CARL VON OSSIETZKY universität OLDENBURG

Gradient-Free Gradient Boosting

Von der Fakultät für Mathematik und Naturwissenschaften der Carl von Ossietzky Universität Oldenburg zur Erlangung des Grades und Titels eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

angenommene Dissertation

von Herrn Tino Werner

geboren am 23.10.1993 in Varel

Gutachter: Prof. Dr. Peter Ruckdeschel

Weiterer Gutachter: Prof. Dr. Matthias Schmid

Datum der Einreichung: 26. September 2019

Tag der Disputation: 8. Januar 2020

Zusammenfassung

Moderne Techniken des maschinellen Lernens sind im Zeitalter der Digitalisierung nicht mehr wegzudenken und ermöglichen die effiziente Analyse großer Datenmengen. Die quasi unbegrenzten Einsatzgebiete solcher Lernmethoden umfassen auch Anwendungen im Kontext der Betrugsdetektion. Angesichts der begrenzten Kapazitäten von Finanzämtern ist eine rigorose Überprüfung aller eingegangenen Steuererklärungen nicht möglich, sodass möglichst geschickt eine entsprechende Auswahl im Vorfeld getroffen werden muss.

Dieses Vorgehen nennt sich „Risk-based auditing” und hat sich in etlichen Studien (z.B. Alm et al. [1993], Gupta and Nagadevara [2007]) als deutlich effizienter erwiesen als eine zufällige Auswahl von zu überprüfenden Steuererklärungen. Das zugrunde liegende Problem lässt sich auf zwei Arten mathematisch erfassen. Wird die Auswahl der zu überprüfenden Steuererklärungen anhand der Wahrscheinlichkeit, dass eine steuerpflichtige Person zu wenig Einkommen angibt, getroffen, so führt dies auf ein sogenanntes binäres Rankingproblem. Soll die Entscheidung allerdings an der zu erwartenden Schadenhöhe getroffen werden, so muss ein stetiges Rankingproblem gelöst werden.

Eine weitere Schwierigkeit bei hochdimensionalen Daten ergibt sich aus der Vielzahl der möglichen Prädiktoren. In diesem Kontext sind Lernverfahren, die eine Modellwahl treffen, unabdingbar, was in der Regel durch die Optimierung regularisierter empirischer Risiken oder durch den Einsatz von sogenannten Forward-Selektions-Techniken erreicht wird. Als besonders effizient haben sich Gradienten-Boosting-Algorithmen herausgestellt, die zu den Forward-Selektions-Techniken zählen und iterativ einfache Modelle, sogenannte Baselearner, kombinieren. Besonders herauszustellen ist der L_2-Boosting-Algorithmus (Bühlmann and Yu [2003]), welcher den quadratischen Abstand der vorhergesagten Responses zu den wahren Responses durch sparsame Modelle minimiert und sowohl attraktive theoretische Eigenschaften (Bühlmann and Yu [2003], Bühlmann [2006], Bühlmann and Van De Geer [2011]) besitzt als auch höchst effizient implementiert wurde (Hothorn et al. [2017]).

Allerdings neigen selbst derart geschickte Verfahren dazu, in der Praxis mehr Prädiktoren als nötig auszuwählen, was man als „Overfitting” bezeichnet, wodurch nicht nur die Qualität des Modells und damit der Prädiktionen, sondern auch die Interpretierbarkeit leidet. Dieser Schwierigkeit kann begegnet werden, indem man mehrere, auf unterschiedlichen Teilmengen des Datensatzes berechnete Modelle betrachtet und die aggregierten Selektionshäufigkeiten der Prädiktoren geeignet kombiniert. Anschließend werden durch die Vorgabe eines Cutoffs am Ende die Prädiktoren selektiert, die eine hinreichend große aggregierte Selektionshäufigkeit besitzen. Dieses Verfahren nennt sich Stabilitätsselektion und wurde ursprünglich in Meinshausen and Bühlmann [2010] für Lasso-artige Verfahren entwickelt, in Hofner et al.

[2015] aber für die Anwendung auf Boosting-Modelle übertragen. Zusätzlich zu der weiteren Ausdünnung der ausgewählten Prädiktormenge stabilisiert das Verfahren diese, d.h., auch unter anderen Daten, die aber der gleichen Verteilung wie die gesehenen Daten entstammen, würde die dort berechnete Prädiktormenge kaum davon abweichen.

Obgleich bereits Lernverfahren für binäre Rankingprobleme existieren, die zu Modellselektion fähig sind, so gibt es für stetige Rankingprobleme bisher nur einen Ansatz, der einen Baum-artigen Algorithmus verwendet (Clémençon and Achab [2017]). Boosting-Techniken für stetige Rankingprobleme sind aufgrund der Nicht-Regularität der zugehörigen Verlustfunktionen nicht direkt einsetzbar.

Diese Arbeit beschäftigt sich mit der allgemeineren Frage, wie man für hochdimensionale Daten effektiv Lösungen für statistische Lernprobleme, welche auf irregulären Verlustfunktion basieren, finden kann, die zudem interpretierbar sind, stabile Modellselektion ermöglichen sowie effizient implementiert werden können. Der Kern dieser Arbeit besteht darin, Spalten- und Zeilenselektionsverfahren einheitlich mathematisch durch ein neues Framework zu erfassen. Dazu definieren wird das sogenannte „Spaltenmaß”, welches auf den Spalten einer Datenmatrix definiert ist und die Wichtigkeit jeder Spalte bemisst. Da der Spaltenselektion ein Optimierungsproblem bezüglich einer gegebenen Verlustfunktion zugrunde liegt, sind Spaltenmaße abhängig von der konkreten Verlustfunktion. Das führt dazu, dass auch die von uns postulierten theoretischen Spaltenmaße abhängig von der Verlustfunktion sind.

Da das stetige Rankingproblem durch den bedingten Erwartungswert von der abhängigen Variable gegeben der Prädiktoren gelöst wird, wird durch L_2-Boosting bereits eine Lösung des Problems genähert. Das naive Vorgehen, dass man das L_2-Boosting-Modell direkt als Lösung des stetigen Rankingproblems verwendet, berücksichtigt allerdings nicht den Umstand, dass das theoretische Spaltenmaß bezüglich der quadratischen Verlustfunktion von dem theoretischen Spaltenmaß bezüglich einer Ranking-Verlustfunktion abweicht. Mathematisch lassen sich Spalten, die durch die Lösung bezüglich einer Verlustfunktion $\tilde{L}$ selektiert werden, aber nicht durch die Lösung bezüglich einer Verlustfunktion L gewählt werden können, als Singulärteile zwischen (empirischen) Spaltenmaßen auffassen.

Darüber hinaus können wir nachweisen, dass die Verwendung von paarweisen Surrogat-Verlustfunktionen für stetige Rankingprobleme und eines daraus resultierenden Boosting-Algorithmus kein sinnvolles Vorgehen darstellt, sowohl hinsichtlich der Qualität der Ergebnisse als auch der benötigten Rechnerkapazitäten.

Wir begegnen Singulärteilen zwischen Spaltenmaßen durch einen speziellen auf L_2-Boosting basierenden Boosting-Algorithmus, den wir SingBoost nennen und der die Möglichkeit bietet, auch solche Spalten in das Modell aufzunehmen. Wir beweisen statistische Eigenschaften

des Algorithmus und implementieren diesen in R. Da auch SingBoost bei hochdimensionalen Daten zum Overfitting neigt, entwickeln wir eine spezielle Stabilitätsselektion, die verlustabhängig durch eine geeignete Gittersuche das optimale stabile Modell aus der Aggregation verschiedener, auf Unterstichproben berechneter Modelle, bestimmt. Diese ist auch für den Fall verrauschter Daten einsetzbar. Durch Verknüpfung mit SingBoost ergibt sich der Algorithmus CMB-3S und durch Kombination mit einer Kreuzvalidierung der Algorithmus CV.CMB-3S, den wir sehr flexibel in R implementieren. Als Nebenprodukt bei der Stabilitätsselektion ergeben sich Zeilenmaße, die analog zu den Spaltenmaßen auf den Zeilen der Datenmatrix definiert sind.

Unsere Beiträge zu Rankingproblemen reichen von einer systematischen Untersuchung der verschiedenen Rankingprobleme inklusive ihrer Verlustfunktionen über Vorschläge für Influenzkurven und einer Diskussion existierender Ansätze. Darüber hinaus zeigen wir, wie man die in der Arbeit relevanten Verlustfunktionen schnell auswerten kann. Am Ende wenden wir unsere Algorithmen auf das sogenannte harte stetige Rankingproblem an und können zeigen, dass wir den L_2-Boosting-Ansatz hinsichtlich des kreuzvalidierten Testverlustes schlagen können. Ein wesentlicher Beitrag besteht darin, dass wir zeigen, dass Rankingfunktionale elizitierbar sind, sodass verschiedene konkurrierende Ranking-Modelle objektiv miteinander vergleichbar sind, selbst in dem Fall, dass das optimale Ranking nicht eindeutig ist.

Wir erweitern unsere Konzepte für den Fall multivariater Responses und zeigen, dass ein SingBoost-Algorithmus auch in diesem Kontext über analoge statistische Eigenschaften verfügt wie im univariaten Fall. Wir zeigen, dass wir auch unterschiedlichste Lernverfahren wie Ausreißerdetektion, Kovarianzschätzung und sogenanntes „Consensus Ranking", bei dem partielle Rankings zu einem gemeinsamen Ranking kombiniert werden, in unser Framework einbetten können. Wir identifizieren robuste Kovarianz-Schätzverfahren und ein spezielles robustes Regressionsverfahren als verallgemeinerte Boosting-Varianten und stellen prototypisch ein verallgemeinertes L_2-Boosting auf.

Für den Fall heterogener Daten, beispielsweise durch strukturell fehlende Werte oder Kontamination, entwickeln wir Ideen, wie wir unsere Algorithmen anpassen können, sodass sie auch auf derartige Daten anwendbar sind.

Über die Verknüpfung von sparsamen Modellen, Stabilität und Robustheit durch unser Framework hinaus liefern wir theoretische Resultate, die zeigen, dass regularisierte Regressionsverfahren wie das Lasso oder das Adaptive Lasso, wenngleich die zugehörigen statistischen Funktionale hochgradig nichtlinear sind, sich asymptotisch linear entwickeln lassen, sodass sich diese Verfahren als Aggregation über die Zeilen und über die Spalten der Datenmatrix auffassen lassen.

Abstract

Modern machine learning techniques are indispensable in the era of digitization and enable the efficient analysis of big data. The nearly limitless applications of such learning methods include the field of fraud detection. Given the limited capacities of tax offices which restrict the number of income tax statements which can be reviewed, one needs to make a sophisticated selection of income tax statements for review beforehand.

This procedure is called ”risk-based auditing” and has been shown to be much more efficient than a simple random selection of income tax statements (see for example Alm et al. [1993], Gupta and Nagadevara [2007]). The underlying problem can be described mathematically in two ways. If the selection is based on the probability that the respective person misreports her or his income, we get a so-called binary ranking problem. If the decision is based on the expected amount of damage, one has to solve a continuous ranking problem.

Another issue when analyzing high-dimensional data arises from the large number of possible predictors. In this setting, learning algorithms which include model selection are indispensable which generally either corresponds to the optimization of regularized empirical risks or to so-called forward selection techniques. A very efficient class of models are Gradient Boosting algorithms which are special forward selection procedures that iteratively combine simple models, referred to as baselearners. A Boosting algorithm which has to be highlighted is the L_2-Boosting algorithm (Bühlmann and Yu [2003]) which minimizes the quadratic distance of the predicted responses to the true responses by means of sparse models and which has attractive theoretical properties (Bühlmann and Yu [2003], Bühlmann [2006], Bühlmann and Van De Geer [2011]), besides being implemented very efficiently (Hothorn et al. [2017]).

However, even these sophisticated learning strategies tend to select more predictor variables than necessary in practical applications which is referred to as ”overfitting” which not only decreases the quality of the resulting model and therefore of the predictions but also causes the model to be less interpretable. To handle this issue, one considers different models which have been computed on different subsets of the data and suitably combines the aggregated selection frequencies of the predictors. By defining some cutoff, one finally selects only those predictor variables whose aggregated selection frequency is sufficiently high. This strategy is referred to as Stability Selection and originates from Meinshausen and Bühlmann [2010] where it has been applied to Lasso-type algorithms. It has been generalized to the application to Boosting models in Hofner et al. [2015]. Apart from reducing the number of predictors again, the predictor set gets stabilized by the Stability Selection, i.e., when concerning other observations from the same distribution as the given ones, the corresponding predictor set

would only marginally differ from the stable predictor set.

Despite there already exist learning algorithms for binary ranking problems, even with model selection, there do not yet exist strategies for continuous ranking problems but a tree-type approach (Clémençon and Achab [2017]). The application of Boosting algorithms to continuous ranking problems is not directly possible due to the non-regularity of the corresponding loss functions.

This work concerns about the more general question how one could effectively find interpretable solutions for statistical learning problems on high-dimensional data, based on non-regular loss functions, that perform stable model selection and that can be implemented efficiently. The heart of this thesis consists of a unified mathematical treatment of row and column selection procedures. We propose a so-called "column measure" which is defined on the columns of a data matrix and which quantifies the importance of each column. Since column selection is based on an optimization problem w.r.t. a given loss function, the column measures depend on the concrete loss function. Therefore, even the postulated theoretical column measures depend on the loss function.

Since continuous ranking problems are solved by the conditional expectation of the response variable given the predictor variables, L_2-Boosting already approximates a solution for this problem. However, the naïve approach to use L_2-Boosting models directly as solution of continuous ranking problems does not respect the issue that the theoretical column measure w.r.t. the squared loss differs from the theoretical column measure w.r.t. some ranking loss. The set of columns which are selected when solving the problem w.r.t. some loss function $\tilde{L}$ and which are not selected for some underlying loss function L can be mathematically described as singular parts between the (empirical) column measures.

Additionally, we show that using pair-wise surrogate loss functions for continuous ranking problems to get a corresponding Boosting algorithm is not meaningful regarding the quality of the results and the computational costs.

To handle singular parts between column measures, we propose a special algorithm which we call SingBoost and which is based on L_2-Boosting and allows to include even those columns. We prove statistical properties of SingBoost and provide an implementation in R. Since SingBoost also tends to overfit in the presence of high-dimensional data, we propose a loss-based Stability Selection which includes the aggregation of models computed on different subsets of the data and which computes the optimal stable model by the application of a grid search. This Stability Selection can also handle noisy data. Combining it with SingBoost results in the algorithm CMB-3S and combining this again with a cross validation, we get the algorithm CV.CMB-3S of which we provide a very flexible implementation in R. As a by-product of the Stability Selection, we get row measures which are defined on the rows of the data matrix, analogously to the column measures.

Our contributions to ranking problems range from a systematic investigation of the different types of ranking problems, including their loss functions, to proposals for influence curves and a discussion of existing approaches. Additionally, we show how such ranking loss functions can be evaluated efficiently. We apply our algorithms to the so-called hard continuous ranking problem at the end and show that we can outperform the L_2-Boosting approach w.r.t. the cross-validated test loss. A major contribution is that we show that ranking functionals are elicitable so that different competing ranking models can be compared in an objective way, even in the case that the optimal ranking is not unique.

We extend our concepts to the case of multivariate responses and show that a SingBoost algorithm would have analogous statistical properties as in the case of univariate responses. We show that we can embed inherently different learning algorithms like outlier detection, covariance estimation and so-called ”consensus ranking” which combines different partial rankings to an overall ranking into our framework. We identify robust covariance estimation procedures and a special robust regression procedure as generalized Boosting variants and provide a prototype of a Generalized L_2-Boosting.

In the case of heterogeneous data, for example due to structural missings or contamination, we provide ideas how we could modify our algorithms to handle those peculiarities.

Apart from the connection of sparse models, stability and robustness by our framework, we provide theoretical results that show that regularized regression procedures like the Lasso or the Adaptive Lasso, although the corresponding statistical functionals are highly non-linear, can be asymptotically linearly expanded so that these algorithms can be identified with an aggregation over both the rows and the columns of the data matrix.

Vorwort

Eine Konstante, die sich zunächst durch meine Schul- und später durch meine Studienlaufbahn zog, war die Irritation anderer Personen über mein Alter. Selbst die in meinen Augen gar nicht mehr so ungewöhnliche Tatsache, dass ich noch als 25-Jähriger meine Dissertation eingereicht habe, scheint immer noch für Verwunderung zu sorgen.

Diesen Weg haben mir insbesondere meine Eltern ermöglicht, die sich dafür eingesetzt haben, dass ich trotz vorgebrachter Bedenken und Widerstände vorzeitig eingeschult werden konnte. Mit der Unterstützung meiner ersten Klassenlehrerin, Talea ter Hell-Zorn, konnte ich im Oktober 2000 noch vor meinem siebten Geburtstag in die dritte Klasse überspringen.

Bezüglich meiner Schulzeit ist hier meine Klassenlehrerin der 9. und 10. Klasse, Beate Leddin, hervorzuheben. Auch verdient Erwähnung, dass der Lehrer meines Mathematik-Kurses in der Oberstufe, Hartwig Brüning, als Erster die Idee hatte, dass ich anschließend Mathematik studieren soll.

Den ersten Anstoß in Richtung Promotion habe ich während meines Masterstudiums von Dr. Frank Schöpfer, der meine Bachelor- und Masterarbeit betreut hat, bekommen. Es sei hier die Anekdote erwähnt, dass Prof. Dr. Peter Ruckdeschel mich am 25.1.2016 vorsichtig gefragt hat, ob ich mir generell vorstellen könne, zu promovieren, was ich noch sehr entschlossen verneint habe. Die ganze Idee musste lange in mir reifen, bis ich mich dann doch für eine Promotion entschieden habe.

An dieser Stelle möchte ich mich bei Prof. Dr. Peter Ruckdeschel für die Möglichkeit der Promotion bedanken, für die Ideen, die ich ausarbeiten und erweitern durfte sowie für die Hilfestellung bei der Gestaltung von Konferenzbeiträgen und Einreichungen für Publikationen.

Ein Dank geht an Prof. Dr. Matthias Schmid für die Zweitbegutachtung der Dissertation sowie an Prof. Dr. Marcus C. Christiansen und Prof. Dr. Thorsten Dickhaus für die Bereitschaft, der Prüfungskommission beizuwohnen.

Abschließend möchte ich noch M. Sc. Jörg Thomas Best erwähnen, mit dem ich mir ein Büro geteilt habe und der mir die Zeit mit dem einen oder anderen nicht-mathematischen Gespräch bereichert hat.

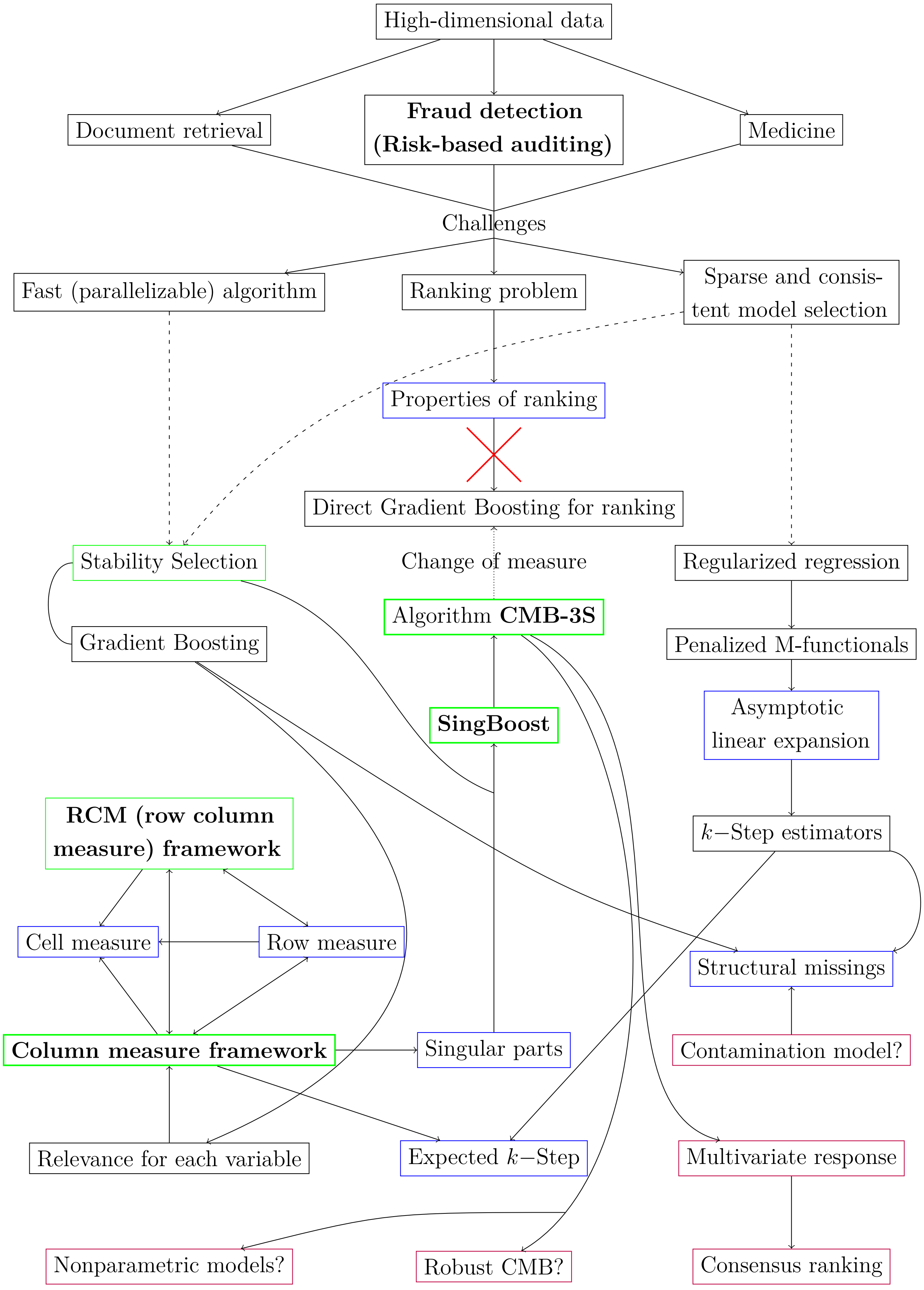
How to read this thesis

To help navigating through this thesis and its parts and chapters, we provide illustrative diagrams.

The diagram on the next page gives an overview of the structure of this thesis. At the beginning of each part, the same diagram will be presented, but only the relevant substructures of the respective part will be highlighted.

Green boxes indicate major contributions of this thesis while blue boxes indicate minor contributions. Purple boxes refer to ideas, first approaches or open questions for future research.

After this bird's eye's view figure is placed at the beginning of each part, a more detailed figure, zooming into the respective part's topics follows.


High-dimensional data
Document retrieval
Fraud detection (Risk-based auditing)
Medicine
Challenges
Fast (parallelizable) algorithm
Ranking problem
Sparse and consistent model selection
Properties of ranking
Direct Gradient Boosting for ranking
Change of measure
Stability Selection
Gradient Boosting
Algorithm CMB-3S
SingBoost
Regularized regression
Penalized M-functionals
Asymptotic linear expansion
k−Step estimators
RCM (row column measure) framework
Cell measure
Row measure
Structural missings
Column measure framework
Singular parts
Contamination model?
Relevance for each variable
Expected k−Step
Multivariate response
Nonparametric models?
Robust CMB?
Consensus ranking

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

We do not list all defined variables, abbreviations etc. here, but only the recurring or important ones. We made use of hebrew symbols due to the shortness of the (classical) greek alphabet.

General notation

$(\cdot)^T$	Transposed object
$\langle f, g \rangle$	Inner product in L_2 in the Temlyakov scheme
$\langle f, g \rangle_{(n)}$	Empirical version of $\langle f, g \rangle$ in the Temlyakov scheme
∂	(Partial) Derivative
∇	Gradient (vector of partial derivatives)
0_k	$k-$dimensional vector only containing zeroes
$\hat{\beta}^{(0)}$	Initial coefficient in Boosting algorithms
$\hat{\beta}^{(k)}$	Estimated coefficient after the $k-$th Boosting iteration
$\hat{\beta}^{init}$	Initially computed coefficient vector in multi-stage procedures
B	Number of Bootstrap samples / subsamples
B^{sing}	Number of SingBoost models computed in Column Measure Boosting
$\beth$	Coefficient matrix
$\gimel$	RCM matrix
$\hat{\gimel}^{stab}$	Stable empirical RCM matrix
d_B	Bouligand functional derivative
d_F	Fréchet or bounded functional derivative
d_G	Gâteaux or weak functional derivative
d_H	Hadamard or compact functional derivative
$d_{\mathcal{R}}$	General $\mathcal{R}-$derivative
δ_x	Dirac measure at x
Δ	Mesh size for the grid π_{grid}

$\daleth$	Coefficient matrix corresponding to the baselearner in multivariate Boosting
$e_j \in \mathbb{R}^p$	$j-$th unit vector of dimension p
ϵ	Error term
E	Error matrix
η_θ	General notation for a partial influence function at P_θ
F	General notation for a cumulative distribution function
$\gamma^*(T,P)$	Gross error sensitivity of the functional T w.r.t. distribution P
h	Cardinality of "clean" subsets in concentration steps
I_{θ_0}	Fisher information at θ_0
$\hat{j}_k$	Selected coefficient in the $k-$th Boosting iteration
J_λ	Penalty term depending on λ
J_λ^m	Element of an approximating sequence of penalty terms
κ	Step size / learning rate in Boosting algorithms
λ	Regularization parameter
Λ_{θ_0}	L_2-derivative / score at θ_0
m_{iter}	Number of Boosting iterations
M	Frequency of singular iterations in SingBoost
M^{final}	Frequency of singular iterations in SingBoost applied to the stable predictor set
n	General notation for the number of observations (rows) of a data set
n_{CC}	Number of complete rows in the data
n_{sing}	Number of observations used for SingBoost
n_{CMB}	Number of observations used for Column Measure Boosting
n_{train}	Number of observations of the training set
n_{val}	Number of observations of the validation set
n_{test}	Number of observations of the test set
o, $\mathcal{O}$	Landauer $o-$ and $\mathcal{O}-$terms
o_P, $\mathcal{O}_P$	Stochastic $o-$ and $\mathcal{O}-$terms
p	Number of variables (columns) of a data set
p_θ	Parametric density at θ
P_θ	Element of a parametric distribution family
π_{thr}	Threshold that defines the stable set of predictors
π	General notation for a permutation
π^*	Consensus ranking
q	Number of variables chosen by each Boosting model in Hofner's Stability Selection; number of final variables chosen in our Stability Selection
r	Contamination radius
ψ_θ	General notation for an influence function at the model P_θ
$r^{(k)}$	Residual vector after the $k-$th Boosting iteration

$\hat{R}_n^m f$	Residual after the $m-$th iteration of a greedy algorithm based on n observations in the Temlyakov scheme
R	Residual matrix in multivariate regression
s^0	True sparsity, i.e., number of non-zero entries of the true coefficient vector
$\hat{s}$	Number of non-zero entries of the estimated coefficient vector
$\hat{s}^{init}$	Number of non-zero entries of an estimated initial coefficient
$\hat{s}^{(b)}$	Score assigned to the Boosting model fitted on the $b-$th subsample
S_n	General notation for an estimator based on n observations
S_n^k	$k-$Step estimator
$\mathbb{E}_{\hat{\nu}_n}[S_n^k]$	Expected $k-$Step estimator w.r.t. some empirical column measure $\hat{\nu}_n$
$S_{n,J}^k$	$k-$Step estimator based on the columns with index in J
Σ	General notation for a covariance matrix
$\tau(\cdot,\cdot)$	Kendall's τ concordance measure between two vectors
V	Number of different partitions of the data into training, validation and test set
φ	General notation for the derivative of a parametric loss function w.r.t. the parameter
φ_θ	Short notation for $\varphi(\cdot,\theta)$
φ^λ	General notation for the derivative of a regularized parametric loss function w.r. t. the parameter
$\hat{w}^{(b)}$	Weight assigned to the Boosting model fitted on the $b-$th subsample
χ	Forgetting factor in Generalized Boosting

Special measures

$\nu^{(L)}$	Column measure w.r.t. loss function L
$\hat{\nu}_n^{(L)}$	Empirical column measure w.r.t. loss function L based on n observations (n may be suppressed)
$\hat{\nu}_{vec}$	Column measure written as $p-$dimensional vector
$\hat{\nu}_t$	Column measure at time t
$(\hat{\nu}^{(L)})^{stab}$	Stable empirical column measure w.r.t. loss function L
$\hat{\nu}^{(L)}(\zeta)$	Empirical column measure w.r.t. loss function L induced by row measure ζ
$\hat{\nu}_{\tilde{L}}^L$	Empirical column measure computed by Column Measure Boosting
$(\hat{\nu}_{\tilde{L}}^L)^{ultrastable}$	Ultra-stable empirical column measure
$(\hat{\nu}_{\tilde{L}}^L)^{(CMB,b)}$	Empirical column measure computed by CMB in the $b-$th stability iteration

$(\hat{\nu}_{\tilde{L}}^{L})^{CMB}$	Aggregated column measure computed by aggregating the empirical column measures computed by CMB in each stability iteration
$(\hat{\nu}_{\tilde{L}}^{L})^{(b)}$	Empirical column measure computed by Column Measure Boosting on the $b-$th subsample
$\hat{\nu}^{cons}$	Empirical consensus column measure
$\hat{\nu}^{block}$	Block column measure
$\check{\nu}_{\tilde{L}}^{L}$	Empirical column measure computed by TSCMB on the first stage
ν^{init}	Initial/prior column measure
ζ	Row measure
$\hat{\zeta}_n$	Empirical row measure
ζ^{init}	Initial/prior row measure
ζ_t	Row measure at time t
$\hat{\zeta}^{block}$	Block row measure
$\hat{\zeta}_{\tilde{L}}^{L}$	Empirical row measure computed by CMB
$(\hat{\zeta}_i, \hat{\nu}_j)$	Empirical RCM pair
$\aleph$	Cell measure
$\hat{\aleph}$	Empirical cell measure
$\hat{\aleph}^{stab}$	Stable cell measure
$\hat{\aleph}(\hat{\zeta}, \hat{\nu})$	Empirical cell measure induced by the empirical RCM pair $(\hat{\zeta}, \hat{\nu})$
$\aleph_t$	Cell measure at time t

Sets and data

$\subset\subset$	Compact subset
$x_{i:n}$	$i-$th smallest component of vector x
$(\Omega, \mathcal{A}, P)$	General notation for a measure space
A	Action domain
$\mathbb{B}^p$	Borel sigma algebra in p dimensions
$Best_K$	The index set of the top K instances
$\widehat{Best_K}$	Estimated sets of top K instances
$\mathcal{C}^p(\Theta)$	Space of all continuous bounded $\mathbb{R}^p-$valued functions on Θ
$\mathcal{D}$	General notation for a data set
$\mathcal{D}^{train}$	Training data
$\mathcal{D}^{(b,train)}$	General notation for the training set of the $b-$th subsample
$\mathcal{D}^{test}$	Test data
$\mathcal{D}^{(b,test)}$	General notation for the test set of the $b-$th subsample

$\mathcal{D}^{CC}$	Subset of complete cases of the data
$\mathcal{D}^{sing}$	Data set used for SingBoost
$\mathcal{D}^{(sing,b,train)}$	Training set for the $b-$th SingBoost model in Column Measure Boosting
$\mathcal{D}^{(sing,b,test)}$	Test set for the $b-$th SingBoost model in Column Measure Boosting
$\mathcal{D}^{CMB}$	Data set used for Column Measure Boosting
$\mathcal{D}^{(CMB,b)}$	Data set used for Column Measure Boosting in the $b-$th stability iteration
$\mathcal{F}_{\beta}$	Parametric function class containing linear models
Θ	Parameter space (subset of $\mathbb{R}^p$)
$I^{(b)}$	Set of row indices used for the $b-$th SingBoost model in CMB
$J_{\tilde{L}}^{L}$	Indices that form the singular part of $\nu^{\tilde{L}}$ w.r.t. $\nu^{(L)}$
$L(X,Y)$	Space of continuous linear maps from X into Y
Λ	Subset of $\mathbb{R}_{\geq 0}$ in which the regularization parameter can take values
O	Observation domain
$\mathcal{P}(A)$	Power set of A
$\text{Perm}(x)$	Set of all permutations of the finite-dimensional vector x
$\mathcal{P}$	Parametric distribution family
$\mathcal{P}(A)$	Power set of a set A
$\hat{\Pi}_J(\lambda)$	Probability to select every variable with column index contained in J using a regularized learning method with penalty parameter λ
π_{grid}	Grid of thresholds for our Stability Selection
$\Psi_2(\theta)$	Set of suitably regular influence functions at P_θ
$\Psi_2^D(\theta)$	Set of suitably regular partial influence functions at P_θ according to Jacobian D
q_{grid}	Grid of numbers of final coefficients for our Stability Selection
S^0	True set of predictors
$\hat{S}$	Estimation of S^0
S_0^{rel}	True set of relevant variables
$\hat{S}^{stab}$	Estimated stable predictor set
$\hat{S}(\lambda)$	Estimated set of predictors when regularization with penalty parameter λ is concerned
$\hat{S}^{init}(\lambda)$	Initial estimation for $\hat{S}(\lambda)$
$\hat{S}^{cell,\lambda}$	Empirical cell set computed by the Graphical Lasso with regularization parameter λ
$\hat{S}^{cell,0}$	True cell set
$\hat{S}^{cell,stab}$	Estimated stable cell set
$\hat{S}^{row,stab}$	Estimated stable row set
$\hat{S}^{cons}$	Predictor set corresponding to a consensus ranking

$\hat{S}^{cons,stab}$	Stable predictor set corresponding to a consensus ranking
$\hat{S}_{\tilde{L}}^{L}$	Set of predictors selected by Column Measure Boosting
$(\hat{S}_{\tilde{L}}^{L})^{stab}(\pi_{thr})$	Stable model reported by our Stability Selection based on a grid π_{grid}
$(\hat{S}_{\tilde{L}}^{L})^{stab}(q)$	Stable model reported by our Stability Selection based on a grid q_{grid}
$U_*(\theta_0, r)$	Contamination ball with radius r around P_{θ_0}
$U^{cell}(\theta_0, r)$	Cell-wise contamination ball with radius r around P_{θ_0}
$\mathcal{U}_*(\theta_0)$	Set of contamination balls with different radii around P_{θ_0}
$W^{2,2}$	Sobolev space
$\mathcal{X}$	Subset of $\mathbb{R}^p$ in which the predictor variables take values
X^{train}	Regressor variables of the training set
X^{test}	Regressor variables of the test set
X_i	$i-$th row of X
$X_{\cdot,j}$	$j-$th column of X
$X^{(l)}$	Columns of the regressor matrix corresponding to the $l-$th block/group
x_I	Subvector of x only containing the indices $i \in I$
$\mathcal{Y}$	Subset of $\mathbb{R}$ (at identified places $\mathbb{R}^k$) in which the response variables take values
Y^{train}	Response vector of the training set
Y^{test}	Response vector of the test set

