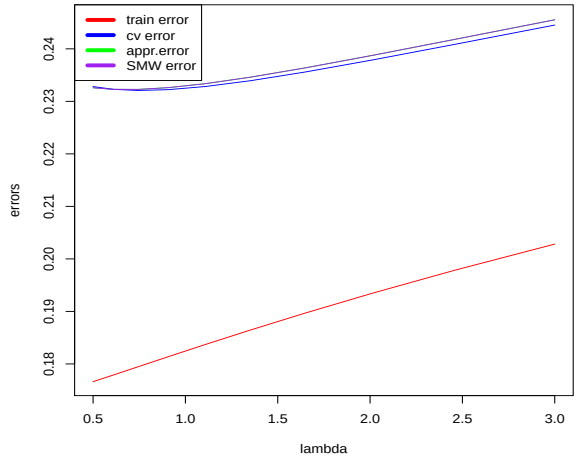


Figure 4: real 10-fold cross-validation vs. approximated 10-fold cross-validation

Even for 5-fold cross-validation the graph still looks nice:

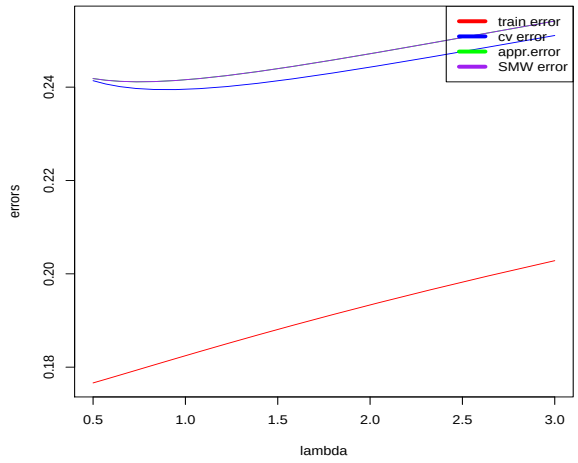


Figure 5: real 5-fold cross-validation vs. approximated 5-fold cross-validation

7 Survival analysis

After describing a method to find an approximation for $\hat{\boldsymbol{\beta}}_{(-i)}$ in linear and logistic (ridge) regression, it would be really useful to find a way to extend this idea to Cox regression. Since fitting a Cox proportional hazards model is usually more difficult than fitting a logistic model, the amount of time that can be saved in this way will probably even be more substantial.

In survival analysis, we have data of the following form: (t_i, d_i, X_i). Here the (censored) survival time is denoted by t_i, the censoring indicator is given by d_i (with $d_i = 0$ in case of censoring) and X_i is a p-dimensional vector of covariates for the i^{th} individual. In the Cox proportional hazards model ([3]), the hazard function for individual i depends on a common baseline hazard, denoted by h_0 and the covariates of this person in the following way:

$$h_i(t) = h_0(t) \exp(\beta' X_i).$$

To find an estimate for the unknown vector of regression coefficients, two strategies can be adopted. The first one uses the partial (log-)likelihood, the second one the full (log-)likelihood. Maximizing the partial likelihood has the advantage that we do not have to take the baseline hazard into account. However, the information matrix (i.e. the negative of the matrix of second derivatives) that comes with this partial likelihood is of the form $X'WX$, where W is given by

$$diag(e_1, ..., e_n) - PP',$$

where

$$e_i = \sum_{\tau_j \leq t_i} \frac{\exp(X_i\beta)}{\sum_{k \in R_j} \exp(X_k\beta)}$$

and

$$p_{ij} = I[t_i \geq \tau_j] \frac{\exp(X_i\beta)}{\sum_{k \in R_j} \exp(X_k\beta)},$$

as given in [13].
The W-matrix is, unlike the one in the logistic regression model, no longer a diagonal matrix. In previous derivations, we used the following equality:

$$X'WX = \sum_{i=1}^{n} w_{ii} X_i X_i',$$

which does not hold anymore. We used to write $(X'_{(-i)} W_{(-i)} X_{(-i)})^{-1} = (X'WX - w_{ii} X_i X_i')^{-1}$ in order to apply the Sherman-Morrison-Woodbury theorem, but now we have to find a different derivation. However, if we would use the full likelihood instead of the partial likelihood, we would not run into this problem, as will be shown next.

The full log-likelihood of the proportional hazards model is given by:

$$l(\beta, h_0) = \sum_{i=1}^{n} \left(-\exp(X_i'\beta)H_0(t_i) + d_i \left(\ln(h_0(t_i)) + X_i'\beta\right)\right),$$

with

$$H_0(t) = \sum_{s \leq t} h_o(s),$$

as can be found for example in [7].

A penalization term can be added to this likelihood. In ridge regression, we get:

$$l_{pen}(\beta, h_0) = l(\beta, h_0) - 0.5\lambda\beta'\beta.$$

The first and second derivative with respect to β are now given by:

$$\frac{\partial l_{pen}(\beta, h_0)}{\partial \beta} = X'\Delta - \lambda\beta,$$

with

$$\Delta_i = d_i - H_0(t_i)\exp(X_i'\beta)$$

and

$$\frac{\partial^2 l_{pen}(\beta, h_0)}{\partial \beta^2} = -(X'DX + \lambda I_p),$$

where

$$D = diag(\exp(X_i'\beta)H_0(t_i)).$$

The same notation is used as in [7].

We see that D (equivalent to W in the logistic regression model) is again a diagonal matrix. Therefore, we can apply the same techniques as before.

We still denote the log-likelihood where the i^{th} observation is left out by $l_{(-i)}(\beta)$ and after constructing a first order Taylor approximation of $l'_{(-i)}(\beta)$ around $\hat{\beta}$ we find:

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} - (l''_{(-i)}(\hat{\boldsymbol{\beta}}))^{-1} l'_{(-i)}(\hat{\boldsymbol{\beta}}).$$

If we put in the formulas for the gradient and the hessian we just found, we get:

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} + (X'_{(-i)}D_{(-i)}X_{(-i)} + \lambda I_p)^{-1}(X'_{(-i)}\Delta_{(-i)} - \lambda\hat{\boldsymbol{\beta}}),$$

where the values of the $\Delta_{(-i)}$ and $D_{(-i)}$ are calculated based on the value for $\hat{\boldsymbol{\beta}}$. To estimate the values of $H_0(t_i)$, we use the Breslow estimator:

$$\hat{h_0}(t_i) = 1/\sum_{t_j \geq t_i} \exp(X_j'\hat{\boldsymbol{\beta}}).$$

Note that it does not matter in what way the actual value of $\hat{\boldsymbol{\beta}}$ is computed, since maximizing the partial or the full-likelihood will give the same results. That we use the full log-likelihood in the derivation does not mean that this same likelihood has to be used in the calculation of the maximum penalized likelihood estimator $\hat{\boldsymbol{\beta}}$.

Now, the Sherman-Morrison-Woodbury theorem can be applied:

$$\begin{aligned}(X'_{(-i)}D_{(-i)}X_{(-i)} + \lambda I_p)^{-1} &= (X'DX + \lambda I_p - d_{ii}X_iX_i')^{-1} \\ &= (X'DX + \lambda I_p)^{-1} + \frac{(X'DX + \lambda I_p)^{-1}d_{ii}X_iX_i'(X'DX + \lambda I_p)^{-1}}{1 - d_{ii}X_i'(X'DX + \lambda I_p)^{-1}X_i}.\end{aligned}$$

Furthermore, we rewrite $(X'_{(-i)}\Delta_{(-i)}$ as $X\Delta - X_i\Delta_i$. We can introduce similar variables as before: $Z = D^{\frac{1}{2}}X$ and $v = D^{-\frac{1}{2}}\Delta$. This gives us:

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} + \left((Z'Z + \lambda I_p)^{-1} + \frac{(Z'Z + \lambda I_p)^{-1}\boldsymbol{z}_i\boldsymbol{z}_i'(Z'Z + \lambda I_p)^{-1}}{1 - \boldsymbol{z}_i'(Z'Z + \lambda I_p)^{-1}\boldsymbol{z}_i}\right) \left(Z'v - \boldsymbol{z}_iv_i - \lambda\hat{\boldsymbol{\beta}}\right).$$

Because the Newton Raphson algorithm converged to $\hat{\boldsymbol{\beta}}$, we know that $(X'DX + \lambda I_p)^{-1}(X'\Delta - \lambda\hat{\boldsymbol{\beta}}) = (Z'Z + \lambda I_p)^{-1}(Z'v - \lambda\hat{\boldsymbol{\beta}})$ equals 0, when D and Δ are calculated based on $\hat{\boldsymbol{\beta}}$. Now we have exactly the same situation as we had in the case of logistic ridge regression. Therefore, we can directly jump to the conclusions of the corresponding derivation, which gives us:

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} - \frac{(Z'Z + \lambda I_p)^{-1}\boldsymbol{z}_iv_i}{1 - \boldsymbol{z}_i'(Z'Z + \lambda I_p)^{-1}\boldsymbol{z}_i}.$$

In the original notation, we get:

$$\hat{\boldsymbol{\beta}}_{(-i)} = \hat{\boldsymbol{\beta}} - \frac{(X'DX + \lambda I_p)^{-1}X_i\Delta_i}{1 - d_{ii}X_i'(X'DX + \lambda I_p)^{-1}X_i}. \tag{24}$$

The term $d_{ii}X_i'(X'DX + \lambda I_p)^{-1}X_i$ is the i^{th} diagonal element of the matrix $D^{\frac{1}{2}}X(X'DX + \lambda I_p)^{-1}X'D^{\frac{1}{2}}$, which we will call V from now on.

A computational problem may arise when the number of covariates is really large. The matrix $(X'DX + \lambda I_p)$ is a $p \times p$ matrix and finding its inverse will take a lot of computations and computer memory when p is big. To overcome this problem, we can rewrite the equations in terms of γ.
Because we know that (if the Newton Raphson algorithm converged) $X'\Delta - \lambda\beta = 0$, we see that β lies in the column space of the design matrix X' (see also [7]). For that reason, we can write $\beta = X'\gamma$, for some γ. If we rewrite the likelihood function in terms of γ, we get:

$$l(\gamma, h_0) = \sum_{i=1}^{n}\left(-\exp(X_i'X'\gamma)H_0(t_i) + d_i\left(\ln(h_0(t_i)) + X_i'X'\gamma\right)\right) - 0.5\lambda\gamma'XX'\gamma.$$

If we rename the matrix XX' into A, we get:

$$l(\gamma, h_0) = \sum_{i=1}^{n} (-\exp(A_i'\gamma)H_0(t_i) + d_i\,(\ln(h_0(t_i)) + A_i'\gamma)) - 0.5\lambda\gamma' A\gamma,$$

for which the corresponding first and second derivative with respect to γ are given by:

$$\frac{\partial l_{pen}(\gamma, h_0)}{\partial \gamma} = A'\Delta - \lambda A\gamma,$$

with

$$\Delta_i = d_i - H_0(t_i)\exp(A_i'\gamma)$$

and

$$\frac{\partial^2 l_{pen}(\gamma, h_0)}{\partial \gamma^2} = -(A'DA + \lambda A),$$

where

$$D = diag(\exp(A_i'\gamma)H_0(t_i)).$$

Using these expressions, together with the Sherman-Morrison-Woodbury theorem, gives as final result:

$$\hat{\boldsymbol{\gamma}}_{(-i)} = \hat{\boldsymbol{\gamma}} - \frac{(A'DA + \lambda A)^{-1} a_i \Delta_i}{1 - v_{ii}}, \tag{25}$$

where v_{ii} is the i^{th} diagonal element of the matrix V given by $D^{\frac{1}{2}}A(A'DA + \lambda A)^{-1}A'D^{\frac{1}{2}}$.
(A more detailed explanation can be found in subsection 6.2: "Computational problems", where the same strategy is already used in the logistic ridge regression case.)

Starting with $\hat{\boldsymbol{\gamma}}$ we have now a way to find $\hat{\boldsymbol{\gamma}}_{(-i)}$ and $\hat{\boldsymbol{\beta}}_{(-i)}$ can now be easily obtained since $\hat{\boldsymbol{\beta}}_{(-i)} = X'\hat{\boldsymbol{\gamma}}_{(-i)}$.
Sometimes, it is also useful to find $\hat{\boldsymbol{\gamma}}$ from a known $\hat{\boldsymbol{\beta}}$ (which is for example found by the R package "penalized"). This can be done in the following way: $\hat{\boldsymbol{\gamma}} = (XX')^{-1}X\hat{\boldsymbol{\beta}}$.