Functions

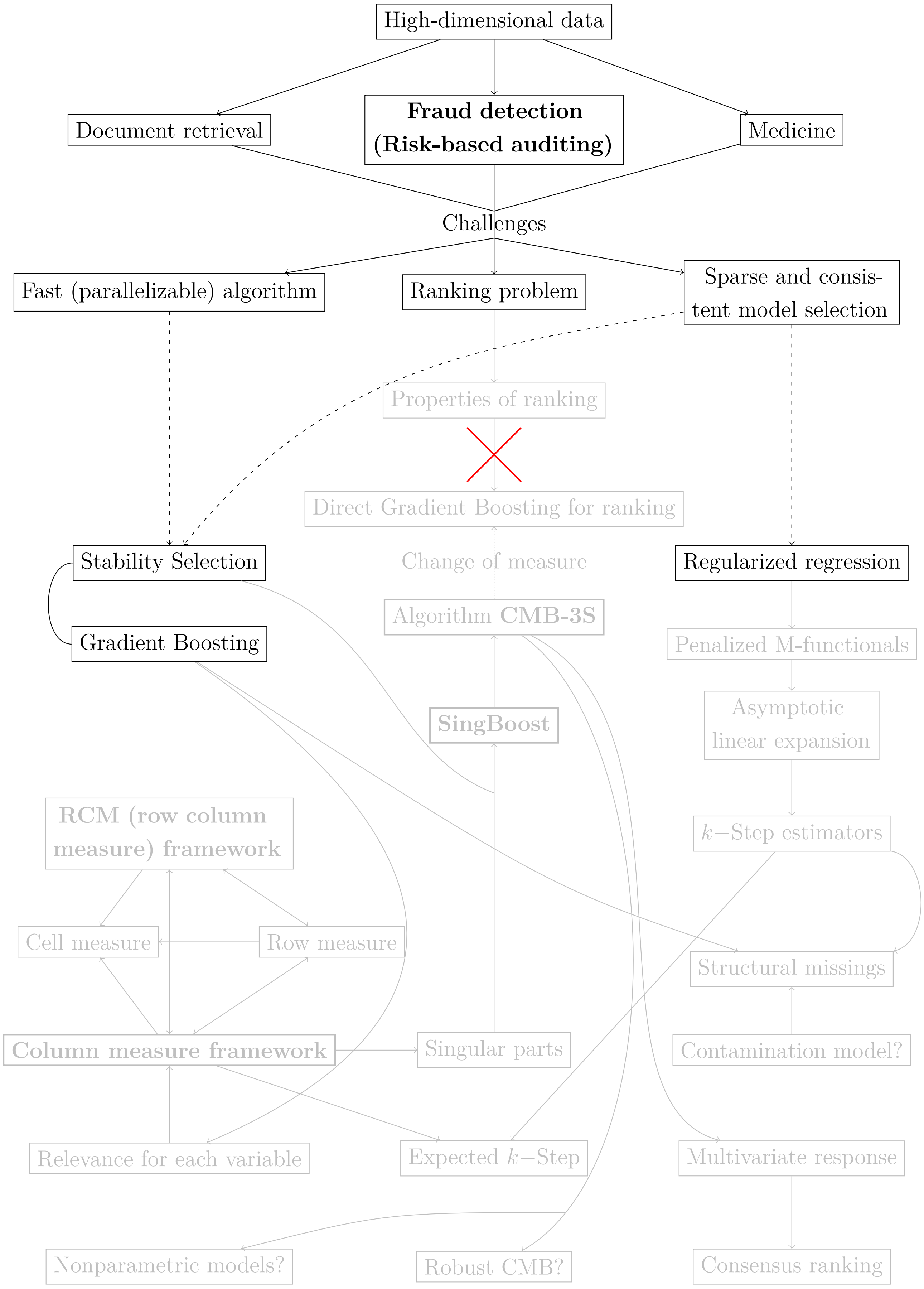
$\circ$	Composition of maps
$\circ_H$	Hadamard product
$\circ^{vec}$	Component-wise composition of maps
$C(\cdot)$	Cost operator
η	General notation for the target function of a Z-estimator
η^{λ}	General notation for the target function of a regularized Z-estimator
$\eta^{r,\lambda}$	General notation for the target function of a regularized ranking Z-estimator
$\hat{f}^{(m)}$	Strong model computed in the $m-$th Boosting iteration
$\hat{f}^{(b)}$	Final model computed by Boosting on the $b-$th subsample
$\hat{g}^{(m)}$	Weak model computed in the $m-$th Boosting iteration
$I(\cdot)$	Indicator function
$\mathrm{IC}(x, T, P)$	Influence function of functional T at x for distribution P
L	General notation for a loss function

$L(y, \hat{y})$	Loss w.r.t. loss function L for y and the corresponding prediction of y
L_2	Squared loss function
L_τ	Check loss function w.r.t. τ
L_n^{hard}	Hard ranking loss
$L_n^{loc,K}$	Localized ranking loss for the K best instances
$L_n^{loc,K,norm}$	Standardized localized ranking loss for the K best instances
$L_n^{weak,K}$	Weak ranking loss for the K best instances
$L_n^{weak,K,norm}$	Standardized weak ranking loss for the K best instances
M_π	Special matrix-vector-vector product operator respecting a block partition π
$\mathcal{M}_1(\mathcal{A})$	Set of all probability measures on the sigma algebra $\mathcal{A}$
r	Ranking rule
R	General notation for the risk, i.e., the expected loss
R_n	General notation for the empirical risk
ρ_π	Special vector-vector product operator respecting a block partition π
s	Scoring function
T_π	Special vector-vector product operator respecting a block partition π
Z_n	General notation for the target function of an empirical Z-estimator
Z_n^λ	General notation for the target function of an empirical regularized Z-estimator
$Z_n^{r,\lambda}$	General notation for the target function of an empirical regularized ranking Z-estimator

Abbreviations

AUC(s)	Area under the ROC$_s$ curve
BIC	Bouligand influence curve
CMB	Column Measure Boosting
CMB-2S	Column Measure Boosting with Stability Selection
CMB-3S	Column Measure Boosting with SingBoost and Stability Selection
ERM	Empirical risk minimization
FPR	False positive rate
FSBDP	Finite sample breakdown point
GLM	Generalized linear model
IAUC	Volume under the integrated ROC curve
IC	Influence curve, more precisely Gâteaux influence curve

IROC_s	Integrated ROC curve of a scoring function s
LocAUC	Area under the localized ROC
MAR	Missing at random
MCAR	Missing completely at random
MNAR	Missing not at random
pIC	Partial influence curve
RSS	Residual sum of squares
ROC_s	Receiver operation characteristic of a scoring function s
SNR	Signal-to-noise ratio
SRM	Structural risk minimization
TPR	True positive rate



High-dimensional data
Document retrieval
Fraud detection (Risk-based auditing)
Medicine
Challenges
Fast (parallelizable) algorithm
Ranking problem
Sparse and consistent model selection
Properties of ranking
Direct Gradient Boosting for ranking
Change of measure
Stability Selection
Gradient Boosting
Regularized regression
Algorithm CMB-3S
Penalized M-functionals
SingBoost
Asymptotic linear expansion
RCM (row column measure) framework
k−Step estimators
Cell measure
Row measure
Structural missings
Column measure framework
Singular parts
Contamination model?
Relevance for each variable
Expected k−Step
Multivariate response
Nonparametric models?
Robust CMB?
Consensus ranking

Chapter 1

Introduction

1.1 Background and motivation

When talking about supervised learning problems, one usually thinks of regression or classification tasks. While both of them are often the appropriate ones in nearly all application domains, there is another natural goal when performing data analysis, namely to order the instances. Such tasks arise for example in search engines when documents have to be ranked according to their relevance for a respective query (Cao et al. [2006], Herbrich et al. [1999], Page et al. [1999]). Other applications include credit-risk screening where potential borrowers are ranked by their creditworthiness (Clémençon et al. [2013b]). Ranking also plays a crucial role in medicine where patients are ranked by their chance of suffering a specific illness or cancer (Clémençon et al. [2011], Clémençon et al. [2013b]).

In general, the responses in data sets corresponding to those problems are binary, therefore a natural criterion for such binary ranking problems is the probability that an instance belongs to the class of interest. While ranking can be generally seen in between classification and regression, those binary ranking problems are very closely related to binary classification tasks (see also Balcan et al. [2008]). For binary ranking problems, there exists vast literature, including theoretical work as well as learning algorithms that use SVMs (Brefeld and Scheffer [2005], Herbrich et al. [1999], Joachims [2002]), Boosting (Freund et al. [2003], Rudin [2009]), neural networks (Burges et al. [2005]) or trees (Clémençon and Vayatis [2008], Clémençon and Vayatis [2010]).

As for the document ranking, the labels may also be discrete, but with $d > 2$ classes, for example in the OHSUMED data set (Hersh et al. [1994]). For such general $d-$partite ranking problems, there also has been developed theoretical work (Clémençon et al. [2013c]) as well as tree-based learning algorithms (Clémençon and Robbiano [2015a], Clémençon and Robbiano [2015b], see also Robbiano [2013]).

Recently, Clémençon investigated a new branch of ranking problems, namely the continuous ranking problems where the name already indicates that the response variable is continuous. He already thought of applications in natural sciences or quantitative finance (Clémençon and Achab [2017]). This continuous ranking problem can be located on the other flank of the spectrum of ranking problems that is closest to regression.

The continuous ranking problem is especially interesting when trying to rank instances whose response is difficult to quantify. A common technique is to introduce latent variables which are used for example to measure or quantify intelligence (Borsboom et al. [2003]), personality (Anand et al. [2011]) or the familiar background (Dickerson and Popli [2016]). While in these cases, the latent variables are treated as features, a continuous ranking problem would arise once a response variable which is hard to measure is implicitly fitted by replacing it with some latent score which is much more general than ranking binary responses by means of their probability of belonging to class 1. An example is given in Lan et al. [2012] where images have to be ranked according to their compatibility to a given query.

The motivation of this work is to develop machine learning algorithms for risk-based auditing to detect tax evasion, using the restricted personal resources of tax offices as reasonably as possible. Risk-based auditing can be seen as a general strategy for internal auditing, fraud detection and resource allocation that incorporates different types of risks to be more tailored to the real-world situation, see Pickett [2006] for a broad overview, Moraru and Dumitru [2011] for a short survey of different risks in auditing and Khanna [2008] and Bowlin [2011] for a study on bank-internal risk-based auditing resp. for a study on risk-based auditing for resource planning.

Until here, we did not yet point out why we actually need new statistical learning techniques to tackle our problem. Indeed, if we formulate the problem as a binary ranking problem where the response variable is either tax compliance or a wrong report of the tax liabilities, we could just use the existing methods. However, different problems lead to a different amount of information granted by its solution. As classification is not as informative as ranking since the classes do not have to be ordered while ranking also incorporates an ordering, ranking in turn is less informative than regression since regression tries to predict the actual response values themselves where ranking just tries to find the right ordering. An analogous argument is true for binary ranking problems and continuous ranking problems. If we state a binary ranking problem, we would just get information which taxpayer is most likely to misreport his or her income without providing any information on its amount. On the other hand, if we set up a continuous ranking problem where the amount of damage is the variable of interest, we can directly get information about the compliance of the taxpayer by looking at the sign of the response value. In particular, if information on the compliance is available, then one can assume that the information on the amount of additional payment or back-payment has also been collected, so imposing a binary ranking problem would lead to a large loss of information.

We will now briefly recapitulate existing work on risk-based strategies for tax audit planning. Alm et al. [1993] compares the effect of different auditing rules, ranging from pure random auditing where each taxpayer's income tax statement gets reviewed with a fixed probability which is equal for all taxpayers over a deterministic cutoff rule which leads to auditing exactly if less income than this cutoff is reported to deterministic strategies that incorporate the behaviour of each taxpayer in the past. The cited reference, though not examining how to detect fraud, concludes with a strong recommendation of risk-based auditing rules since there mere existence leads the taxpayers to be more honest.

Gupta and Nagadevara [2007] exactly addressed to the problem of fraud detection in the tax auditing context using data mining techniques, despite they formulated the problem as a binary classification problem. They applied discriminant rules, decision trees and logistic regression as well as hybrids of trees and discriminant rules to compare those methods that have already been applied to the problem in different references cited therein. As in Gupta and Nagadevara [2007], the goal was to achieve a high strike rate, i.e., the relative part of the fraudulent income tax statements in the whole set of audited instances should be high, in contrast to Hsu et al. [2015], who validated their models (ranging from Naïve Bayes over SVM and Boosting to neural networks) by a profitability criterion, using the revenues from subsequent payments of taxes divided by the costs of the auditing process. The latter also recommended the application of such learning techniques to improve the efficiency of tax auditing.

Random	0.02
Classification+random	16.05
Regression	35.41

Table 1.1: Simple example of the average profit achieved by randomly drawing instances to review, by using binary classification (here: LogitBoost) and by using linear regression based on 10000 replications with $n_{train} = 100 = n_{test}$, $p = 5$

Although the cited references identified the problem as a classification problem, the lack of informativity causes it to be insufficient for audit planning since classification only results in a bunch of instances that is predicted to be contained in the fraud class. But if this number is N and there are only enough capacities for $K < N$ auditing processes, the final decision may again gets random. Therefore, the ranking problem is exactly the right task for such applications since the K most suspicious instances can be identified.

An example on how this lack of informativity affects the profit made after the auditing process is given in table 1.1. We have run a very simple simulation on a data set with $n = 100$ and $p = 5$ for both the training and the test data and continuous-valued responses that represent the damage, i.e., a positive value indicates a fraud. The first strategy is just to randomly sample $K = 10$ instances from the test data without replacement a cumulating the damage. More sophisticated is to apply a classification algorithm, in our case LogitBoost, and to concentrate on the instances of the test data that have been predicted as fraud class. Since the number of these instances exceeds K, we randomly draw K instances. The last scenario is an application of a regression model, in our case standard linear regression and the selection of the K instances with the highest predicted damage.

Ranking all instances in this fashion is referred to as ”hard ranking problem” in this thesis. A binary classification task which should end up in predicting exactly K instances to the fraud class is called ”weak ranking problem”. A compromise, a ”localized ranking problem”, has been introduced in Clémençon and Vayatis [2007] with the task to predict the correct ordering of the K ”best” instances (in our case: the most suspicious ones in terms of the predicted amount of damage), neglecting the ordering of the remaining ones. The advantage over the hard ranking problem is that it is less difficult since one only focuses on the relevant part of the list. One may ask what happens if a set of K instances is predicted as the set of best instances but after auditing, the tax office still has capacities left. Then the predictions of the localized ranking problem for the next instances may should not be trusted, so either K needs to be chosen suitably high at the beginning or the hard ranking problem has to be applied in such insecure cases.

Furthermore, a good learning algorithm does not only require to have good predictive performance but also provide interpretability of the resulting model and computational feasibility as well as theoretical statistical foundations. The interpretability is closely connected to the number of predictors that enter the model, but a sparse model is not necessarily well-interpretable, for example RandomForests are hard to interpret. Sparse variable selection is especially indispensable in real-world applications when working with high-dimensional data which are a frequently faced issue particularly when having data coming from medicine or genomics (see e.g. Wang et al. [2011], Hofner et al. [2015]), but also if one invokes higher order interaction terms or if there are many multicategorical predictors, the total number of variables can be large and the total number of possible models rapidly explodes as emphasized in table 1.2. Ideally, the set of predictors in the model should only contain a few variables and needs to be stable. Additionally, the data sets that an analyst faces are often not "clean", i.e., they contain heterogeneities like outliers or missings.

p	# models	25;4	#	PI	#
4	17	7	129	10	1025
8	257	14	16385	36	$\approx 6.87 \cdot 10^{10}$
20	$\approx 10^6$	35	$\approx 3.44 \cdot 10^{10}$	210	$\approx 1.65 \cdot 10^{63}$
40	$\approx 1.1 \cdot 10^{12}$	70	$\approx 1.2 \cdot 10^{21}$	820	$\approx 6.99 \cdot 10^{246}$

Table 1.2: Number of possible models (containing an intercept each) if one allows to select categorical variables partially. The column 25;4 indicates the case that 25% of the predictors are discrete-valued with 4 values each, PI refers to the case that we are in the setting of column 1, but that all pair-wise interactions additionally enter the model

The properties of interpretability, stability and computational efficiency are overwhelmingly satisfied by Boosting-type algorithms (see Hothorn et al. [2017]) combined with a so-called Stability Selection (Meinshausen and Bühlmann [2010], Hofner et al. [2015]). However, those Boosting algorithms are so far tailored to regression and classification since they require certain regularity conditions on the loss function which are not fulfilled by ranking loss functions. Notably, the loss function does not have to be smooth to make Gradient Boosting available as for example the squared loss since piece-wise differentiable losses like the L_1-loss are also allowed, but ranking loss functions are not even continuous, so Gradient Boosting cannot be accessible at the first glance. The goal of this work is to make Boosting available for potentially highly non-regular loss functions.

Summing up, our task is to provide a profit-based Boosting-type ranking algorithm which makes reliable predictions of the amount of damage due to fraudulent declaration of tax liabilities and which achieves stable and sparse model selection and ends up with a well-interpretable model.

1.2 Outline of the thesis

The thesis starts with a pilot chapter which shortly summarizes common machine learning paradigms such as penalization or cross validation that will be important throughout this work. We describe the component-wise least squares Boosting algorithm (Bühlmann [2006]) which is fundamental for the whole thesis. Apart from referring to popular competitors like the Lasso and variants, we recapitulate the famous Stability Selection, originally introduced in Meinshausen and Bühlmann [2010], which is an ensemble model selection method relying on the Lasso, which has been transferred to model selection by ensembles of Boosting algorithms in Hofner et al. [2015].

The rest of the thesis is divided into seven parts.

The **first part** is a theoretical part which addresses to the question if general penalized statistical M-estimators can be asymptotically linearly expanded, resulting in the respective M-estimators being representable as arithmetic mean of influence curves, up to a starting estimator and a remainder term.

This part starts with an introductory chapter where several concepts of robust statistics are defined, ranging from influence curves and asymptotic linear estimators to the breakdown point and examples for (non-)robustness of a few selected popular sparse learning algorithms. This chapter provides much more content than we need in the first part, but we also gather further elements of robust statistics like contamination models and $k-$Step estimators which we will need later in the thesis.

The second and last chapter of the first part first recapitulates results on compact differentiability of M-functionals which lead to the asymptotic linear expansion of the corresponding M-estimators which means that the estimators are representable as an arithmetic means of influence curves in the sense that $\hat{\theta}_n - \theta = \text{mean}_i(\psi_{\hat{\theta}_n}) + r_n$ for some remainder term r_n and an influence curve $\hat{\psi}_{\theta_n}$ corresponding to the initial estimator $\hat{\theta}_n$. Based on the results in the classical setting (Rieder [1994]), i.e., in absence of a penalty term., we provide our theoretical results concerning the requirements for compact differentiability of penalized M-functionals and asymptotic linearity of regularized M-estimators, carefully investigating each of the new conditions that we need to impose. We will see that these requirements depend on the regularity of the penalty term so that the Lasso fails to be asymptotically linear at the first glance since the l_1-penalty term is not differentiable, so even the translation from the M-estimator to a Z-estimator would need further investigation. However, we are able to use an approximation lemma from Avella-Medina [2017] that elegantly ensures asymptotic linearity under slightly stricter conditions but such that the concrete shape of the non-differentiable regularization term is no longer important but only the fact if there exists a smooth approximation term. We concretize our results for the Lasso, the elastic net and the Adaptive Lasso using the influence curves that already have been provided in literature (Öllerer et al. [2015], Avella-Medina [2017]) where for the latter, we can identify the corresponding influence curve with a partial influence curve. These results ensure that the regularized M-estimators can also be identified with procedures that collect information separately w.r.t. the rows and the columns as Boosting-type algorithms, so they can be embedded into the framework that we introduce in this work.

The **second part** is devoted to ranking problems and contains, apart from a detailed review on ranking problems and of existing ranking algorithms, proposals for influence curves for ranking problems to embed them into the framework of part I, results on elicitability of ranking functionals as well as arguments why a simple Gradient Boosting for continuous ranking problems using surrogate losses is not reasonable.

The part starts with a detailed chapter on ranking problems where we carefully distinguish between binary (bipartite), $d-$partite and continuous ranking problems by the nature of the response variables and between weak, hard and localized ranking problems by the learning goal. We will see that the three latter types of ranking problems lead to special loss functions (Clémençon et al. [2008], Clémençon and Vayatis [2007]), so that a ranking problem can be solved by empirical risk minimization as theoretically proven in Clémençon et al. [2008]. For better comparison of these losses, we normalize them so that they always take values in $[0, 1]$. We close the chapter with a broad overview of existing algorithms for bipartite, $d-$partite and continuous ranking problems with and without model selection and with a short discussion of similarities and differences of $d-$partite ranking problems and ordinal regression.

The second chapter gathers some properties of ranking losses and ranking problems that we derived. First, we solved the main computational issue of how to evaluate the hard and the localized ranking loss since it requires pair-wise comparisons, so a naïve implementation would be of complexity $\mathcal{O}(n^2)$ resp. $\mathcal{O}(K^2)$, if n is the total number of observations and K the number of best instances. Afterwards, we make proposals for influence functions for the weak, the localized and the hard ranking problem based on the ordered logit model.

The third chapter provides a lemma where we show that hard ranking functionals satisfy the property of elicitability (Gneiting [2011]). Furthermore, in cases where multiple true rankings occur, in other words, where the empirical hard ranking loss does not have a unique minimizer, for example when having ties, we prove that the respective ranking functional is even strongly or exhaustively elicitable (Fissler et al. [2019]). These results allow an objective comparison of competing ranking models to decide which one was the best.

The last chapter provides very enlightening results concerning the applicability of Boosting to the continuous ranking problem. A very common technique in classification settings is to use convex and sufficiently regular surrogate losses for the non-differentiable and non-convex $0/1-$loss. Moreover, surrogate losses have already been applied to the binary ranking problem, resulting in Boosting-type algorithms like RankBoost (Freund et al. [2003]) or p-Norm-Push (Rudin [2009]). Surprisingly, we will see that a similar technique is not applicable to continuous ranking problems due to the range of the response variable, aside from being computationally nearly intractable due to the pair-wise structure of the surrogate losses and their gradients.

The **third part** is the main conceptual part of the thesis. We will introduce a measure on the columns of a data matrix which we call "column measure", its cousin, the "row measure", and the eponymous "gradient-free Gradient Boosting" algorithm "SingBoost" that can capture singular parts between column measures.

The first chapter starts with a recapitulation of the binary option pricing model where one models the option price as an expectation w.r.t. a risk-neutral measure. We continue to state a theoretical Riesz-type result, but do not proceed working with it. The next section briefly recapitulates rejection sampling which leads to a very simple idea to perform L_2-Boosting on subsampled data and to choose the model which has the best test performance according to some ranking loss movitated by the fact that the conditional expectation $\mathbb{E}[Y|X]$ is an optimal ranking rule (Clémençon and Achab [2017]). Before showing why this strategy may not be meaningful when performing (sparse) variable selection, we identify weights assigned to the rows of a data matrix as a "row measure" and mention different examples where essentially empirical row measures are fitted, though they have not been called so yet. The main conceptual contribution of the chapter is the column measure which is similar to the row measure, but which is defined on the columns of a data matrix, assigning importance to the respective predictor variables. Clearly, the column measure depends on the loss function, hence we introduce the notation $\hat{\nu}_n^{(L)}$ for the empirical column measure w.r.t. a loss function L on a data set with n observations. The column measure leads to our so-called "column measure framework" in which we identify all learning procedures based on (a selection of) predictor variables. Furthermore, we state the first type of a connection between row and column measures by emphasizing that models based on subsamples or Bootstrap samples essentially lead to empirical column measures which are induced by a prior row measure. Our main theoretical contribution of this chapter is the Expected One-Step which can be seen as generalization of the usual One-Step estimator in the sense that it cannot only handle variable selection but any relevance allocation for the predictors according to some empirical column measure. We close the chapter by mentioning Online Learning algorithms that include time-dependent row and column measures.

The second chapter finally mathematically describes why a rejection sampling strategy cannot be meaningful when computing sparse models according to one loss function L but regarding another loss function $\tilde{L}$ as target loss function. The problem can be described as facing singular parts of column measures, in our special case the singular parts $J_{\tilde{L}}^L$ of the column measure $\nu^{(\tilde{L})}$ w.r.t. the column measure $\nu^{(L)}$. We see that the situation is even worse than it looks at the first glance because we cannot control à priori for singular parts since both (theoretical) column measures are unknown and if $\tilde{L}$ is a complicated loss function which in our case is some ranking loss function, we additionally have no chance to approximate $\nu^{(\tilde{L})}$ at all. We introduce SingBoost, a "gradient-free Gradient Boosting" strategy that alternatingly finds the optimal simple least squares baselearner according to either the standard squared loss L_2 and the target loss $\tilde{L}$. We provide some comments on the implementation of this algorithm, on the updating scheme and how we get the coefficient paths. As the main theoretical contribution of this part, we adapt the two theorems [Bühlmann, 2006, Thm. 1] and [Bühlmann and Van De Geer, 2011, Thm. 12.2] to prove that SingBoost has the properties of estimation and prediction consistency even in very high dimensions provided that a Corr-min condition for the selection of the predictor variable in each singular iteration holds.

The **fourth part** is the heart of the thesis and combines Stability Selection and SingBoost with a new framework, the so-called "row column measure (RCM) framework", resulting in an algorithm which we call CMB-3S, providing sparse and stable $\tilde{L}-$adapted model selection and $\tilde{L}-$adapted coefficients.

The first chapter is again devoted to empirical column measures. After shortly identifying the Random Lasso (Wang et al. [2011]) and the BlockForest (Hornung and Wright [2018]) as examples for algorithms that partially perform sampling from some column measure, we discuss that a single SingBoost algorithm does, besides neither providing stable nor sparse models, not compute appropriate empirical column measures w.r.t. $\tilde{L}$. Since we do not have invoked a fixed loss function like standard Stability Selection, the possibly selected variables from singular parts are inherently disadvantaged in terms of selection frequencies, so we do not adapt the column measure w.r.t. $\tilde{L}$ sufficiently. Therefore, we need to run SingBoost on subsamples of the data which gives us the opportunity to aggregate them in an $\tilde{L}-$adaptive way. More precisely, we will assign weights to each SingBoost model and therefore to each empirical column measure, depending on the out-of-sample-performance of the computed coefficients w.r.t. $\tilde{L}$. Furthermore, we will only concentrate on the best few models which results in performing a change of measure. We call this procedure which finally provides an aggregated column measure "Column Measure Boosting (CMB)". The rest of the chapter concerns about possible accelerations of CMB by combining sampling from an empirical column measure and optimization in a stage-wise manner.

The second chapter starts with showing how an $\tilde{L}-$based Stability Selection can be realized. We mimic the standard Stability Selection, but instead of just fixing a threshold for the selection frequencies, we perform an optimization w.r.t. $\tilde{L}$, for one of those parameters, on a grid of reasonably chosen values by computing the out-of-sample-performance on a validation set. Therefore, we provide three different variants how the coefficients may be computed. One of those variants will directly include an empirical row measure which is merely a by-product of Column Measure Boosting, assigning relevance w.r.t. $\tilde{L}$ to the rows of the data and maybe being able to identify suspicious observations which surely should be downweighted. We call the Column Measure Boosting algorithm with Stability Selection "CMB-3S". For reasonably evaluating the performance of the resulting CMB-3S model, we also provide an algorithm which we call CV.CMB-3S which is able to compare competitor CMB-3S models (and Boosting models as a special case) in terms of their cross-validated test performance according to $\tilde{L}$ by applying the underlying algorithm to different partitions of the data. We also mention that CV.CMB-3S leads to an ultra-stable empirical column measure by aggregating the stable column measures computed w.r.t. each partition. Working simultaneously with row and column measures is the meaning of our RCM framework in which we can at least embed all learning algorithms with univariate response. A popular example which perfectly fits into it is the SLTS algorithm of Alfons et al. [2013] which alternatingly works with empirical row and column measures. We go even further and propose a systematization of learning algorithms which enables us to express the paradigms of sparsity, stability and robustness in terms of row and column measures. At the end, we briefly list some ideas how one maybe could deal with grouped predictors and row weights.

The **fifth part** extends our former results and concepts to the case of multivariate respones and provides deeper insights into the connection of robustness, stability and sparsity.

The first chapter of this part is devoted to ranking and regression problems with multivariate responses. For ranking problems, we can identify the rankings w.r.t. each response column as a partial ranking which leads to the question if there is an overall ranking, i.e., a so-called "consensus ranking", which finally provides an ordering of the regressor rows. We provide an idea how to get an overall stable empirical column measure which can be thought of defining a stable predictor set corresponding to the consensus ranking. As for regression problems with multivariate responses, we recapitulate the multivariate L_2-Boosting algorithm (Lutz and Bühlmann [2006b]) and provide a first simple idea how a suitable SingBoost algorithm ("MultiSingBoost") may look like. We are already able to adapt [Lutz and Bühlmann, 2006b, Thm. 1] to this case, showing that this MultiSingBoost would be estimation consistent in very high dimensions under an additional suitable Corr-min condition. We close the chapter by extending our results on elicitability and strong elicitability of hard ranking functionals to (strong) $k-$elicitability of multivariate hard ranking functionals in the case of uncorrelated response columns.

The second and last chapter of this part starts with introducing the cell measure which is related to (sparse) covariance and precision matrix estimation. We will see that the cell measure is essentially induced by RCM pairs and that Meinshausen and Bühlmann [2010] already worked with stable cell measures and corresponding stable cell sets. The next section gives a brief overview of robust covariance estimation, especially the MCD estimator (Rousseeuw [1985]), and of outlier detection procedures that are able to detect cell-wise outliers. We provide two examples where we identify the MCD estimator and the DDC procedure for cell- and case-wise outlier detection and imputation (Rousseeuw and Van Den Bossche [2018]) as instances of our RCM framework. In the third section, we restate a classical outlier detection algorithm from Rocke and Woodruff [1996], the Fast-MCD algorithm and the SLTS algorithm (Alfons et al. [2013]) in L_2-Boosting form with multivariate responses resp. univariate responses (for SLTS). This motivates an idea for Generalized L_2-Boosting where concentration steps and forgetting factors are included and which may select more than one predictor in each iteration. Finally, we propose a Stability Selection for the rows, based on either robust algorithms or some Generalized Boosting and provide conceptual ideas how to find a suitably stable row set.

The **sixth** part is the practical part which demonstrates the vast flexibility of our implemented algorithms.

The first chapter provides an overview of the main functionalities of our algorithms by applying it to small data sets like the `iris` data set and the `bodyfat` data set from the `R`$-$package `TH.data`. We show that our `CV.CMB3S` function can also be applied to the high-dimensional `Real.2` data set from the package `PRIMsrc` and to the ultrahigh-dimensional `riboflavin` data set from the package `hdi`.

The second chapter compares the performance of our CMB-3S algorithm (i.e., respecting singular parts) with the performance of L_2-Boosting and L_2-Boosting combined with our Stability Selection in terms of the hard ranking loss and see that our algorithm reliably beats L_2-Boosting in the majority of cases. We also study the performance of our loss-based Stability Selection, without considering singular parts (”CMB-2S”), for the squared, the Huber and the hard ranking loss by comparing its cross-validated test loss with the respective test loss resulting from L_2-Boosting resp. Huber-Boosting resp. L_2-Boosting. We observe that our models outperform the simple Boosting models. Additionally, we learn that Hofner’s Stability Selection (Hofner et al. [2015]) gets unreliable on high-dimensional noisy data sets but that our Stability Selection can indeed cope with such situations.

The **last part** is devoted to different ideas for future research.

The first chapter deals with the problem of missings in the data. We will not provide any new ideas for dealing with common missings, but we try to find solutions for data sets with structural missings that are especially interesting in audit and medicine applications. The first idea that we present is based on asymptotically linear estimators while the second one is a Boosting approach where we recommend to treat the missings column-wisely, separately for each column-specific baselearner. We see that the second approach has the potential to be applicable for data sets with arbitrary missing structures, but some theoretical work is still required to reasonably choose the required weights for an aggregation of the losses.

The second chapter shows the basic steps that were required to transfer Column Measure Boosting to nonparametric models where smoothing splines would be used as baselearners. We also shortly discuss if non-linearity is an issue if one is interested in ranking. Additionally, we provide an idea how to robustify the SingBoost algorithm by using optimally-robust influence curves. Again, more theoretical work is necessary.

The last chapter of this thesis summarizes again the theoretical, conceptual and algorithmic contributions of this thesis.

At the very end, an appendix containing several definitions, tools from (functional) analysis and asymptotic statistics follows.

Chapter 2

Learning sparse and stable models

This chapter recapitulates important paradigms of computing and selecting models as well as sophisticated variable selection procedures that will be fundamental throughout this thesis.

The first two sections are not meant to be a survey on concepts of data analysis but just to resume the very basic ideas behind the common concepts of machine learning in the notation of this work.

The last sections are devoted to sparse and stable models that play a crucial role for the whole thesis. At the end, we present two important algorithms that can efficiently and flexibly provide well-interpretable models: Stability Selection and Boosting. Since our contributions are built on those techniques and require some recommendations on tuning parameters and sampling paradigms, the respective subsections are rather deeply detailed.

Note that all the algorithms that we recapitulate here require loss functions that satisfy some regularity conditions that do not hold for the ranking losses that we are actually interested in.

2.1 A brief review on model fitting and validation

Let $\mathcal{D} := (X, Y) \in \bigotimes_{i=1}^m \mathcal{X} \times \bigotimes_{i=1}^m \mathcal{Y} \subset \mathbb{R}^{m \times (p+1)}$ for some subsets $\mathcal{X} \subset \mathbb{R}^p$, $\mathcal{Y} \subset \mathbb{R}$ be a data set consisting of $p-$dimensional predictors and one-dimensional responses. The predictors can be either continuous or discrete. Throughout the thesis, we assume models of the form

$$Y = f(X) + \epsilon \tag{2.1.1}$$

for an error term $\epsilon \in \mathbb{R}^m$ with $\epsilon_i \overset{i.id.}{\sim} F_\epsilon$ with $\mathbb{E}[\epsilon_i] = 0$ and $\text{Var}(\epsilon_i) = \sigma^2 \in]0, \infty[$ for all i. The function f may be any measurable function mapping from $\mathcal{X}$ into $\mathcal{Y}$. An important special case on which we will focus nearly throughout the whole thesis is that f is an element of the parametric function class

$$\mathcal{F}_\beta := \{f_\beta(X) = X\beta \mid \beta \in \Theta \subset \mathbb{R}^p\}.$$

Remark 2.1.1 (Intercept). *Note that if not explicitly specified, the first column of the regressor matrix X is allowed to consist only of ones, which means that the first component of the parameter is the intercept. At important places, especially when concerning about Boosting models, we generally highlight the inclusion of intercepts by defining the parameter as $(p+1)-$dimensional vector.*

We always denote the $i-$th row of the predictor matrix X by X_i. Then, the pair (X_i, Y_i) is referred to as the $i-$th instance. The $j-$th column of X is denoted by $X_{\cdot,j}$. There are two different types of design for the regressor matrix X. Either one assumes a stochastic design, so the rows X_i of X are modelled as i.id. random variables, or the regressor matrix is assumed to be fixed which leads to the disadvantage that the rows cannot be regarded as being identically distributed anymore (cf. [Pupashenko et al., 2015, Sec. 2.2]). In this work, the stochastic design will more likely be assumed, except for theoretical results on ranking problems where a random design would not be meaningful.