In linear regression we used the cross-validated sum of squares as a performance measure and in logistic regression we used cross-validated mean minus log-likelihood or the cross-validated mean squared error. The question is what measure we can use in survival analysis.

Because of the censoring, it is impossible to define a measure which is comparable to the sum of squares, but we can use a cross-validated likelihood. In the full likelihood we could simply fill in all the values of the $\hat{\boldsymbol{\beta}}_{(-i)}$'s (for a given value of λ), but the problem lies in the construction of the estimated baseline hazard. The baseline hazard only differs from zero in observed event points and therefore equals zero for all new observations. In this case, the i^{th} observation is treated as a new observation and for that reason, the cross-validated full (log-)likelihood can not be used.
Since the partial likelihood does not depend on the baseline hazard, the cross-validated partial likelihood can be used. The cross-validated partial log-likelihood, as derived in [11] looks like this:

$$cvpl(\lambda) = \sum_{i=1}^{n} pl_i(\hat{\boldsymbol{\beta}}^{\lambda}_{(-i)}).$$

Since the partial likelihood is not defined for a single observation, the term $pl_i(\hat{\boldsymbol{\beta}}^{\lambda}_{(-i)})$, which gives the contribution of individual i to the partial log-likelihood, is a difference between the contribution of all n observations and the contribution of $n-1$ observations:

$$pl_i(\hat{\boldsymbol{\beta}}^{\lambda}_{(-i)}) = pl(\hat{\boldsymbol{\beta}}^{\lambda}_{(-i)}) - pl_{(-i)}(\hat{\boldsymbol{\beta}}^{\lambda}_{(-i)}).$$

In [11] it is shown that the cross-validated partial log-likelihood can be written as

$$cvpl(\lambda) = \sum_{i=1}^{n} \left(\sum_{t_j < t_i} d_j \ln(1 - p_{ij}) + d_i \ln(p_{ii}) \right),$$

with

$$p_{ij} = \frac{\exp(X_i' \hat{\boldsymbol{\beta}}^{\lambda}_{(-i)})}{\sum\limits_{t_k \geq t_j} \exp(X_k' \hat{\boldsymbol{\beta}}^{\lambda}_{(-i)})}$$

The just described *cvpl* is derived under the assumption that no ties are present in the data set, so all time points are distinct. Unfortunately, when we use some real data set it can of course happen that there are ties. I adjusted the *cvpl* in order to work with ties, following the same reasoning from which the above *cvpl* resulted.
The final formula only changed in one specific location, namely the index of the inner summation, which changed from $t_j < t_i$ to $t_j \leq t_i, i \neq j$.

To decide whether the approximations of the $\hat{\boldsymbol{\beta}}^{\lambda}_{(-i)}$'s are useful in practice, the *cvpl* based on the real leave-one-out estimates (as calculated by the R-package "penalized") can now be compared to the *cvpl* based on the approximated leave-one-out estimates.

7.1 Adjustments

In the actual simulations, as given in the article, we did change the approximation procedure a little. In the just described approximation procedure, the approximation for $\hat{\boldsymbol{\beta}}_{(-i)}$ depends on the baseline hazard of the full model (this baseline hazard is incorporated in the matrix D and the vector Δ). Of course the actual baseline hazard in the model with one observation left out will differ (a little) from the one belonging to the full model. Unfortunately, there is not much that can be done about this, since all the information we have is based on the full model, but in order to allow for multiplicative changes we can incorporate an extra intercept term in the model. Normally, a Cox regression model is fitted without an intercept term since this term is "absorbed" in the baseline hazard, but this time the extra intercept term may result in a baseline hazard that is closer to the real leave-one-out baseline hazard.
The approximation formula, as given in equation (24) can still be used, but a few changes have to be made:
The matrix X has to be replaced by the new matrix X_{int} which has an extra column:

$$X_{int} = \begin{pmatrix} 1 & \\ \vdots & X \\ 1 & \end{pmatrix},$$

the identity matrix I_p is replaced by a new $p+1 \times p+1$ diagonal matrix which has a 0 as its first element to make sure that the new intercept term is not penalized

$$I_{int} = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & I_p & \\ 0 & & \end{pmatrix}.$$ and finally, the initial β, which used to be $\hat{\boldsymbol{\beta}}$ is now given by

$$\hat{\boldsymbol{\beta}}^{\lambda}_{int} = \begin{pmatrix} 0 \\ \hat{\boldsymbol{\beta}}^{\lambda} \end{pmatrix}.$$

The $\hat{\boldsymbol{\beta}}_{(-i)}$ we find in this way is still a vector of length $p+1$, but we are only interested in the last p regression coefficients.

If we again want to reduce the dimension of the covariate space by the γ-method, we have to be careful. Before, we used the fact that, if the Newton-Raphson algorithm had converged, $X'\Delta - \lambda\beta = 0$. For the regression coefficients not corresponding to the intercept term, this equation still holds, but for the added coefficient, we can not formulate an expression like this, However, we can use the reparametrization matrix G of the form:

$$G = \begin{pmatrix} 1 & \dots & 0 \\ \vdots & X & \\ 0 & & \end{pmatrix},$$

and now it is clear that $\boldsymbol{\beta} = G^T\boldsymbol{\gamma}$.
As before, we want to find an expression for $\boldsymbol{\gamma}_{-i}$ in terms of $\boldsymbol{\gamma}$. The first remark we should make is that we can use the same reparametrization matrix in case of $\boldsymbol{\gamma}_{-i}$, because β_{-i} lies in the column space of X_{-i}^T and for that reason also in the column space of X^T.
In the first and second derivative, given by:

$$\frac{\partial l^\lambda(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}} = GX_{int}^T\Delta - \lambda A\boldsymbol{\gamma} \tag{26}$$

and

$$\frac{\partial^2 l^\lambda(\boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}\partial \boldsymbol{\gamma}^T} = -GX_{int}^T D X_{int} G^T - \lambda A, \tag{27}$$

where $A = GI_{int}G^T$ we thus only have to remove individual i from X_{int}, D and Δ but not from G. Therefore, the leave-one-out hessian can be written as:

$$-(G(X_{int}^T D X_{int} - \boldsymbol{x}_i d_{ii} \boldsymbol{x}_i^T)G^T + \lambda A) = -(GX_{int}^T D X_{int} G^T - G\boldsymbol{x}_i d_{ii} \boldsymbol{x}_i^T G^T + \lambda A).$$

If we now introduce a new matrix $B = X_{int}G^T$ and notice that $G\boldsymbol{x}_i$ is just the i^{th} row of this matrix, above expression is just

$$-\left(B^T D B - \boldsymbol{b}_i d_{ii} b_i^T + \lambda A\right)$$

and to find the inverse of this matrix, we can again apply the Sherman-Morrison-Woodbury theorem. The final formula can be found in the article.

8 Lasso regression

Until now, the focus lay on models with a ridge penalty. In this section a different penalty that is used often, the lasso penalty, will be discussed. This penalty will be discussed in combination with a survival model, but the final formulas can easily be adjusted to work with generalized linear models as well.

In lasso regression, the penalized likelihood looks like:

$$l_{pen}(\beta, h_0) = l(\beta, h_0) - \lambda \sum_{i=1}^{p} |\beta_i|.$$

To come up with the approximation formula for the leave-one-out estimate $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$, we again use the earlier derived formula

$$\hat{\boldsymbol{\beta}}^{\lambda}_{(-i)} = \hat{\boldsymbol{\beta}}^{\lambda} - (l''_{(-i)}(\hat{\boldsymbol{\beta}})^{\lambda})^{-1} l'_{(-i)}(\hat{\boldsymbol{\beta}}^{\lambda}).$$

As long as the all β_i's are unequal to zero, the first and second derivative do exist and are given by

$$\frac{\partial l_{pen}(\beta, h_0)}{\partial \beta} = X'\Delta - \lambda \sum_{i=1}^{p} sign(\beta_i),$$

with

$$\Delta_i = d_i - H_0(t_i) \exp(X_i'\beta)$$

and

$$\frac{\partial^2 l_{pen}(\beta, h_0)}{\partial \beta^2} = -(X'DX),$$

where

$$D = diag(\exp(X_i'\beta) H_0(t_i)).$$

The first approach is to look only at those coefficients of $\hat{\boldsymbol{\beta}}^{\lambda}$ that are unequal to zero (the coefficients belonging to the active set $\mathcal{A}$), to alter these values in order to come up with an approximate for $\hat{\boldsymbol{\beta}}_{-i}$ and assume that the zeros in $\hat{\boldsymbol{\beta}}^{\lambda}$ will still be zero in $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$. Although the active set might change a little after eliminating one observation, the lasso is designed to be a stable (see [10]) and therefore the changes will be small which justifies the assumption.

If we denote the design matrix corresponding to the active set by $X_{\mathcal{A}}$, we have (for the nonzero elements):

$$\hat{\boldsymbol{\beta}}^{\lambda}_{(-i)} \approx \hat{\boldsymbol{\beta}}^{\lambda} + ((X'_{\mathcal{A}} D X_{\mathcal{A}} - \boldsymbol{x}_{\mathcal{A}i} d_{ii} \boldsymbol{x}'_{\mathcal{A}i})^{-1} \left(X'_{\mathcal{A}}\Delta - \boldsymbol{x}_{\mathcal{A}i}\Delta_i - \lambda \sum_{i \in \mathcal{A}} sign(\beta_i) \right).$$

Using the Sherman-Morrison-Woodbury theorem and the fact that the Newton-Raphson algorithm converged to $\hat{\boldsymbol{\beta}}^{\lambda}$ from which we know that the gradient, calculated based on $\hat{\boldsymbol{\beta}}^{\lambda}$ will be zero, we get:

$$\hat{\boldsymbol{\beta}}^{\lambda}_{(-i)} \approx \hat{\boldsymbol{\beta}}^{\lambda} - \frac{(\boldsymbol{X}^T_{\mathcal{A}}\boldsymbol{D}\boldsymbol{X}_{\mathcal{A}})^{-1}\boldsymbol{x}_{\mathcal{A}i}\Delta_i}{1 - v_{ii}}, \tag{28}$$

with $\boldsymbol{V} = \boldsymbol{D}^{\frac{1}{2}}\boldsymbol{X}_{\mathcal{A}}(\boldsymbol{X}^T_{\mathcal{A}}\boldsymbol{D}\boldsymbol{X}_{\mathcal{A}})^{-1}\boldsymbol{X}^T_{\mathcal{A}}\boldsymbol{D}^{\frac{1}{2}}$. The $\hat{\boldsymbol{\beta}}^{\lambda}_{(-i)}$'s outside the active set stay 0.

All graphs in the final paper are based on above formula and the assumption that the active set stays the same. However, it is clearly visible that the curves based on this method do have many local optima. Although this is an phenomenon that we also see in the real leave-one-out cross-validated log-likelihood, the approximated curves definitely have more optima and this is a problem when the global optimum has to be found.
Our first idea was that (part of) the spiky behavior may result from the assumption that the active set stays the same when an observation is left out.

In trying to come up with better approximation, we decided to choose the active set of the model with one observation less in a different way. The new derivation is based on an article by Goeman ([4]). In this article it is shown that the gradient $\boldsymbol{g}(\boldsymbol{\beta}) = (g_1(\boldsymbol{\beta}), \ldots, g_p(\boldsymbol{\beta}))'$ of the Cox model with a lasso penalty is given by

$$g_i(\boldsymbol{\beta}) = \begin{cases} h_i(\boldsymbol{\beta}) - \lambda \text{ sign}(\beta_i) & \text{if } \beta_i \neq 0 \\ h_i(\boldsymbol{\beta}) - \lambda \text{ sign}(h_i(\boldsymbol{\beta})) & \text{if } \beta_i = 0 \text{ and } |h_i(\boldsymbol{\beta})| > \lambda \\ 0 & \text{otherwise,} \end{cases}$$

where

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

and $\boldsymbol{h}(\boldsymbol{\beta}) = \partial l(\boldsymbol{\beta})/\partial\boldsymbol{\beta} = (h_1(\boldsymbol{\beta}), \ldots, h_p(\boldsymbol{\beta}))'$ is just the gradient of the unpenalized log-likelihood.
This formula shows that a regression coefficient that equals zero once during the optimization process does not necessarily have to stay at zero, because its gradient can differ from zero. If we look at $\hat{\boldsymbol{\beta}}^{\lambda}$ as an intermediate point in the optimization process in which we try to find $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$, we could look at $\boldsymbol{h}_{(-i)}(\hat{\boldsymbol{\beta}}^{\lambda})$ (which is just $X'_{-i}\Delta_{-i}$, with Δ based on $\hat{\boldsymbol{\beta}}^{\lambda}$) to determine if a regression coefficient belongs to the active set.
The new active sets consist of coefficients for which:

- $\hat{\boldsymbol{\beta}}^{\lambda}_j \neq 0$, or
- $\hat{\boldsymbol{\beta}}^{\lambda}_j = 0$ and $|h_{(-i)_j}(\hat{\boldsymbol{\beta}}^{\lambda})| > \lambda$.

Equation (28) can again be used, but in the new situation more coefficients will be changed.

A disadvantage of this new approach is that the active set can be different for every observation that is left out and for that reason more than one matrix inverse has to be calculated to find one value of the cross-validated log-likelihood. Since the matrices are really similar there will probably be clever ways to keep the computational costs at a minimum but I have not looked into that.
Unfortunately, the first test results based on this new approach did not look too promising. In the graph below, based on the Van de Vijver data set with only the first 100 covariates, 4 curves are visible. The blue line still represents the real curve, the green one represents the standard approach and the purple curve belongs to the approach where the size of the active set can vary.

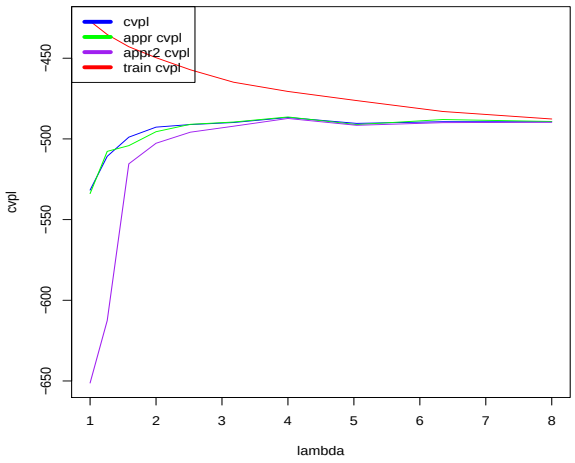


Figure 6: real LOOCV vs. 2 approximation methods

An advantage of the purple curve is that it seems to be less "spiky" than the green curve and finding its global optimum will for that reason be easier. However, it is clear that the new approximation procedure predicts the actual values less accurate, at least for small values of λ. More experiments are needed in order to find out if these findings can be generalized to other data sets.
One remark about the new method can already be made. Although one might be tempted to assume that an extension can only lead to better results, in this case the extension has a drawback. Simulations showed that the new method overestimates the size of the real active set in many cases. It happens quite often that an regression coefficient that equals 0 in $\hat{\boldsymbol{\beta}}^{\lambda}$ but is made part of the

active set in approximating $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$ should actually not be part of this set (but this only shows after more Newton-Raphson steps).

9 Future Work

In this last section a few ideas for future work are discussed briefly.
Although an approximation method for k-fold cross-validation is discussed in chapter 6, more research is needed to find out if the results of this approximation procedure are really useful in practice. An important question is if the amount of time that is saved outweighs the loss of accuracy. For different values of k, the answer to this question may vary.
A different method which leaves room for improvement is the approximation method for models with a lasso penalty. The method which allows for changes in the size of the active set, as proposed in the previous section, should be investigated in more detail.
A third remark is that we only looked at a limited amount of error measures. In case of survival models, we only looked at the cross-validated likelihood and for the logistic model, we just looked at the minus log-likelihood and mean squared error. In those situations, the approximations work well, but for discontinuous measures, like the classification error, the approximations might very well result in quite different values of the tuning parameters than those found by real cross-validation. More work is needed to find out which error measures can be used safely in combination with the approximation method.
A final observation is that the new approximation method is only compared to real cross-validation. Although this comparison is a very important one, more comparisons should be made. Methods like AIC and generalized cross-validation are also designed to approximate cross-validation results and our method should for that reason be compared to those approximation methods in order to find out which approximation method is most useful in terms of computational costs and accuracy.

References

[1] le Cessie S, van Houwelingen JC. (1992) Ridge estimators in logistic regression. *Applied Statistics,* **41**, 191-201

[2] Cook RD, Weisberg S. *Residuals and influence in regression*, Chapman and Hall, New York, 1982.

[3] Cox DR. (1972) Regression models and life-tables. *J.R. Statist. Soc. B,* **34**, 187-220

[4] Goeman JJ. (2010) L_1 penalized estimation in the Cox proportional hazards model. *Biometrical Journal* **52**(1) 70-84

[5] Golub TR. *et al.* (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science,* **286**, 531-537

[6] Hager WW. (1989) Updating the inverse of a matrix. *Society for Industrial and Applied Mathematics,* **31**, 221-239

[7] van Houwelingen JC et al. (2006) Cross-validated Cox regression om microarray gene expression data. *Statistics in Medicine,* **25**, 3201-3216

[8] Magnus JR, Neudecker H. *Matrix Differential Calculus with Applications in Statistics and Econometrics*, revised edition. John Wiley, Chichester, England, 1999.

[9] Myers RH. *Classical and modern regression with applications*, 2nd edn, PWS-Kent, Boston, 1990.)

[10] Tibshirani R. (1996) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, **58**(1), 267-288

[11] Verweij PJM, van Houwelingen HC. (1993) Cross-validation in survival analysis. *Statistics in Medicine,* **12**, 2305-2314

[12] Verweij PJM, van Houwelingen HC. (1994) Penalized likelihood in Cox regression. *Statistics in Medicine,* **13**, 2427-2436

[13] Verweij PJM et al. (1998) A goodness-of-fit test for Cox's proportional hazards model based on martingale residuals. *Biometrics,* **54**, 1517-1526

The Article

Efficient approximate leave-one-out cross-validation for ridge and lasso

Rosa Meijer and Jelle Goeman

Abstract

In evaluating statistical models, leave-one-out cross-validation (LOOCV) is often used. In regression methods suitable for high dimensional covariate spaces, not only the regression coefficient $\boldsymbol{\beta}$ has to be estimated, but also the value of some tuning parameter. The objective of this study is to come up with a method that produces similar output to ordinary cross-validation, but is less time consuming. Estimating the optimal values of ridge and lasso parameters will take less time and carrying out (an approximated version of) double LOOCV will become practically feasible in this way.

We derive a method in which approximations of the real maximum likelihood leave-one-out regression coefficients are used to find optimal values of the tuning parameters in ridge and lasso regression, without actually having to refit the model. The approximations are based on a first order Taylor approximation of the gradient of the log-likelihood around the maximum penalized likelihood estimator of the full model. When the number of observations increases, the real cross-validated estimates will be closer to the estimate of the full model and the error term in the Taylor approximation will diminish. Therefore, this method is especially suitable for large data sets, for which ordinary cross-validating takes much time. The method can be used in generalized linear models as well as in Cox' proportional hazards model. To compare the results of this method to the results that would have been obtained by using ordinary LOOCV, both methods are applied to several microarray data sets.

1 Introduction

As high dimensional data sets are nowadays frequently encountered in statistical analysis, penalized likelihood functions (as used for example in ridge and lasso regression) are more frequently used than ever before. These penalization methods are widely applicable and are used in various settings ranging from linear regression models to Cox' proportional hazards model. The effectiveness of penalization methods depends strongly on the used value of the penalty (or tuning) parameter, which controls the degree of penalization. Hence, selecting this penalty parameter should be done carefully.

Unfortunately, determining the optimal value of this parameter can be troublesome and computationally intensive. The more complex the model (where complexity relates to the specific relation between the predictor and response variables, ranging from a linear relation to for example Cox' proportional hazards model), the more difficult it becomes to find a reliable estimate of the

optimal penalty parameter within a reasonable amount of time. Many different methods have been proposed to find the optimal value of the penalty parameter, but the faster methods can result in crude approximations while the more exact methods can be very time-consuming when there are many observations in combination with a statistical model that is difficult to fit.

The methods most commonly used in order to find a good approximation of the penalty parameter are (variations on) the well known model selection criteria AIC, BIC and the generalized information criterion (GIC) which is a combination of the two [29], generalized cross-validation (GCV) and leave-one-out or k-fold cross-validation methods. Although having a direct formula from which the optimal penalty parameter could be derived immediately would by far be the most appealing option, until now, there are no direct methods that work in many situations. In a recent article by Ueki and Fueda a so called "direct plug-in method" is proposed [20], but this method is definitely not yet applicable in complex statistical models.

The method that is easiest adaptable to all possible models is the so called leave-one-out cross-validated log-likelihood (*cvl*), as described in [22]. The optimal penalty parameter can be found by maximizing the value of the *cvl*. This method is used for example in [28], [11], [8] and [12]. A major advantage of the *cvl* is that the penalty parameter is estimated by maximizing the actual fit of the model and for this reason can hardly be considered as an approximation. Besides, it is applicable to every regression model. Unfortunately, the method can be very time-consuming, especially when the number of observations and covariates gets large and the used regression model is difficult to fit.

One suggested solution in order to speed up the procedure is to turn to the k-fold cross-validated log-likelihood (where k is usually chosen to be either 5 or 10), which is done by many researchers ([10], [1], [16]). Some authors argue that this approach is not only faster, but might also give better results, because there are studies conducted that show that k-fold cross-validation outperforms leave-one-out cross-validation in specific situations, but which resampling method is the best overall option is still a subject of ongoing debate. Furthermore, a huge disadvantage concerning the use of k-fold cross-validation is that the division into train and test groups is completely random and the outcome (the specific value of the tuning parameter in this case) will vary according to this partition. The only way to overcome this problem is to average over all possible partitions (see [3]), which would take far more time than leave-one-out cross-validation (LOOCV).

In an attempt to save computational time, one could also use more approximative methods. The main advantage of AIC/BIC methods and generalized cross-validation is that the computational costs involved with these methods are relatively low. Much attention has been paid to the performance of AIC and BIC in a linear regression framework and it has been shown that good approximations of tunings parameter values can be found by applying these methods ([19], [25]). However, in more advanced models, like the Cox' proportional hazards model, these criteria can fail spectacularly as is shown by Schumacher et al. [16]. Therefore, extensive research is needed before methods derived from AIC or BIC can be introduced to find penalty parameters in more complex models.

Generalized cross-validation is also mostly applied in linear regression settings. In comparison with the methods derived from AIC and BIC, GCV seems to result in overfitting ([19], [25]). Besides, generalized cross-validation is used

less frequently when the model complexity increases. In combination with Cox regression for example, GCV is not often used (although there are exceptions [18], [26]).

The solution we suggest in this article is to stick to LOOCV and use approximations of the real cross-validated parameters that can be found in a relatively easy way. In that case, the model has not to be refitted n times (with n the number of observations) to calculate one specific value of the *cvl*. In this article, we will derive a approximated *cvl*-method in which approximations of the real maximum likelihood leave-one-out regression coefficients are used to find optimal values of the tuning parameters in ridge and lasso regression, without actually having to refit the model. The approximations are based on the maximum penalized likelihood estimator of the full model. In a linear (ridge) regression model, the method will even result in exact results, whereas in other generalized linear models and the Cox' model Taylor approximations will be used. Some work on this subject has already be done ([5], [4], [22]), but despite the fact that the possibility of using these approximative leave-one-out regression coefficients is mentioned multiple times, these approximations have only been used in logistic ridge regression [4] and in linear regression [2]. In this article, we will extend its use to the Cox proportional hazards model en we will also investigate its useability in combination with lasso regression.

The penalty parameter estimation method we will obtain in this way is far less time-consuming than the original leave-one-out cross validation method and is still less approximative than for example GCV or AIC. In addition, when the number of observations grows, the approximations of the cross-validated parameters get closer to the actual values, while this is exactly the situation where real LOOCV will take a considerable amount of time. With this method, even an approximated version of double LOOCV will become practically feasible.

The approximation procedure will be derived in section 2 for generalized linear models and Cox' proportional hazards model with a ridge or lasso penalty. In section 3, the method will be tested on four well-known survival data sets. The actual results as well as the efficiency of the approximation method will be discussed in detail. In addition, the dependence of the performance of the approximation method on the size as well as the effective dimension of the data set is investigated.

2 Approximation procedure

In building a regression model, the aim is to find the value of the regression coefficient $\boldsymbol{\beta}$ that best fits the data. Since the model is supposed to be tested on an independent test set, but independent test data is usually absent, some resampling method should be used. In case of LOOCV, each observation is chosen once to be the test set. The optimal value of $\boldsymbol{\beta}$ is calculated based on the remaining $n-1$ observations (where n is the total number of observations). Let us denote this estimate by $\hat{\boldsymbol{\beta}}_{-i}$. For every i $(i = 1 \dots n)$ $\hat{\boldsymbol{\beta}}_{-i}$ has to be calculated by refitting the model.