In the case that the response is continuous, we face a regression or a ranking problem whereas discrete-valued responses lead to classification or ranking problems. Those types of problems are summarized as supervised learning problems. Unsupervised learning problems do not include response variables and the main task is to find structures in the data like dividing them into clusters (see any statistical learning reference, e.g. Friedman et al. [2001]). For details on ranking problems, see chapter 5.

A widely spread technique to fit models is the empirical risk minimization (ERM) principle (see any statistical learning reference, e.g. Devroye et al. [2013]) that manifests itself in defining a suitable loss function and a (parametric) class of models. Having data, one tries to pick the model that minimizes the empirical counterpart of the risk, typically an arithmetic mean of the given loss function comparing each component of the predicted vector $\hat{Y}$ with the corresponding component of the true response vector Y in the case of regression or classification problems. Loss functions that are tailored to ranking problems are more complicated due to their global nature and compare permutations of the true and the predicted response vector.

Assumption 2.1.1 (**Loss function**). *In a regression setting, a loss function is given by a function $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$. Throughout the thesis, we assume $L(y, y) = 0$ for all y.*

Of course, the more instances are available, the more evidence we have, but using the whole data set for training a model does not provide any opportunity to test how the model works on other data following the same distribution. A standard approach (see e.g. Friedman et al. [2001]) is to divide the data set into a training set $\mathcal{D}^{train} = (X^{train}, Y^{train}) \in \mathbb{R}^{n \times (p+1)}$ and a test set $\mathcal{D}^{test} = (X^{test}, Y^{test}) \in \mathbb{R}^{(m-n) \times (p+1)}$. Then the validation of the fitted model on $\mathcal{D}^{train}$ is done by predicting a vector $\hat{Y} \in \mathbb{R}^{m-n}$ based on the predictors X^{test} and evaluating the empirical risk on $\mathcal{D}^{test}$. In general, the error made on the training set is referred to as the **training error** and the error on the test set as the **test error**.

This approach is in general insufficient since dividing the data only once just provides one test set which generally does not lead to enough statistical evidence. An extended approach that is very popular if one has to compare competing models or to adjust hyperparameters is the **cross validation** technique (see. e.g. Arlot et al. [2010]). The main idea is to simply divide the data set into V almost equally sized parts and to train V models, each on the whole data set except one part, respectively. Then, the single part that was not included in the training set serves as the test set. Aggregating the test errors provides the **cross-validated test error**, so the model whose cross-validated test error is the smallest will be finally selected. If the data contain heterogeneities like possible outliers, it is recommended to use a **randomized cross validation** where one generates V times a subsample of the data and uses the remaining part, respectively, as test set.

A simple example for cross validation arises when hyperparameters have to be tuned. The most straightforward way to choose the hyperparameters is to define a grid on (a subset of) the cartesian product of the sets in which the hyperparameters can take values and to train models with each of the configurations defined by the grid in the spirit of cross validation. The model with the combination of hyperparameters that resulted in the lowest cross-validated test error is then taken.

2.2 Regularization and model complexity

In our parametric setting, the complexity of a model is given by the number of predictors that are included. The more complex the model is, the more functions can be represented, but usually it is more likely to model the stochastic errors in this case. Additionally, more complex models are less interpretable.

Motivated by these issues, regularized regression methods penalize certain quantities depending on the parameter β and a regularization parameter $\lambda > 0$. Combining regularization with ERM is referred to as structural risk minimization (SRM) going back to Vapnik (Vapnik [2013], Vapnik [1998]). There exists vast literature studying the theoretical aspects of regularization including the Vapnik-Chervonenkis dimension, the fat shattering dimensions, Gaussian and Rademacher complexities and risk bounds (see e.g. Vapnik [1998], Duan [2011], Bartlett and Mendelson [2002], Koltchinskii and Panchenko [2002], Boucheron et al. [2005], Bartlett et al. [2002]).

Two well-known criteria that penalize the complexity of a model are the Akaike information criterion and the Bayes information criterion. While the Akaike information criterion can be seen as an unbiased approximant of the Kullback-Leibler divergence of the true and the predicted model, the Bayes information criterion approximates the Bayes factor. Yang [2005] already showed that the benefits of each criterion, more precisely, the consistency of the Bayes information criterion and the minimax-rate optimality of the Akaike information criterion, cannot be combined, so there is no overall best choice.

For practical applications, both information criteria provide a ranking of the quality of the fitted models, but of course, these models have to be fitted first. A naïve approach is the best subset selection where all 2^p possible models (not counting the model containing only an intercept) are fitted and compared. More sophisticated extensions are step-up and step-down procedures.

Another approach to incorporate the model complexity is to directly intervene into the fitting procedure, say, the optimization process by adding a regularization term to the loss function. A well-known regularized regression technique is the **Ridge regression** (cf. p.e. Friedman et al. [2001]) which penalizes the l_2-norm of β, i.e.,

$$\hat{\beta}^{ridge} = \underset{\beta \in \Theta}{\operatorname{argmin}} \left(\frac{1}{n} ||Y - X\beta||_2^2 + \lambda ||\beta||_2^2 \right),$$

resulting in shrunken but usually still non-zero coefficients. A popular method that produces sparse models is the **Lasso** (cf. Bühlmann and Van De Geer [2011]) which penalizes the l_1-norm of β, i.e., it solves

$$\hat{\beta}^{lasso} = \underset{\beta \in \Theta}{\operatorname{argmin}} \left(\frac{1}{n} ||Y - X\beta||_2^2 + \lambda ||\beta||_1 \right).$$

It theoretically would be even more effective to penalize the l_0-norm of β itself, but the l_1-norm is the best convex surrogate of it which is crucial for numerical optimization of the regularized loss. From a geometric point of view which is graphically given in Tibshirani [1994], it is evident that much more coefficients will be shrunken towards zero than for Ridge regression. Nevertheless, l_q-penalization for $0 < q < 1$ is a popular technique for example in compressed sensing (see p.e. Elad [2010]).

Nonconvex regularization like the already mentioned l_q-penalty has been tackled by Zou and Li [2008] using a local linear approximation, resulting in certain weights than enter the penalty term. Even more sophisticated is the **Multistep Adaptive Lasso** introduced in Bühlmann et al. [2008] which is an extension of the **Adaptive Lasso** proposed by Zou (Zou [2006]). The Adaptive Lasso first performs an initial estimation step which provides an initial coefficient $\hat{\beta}^{init}$. Then, in the second step, one solves

$$\hat{\beta} = \underset{\beta \in \Theta}{\operatorname{argmin}} \left(\frac{1}{n} ||Y - X\beta||_2^2 + \lambda \sum_j \frac{|\beta_j|}{|\hat{\beta}_j^{init}|} \right).$$

Zou [2006] propose to use a $\sqrt{n}-$consistent algorithm in the first step. Bühlmann and Van De Geer [2011] recommend to apply the Lasso where an optimal λ is chosen by cross validation. This increases the sparsity since an initial coefficient of zero directly leads to a zero coefficient in the second step. The Multistep Adaptive Lasso just invokes more steps where the regularization parameter is updated between each step and enters also as a weight. It is shown in Bühlmann et al. [2008] that the Multistep Adaptive Lasso essentially performs $ln-$regularized least squares minimization asymptotically.

For other work on nonconvex regularization, see for example Breheny and Huang [2011] for regression and logistic regression, Laporte et al. [2014] for document retrieval with SVMs or Loh and Wainwright [2015] and Loh et al. [2017] for magnificent results on local optima resp. on model selection consistency when both loss and penalty can be nonconvex. Furthermore, in Mazumder et al. [2011], an algorithm called `SparseNet` is proposed to minimize the squared loss with some nonconvex penalty, in Taddy [2017], the penalty is weighted in each iteration according to the current coefficients, Wang et al. [2013] provide further theoretical results for least squares regression with nonconvex penalty in very high dimensions and Wei and Zhu [2012] extend the usage of popular nonconvex penalties like SCAD (Fan and Li [2001]) and MCP (Zhang et al. [2010]) to grouped variables.

In this work, we will focus on regularized regression techniques rather than on information criteria for the following reasons:
i) The information criteria are independent of the concrete setting where regularized regression procedures have an additional penalty parameter that can be suitably chosen;
ii) Subset selection gets infeasible if $p > n$ except for the step-up procedure that starts with an empty model and successively adds one predictor whereas regularized methods are basi-

cally designed for the high-dimensional case.

Besides many modifications of the Lasso like the already mentioned adaptive Lasso to correct the overestimation behaviour of the standard Lasso (Zou [2006]) or the elastic net (Zou and Hastie [2005]) where the penalty term is a convex combination of an l_1- and an l_2-penalty, there exists vast literature on regularized methods for other learning tasks. Examples range from robust regression methods like sparse MM-estimators (Smucler and Yohai [2017]) or sparse least trimmed squares (Alfons et al. [2013]) to the estimation of a precision matrix via the Graphical Lasso (Banerjee et al. [2008] and Friedman et al. [2008], see also Bühlmann and Van De Geer [2011] or Van de Geer [2016]), and the Sparse-Group Lasso from Simon et al. [2013] which, in contrast to the Group Lasso of Yuan and Lin [2006], does not only select a few groups of variables but even only a few variables in the selected groups.

Remark 2.2.1 (**Sparsity**)**.** *One should always keep in mind that applying regularization techniques implicitly states the assumption that the true underlying model is sparse, i.e., that the true coefficient vector β_0 has $s^0 \ll p$ non-zero entries, thus $||\beta_0||_0 = s^0$.*

2.3 Stability Selection

For the rest of the thesis, let $\Lambda \subset \mathbb{R}_{\geq 0}$ always be the set of regularization parameters λ. For a variable selection technique in which λ enters as penalty parameter, we denote the estimated set of variables by this algorithm with a specified λ by $\hat{S}(\lambda)$. When tuning the algorithm, usually by defining a grid of candidate values for λ and fitting a model for each element of the grid, one essentially would pick one of the indexed models which behaves best w.r.t. some quality criterion. Following Meinshausen and Bühlmann [2010] or [Bühlmann and Van De Geer, 2011, p. 346], variable selection by just choosing one element of the set $\{\hat{S}(\lambda) \mid \lambda \in \Lambda\}$ does generally not suffice due to the overestimation behaviour of algorithms like the Lasso. Instead, the data is resampled many times randomly with a sample size of around $\lfloor n/2 \rfloor$ and only the variables that have most frequently been chosen are ultimately selected. Bühlmann and Meinshausen recommend to generate around 100 subsamples.

In general, one defines the **set of relevant variables** by

$$\hat{S}_0^{rel}(C) := \{j \mid |\beta_j| \geq C\},$$

i.e., the set of variables whose true coefficients have an absolute value far away enough from zero. Define the probability that the set $J \subset \{1, ..., p\}$ of indices will be included in the set of selected variables by

$$\hat{\Pi}_J(\lambda) := P(J \subset \hat{S}(\lambda)).$$

These probabilities are empirically computed many times by resampling and counting the relative number of subsamples that led to the covariable set J.

Then, by defining a **cutoff** π_{thr}, the Stability Selection results in the set

$$\hat{S}^{stab} := \{j \mid \max_{\lambda \in \Lambda}(\hat{\Pi}_j(\lambda)) \geq \pi_{thr}\}.$$

So, instead of choosing one element of $\{\hat{S}(\lambda) \mid \lambda\}$ as in the traditional setting, one takes a look at the **stability paths** of each predictor. They are given by the selection probabilities of each variable, given the regularization parameter, as an analog to regularization paths which show the evolution of the coefficients in dependence of λ (see for instance Rosset and Zhu [2007]). Graphically, Stability Selection chooses every predictor whose stability path (partially) exceeds the cutoff. Those stability paths are implemented in the R−package `stabs` (Hofner and Hothorn [2017], Hofner et al. [2015], Thomas et al. [2018]).

An important issue is always the type I error, i.e., the amount of falsely selected variables. [Meinshausen and Bühlmann, 2010, Thm. 1] show a bound for the expected value of false positives. They additionally show that exact error control is possible with Stability Selection even in high-dimensional settings whereas otherwise, it is very difficult to even estimate the noise level.

Another idea which is also presented in [Meinshausen and Bühlmann, 2010, App. A] is the following one. Instead of randomly choosing a subset of around $\lfloor n/2 \rfloor$ observations of the data, one may split the sample into two equally sized disjoint parts with index sets I_1, I_2. Then the **simultaneous selection probability** is defined as

$$\hat{\Pi}_J^{simult,\lambda} := P(J \subset \hat{S}^{simult,\lambda}) = P(J \subset \hat{S}_{I_1}(\lambda) \cap \hat{S}_{I_2}(\lambda))$$

where $\hat{S}_{I_k}$, $k = 1, 2$, are the estimated sets of true predictors, respectively. One can show ([Meinshausen and Bühlmann, 2010, Lem. 1]) that

$$\hat{\Pi}_J^{simult,\lambda} \geq 2\Pi_J^{\lambda} - 1.$$

An extension of the Stability Selection is introduced in Shah and Samworth [2013] where the selection strategy is essentially the same as above with simultaneous selection, but Shah and Samworth [2013] provide bounds for the type I error that are free from the exchangeability assumption and the assumption that the selection procedure is better than random guessing needed in Meinshausen and Bühlmann [2010]. This exchangeability assumption was already discussed in the discussion section of Meinshausen and Bühlmann [2010] and in Hofner et al.

[2015] who point out that this implies that every noise variable would have the same selection probability and therefore the same correlation with the response vector.

Helpful recommendations how to set the cutoff π_{thr} are also given in the practical section of Hofner et al. [2015].

An algorithm called Bolasso (Bach [2008]) can be seen as related work, but with the main difference that no subsampling strategy but a Bootstrap strategy is used. Then the final set of selected variables is the intersection of all selected sets. Therefore, one may think of a "cutoff" which would be 1 for the Bolasso.

We close this subsection by recapitulating two important definitions in the context of variable selection which can be found in the book of Bühlmann and Van de Geer (Bühlmann and Van De Geer [2011]) which is an excellent reference for the theory of the Lasso, including other variants, and their oracle properties as well as for Boosting.

Definition 2.3.1. *Assume that S^0 is the true set of variables, S_0^{rel} the set of relevant variables and $\hat{S}$ is the set of parameters selected by the model selection procedure.*
a) *The model selection procedure has the* ***screening property*** *if*

$$P(\hat{S} \supset S_0^{rel}) \longrightarrow 1$$

for $n \to \infty$.
b) *The model selection procedure is* ***variable selection consistent*** *if*

$$P(\hat{S} = S^0) \longrightarrow 1$$

for $n \to \infty$.

2.4 Boosting and variable selection

The idea behind Boosting is to combine simple models (**"weak learners"**, **"baselearners"**) that are easy to compute in order to generate a final "strong" learning model. The first algorithm of this kind was the `Adaboost` algorithm (Freund and Schapire [1997]) for binary classification problems. It has been shown in Breiman [1999] that `AdaBoost` can be seen as a gradient descent algorithm for the exponential loss. Friedman et al. [2000] showed that `AdaBoost` can also be identified with a Forward Stage-wise procedure.

In general, one can think of a loss function $L(y, f)$ where y represents a response component and f is the selected model. This loss function should be smooth and convex in the second argument (see Bühlmann and Yu [2003]). Then **Gradient Boosting** is an iterative procedure that computes the negative gradients $-\partial_f L(Y_i, f)$ for every i in the $(m+1)-$th step and evaluates it at the current model $\hat{f}^{(m)}$ that has been fitted in the previous iteration and that supplies a predicted value for Y_i. Then a new weak learner $\hat{g}^{(m+1)}$ is fitted using this negative gradient as response and the current model $\hat{f}^{(m)}$ is updated via $\hat{f}^{(m+1)} = \hat{f}^{(m)} + \hat{\kappa}^{(m+1)}\hat{g}^{(m+1)}$ where the step sizes or learning rates $\hat{\kappa}^{(m)}$ are possibly chosen adaptively, often a fixed learning rate κ is used. The general **functional gradient descent (FGD)** algorithm can be found in the appendix (A.8).

The weak learners can for example be either trees, smoothing splines or simple least squares models. Depending on the problem that has to be solved, one uses different loss functions like the negative binomial log-likelihood loss

$$\log_2(1 + \exp(-2yf))$$

for binary classification (**LogitBoost**), the exponential loss

$$\exp(-yf)$$

as a convex surrogate of the $0/1-$loss (**AdaBoost**) or the absolute resp. the quadratic loss for L_1- resp. Least Squares Boosting. For an overview of Gradient Boosting algorithms and their paradigms, we refer to Bühlmann and Yu [2003] and Bühlmann and Hothorn [2007].

In this work, we will rather concentrate on **Least Squares Boosting**, hereafter always L_2-Boosting, which has been shown to be estimation and prediction consistent even for very high dimensions ([Bühlmann, 2006, Thm. 1], [Bühlmann and Van De Geer, 2011, Thm. 12.2]). It can be extremely efficiently computed using a component-wise linear regression procedure as described in algorithm 1 (cf. Bühlmann [2006]).

Since it is easier to understand for this Boosting algorithm, we directly expressed the steps through coefficients. Bühlmann himself uses different notation, calling variants like the one above "coefficient version" and those like algorithm 23 "function version" (see Lutz and Bühlmann [2006a]).

The update step has to be understood in the sense that every $j-$th component of $\hat{\beta}^{(k)}$ remains unchanged for $j \notin \{1, \hat{j}_k\}$ and that only the intercept and the $(\hat{j}_k)-$th component are modified (recall that if an intercept is fitted, we have $X_{\cdot,1} = 1_n$). If one computes intercept-free models, only the $\hat{j}_k-$th component changes. However, in algorithm 1 and other Boosting pseudo-codes, we always assume to include intercepts. Note that the formula to compute the residuals indicates that one is interested in following the steepest gradient. This is true in component-wise L_2-Boosting since for the standardized loss function

$$L(y,f) := \frac{1}{2}(y-f)^2,$$

the negative gradient w.r.t. f is given by $(y-f)$.

```
**Initialization:** Data $(X,Y)$, step size $\kappa \in ]0,1]$, number $m_{iter}$ of iterations and
    parameter vector $\hat{\beta}^{(0)} = 0_{p+1}$;
Compute the offset $\bar{Y}$ and the residuals $r^{(0)} := Y - \bar{Y}$;
for $k = 1,...,m_{iter}$ do
    for $j = 1,...,p$ do
        Fit the current residual $r^{(k-1)}$ by a simple least squares regression model using
          the predictor variable $j$;
        Compute the residual sum of squares;
    end
    Take the variable $\hat{j}_k$ whose simple model $\hat{\beta}_{\hat{j}_k} \in \mathbb{R}^{p+1}$ provides the smallest residual
      sum of squares;
    Update the model via $\hat{\beta}^{(k)} = \hat{\beta}^{(k-1)} + \kappa\hat{\beta}_{\hat{j}_k}$;
    Compute the current residuals $r^{(k)} = Y - X\hat{\beta}^{(k)}$
end
```

Algorithm 1: Component-wise least squares Boosting

There are still two tuning parameters that we did not concern about yet. The step size or learning rate κ decides how much weight we assign to a new baselearner compared to the weight assigned to the current model (which is 1). As suggested by Friedman [2001] and done for L_2-Boosting in Bühlmann and Yu [2003], one can choose its value adaptively by performing a line search, i.e., with the current model $\hat{\beta}^{(k-1)}$ and the new base model $\hat{\beta}_{\hat{j}_k}$, one chooses κ such that the combined model provides the minimal loss over a grid of step sizes. However, Bühlmann [2006] recommend a small step size which causes the necessity of more iterations m_{iter}, but giving the resulting models in general a better out-of-sample performance. Bühlmann and Hothorn [2007] found out that a line search is not necessary because for reasonably small step sizes, its influence is fairly small. Therefore, we do not experiment with the learning rate throughout this thesis and generally use the default value of $\kappa = 0.1$ in our practical applications.

The other tuning parameter is m_{iter}, the number of Boosting iterations. Contrary to the step size, the number of Boosting iterations becomes very important if a pure Boosting algorithm is applied without further variable selection criteria due to a general overfitting behaviour of Boosting (see e.g. Bühlmann and Hothorn [2007]). More precisely, a small number of

m_{iter} leads to biased models with a low variance whereas performing many iterations reduces the bias but increases the variance (Mayr et al. [2012b]). While Bühlmann and Yu [2003] used the relative difference of mean squared errors as a stopping criterion, Bühlmann [2006] invoked the corrected AIC, minimizing it over a suitable set of values for m_{iter}. In fact, stopping criteria that are not based on loss differences (which is a convergence criterion) are often referred to as "early stopping" criteria (see Bühlmann and Hothorn [2007] and Mayr et al. [2012b]). Mayr et al. [2012b] complain that early stopping criteria often require a very high initial number of iterations. Then, in a cross validation scheme, one proceeds computing the Boosting model, comparing some criterion like an AIC for every model and choosing the model that achieves the best value, so an optimal number for m_{iter} is found just afterwards. Thus, due to potentially many iterations and evaluations that were made in vain, this procedure is not computationally effective. They themselves propose to set m_{iter} initially to 40. Then they search for the AIC-optimal number of iterations afterwards by subsampling and cross validation. If this number is sufficiently far away from the current m_{iter} (they propose to use 0.9 times this current number as threshold), the algorithm has finished, otherwise they set $m_{iter} = m_{iter} + 40$ and repeat the procedure.

Boosting with component-wise least-squares as baselearners, among a variety of other Boosting algorithms, is implemented in the R−package `mboost` (Hothorn et al. [2017], Hofner et al. [2014], Hothorn et al. [2010], Bühlmann and Hothorn [2007], Hofner et al. [2015], Hothorn and Bühlmann [2006]).

Remark 2.4.1 (**Lasso vs.** L_2−**Boosting**). *Some work has been done to detect and to understand similarities and differences of Lasso and L_2−Boosting, for example Efron et al. [2004], Bühlmann and Hothorn [2007]. A considerable difference of variable selection via Boosting compared to Lasso variable selection is pointed out in [Bühlmann, 2006, Sec. 4.3]. The Lasso cannot pick more than $min(n, p+1)$ variables while the number of the variables selected by Boosting is not restricted in advance. Empirically, Bühlmann and Hothorn [2007] argue that it is not clear if Lasso or L_2−Boosting are to be preferred. Additionally, Meinshausen et al. [2007] empirically compared Lasso and L_2−Boosting to the Dantzig Selector proposed by Candès et al. [2008]. As Efron et al. [2007] concluded that the Dantzig Selector is inferior to the Lasso, Meinshausen et al. [2007] recommended to use rather L_2−Boosting or Lasso than the Dantzig Selector.*

Recently, Freund et al. [2017] deepened the understanding of the relation of L_2−Boosting and the Lasso by providing a new unifying framework. They proposed the convex optimization problem

$$\min_{r \in P_{res}} (f(r)) := ||X^T r||_\infty$$

where $P_{res} := \{r \mid r = Y - X\beta \; \exists \beta \in \mathbb{R}^p\}$, i.e., the space of attainable residuals (for $\Theta = \mathbb{R}^p$). In other words, the maximal absolute correlation of the residuals with the predictors has to be minimized, but the optimization variable is r and not β! Since the target function is convex but not differentiable, they provided a subgradient descent method to solve the problem. Furthermore, they showed that Forward Stage-wise Regression and L_2-Boosting are special cases of this subgradient descent. Apart from these results, they even proceeded further, showing that the Lasso is equivalent to the solution of a subgradient descent for a regularized version of the correlation minimization problem where the target function is additionally penalized by $||r - Y||_2^2$, scaled with a regularizing factor such that the original problem is just the unregularized special case. In summary, Freund et al. [2017], besides many new theoretical and practical insights, unified both Lasso and L_2-Boosting as instances of the same underlying optimization procedure.

A great benefit of Boosting is the opportunity to perform variable selection again afterwards. One simply invokes the relative selection frequency of all predictors and only includes those in the final model whose selection frequency exceeds some predefined threshold. In fact, Hofner et al. [2015] provided a Stability Selection for Boosting models. As in the Stability Selection setting, one generates a number of subsamples and performs Boosting on each subsample. Each Boosting model is iterated until a predefined number q of variables is selected, respectively for each subsample. Then one counts the absolute selection frequencies of each variable. Aggregating over the subsamples, the overall absolute selection frequencies are determined and only those variables whose relative selection frequency exceeds some threshold is ultimately chosen. The number q and the threshold π_{thr} are related by the per-family error-rate, so holding two of these quantities fixed, the third can be reasonably set (Hofner et al. [2015]). In general, Meinshausen and Bühlmann [2010] recommend to set $\pi_{thr} \in]0.6, 0.9[$, but mention the low importance of this parameter since the resulting models are only insignificantly affected. Hofner et al. [2015] think of $]0.5, 1[$ as a reasonable interval for π_{thr}. Again, this strategy is implemented in the R$-$package `stabs` (Hofner and Hothorn [2017], Hofner et al. [2014], Thomas et al. [2018]).

Remark 2.4.2. *In a pure theoretical scenario, Stability Selection could even lead to less sparse models than all the single Boosting models. Consider $B = 100$ and let $q = 8$. If it happened that for example the union of all sets of selected variables contained 10 variables and that each of these variables had been chosen by exactly 80 models, then a Stability Selection with a cutoff not larger than 0.8 would select all 10 variables.*

However, this scenario is very unlikely to occur in practice.

Remark 2.4.3. *L_2-Boosting just means that the squared loss is minimized by Boosting. Aside from component-wise linear models, there are other possible baselearners such as stumps or smoothing splines (Bühlmann and Yu [2003]). Note that when referring to L_2-Boosting in the first parts, we always think of component-wise linear baselearners. Smoothing splines will be mentioned shortly in the last part.*

Remark 2.4.4. *Although L_2-Boosting is adapted to regression problems, Bühlmann and Yu [2003] provided a modified version which they called L_2-WC-Boosting ("with constraints") which can also handle classification problems. An example which is given in [Bühlmann and Yu, 2003, Ch. 5.1] assumes that the responses take values in $\{-1,1\}$. Then, L_2-Boosting is applied, but the predicted responses are always enforced to be contained in the interval $[-1,1]$ so that the classification is done by discretization. [Bühlmann and Yu, 2003, Ch. 5.2] describes how even multi-class problems can be solved.*

A more sophisticated result is given in Bühlmann [2006], showing that applying L_2-Boosting to binary classification problems by modelling $Y_i = f_n(X_i) + \epsilon_i$ with $f_n(x) = \mathbb{E}[Y|X=x]$ (the subscript n emphasizes that the number n can grow) and heteroskedastic but centered errors ϵ_i is consistent.

Remark 2.4.5. *For the case of L_2-Boosting, Lutz and Bühlmann [2006a] derived an algorithm that does not necessarily proceed along steepest gradients but that searches for a direction which is conjugate to all directions in the iterations before. The authors show experimentally that their `CDBoost` algorithm can outperform Lasso or standard L_2-Boosting in certain settings.*

Remark 2.4.6. *Boosting has been extended to the case of fitting generalized additive models with more than one parameter (see Mayr et al. [2012a]). The idea is to compute baselearners for the predictor corresponding to each parameter separately by fitting negative partial residuals w.r.t. these predictors.*

Remark 2.4.7 (Online Learning). *We note that Online Boosting algorithms already have been developed, starting with the pioneering work of Oza [2005]. Until now, there has been done much work on Online Boosting, see for example Chen et al. [2012], Beygelzimer et al. [2015b] and Leistner et al. [2009] for binary classification or Chen et al. [2014] for multiclass classification in the bandit setting which all essentially have a similar structure as AdaBoost. We also note that for the regression context with smooth convex loss functions, proposals for Online Gradient Boosting have been made, see Beygelzimer et al. [2015a] or Hu et al. [2017].*

However, we restrict ourselves to the Batch Learning framework in this thesis.

2.5 Boosting with categorical predictors and interactions

Categorical predictors or even interactions can easily be encoded by generating extra columns in the model matrix. If a variable corresponding to column $X_{\cdot,j}$ has d levels, then one just introduces $(d-1)$ columns where the $k-$th of them has ones exactly where the $(k+1)-$th level is observed and replaces the original column by these artificial columns. In the following simple example, the encoding is as follows:

$$\begin{pmatrix} A \\ B \\ C \\ C \\ D \\ A \\ B \\ D \end{pmatrix} \mapsto \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

This works since the intercept already contains all information for class A. A first-order interaction term between two variables $X_{\cdot,j}$ and $X_{\cdot,k}$ is encoded by adding a column whose entries are $(X_{ij}X_{ik})_{i=1}^n$. In cases where one of the variables is categorical with d levels, we additionally need $(d-1)$ columns since each of the artificial columns corresponding to the categorical variable is component-wisely multiplied with the other column. If both are categorical with d_j resp. d_k levels, we summa summarum need $(d_j-1)+(d_k-1)+(d_j-1)(d_k-1)$ extra columns.

As described in [Bühlmann and Van De Geer, 2011, Ch. 4], solving the Lasso exactly defines such a model matrix and runs the respective algorithm on it. However, issues can occur both when applying Lasso or component-wise Boosting. Following Gertheiss et al. [2010], the variable selection may depend on the coding scheme, i.e., which variable is chosen as base category. Therefore, they provide a penalty term that incorporates the absolute differences of the coefficients corresponding to each level of a categorical variable, thus it resembles the fused Lasso penalty (see Tibshirani et al. [2005]). This results in uniting categories implicitly by motivating them to have nearly the same coefficient. A similar idea from Gertheiss and Tutz [2009] is to define a Ridge-type penalization for the coefficients of adjacent categories which is implemented in the Boosting package `mboost`.

Alternatively, an approach called **BlockBoost** (see Gertheiss et al. [2011]) was provided. For metric variables, nothing changes compared to standard Boosting techniques. In other words, each metric variable itself forms a block. But concerning categorical variables, all artificial columns that have been used for encoding are treated as one block. On these blocks, a

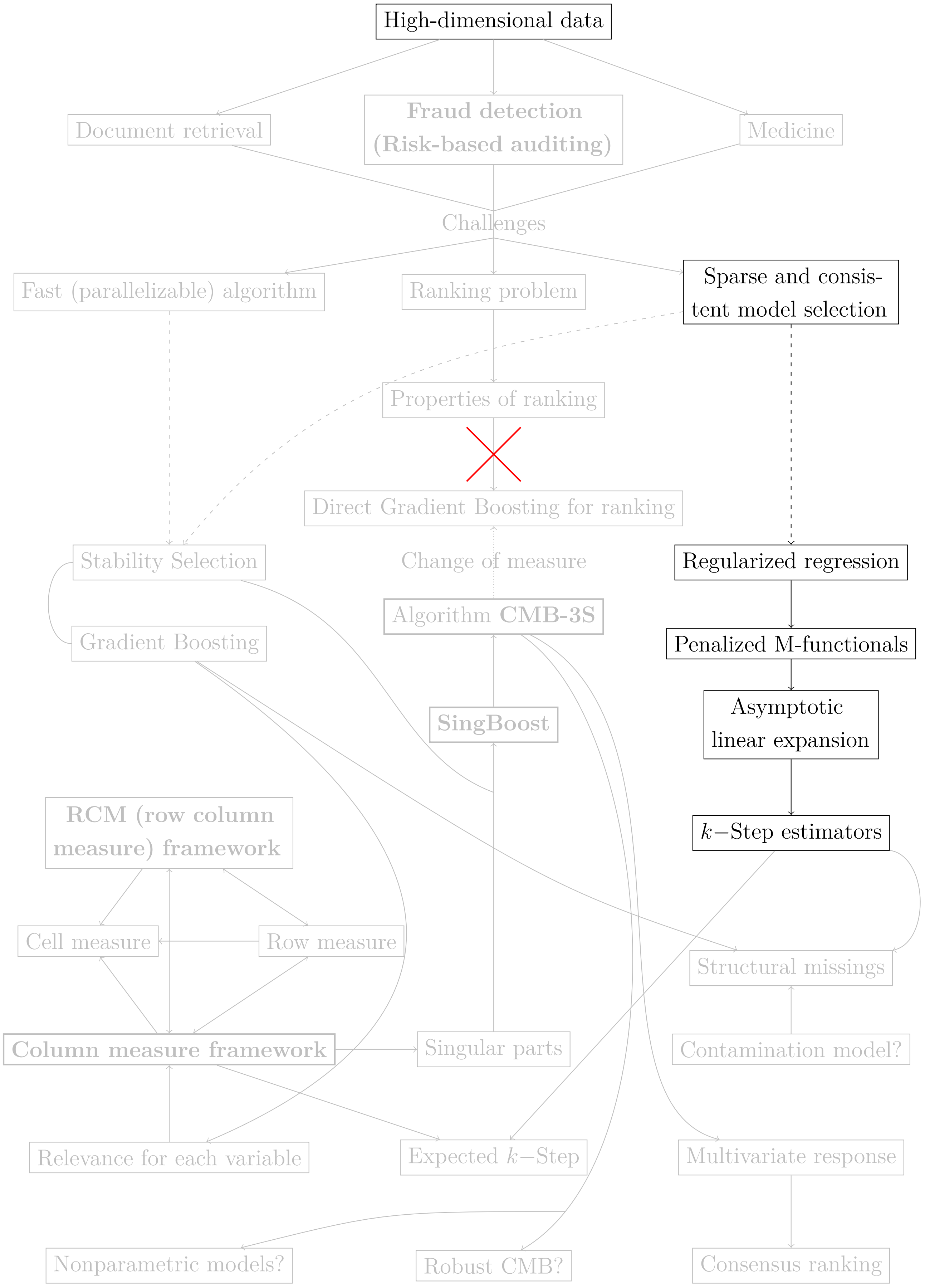
Ridge-regularized weak learner is computed and if one of those models improves the current strong model the most, all coefficients corresponding to this block are updated. This can be formalized by ”structured terms” as done in Tutz [2011] or in the discussion paper Bühlmann et al. [2014].

Apart from linear techniques, Bühlmann and Hothorn [2007] state that interactions can be represented by smoothing splines if one uses pair-wise splines instead of component-wise splines for first-order interactions and so on, whereas trees can handle interactions up to level k if they have at least $(k+1)$ terminal nodes (see Friedman [2001]).

Since this work mainly restricts itself to linear underlying models, varying coefficients (see e.g. Hastie and Tibshirani [1993], Fan and Zhang [1999], Fan and Zhang [2008]) that are capable to represent interactions between variables in nonlinear models should not be a topic here.

Part I

Regularized M-estimators and their asymptotic linear expansion


High-dimensional data
Document retrieval
Fraud detection (Risk-based auditing)
Medicine
Challenges
Fast (parallelizable) algorithm
Ranking problem
Sparse and consistent model selection
Properties of ranking
Direct Gradient Boosting for ranking
Stability Selection
Change of measure
Regularized regression
Algorithm CMB-3S
Gradient Boosting
Penalized M-functionals
SingBoost
Asymptotic linear expansion
RCM (row column measure) framework
k−Step estimators
Cell measure
Row measure
Structural missings
Column measure framework
Singular parts
Contamination model?
Relevance for each variable
Expected k−Step
Multivariate response
Nonparametric models?
Robust CMB?
Consensus ranking

This theoretical part of the thesis is devoted to M-estimators which are a well-known type of estimators in statistics, especially the maximum likelihood estimator is a prominent example. However, a weakness of M-estimators is that they generally do not have an analytical solution in closed form, so one needs iterative procedures to approximate them. But indeed, estimators that correspond to the SRM principle like for example the Lasso from the previous section belong to the class of regularized M-estimators, so it is adequate to study the asymptotic properties of general regularized M-estimators.