If the model also depends on some tuning parameter λ, not only $\boldsymbol{\beta}$ but also λ has to be estimated. To find the optimal value of λ, for different values of λ the above procedure should be carried out and the value of λ that leads to the optimal value of some chosen measure of predictive performance should be used in the final model. The $\hat{\boldsymbol{\beta}}_{-i}$'s now depend on λ and can be denoted by $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$. To determine $\hat{\lambda}$, we have to refit the model many times.

Instead of using the real values of $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$'s, approximations can be used that are based on $\hat{\boldsymbol{\beta}}^{\lambda}$, the optimal value of $\boldsymbol{\beta}$ based on all observations (and some specific value of λ). In this way, the model does not have to be refitted n times per value of λ and finding the optimal value of λ will be far less time consuming. In this section, it will be shown that the exact value of $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$ can be calculated based on $\hat{\boldsymbol{\beta}}^{\lambda}$ in a linear ridge regression model and this method can be extended to generalized linear models and Cox' proportional hazards model with a ridge or lasso penalty, although the calculations are then based on Taylor expansions and the values of the $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$'s are no longer the exact values but approximations.

In all calculations, the Sherman-Morrison-Woodbury theorem is applied. This theorem states (as given in [14])

$$\left(\boldsymbol{B} + \boldsymbol{u}\boldsymbol{v}^T\right)^{-1} = \boldsymbol{B}^{-1} - \frac{\boldsymbol{B}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{B}^{-1}}{1 + \boldsymbol{v}^T\boldsymbol{B}^{-1}\boldsymbol{u}}, \tag{1}$$

where $\boldsymbol{B}$ is a nonsingular $p \times p$ matrix, and $\boldsymbol{u}$ and $\boldsymbol{v}$ are p-dimensional column vectors.

2.1 Linear Model

We start by deriving a closed formula for $\hat{\boldsymbol{\beta}}_{-i}$ in an ordinary linear ridge model. Let $(y_i, \boldsymbol{x}_i)$ $(i = 1 \dots n)$ be n independent and identically distributed observations, where y_i is the response of interest and $\boldsymbol{x}_i = (x_{i1}, \dots, x_{ip})^T$ is the associated p-dimensional vector of regressors. We assume the data to be generated according to the following model:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2}$$

where $E[\boldsymbol{\epsilon}] = \boldsymbol{0}$ and $\mathrm{Var}(\boldsymbol{\epsilon}) = \sigma^2\boldsymbol{I}$. $\boldsymbol{X}$ is the $n \times p$ design matrix, with the $\boldsymbol{x}_i$'s as its rows and $\boldsymbol{\beta}$ is a p-dimensional vector of unknown parameters.
Just as in ordinary linear regression, we minimize the residual sum of squares (RSS) in order to find the best approximation of $\boldsymbol{\beta}$, but now with an extra penalty term added:

$$RSS(\lambda) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{A}\boldsymbol{\beta}. \tag{3}$$

Here $\boldsymbol{A}$ is a diagonal matrix which can be the identity matrix if all covariates should be penalized, or a diagonal matrix with zero's as well as ones on the diagonal, in case there are any additional covariates that should remain unpenalized. In our calculations, the only restriction we need to impose on $\boldsymbol{A}$ is that it must be a symmetrical matrix and the inverse of $(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{A})$ must exist. Minimizing (3) with respect to $\boldsymbol{\beta}$, for a fixed value of λ gives us

$$\hat{\boldsymbol{\beta}}^{\lambda} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{A})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{4}$$

To find the optimal value of λ by means of LOOCV the cross-validated sum of squares should be minimized:

$$RSS_{cv}(\lambda) = \sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^T\hat{\boldsymbol{\beta}}_{-i}^{\lambda})^2, \tag{5}$$

where $\hat{\boldsymbol{\beta}}_{-i}^{\lambda}$ is given by (4), only this time $\boldsymbol{X}$ is a $(n-1)\times p$ matrix (from now on denoted by $\boldsymbol{X}_{-i}$) and the i^{th} element of $\boldsymbol{y}$ is left out ($\boldsymbol{y}_{-i}$).
Instead of calculating all $\hat{\boldsymbol{\beta}}_{-i}^{\lambda}$'s directly, which would involve n different inverse matrices, we can compute these $\hat{\boldsymbol{\beta}}_{-i}^{\lambda}$'s in an easier way from $\hat{\boldsymbol{\beta}}^{\lambda}$ by making convenient choices for the matrix $\boldsymbol{B}$, and the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ as defined in theorem 1.
Setting $\boldsymbol{B} = \boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{A}$, $\boldsymbol{u} = \boldsymbol{x}_i$ and $\boldsymbol{v} = -\boldsymbol{x}_i$, we get:

$$\begin{aligned}(\boldsymbol{X}_{-i}^T\boldsymbol{X}_{-i} + \lambda\boldsymbol{A})^{-1} &= (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{A} - \boldsymbol{x}_i\boldsymbol{x}_i^T)^{-1} \\ &= (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{A})^{-1} + \frac{(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{A})^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^T(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{A})^{-1}}{1 - \boldsymbol{x}_i^T(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{A})^{-1}\boldsymbol{x}_i}.\end{aligned} \tag{6}$$

After multiplying (6) by $(\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{x}_i y_i)$ we find, after some simplifications:

$$\hat{\boldsymbol{\beta}}_{-i}^{\lambda} = \hat{\boldsymbol{\beta}}^{\lambda} - \frac{(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{A})^{-1}\boldsymbol{x}_i e_i^{\lambda}}{1 - h_{ii}^{\lambda}}, \tag{7}$$

where h_{ii}^{λ} is the i^{th} diagonal element of the hat-matrix $\boldsymbol{H}^{\lambda}$ given by $\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{A})^{-1}\boldsymbol{X}^T$ and $\boldsymbol{e}^{\lambda}$ is the vector of residuals given by $\boldsymbol{e}^{\lambda} = \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}^{\lambda}$.
From equation (7) we can conclude that only one inverse is needed in order to compute all n $\hat{\boldsymbol{\beta}}_{-i}^{\lambda}$'s.

To find the optimal value of the ridge parameter λ, equation (5) needed to be minimized. By substituting (7) into (5), we get the following expression:

$$RSS_{cv}(\lambda) = \sum_{i=1}^{n}\left(\frac{e_i^{\lambda}}{1 - h_{ii}^{\lambda}}\right)^2. \tag{8}$$

In order to make the calculation of this sum as fast as possible, we looked for a way to express this sum as a matrix multiplication.
From (5) and (8) we can deduce

$$\boldsymbol{y} - \hat{\boldsymbol{y}}_{cv} = (diag(\boldsymbol{I}_n - \boldsymbol{H}^{\lambda}))^{-1}(\boldsymbol{y} - \hat{\boldsymbol{y}}) = (diag(\boldsymbol{I}_n - \boldsymbol{H}^{\lambda}))^{-1}(\boldsymbol{I}_n - \boldsymbol{H}^{\lambda})\boldsymbol{y},$$

where $\hat{\boldsymbol{y}}_{cv}$ is the vector with predicted y-values, based on the cross-validated $\hat{\boldsymbol{\beta}}$'s. Taking the inner product of this vector and using that $(diag(\boldsymbol{I}_n - \boldsymbol{H}^\lambda))^{-1}$ and $(\boldsymbol{I}_n - \boldsymbol{H}^\lambda)$ are symmetric matrices, we get:

$$RSS_{cv}(\lambda) = \boldsymbol{y}^T(\boldsymbol{I}_n - \boldsymbol{H}^\lambda)(diag(\boldsymbol{I}_n - \boldsymbol{H}^\lambda))^{-2}(\boldsymbol{I}_n - \boldsymbol{H}^\lambda)\boldsymbol{y}. \tag{9}$$

Using equation (9), for every value of λ the cross-validated sum of squares can be directly calculated without having to fit the model multiple times. It is even possible to find the derivative of equation (9) with respect to λ, which can be used in minimizing $RSS_{cv}(\lambda)$ in order to find the optimal value of λ.

In above derivations, we started from a linear ridge model. If one would want to use a similar expression for the cross-validated residual sum of squares in an unpenalized linear regression model, for instance to use as a measure of predictive performance to compare several models, one could just use $\lambda = 0$ in above formulas. However, in that case one should make sure that the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is invertible and therefore the number of covariates can not exceed the number of observations.

A very important aspect of the derivation is that $\hat{\boldsymbol{\beta}}^\lambda$ (and therefore $\hat{\boldsymbol{\beta}}^\lambda_{-i}$) is not found in an iterative way, but results from a closed formula. When $\hat{\boldsymbol{\beta}}^\lambda$ is found by means of some numerical procedure, as will be the case in the models discussed in the upcoming sections, it will not be possible to find a direct formula based on $\hat{\boldsymbol{\beta}}^\lambda$ that gives the exact value of $\hat{\boldsymbol{\beta}}^\lambda_{-i}$. Derivations analogous to the one just described will result in approximative rather than exact values. However, based on the results in the linear model, these approximations can be expected to come close to the real values.

2.2 Generalized linear models

After some adjustments, the procedure as described in the previous subsection can be applied to other generalized linear models as well. Our data still consists of the pairs $(y_i, \boldsymbol{x}_i)$ $(i = 1 \dots n)$ only this time $\boldsymbol{y}$ follows a distribution in the exponential family with mean $\boldsymbol{\mu} = E[\boldsymbol{y}]$ and variance $\boldsymbol{V} = Var(\boldsymbol{y})$. The linear predictor is denoted by $\boldsymbol{\eta}$ and is given by $\boldsymbol{X\beta}$. In the linear model, the relationship between $\boldsymbol{\mu}$ and $\boldsymbol{\eta}$ was just $\boldsymbol{\mu} = \boldsymbol{\eta}$ but now this relationship is given by $\boldsymbol{\mu} = g^{-1}(\boldsymbol{\eta})$, where $g(.)$ is said to be the *link function*.

The probability density function of $\boldsymbol{y}$ is given by

$$f_y(y; \theta, \phi) = \exp\{(y\theta - b(\theta))/a(\phi) + c(y, \phi)\} \tag{10}$$

where $a(.)$, $b(.)$ and $c(.)$ are functions that vary according to the specific distributions [13]. Let us denote the corresponding log-likelihood by $l(\boldsymbol{\beta})$ and the penalized log-likelihood by

$$l^\lambda(\boldsymbol{\beta}) = l(\boldsymbol{\beta}) - \frac{1}{2}\lambda\boldsymbol{\beta}^T\boldsymbol{A}\boldsymbol{\beta}, \tag{11}$$

with $\boldsymbol{A}$ a symmetric $p \times p$ matrix as defined earlier.

If we only consider the models with a canonical link function, we know that

there exists a diagonal weightmatrix $\boldsymbol{W}$ (which equals $\boldsymbol{V} = Var(\boldsymbol{y})$), such that the first and second derivative of the penalized log-likelihood with respect to $\boldsymbol{\beta}$ are of the following form:

$$\frac{\partial l^{\lambda}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{\mu}) - \lambda \boldsymbol{A}\boldsymbol{\beta} \tag{12}$$

and

$$\frac{\partial^2 l^{\lambda}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} = -\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X} - \lambda \boldsymbol{A}, \tag{13}$$

As we already mentioned, usually there exists no closed formula for $\hat{\boldsymbol{\beta}}^{\lambda}$ and its value may be obtained by some numeric maximization procedure such as the Newton-Raphson algorithm.

To find an approximation of $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$ based on $\hat{\boldsymbol{\beta}}^{\lambda}$ we make a first-order Taylor approximation of the derivative of $l^{\lambda}_{(-i)}$ at $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}^{\lambda}$ as suggested in [22] where $l^{\lambda}_{(-i)}$ is given by

$$l^{\lambda}_{(-i)}(\boldsymbol{\beta}) = l^{\lambda}(\boldsymbol{\beta}) - l^{\lambda}_i(\boldsymbol{\beta}) \tag{14}$$

with $l^{\lambda}_i(\boldsymbol{\beta})$ the contribution of observation i to the likelihood. This gives us

$$\frac{\partial l^{\lambda}_{(-i)}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \frac{\partial l^{\lambda}_{(-i)}(\hat{\boldsymbol{\beta}}^{\lambda})}{\partial \boldsymbol{\beta}} + \frac{\partial^2 l^{\lambda}_{(-i)}(\hat{\boldsymbol{\beta}}^{\lambda})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T}(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}^{\lambda}) + O\left((\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}^{\lambda})^2\right). \tag{15}$$

If we replace $\boldsymbol{\beta}$ by $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$ in (15) and use that $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$ by definition maximizes $l^{\lambda}_{(-i)}(\boldsymbol{\beta})$ (and is therefore the root of its derivative), we find

$$\hat{\boldsymbol{\beta}}^{\lambda}_{-i} = \hat{\boldsymbol{\beta}}^{\lambda} - \left(\frac{\partial^2 l^{\lambda}_{(-i)}(\hat{\boldsymbol{\beta}}^{\lambda})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T}\right)^{-1} \frac{\partial l^{\lambda}_{(-i)}(\hat{\boldsymbol{\beta}}^{\lambda})}{\partial \boldsymbol{\beta}} + O\left((\hat{\boldsymbol{\beta}}^{\lambda}_{-i} - \hat{\boldsymbol{\beta}}^{\lambda})^2\right). \tag{16}$$

Note that this expression exactly equals one iteration of the Newton-Raphson algorithm, starting form the maximum likelihood estimator (MLE) belonging to the full model. From now on, for notational simplicity, we will leave the error term out of the equations, and use the approximation sign ($\approx$) instead of the equality sign.

If we substitute the expression for the first and second derivative, as given in (12) and (13), use the Sherman-Morrison-Woodbury theorem and introduce the variables $\boldsymbol{Z} = \hat{\boldsymbol{W}}^{\frac{1}{2}}\boldsymbol{X}$ and $\boldsymbol{v} = \hat{\boldsymbol{W}}^{-\frac{1}{2}}(\boldsymbol{y} - \hat{\boldsymbol{\mu}})$ we can rewrite (16) into

$$\hat{\boldsymbol{\beta}}^{\lambda}_{-i} \approx \hat{\boldsymbol{\beta}}^{\lambda} + \left((\boldsymbol{Z}^T\boldsymbol{Z} + \lambda\boldsymbol{A})^{-1} + \frac{(\boldsymbol{Z}^T\boldsymbol{Z} + \lambda\boldsymbol{A})^{-1}\boldsymbol{z}_i\boldsymbol{z}_i^T(\boldsymbol{Z}^T\boldsymbol{Z} + \lambda\boldsymbol{A})^{-1}}{1 - \boldsymbol{z}_i^T(\boldsymbol{Z}^T\boldsymbol{Z} + \lambda\boldsymbol{A})^{-1}\boldsymbol{z}_i}\right)(\boldsymbol{Z}^T\boldsymbol{v} - \boldsymbol{z}_i v_i - \lambda\boldsymbol{A}\hat{\boldsymbol{\beta}}^{\lambda}), \tag{17}$$

where the values of $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{\mu}}$ are based on the value for $\hat{\boldsymbol{\beta}}^{\lambda}$. Since the Newton-Raphson algorithm has converged to $\hat{\boldsymbol{\beta}}^{\lambda}$, we know that $\boldsymbol{Z}^T\boldsymbol{v} - \lambda\boldsymbol{A}\hat{\boldsymbol{\beta}}^{\lambda}$ equals 0 and our result simplifies to:

$$\hat{\boldsymbol{\beta}}^{\lambda}_{-i} \approx \hat{\boldsymbol{\beta}}^{\lambda} - \frac{(\boldsymbol{X}^T\hat{\boldsymbol{W}}\boldsymbol{X} + \lambda\boldsymbol{A})^{-1}\boldsymbol{x}_i(y_i - \hat{\mu}_i)}{1 - v_{ii}}, \tag{18}$$

where v_{ii} is the i^{th} diagonal element of the matrix $\boldsymbol{V}$ given by $\hat{\boldsymbol{W}}^{\frac{1}{2}}\boldsymbol{X}(\boldsymbol{X}^T\hat{\boldsymbol{W}}\boldsymbol{X}+\lambda\boldsymbol{A})^{-1}\boldsymbol{X}^T\hat{\boldsymbol{W}}^{\frac{1}{2}}$. Again, all n $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$'s can be found by simple matrix calculations and only one inverse calculation.

A problem can still occur if this inverse is computationally hard to calculate. This may happen when the number of covariates (p) is extremely large, since $(\boldsymbol{X}^T\hat{\boldsymbol{W}}\boldsymbol{X}+\lambda\boldsymbol{A})$ is a $p\times p$ matrix. To overcome this problem, we can reparameterize the model in terms of $\boldsymbol{\gamma}$ instead of $\boldsymbol{\beta}$, where $\boldsymbol{\beta}=\boldsymbol{G}^T\boldsymbol{\gamma}$ in order to reduce the dimension of the covariate space from p to n, as explained in [11]. In [11] the simplifying assumption is made that the matrix $\boldsymbol{A}$ equals the identity matrix $\boldsymbol{I}_p$. Although this assumption is not strictly necessary, for now, we will make this same assumption for simplicity reasons.
The first and second derivative, as given by equations (12) and (13), can now be given in terms of $\boldsymbol{\gamma}$ and become

$$\frac{\partial l^{\lambda}(\boldsymbol{\gamma})}{\partial\boldsymbol{\gamma}}=\boldsymbol{G}\boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{\mu})-\lambda\tilde{\boldsymbol{A}}\boldsymbol{\gamma} \tag{19}$$

and

$$\frac{\partial^2 l^{\lambda}(\boldsymbol{\gamma})}{\partial\boldsymbol{\gamma}\partial\boldsymbol{\gamma}^T}=-\boldsymbol{G}\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}\boldsymbol{G}^T-\lambda\tilde{\boldsymbol{A}}, \tag{20}$$

where $\tilde{\boldsymbol{A}}=\boldsymbol{G}\boldsymbol{A}\boldsymbol{G}^T$.

To find $\hat{\boldsymbol{\gamma}}^{\lambda}_{-i}$ in terms of $\hat{\boldsymbol{\gamma}}^{\lambda}$, we can again use equation (16), this time with $\boldsymbol{\gamma}$ instead of $\boldsymbol{\beta}$. To make the formula's more specific, we will choose the reparametrization matrix $\boldsymbol{G}$ to be equal to the design matrix $\boldsymbol{X}$, as is done in [11]. If we leave out one observation (which gives the design matrix $\boldsymbol{X}_{-i}$), we can still use the same reparametrization matrix considering the fact that if $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$ lies in the column space of $\boldsymbol{X}^T_{-i}$, as proven in [11], it certainly lies in the column space of $\boldsymbol{X}^T$ as well. ($\hat{\boldsymbol{\beta}}^{\lambda}_{-i}\in\text{span}\{\boldsymbol{X}^T_{-i}\}\Rightarrow\hat{\boldsymbol{\beta}}^{\lambda}_{-i}\in\text{span}\{\boldsymbol{X}^T\}$.)
From the fact that we only have to remove observation i from the "inner" matrix $\boldsymbol{X}\boldsymbol{W}\boldsymbol{X}^T$ and the observation that the i^{th} column of the matrix $\boldsymbol{X}\boldsymbol{X}^T$ is just $\boldsymbol{X}\boldsymbol{x}_i$ it follows that the Sherman-Morrison-Woodbury theorem can again be applied and this will eventually lead to the following equation:

$$\hat{\boldsymbol{\gamma}}^{\lambda}_{-i}\approx\hat{\boldsymbol{\gamma}}^{\lambda}-\frac{(\boldsymbol{B}^T\hat{\boldsymbol{W}}\boldsymbol{B}+\lambda\tilde{\boldsymbol{A}})^{-1}\boldsymbol{b}_i(y_i-\hat{\mu}_i)}{1-v_{ii}}, \tag{21}$$

with $\boldsymbol{B}=\boldsymbol{X}\boldsymbol{X}^T$, $\tilde{\boldsymbol{A}}=\boldsymbol{X}\boldsymbol{A}\boldsymbol{X}^T$ and $\boldsymbol{V}=\hat{\boldsymbol{W}}^{\frac{1}{2}}\boldsymbol{B}(\boldsymbol{B}^T\hat{\boldsymbol{W}}\boldsymbol{B}+\lambda\tilde{\boldsymbol{A}})^{-1}\boldsymbol{B}^T\hat{\boldsymbol{W}}^{\frac{1}{2}}$.

The approximations of the $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$'s (which are if necessary also easily obtainable by multiplying $\hat{\boldsymbol{\gamma}}^{\lambda}_{-i}$ with $\boldsymbol{X}^T$) can now be used in combination with some error measure (like the residual sum of squares in the linear case). In the result section we will use the cross-validated log-likelihood as an indicator for predictive performance, but a different continuous error measure (the squared error for example) could also be used as a performance measure.

In the result section it will be shown that the cross-validated log-likelihood based on the approximations for $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$ resembles the real cross-validated log-likelihood very well. Especially when n grows large, which is exactly the situation in which regular cross-validation can be very time consuming. This can also

be concluded from equation (16), where we see that the error between the approximated an the real cross validated estimate of $\boldsymbol{\beta}$ is of the order $(\hat{\boldsymbol{\beta}}^{\lambda}_{-i} - \hat{\boldsymbol{\beta}}^{\lambda})^2$. The more observations, the smaller the effect of leaving out one of them will be on the optimal value of the regression coefficient and therefore, the difference between $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$ and $\hat{\boldsymbol{\beta}}^{\lambda}$ will be smaller, which results in a smaller error.

2.3 Cox' proportional hazards model

In the previous section, an approximation method to find $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$ based on $\hat{\boldsymbol{\beta}}^{\lambda}$ was derived for GLM's. In this section a similar method will be derived for Cox' proportional hazards model.

Again, we have n samples, this time of the form $(t_i, d_i, \boldsymbol{x}_i)$ $(i = 1 \ldots n)$. Here the (censored) survival time is denoted by t_i, the censoring indicator is given by d_i (with $d_i = 0$ in case of censoring) and $\boldsymbol{x}_i$ is still a p-dimensional vector of covariates. In Cox' proportional hazards model ([6]), the hazard function for individual i is given by:

$$h_i(t) = h_0(t) \exp(\boldsymbol{x}_i^T \boldsymbol{\beta}),$$

where $h_0(t)$ is the well-known baseline hazard.
Usually one finds $\hat{\boldsymbol{\beta}}$ by maximizing the partial log-likelihood, given by

$$pl(\boldsymbol{\beta}) = \sum_{i=1}^{n} d_i (\boldsymbol{x}_i^T \boldsymbol{\beta}) - \ln \left(\sum_{t_j \geq t_i} \exp(\boldsymbol{x}_i^T \boldsymbol{\beta}) \right). \tag{22}$$

However, we choose to work with the total log-likelihood instead, because the second derivative of the partial log-likelihood has not the desired form. Although the hessian matrix is given by $-\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X}$, this time the matrix $\boldsymbol{W}$ is not a diagonal matrix. The hessian of total log-likelihood is similar to those described in the previous section and is therefore more suitable if we want to apply (a slightly altered version of) the already described approximation procedure.

The total log-likelihood (as used for example in [11]) is given by:

$$l(\boldsymbol{\beta}, h_0) = \sum_{i=1}^{n} \left[-\exp(\boldsymbol{x}_i^T \boldsymbol{\beta}) H_0(t_i) + d_i \left(\ln(h_0(t_i)) + \boldsymbol{x}_i^T \boldsymbol{\beta} \right) \right], \tag{23}$$

with

$$H_0(t) = \sum_{s \leq t} h_0(s). \tag{24}$$

As before, we add a ridge penalty to this likelihood, which results in:

$$l^{\lambda}(\boldsymbol{\beta}, h_0) = l(\boldsymbol{\beta}, h_0) - \frac{1}{2} \lambda \boldsymbol{\beta}^T \boldsymbol{A} \boldsymbol{\beta}. \tag{25}$$

The first and second derivative with respect to $\boldsymbol{\beta}$ are now given by:

$$\frac{\partial l^{\lambda}(\boldsymbol{\beta}, h_0)}{\partial \boldsymbol{\beta}} = \boldsymbol{X}^T \boldsymbol{\Delta} - \lambda \boldsymbol{A} \boldsymbol{\beta}, \tag{26}$$

with

$$\Delta_i = d_i - H_0(t_i)\exp(\boldsymbol{x}_i^T\boldsymbol{\beta})$$

and

$$\frac{\partial^2 l^\lambda(\boldsymbol{\beta}, h_0)}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T} = -\boldsymbol{X}^T\boldsymbol{D}\boldsymbol{X} - \lambda\boldsymbol{A}, \tag{27}$$

where

$$\boldsymbol{D} = \text{diagonal}(\exp(\boldsymbol{x}_i^T\boldsymbol{\beta})H_0(t_i)).$$

The only difference between equation (12) and (26) is that the vector $\boldsymbol{y} - \boldsymbol{\mu}$ is replaced by the vector Δ. The formula for the hessian in the Cox model is also similar to the one found in the previous section (equation (13)). The diagonal weightmatrix $\boldsymbol{W}$ is replaced by $\boldsymbol{D}$, which is again a diagonal matrix.