In this sense, this part gives an approach to the following problem:

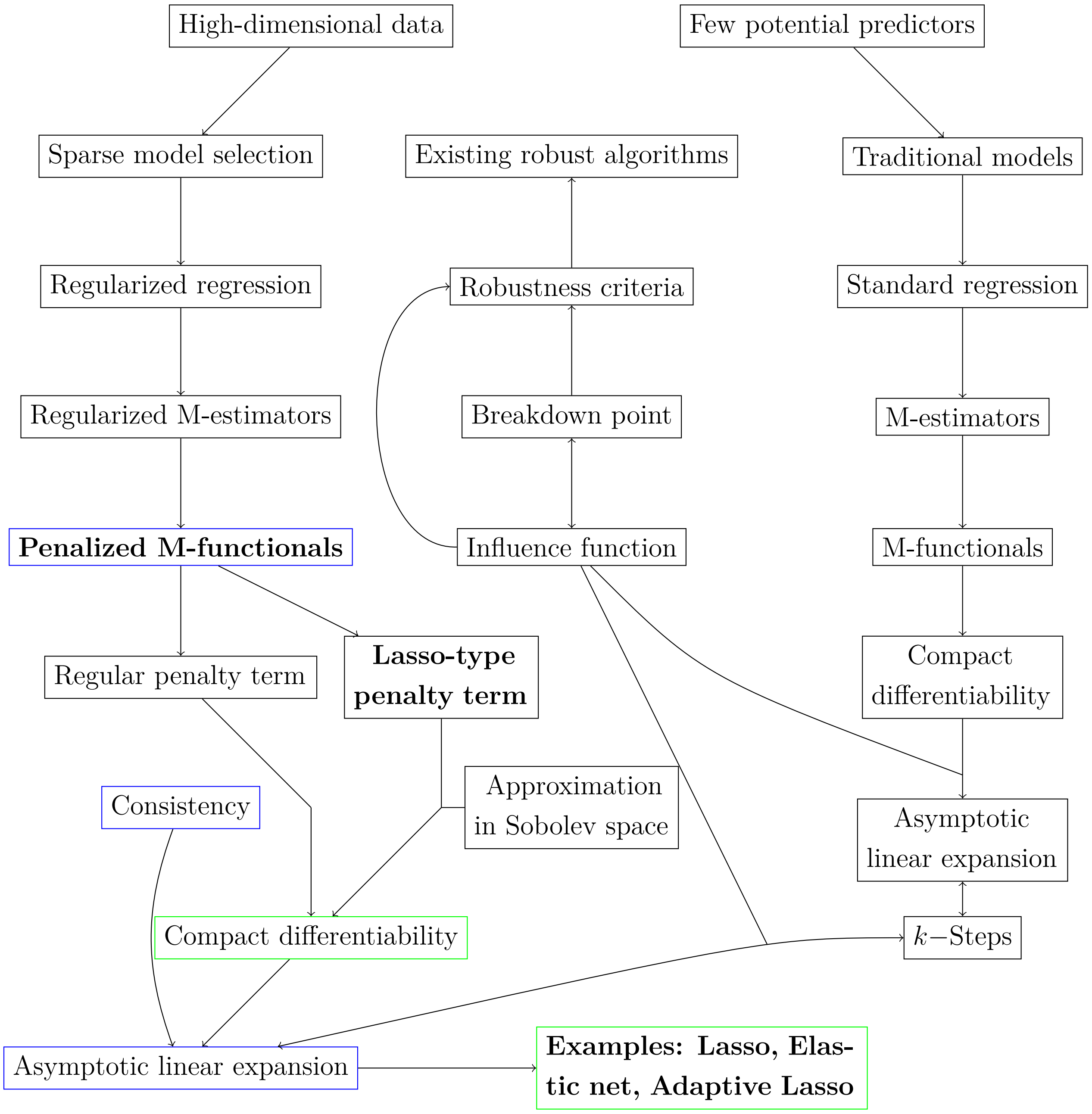
Can regularized M-estimators be asymptotically linearly expanded?

Estimators from sophisticated learning algorithms that are capable to analyze high-dimensional data where the number of predictors exceeds the number of observations like the Lasso (Tibshirani [1994]), the elastic net (Zou and Hastie [2005]) and the adaptive Lasso (Zou [2006]) can be represented by regularized M-functionals. Therefore, studying the asymptotically linear expansion of estimators corresponding to such functionals could give insights into asymptotic properties of modern machine learning techniques.

Related contributions concern about the computation of the Gâteaux derivative of regularized M-functionals (Öllerer et al. [2015], Avella-Medina [2017]) and the asymptotic linearity of a few selected estimators (Van de Geer [2016], LeDell et al. [2015]). Apart from this, impressive results in the nonconvex setting are provided in literature (Loh [2017]). Our contribution should go into the direction of a general theory for the asymptotic linear expansion of regularized M-functionals.

This part is organized as follows. Before presenting our results, an introductory chapter on robust statistics is provided that contains the main definitions of robust statistics and their connections to the asymptotic theory of M-estimation. A short guide how these techniques can enter practical estimation problems closes this chapter.

The second chapter addresses to the initially posed question and shows under which conditions general regularized M-estimators are asymptotically linear. Concrete examples of estimators that satisfy these requirements like the Adaptive Lasso are provided at the end.


High-dimensional data
Few potential predictors
Sparse model selection
Existing robust algorithms
Traditional models
Regularized regression
Robustness criteria
Standard regression
Regularized M-estimators
Breakdown point
M-estimators
Penalized M-functionals
Influence function
M-functionals
Regular penalty term
Lasso-type penalty term
Compact differentiability
Consistency
Approximation in Sobolev space
Asymptotic linear expansion
Compact differentiability
$k-$Steps
Asymptotic linear expansion
Examples: Lasso, Elastic net, Adaptive Lasso

Chapter 3

Robust statistics

This chapter recapitulates important concepts of robust statistics that we will need as preliminaries for our own results presented in the next chapter as well as for later purposes.

A key step when working with empirical data is to define an underlying model to estimate the distribution of the given data. However, the quality of the resulting estimator often suffers from wrong model assumptions that lead to the real data appear as being contaminated. Since this is a frequently faced problem in real-data analysis, it is necessary to make use of the concepts of robust statistics (Huber and Ronchetti [2009], Hampel et al. [2011], Rieder [1994], Maronna et al. [2006]) to handle these phenomena without just removing "outliers" which is definitely not the right track (see Hampel et al. [2011]).

A general functional-analytic approach is to write an estimator as a statistical functional (Von Mises [1947], Reeds [1976]) and to investigate its directional differentiability (Averbukh and Smolyanov [1967]). The influence function or influence curve (Hampel et al. [2011]) of an estimator can be regarded as a special Gâteaux derivative (Christmann and Steinwart [2004]) of the corresponding functional and is a popular tool provided by robust statistics that quantifies the infinitesimal influence of a single observation on the estimator. A necessary condition for robustness is the boundedness of the influence curve. Robustness results by proving this boundedness have been established for example for kernel-based regression problems (Christmann and Steinwart [2004], Christmann and Steinwart [2007]), especially for support vector machines (Hable and Christmann [2011]). Sophisticated approaches to determine optimally-robust influence curves have been developed by Rieder et al. (Rieder [1994], Rieder et al. [2008]).

This chapter gives a broad overview of these concepts. Starting with the abstract definition of $\mathcal{R}-$differentiability of maps between normed vector spaces, we continue to provide the main properties of functional derivatives, carefully distinguishing between Gâteaux-, Hadamard-, Bouligand- and Fréchet-derivatives. The next section contains the most important definitions of quantitative robust statistics like the influence curve, the L_2-differentiability of a parametric statistical model, the breakdown point of an estimator and asymptotically linear estimators, i.e., estimators that can be represented by an arithmetic mean of influence curves, up to an error term of order $o_P(n^{-1/2})$. Afterwards, we present a short survey on existing regularized machine learning algorithms and their robustness properties.

The last section transfers the concept of asymptotically linear estimators to practical situations, by showing how robust estimators with given influence curves can be achieved via a $k-$step construction. We also show how influence curves that arise when estimating a transformation of the parameter of interest, so called "partial influence curves", are related to (sparse) model selection via $k-$Step estimators.

3.1 Functional derivatives

We begin to define $\mathcal{R}-$differentiability of maps between (possibly infinite-dimensional) normed real vector spaces (cf. Rieder [1994], Averbukh and Smolyanov [1967]).

Definition 3.1.1. *Let X, Y be normed real vector spaces. A map $T: X \to Y$ is $\mathcal{R}-$**differentiable** in $x \in X$ if there exists $d_{\mathcal{R}}T(x) \in L(X,Y)$, $s_0 > 0$, such that*

$$T(x+sh) = T(x) + d_{\mathcal{R}}T(x)sh + \rho(sh) \ \forall |s| \leq s_0,$$

where the remainder term ρ satisfies the following conditions:
i) *$\rho(0) = 0$,*
ii) *$\rho \in \mathcal{R}(X,Y)$ where $\mathcal{R}(X,Y)$ is a real vector subspace of Y^X with $\mathcal{R}(X,Y) \cap \mathcal{L}(X,Y) = \{0\}$.*

*Then the continuous, linear map (w.r.t. h per definition) $d_{\mathcal{R}}T(x)$ is referred to as the $\mathcal{R}-$**derivative of** T **at** x.*

The following definition of Rieder [1994] is helpful to distinguish between different types of $\mathcal{R}-$differentiation.

Definition 3.1.2. *Let X, Y, T be as in definition 3.1.1. Let $\mathcal{S}$ be a covering of X. Define the **remainder class***

$$\mathcal{R}_{\mathcal{S}}(X,Y) := \left\{\rho : X \to Y \;\middle|\; \lim_{t\to 0}\left(\sup_{h\in S}\left(\frac{||\rho(th)||}{t}\right)\right) = \rho(0) = 0 \;\forall S \in \mathcal{S}\right\}.$$

By definition 3.1.2, it is clear that $\mathcal{R}-$differentiability can be seen as a linear approximation of some functional T such that the remainder term converges uniformly on all sets $S \in \mathcal{S}$. We now define three special concepts of $\mathcal{R}-$differentiability (cf. Rieder [1994]).

Definition 3.1.3. *Let X, Y, T be as above. Then the functional T is*
i) Gâteaux or weakly differentiable *if $\mathcal{S} = \{S \subset X \mid S \text{ finite}\}$,*
ii) Hadamard or compactly differentiable *if $\mathcal{S} = \{S \subset X \mid S \text{ compact}\}$,*
iii) Fréchet or boundedly differentiable *if $\mathcal{S} = \{S \subset X \mid S \text{ bounded}\}$.*

In the case of Gâteaux differentiability, it is convenient to write $\mathcal{S}_G$ and d_GT, in the case of Hadamard differentiability, we write $\mathcal{S}_H$ and d_HT, and in the case of Fréchet differentiability, the notation is $\mathcal{S}_F$ and d_FT. By compactness resp. boundedness of the elements of $\mathcal{S}_H$ resp. $\mathcal{S}_F$, we can equivalently define Hadamard differentiability of T by the existence of a map $d_HT \in \mathcal{L}(X,Y)$ such that for any $t_n \to 0$, $h_n \to h$, it holds that

$$\frac{T(x + t_nh_n) - T(x)}{t_n} \longrightarrow d_HT(x)h,$$

and Fréchet differentiability of T by the existence of a map $d_FT \in \mathcal{L}(X,Y)$ such that for any $||h|| \to 0$, it holds that

$$\left|\left|\frac{T(x+h) - T(x) - d_FT(x)h}{||h||}\right|\right| \longrightarrow 0$$

which is equivalent to

$$||\rho(h)|| = ||T(x+h) - T(x) - d_FT(x)h|| = o(||h||).$$

We continue to list some properties of these functional derivatives which we borrowed from Rieder [1994], Shapiro [1990], Zajíček [2015].

Remark 3.1.1. *Let X, Y be normed real vector spaces and let $T : X \to Y$.*
i) *If $X = \mathbb{R}$, all $\mathcal{R}-$derivatives coincide with the usual derivatives. If $X = \mathbb{R}^n$, Fréchet and Hadamard differentiability are equivalent.*
ii) *Fréchet differentiability implies Hadamard differentiability which implies Gâteaux differentiability. The derivatives then coincide. The reverse implications do not hold in general.*
iii) *If T is **continuously Gâteaux differentiable**, i.e., the derivative d_GT is continuous in x, then it is Fréchet differentiable.*

For later purposes, roughly speaking, for connecting empirical estimation with statistical functionals, it is necessary to investigate whether the **chain rule** holds for functional derivatives. The following theorem can be found in [Rieder, 1994, Prop. 1.2.6+Thm. 1.2.9] and [Averbukh and Smolyanov, 1967, Thm. 1.6].

Theorem 3.1.1 (Chain rule). *Let X, Y, Z be normed real vector spaces and let $T : X \to Y$, $U : Y \to Z$. If T and U are compactly differentiable, then the chain rule holds, i.e.,*

$$d_H(U \circ T)(x) = d_H U(T(x)) \circ d_H T(x).$$

Conversely, if the chain rule holds, then the maps T and U are already compactly differentiable.

The chain rule does not hold for Gâteaux differentiable maps in general. Counterexamples can be found in Averbukh and Smolyanov [1967] or Fréchet [1937]. Thus, regarding these concepts of functional derivatives one can state that compact differentiability is the weakest form of $\mathcal{R}-$differentiability such that the chain rule holds. Moreover, the composition of a Fréchet differentiable map and a Gâteaux differentiable map does not need to be even Gâteaux differentiable. An example can be found in Averbukh and Smolyanov [1967]. But the following assertion holds (cf. Averbukh and Smolyanov [1967]).

Lemma 3.1.1. *Let T, U, X, Y, Z be as in the theorem. If T is compactly differentiable at x and if U is continuously compactly differentiable at $T(x)$, then the composition $T \circ U$ is continuously compactly differentiable at x.*

3.1.1 Excursus: Further concepts of functional differentiability

This small subsection gives an overview of functionals where Hadamard differentiability fails. Moreover, additional differentiability concepts for functionals are reviewed, but they do not help us for results on differentiability of regularized M-functionals.

We begin with the recapitulation of examples where Hadamard differentiability fails. One typical example are L-statistics where the underlying distribution has an unbounded support as pointed out in Van der Vaart [2000]. Then, compact differentiability is impossible w.r.t. $||\cdot||_\infty$. Such functionals are studied in Beutner and Zähle [2010] and written in the form

$$T_g(F) := -\int x dg(F(x)),$$

so it is required that g has a compact support in $]0,1[$ to ensure compact differentiability. It is shown in Beutner et al. [2016] that if the support of g contains at least one of the boundary points of $[0,1]$, even the negative expectation value (set $g := \mathrm{id}$) is not compactly differentiable.

The functional T_g covers relevant statistical functionals like the Value at Risk or the Average Value at Risk as remarked in Beutner and Zähle [2010]. In fact, Krätschmer et al. [2012] stated that tail-dependent functionals are in general not compactly differentiable w.r.t. uniform norms.

We note that Beutner and Zähle [2010] proposed a weaker form of compact differentiability which they called **quasi-Hadamard differentiability** and a corresponding functional delta method (Beutner and Zähle [2010]) as well as a Bootstrapped version (Beutner et al. [2016]) for such maps.

Moreover, the functional that maps two Borel probability measures defined on the same underlying complete metric space (X,d) onto their Wasserstein distance is not compactly differentiable (cf. Sommerfeld and Munk [2018]).

There exist other notions of functional differentiability that differ from the abovely mentioned concepts. This is the so-called **Bouligand differentiability** that was investigated by Robinson (Robinson [1988]) and the **Hadamard directional differentiability** introduced by Römisch [2004].

Definition 3.1.4. *Let X, Y be normed real vector spaces and let $T : X \to Y$. Then T is **Bouligand differentiable** at $x \in U \subset X$ if there exists a positively homogeneous map $d_BT(x) : X \to Y$ satisfying*

$$T(x+h) = T(x) + d_BT(x)h + o(h).$$

Definition 3.1.5. *Let X, Y be linear topological spaces and let $T : X_T \subset X \to Y$. Then T is **Hadamard directionally differentiable** at $x \in X_T$ if there exists a map $T' : X \to Y$ such that for all $t_n \in \mathbb{R}$, $h_n \in X$ with $t_n \searrow 0$, $h_n \to h \in X$ and $x + t_nh_n \in X_T$ for all n, it holds that*

$$\frac{T(x+t_nh_n) - T(x)}{t_n} \longrightarrow T'(h).$$

Note that the Bouligand derivative and the Hadamard directional derivative do not have to be linear in contrast to the other concepts, so Bouligand differentiability and Hadamard directional differentiability do not belong to the concept of $\mathcal{R}-$differentiability.

3.2 Basic concepts of quantitative robustness

Every real data analysis requires model assumptions. However, these assumptions are in general not fulfilled, hence the real data differ from data that would have been generated by the ideal model (i.e., under the assumed conditions). Therefore, fitting models by using the real data can be seen as if one analyzed a contaminated data set which affects the quality of the fitted model. This is the main reason to invoke robust statistics (Maronna et al. [2006],Huber and Ronchetti [2009]). In this thesis, we only concern about quantitative robustness. There also exists the concept of qualitative robustness (Hampel [1971]).

The concepts of robust statistics are focused to detect data points that are outliers under the ideal model. It is not desirable to exclude these points from the data set (due to the loss of information) but to find strategies that downweight them, like iteratively reweighted least squares (IRWLS) (Huber and Ronchetti [2009]).

Large parts of this section are based on the book of Rieder (Rieder [1994]), including most of the notation.

3.2.1 Influence curves

We start with the important definition of influence functions that can be found in Hampel et al. [2011] and that was originally stated in Hampel [1968].

Definition 3.2.1. *Let X be a normed function space containing distributions on a probability space $(\Omega, \mathcal{A})$ and let Θ be a normed real vector space. Let $T : X \to \Theta$ be a statistical functional. The* ***influence function or influence curve*** *of T at x for a probability measure P is defined as the derivative*

$$\mathrm{IC}(x,T,P) := \lim_{t \to 0} \left(\frac{T((1-t)P + t\delta_x) - T(P)}{t} \right) = \partial_t \left[T((1-t)P + t\delta_x) \right] \Big|_{t=0}$$

where δ_x denotes the Dirac measure at x.

This is just a special Gâteaux derivative with $h := \delta_x - P$. This is easily seen since

$$T((1-t)P + t\delta_x) = T(P + t(\delta_x - P)) = T(P + th).$$

The influence curve can be regarded as an estimate for the infinitesimal influence of a single observation. If the IC is unbounded, then a single observation can have an infinite impact on the resulting estimator which definitely disagrees with the meaning of ”robustness”. For robustness properties, it is necessary that the influence curve fulfills at least the condition in the following definition (see. e.g. [Hampel et al., 2011, Sec. 2.1]).

Definition 3.2.2. *Let the assumptions in definition 3.2.1 hold and let* $\mathrm{IC}(x, T, P)$ *be some influence function. Then the* ***gross-error sensitivity***

$$\gamma^*(T, P) := \sup_x (|\,\mathrm{IC}(x, T, P)|)$$

can be regarded as a worst-case non-robustness measure for the functional T based on the distribution P. T is called ***B-robust (”bias-robust”) at*** *F if $\gamma^*(T, P)$ is finite.*

Remark 3.2.1. *B-robustness of T at P means that a single observation coming from distribution P cannot have an infinite impact on T. There is a related concept for the change-of-variance function of an M-estimator ([Hampel et al., 2011, Sec. 2.5a, Def. 1]) that gives insights into the behaviour of its asymptotic variance. An M-estimator is called* ***V-robust (”variance-robust”)*** *if the impact on the change-of-variance function is bounded.*

For the sake of completeness, note that there exists a similar concept for Bouligand-differentiable functionals. Then the **Bouligand influence curve** BIC is a special Bouligand derivative satisfying

$$\lim_{t \to 0} \left(\frac{||T((1-t)P + t\delta_x) - T(P) - \mathrm{BIC}(x, T, P)||}{t} \right) = 0.$$

Analogously, robustness can be achieved if the Bouligand influence curve is bounded which has been shown in Christmann and Van Messem [2008], Christmann et al. [2009] to prove robustness of support vector machines. Another differentiability property of functionals is the so-called Hellinger differentiability (see [Rieder, 1994, Ex. 4.2.14]).

The robustification of an estimator can be done by robustifying its influence function. Minimax results for optimally robust influence curves have been established in several works (Rieder [1994], Rieder et al. [2008], Hampel et al. [2011], Fraiman et al. [2001]). However, for guaranteeing optimality of these approaches, it is crucial that the estimator is **asymptotically linear** and that the model is L_2-**differentiable**.

3.2.2 Asymptotic linearity

The L_2-differentiability of a statistical model goes back to Le Cam (LeCam [1970]). Assume for the moment for simplicity that there exist densities. Given a parametric statistical model consisting of probability measures P_θ, one would like to have the following first order expansion of the densities:

$$p_\theta = p_{\theta_0} + (\partial_\theta p_{\theta_0})^T(\theta - \theta_0) + o_{L_2}(||\theta - \theta_0||).$$

Since the remainder term is an L_2-remainder term, writing it down explicitly would require the densities to be square integrable w.r.t. some dominating measure. Of course, this is generally not true, so Le Cam had the idea to go over to the square roots of the densities which are always defined since $p_\theta \geq 0$ almost surely for all θ. Assuming for simplicity that $\partial_\theta p_\theta(x) = 0$ if $p_\theta(x) = 0$ and defining

$$\Lambda_\theta := \frac{\partial_\theta p_\theta}{p_\theta},$$

so that we get

$$\partial_\theta \sqrt{p_\theta} = \frac{1}{2}\frac{\partial_\theta p_\theta}{\sqrt{p_\theta}} = \frac{1}{2}\Lambda_\theta \sqrt{p_\theta},$$

then setting $h := \theta - \theta_0$ leads to

$$\sqrt{p_{\theta_0+h}} = \sqrt{p_{\theta_0}} + \frac{1}{2}\Lambda_{\theta_0}^T h \sqrt{p_{\theta_0}} + o_{L_2}(||h||)$$

$$\Longrightarrow \sqrt{p_{\theta_0+h}} = \sqrt{p_{\theta_0}}\left(1 + \frac{1}{2}\Lambda_{\theta_0}^T h\right) + o_{L_2}(||h||).$$

This heuristically explains the following definition of Le Cam. For part b), we refer to Pupashenko et al. [2015].

Definition 3.2.3. ***a)*** *Let $\mathcal{P} := \{P_\theta \mid \theta \in \Theta\}$ be a family of probability measures on some measurable space $(\Omega, \mathcal{A})$ and let Θ be a subset of $\mathbb{R}^p$. Then $\mathcal{P}$ is L_2-**differentiable at** θ_0 if there exists $\Lambda_{\theta_0} \in L_2^p(P_{\theta_0})$ such that*

$$\left|\left|\sqrt{dP_{\theta_0+h}} - \sqrt{dP_{\theta_0}}\left(1 + \frac{1}{2}\Lambda_{\theta_0}^T h\right)\right|\right|_{L_2} = o(||h||)$$

for $h \to 0$. In this case, the function Λ_{θ_0} is the L_2-derivative and

$$I_{\theta_0} := \mathbb{E}_{\theta_0}[\Lambda_{\theta_0}\Lambda_{\theta_0}^T]$$

is the Fisher information of $\mathcal{P}$ at θ_0.

b) *$\mathcal{P}$ is **continuously** L_2-**differentiable at** θ_0 if for any $h \to 0$, it holds that*

$$\sup_{t\in\mathbb{R}^p, ||t||_2\leq 1}\left(\left|\left|\sqrt{dP_{\theta_0+h}}\Lambda_{\theta_0+h}^T t - \sqrt{dP_{\theta_0}}\Lambda_{\theta_0}^T t\right|\right|_{L_2}\right) = o(1).$$

Note that the L_2-differentiability is a special case of the wider concept of L_r-differentiability (definition A.7.4). The following lemma provides conditions that ensure continuous L_2-differentiability that were originally stated in Hájek [1972] (see also Pupashenko et al. [2015]).

Lemma 3.2.1 (Hájek's Lemma). *Let $\theta_0 \in \Theta$ and let U be an open neighborhood of θ_0. Let $\mathcal{P} := \{p_\theta \mid \theta \in \Theta\}$. If*
i) *the densities p_θ are absolutely continuous for every $\theta \in U$ and for $P_{\theta_0}-$a.e. x,*
ii) *$\partial_\theta p_\theta = \Lambda_\theta p_\theta$ exists for every $\theta \in U$ and $P_{\theta_0}-$a.e. x,*
iii) *the Fisher information $I_\theta = \mathrm{Cov}(\Lambda_\theta)$ exists and is continuous on U (in θ),*
then the model $\mathcal{P}$ is continuously L_2-differentiable in θ_0 with L_2-derivative Λ_{θ_0} and Fisher information I_{θ_0}.

The L_2-differentiability holds for many distribution families, including normal location and scale families, Poisson families, Gamma families, and even for ARMA, ARCH and GPD families (Rieder et al. [2008], Pupashenko [2015]).

A classical example of a distribution family that is not L_2-differentiable is the following one where the support depends on θ.

Example 3.2.1. *Consider the model $\mathcal{P} := \{P_\theta = U[0,\theta] \mid \theta > 0\}$ and let μ be some dominating measure. Assume that this model was L_2-differentiable at any $\theta_0 \in \Theta$. Then there would exist Λ_{θ_0} and*

$$\int\left(\sqrt{p_{\theta_0+h}} - \sqrt{p_{\theta_0}}\left(1+\frac{1}{2}\Lambda_{\theta_0}h\right)\right)^2 d\mu \geq \int_{\theta_0}^{\theta_0+h}(\sqrt{p_{\theta_0+h}})^2 d\mu$$
$$= \frac{1}{\theta_0+h}\int_{\theta_0}^{\theta_0+h} d\mu = \frac{h}{\theta_0+h} = o(h)$$

since the density of P_{θ_0} is zero on $[0,\theta_0]^c$. Since the remainder is of order $o(h)$ instead of $o(h^2)$, the L_2-differentiability is not valid.

Remark 3.2.2. *Note that in the case of GPD families where the support varies in dependence of the mean, scale and shape parameters μ, σ and ξ, respectively, L_2-differentiability also covers singular parts in the sense that $p_{\theta+h}$ may have a different support than p_θ (here, we have $\theta = (\mu,\sigma,\xi)$), provided that the singular mass is not too large (which requires $\xi > -0.5$ in the notation of Pupashenko [2015]).*

Assumption 3.2.1 (**L_2-differentiability**). *When working with influence curves in this thesis, we always assume L_2-differentiability of the underlying model. Other types of differentiability of the model are possible and lead to other types of influence curves (see p.e. [Rieder, 1994, Ex. 4.2.15]), but we do not concern about these techniques in this thesis.*

The next definition is borrowed from [Rieder, 1994, Def. 4.2.16].

Definition 3.2.4. *Let $(\Omega^n, \mathcal{A}^n)$ be a measurable space and let $S_n : (\Omega^n, \mathcal{A}^n) \to (\mathbb{R}^p, \mathbb{B}^p)$ be an estimator. Then the sequence $(S_n)_n$ is **asymptotically linear at** P_{θ_0} if there exists an influence curve $\psi_{\theta_0} \in \Psi_2(\theta_0)$ such that the expansion*

$$S_n = \theta_0 + \frac{1}{n}\sum_{i=1}^n \psi_{\theta_0}(x_i) + o_{P^n_{\theta_0}}(n^{-1/2})$$

holds. The family $\Psi_2(\theta_0)$ of influence curves is defined by the set of all maps η_{θ_0} that satisfy the conditions

i) $\eta_{\theta_0} \in L_2^p(P_{\theta_0})$, ***ii)*** $\mathbb{E}_{\theta_0}[\eta_{\theta_0}] = 0$, ***iii)*** $\mathbb{E}_{\theta_0}[\eta_{\theta_0}\Lambda_{\theta_0}^T] = I_p$

where I_p denotes the identity matrix of dimension $p \times p$.

In this definition, condition i) is vital for integrability and for the application of a central limit theorem to conclude that S_n is asymptotically normal, i.e.,

$$\sqrt{n}(S_n - \theta_0) \circ P^n_{\theta_0} = \left(\frac{1}{\sqrt{n}}\sum_{i=1}^n \psi_{\theta_0}(x_i) + o_{P^n_{\theta_0}}(n^0)\right) \circ P^n_{\theta_0} \xrightarrow{w} \mathcal{N}_p(0, \mathbb{E}_{\theta_0}[\psi_{\theta_0}\psi_{\theta_0}^T]).$$

Condition ii) ensures unbiasedness of the asymptotically linear estimator. The third condition leads to uniform unbiasedness (w.r.t. θ_0), more precisely, if ψ_{θ_0} satisfies i) and ii), [Rieder, 1994, Lemma 4.2.18] shows that the condition iii) is equivalent to

$$\sqrt{n}(S_n - \theta_0)(P^n_{\theta_0 + t_n/\sqrt{n}}) \xrightarrow{w} \mathcal{N}_p(t, \mathbb{E}_{\theta_0}[\psi_{\theta_0}\psi_{\theta_0}^T])$$

for all $t_n \to t$ where t_n, $t \in \mathbb{R}^p$, so the asymptotic normality granted by a central limit theorem will hold locally uniformly over compacts (Rieder [1994]).

The advantages of the ALE are obvious: The arithmetic mean can be computed very fast, it is linear, easy to understand and can be updated rapidly if new observations appear without requiring to refit the whole model.

An extension of this concept arises if one wants to estimate the transformed parameter $\tau(\theta)$ leading to so-called "partial" influence curves in the (to be honest: rather confusing) terminology of Rieder ([Rieder, 1994, Def. 4.2.10], Rieder et al. [2008]).

Definition 3.2.5. *Let $(\Omega^n, \mathcal{A}^n)$ be a measurable space and let $S_n : (\Omega^n, \mathcal{A}^n) \to (\mathbb{R}^q, \mathbb{B}^q)$ be an estimator for the transformed quantity of interest $\tau(\theta)$. Assume that $\tau : \Theta \to \mathbb{R}^q$ is differentiable at $\theta_0 \in \Theta$ where $\Theta \subset \mathbb{R}^p$ and $q \leq p$. Denote the Jacobian by $\partial_{\theta_0}\tau =: D_{\theta_0} \in \mathbb{R}^{q\times p}$. Then the set of* ***partial influence curves (pIC)*** *is defined by*

$$\Psi_2^D(\theta_0) := \{\eta_{\theta_0} \in L_2^q(P_{\theta_0}) \mid \mathbb{E}_{\theta_0}[\eta_{\theta_0}] = 0,\ \mathbb{E}_{\theta_0}[\eta_{\theta_0}\Lambda_{\theta_0}^T] = D_{\theta_0}\}.$$

Then the sequence $(S_n)_n$ is asymptotically linear at P_{θ_0} if there exists a partial influence curve $\eta_{\theta_0} \in \Psi_2^D(\theta_0)$ such that the expansion

$$S_n = \tau(\theta_0) + \frac{1}{n}\sum_{i=1}^n \eta_{\theta_0}(x_i) + o_{P_{\theta_0}^n}(n^{-1/2})$$

is valid.

Remark 3.2.3. *Since it holds that*

$$\Psi_2^D(\theta_0) = \{D_{\theta_0}\psi_{\theta_0} \mid \psi_{\theta_0} \in \Psi_2(P_{\theta_0})\} \tag{3.2.1}$$

(see [Rieder, 1994, Rem. 4.2.11 e)]), the asymptotically linear expansion of transformed estimators in terms of partial influence curves clearly mimics the traditional delta-method (lemma A.7.1).

Asymptotic linearity has been proven for example for asymptotically normal M-, R- and MD-estimators ([Rieder, 1994, Rem. 4.2.17]), so especially for maximum likelihood estimators, quantiles or least squares estimators. An important example for an influence curve is the following one for general M-estimators, see e.g. [Huber and Ronchetti, 2009, Ch. 3.2] or [Maronna et al., 2006, Sec. 3.1].

Theorem 3.2.1. *Suppose that the functional T is implicitly defined by the Z-equation (see also Def. 4.1.1)*

$$\int \psi(x, T(F))dF(x) = 0 \tag{3.2.2}$$

for some distribution F and some function ψ that is differentiable w.r.t. the second argument. Then the influence curve of T is

$$\mathrm{IC}(x, T, F) = \frac{\psi(x, T(F))}{-\mathbb{E}_F[\partial_\theta\psi(x,\theta)|_{\theta=T(F)}]},$$

so it is proportional to the score function ψ.

Example 3.2.2 (**Linear regression**). *An important example is the influence curve for linear regression which is clearly*

$$\mathrm{IC}((x,y),\beta^{LS},F)=\left(\mathbb{E}_F[xx^T]\right)^{-1}x^T(y-x^T\beta_0)$$

for the coefficient vector β_0 supplied by the ideal joint distribution F of (x,y). Just for reasons of completeness, we refer to [Öllerer et al., 2015, Prop. 4.1] where the influence curves of penalized regression estimators have been derived. With $\rho(r)=r^2$, so $\psi(r)=2r$, $\psi'(r)=2$ and $J\equiv 0$, we – of course – get the same influence curve since the term $\psi'(y-x^T\beta_0)$ is nothing but 2 and the expectation of the score is zero, so the nominator in the equation (16) of Öllerer et al. [2015] is only the score evaluated at (x,y).

Example 3.2.3 (**Quantile regression**). *The aim of quantile regression (Koenker and Bassett Jr [1978]) is to fit the conditional $\tau-$quantile of the response vector given the predictors, for $\tau\in]0,1[$. The corresponding empirical risk minimization problem is*

$$\min_{\beta}\left(\sum_{i=1}^{n}\tau(y_i-x_i\beta)_+ +(1-\tau)(y_i-x_i\beta)_-\right) \tag{3.2.3}$$

which can be directly written as

$$\min_{\beta}\left(\sum_{i=1}^{n}L_\tau(y,x,\beta)\right)$$

for the loss function $L_\tau(y,x,\beta)=\rho_\tau(y-x\beta)$ where

$$\rho_\tau(u):=u(\tau-I(u<0))$$

is the check function, sometimes called pinball function (Christmann et al. [2009]). Koenker and Portnoy ([Koenker and Portnoy, 1996, Ch. 3.4]) provided the influence function

$$\mathrm{IC}((x,y),\beta^{quant},F_{XY})=(\mathbb{E}_{F_X}[xx^Tf_{Y|X}(x\beta_0)])^{-1}x\,\mathrm{sign}(y-x\beta_0). \tag{3.2.4}$$

Of course, the L_1-regression that fits the median of the data is the special case of quantile regression for $\tau=0.5$ (note that in the influence curve, the beta's are essentially depending on τ).