Since the expressions are very similar, we can use exactly the same procedure as we used before to find an approximative value of $\hat{\boldsymbol{\beta}}_{-i}^\lambda$. This time, we find the following formula:

$$\hat{\boldsymbol{\beta}}_{-i}^\lambda \approx \hat{\boldsymbol{\beta}}^\lambda - \frac{(\boldsymbol{X}^T\hat{\boldsymbol{D}}\boldsymbol{X} + \lambda\boldsymbol{A})^{-1}\boldsymbol{x}_i\hat{\Delta}_i}{1 - v_{ii}}, \tag{28}$$

with $\boldsymbol{V} = \hat{\boldsymbol{D}}^{\frac{1}{2}}\boldsymbol{X}(\boldsymbol{X}^T\hat{\boldsymbol{D}}\boldsymbol{X} + \lambda\boldsymbol{A})^{-1}\boldsymbol{X}^T\hat{\boldsymbol{D}}^{\frac{1}{2}}$.

The values of $H_0(t_i)$ are calculated with the Breslow estimator:

$$\hat{h_0}(t_i) = 1/\sum_{t_j \geq t_i} \exp(\boldsymbol{x}_j^T\hat{\boldsymbol{\beta}}^\lambda).$$

The approximations of the leave-one-out estimates are thus based on the baseline hazard corresponding to the full model. Usually, the baseline hazard corresponding to the model where one sample is left out, will be different from the one belonging to the full model. A little adjustment in the approximation procedure can be made in order to allow for small changes in the baseline hazard. By altering the design matrix $\boldsymbol{X}$ by adding an extra column of ones (or a different number, for computational reasons) and adding an extra zero to the matrix $\boldsymbol{A}$ (to make sure that this new intercept term will not be penalized) we allow for multiplicative changes in the baseline hazard. In the result section, it will be shown that this small change will indeed result in better approximations.
The new approximations are thus given by equation (28), but this time with

$$\boldsymbol{X_{int}} = \begin{pmatrix} 1 & \\ \vdots & \boldsymbol{X} \\ 1 & \end{pmatrix}, \boldsymbol{A_{int}} = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \boldsymbol{A} & \\ 0 & & \end{pmatrix} \text{ and } \hat{\boldsymbol{\beta}}_{int}^\lambda = \begin{pmatrix} 0 \\ \hat{\boldsymbol{\beta}}^\lambda \end{pmatrix}.$$

In this way, $p+1$ regression coefficients are calculated, but only the last p of them are the approximations we are interested in. The first coefficient is only to find better approximations of the remaining p coefficients.

As mentioned before, problems can occur when the inverse in equation (28) is a very large matrix (this time a $(p+1) \times (p+1)$ matrix). In view of the fact that Cox' proportional hazards model is very similar to generalized linear

models, it is not surprising that the reparametrization procedure can again be applied.

This time, we have an intercept term incorporated in the model and therefore we can not take the reparametrization $\boldsymbol{G}$ to be equal to $\boldsymbol{X}$. However, the following form suffices:

$$\boldsymbol{G} = \begin{pmatrix} 1 & \dots & 0 \\ \vdots & \boldsymbol{X} & \\ 0 & & \end{pmatrix},$$

where $\boldsymbol{X}$ is the design matrix without the extra column for the intercept term. (Note that this reparametrization matrix could also be used when we want to add an intercept term to a generalized linear model.)

Applying the same procedure as in the previous section, we obtain the following formula:

$$\hat{\boldsymbol{\gamma}}^{\lambda}_{-i} \approx \hat{\boldsymbol{\gamma}}^{\lambda} - \frac{(\boldsymbol{B}^T\hat{\boldsymbol{D}}\boldsymbol{B} + \lambda\tilde{\boldsymbol{A}})^{-1}\boldsymbol{b}_i\hat{\Delta}_i}{1 - v_{ii}}, \tag{29}$$

with $\boldsymbol{B} = \boldsymbol{X_{int}}\boldsymbol{G}^T$, $\tilde{\boldsymbol{A}} = \boldsymbol{G}\boldsymbol{A_{int}}\boldsymbol{G}^T$ and $\boldsymbol{V} = \hat{\boldsymbol{D}}^{\frac{1}{2}}\boldsymbol{B}(\boldsymbol{B}^T\hat{\boldsymbol{D}}\boldsymbol{B} + \lambda\tilde{\boldsymbol{A}})^{-1}\boldsymbol{B}^T\hat{\boldsymbol{D}}^{\frac{1}{2}}$.

In the results section, the real leave one out estimates will be compared to the ones obtained by the formula above. It will be shown that the approximated estimates are close to the real estimates and can be found in far less time.

2.4 A different penalty: the lasso

Up to this point, the approximation procedure has only been derived for a model with a ridge penalty (or no penalty at all). In this section, the procedure will be adjusted in order to be used in combination with lasso regression. The procedure will be derived in a Cox regression model, but the derivation for generalized linear models is completely analogous.

The log-likelihood is still denoted by $l(\boldsymbol{\beta}, h_0)$ as given in equation(23), but this time the penalized log-likelihood is given by:

$$l^{\lambda}(\boldsymbol{\beta}, h_0) = l(\boldsymbol{\beta}, h_0) - \lambda\sum_{j=1}^{p}|\beta_j|. \tag{30}$$

Again, we want to use this approximation:

$$\hat{\boldsymbol{\beta}}^{\lambda}_{-i} \approx \hat{\boldsymbol{\beta}}^{\lambda} - \left(\frac{\partial^2 l^{\lambda}_{(-i)}(\hat{\boldsymbol{\beta}}^{\lambda})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T}\right)^{-1}\frac{\partial l^{\lambda}_{(-i)}(\hat{\boldsymbol{\beta}}^{\lambda})}{\partial\boldsymbol{\beta}}. \tag{31}$$

However, the penalty term in equation (30) is only differentiable at points where all β_j's ($j = 1 \dots p$) are unequal to zero, since the function $f(x) = |x|$ is not differentiable in $x = 0$.

To make sure that the first and second derivative of the penalized log-likelihood do exist, we only take those coefficients of $\hat{\boldsymbol{\beta}}^{\lambda}$ that are not equal to 0 (the so called "active set", denoted by $\mathcal{A}$) and alter these values to find the

coefficients of $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$. For all j for which $\hat{\beta}^{\lambda}_{j} = 0$ holds, we will take the approximations of $\hat{\beta}^{\lambda}_{(-i)_j}$ to be equal to 0 as well. Since the lasso is designed to be stable ([17]) it can be expected that the active set will not be influenced much by small changes in the data (or leaving out one observation), especially for large data sets.

If we assume that the elements of $\boldsymbol{\beta}$ are unequal to zero, the derivatives of the penalized log-likelihood are now given by the following expressions:

$$\frac{\partial l^{\lambda}(\boldsymbol{\beta}, h_0)}{\partial \boldsymbol{\beta}} = \boldsymbol{X}^T \boldsymbol{\Delta} - \text{sign}(\boldsymbol{\beta}), \tag{32}$$

where sign($\boldsymbol{\beta}$) is defined to be a vector with sign(β_j) as its j^{th} element and

$$\frac{\partial^2 l^{\lambda}(\boldsymbol{\beta}, h_0)}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} = -\boldsymbol{X}^T \boldsymbol{D} \boldsymbol{X}, \tag{33}$$

where Δ and $\boldsymbol{D}$ are defined in the same way as in the previous section.

After substituting these equations in equation (31) and applying the Sherman-Morrison-Woodbury theorem to rewrite the leave-one-out hessian matrix, we find again an approximation formula.
Since we only look at the non-zero regression coefficients, we can eliminate the covariates in the design matrix which correspond to the zero's in the parameter vector. This smaller design matrix will be denoted by $\boldsymbol{X}_{\mathcal{A}}$ (the design matrix of the active set).
The approximations (for the nonzero coefficients) become:

$$\hat{\boldsymbol{\beta}}^{\lambda}_{-i} \approx \hat{\boldsymbol{\beta}}^{\lambda} - \frac{(\boldsymbol{X}^T_{\mathcal{A}} \hat{\boldsymbol{D}} \boldsymbol{X}_{\mathcal{A}})^{-1} \boldsymbol{x}_{\mathcal{A}i} \hat{\Delta}_i}{1 - v_{ii}}, \tag{34}$$

with $\boldsymbol{V} = \hat{\boldsymbol{D}}^{\frac{1}{2}} \boldsymbol{X}_{\mathcal{A}} (\boldsymbol{X}^T_{\mathcal{A}} \hat{\boldsymbol{D}} \boldsymbol{X}_{\mathcal{A}})^{-1} \boldsymbol{X}^T_{\mathcal{A}} \hat{\boldsymbol{D}}^{\frac{1}{2}}$.

This time it will not be necessary to use a reparametrization trick in order to be able to calculate the inverse of $\boldsymbol{X}^T_{\mathcal{A}} \hat{\boldsymbol{D}} \boldsymbol{X}_{\mathcal{A}}$. The matrix is not longer a $p \times p$ matrix, but a $k \times k$ matrix, where k is the number of elements in the active set. Usually, k will be much smaller than p and taking the inverse of a $k \times k$ matrix will for this reason not be a problem.

To account for multiplicative changes in the baseline-hazard, the design matrix $\boldsymbol{X}_{\mathcal{A}}$ can again be extended with an extra column of ones. In the result section it will be shown that adding this intercept term will have a less beneficial influence on the outcomes in a lasso model than in a ridge model.

3 Results

In this section we investigate the usefulness of the approximation method in practice. The focus will be on two different aspects of the approximated leave-one-out maximum (penalized) likelihood estimates. First of all, we want to know if the approximations are on average close to the real leave-one-out estimates, so that they can be used in cross-validated measures of predictive performance, for example to compare different prediction models. Furthermore, we are interested in the specific value of the penalty parameter we obtain by maximizing (or

minimizing) some measure of predictive performance based on the approximated values for the leave-one-out estimates. The specific values of the approximations do not necessarily have to be close to the real values in order to be able to find a reasonable estimate for the penalty parameter.

In the remainder of this section, we will compute the cross-validated log-likelihood based on the real leave-one-out estimates as well as the approximated leave-one-out estimates in four different microarray data sets. The results will be compared and discussed for ridge as well as lasso regression.

3.1 Measure of predictive performance: leave-one-out cross-validated log-likelihood

In order to compare the approximated leave-one-out regression coefficients to the real ones in a meaningful way, we have to introduce a measure of predictive performance. In principle, every measure that depends on the regression parameter $\boldsymbol{\beta}$ could be used.

We choose to work with the cross-validated partial log-likelihood (*cvpl*), as given in [22]. The *cvpl* is given by the following sum:

$$cvpl(\lambda) = \sum_{i=1}^{n} l_i(\hat{\boldsymbol{\beta}}^{\lambda}_{-i}) \tag{35}$$

where $l_i(\boldsymbol{\beta})$ is the contribution of the i^{th} component to the likelihood and is given by

$$l_i(\boldsymbol{\beta}) = \sum_{\substack{j:t_j \leq t_i \\ j \neq i}} d_j \ln(1 - p_{ij}) + d_i \ln(p_{ii}) \tag{36}$$

with

$$p_{ij} = \frac{\exp\left(\boldsymbol{x}_i^T \boldsymbol{\beta}\right)}{\sum_{k:t_k \geq t_j} \exp\left(\boldsymbol{x}_k^T \boldsymbol{\beta}\right)}. \tag{37}$$

The original formula from [22] has already been modified so it works also when ties are present in the data.

In several datasets, for different values of λ, we will calculate the *cvpl*, either based on the real leave-one-out estimates (as found by the `R` package `penalized`) or on the approximated version of the $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$'s. We will compare the actual values and the optimal value of λ, found in these two different ways.

3.2 Several real-data examples

The approximation procedure will be tested on four gene expression data sets with a survival response. They will be referred to as Van de Vijver ([24], 295 subjects, 79 events, 4919 covariates, time to event is the actual survival time), Rosenwald ([15], 240 subjects, 138 events, 7399 covariates, time to event is the actual survival time), Wang([27], 286 subjects, 107 events, 22283 covariates, time to event is distant metastasis free survival) and Desmedt ([7], 198 subjects, 62 events, 22283 covariates, time to event is distant metastasis free survival).

3.2.1 Ridge regression

First, a Cox model with a ridge penalty is fitted on all four data sets. All covariates are penalized equally and since there is no intercept term in Cox regression (this term is incorporated in the baseline hazard), we have the model as given in equation (25) where $\boldsymbol{A}$ is just the identity matrix and can therefore be left out.

To find the right amount of penalization in the final model, the model performance (as given by the *cvpl*) is plotted for several values of λ and is maximized. In figure 1, four lines are plotted for every data set. The line corresponding to "*cvpl*" is the *cvpl* based on the real leave-one-out estimates, the line denoted by "*appr cvpl*" is the *cvpl* based on the approximated versions of the $\hat{\boldsymbol{\beta}}_{-i}^{\lambda}$'s as given in equation (28) and the line labeled "*appr int cvpl*" gives the *cvpl* based on approximations with an extra intercept term incorporated. Since in every data set the number of covariates is large compared to the number of individuals, all approximations are calculated using the γ-reparametrization trick. The fourth line, "*train cvpl*", is calculated using the *cvpl* as given above, but this time all $\hat{\boldsymbol{\beta}}_{-i}^{\lambda}$'s are replaced by $\hat{\boldsymbol{\beta}}^{\lambda}$, the estimate based on the complete data set.

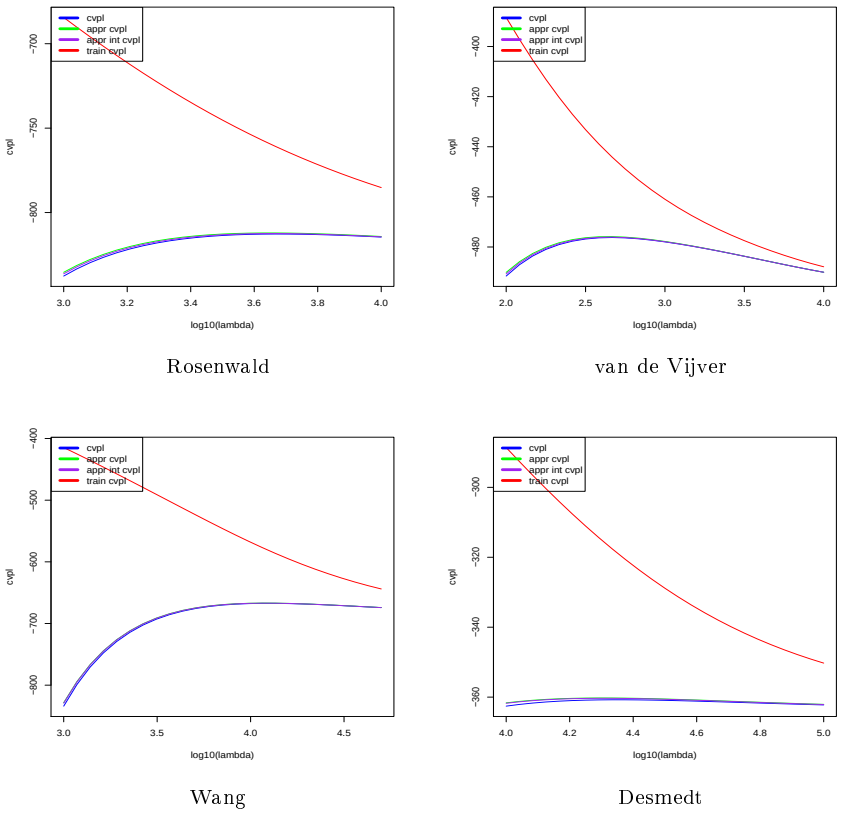


Figure 1: *cvpl* based on real leave-one-out estimates or approximations

From this figure it can be seen that the *cvpl* based on the approximations does closely follow the shape of the real *cvpl* and although the approximations are based on $\hat{\boldsymbol{\beta}}^{\lambda}$, the *cvpl* based on these approximations will apparently not lead to too small values of λ (which would result in an overtrained model).

Although the *cvpl* curves based on the approximations look very similar to the real *cvpl*, still differences are to be expected and to gain better insight in these differences the graphs are displayed in more detail in figure 2. In table 1, the λ values corresponding to the maximum of the three *cvpl*-functions are given.

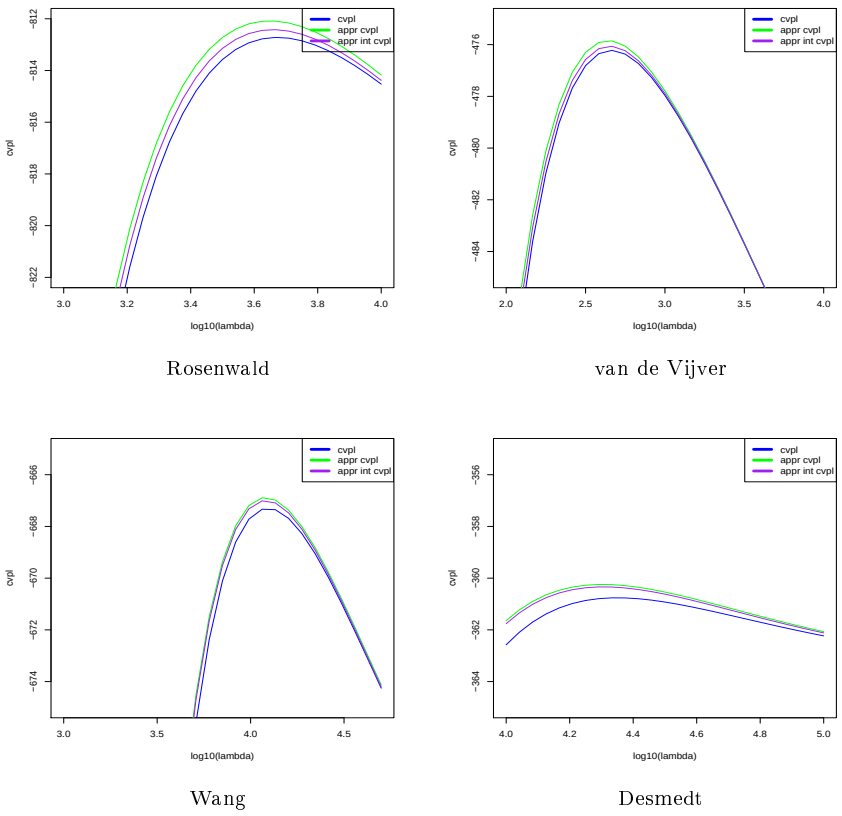


Figure 2: *cvpl* based on real leave-one-out estimates or approximations in more detail

From figure 2, we can conclude that for a certain λ, the function values of the approximated functions are very close to the real one. The approximation method with the extra intercept term involved does consistently better than the one without this term and is therefore the recommended approximation procedure. Since its values are so close to the real cross-validated values, the method can safely be used for comparison purposes.

From table 1 it can be seen that the optimal λ values found with the approximation method are also reasonably close to the values found by real leave-one-out cross-validation. Although the approximated λ may not be close to

Table 1: Optimal value for the penalty parameter in the ridge model

	real cvpl λ	appr cvpl λ	appr int cvpl λ
Rosenwald	4706.7	4457.3	4561.6
van de Vijver	458.5	438.3	448.3
Wang	12386	11997	12040
Desmedt	22376	20104	20265

the real λ in absolute value, the final model based on the approximated value of the penalty term will perform almost as good as the one based on the real cross-validated λ, as can be seen from table 2, where for every data set the real *cvpl* is calculated twice. Once based on the optimal λ value and once based on the approximated value. The very small differences in function values for λ's near the optima reflect rather "flat" maxima and the small differences also imply that the optimal λ found by our approximation method will result in a nearly optimal model.

Table 2: Difference in model performance for different values of λ

	λ	real $cvpl(\lambda)$
Rosenwald	4706.7	-812.7202
	4561.6	-812.7246
van de Vijver	458.5	-476.2204
	448.3	-476.2223
Wang	12386	-667.2978
	12040	-667.3034
Desmedt	22376	-360.7596
	20265	-360.7774

3.2.2 Lasso regression

For Cox regression with a lasso penalty, we made exactly the same figures and tables. From figure 3 we can conclude that the curves based on the approximated leave-one-out estimates again resemble the one based on the real estimates, but the differences are more substantial than in the previous analysis, which is even more visible in figure 4.

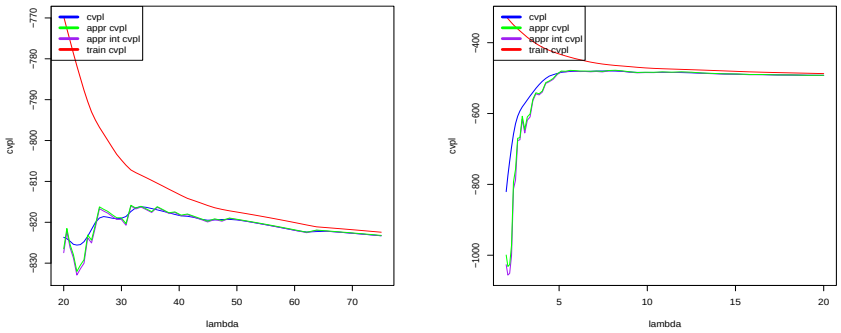
Although the exact function values will not always be estimated properly by the approximation methods, the curves do follow the shape of the real *cvpl* and the optimal value of the penalty parameter found by the approximation methods, are close to the real optimum, as can be seen from table 3. The optimal λ values found by the two different approximation procedures almost completely coincide and this time the procedure with the extra intercept term is not superior to the simpler procedure.

In table 4, two final models are again compared; the first one found by real cross-validation, the second one by the approximation procedure. As before, the two models are comparable (although to a lesser extent than in the model

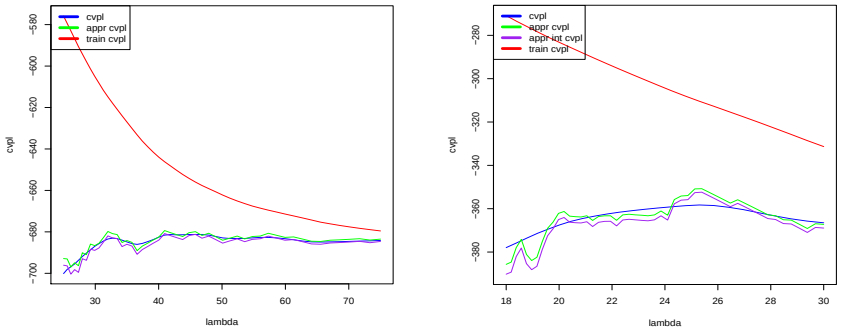
with a ridge penalty).

Table 3: Optimal value for the penalty parameter in the lasso model

	real cvpl λ	appr cvpl λ	appr int cvpl λ
Rosenwald	33.37	32.57	32.57
van de Vijver	7.70	7.61	7.61
Wang	45.85	47.14	47.14
Desmedt	25.39	25.36	25.36



Rosenwald van de Vijver

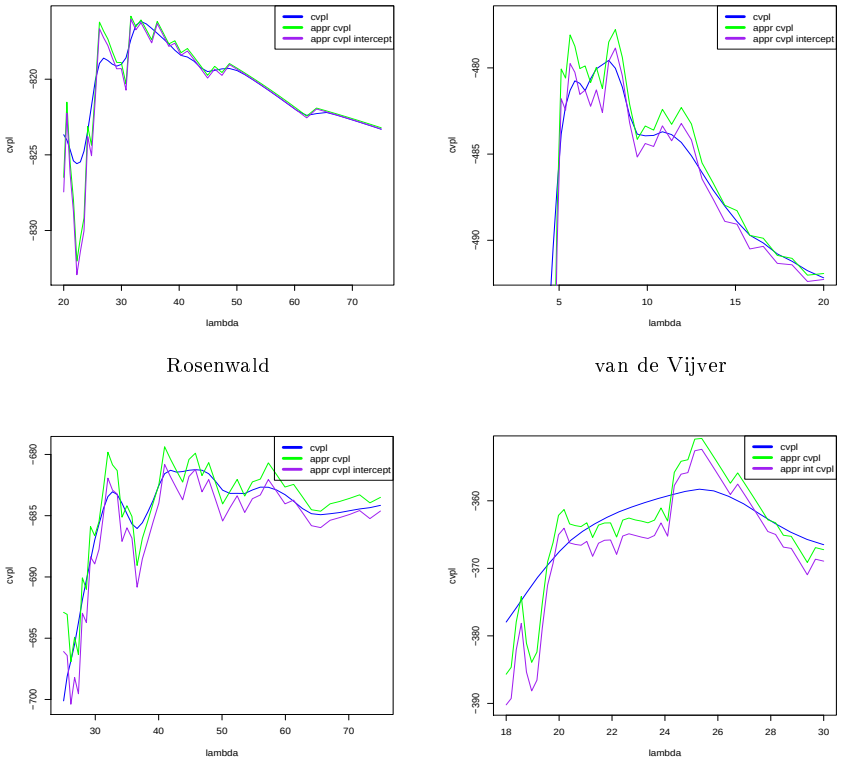


Wang Desmedt

Figure 3: *cvpl* based on real leave-one-out estimates or approximations

Table 4: Difference in model performance for different values of λ

	λ	real $cvpl(\lambda)$
Rosenwald	33.37	-816.188
	32.57	-816.392
van de Vijver	7.70	-479.4856
	7.61	-479.5521
Wang	45.85	-681.2446
	47.14	-681.3321
Desmedt	25.39	-358.2709
	25.36	-358.2719



Rosenwald van de Vijver

Wang Desmedt

Figure 4: *cvpl* based on real leave-one-out estimates or approximations in more detail

It can be seen from the graphs that the approximated curves are less smooth than the curve corresponding to the real *cvpl*. For this reason, more local optima are present and maximizing the approximated curve by a standard maximization procedure can result in wrong answers. To check whether an optimum found is really the global optimum, it is suggested to look at a rough plot of

the curve itself. The same suggestion holds for maximizing the real *cvpl* since this function is not unimodal either, but the chance of finding a suboptimal λ is higher if the approximated function is maximized.