Remark 3.2.4. *At this point, regarding these influence curves, we want to emphasize once more that we made no assumptions on the intercept. So, p is just the number of columns of the regressor matrix. It is allowed that there are only $(p-1)$ predictors and that the first column represents the intercept.*

3.2.3 The breakdown point

Another concept of quantitative robustness that generalizes investigating the changes of the estimator with respect to a single observation is the **breakdown point** which is a rather global object. There exists a functional version of the breakdown point ([Hampel, 1971, Sec. 6]) for sequences of estimators which indicates the minimal deviation of a distribution from the ideal distribution in terms of the Prokhorov distance such that the estimator becomes unreliable.

For our purposes, the finite-sample version in the following definition (see e.g. [Maronna et al., 2006, Sec. 3.2.5], introduced in Donoho and Huber [1983]), is more appropriate.

Definition 3.2.6. *Let $S_n : \bigotimes_{i=1}^n \mathcal{X} \to \Theta$ be an estimator based on the sample $x := (x_1, ..., x_n)$. Then the* ***finite-sample breakdown point (FSBDP)*** *is defined as*

$$\frac{1}{n} \max\{m \in \{1, ..., n\} \mid \sup_y(|S_n(y)|) < \infty \wedge \inf_y(d(\partial\Theta, S_n(y))) > 0 \ \forall y \in \mathcal{X}_m\}$$

for some distance d on Θ and for the set $\mathcal{X}_m$ of samples y that have exactly $(n-m)$ points in common with x. In other words, the FSBDP of S_n is the largest amount of data that can be arbitrarily replaced such that the estimator still stays reliable.

Remark 3.2.5. *Note that the FSBDP does not depend on the sample x. The condition that the estimator needs to stay away from the boundary is necessary since for example a scale estimator breaks down if inliers let the estimated scale become zero.*

Remark 3.2.6. *If an estimator is not B-robust, a single observation can have infinite impact, so manipulating a fraction of $1/n$ of the length of sample x lets the estimator break down, thus resulting in a breakdown point of $1/n \to 0$ asymptotically (see [Huber and Ronchetti, 2009, Sec. 3.2.3]).*

Remark 3.2.7. *The maximal breakdown point for equivariant location estimators is $1/2$ ([Maronna et al., 2006, Sec. 3]).*

3.3 Contamination models

We already mentioned that a goal of robust statistics is to find reliable estimators even on contaminated data and we already worked with some kind of contaminated distribution when having concerned about influence curves (see definition 3.2.1), without giving a concrete definition of "contamination". This section is not proposed give an overview of all existing contamination models but only to show the mathematical framework for working with contaminated models which is necessary for the rest of this thesis.

We begin with the definition of contamination balls ([Rieder, 1994, Sec. 4.2]).

Definition 3.3.1. *Let $\mathcal{P} := \{P_\theta \mid \theta \in \Theta\}$ be a parametric model on some measurable space $(\Omega, \mathcal{A})$. Let P_{θ_0} be the ideal distribution ("model distribution", [Rieder, 1994, Sec. 4.2]). Then a* ***contamination model*** *is the set of all distributions given by the system*

$$\mathcal{U}_*(\theta_0) := \{U_*(\theta_0, r) \mid r \in [0, \infty[\}$$

of ***contamination balls***

$$U_*(\theta_0, r) = \{Q \in \mathcal{M}_1(\mathcal{A}) \mid d_*(P_{\theta_0}, Q) \leq r\}.$$

The radius r is also called "contamination radius".

To emphasize different standard types of contamination balls, a subscript (here represented by the "$*$") is usually added to concretize the contamination model.

Example 3.3.1. *A convex contamination model $\mathcal{U}_c(\theta_0)$ is the system of contamination balls*

$$U_c(\theta_0, r) = \{(1-r)_+ P_{\theta_0} + \min(1, r)Q \mid Q \in \mathcal{M}_1(\mathcal{A})\}.$$

Thus, the contamination models that we already defined in the previous section were convex contamination models. Many other metrics have entered the theory of robust statistics, see [Rieder, 1994, Sec. 4.2] for an overview.

The convex contamination model is very simple to simulate from (see e.g. Kohl [2005]). Having a realization x_i from the ideal distribution P_{θ_0}, one fixes a distribution Q on the same measurable space and generates the data point

$$(1-u_i)x_i + u_i\tilde{x}_i$$

for a realization $\tilde{x}_i$ of Q and a realization u_i of $U_i \sim Bin(1, r)$. Thus, with a probability of r, we get an observation of the distribution Q, otherwise from the ideal distribution.

While in standard estimation problems like estimating a location or scale parameter we just have a sample of observations, there is far more flexibility to define outliers in for example regression settings. See Kohl [2005] and Rieder [1994] for the following definition and more details.

Definition 3.3.2. *Let $P_{\theta_0}(dx, dy)$ be a distribution on the Borel set $\mathbb{B}^{p+1}$. Writing $P_{\theta_0}(dx, dy)$ $= P_{\theta_0}(dy|x)P_{\theta_0}(dx)$, the **conditional convex contamination neighborhood (error-free-variables neighborhood)** is the set of all distributions*

$$Q(dy|x) = (1 - r\epsilon(x))P_{\theta_0}(dy|x) + r\epsilon(x)M(dy|x)$$

*for any Markov kernel $M(dy|x)$ mapping from $\mathbb{R}^p$ into $\mathbb{B}$ and any contamination curve ϵ (see [Rieder, 1994, Sec. 7.2.2]). In contrast, **unconditional convex contamination neighborhoods (errors-in-variables neighborhoods)** allow also $P_{\theta_0}(dx)$ to be contaminated.*

Remark 3.3.1 (**Regression outliers**). *We will always refer to $y-$outliers when facing contaminated observations from error-free-variables neighborhoods and to $x-$outliers if the regressors are contaminated.*

Remark 3.3.2 (**Leverage points**). *It is evident that errors-in-variables are harder to deal with than just contaminated responses. This has been mentioned for example in Alfons et al. [2013] and was studied in detail in Maronna et al. [2006]. As explained in the latter reference, $x-$outliers can be seen as leverage points, but it is hard to decide whether a leverage point is a "good" leverage point or a "bad" leverage point. While the first type of leverage points are still concordant with the regression hyperplane, i.e., they produce only small residuals, bad leverage points can be treated as outliers w.r.t. the regression model, so that they would drastically mislead the regression hyperplane if they were not downweighted (see also Rousseeuw and Hubert [2011]).*

Note that we restricted ourselves so far to case-wise outliers, i.e., either a whole row in the regressor matrix or in the response follows the ideal distribution or not, but Alqallaf and co-authors (Alqallaf et al. [2009]) proposed a more realistic scenario which allows the entries of the regressor matrix to be independently perturbed. See also Agostinelli et al. [2015] for the notation.

Definition 3.3.3. *Let $\mathbb{R}^p \ni X \sim P_{\theta_0}$. Let $U_1, ..., U_p \sim Bin(1, r)$ i.id.. Then the **cell-wise convex contamination model** consists of all sets*

$$U^{cell}(\theta_0, r) := \{Q \mid Q = \mathcal{L}(UX + (1 - U)\tilde{X})\}$$

where $\tilde{X} \sim \tilde{Q}$ for any distribution $\tilde{Q}$ on the same measurable space as P_{θ_0} and the matrix U with diagonal entries U_i.

Remark 3.3.3. *In the regression setting, we assume each row X_i of the regressor matrix comes from a cell-wise contamination model. As pointed out by Alqallaf et al. [2009], if all U_j are perfectly dependent, one either gets the original row or a fully contaminated row which is the classical convex contamination model of Tukey and Huber in definition 3.3.1.*

The cell-wise contamination model should reflect the curse of dimensionality since the probability that at least one case is contaminated increases with the dimension p of the rows. Of course, robust estimation in the presence of cell-wise outliers can be much harder than with case-wise outliers since a single contaminated cell already makes an observation an outlier (see Öllerer and Croux [2015], Croux and Öllerer [2016]). While the cited references, including Loh et al. [2018], concentrated on precision matrix estimation, Agostinelli et al. [2015] studied location and scatter estimation in multivariate data. Leung et al. [2016] and Leung et al. [2017] proposed robust methods for regression and scatter estimation for data with cell-wise contamination.

3.4 Robustness properties of existing variable selection procedures

This section reviews some selected popular sparse learning algorithms and summarizes their robustness properties. Note that some of these algorithms claim robustness, but in fact they are only robust against $y-$outliers but not against leverage points. We admit that the raw algorithms that are presented in the subsequent chapters except for section 18.2 are also not robust, not even against $y-$outliers.

The **standard Lasso**, i.e.,

$$\hat{\beta}^{Lasso} = \underset{\beta}{\operatorname{argmin}} \left(\frac{1}{n} \sum_i (Y_i - X_i\beta)^2 + \lambda \sum_j |\beta_j| \right),$$

is of course not robust due to its non-robust loss function. In addition, it has been shown in Alfons et al. [2013] that the FSBDP (see definition 3.2.6) for the Lasso is $1/n$ by applying [Alfons et al., 2013, Thm. 1]. That is, a single outlier in the data can cause the Lasso estimator to become worthless. The same is true for the **adaptive Lasso** (Zou [2006])

$$\hat{\beta}^{ALasso} = \underset{\beta}{\operatorname{argmin}} \left(\frac{1}{n} \sum_i (Y_i - X_i\beta)^2 + \lambda \sum_j \frac{|\beta_j|}{|\hat{\beta}_j^{init}|} \right)$$

for the coefficient vector $\hat{\beta}^{init}$ fitted by a $\sqrt{n}-$consistent procedure or especially by the standard Lasso in the first step.

Implementations of the Lasso in R are given in the package `lars` (Hastie and Efron [2013]) using the least angle procedure from Efron et al. [2004] and in the package `glmnet` (Friedman et al. [2010]) where the coordinate-descent algorithm from Friedman et al. [2007] has been realized.

The **LAD-Lasso** introduced in Wang et al. [2007] solves the problem

$$\hat{\beta}^{LADL} = \underset{\beta}{\operatorname{argmin}} \left(\frac{1}{n} \sum_i |Y_i - X_i\beta| + \lambda \sum_j |\beta_j| \right).$$

Consistency results are presented in Wang et al. [2007]. While being robust against $y-$outliers, it is sensible to $x-$outliers (Chang et al. [2018]). Additionally, using once more [Alfons et al., 2013, Thm. 1], its FSBDP is $1/n$. Besides its insufficient robustness properties, Lambert-Lacroix et al. [2011] and Chang et al. [2018] point out that LAD-Lasso is less efficient than standard least squares and the adaptive Lasso in the ideal model. An implementation can be found in the R$-$package `flare` (Li et al. [2018]).

The **robustified LARS (RLARS)** (Khan et al. [2007]) is based on a plug-in approach replacing the non-robust estimates mean, covariance and correlation by their robust counterparts. The robust correlation uses a multivariate winsorization provided in Khan et al. [2007]. There are no theoretical results about oracle properties of RLARS. An implementation is given in the package `robustHD` (Alfons [2016]). Despite having empirically been shown to be robust against $x-$ and $y-$outliers (Chang et al. [2018]), due to its main weakness that there is no objective function (Alfons et al. [2013]), it is inappropriate for regarding asymptotic linearity and it does not fit into the framework that we will propose later in the thesis.

Sparse Least Trimmed Squares (SLTS) is the regularized counterpart of LTS (Rousseeuw [1984]) provided in Alfons et al. [2013] and solving

$$\hat{\beta}^{SLTS} = \underset{\beta}{\operatorname{argmin}} \left(\frac{1}{h} \sum_{i=1}^{h} (r^2(\beta))_{i:n} + \lambda \sum_j |\beta_j| \right)$$

where $r(\beta)_{i:n}$ is the $i-$th smallest residual corresponding to the coefficient vector β. The algorithm essentially computes Lasso models and performs a concentration step (cf. Rousseeuw and Van Driessen [1999] for MCD, Rousseeuw and Van Driessen [2006] for LTS) to choose the instances producing the lowest residuals. A reweighted version of SLTS is also provided in Alfons et al. [2013] to improve the false positive rate and the efficiency in data sets without outliers. Alfons et al. [2013] did not provide asymptotic properties of SLTS, but they showed that a clever choice of the subset size h leads to reasonably high breakdown points. The STLS is robust against $x-$ and $y-$outliers, but it turns out that it gets rather slow for a large number of predictors (cf. Chang et al. [2018]). An R$-$implementation of SLTS is available in the package `robustHD`.

The **Huberized Lasso (HLasso)**, originally introduced in Rosset and Zhu [2007] and studied in Chen et al. [2010a] and Chen et al. [2010b], is a variant of the Lasso where the quadratic loss function is replaced by Huber's loss function. [Chen et al., 2010a, Cor. 2.2+Thm. 2.3] show under which conditions the HLasso is $\sqrt{n}-$consistent resp. variable selection consistent (see definition 2.3.1) and asymptotically normal. Furthermore, an adaptive version, the AHLasso, is provided by combining the ALasso with the Huber loss. The HLasso is highly efficient and robust against $y-$outliers, but not against $x-$outliers (see Chang et al. [2018]). An implementation of the HLasso using a semi-smooth Newton coordinate descent algorithm (see Yi and Huang [2017]) is available in the package `hqreg` (Yi [2017]).

The **Tukey-Lasso (TLasso)** (Chang et al. [2018]) can be regarded as a robustification of the ALasso. It solves

$$\hat{\beta}^{TLasso} = \underset{\beta}{\operatorname{argmin}} \left(2 \sum_{i=1}^{n} \rho_T \left(\frac{Y_i - X_i \beta}{\hat{\sigma}} \right) + \lambda \sum_j \hat{w}_j |\beta_j| \right)$$

where ρ_T is Tukey's biweight function (cf. [Maronna et al., 2006, Sec. 2.2.4]). The scale $\hat{\sigma}$ is computed via an S-estimator whereas the weights $\hat{w}_j = 1/|\hat{\beta}_j^{MM}|$ come from an MM-regression estimator, hence the adaptive nature of this approach. The TLasso is robust against $x-$ and $y-$outliers and provides variable selection consistency as well as asymptotic normality under conditions given in [Chang et al., 2018, Thm. 1]. Following the experiments in Chang et al. [2018], the TLasso outperforms all other Lasso algorithms. Compared to SLTS and RLARS, the computation time of TLasso is significantly lower. It is implemented in MATLAB using an accelerated proximal gradient (APG) method.

3.5 One-Step estimators

Suppose we have a parametric model $\mathcal{P}$ and a "nice" influence curve $\psi_\theta \in \Psi_2(\theta)$.

Is it possible to derive an estimator S_n that has exactly this influence curve?

The following theorem ([Rieder, 1994, Thm. 6.4.8a)]) provides the answer.

Theorem 3.5.1. *Let $\mathcal{P}$ be L_2-differentiable at $\theta_0 \in \Theta$. Let $\hat{\theta}_n \in \Theta$ be a $\sqrt{n}-$consistent estimator for θ_0. Let $(\psi_{\theta,n})_{n\in\mathbb{N}}$ be an approximating sequence for $\psi_{\theta_0} \in \Psi_2(\theta_0)$ with the following properties:*

$$\textbf{\textit{i)}}\ \lim_n \left(\int ||\psi_{n,\hat{\theta}_n} - \psi_{\theta_0}||^2 dP_{\theta_0} \right) = 0, \quad \textbf{\textit{ii)}}\ \sup_{x\in\mathcal{X}} (||\psi_{n,\hat{\theta}_n}(x)||) = o(\sqrt[4]{n}),$$

$$\textbf{\textit{iii)}} \int \psi_{n,\hat{\theta}_n} dP_{\hat{\theta}_n} = o\left(\frac{1}{\sqrt{n}} \right).$$

Then, the estimator

$$S_n^1 := \hat{\theta}_{n*} + \frac{1}{n}\sum_{i=1}^{n} \psi_{n,\hat{\theta}_{n*}}(x_i) \tag{3.5.1}$$

for a discretized version $\hat{\theta}_{n*}$ of $\hat{\theta}_n$ (cf [Rieder, 1994, Ch. 6.4.2]) is asymptotically linear at P_{θ_0} with influence curve ψ_{θ_0}.

Definition 3.5.1. *We call the estimator S_n^1 in equation (3.5.1)* ***One-Step estimator*** *or just* ***One-Step****.*

In fact, [Rieder, 1994, Thm. 6.4.8] has much more implications, but theorem 3.5.1 already clarifies that given a suitable $\sqrt{n}-$consistent initial estimator for θ_0, one can simply construct an ALE for θ_0 with a pre-determined influence curve.

Remark 3.5.1. *As described in Kohl et al. [2010], one can show that the estimator S_n inherits the breakdown point of $\hat{\theta}_n$ if the initial estimator is already asymptotically linear.*

The One-Step estimator can be identified with a Newton iteration as described in Bickel [1975] for the linear model. According to Green [1984], the idea of using a Newton method essentially goes back to Fisher (Fisher [1925]). Following [Ruckdeschel, 2001, Ch. 9.5.3], we write

$$\operatorname*{zero}_{\theta}\left(\frac{1}{n}\sum_i \psi_\theta(x_i)\right)$$

and given an initial estimator $\hat{\theta}_n$, the Newton step would be

$$\tilde{S}_n^1 = \hat{\theta}_n - \left[\partial_\theta\left(\frac{1}{n}\sum_i \psi_{\hat{\theta}_n}(x_i)\right)\right]^{-1} \cdot \frac{1}{n}\sum_i \psi_{\hat{\theta}_n}(x_i).$$

Since the arithmetic mean tends to the expectation of the influence curve, it asymptotically holds that

$$\partial_\theta\left(\frac{1}{n}\sum_i \psi_{\hat{\theta}_n}(x_i)\right) = \partial_\theta \mathbb{E}_\theta[\psi_{\hat{\theta}_n}] = -\mathbb{E}_\theta[\psi_{\hat{\theta}_n}\Lambda_\theta^T].$$

The second equality is true if the L_2-differentiability of the parametric model holds (see also lemma 4.1.2) and in the case of consistency of $\hat{\theta}_n$, the right hand side equals $-I_k$ due to property iii) of the influence curve in definition 3.2.4. Putting everything together, one once more gets the One-Step estimator

$$S_n^1 = \hat{\theta}_n + \frac{1}{n}\sum_i \psi_{\hat{\theta}_n}(x_i).$$

One can easily extend the One-Step construction principle if the One-Step estimate is taken as initial estimator.

Definition 3.5.2. *Let the estimator S_n^1 be the One-Step estimator for θ_0. Then define*

$$S_n^2 := S_n^1 + \frac{1}{n}\sum_i \psi_{S_n^1}(x_i).$$

Generally, for $k \geq 2$,

$$S_n^k := S_n^{k-1} + \frac{1}{n}\sum_i \psi_{S_n^{k-1}}(x_i)$$

*is called $k-$**Step estimator** or just $k-$**Step**.*

Given the initial estimator, the computation of the $k-$Step is clearly extremely fast since it just requires the evaluation of an arithmetic mean. This is especially useful if training points are added after computing the estimator like in Online Learning contexts. Then, instead of performing cumbersome computation again, the estimator can be easily updated by a $k-$Step.

Furthermore, these types of estimators are especially important if one works with optimally robust influence curves, i.e., robust versions of given influence curves that solve a certain optimization problem. Usually, one solves a Hampel-type optimization problem that requires to find the (partial) influence curve in the set $\Psi_2(\theta_0)$ (or $\Psi_2^D(\theta_0)$) whose trace of covariance under P_{θ_0} is minimal subject to a bound on the supremal bias on the respective contamination neighborhood (see Chapter 5 of Rieder [1994] for details and solutions for different metrics). For numerous applications of the One-Step construction principle, we refer to Kohl [2005].

The adaption of $k-$Step estimation to model selection is straightforward. Having selected a set $J \subset \{1, ..., p\}$ of columns, w.l.o.g. ordered in an ascending sense, the starting estimator is a $\sqrt{n}-$consistent estimator which is essentially based on the reduced data set $(X_{\cdot,J}, Y)$. This is the usual procedure for estimation. We just have to account for the influence curve.

In fact, by the previously performed model selection, we just have to estimate the reduced parameter θ_J. This reduction can be thought of the mapping

$$\tau : \mathbb{R}^p \to \mathbb{R}^{|J|}, \quad \theta \mapsto (\theta_j)_{j \in J}.$$

Since τ is differentiable and $|J| =: q \leq p$, we can compute the partial influence curve

$$\partial_\theta \tau(\theta)\psi_\theta = (\psi_\theta)_J.$$

Due to the relation (3.2.1), this influence curve is already admissible if $\psi_\theta \in \Psi_2(P_\theta)$.

Thus, the One-Step estimator for θ_J is given by

$$S_{n,J}^1 = \widehat{(\theta_J)_n} + \frac{1}{n}\sum_i (\psi_{\widehat{(\theta_J)_n}})_J(X_i, Y_i). \tag{3.5.2}$$

The extension to the $k-$Step is straightforward. Formally, given the $k-$Step $S_{n,J}^k$ for the parameter θ_J, the resulting $p-$dimensional parameter is clearly given by

$$S_n^k = \tau^{-1}(S_{n,J}^k).$$

Remark 3.5.2. *Assume for simplicity that $J = \{1, ..., q\}$. Note that due to remark 3.2.3, the partial influence curve $(\psi_\theta)_J$ satisfies*

$$\mathbb{E}[(\psi_\theta)_J \Lambda_\theta^T] = (I_q, 0_{p-q}) \in \mathbb{R}^{q \times p}$$

which indeed is obviously true since the last $(p-q)$ rows of I_q just have been dropped.

Remark 3.5.3. *For very large data sets, the computation of the One-Step may take a long time due to expensive evaluation of the influence curve and/or many observations. Since one just has to compute an arithmetic mean, one can parallelize it as usual, i.e., divide the vectors X and Y into partitions and compute the corresponding summands separately on different cores, so that they just need to be aggregated to get the arithmetic mean.*

Chapter 4

Compact differentiability of regularized M-functionals

Modern data analysis frequently requires working with high-dimensional data where the number of predictors is much larger than the number of observations. Model fitting consists of minimizing some loss function by structural risk minimization, i.e., the loss is penalized by a term that encourages sparsity of the predicted parameter. Popular choices of this penalty term include the Lasso penalty (Tibshirani [1994]) or the elastic net penalty (Zou and Hastie [2005]), among a vast variety of other variants which we touched on before.

Estimators of this kind can be represented as statistical functionals, more precisely as regularized M-functionals. Such functionals are highly non-linear. But in fact, if the functional satisfies some regularity properties which in our case will be compact differentiability, then an infinite-dimensional Delta-method provides an asymptotic linear expansion in terms of influence curves up to some error term depending on the number of observations so that the estimator is essentially representable as an arithmetic mean of influence curves.

Although there are already results on asymptotic linearity for certain estimators (Van de Geer [2016], LeDell et al. [2015]), there does not yet seem to be a general theory on compact differentiability of regularized M-estimators which is the intention of this chapter. Öllerer et al. [2015] indeed provided influence curves for regularized M-estimators like the Lasso, but compact differentiability of the corresponding statistical functional is needed to use them for the asymptotic linear expansion. Additionally, our results can also handle sequences of regularization parameters.

The rest of this chapter is organized as follows. In section 4.1, we introduce M-estimators and M-functionals by their formal definition. Then, we concentrate on the task under which conditions an M-estimator can be rewritten as a Z-estimator, even if the integrand is not differentiable. The next section is devoted to a result that proves compact differentiability of M-functionals and the validity of the resulting asymptotic linear expansion as well as asymptotic normality (Rieder [1994], Jain and Marcus [1975]).

In the main theoretical part of this chapter, section 4.5, we use these preliminary results to determine the conditions that are necessary to expand regularized M-estimators asymptotically linearly. We investigate each of the respective conditions and transfer them into the statistical learning context with convex loss functions and convex regularization terms and extend the results to ranking problems. We will heavily make use of an approximation lemma of Avella-Medina [2017]. Concrete examples of machine learning algorithms that fit into this framework are given afterwards.

4.1 M-estimators and M-functionals

We start with the definition M-estimators and M-functionals in the spirit of Van der Vaart [2000] and Rieder [1994]. Since this concept is not restricted to regression settings, we think of L as a general loss or goodness function in this section which does not need to satisfy the property in definition 2.1.1.

Definition 4.1.1. *Let $L : \mathbb{R}^m \times \Theta \to \mathbb{R}$. Then an estimator solving*

$$\max_{\theta \in \Theta} \Big(\theta \mapsto \int_{\mathbb{R}^m} L(x, \theta) dF(x) \Big) \quad \textit{or} \quad \min_{\theta \in \Theta} \Big(\theta \mapsto \int_{\mathbb{R}^m} L(x, \theta) dF(x) \Big)$$

*is called an **M-estimator** of θ. If L is differentiable w.r.t. θ with derivative $\varphi : \mathbb{R}^m \times \Theta \to \mathbb{R}^p$, then the maximization resp. minimization can be done solving the estimating equation*

$$\operatorname*{zero}_{\theta \in \Theta} \Big(\theta \mapsto \int_{\mathbb{R}^m} \varphi(x, \theta) dF(x) \Big).$$

*A functional $T : \mathcal{C}(\Theta, \mathbb{R}^p) \to \Theta$ is called an (vector-valued) **M-functional** if T finds a zero of $f \in \mathcal{C}(\Theta, \mathbb{R}^p)$, i.e., if a zero exists, it holds that*

$$f(T(f)) = 0.$$

Note that the solution of the estimating equation that intends to find the zero is sometimes referred to as a **Z-estimator** (Van der Vaart [2000]).

Remark 4.1.1 (**Empirical counterparts**). *If data x_i, $i = 1, ..., n$ are given, the empirical M-estimator of θ is the solution of*

$$\max_{\theta \in \Theta} \left(\theta \mapsto \frac{1}{n} \sum_{i=1}^{n} L(x_i, \theta) \right) \quad or \quad \min_{\theta \in \Theta} \left(\theta \mapsto \frac{1}{n} \sum_{i=1}^{n} L(x_i, \theta) \right).$$

In the case of Z-estimators, one solves

$$\operatorname*{zero}_{\theta \in \Theta} \left(\theta \mapsto \int \varphi(x, \theta) d\hat{F}_n(x_1, ..., x_n) = \frac{1}{n} \sum_{i=1}^{n} \varphi(x_i, \theta) \right)$$

where $\hat{F}_n$ denotes the empirical cumulative distribution function defined by $x_1, ..., x_n$.

Remark 4.1.2. *Note that a standard example of M-estimation is maximum likelihood estimation. In this case, L is given by the log-likelihood and φ_θ is the partial derivative of the log-likelihood w.r.t. the parameter θ.*

It remains to investigate under which conditions the integral and the derivative are allowed to be interchanged when going over to Z-estimators. We recap the following well-known result.

Lemma 4.1.1. *Let $(\mathcal{X}, \mathcal{A}, \mu)$ be a measure space. Let $f : \mathcal{X} \times \Omega \to \mathbb{R}$ be integrable w.r.t. x for any $\omega \in \Omega$ where $\Omega \subset \mathbb{R}^p$ is an open subset. If the partial derivative $\partial_\omega(x, \omega)$ exists for every $x \in \mathcal{X}$ and if there exists $g \in L^1(\mu)$ such that for any $\omega \in \Omega$ it holds that $|\partial_\omega(x, \omega)| \leq g(x)$, then*

$$\partial_{\omega_i} \int f(x, \omega) d\mu = \int \partial_{\omega_i} f(x, \omega) d\mu.$$

Trivially, if f is continuously partially differentiable w.r.t. ω, then the derivation under the integral sign is allowed. However, the regularity assumptions made on f can be weakened by regarding the difference quotients and making use of the notion of uniform integrability (definition A.2.1).

Concerning the M-estimator from definition 4.1.1, let us denote the difference quotients w.r.t. the second argument by

$$\frac{1}{h} \left(\int L(x, \theta + he_j) dF(x) - \int L(x, \theta) dF(x) \right)$$
$$= \int \frac{L(x, \theta + he_j) - L(x, \theta)}{h} dF(x) =: \int f_{h,j}(x) dF(x).$$

Then the following lemma answers the question when the expectation of these difference quotients converges to $\mathbb{E}_F[\varphi]$.

Lemma 4.1.2. *Let $L : \mathbb{R}^m \times \Theta \to \mathbb{R}$ be differentiable w.r.t. θ with partial derivative $\varphi := \partial_\theta(L(x,y),\theta)$. Let F be a cumulative distribution function on $(\mathbb{R}^m, \mathbb{B}^m)$ with the corresponding probability measure P. Let φ be measurable w.r.t. the image measure $\mu := P \circ X^{-1}$. If the families $\{f_{h,j}\}$ are uniformly integrable w.r.t. μ for all $j = 1, ..., p$, then it holds that*

$$\int f_h d\mu \xrightarrow{h\to 0} \int \varphi d\mu$$

where f_h denotes the vector with components $f_{h,j}$.

Proof. *For the sake of notional simplicity, we just proof the one-dimensional case and suppress the double subscript. By the transformation formula, we have the equality*

$$\int f_h \circ X dP = \int f_h d(P \circ X^{-1}) = \int f_h d\mu$$

and μ is a finite measure on the image space $(\Omega, \mathcal{A})$.

Since L is partially differentiable w.r.t. θ, the difference quotients f_h converge pointwise to φ for $h \to 0$. Then, Egorov's theorem (theorem A.3.1) states that the convergence even holds almost uniformly, that is, for any $\delta > 0$ there exists $A \in \mathcal{A}$ with $\mu(A) < \delta$ such that $(f_h)_h$ converges uniformly on the complement of A, i.e.,

$$\lim_{h\to 0}\left(\sup_{\omega\in\Omega\setminus A}(|f_h(\omega) - \varphi(\omega)|)\right) = 0.$$

Thus, for any $\epsilon > 0$ there exists h_0 such that for every $h \le h_0$ it holds that $\{|f_h - \varphi| \ge \epsilon\} \subset A$, so we have

$$\mu(\{|f_h - \varphi| \ge \epsilon\}) < \delta$$

and hence the sequence $(f_h)_h$ converges in measure (cf. p.e. [Bogachev, 2007a, Thm. 2.2.3]).

By assumption, φ is $\mu-$measurable. Since μ is a finite measure, $\{f_h\}$ is uniformly integrable and $f_h \to \varphi$ in measure, the theorem of Lebesgue-Vitali (theorem A.2.1) is applicable and supplies integrability of φ w.r.t. μ, so the expression $\int \varphi(x)dF(x)$ is reasonable and the desired convergence $f_h \to \varphi$ in $L^1(\mu)$ holds.

□

Remark 4.1.3. *Note that the uniform integrability of the family $\{f_h\}$ in the previous lemma really is an assumption. It does not suffice to assume just the integrability of each f_h because in this case, uniform integrability of this set would require that f_h converges to φ uniformly, but as seen in the proof, only almost uniform convergence can be concluded.*

However, if the assumption of integrability is sharpened in the following way, then the uniform integrability is also given and the lemma is valid.

Remark 4.1.4. *The assumption of uniform integrability of the family $\{f_h\}$ in the previous lemma can be equivalently replaced by the assumption that each f_h is integrable w.r.t. μ and that the closure of this family is compact in $L^1(\mu)$ by using the criterion of Dunford-Pettis (theorem A.2.2).*

4.2 Weak derivatives and the density method

In contrast to the case above, there also exists an approach to deal with non-differentiable integrands. This typically occurs in the context of option pricing where the expectation over a non-differentiable payoff function is taken (see Khedher [2011]). This procedure is referred to as the **density method** and was originally stated in Broadie and Glasserman [1996]. In the framework of Khedher [2011], the goal is to differentiate functionals of the form

$$F(x) := \mathbb{E}[f(x+Y)]$$

for a random variable Y and a measurable function f such that $f \in L^1(\mu)$ w.r.t. x and to write the derivative as

$$F'(x) = \mathbb{E}[f(x+Y)w]$$

for some weight function w. Khedher argued that in the case $f \in L^1(\mathbb{R})$, one needs a strictly positive and differentiable density p_Y of Y w.r.t. the Lebesgue measure λ. Then, provided that $f(\cdot)p_Y'(\cdot - x)$ is uniformly bounded by an integrable function on $\mathbb{R}$, one can represent the desired derivative by

$$F'(x) = \partial_x \int f(x+y)p_Y(y)\lambda(dy) = \int \partial_x f(y)p_Y(y-x)\lambda(dy) =$$
$$-\int f(y)p_Y'(y-x)\lambda(dy) = \mathbb{E}[f(x+Y)(-\partial_y \ln(p_Y(Y)))]$$

using integration by parts.

Clearly, this idea mimics the concept of weak derivatives (see definition A.6.1) where integration by parts transfers the derivative of f to the derivative of the test function. However, the density method requires the existence of such a density. The question whether a density w.r.t. to some measure exists is strongly related to absolute continuity (Capatina et al. [2014]) and the Radon-Nikodym property of the image space (cf. e.g. Bárcenas [2003], Diestel and Uhl [1977]).

Remark 4.2.1. *We need a further extension since we have to deal with parametric densities. One way out would be to assume that the required conditions that are needed for the existence* *of the Radon-Nikodym density hold for every fixed θ in the parameter space. We would like to refer to [Bogachev, 2007b, 6.10.72] where already an extension of the Radon-Nikodym theorem is provided. Putting this theorem into our framework, we see that $(\Theta, \mathbb{B} \cap \Theta)$ is a measurable space and that the sigma-algebra $\mathcal{A}$ of $\mathcal{X}$ is countably generated. The latter is true since Borel sigma-algebras of separable metric spaces are countably generated (see e.g. [Bogachev, 2007b, Ex. 6.5.2]) and since we always assume that $\mathcal{X} \subset \mathbb{R}^p$. Then, because $\mathbb{R}^p$ is separable and subspaces of separable metric spaces are separable (see. e.g. Werner [2006]), the assertion follows.*

Following an idea in the proof of [Rieder, 1994, Thm. 6.2.5] or [Ruckdeschel, 2001, Ch. 9.5.3], an interesting special case is given by the parametric densities of the underlying L_1-differentiable (see definition A.7.4) statistical model.

Lemma 4.2.1. *Let $\mathcal{P} = \{P_\theta \mid \theta \in \Theta\}$ be L_1-differentiable at θ_0 with L_1-derivative Λ_{θ_0}. Let the densities p_θ w.r.t. the Lebesgue-measure λ on $(\mathcal{X}, \mathcal{A})$ for $\mathcal{X} \subset \mathbb{R}^p$ be continuously differentiable and let $f : \mathcal{X} \times \Theta \to \mathbb{R}$ be some function such that $f_\theta := f(\cdot, \theta)$ is integrable w.r.t. λ and such that $\mathbb{E}_\theta[f_\theta] = c$ for every θ in a neighborhood of θ_0. If for every h in a neighborhood of 0, $f_{\theta_0+h}^T \Lambda_{\theta_0} p_{\theta_0}$ is bounded by a $\lambda-$integrable function, then*

$$\partial_\theta \mathbb{E}_{\theta_0}[f_{\theta_0}] = \mathbb{E}_{\theta_0}[-f_{\theta_0}^T \Lambda_{\theta_0}].$$

Proof. *We restrict ourselves to the one-dimensional case. By L_1-differentiability in θ_0, we get*

$$p_{\theta_0+h} = p_\theta(1 + \Lambda_{\theta_0} h) \Longleftrightarrow \frac{p_{\theta_0+h} - p_{\theta_0}}{h} = p_{\theta_0} \Lambda_{\theta_0} \tag{4.2.1}$$

where we suppressed the $o(h)$ term. Then, assuming that $\mathbb{E}_\theta[f_\theta] = c$ in a neighborhood of θ_0, we have

$$\partial_\theta \mathbb{E}_{\theta_0}[f_{\theta_0}] = \lim_{h \to 0} \left(\int \frac{f_{\theta_0+h}(x) - f_{\theta_0}(x)}{h} p_{\theta_0}(x) d\lambda(x) \right)$$

$$= \lim_{h \to 0} \left(\int \frac{f_{\theta_0+h}(x) p_{\theta_0}(x)}{h} d\lambda - \int \frac{f_{\theta_0}(x) p_{\theta_0}(x)}{h} d\lambda \right)$$

$$= \lim_{h \to 0} \left(\int \frac{f_{\theta_0+h}(x) p_{\theta_0}(x)}{h} d\lambda - \int \frac{f_{\theta_0+h}(x) p_{\theta_0+h}(x)}{h} d\lambda \right)$$

$$= \lim_{h \to 0} \left(- \int f_{\theta_0+h}(x) \frac{p_{\theta_0+h}(x) - p_{\theta_0}(x)}{h} d\lambda \right)$$

$$\overset{(4.2.1)}{=} \lim_{h \to 0} \left(- \int f_{\theta_0+h}(x) p_{\theta_0}(x) \Lambda_{\theta_0}(x) d\lambda(x) \right) = \mathbb{E}_{\theta_0}[-f_{\theta_0} \Lambda_{\theta_0}]$$

by the regularity assumptions.

□

Remark 4.2.2. *Obviously, one can list conditions under which the integrability as required in the lemma are satisfied, but we restrict ourselves to the stated result.*

Remark 4.2.3. *The abovely stated lemma is especially useful if $f_\theta = \psi_\theta$ is an influence function. By standard assumptions (see definition 3.2.4), $\mathbb{E}_\theta[\psi_\theta] = 0$ for every θ, so the expectation of an influence curve in its ideal model should always be centered to make the asymptotic linear expansion unbiased. Furthermore, a general assumption is $\psi_\theta \in L_2(P_\theta)$, so the integrability condition is valid. Since we always assume L_2-differentiable models when concerning about influence curves, the L_1-differentiability directly follows from lemma A.7.2.*

However, we will see in section 4.5 that we will find a much more elegant way to deal with non-differentiable target functions.

4.3 Asymptotic linearity of M-estimators

This section very closely follows [Rieder, 1994, Ch. 1], including most of the notation. Throughout this section, let F be a distribution on $(\mathbb{R}^p, \mathbb{B}^p)$ and let $X_1, ..., X_n \overset{i.id.}{\sim} F$. For some $\Theta \subset \mathbb{R}^p$ (p finite), denote by $\mathcal{C}^p(\Theta)$ the space of all continuous $\mathbb{R}^p-$valued functions on Θ w.r.t. the supremum norm.

Remark 4.3.1. *We carefully note that Rieder used a different notation, by denoting the space of all continuous and bounded $\mathbb{R}^p-$valued functions by $\mathcal{C}^p(\Theta)$. In light of assumption (A1) below which already bounds the space Θ, it suffices to assume that the function η below is continuous in our context, immediately providing boundedness.*

Assumption 4.3.1. *Respecting the statements of subsection 3.2, a general assumption throughout this section is*
(A0) *The parametric model $\mathcal{P} = \{P_\theta \mid \theta \in \Theta\}$ is L_2-differentiable and if ψ_θ is an influence curve, it belongs to the set Ψ_2.*

With the definitions A.7.1, A.7.2, A.7.3, we are ready to list the assumptions under which Rieder proved the asymptotic linearity of M-estimators. First, define the function

$$\eta : \Theta \to \mathbb{R}^p, \quad \eta(\theta) := \int \varphi(x,\theta) dF(x) = \mathbb{E}_F[\varphi(X,\theta)].$$

Furthermore, define its empirical counterpart

$$Z_n(x_1, ..., x_n, \theta) := \frac{1}{n}\sum_{i=1}^n \varphi(x_i, \theta) = \int \varphi(x,\theta) d\hat{F}_n(x_1, ..., x_n) \tag{4.3.1}$$

with the empirical cumulative distribution function $\hat{F}_n$.

Assumption 4.3.2. *Two assumptions throughout this section are*
(A1) *The parameter space $\Theta \subset \mathbb{R}^p$ is nonempty, compact and equals the topological closure of its interior.*
(A2) *The function φ satisfies*

$$\varphi(x, \cdot) \in \mathcal{C}^p(\Theta) \ F(dx) - a.e., \quad \varphi_\theta := \varphi(\cdot, \theta) \in L_2^p(F) \ \forall \theta \in \Theta.$$

The following theorem was proven by Jain and Marcus ([Jain and Marcus, 1975, Thm. 1]). We use the notation of [Rieder, 1994, Prop. 1.4.3].

Theorem 4.3.1. *Under (A2), assume that:*
(A3) *There exists a pseudo-distance d on Θ such that $d(\theta, \theta_0) \to 0$ as θ converges to θ_0 and such that the metric integral*

$$\int_0^1 \sqrt{H(\epsilon)} d\epsilon$$

is finite.
(A4) *There exists $M \in L_2(F)$ such that*

$$|\varphi(x,\zeta) - \varphi(x,\theta)| \leq d(\zeta, \theta) M(x) \ \forall \zeta, \theta \in \Theta$$

$F(dx)$−a.e..

Then there exists a Gaussian process Z on $\mathcal{C}^p(\Theta)$ such that

$$\mathbb{E}[Z(\theta)] = 0, \quad \mathbb{E}[Z(\zeta) Z^T(\theta)] = \underset{F}{\mathrm{Cov}}(\varphi_\zeta, \varphi_\theta) \ \forall \zeta, \theta \in \Theta$$

and such that

$$\sqrt{n}(Z_n - \eta) \circ F^n \xrightarrow{w} \mathcal{L}(Z)$$

in $\mathcal{C}^p(\Theta)$. Moreover, the sequence $\sqrt{n}(Z_n - \eta) \circ F^n$ is tight in $\mathcal{C}^p(\Theta)$.

The main theorem that we borrow from [Rieder, 1994, Thm. 1.4.2] proves compact differentiability of M-functionals under certain conditions.

Theorem 4.3.2 (Compact differentiability of M-estimators). *Under (A1), (A2), assume additionally that:*
(A5) *There exists a zero $\theta_0 \in \Theta^\circ$ of η and $\eta \in \mathcal{C}^p(\Theta)$. Moreover, η is locally homeomorphic at θ_0 with bounded and invertible derivative $d\eta(\theta_0)$.*

Then there exists a neighborhood $V \subset \mathcal{C}^p(\Theta)$ of η and a functional $T : V \to \Theta$ satisfying

$$f(T(f)) = 0 \ \forall f \in V.$$

T is compactly differentiable at η with derivative given by

$$d_H T(\eta) = -(d\eta(\theta_0))^{-1} \circ \Pi_{\theta_0},$$

where Π_{θ_0} is the evaluation functional at θ_0.

This result is proven in [Rieder, 1994, p. 12 f.] using an implicit function theorem and the Arzéla-Ascoli theorem.

Combining theorems 4.3.1 and 4.3.2 and all the assumptions, Rieder showed the following corollary (cf. [Rieder, 1994, Cor. 1.4.5]).

Corollary 4.3.1. *Under the assumptions (A0)-(A5), the sequence $(S_n)_n := (T \circ Z_n)_n$ of M-estimators has the asymptotic linear expansion*

$$\sqrt{n}(S_n - \theta_0) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \psi_\theta(x_i) + o_{(F^n)_*}(n^0)$$

where the influence function is given by

$$\psi_\theta(x) := -(d\eta(\theta_0))^{-1} \varphi(x, \theta_0).$$

If the S_n are measurable, then asymptotic normality follows, i.e.,

$$\sqrt{n}(S_n - \theta_0) \circ F^n \xrightarrow{w} \mathcal{N}(0, ACA^T)$$

where

$$A := (d\eta(\theta_0))^{-1}, \quad C := \mathbb{E}_F[\varphi_{\theta_0} \varphi_{\theta_0}^T].$$

Remark 4.3.2. *Rieder introduced inner probabilities to be safe against non-measurable functions Z_n since these probabilities always exist (see remark A.7.1). Indeed, prominent non-measurable maps include the empirical cumulative distribution function and the empirical process (see Van der Vaart and Wellner [2013]).*

Remark 4.3.3 (Fréchet differentiability). *The proof uses an infinite-dimensional version of the delta method (see e.g. Van der Vaart and Wellner [2013], [Rieder, 1994, Thm. 1.3.3]) that requires the chain rule. By theorem 3.1.1, the functionals have to be at least compactly differentiable. Since the chain rule holds for Fréchet differentiable maps, one may ask if the gap between compactly and Fréchet differentiable statistical functionals is considerable. The following two examples provide an answer.*

Example 4.3.1. *We refer to [Rieder, 1994, Thm. 1.5.1] who shows that for distribution functions F that are continuous in some neighborhood U around $a = F^{-1}(\alpha)$, the location quantile is compactly but not boundedly differentiable along $\mathcal{C}(U) \cap \mathbb{D}(\mathbb{R})$, provided that $f(a) > 0$, where $\mathbb{D}(\mathbb{R})$ denotes the Skorohod space, i.e., the space of all real-valued càdlàg functions.*

Example 4.3.2. *Another example is given by the functional $T(F, G) := \int F dG$ for distribution functions F, G. It is shown that this functional is compactly differentiable with Hadamard-derivative*

$$d_H T(x, y) = \int x dG - \int y dF,$$

and the empirical version corresponding to the Wilcoxon statistic is compactly differentiable as well (cf. Gill et al. [1989], Van der Vaart and Wellner [2013]). This fact has been used to prove asymptotic linearity of the area under the curve (AUC) and the cross-validated AUC as it has been done in LeDell et al. [2015]. However, Wellner [1992] showed that T is not Fréchet differentiable if one considers the $|| \cdot ||_\infty-$norm. The given counterexample relies on the fact that in the case of Fréchet differentiability, the derivative $d_F T$ coincides with the Hadamard derivative $d_H T$, so $d_H T$ would be the only candidate for $d_F T$, but $d_H T$ does not supply the $o-$term in the first-order expansion in every case.

For completeness, we refer to Dudley [1992] who showed that Fréchet differentiability of the functional in the latter example is true if the distribution functions provide special bounded variation properties and if the integral is replaced by a Young integral. Note also that loss functions like the $\epsilon-$insensitive loss or the check loss (in quantile regression) are not Fréchet differentiable (cf. Christmann and Van Messem [2008]).

4.4 Regularized M-functionals

Fitting a model based on a training set by minimizing some loss function without any restriction generally leads to overfitting, especially in the case of high-dimensional data. This issue has been investigated by Vapnik (Vapnik [1998]) who introduced the structural risk minimization principle which performs an optimization on structures that have finite Vapnik-Chervonenkis dimension. In practice, this idea manifests itself when penalizing the loss function by a regularization term.

In the regression context, we have a model

$$Y = f(X) + \epsilon$$

as introduced in equation (2.1.1).

Assumption 4.4.1. *Assume that the pair (X, Y) follows some distribution F on $(\mathcal{X} \times \mathcal{Y}, \mathbb{B}(\mathcal{X} \times \mathcal{Y}))$, hence we have a random design of X and Y.*

Define the (essentially complete) parametric function class

$$\mathcal{F}_\theta := \{f_\theta : \mathcal{X} \to \mathcal{Y} \mid \theta \in \Theta\}.$$

When facing such a function class, we can recover the true map f_θ by estimating the parameter θ. This is done by defining a loss function $L : (\mathcal{Y} \times \mathcal{Y}) \times \Theta \to [0, \infty[$.

Assumption 4.4.2. *To be notionally consistent with the previous section, we may write*

$$L((x, y), \theta) := L(y, f_\theta(x))$$

in some situations in this chapter. For practical applications, we will again assume that $L((x, y), \theta) = 0$ if $f_\theta(x) = y$ as it was done for example in Christmann et al. [2009].

Assumption 4.4.3. *The penalty term $J_\lambda : \Theta \to [0, \infty[$ that should enforce sparseness of the solution has to satisfy the following conditions:*
(A6) *J_λ is non-negative with $J_\lambda(0_p) = 0$ and J_λ is convex.*

The assumption that a regularization term must be non-negative is natural. On the other hand, since it penalizes the model complexity, the assumption that $J_\lambda(0_p) = 0$ is reasonable since the parameter $\theta = 0_p$ leads to a model consisting at most of the intercept which would not make sense to penalize. The convexity assumption is needed for practical applications to prevent the solution from overfitting and, of course, to guarantee the existence of a unique solution in combination with a convex loss function.

Given this notation, we can write the problem of minimizing the risk w.r.t. θ as

$$R(\theta) := \mathbb{E}_F[L((X, Y), \theta)] + J_\lambda(\theta) = \int_{\mathcal{X} \times \mathcal{Y}} L((x, y), \theta) dF(x, y) + J_\lambda(\theta) = \min_{\theta \in \Theta}!.$$

Since in practice we do not know the true distribution F, but the observed data lead to an empirical distribution $\hat{F}_n$, one minimizes the empirical counterpart of the risk, i.e.,

$$R_n(\theta) := \int_{\mathcal{X}\times\mathcal{Y}} L((x,y),\theta)d\hat{F}_n(x,y) + J_\lambda(\theta) = \frac{1}{n}\sum_{i=1}^n L((X_i,Y_i),\theta) + J_\lambda(\theta)$$

w.r.t. θ. The corresponding Z-equation is, as stated above,

$$Z_n^\lambda(\theta) := \frac{1}{n}\sum_{i=1}^n \varphi((X_i,Y_i),\theta) + J_\lambda'(\theta) := \varphi^\lambda((X_i,Y_i),\theta) \overset{!}{=} 0$$

where $\varphi = \partial_\theta L$ is the score function, thus the first term of Z_n^λ is the empirical counterpart of the expected score and Z_n^λ itself is the empirical counterpart of

$$\eta^\lambda(\theta) := \int_{\mathcal{X}\times\mathcal{Y}} \varphi((x,y),\theta)dF(x,y) + J_\lambda'(\theta).$$

If the penalty term is not of a particular interest, we suppress the superscript and just write Z_n or η, running into the framework of the previous section.

4.4.1 L_2-differentiability of linear regression models

One may ask which requirements our general assumption of L_2-differentiability of the underlying model (A0) actually implies in the regression context. The derivation of sufficient conditions for L_2-differentiability of linear regression models (cf. [Rieder, 1994, Ch. 2.4]) as the definition of L_2-differentiability itself depends on the underlying design of the regressor matrix X.

If one assumes a stochastic or random design of X, we have the model

$$Y_i = X_i\beta + \epsilon_i \tag{4.4.1}$$

where the rows X_i of the regressor matrix are i.id. realizations of the $\mathcal{X}-$valued distribution F_X on $\mathbb{B}^p$. Independently from the regressor variables, the error variables $\epsilon_1, ..., \epsilon_n$ are i.id. distributed and follow some $\mathbb{R}-$valued absolute continuous distribution F_ϵ with density f_ϵ w.r.t. some dominating measure μ. Then Rieder defines the parametric linear regression model family

$$\{P_\beta(dx,dy) = F_\epsilon(dy - x^T\beta)F_X(dx) = f_\epsilon(y - x^T\beta)\mu(dy)F_X(dx) \mid \beta \in \Theta\} \tag{4.4.2}$$

for some parameter space $\Theta \in \mathbb{R}^p$.

Introducing the score function and the Fisher information of F,

$$\Lambda_{f_\epsilon} = -\frac{f'_\epsilon}{f_\epsilon}, \quad I_{f_\epsilon} = \mathbb{E}_{F_\epsilon}[\Lambda^2_{f_\epsilon}],$$

respectively, the L_2-differentiability of the family (4.4.2) of linear regression models is guaranteed by the following conditions (cf. [Rieder, 1994, Thm. 2.4.7]).

Theorem 4.4.1. *The family of linear regression models (4.4.2) is L_2-differentiable at every $\beta \in \Theta$ with L_2-derivative*

$$\Lambda_\beta(x, y) = \Lambda_{f_\epsilon}(y - x^T\beta)x$$

and Fisher information

$$I_\beta = I_{f_\epsilon} \int_{\mathcal{X}} xx^T F_X(dx)$$

if $I_{f_\epsilon} < \infty$ and if

$$\operatorname{rk}\left(\int xx^T F_X(dx)\right) = p.$$

In the case of a fixed or deterministic design of the regressor matrix, one can no longer treat the rows as identically distributed (see [Pupashenko et al., 2015, Sec. 2.2]), so one essentially needs some Lindeberg condition. Therefore, the model is

$$Y_{n,i} = X_{n,i}\beta + \epsilon_{n,i} \tag{4.4.3}$$

for $n \geq 1$, $i = 1, ..., i_n$, $X_{n,i} \in \mathbb{R}^p$ and $\epsilon_{n,1}, ..., \epsilon_{n,i_n} \overset{i.id.}{\sim} F_\epsilon$ for each n. Similarly, the chain rule leads to the score function

$$\Lambda_{n,i,\beta}(y) = \Lambda_{f_\epsilon}(y - x^T_{n,i}\beta)x_{n,i}$$

and the Fisher information

$$I_{n,\beta} := \sum_{i=1}^{i_n} \mathbb{E}_{n,i,\beta}[\Lambda_{n,i,\beta}\Lambda^T_{n,i,\beta}] = I_{f_\epsilon} X_n^T X_n$$

with regressor matrix X_n at time n. The parametric regression model is then

$$\{P_{n,i,\beta}(dy) = f_\epsilon(y - x_{n,i}\beta)\mu(dy) \mid \beta \in \Theta\}. \tag{4.4.4}$$

Rieder shows (see [Rieder, 1994, Thm. 2.4.2]) that if the condition $I_{f_\epsilon} < \infty$ stays the same and if the rank condition is replaced by $\operatorname{rk}(X_n) = p$ for any n, a necessary and sufficient condition for L_2-differentiability of the linear regression model is uniformly smallness of the hat matrix, i.e.,

$$\lim_n(\max_i((X_n(X_n^T X_n)^{-1} X_n^T)_{i,i})) = 0.$$

In this case, one needs L_2-differentiability of parametric arrays (see definition A.7.5).

4.5 Asymptotic linearity of regularized M-estimators

This section provides our main theorems concerning the asymptotic linear expansion of regularized M-estimators.

4.5.1 Compactness assumption of the parameter space

Assuming compactness of the parameter space is in general problematic if we only minimize the loss function.

Example 4.5.1. *Assume that we have the squared loss, i.e.,* $L((x, y), \theta) = (y - x\theta)^2$*. Then the loss is zero if* $y = x\theta$ *holds. But under this restriction, the loss is always zero, independently of the concrete value of* θ*, even if* $||\theta|| \to \infty$*. Thus, we cannot ensure that the minimizer of the loss function is contained in any compact.*

Taking a look at this example, we may conclude that this loss function does not provide a coercivity property (cf. definition A.1.1 and lemma A.1.1). In the presence of a suitable regularization term, this issue does not appear.

Lemma 4.5.1. *Let* $\mathcal{X}$*,* $\mathcal{Y}$*,* Θ *be real vector spaces. Let* $L : \mathcal{X} \times \mathcal{Y} \times \Theta \to [0, \infty[$ *be a continuous loss function and let* $J_\lambda : \Theta \to [0, \infty[$ *be a convex penalty function where* $J_\lambda \not\equiv 0$*. Let* F *be a distribution on* $\mathbb{B}(\mathcal{X} \times \mathcal{Y})$*. Then the risk function* $\mathbb{E}_F[L((X, Y), \theta)] + J_\lambda(\theta)$ *is coercive w.r.t.* θ*, so the parameter space can be restricted to a compact.*

Proof. *By convexity, the penalty terms always must satisfy* $\lim_{||\theta|| \to \infty}(J_\lambda(\theta)) = \infty$ *, otherwise it would have to be constantly zero which we excluded by assumption. The coercivity is inherited from the penalty term since the loss function is convex and by linearity of the integral, its expectation is as well, so the risk is coercive w.r.t.* θ *(lemma A.1.1). In fact, we get* $R(\theta) \to \infty$ *for* $||\theta|| \to \infty$*, so we are allowed to restrict the parameter space to a compact due to lemma A.1.1.*

□

Remark 4.5.1. *Note that the lemma just guarantees the existence of at least a minimum in a compact. The assumptions of the lemma however do not provide uniqueness of the minimizer* *which would require the loss function to be strictly convex (see e.g. Peypouquet [2015]) or the loss function to be convex and the penalty to be strictly convex since the sum of a convex and a strictly convex function is strictly convex.*

This reasoning is of course not new and has been already done to show the existence of solutions for the Huberized Lasso (Lambert-Lacroix et al. [2011]), for regularized kernel methods in Vito et al. [2004] or in De los Reyes et al. [2016] for regularized functionals in the context of image restoration. Analogously, Le et al. [2017] use the non-negativity of a family of penalty terms J_λ^n to show that the corresponding family $\{L + J_\lambda^n\}$ is equi-coercive (definition A.1.2) if the loss L is already coercive and lower semi-continuous.

Remark 4.5.2. *It is easy to see that the usual penalty terms like the l_1-, l_2- or elastic net penalty are coercive (see p.e. [Aravkin et al., 2013, Cor. 11]). On the other hand, non-convex penalties do not have to be coercive, for example the SCAD (smoothly clipped absolute deviation) penalty (cf. Fan and Li [2001]) is constant outside a neighborhood of zero whose width depends on the penalty parameter.*

In fact, since we are now allowed to assume compactness of the parameter space, we face another potential issue. The compactness assumption leads to the problem that the M-estimator $\hat{\theta}_n$ may be located at the boundary of Θ. We invoke the idea of general One-Step estimators from Van der Vaart [2000] to resolve this problem.

Having a $\sqrt{n}-$consistent preliminary solution $\tilde{\theta}_n$ of the estimating equation $Z_n(\theta) = 0$, then an application of the Newton-Raphson algorithm leads to an improved One-Step solution

$$\hat{\theta}_n := \tilde{\theta}_n - (Z'_{n,0}(\tilde{\theta}_n))^{-1} Z_n(\tilde{\theta}_n)$$

where $Z'_{n,0}$ is regular and converges in probability to a regular matrix Z'_0. The following theorem can be found in [Van der Vaart, 2000, Thm. 5.45].

Theorem 4.5.1. *Let the notation be as above. Let the condition that for every constant M it holds that*

$$\sup_{\sqrt{n}||\theta-\theta_0||<M} \left(||\sqrt{n}(Z_n(\theta) - Z_n(\theta_0)) - Z'_0\sqrt{n}(\theta - \theta_0)|| \right) \xrightarrow{P} 0 \tag{4.5.1}$$

be satisfied for a regular matrix Z'_0. If it holds additionally that $\sqrt{n}(Z_n(\theta_0))$ converges to some limit, then the One-Step estimator $\hat{\theta}_n$ is already $\sqrt{n}-$consistent.

Using this theorem, we can state the following result.

Lemma 4.5.2. *Let all the notation be as above. Under (A2), (A5) and the additional assumptions*
(A7) *The learning procedure is $\sqrt{n}-$consistent,*
(A8a) *The function Z_n is twice differentiable w.r.t. θ,*

the One-Step estimator is not located at the boundary of the parameter space.

Proof. *Since condition (4.5.1) is weaker than differentiability of Z_n at θ, this part is already satisfied by (A8a).*

The only condition of theorem 4.5.1 that remains to be proven is the convergence of $\sqrt{n}(Z_n(\theta_0))$ to some limit Z. But we already know by (A7) that the learning algorithm is $\sqrt{n}-$consistent, so $\hat{\theta}_n - \theta_0 = o_F(n^{-1/2})$, hence we get

$$\sqrt{n}(\theta_0 - \hat{\theta}_n) = o_F(n^0).$$

An application of a delta method (A.7.1) which is possible under (A8a) provides that

$$\sqrt{n}(Z_n(\theta_0) - Z_n(\hat{\theta}_n))$$

has a limiting distribution (which is the Dirac measure at zero) and we note that by definition, it holds that $Z_n(\hat{\theta}_n) = 0$, so the convergence of $\sqrt{n}Z_n(\theta_0)$ has been established and theorem 4.5.1 applies.

□

Remark 4.5.3. *We admit that it is not common to assume learning rates like we did in assumption (A7). Since functions that are too complex may not be able to be approximated with a pre-determined rate, this assumption results in the class of approximable functions getting smaller.*

Regarding carefully the condition for consistency of the One-Step estimator, (A8a) may seem a bit too strict since the differentiability of the whole function is not needed. On the other hand, the score function φ has to exist and has to be square-integrable due to (A2), so the additional regularity condition of twice differentiability is justifiable since the loss functions that are used in machine learning are in general either non-differentiable or smooth. In addition, the same argumentation holds for the regularization term and we will see that for non-differentiable penalty terms, approximation methods can be applied.

Remark 4.5.4. *We note that non-differentiable loss functions have already been studied in Christmann and Van Messem [2008]. It is worthy to mention that Hable [2012] derived* *results on asymptotic normality of kernel-based regression methods by first showing that the corresponding functional is Gâteaux differentiable and that the derivative is continuous which is a sufficient condition for Fréchet differentiability (see remark 3.1.1), hence also for compact differentiability. Then he applied an infinite-dimensional Delta method to prove asymptotic normality.*

Since the One-Step estimation guarantees that the estimator is not located at the boundary of the parameter space, assuming compactness of Θ is fine in the presence of the assumptions (A2), (A5), (A7) and (A8a). Note that disregarding this argumentation, (A7) and (A8a) are not absolutely necessary for the results in the next subsections.

Remark 4.5.5. *We note that in the case of convex risk functions, one maybe could use Alexandroff's theorem (cf. Rockafellar [1999]) that provides almost everywhere a second-order expansion of a convex function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ to ensure condition (4.5.1). However, we will not pursue this idea any further in this work.*

4.5.2 Twice differentiable Z-function

In this subsection, we assume twice differentiability of the target function Z_n^λ which leads to the following result.

Theorem 4.5.2. *Under the conditions*
(A0) *The parametric model $\mathcal{P} = \{P_\theta \mid \theta \in \Theta\}$ is L_2-differentiable and if ψ_θ is an influence curve, it belongs to the set Ψ_2,*
(A1) *The parameter space $\Theta \subset \mathbb{R}^p$ is nonempty, compact and equals the topological closure of its interior,*
(A2') *$\varphi^\lambda(\cdot, \theta) \in L_2^p(F)\ \forall \theta \in \Theta$,*
(A5') *$\eta^\lambda(\theta_0) = 0$ for a $\theta_0 \in \Theta^\circ$ and η^λ is locally homeomorphic at θ_0 with bounded and invertible derivative $d\eta^\lambda(\theta_0)$,*
(A6) *J_λ is non-negative with $J_\lambda(0_p) = 0$ and J_λ is convex.*
(A7) *The learning procedure is $\sqrt{n}-$consistent,*
(A8a) *L is convex and Z_n^λ is twice differentiable,*

the sequence $(S_n^\lambda)_n := (T \circ Z_n^\lambda)_n$ of regularized M-estimators is asymptotically linear.

Proof. *As a byproduct of (A1), we immediately get (A3). This is true since $\mathbb{R}^p$ is a normed space, hence the pseudo-distance is just the standard euclidean norm on $\mathbb{R}^p$ and by boundedness of Θ and since p is finite, we can conclude that the metric integral is finite. Even more general, the metric integral is finite provided that $d(\theta, \theta_0) = ||\theta - \theta_0||_2^\delta$ for some $\delta \in]0, \infty[$ (see [Rieder, 1994, Rem. 1.4.6.b)]).*

From twice differentiability of η^λ, the first part of (A2) is trivially satisfied and the derivative φ of L is continuous w.r.t. θ. By (A6), (A7), (A8a) and lemma 4.5.2, the assumption that we can restrict the parameter space Θ to a compact set is justifiable. Using this compactness of Θ, we deduce that the function $\varphi(x, \cdot)$ is Lipschitz-continuous, thus there exists a constant (w.r.t. θ) such that

$$|\varphi(x, \xi) - \varphi(x, \theta)| \leq L_x ||\xi - \theta||$$

where the Lipschitz constant L_x must be finite by compactness of Θ. We conclude that (A4) holds.

Thus, corollary 4.3.1 is applicable and we get the desired result.

$\square$

The influence curve for regularized regression M-estimators with twice differentiable penalty term and twice differentiable loss function has been derived in [Öllerer et al., 2015, Prop. 4.1]. If one can write $L((x, y), \beta) = L(y - x\beta)$, then it is given by

$$\begin{gathered} \mathrm{IC}((x_0, y_0), \beta, F) = \\ (\mathbb{E}_{F_0}[\psi'(y - x\beta)x^T x] + 2\lambda \operatorname{diag}(J''(\beta)))^{-1}(\psi(y_0 - x_0\beta)x_0 - \mathbb{E}_{F_0}[\psi(y - x\beta)x]) \end{gathered} \tag{4.5.2}$$

4.5.3 Twice continuously differentiable loss function, non-differentiable penalty term

If the penalty term is non-differentiable, like the Lasso loss, then we invoke a result of Avella-Medina [2017]. Note that despite we cannot assume differentiability of the penalty term, we can at least assume continuity. For the very basic definitions from distribution theory that we need in this subsection, see section A.6.

The following lemma is a combination of [Avella-Medina, 2017, Lemma 2] and [Avella-Medina, 2017, Prop. 1].

Lemma 4.5.3 (Approximating influence curves). *Assume that the parameter space $\Theta \subset \mathbb{R}^p$ is compact and that the loss function is twice continuously differentiable w.r.t. θ. If there exists a sequence $(J_\lambda^m)_m$ with $J_\lambda^m \in C^\infty(\Theta)$ that converges to J_λ in the Sobolev space $W^{2,2}(\Theta)$, i.e.,*

$$||J_\lambda^m - J_\lambda||_{W^{2,2}} = \left(\sum_{|\alpha| \leq 2} \int_\Theta |\partial^\alpha (J_\lambda^m(\theta) - J_\lambda(\theta))|^2 d\theta \right)^{1/2} \longrightarrow 0,$$

then

$$\lim_m (T_m) = T$$

where T_m denotes the M-functional that intends to find the zero of the Z-equation corresponding to R_m where R_m denotes the risk function where J_λ is replaced by J_λ^m. This does not depend on the particular choice of the approximating sequence J_λ^m.

The proof of lemma 4.5.3 relies on a result presented in [Berge, 1963, Ch. VI] which was referred to as "Berge's maximum theorem", theorem A.6.1. The main idea is essentially to define a set of parameters θ in dependence of the penalty term J_λ and to show that the map that translates J_λ into the set of optimal parameters that lead to minimal risk is upper semi-continuous. Then, for a convergent sequence of penalty terms as in lemma 4.5.3, auxiliary lemmas from [Berge, 1963, Ch. VI] guarantee convergence of these optimal sets to the optimal set for J_λ.

This has a very nice consequence for our task.

Theorem 4.5.3. *Assume that there exists a sequence $(J_\lambda^m)_m$ with $J_\lambda^m \in \mathcal{C}^\infty(\Omega)$ of regularization functions that converge to J_λ in the Sobolev space $W^{2,2}(\Theta)$. Under the conditions*
(A0) *The parametric model $\mathcal{P} = \{P_\theta \mid \theta \in \Theta\}$ is L_2-differentiable and if ψ_θ is an influence curve, it belongs to the set Ψ_2,*
(A1) *The parameter space $\Theta \subset \mathbb{R}^p$ is nonempty, compact and equals the topological closure of its interior,*
(A2') *$\varphi^\lambda(\cdot, \theta) \in L_2^p(F)$ $\forall \theta \in \Theta$,*
(A5') *$\eta^\lambda(\theta_0) = 0$ for a $\theta_0 \in \Theta^\circ$ and η^λ is locally homeomorphic at θ_0 with bounded and invertible derivative $d\eta^\lambda(\theta_0)$,*
(A6) *J_λ is non-negative with $J_\lambda(0_p) = 0$ and J_λ is convex.*
(A7) *The learning procedure is $\sqrt{n}-$consistent,*
(A8b) *The loss function L is convex and twice continuously differentiable w.r.t. θ,*

the sequence $(S_n^\lambda)_n$ of regularized M-estimators is asymptotically linear.

Proof. *From theorem 4.5.2, we can conclude that the estimator has an asymptotic linear expansion and that it is asymptotically normal if the respective assumptions collected there are satisfied. But since this is just an asymptotic property up to a remainder term of order $n^{-1/2}$, it suffices to approximate J_λ by J_λ^m such that the difference in the respective influence functions is negligible, i.e., the aggregated difference is already of order $n^{-1/2}$. Note that by continuity of the Gâteaux derivative w.r.t. the direction and by the abovely stated lemma, it holds that $\lim_m(\mathrm{IC}(x, T_m, P)) = \mathrm{IC}(x, T, P)$.*

Finally, we can conclude that we can work with infinitely differentiable penalty terms satisfying the conditions of the previous subsection but that this results in the same asymptotic linear expansion as if we used the true non-differentiable penalty term. Thus, we only need the existence of an approximating sequence of penalty terms.

□

Remark 4.5.6. *[Öllerer et al., 2015, Lemma 5.4] showed for the Lasso and a concrete hyperbolic tangent approximation of the penalty term that the influence function of the approximating estimator derived by [Öllerer et al., 2015, Prop. 4.1] converges to the influence function of the Lasso. So, Avella-Medina generalized their result with 4.5.3 for any losses and penalties satisfying the given conditions.*

4.5.4 Extension to ranking

In the ranking setting (see part II of this thesis), we assume the same underlying model as in the first part of this section, with the only difference that we have to invoke a joint distribution $F_r : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y})$ on the measurable space $((\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}), \mathbb{B}((\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y})))$ where the notation F_r is introduced to distinguish it from the joint distribution in the previous part of this section.

In contrast to prediction problems, it is not the goal to recover the true values of the Y_i but just to predict their true order. Therefore, the ranking model can be fitted by defining a ranking loss function $L^r : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \times \Theta \to [0, \infty[$ which quantifies a ranking loss, that is some loss of a misranking of a pair of instances. Defining a penalty term and the ranking risk analogously to the risk R in section 4.4, the corresponding Z-equation resulting from the problem to minimize the regularized risk is

$$Z_n^{r,\lambda}(\theta) := \frac{1}{n(n-1)} \sum\sum_{i \neq j} \varphi^r(((X_i, Y_i), (X_j, Y_j)), \theta) + J_\lambda'(\theta) \overset{!}{=} 0$$

where $\varphi^r = \partial_\theta L^r$ is the score function, hence the first term of $Z_n^{r,\lambda}$ is the empirical counterpart of the expected score.

We can easily adapt the theorem of Rieder [1994] to the ranking setting and conclude compact differentiability of regularized ranking functionals and asymptotic linearity of the sequence $(S_n^{r,\lambda})_n := (T \circ Z_n^{r,\lambda})_n$ of regularized ranking M-estimators.