3.3 Computation Time

The approximation method works well, but the efficiency of the method is as important as the outcomes. To get an indication of the amount of time that can be saved by using this approximation procedure in table 5 several calculation times are shown. For every data set the value of the *cvpl* for one specific value of λ (the optimal cross-validated estimate) is calculated, either by using the real leave-one-out regression coefficients or the approximated estimates. The corresponding computation times are displayed in this table.

Table 5: Computation time

	ridge regression		lasso regression	
	time (in seconds) needed to calculate:			
	$cvpl(\hat{\lambda})$	appr int $cvpl(\hat{\lambda})$	$cvpl(\hat{\lambda})$	appr int $cvpl(\hat{\lambda})$
Rosenwald	322.85	10.00	38.95	3.68
van de Vijver	380.09	10.64	46.46	4.76
Wang	1490.33	38.78	281.27	10.63
Desmedt	565.88	19.81	180.93	8.00

The values clearly indicate that the approximation procedure takes much less time than actual cross-validation. For ridge regression, the ratio between the computation times can be expected to vary little over different values of λ. For lasso regression, the amount of time saved will be much bigger for smaller values of λ. However, the approximation method will not give very reliable answers for these values of λ. Nevertheless, in optimizing the approximated *cvpl* this does not have to be a problem since the approximated *cvpl* is expected to have its maximum in approximately the right place.

Furthermore, most time is saved in data sets with a high number of individuals and covariates (like the Wang data set), especially for the lasso approximation procedure.

In the introduction, the claim was made that with this approximation procedure double leave-one-out cross-validation would become practically feasible. To get an indication on how much time double LOOCV will take using the approximation procedure, we look at the most time-consuming analysis (performing ridge regression on the Wang data set, with 286 observations) and assume that the procedure that maximizes the *cvpl* needs 20 steps to converge to the optimum. Real double LOOCV will take $286 \times 20 \times 1490.33$ seconds, which equals 98.7 days. The approximation method on the other hand would take $286 \times 20 \times 38.78$ seconds, the equivalent of 2.6 days. So where real leave-one-out cross-validation will take more than 3 months, the approximated version will give answers in less than 3 days. Of course, this is just an indication, but it does show that performing an approximated version of double leave-one-out

cross-validation could be worth considering.

3.4 Influence of n, λ and amount of censoring on approximations

In this last part of the result section, some remarks will be made on the applicability of the approximation method under different circumstances.

All four data sets used to demonstrate the usefulness of the approximation method are large data sets, where the number of individuals is in between 200 and 300. The method is supposed to work well exactly in this situation, since the leave-one-out estimates $\hat{\boldsymbol{\beta}}^{\lambda}_{-i}$'s are closer to $\hat{\boldsymbol{\beta}}^{\lambda}$ when there are more observations and the error term in the Taylor approximation diminishes in that case. Although it is difficult to find out if the method will work even better for larger data sets since we did not have access to much larger data sets and extending a data set by for example bootstrapping will result in many ties and will probably not be representative for large real data sets, examining the method on smaller data sets was possible.

In order to create multiple smaller data sets, we took respectively the first 25, 50, 100, 150 and 200 observations of the Van de Vijver data set and used these data sets to test whether our approximation procedure (in ridge as well as lasso regression) gave less reliable answers in smaller data sets. The function values of the approximated *cvpl* for ridge regression are already very close to the real values in the data set with 50 individuals and the approximated optimal value of λ will also result in a model which comes close to the optimal model (based on LOOCV) in terms of model performance. When the number of individuals increases, the approximated λ values get even closer to the real values. So we observe exactly the behavior we expected. In table 6 the real and approximated values of the tuning parameters are given for every data set we constructed.

Table 6: Values for the penalty parameter by varying values of n

size of	ridge regression		lasso regression	
data set	real $\hat{\lambda}$	appr $\hat{\lambda}$	real $\hat{\lambda}$	appr $\hat{\lambda}$
25	554.7	248.6	3.83	3.85
50	922.3	817.7	6.69	6.27
100	283.4	265.4	4.28	3.69
150	264.3	252.5	9.06	8.81
200	268.2	257.6	5.82	5.63
295	458.5	448.3	7.70	7.61

From the table it can be seen that for lasso regression approximately the same conclusions can be drawn as for ridge regression, except for the smallest values of n. By looking only at the approximated values of λ, the approximation method for lasso regression seems to perform very well for small data sets. Unfortunately, these values are a little misleading, since for the data sets of size $n = 25$ and $n = 50$ the null model was found to be the best model by real LOOCV and if all coefficients equal zero, the approximation method will be

very accurate. Another remark that should be made is that the actual function values of the approximated *cvpl* for lasso models can be rather inaccurate when the function is calculated for small λ-values.

This last observation holds in general: the approximation procedure works better for larger values of λ. The higher the value of λ, the more restricted the possible values of the regression coefficients and for that reason the leave-one-out estimates will be closer to the penalized maximum-likelihood-estimate and the approximations will be more accurate. In ridge regression, the accuracy of the approximations is also dependent on the actual λ-value, but to a lesser extent.

We also examined the influence of the amount of censoring on the approximation procedure. One may argue that more censoring leads to a smaller number of influential observations (i.e. observations whose in- or exclusion result in substantial changes in the regression parameter) and therefore leaving out one of these will result in a bigger change in the regression coefficients which could result in worse approximations. To test this, we again used the Van de Vijver dataset (295 observation, 79 events, maximum follow-up time 18.3 years). To vary the amount of censoring, we altered the maximum follow-up time by taking a new time $t_{max} < 18.3$ and rearranged the data in such a way that all tuples $(t_i, d_i, \boldsymbol{x}_i)$ with $t_i > t_{max}$ were set to $(t_{max}, 0, \boldsymbol{x}_i)$. Different values of t_{max} were chosen resulting in four new data sets with respectively 68, 46, 34 and 16 events. No clear patterns were observed for the first three data sets, but the approximation method's accuracy seemed to decline a little in the data set with just 16 events, suggesting that too few events can lead to a worse approximation result. The real and approximated values of λ can be found in table 7.

Table 7: Values for the penalty parameter by varying effective dimensions

effective	ridge regression		lasso regression	
dimension	real $\hat{\lambda}$	appr $\hat{\lambda}$	real $\hat{\lambda}$	appr $\hat{\lambda}$
16	80.4	68.7	6.22	6.16
34	242.6	234.7	4.71	4.81
46	359.6	353.0	4.96	4.92
68	344.9	339.6	6.65	6.75
79	458.5	448.3	7.70	7.61

4 Discussion

In this article a method has been described that approximates leave-one-out cross-validation results. The method can be used in generalized linear models as well as in Cox' proportional hazards model and works with ridge and lasso penalty terms. It has been shown that the method performs well in finding the optimal penalty parameter and works efficiently.

When the sample size increases, the accuracy of the approximation method improves. So especially in situations where regular leave-one-out cross-validation can be very time-consuming, this approximation method will perform well in finding the appropriate amount of penalization and can also be used for model validation.

In this report, only leave-one-out cross-validation has been studied but a similar method could also be used to approximate k-fold cross-validation. Another generalization of the method would be to extend the method to work with different penalty terms as well. One could for example think of using this method in combination with elastic net models.

References

[1] Bøvelstad HM. *et al.* (2007) Predicting survival from microarray data - a comparative study. *Bioinformatics,* **23**(16), 2080-2087

[2] Cawley CC, Talbot NLC. (2004) Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks* **17** 1467-1475

[3] Celisse A, Robin S. (2008) Nonparamtric density estimation by exact leave-p-out cross-validation. *Computational Statistics and Data Analysis*, **52**, 2350-2368

[4] le Cessie S, van Houwelingen JC. (1992) Ridge estimators in logistic regression. *Applied Statistics,* **41**, 191-201

[5] Cook RD, Weisberg S. *Residuals and influence in regression*, Chapman and Hall, New York, 1982.

[6] Cox DR. (1972) Regression models and life-tables. *J.R. Statist. Soc. B,* **34**, 187-220

[7] Desmedt C. *et al.* (2007) Strong time-dependency of the 76-gene prognostic signature for node-negative breast cancer patients in the transbig multi-centre independent validation series. *Clin. Cancer Res.* **13** 3207-3214

[8] Gui J, Li H. (2005) Penalized cox regression analysis in the high-dimensional and low-sample size settings, with applications to microarray gene expression data. *Bioinformatics,* **21**(13), 3001-3008

[9] Hager WW. (1989) Updating the inverse of a matrix. *Society for Industrial and Applied Mathematics,* **31**, 221-239

[10] Haibe-Kains B. *et al.* (2008) A comparative study of survival models for breast cancer prognostication based on microarray data: does a single gene beat them all? *Bioinformatics,* **24**(19), 2200-2208

[11] van Houwelingen JC *et al.* (2006) Cross-validated Cox regression om microarray gene expression data. *Statistics in Medicine,* **25**, 3201-3216

[12] Huang J, Harrington D. (2002) Penalized partial likelihood regression for right-censored data with bootstrap selection of the penalty parameter. *Biometrics,* **58**, 781-791

[13] McCullagh P, Nelder JA. (1989) *Generalized Linear Models*, 2nd edition. London: Chapman and Hall.

[14] Myers RH. *Classical and modern regression with applications*, 2nd edn, PWS-Kent, Boston, 1990.)

[15] Rosenwald A. *et al.* (2002) The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma. *New England Journal of Medicine*, **346**(25), 1937-1947

[16] Schumacher M. *et al.* (2007) Assesment of survival prediction models based on microarray data. *Bioinformatics* **23** 1768-1774

[17] Tibshirani R. (1996) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, **58**(1), 267-288

[18] Tibshirani R. (1997) The lasso method for variable selection in the Cox model. *Statistics in Medicine,* **16** 385-395

[19] Tran MN. (2009) Penalized maximum likelihood principle for choosing ridge parameter. *Communication in Statistics - Simulation and Computation,* **38**, 1610-1624

[20] Ueki M, Fueda K. (2010) Optimal tuning parameter estimation in maximum penalized likelihood models *Annals of the Institute of Statistical Mathematics,* **62** 413-438

[21] Verweij PJM *et al.* (1998) A goodness-of-fit test for Cox's proportional hazards model based on martingale residuals. *Biometrics,* **54**, 1517-1526

[22] Verweij PJM, van Houwelingen HC. (1993) Cross-validation in survival analysis. *Statistics in Medicine,* **12**, 2305-2314

[23] Verweij PJM, van Houwelingen HC. (1994) Penalized likelihood in Cox regression. *Statistics in Medicine,* **13**, 2427-2436

[24] Van de Vijver M. *et al.* (2002) A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine*, **347**(25), 1999-2009

[25] Wang H. *et al.* (2009) Shrinkage tuning parameter selection with a diverging number of parameters. *J.R. Statist. Soc. B,* **71** 671-683

[26] Wang S. *et al.* (2008) Doubly penalized Buckley-James method for survival data with high-dimensional covariates. *Biometrics,* **64** 132-140

[27] Wang Y. *et al.* (2005) Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *Lancet* **365** 671-679

[28] van Wieringen WN. *et al.* (2009) Survival prediction using gene expression data: A review and comparison. *Computational statistics & data analysis* **53** 1590-1603

[29] Zhang Y. *et al.* (2010) Regularization parameter selections via generalized information criteria. *Journal of the American statistical association,* **105** 312-323