Theorem 4.5.4. *Define*

$$\eta^{r,\lambda} : \Theta \to \mathbb{R}^p, \quad \eta^{r,\lambda}(\theta) := \int \varphi^r(((x,y),(x',y')),\theta)dF^r(((x,y),(x',y'))) + J'_\lambda(\theta).$$

Then, under the conditions

(A0) *The parametric model $\mathcal{P} = \{P_\theta \mid \theta \in \Theta\}$ is L_2-differentiable and if ψ_θ is an influence curve, it belongs to the set Ψ_2,*
(A1) *The parameter space $\Theta \subset \mathbb{R}^p$ is nonempty, compact and equals the topological closure of its interior,*
((A2r)') *$\varphi^r(\cdot,\theta) + J'_\lambda(\theta) \in L_2^p(F^r)\ \forall \theta \in \Theta$,*
(A5) *There exists a zero $\theta_0 \in \Theta^\circ$ of $\eta^{r,\lambda}$ and $\eta^{r,\lambda} \in \mathcal{C}^p(\Theta)$. Moreover, $\eta^{r,\lambda}$ is locally homeomorphic at θ_0 with bounded and invertible derivative $d\eta^{r,\lambda}(\theta_0)$,*
(A6) *J_λ is non-negative with $J_\lambda(0_p) = 0$ and J_λ is convex.*
(A7) *The learning procedure is $\sqrt{n}-$consistent,*
(A8a) *L^r is convex and $Z_n^{r,\lambda}$ is twice differentiable,*

there exists a neighborhood $V \subset \mathcal{C}^p(\Theta)$ of $\eta^{r,\lambda}$ and a functional $T : V \to \Theta$ satisfying

$$f(T(f)) = 0\ \forall f \in V.$$

T is compactly differentiable at $\eta^{r,\lambda}$ with derivative given by

$$d_H T(\eta^{r,\lambda}) = -(d\eta^{r,\lambda}(\theta_0))^{-1} \circ \Pi_{\theta_0},$$

where Π_{θ_0} is the evaluation functional at θ_0.

Moreover, the sequence $(S_n^{r,\lambda})_n := (T \circ Z_n^{r,\lambda})_n$ of regularized ranking M-estimators is asymptotically linear.

Proof. *This directly follows from corollary 4.3.1 since the optimization is done w.r.t. θ whereas the dimension of the space of the other arguments of φ is not explicitly used in the proof.*

□

In the case of non-differentiable loss functions, we would once more make use of the smooth approximation of the penalty term.

Theorem 4.5.5. *Assume that there exists a sequence $(J_\lambda^m)_m$ with $J_\lambda^m \in \mathcal{C}^\infty(\Omega)$ of regularization functions that converge to J_λ in the Sobolev space $W^{2,2}(\Theta)$. Then, under the conditions*

(A0) *The parametric model $\mathcal{P} = \{P_\theta \mid \theta \in \Theta\}$ is L_2-differentiable and if ψ_θ is an influence curve, it belongs to the set Ψ_2,*

(A1) *The parameter space $\Theta \subset \mathbb{R}^p$ is nonempty, compact and equals the topological closure of its interior,*

((A2r)') *$\varphi^\lambda(\cdot, \theta) + J_\lambda'(\theta) \in L_2^p(F^r)\ \forall \theta \in \Theta$,*

(A5') *$\eta^{r,\lambda}(\theta_0) = 0$ for a $\theta_0 \in \Theta^\circ$ and $\eta^{r,\lambda}$ is locally homeomorphic at θ_0 with bounded and invertible derivative $d\eta^{r,\lambda}(\theta_0)$,*

(A6) *J_λ is non-negative with $J_\lambda(0_p) = 0$ and J_λ is convex.*

(A7) *The learning procedure is $\sqrt{n}-$consistent,*

(A8b) *The loss function L^r is convex and twice continuously differentiable w.r.t. θ,*

there exists a neighborhood $V \subset \mathcal{C}^p(\Theta)$ of $\eta^{r,\lambda}$ and a functional $T : V \to \Theta$ satisfying

$$f(T(f)) = 0\ \forall f \in V.$$

T is compactly differentiable at $\eta^{r,\lambda}$ with derivative given by

$$d_H T(\eta^{r,\lambda}) = -(d\eta^{r,\lambda}(\theta_0))^{-1} \circ \Pi_{\theta_0},$$

where Π_{θ_0} is the evaluation functional at θ_0.

Moreover, the sequence $(S_n^{r,\lambda})_n := (T \circ Z_n^{r,\lambda})_n$ of regularized ranking M-estimators is asymptotically linear.

Remark 4.5.7. *It is important to emphasize that ranking loss functions like the hard ranking loss (see (5.2.3)) and other losses that we introduce in part II are not continuous and not convex, so they fail the assumptions of these theorems. However, the hard ranking loss is bounded, so combining it with a suitable regularity term again leads to a coercive target function.*

Examples for which the regularity conditions hold are smooth convex surrogates of those ranking losses that we will also discuss later.

4.6 Examples for asymptotically linear estimators in machine learning

The conditions for asymptotic linearity of the regularized M-estimators in the previous section are quite general. The goal of this section is to provide examples for machine learning algorithms to which the derived results can be applied and to specify the required conditions for each of them.

4.6.1 Lasso

Lasso regression (cf. Bühlmann and Van De Geer [2011]) is an l_1-penalized least squares regression, i.e.,

$$\hat{\beta}^{lasso} = \underset{\beta \in \Theta}{\operatorname{argmin}} \Big(\frac{1}{n}||Y - X\beta||_2^2 + \lambda||\beta||_1\Big).$$

The Lasso regression results in a shrinkage of the coefficients and in a sparse fitted model. The score function for the unregularized loss is given by

$$\varphi(\cdot, \beta) = \frac{2}{n} X^T (Y - X\beta). \tag{4.6.1}$$

We invoke the approximation of the non-differentiable penalty term. There exists an example of such a smooth penalty term converging to the absolute value in Avella-Medina [2017] and another one in Öllerer et al. [2015].

Theorem 4.6.1. *Assume that*
(A0) *The parametric model $\mathcal{P} = \{P_\theta \mid \theta \in \Theta\}$ is L_2-differentiable and if ψ_θ is an influence curve, it belongs to the set Ψ_2,*
(A1) *The parameter space $\Theta \subset \mathbb{R}^p$ is nonempty, compact and equals the topological closure of its interior,*
((A2^{Lasso})') *The ideal distribution F has finite fourth moments,*
(A5') *The true solution β^0 lies in the interior of Θ and the derivative $d\eta^\lambda(\beta^0)$ is invertible,*
(A7^{Lasso}) *$||\beta^0||_1 = o(\sqrt{n/\ln(p)})$ and that the regularization parameter in dependence of n is chosen in the range of $\lambda_n = \sqrt{\ln(p)/n}$.*

Then the sequence $(S_n^{Lasso})_n := (T \circ Z_n^{Lasso})_n$ of Lasso estimators is asymptotically linear.

Proof. *We need to verify the conditions of theorem 4.5.3. Consider a smooth approximation J_λ^m of the absolute value in the sense of the Sobolev space $W^{2,2}$, as given in Öllerer et al. [2015] or Avella-Medina [2017], respectively. Then we set*

$$\tilde{J}_\lambda^m(x) := \sum_i J_\lambda^m(x_i) \longrightarrow \sum_i |x_i| = ||x||_1,$$

and thus

$$\nabla_x \tilde{J}_\lambda^m(x) = (\partial_{x_1} \tilde{J}_\lambda^m(x), ..., \partial_{x_p} \tilde{J}_\lambda^m(x)) \longrightarrow (\text{sign}(x_1), ..., \text{sign}(x_p)) = \nabla_x ||x||_1$$

and the (component-wise) convergence of the Hessian holds as well due to the properties of $W^{2,2}$. For this idea, we refer to [Öllerer et al., 2015, Lemma 5.4]. The loss function and the approximating penalty term are smooth, hence (A8b) is satisfied and lemma 4.5.3 is applicable.

The target function is coercive w.r.t. β (see definition A.1.1). This holds because as $||\beta|| \to \infty$, the penalty will tend to infinity and so does the target function. Note that this does not hold for the loss function itself since $||\beta|| \to \infty$ can result in a small loss, cf. example 4.5.1. One may argue that even in the penalized case, it can happen that $||(x, y, \beta)|| \to \infty$ without resulting in the target function growing as well. If for example $Y = 0$ and X is large, then $Y = X\beta$ for $\beta = 0_p$. But in this case, we do not lose anything if we restrict the parameter space. Furthermore, we can write the optimization problem in the form

$$\min(||Y - X\beta||_2^2/n) \quad s.t. \quad ||\beta||_1 \leq c_\lambda$$

for some constant c_λ depending on λ. So we have a convex optimization problem with a continuous, strictly convex and coercive target function, so by Werner [2006], there definitely exists a solution β^0 of η^λ and the local homeomorphicity around the solution follows.

Combining ((A2^{Lasso})') with equation (4.6.1), we derive that the score function is square-integrable w.r.t. the distribution F. Then (A2') is satisfied and by boundedness of the integral by the previous assumption and by compactness of the parameter space, this derivative is bounded.

The Lasso is generally inconsistent, but under (A7^{Lasso}), it follows from Bühlmann and Van De Geer [2011] that the Lasso is $\sqrt{n}-$consistent in this case. Note that despite we solve a convex optimization problem assuming that the true solution is already located in the interior of Θ, that does not suffice to guarantee that the computed solution does not lie on the boundary of the parameter space. Finally, theorem 4.5.3 applies and the assertion is proven.

□

Note that a suitable influence function has already been derived in Öllerer et al. [2015] and by the considerations of our approximation theorem, the corresponding asymptotic linear expansion holds for the Lasso.

4.6.2 Elastic net

The elastic net (cf. Zou and Hastie [2005]) can be regarded as a compromise between Lasso and Ridge regression. Given two penalty parameters λ_1, λ_2, the elastic net solution is given by

$$\hat{\beta}^{EN} = \underset{\beta}{\operatorname{argmin}} \left(\frac{1}{n} ||Y - X\beta||_2^2 + \lambda_1 ||\beta_1||_1 + \lambda_2 ||\beta||_2^2 \right)$$

and by defining $\alpha := \frac{\lambda_2}{\lambda_1 + \lambda_2}$, this can be rewritten as a convex combination of l_1- and l_2- penalties, i.e.,

$$\hat{\beta}^{EN} = \underset{\beta}{\operatorname{argmin}} \left(\frac{1}{n} ||Y - X\beta||_2^2 + (1-\alpha) ||\beta_1||_1 + \alpha ||\beta||_2^2 \right)$$

where $(1-\alpha)||\beta||_1 + \alpha||\beta||_2^2$ is referred to as the elastic net penalty.

Corollary 4.6.1. *Under the assumptions of theorem 4.6.1, the sequence* $(S_n^{orthEN})_n := (T \circ Z_n^{orthEN})_n$ *of elastic net estimators with orthonormal design is asymptotically linear.*

Proof. *Note that for orthonormal design, the EN solution is just a rescaled Lasso solution with factor* $\frac{1}{1+\lambda_2}$. *In this case, we can simply rescale the influence function derived in Öllerer et al. [2015], proving the result.*

□

Corollary 4.6.2. *Under the assumptions of theorem 4.6.1, the sequence* $(S_n^{EN})_n := (T \circ Z_n^{EN})_n$ *of elastic net estimators is asymptotically linear.*

Proof. *It is shown in Zou and Hastie [2005] that the elastic net can be rewritten as a special Lasso with the augmented data*

$$X^* := \frac{1}{\sqrt{1+\lambda_2}} \begin{pmatrix} X \\ \sqrt{\lambda_2} I_p \end{pmatrix}, \quad y^* := \begin{pmatrix} y \\ 0_p \end{pmatrix}$$

and the penalty $\gamma := \frac{\lambda_1}{\sqrt{1+\lambda_2}}$. *If* $\hat{\beta}^{Lasso}$ *is the respective Lasso solution, the elastic net solution is a rescaling with factor* $\frac{1}{1+\lambda_2}$ *as before.*

Using these results and the idea of Öllerer et al. [2015], the respective influence curve of the elastic net can been computed by just adapting the already calculated influence curve and theorem 4.6.1.

□

4.6.3 Adaptive Lasso

The adaptive Lasso (cf. Zou [2006]) is a two-stage estimator that first computes a $\sqrt{n}-$consistent starting estimator (or, as suggested in Bühlmann and Van De Geer [2011], the standard Lasso estimator), denoted by $\hat{\beta}^{init}$, and then in a second step, one minimizes

$$\hat{\beta}^{adapt} = \underset{\beta}{\operatorname{argmin}} \left(\frac{1}{n} ||Y - X\beta||_2^2 + \lambda \sum_j \frac{|\beta_j|}{|\hat{\beta}_j^{init}|} \right).$$

Borrowing the consistency requirements for the adaptive Lasso from Zou [2006], we have the following result.

Theorem 4.6.2. *Assume that*
(A0) *The parametric model $\mathcal{P} = \{P_\theta \mid \theta \in \Theta\}$ is L_2-differentiable and if ψ_θ is an influence curve, it belongs to the set Ψ_2,*
(A1) *The parameter space $\Theta \subset \mathbb{R}^p$ is nonempty, compact and equals the topological closure of its interior,*
((A2^{Lasso})') *The ideal distribution F has finite fourth moments,*
(A5') *The true solution β^0 lies in the interior of Θ and the derivative $d\eta^\lambda(\beta^0)$ is invertible,*
(A7^{ALasso}) *The regularization parameter in dependence of n satisfies $\lambda_n = o(\sqrt{n})$ and $\lambda_n n^{(\gamma-1)/2} \to \infty$ for $\gamma > 0$.*

Then the sequence $(S_n^{adapt})_n := (T \circ Z_n^{adapt})_n$ of Adaptive Lasso estimators is asymptotically linear and the influence curve of the $j-th$ component of $\hat{\beta}^{adapt}$ is given by

$$\mathrm{IC}((x_0, y_0), \hat{\beta}_j^{adapt}(\lambda), P_{\theta_0}) = \begin{cases} 0, & \hat{\beta}_j^{init}(\lambda) = 0 \\ 0, & \hat{\beta}_j^{adapt}(\lambda) = 0 \\ \mathrm{IC}((x_0, y_0), \hat{\beta}_j^{Lasso}(\lambda / |\hat{\beta}_j^{init}(\lambda)|)), & otherwise \end{cases}.$$

where we denote by $\hat{\beta}^{Lasso}(\lambda)$ the Lasso estimator using the penalty factor λ.

Proof. *Obviously, if $\hat{\beta}_j^{init} = 0$, we immediately know that $\hat{\beta}_j^{adapt} = 0$. Hence, if we have the initial solution, we can rewrite the adaptive Lasso optimization problem as*

$$\hat{\beta}^{adapt}_{\hat{S}^{init}_\lambda} = \underset{\beta_{\hat{S}^{init}(\lambda)}}{\operatorname{argmin}} \left(\frac{1}{n} \sum_{i=1}^n \sum_{j \in \hat{S}^{init}(\lambda)} (Y_i - X_{ij}^T \beta_j)^2 + \lambda \sum_{j \in \hat{S}^{init}(\lambda)} \frac{|\beta_j|}{|\hat{\beta}_j^{init}|} \right)$$

where $\hat{S}^{init}(\lambda) := \{j \mid \hat{\beta}_j^{init} \neq 0\}$. Then this optimization problem is just a Lasso optimization problem with a weighted penalty term which can be approximated coordinate-wisely in the spirit of Avella-Medina [2017].

The corresponding influence function has implicitly been derived in Avella-Medina [2017]. Note that by our method, we would only derive $|\hat{S}^{init}(\lambda)|$ *components of the influence function. However, it was proven in Öllerer et al. [2015] that the components of the influence function corresponding to the coefficients that are excluded from the model are zero, i.e., if the Lasso in the first step already sets some coefficients to zero, the final coefficients will be zero, so we can just plug in zeroes into the respective components of the influence function, providing the usual asymptotic linear expansion.*

Since the first step does not compute the final non-zero coefficients but just regularizing weights, its influence implicitly arises in this expansion as a factor, leading to the stated result.

□

Remark 4.6.1 (**Partial influence curves**). *Note that the influence curves derived in theorem 4.6.2 correspond to the concept of "partial" influence functions (see definition 3.2.5). This is true since in the proof of theorem 4.6.2, we are implicitly using the smooth transformation* $\beta \mapsto \beta_{\hat{S}_{\lambda}^{init}}$ *to derive the components of the influence curve corresponding to the coefficients that not already have been excluded from the model in the initialization step. In other words (after suitable renumeration of the columns), we get the matrix* $D_{\beta_{\hat{S}^{init}(\lambda)}} := (\mathrm{diag}(1, \hat{s}^{init}), 0_{p-\hat{s}^{init}}) \in \mathbb{R}^{\hat{s}^{init} \times p}$ *where* $\hat{s}^{init} := |\hat{S}^{init}(\lambda)|$. *See also section 3.5.*

Remark 4.6.2 (**Asymptotic normality**). *Note that the additional assumption of measurability of the sequences of estimators provides asymptotic normality of the estimating sequence due to corollary 4.3.1. Of course, we are not the first ones with results on asymptotic normality. See for example [Loh et al., 2017, Sec. 3] where Corollary 1 shows under which conditions regularized M-estimators of a very general form, including the Lasso, are asymptotically normal.*

4.7 Concrete influence curves for the Lasso and the Adaptive Lasso

We already mentioned that the influence curve for the Lasso has been computed in Öllerer et al. [2015]. However, working with them requires the knowledge of the positions of the non-zero coefficients of the Lasso model. We always consider the general case where $p > 1$ and note that the influence curve for the simple Lasso regression model which is very handy has been derived in [Öllerer et al., 2015, Lemma 5.1].
Let

$$Y = X\beta + \epsilon \tag{4.7.1}$$

be a linear regression model and let F be the true joint distribution of (X, Y). Then [Öllerer et al., 2015, Lemma 5.2] and [Öllerer et al., 2015, Prop. 5.3] provide the influence functions stated in the following two lemmas.

Lemma 4.7.1. *Assume that the Lasso is solved using the coordinate descent algorithm from Friedman et al. [2007]. Then the influence function for the $j-th$ coordinate of the Lasso parameter is*

$$\mathrm{IC}((x_0, y_0), \beta, F) =$$
$$\begin{cases} 0, \quad |\mathbb{E}_F[x_j \tilde{y}^{(j)}]| < \lambda \\ \frac{-\mathbb{E}_F[x_j (x^{(j)})^T \,\mathrm{IC}((x_0,y_0),(\beta^*)^{(j)},F)] + (y_0 - (x_0^{(j)})^T (\beta^*)^{(j)}(F))(x_0)_j}{\mathbb{E}_F[x_j^2]} - \frac{\mathbb{E}_F[x_j \tilde{y}_j^{(j)}](x_0)_j^2}{(\mathbb{E}_F[x_j^2])^2} - \\ -\lambda \frac{\mathbb{E}_F[x_j^2] - (x_0)_j^2}{(\mathbb{E}_F[x_j^2])^2} \,\mathrm{sign}(\mathbb{E}_F[x_j \tilde{y}^{(j)}]), \quad \textit{otherwise} \end{cases},$$

where

$$z^{(j)} := (z_1, ..., z_{j-1}, z_{j+1}, ..., z_p), \quad \tilde{y}^{(j)} := y - (x^{(j)})^T (\beta^*)^{(j)}(F)$$

and where $(\beta^*)^{(j)}$ is the functional that takes the value of the last iteration of the algorithm.

Despite the coordinate decent algorithm converges for any starting value (Friedman et al. [2007]), it is not clear how to work with this influence curve. If the position of the non-zero entries is already known, the following influence curve is valid.

Lemma 4.7.2. *Let w.l.o.g. the true coefficients be zero for $j > k$ and nonzero for $1 \leq j \leq k \leq p$. Then*

$$\mathrm{IC}((x_0, y_0), \beta, F) =$$
$$\begin{pmatrix} (\mathbb{E}_F[x_{1:k} x_{1:k}^T])^{-1}((x_0)_{1:k}(y_0 - x_0^T \beta(F)) - \mathbb{E}_F[x_{1:k}(y - x^T \beta(F))]) \\ 0_{p-k} \end{pmatrix}$$

where $x_{1:k}$ is the subvector of x containing only the first k components.

This result is shown by computing first order conditions for the solution where the derivative of the penalty term is derived using subgradient calculus. The proof is similar to the proof of lemma 2.1 in Bühlmann and Van De Geer [2011] with the difference that Alfons et al. used the derivative of the expected loss function instead of the loss function itself. Then the influence curve of [Öllerer et al., 2015, Lemma 5.2] and the convergence properties of the coordinate gradient algorithm lead to the stated result.

One can work with the latter influence curve if model selection is performed before. Similarly, this strategy can be pursued when working with approximating influence curves. Then the idea would be to compute an approximating influence curve using equation (4.5.2) and to set all components of this influence curve to zero whose corresponding coefficients are excluded from the selected model.

The advantage of the approximating influence curves over the one in equation (4.7.2) is that the penalty parameter λ is directly included in the formula, so the computation of the influence curve of the corresponding Adaptive Lasso estimator using theorem 4.6.2 is possible.

4.8 Data-driven penalty parameters

It is common that asymptotic results for regularized methods respect the case that the regularization parameter is data-driven which manifests itself in a sequence $(\lambda_n)_n$ of regularliztion parameters. The same is true for Boosting where the amount of regularization does not depend on a penalty parameter but implicitly on the number of iterations such that a diverging sequence of iterations is the analogue to a sequence $(\lambda_n)_n$ with $\lambda_n \to 0$ for $n \to \infty$.

To keep the asymptotic results valid uniformly for $n \to \infty$, results for Lasso methods as in Bühlmann and Van De Geer [2011] or Zou [2006] and for Boosting methods as for example in Bühlmann [2006] require penalty parameters which fall into a suitable range in dependence of n or numbers of iterations that grow sufficiently slow w.r.t. n.

As for our results, we would need a suitable degree of approximation, i.e., a suitable sequence $(m_n)_n$, leading to a sequence of regularization terms of the form $(J_{\lambda_n}^{m_n})_n$, to get similar statements.

In fact, we already used sequences $(m_n)_n$ implicitly when proving theorem 4.5.3. Our argument was to set m_n sufficiently large to get a degree of approximation that leads to an error term which is already absorbed by $o_P(n^{-1/2})$.

If we concern about sequences of penalty terms, we essentially need to have a sequence $(m_n)_n$ which again grows sufficiently fast to keep the error term small enough. Since we assumed that J_λ is approximable by a sequence of smooth penalty terms J_λ^m and since λ usually just enters as a factor, a diminishing sequence of regularization terms still keeps the approximability valid since for smaller penalty parameters, the regularization term gets „less wiggly", so we assume that for fixed n, one would generally need a smaller number m for a smaller λ than for a large λ.

However, we do not think that a general approximation statement that provides a universal result for arbitrary penalty terms would be possible to derive. But for a concrete penalty term with an existing approximating sequence, one could derive conditions on the sequence $(m_n)_n$ in dependence on the sequence of the regularization terms, especially for the Lasso or the Adaptive Lasso.

4.9 Conclusion

Avella-Medina (Avella-Medina [2017]) already discussed whether one can expand regularized M-estimators asymptotically linearly. However, he did not provide a rigorous theory in his work. Furthermore, he concentrated on Fréchet differentiability which, as we pointed out, is not necessary but only Hadamard differentiability.

For non-differentiable penalty terms, we faced the problem that the interchangeability of integration and differentiation to transfer M-estimation into Z-estimation is not given and that one needs techniques like the density method. Instead of requiring a tedious case-wise analysis of each of the regularization terms that we discussed, we invoked the approximation result of Avella-Medina [2017] to elegantly derive a general theory.

First, we showed under which conditions regularized M-functionals are compactly differentiable and extended existing results on compact differentiability of standard M-functionals. For "nice" penalization terms, this has been combined with another existing result, so that we directly got conditions for asymptotic linearity of regularized M-estimators with such a regularization term.

Then, for regularization terms that are not differentiable like the l_1-norm, we provided similar results for the price of relatively weak extra conditions.

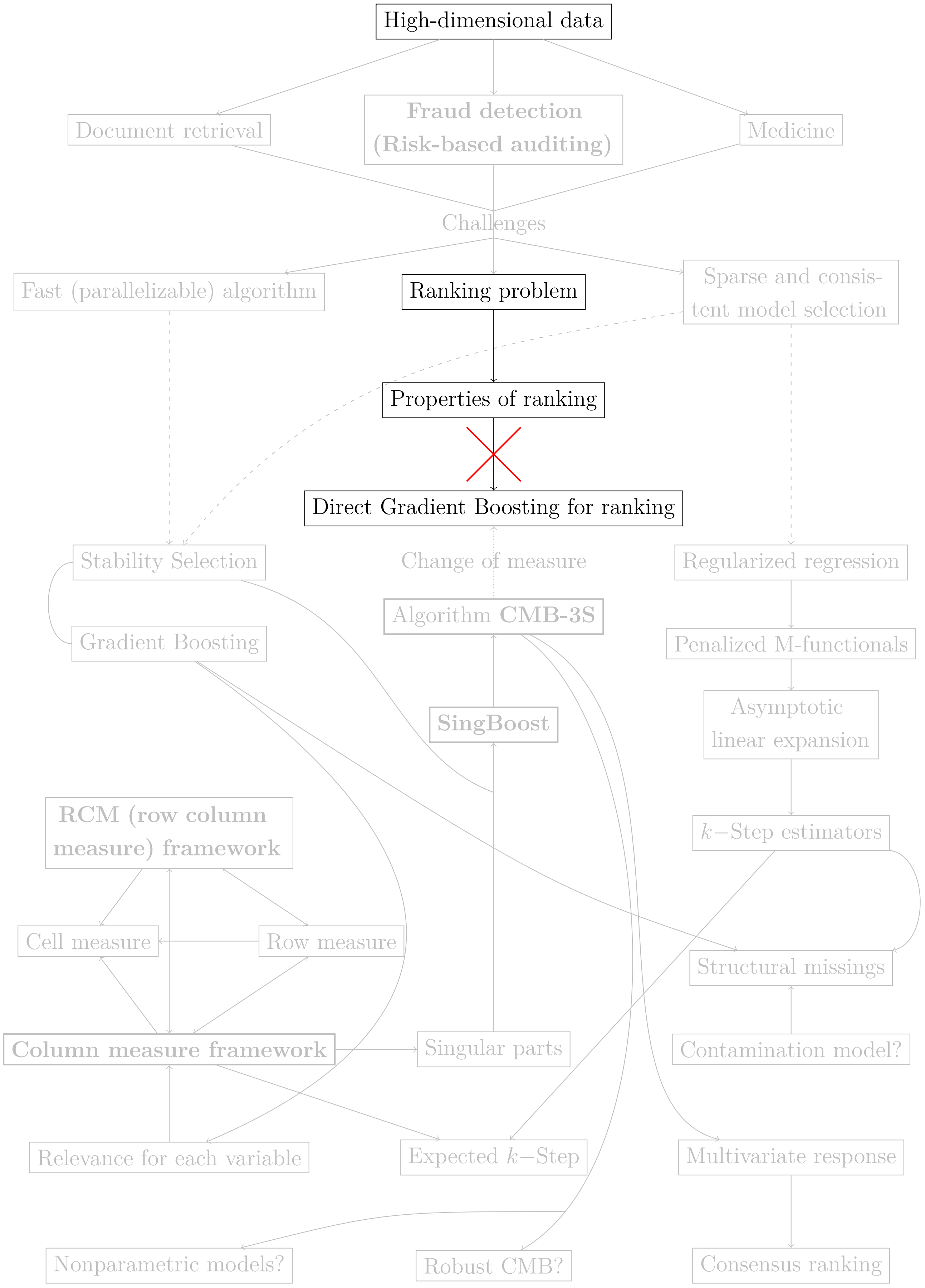
As for the case of data-dependent regularization parameters, we discussed that our results can be extended to this case if we use a sequence of indices m_n, corresponding to the approximating functions for the penalty term, according to the sequence of regularization parameters w.r.t. n.

An open problem for future work is the following:

How can the ALE of certain regularized M-estimators be used in practice, for example by constructing suitable $k-$Step estimators or for updating an existing estimator if one gets new observations?

Part II

Mathematics of ranking problems


High-dimensional data
Document retrieval
Fraud detection (Risk-based auditing)
Medicine
Challenges
Fast (parallelizable) algorithm
Ranking problem
Sparse and consistent model selection
Properties of ranking
Direct Gradient Boosting for ranking
Stability Selection
Change of measure
Regularized regression
Gradient Boosting
Algorithm CMB-3S
Penalized M-functionals
SingBoost
Asymptotic linear expansion
RCM (row column measure) framework
$k-$Step estimators
Cell measure
Row measure
Structural missings
Column measure framework
Singular parts
Contamination model?
Relevance for each variable
Expected $k-$Step
Multivariate response
Nonparametric models?
Robust CMB?
Consensus ranking

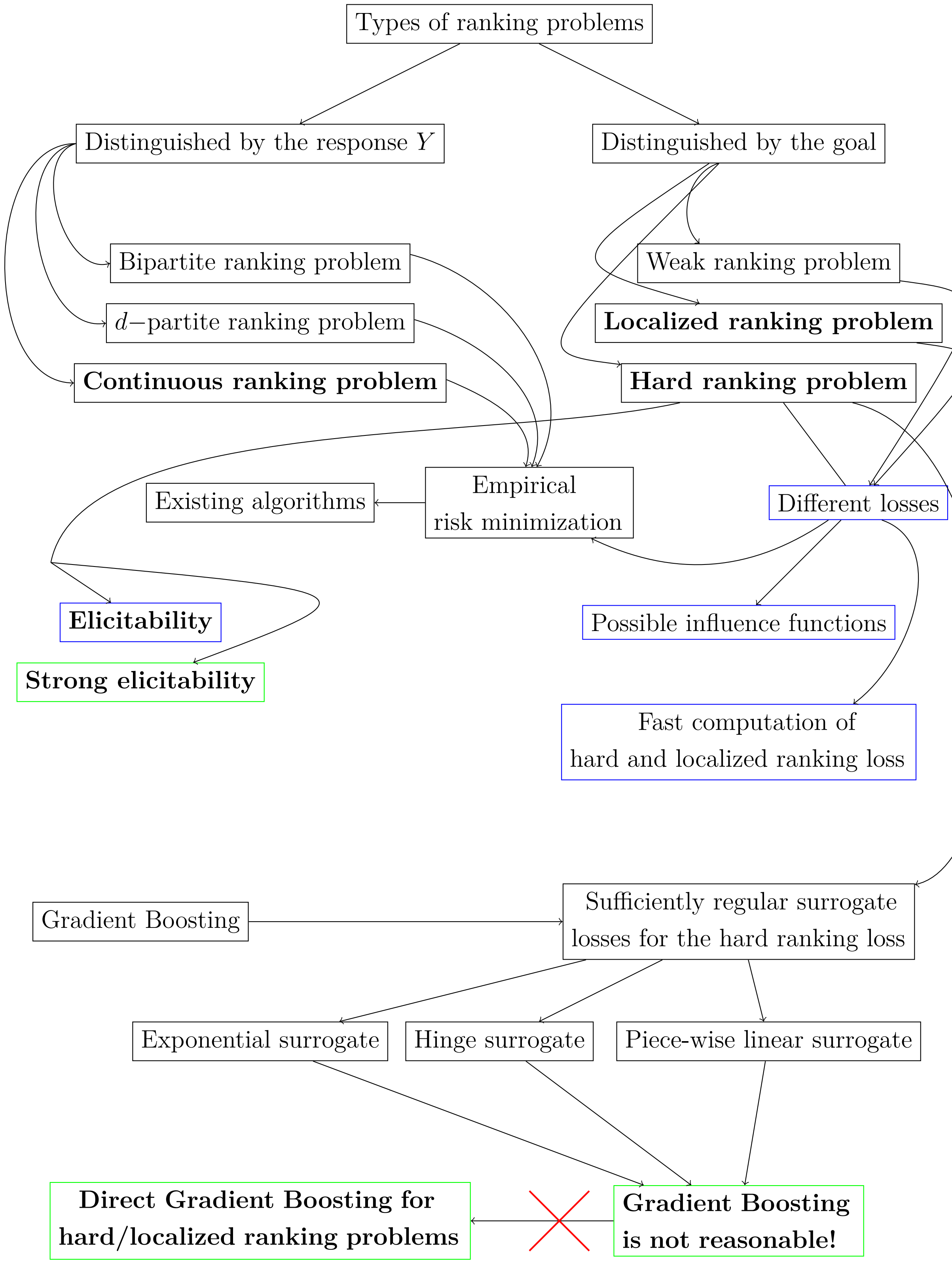
This intermediary part of the thesis is divided into four chapters.

The first chapter contains the mathematical formulation of ranking problems as empirical risk minimization problems (in the spirit of Clémençon et al. [2008]). We will further see that we should not refer to ”the” ranking problem since there exist various variants that are distinguished by either the goal or the response values. This chapter also gathers relevant existing algorithms that have been developed to solve different types of ranking problems.

In the second chapter, we analyze some properties of ranking problems, including the computation of the losses and the derivation of possible influence curves.

The third chapter concerns about the question if different competing models for the hard ranking problem can be reasonably compared to determine which one has been the best model. This directly corresponds to a property of the risk minimization problem, represented by a statistical functional itself, namely elicitability. We show that the hard ranking problem is elicitable and even strong elicitable which means that even in the case of ties, a comparison of different models is possible.

In the last chapter, we try pragmatic approaches to solve the hard ranking problem using Gradient Boosting algorithms. More precisely, we use standard surrogate losses for the indicator function as it has already been done in classification settings and transfer those ideas to get surrogates of the hard ranking loss. Simulations show that this rather naïve approach is not meaningful at all, indicating that the ranking problems that are relevant in this thesis require a lot of further work which will be done in the subsequent parts.


Types of ranking problems
Distinguished by the response Y
Distinguished by the goal
Bipartite ranking problem
$d-$partite ranking problem
Continuous ranking problem
Weak ranking problem
Localized ranking problem
Hard ranking problem
Existing algorithms
Empirical risk minimization
Different losses
Elicitability
Strong elicitability
Possible influence functions
Fast computation of hard and localized ranking loss
Gradient Boosting
Sufficiently regular surrogate losses for the hard ranking loss
Exponential surrogate
Hinge surrogate
Piece-wise linear surrogate
Direct Gradient Boosting for hard/localized ranking problems
Gradient Boosting is not reasonable!

Chapter 5

Ranking

Finding the most suspicious individuals based on machine learning techniques is the main motivation for this work. This task can be embedded into the framework of ranking problems.

Starting with the definition of several different ranking problems that are distinguished by the goal of the analyst, it becomes evident that suitable loss functions have at least a pair-wise structure in this case. We show how these loss functions look like and provide standardized versions of them for better interpretability. The respective part is closed with a very simple but illustrative example and short definitions of quality measures that originally come from classification but which also entered the ranking setting.

The formal setup is followed by two lengthy sections where we refer to related work concerning ranking problems with discrete-valued outcomes, for example in document retrieval. We also list current approaches for high-dimensional data resulting in ranking algorithms that explicitly focus on model selection.

The last part introduces the continuous ranking problem which is exactly our problem when ranking different tax payers based on the real-valued amount of damage. This kind of problem has recently been mathematically formulated, so there is not much related work yet.

5.1 Different types of ranking problems

Solutions to ranking problems do not necessarily need to recover the responses Y_i based on the observations X_i. In fact, the goal is in general to predict the right ordering of the responses albeit there exist some relaxations of this (hard) ranking problem, e.g., only the top K instances have to be ranked exactly while the predicted ranking of the other instances is not a quantity of interest. Clémençon et al. [2005] and Clémençon et al. [2008] provided the theoretical statistical framework for empirical risk minimization in the ranking setting.

In the spirit of our work, we try to rank the instances X_i by comparing their predicted response, i.e., X_i will be ranked higher than X_j if $\hat{Y}_i > \hat{Y}_j$. Then, Clémençon et al. [2008] provide the following definitions.

Definition 5.1.1. *With the convention above,*

a) *a* ***ranking rule*** *is a mapping $r : \mathcal{X} \times \mathcal{X} \to \{-1, 1\}$ where $r(x, x') = 1$ indicates that x is ranked higher than x' and vice versa.*
b) *a* ***ranking rule induced by a scoring rule*** *s is given by*

$$r(x, x', s) = 2I(s(x) \geq s(x')) - 1$$

with a scoring function $s : \mathcal{X} \to \mathbb{R}$ where $r(x, x') = 1$ precisely if $s(x) \geq s(x')$.

Remark 5.1.1. *We will always denote the index set of the true best $K \leq n$ instances by $Best_K$ and its empirical counterpart, i.e., the indices of the instances that have been predicted to be the best K ones, by $\widehat{Best_K}$.*

In this work, we will refer to the problem to correctly rank all instances as the **hard ranking problem** which is a global problem. A weaker problem is the **localized ranking problem** that intends to find the correct ordering of the best K instances, so misrankings at the bottom of the list are not taken into account. However, misclassifications have to be additionally penalized in this setting. It is obvious that these two problems are stronger problems than classification problems.

In contrast, sometimes it suffices to tackle the **weak ranking problem** where we only require to reliably detect the best K instances but where their pair-wise ordering is not a quantity of interest. For example, in the tax fraud detection context, we try to find the K most suspicious tax payers whose income tax statements need to be rigorously verified. If one knew that one exactly will review K instances, it would not be necessary to try to predict which of them is the most suspicious one. This problem has been identified in Clémençon and Vayatis [2007] as a **classification problem with a mass constraint**, since we require to get exactly K class 1 objects if class 1 is defined to be the "interesting" class.

Worked out theory for the weak and localized ranking problem is given in Clémençon and Vayatis [2007].

On the other hand, one distinguishes between three other types of ranking problems in dependence of the set $\mathcal{Y}$. If Y is binary-valued, w.l.o.g. $\mathcal{Y} = \{-1, 1\}$, then a ranking problem that intends to retrieve the correct ordering of the probabilities of the instances to belong to class 1 is called a **bipartite ranking problem (binary ranking problem)**. If Y can take d different values, a corresponding ranking problem is referred to as a $d-$**partite ranking problem** and for continuously-valued responses, one faces a **continuous ranking problem**.

Further discussions on possible combinations of these types of ranking problems and their relation to classification and regression follow in section 5.6.

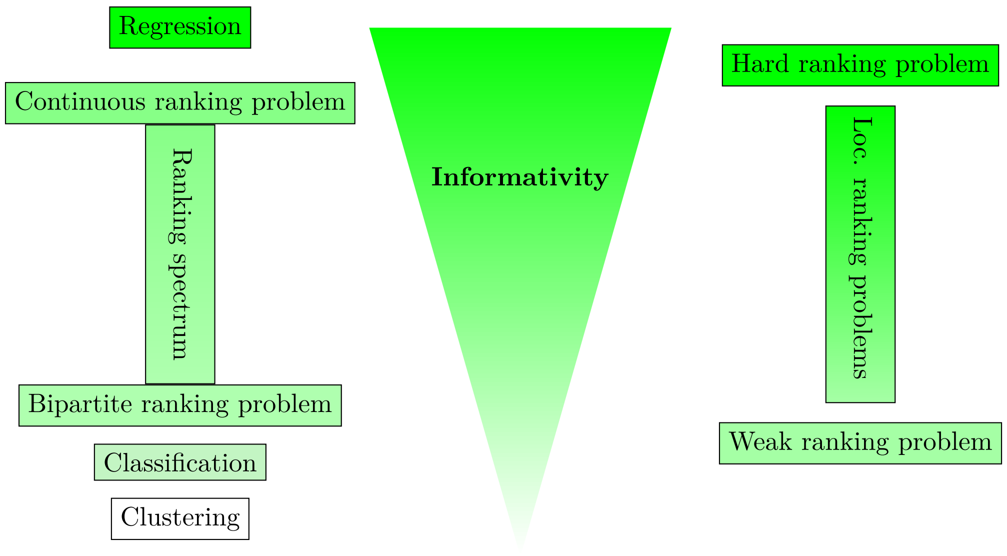


Figure 5.1: Informativity of different types of ranking problems

In figure 5.1, we illustrate how the ranking problems are related to other machine learning problems in terms of informativity. Since we do not have responses in the clustering setting, the only information that a computed clustering delivers is that the observations from a specific cluster are more "similar" to each other than to all other observations. Classification problems are comparable, but they have labels, so the "clusters" can be distinguished. Regression problems even provide information on the values of the responses which is not necessarily the case for ranking problems, so ranking problems are less informative than regression problems. For further discussion on the informativity of ranking problems compared to classification problems, see section 5.6.

The right part of the figure 5.1 is obvious since the higher the K in the localized ranking problem is, the more informative it is, with the hard ranking problem as limit case for $K = n$.

5.2 Ranking by empirical risk minimization

Empirical risk minimization needs the definition of a suitable risk function. The hard ranking risk, i.e., the risk function of the hard ranking problem, introduced in Clémençon et al. [2005] is given by

$$R^{hard}(r) := \mathbb{E}[I((Y - Y')r(X, X') < 0)], \tag{5.2.1}$$

so in fact, this is nothing but the probability of a misranking of X and X'. Thus, empirical risk minimization intends to find an optimal ranking rule by solving the optimization problem

$$\min_{r \in \mathcal{R}} \left(L_n^{hard}(r) = \frac{1}{n(n-1)} \sum\sum_{i \neq j} I((Y_i - Y_j)r(X_i, X_j) < 0) \right) \tag{5.2.2}$$

where $\mathcal{R}$ is some class of ranking rules $r : \mathcal{X} \times \mathcal{X} \to \{-1, 1\}$. For the sake of notation, the additional arguments in the loss functions are suppressed. Note that L_n^{hard}, i.e., the hard empirical risk, is also the hard ranking loss function which reflects the global nature of hard ranking problems.

The optimization problem above suffers from the difficulty that one had to optimize over a set of ranking rules whose cardinality grows with the number n of observations. This leads to a combinatorial nature of the optimization which gets infeasible for data with many observations. Additionally, just taking the ranking of the response vector of the training set obviously does not use any information of the regressors X_i, so even such a trivial approach which indeed would be a solution of the combinatorial problem (without computing scoring functions!) would not provide any predictive power. It would be comparable to the absolutely meaningless classification rule

$$s(x) = \begin{cases} Y_i, & x = X_i \ \exists i = 1, ..., n \\ 0, & \text{otherwise} \end{cases}$$

from [Devroye et al., 2013, p. 188] (which could be analogously defined in a regression setting).

In this thesis, we restrict ourselves to ranking rules that are induced by scoring rules. Considering some parameter space $\Theta \subset \mathbb{R}^p$, it suffices to empirically find the best scoring function (and with it, the empirically optimal induced ranking rule) by solving the parametric optimization problem

$$\min_{\theta \in \Theta} \left(L_n^{hard}(\theta) = \frac{1}{n(n-1)} \sum\sum_{i \neq j} I((Y_i - Y_j)(s_\theta(X_i) - s_\theta(X_j)) < 0) \right). \tag{5.2.3}$$

For the weak ranking problem, Clémençon and Vayatis [2007] introduce the upper $(1-u)-$quantile $Q(s, 1-u)$ for the random variable $s(X)$. To emphasize that this corresponds to a classification problem, we introduce the transformed responses

$$\tilde{Y}_i^{(K)} := 2I(\text{rk}(Y_i) \leq K) - 1$$

where the ranks refer to a descending ordering. Then the misclassification risk corresponding to the weak ranking problem is given by

$$R^{weak,u}(s) := P(\tilde{Y}(s(X) - Q(s, 1-u)) < 0)$$

with the empirical counterpart

$$L_n^{weak,K}(s) = \frac{1}{n} \sum_{i=1}^n I(\tilde{Y}_i^{(K)}(s(X_i) - \hat{Q}(s, 1-u^{(K)})) < 0)$$

for the empirical quantile $\hat{Q}(s, 1-u^{(K)})$. To approximate the $(1-u)-$quantile, we need to set $u^{(K)} = K/n$, i.e., for a given quantile $(1-u)$, we look at the top K instances that represent this upper quantile.

Remark 5.2.1. *Due to the mass constraint, each false positive generates exactly one false negative, so the loss can be equivalently written as*

$$L_n^{weak,K}(s) = \frac{2}{n} \sum_{i \in Best_K} I(\tilde{Y}_i(s(X_i) - \hat{Q}(s, 1-u^{(K)})) < 0).$$

Note that the weak ranking loss is not standardized, i.e., it does not necessarily take values in the whole interval $[0,1]$. More precisely, its maximal value is always $\frac{2K}{n}$, so we can only hit the value one if $K = \frac{n}{2}$ for even n and if all instances that belong to the "upper half" and predicted to be in the "lower half" and vice versa. For better comparison of the losses, we propose the **standardized weak ranking loss**

$$L_n^{weak,K,norm}(s) = \frac{1}{K} \sum_{i \in Best_K} I(\tilde{Y}_i(s(X_i) - \hat{Q}(s, 1-u^{(K)})) < 0). \tag{5.2.4}$$

Remark 5.2.2. *Having get rid of the ratio K/n, the standardized weak ranking loss function has a very intuitive interpretation. For a fixed K, a standardized weak ranking loss of c/K means that c of the instances of $Best_K$ did not have been recovered by the model.*

A suitable loss function for the localized ranking problem was proposed in Clémençon and Vayatis [2007], too. In our notation, it is given by

$$L_n^{loc,K}(s) := \frac{n_-}{n} L_n^{weak,K}(s) +$$
$$\frac{1}{n(n-1)} \sum\sum_{i \neq j} I((s(X_i) - s(X_j))(Y_i - Y_j) < 0, \min(s(X_i), s(X_j)) \geq \hat{Q}(s, 1 - u^{(K)})) \tag{5.2.5}$$

In the second summand, n_- indicates the number of negatives, so the quotient is just an estimation for $P(Y = -1)$. Note that Clémençon and Vayatis [2007] introduced this loss for binary-valued responses. We propose to set $n_- := (n - K)$ for continuously-valued responses since localizing artificially labels the top K instances as class 1 objects, hence we get $(n - K)$ negatives. Again, the second summand may be rewritten as

$$\frac{2}{n(n-1)} \sum\sum_{i<j, i,j \in \widehat{Best}_K} I((s(X_i) - s(X_j))(Y_i - Y_j) < 0).$$

As the weak ranking loss, this loss is not $[0, 1]-$standardized. Taking a closer look on it, the maximal achievable loss given a fixed K is

$$\max(L_n^{loc,K}(s)) = \frac{K(K-1)}{n(n-1)} + \frac{n-K}{n} \cdot \frac{2K}{n} =: m_K,$$

so a standardized version is simply

$$L_n^{loc,K,norm}(s) := \frac{1}{m_K} L_n^{loc,K}(s).$$

Remark 5.2.3. *Note that even in the case $K = \frac{n}{2}$ for even n, the localized ranking loss cannot take the value one. This is true since*

$$L_n^{loc,n/2}(s) \leq \frac{\frac{n}{2}\left(\frac{n}{2} - 1\right)}{n(n-1)} + \frac{\frac{n}{2}}{n} \cdot 1 < \frac{\frac{1}{2}n(n-1)}{n(n-1)} + \frac{1}{2} = 1.$$

A simple example for clarification is given below in example 5.2.1. We insist to once more take a look on the U-statistics that arise for the hard and the localized ranking problem. Clémençon et al. [2008] already mentioned that these pair-wise loss functions can be generalized to loss functions with m input arguments. This leads to U-statistics of order m. But if the whole permutations that represent the ordering of the response values should be compared at once (i.e., $m = n$), then this again boils down to a U-statistic of order 2. Let

$$\mathrm{Perm}(1:n) := \{\pi \mid \pi \text{ is a permutation of } \{1,...,n\}\}$$

and let π, $\hat{\pi} \in \mathrm{Perm}(1:n)$ be the true resp. the estimated permutation, then the empirical hard ranking loss can be equivalently written as

$$L_n^{hard}(\pi,\hat{\pi}) = \frac{2}{n(n-1)} \sum\sum_{i<j} I((\pi_i - \pi_j)(\hat{\pi}_i - \hat{\pi}_j) < 0). \tag{5.2.6}$$

Example 5.2.1. *Assume that we have a data set with the true response values*

$$Y := (-3, 10.3, -8, 12, 14, -0.5, 29, -1.1, -5.7, 119)$$

and the fitted values

$$\hat{Y} := (0.02, 0.6, 0.1, 0.47, 0.82, 0.04, 0.77, 0.09, 0.01, 0.79).$$

Then we order the vectors according to Y, so that $Y_1 \geq Y_2 \geq ...$ and get the permutations

$$\pi = (1,2,...,10), \quad \hat{\pi} = (2,3,1,5,4,8,7,9,10,6).$$

For example, $Y_{10} = 119$ is the largest value of Y, having rank 1. So we reorder $\hat{Y}$ such that $\hat{Y}_{10} = 0.79$ is the first entry. But since this is only the second-largest entry of $\hat{Y}$, we have a rank of 2, leading to the first component $\hat{\pi}_1 = 2$ and so forth.

Setting $K = 4$, we obviously get

$$L_n^{weak,4}(\pi,\hat{\pi}) = \frac{2}{10} = 0.2.$$

The standardized weak ranking loss is then

$$L_n^{weak,4,norm}(\pi,\hat{\pi}) = \frac{10}{8} \cdot \frac{2}{10} = 0.25$$

which is most intuitive since one of the indices of the four true best instances is not contained in the predicted set $\widehat{Best_4}$. The second part of the localized loss is then

$$\frac{2}{90}[0+1+0+1+0+0] = \frac{2}{45}.$$

This makes it obviously why the misclassification loss has to be included since this loss would be same if the instances of rank 4 and 5 were not switched. The complete localized ranking loss is

$$L_n^{loc,4}(\pi,\hat{\pi}) = \frac{2}{45} + \frac{6}{10} \cdot 0.2 = \frac{37}{225}.$$

The standardized localized ranking loss is then

$$L_n^{loc,4,norm}(\pi,\hat{\pi}) = \frac{75}{46} \cdot \frac{37}{225} \approx 0.268.$$

Finally, the hard ranking loss is

$$L_n^{hard}(\pi, \hat{\pi}) = \frac{2}{90} \cdot 8 = \frac{16}{90}.$$

Setting $K = 5$, the weak ranking loss is zero and the localized ranking loss is

$$L_n^{loc,5}(\pi, \hat{\pi}) = \frac{2}{90}[0+1+0+0+1+0+0+0+0+1] + \frac{5}{10} \cdot 0 = \frac{1}{15}.$$

The standardized localized ranking loss is

$$L_n^{loc,5,norm}(\pi, \hat{\pi}) = \frac{18}{13} \cdot \frac{1}{15} \approx 0.092.$$

The hard ranking loss is a global loss and does not change when changing K.

This nice and simple example has shown how important the selection of K can be.

So far, we presented loss functions for ranking problems that lead to algorithms in the spirit of the ERM paradigm. On the other hand, there also exists quality measures that are popular in classification settings but which already have been transferred to the ranking setting. Before we go into detail, we recapitulate the definition of a common quality criterion for classification.

Definition 5.2.1. *Let $Y_1, ..., Y_n$ take values in $\{-1, 1\}$ where the total number of positives is n_+ and the total number of negatives is n_-. Let $\hat{Y}_i \in \{-1, 1\}$, $i = 1, ..., n$, be predicted values.*

a) *The **true positive rate (TPR)** and the **false positive rate (FPR)** are given by*

$$\text{TPR} = \frac{1}{n_+} \sum_i I(\hat{Y}_i = 1) I(Y_i = 1), \quad \text{FPR} = \frac{1}{n_-} \sum_i I(\hat{Y}_i = 1) I(Y_i = -1).$$

b) *The **Receiver Operation Characteristic Curve (ROC Curve)** is the plot of the true positive rate against the false positive rate.*

c) *The AUC is the abbreviation for the **area under the ROC curve**.*

For theoretical aspects of the empirical AUC and its optimization, we refer to Agarwal et al. [2005], Cortes and Mohri [2004] and Calders and Jaroszewicz [2007]. We continue presenting the reparametrization of the ROC curve as it has been introduced in Clémençon et al. [2008] and used in subsequent papers of Clémençon and coauthors.

Definition 5.2.2. *For a scoring function s, the true positive rate and the false positive rate are given by*

$$\underset{s}{\mathrm{TPR}}(x) = P(s(X) \geq x \mid Y = 1), \quad \underset{s}{\mathrm{FPR}}(x) = P(s(X) \geq x \mid Y = -1).$$

Setting

$$q_{s,\alpha} := \inf\{x \in]0,1[\mid \underset{s}{\mathrm{FPR}}(x) \leq \alpha\},$$

the ROC curve is the plot of $\mathrm{TPR}_s(q_{s,\alpha})$ *against the level* α*.*

The ROC curve is a standard tool to validate binary classification rules. If the classification depends on a threshold, different points of the ROC curve are generated by changing the threshold and calculating the TPR and the FPR. Since the goal is to achieve a TPR as high as possible for the price of a FPR as low as possible, one usually chooses the threshold corresponding to the upper-leftmost point of the empirical ROC curve. A combined quality measure that incorporates all points of the ROC curve is the AUC where a classification rule is better the higher the empirical AUC is. Random guessing clearly has a theoretical AUC of 0.5.

The following result that connects the AUC with ranking via a scoring rule was shown in [Clémençon et al., 2008, Lemma B2].

Lemma 5.2.1. *Let* (X, Y)*,* (X', Y') *be independently distributed where* Y*,* Y' *take values in* $\{-1, 1\}$ *and* X*,* X' *are* $\mathcal{X}-$*valued for some measurable space* $\mathcal{X}$*. Let* $s : \mathcal{X} \to \mathbb{R}$ *be a scoring function. Then*

$$\mathrm{AUC}(s) = P(s(X) \geq s(X') \mid Y = 1, Y' = -1).$$

For the localized ranking problem, Clémençon and Vayatis [2007] provide the following localized version of the AUC.

Definition 5.2.3. *The **localized AUC** is defined as*

$$\mathrm{LocAUC}(s, \alpha) := P(s(X) > s(X'), s(X) \geq Q(s, 1 - \alpha) \mid Y = 1, Y' = -1).$$

Remark 5.2.4 (**Optimal ranking rules**)**.** *It has been shown in Clémençon and Achab [2017] that* $\mathbb{E}[Y|X]$ *is an optimal scoring rule for the hard continuous ranking problem.*

Furthermore, as also pointed out in Clémençon and Achab [2017], if there is some optimal scoring rule for the continuous ranking problem, clearly any strictly increasing transformation *of the scoring rule is also optimal.*

Figure 5.2 illustrates this for a very simple example with $p = 1$.

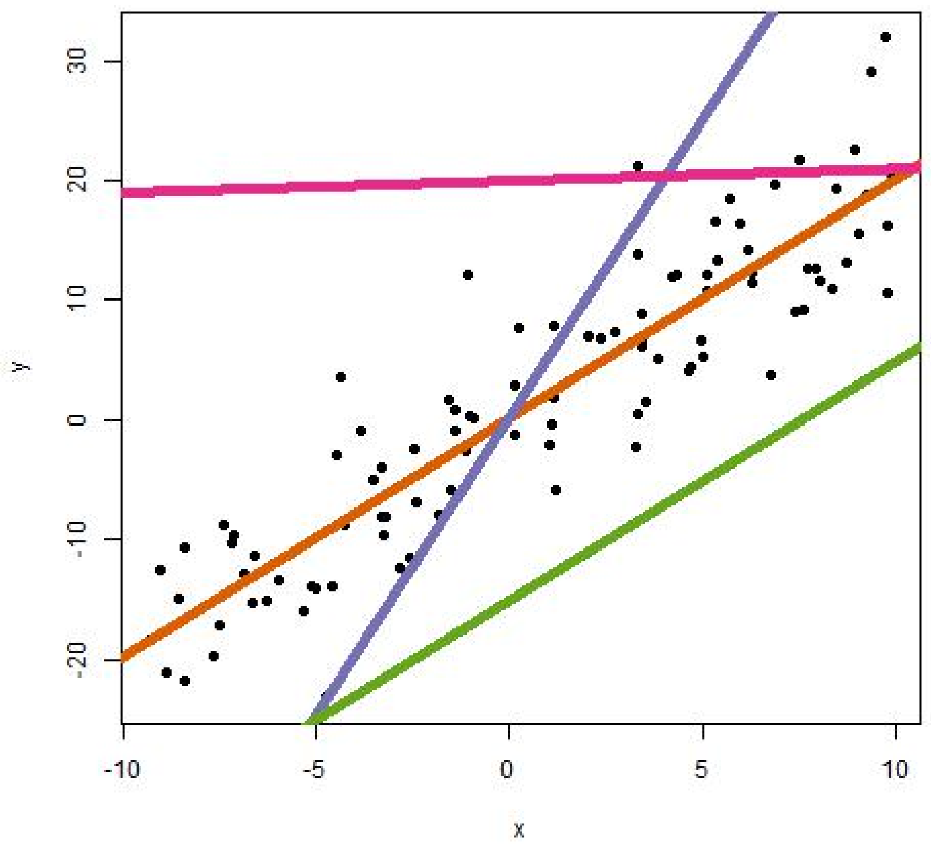


Figure 5.2: The linear regression line (orange) approximates $\mathbb{E}[Y|X]$ and therefore is an optimal scoring rule. Each of the linear models corresponding to the other lines provide optimal scoring rules, too, and result in the same ranking performance

In contrast, for $p > 1$, the choice of the coefficients indeed affects the ranking of the fitted values. For large p, small changes in the coefficients and therefore in the regression hyperplane can have significant effects. The following example demonstrates the effect for $p = 10$.

```
require(mvtnorm)
set.seed(144)
X←rmvnorm(10,mean=rep(0,10))
set.seed(17)
beta←rnorm(10,mean=0.5,sd=2)
```

```
rank(X%*%beta)
```

```
 [1]  8  4  7  1  5  2 10  6  9  3
```

```
set.seed(73)
m<-rnorm(10,mean=1,sd=0.25)
betam<-beta*m
rank(X%*%betam)
```

```
 [1]  6  2  8  1  9  3 10  4  7  5
```

```
set.seed(37)
m2<-rnorm(10,mean=1,sd=0.25)
betam2<-beta*m2
rank(X%*%betam2)
```

```
 [1]  8  3  9  1  7  2 10  5  6  4
```

5.3 Existing algorithms for ranking problems

The existing methods to solve ranking problems make use of standard machine learning algorithms like Support Vector Machines (SVM), Boosting, trees or linear regression.

First, we note that there also exist Bayes-type algorithms where a distribution on the set of possible permutations is computed. Two prominent models are the Mallows model and the Plackett-Luce model. The Mallows model (Mallows [1957]) is based on distances between different permutations, in general based on Kendall's Tau, which leads to a maximum likelihood approach. The Plackett-Luce model (Luce [1959], Plackett [1975]) performs a Bayes estimation. However, we do not go into detail since the types of algorithms that we encounter in this work are different.

In the case of **bipartite ranking**, the sometimes called "plug-in approach" that estimates the conditional probability $P(Y=1|X=x)$ can be realized for example by **LogitBoost**, i.e., minimizing the loss

$$\frac{1}{n}\sum_i \log_2(1+\exp(-2Y_i s(X_i))).$$

The resulting function s is then used as a ($[0,1]-$valued) scoring function for the ranking. However, the plug-in approach has disadvantages when facing high-dimensional data and when trying to establish an approximation of the true ROC curve in certain norms as pointed out in Clémençon and Vayatis [2008], Clémençon and Vayatis [2010].

Remark 5.3.1. *Taking a closer look on this loss function, it is indeed a convex surrogate of the misclassification loss. Concerning informativity, one just applies an algorithm that solves a classification problem which is less informative than a ranking problem which is another aspect why this approach may not be optimal.*

One approach to solve the bipartite ranking problem is to empirically maximize the AUC. This has been done in Rakotomamonjy [2004] and Ataman and Street [2005] by defining a formula of Mann-Whitney-type to calculate the empirical AUC resulting in an SVM-type ranking algorithm. A similar idea was presented in Brefeld and Scheffer [2005], providing the SVM-type algorithm **AUC-SVM**.

A Boosting-type algorithm that essentially minimizes the empirical pair-wise exponential loss

$$\frac{1}{n(n-1)}\sum\sum_{i<j}\exp(-(Y_i-Y_j)(s(X_i)-s(X_j)))$$

for some scoring rule s is the **RankBoost** algorithm developed in Freund et al. [2003]. It is shown in Rudin and Schapire [2009] that in the case of binary outcome variables, RankBoost and the well-known classifier AdaBoost (Freund and Schapire [1997]) are equivalent under very weak assumptions. Therefore, RankBoost can also be seen as an AUC maximizer when concerning the bipartite ranking problem. Another algorithm that includes gradients is the **RankNet** of Burges et al. [2005] that applies neural networks.

Note that there is a small mistake in Section 3.2.1 of Clémençon et al. [2013b] since the minus sign in the exponential function is missing. But if $Y_i > Y_j$ and $s(X_i) > s(X_j)$, the sign of the product is positive which would imply a high loss due to a positive exponent without the minus sign.

Clémençon and coauthors provided two tree-type algorithms, **TreeRank** and **RankOver** (Clémençon and Vayatis [2008], Clémençon and Vayatis [2010]). The idea behind the TreeRank algorithm is to divide the feature space $\mathcal{X}$ into disjoint parts C_j and to construct a piece-wise constant scoring function

$$s_N(x)=\sum_{j=1}^{N}a_jI(x\in C_j)$$

for $a_1 > ... > a_N$. This results in a ROC curve that is piece-wise linear with $(N-1)$ nodes (not counting $(0,0),(1,1)$) as shown in [Clémençon and Vayatis, 2008, Prop. 13]. The TreeRank algorithm then adaptively adds nodes between all existing nodes such that the ROC curve approximates the optimal ROC curve by splitting each region C_j in two parts. Extensions by combining the TreeRank algorithm with bagging in a RandomForest-like sense are given in Clémençon et al. [2009], Clémençon et al. [2013a]. The question how to prune a ranking tree was tackled in Clémençon et al. [2011].

Similarly, the RankOver algorithm constructs a piece-wise linear approximaton of the optimal ROC curve by computing a piece-wise constant scoring function, too, but instead of partitioning the feature space, it generates a partition of the ROC space.

Theoretically, these tree-type algorithms provide an advantage over the algorithms that optimize the AUC since they approximate the optimal ROC curve in an $L_\infty-$sense while the competitors just optimize the ROC in an L_1-sense (see [Clémençon and Vayatis, 2010, Sec. 2.2]). On the other hand, they suffer from strong assumptions since it is required that the optimal ROC curve is known. Additionally, this optimal ROC curve has to fulfill some regularity conditions which is differentiability and concavity for the TreeRank algorithm and twice differentiability with bounded second derivatives for the RankOver algorithm.

It is pointed out in Clémençon et al. [2013b] that ranking algorithms that intend to maximize the global AUC like RankBoost are not expected to be good candidates for the localized ranking problem whereas TreeRank is also constructed to solve that problem.

There are some other ranking algorithms that directly follow the ERM principle. As an extension of RankBoost, Rudin [2009] modified the exponential loss of RankBoost to the power loss

$$\frac{1}{n(n-1)}\sum\sum_{i<j}(\exp((Y_i-Y_j)(s(X_i)-s(X_j))))^p$$

for some $p \geq 1$. This algorithm is called the **p-Norm-Push**. The case $p=\infty$ has been studied in Rakotomamonjy [2012].

The **RankingSVM** algorithm (see Herbrich et al. [1999], Joachims [2002]) minimizes the empirical Hinge-type loss

$$\frac{2}{n(n-1)}\sum\sum_{i<j}[1-(Y_i-Y_j)(f(X_i)-f(X_j))]_+ + \lambda||f||^2_{\mathcal{H}_K}.$$

where $\mathcal{H}_K$ is some Reproducing Kernel Hilbert Space (RKHS) defined by a kernel K (see Schölkopf et al. [2001]).

Note that RankBoost, p-Norm-Push and RankingSVM essentially use surrogates for the hard ranking loss. For theoretical work on surrogate losses for the bipartite ranking problem, see

Agarwal [2014].

Another approach is to try to predict the differences of pair-wise responses by the differences of the corresponding features. This can be achieved by minimizing the least-squares-type criterion

$$\frac{1}{n(n-1)}\sum\sum_{i<j}((f(X_i)-f(X_j))-(Y_i-Y_j))^2+\lambda||f||_{\mathcal{H}_k}^2.$$

for some function $f:\mathcal{X}\to\mathbb{R}$ and some kernel K with corresponding RKHS $\mathcal{H}_K$ (Pahikkala et al. [2007]). Using the representer theorem (see e.g. Schölkopf et al. [2001]), the solution has the form

$$f(X)=\sum_{i=1}^{N}a_iK(X,X_i)$$

for some $a_i\in\mathbb{R}$. The algorithm is called **RankRLS** (”regularized least squares”).

For a useful overview and empirical comparison of some of these ranking algorithms, we refer to Clémençon et al. [2013b].

The $d-$**partite ranking problem**, i.e., where the responses can take d different values, has been theoretically studied in Clémençon et al. [2013c] and Clémençon and Robbiano [2014].

First, an extension of the AUC in the multipartite case was adapted to the concrete setting in Clémençon et al. [2013c]. The AUC is generalized to the **volume under the ROC surface (VUS)** where the ROC surface is the continuous extension of discontinuity points based on hyperplane parts (note that the ROC surface originally was introduced in Yang and Carlin [2000]). Clémençon and Robbiano [2015b] provide the algorithm **TreeRank Tournament** which is based on RankTree and locally optimizes the ROC surface. A bagged and randomized version of TreeRank Tournament has been studied in Clémençon and Robbiano [2015a].

Remark 5.3.2. *Note that approaches that require the computation of the ROC curve and therefore of the AUC get infeasible in high dimensions due to the curse of dimensionality affecting numerical quadrature.*

5.4 Other ranking approaches with model selection

The LogitBoost approach already performs model selection, but as we pointed out, it is just a plug-in approach optimizing a surrogate loss function. Pan et al. [2009] mention that it also has been tried to combine the RankBoost algorithm with stumps, but that this strategy has weaknesses in their search engine context. The first work on solving a ranking problem while explicitly concerning on sparse model selection was done by Geng et al. (Geng et al. [2007]). They propose to assign an importance score to every single feature by computing a ranking based on each of them and computing a measure like the MAP (mean average precision) or NDCG (normalized discounted cumulative gain). Furthermore, a similarity measure between the features based on Kendall's τ is computed. Geng et al. [2007] solve an optimization problem to maximize the importance while minimizing the similarity to get a reduced feature set and eventually perform RankingSVM or RankNet on it. A similar strategy is spelled out in Pan et al. [2009] where stumps are used as baselearners and where the importance measure of [Friedman, 2001, Sec. 8] is used to find the most relevant features.

A robust and sparse ranking algorithm has been established in Sun et al. [2009], called **RSRank** for binary responses. They make use of a result in Balcan et al. [2008] that allows to reduce the binary ranking problem to weighted pair-wise classification. The goal in Sun et al. [2009] is to directly optimize the ranking loss measured by NDCG by minimizing the loss function

$$\sum_{i<j}\sum w(Y_i, Y_j, c)I(Y_i > Y_j)c(X_i, X_j)$$

where c is a pair-wise classifier and w a suitably chosen importance weight. For optimization, the indicator function is replaced by the Huber function and an l_1-penalty term is added. The minimization of the objective function is done by a truncated Gradient Boosting algorithm.

A pointed out in Tsivtsivadze and Heskes [2013], RankingSVM can also provide sparse solutions but they suffer from the lack of interpretability inherited from support vector machines. The objective function of RankingSVM is essentially l_2-constrained. Therefore, Lai et al. [2013a] replace this penalty with an l_1-regularization term and solve the problem by invoking Fenchel duality. Their algorithm is hence called **FenchelRank** which has been empirically shown to be even superior to RSRank in terms of precision. An iterative gradient procedure for this problem has been developed in Lai et al. [2013b] and shows comparable performance. As an extension of FenchelRank, Laporte et al. [2014] tackle the same problem with nonconvex regularization to get even sparser models. They solve the problem with a so-called majorization minimization method where the nonconvex regularization term is represented by the difference of two convex functions. In addition, for convex regularization, they present an approach that relies on differentiability and Lipschitz continuity of the penalty term so that the ISTA-algorithm can be applied.

Of course, trees like the ones Clémençon and co-authors provided, also perform feature selection.

5.5 Ranking with continuous outcomes

All the presented existing algorithms are tailored to ranking problems with discrete labels since most of them are motivated by information retrieval or document ranking. To the best of our knowledge, the only approach explicitly designed for ranking real-valued responses was recently proposed by Clémençon (Clémençon and Achab [2017]).

Let w.l.o.g. $Y \in [0,1]$. Then each **subproblem**

$$\max_s (P(s(X) > t | Y > y) - P(s(X) > t | Y < y))$$

for $y \in [0,1]$, i.e., $s(X)$ given $Y > y$ should be **stochastically larger than** $s(X)$ given $Y < y$, is a bipartite ranking problem, so the continuous ranking problem can be regarded as a so-called **"continuum" of bipartite ranking problems** (Clémençon and Achab [2017]).

As a suitable performance measure, they provide the area under the integrated ROC curve

$$\mathrm{IAUC}(s) := \int_0^1 \underset{s}{\mathrm{IROC}}(\alpha) d\alpha := \int_0^1 \int \underset{s,y}{\mathrm{ROC}}(\alpha) dF_y(y) d\alpha$$

where $\mathrm{ROC}_{s,y}$ indicates the ROC curve of scoring function s for the bipartite ranking problem corresponding to $y \in]0,1[$ and where F_y is the marginal distribution of Y. Alternatively, they make use of Kendall's τ as a performance measure for continuous ranking.

The approach presented in Clémençon and Achab [2017] manifests itself in the tree-type **CRank** algorithm that divides the input space and therefore the training data into disjoint regions. In each step/node, the binary classification problem corresponding to the median of the current part of the training data is formulated and solved. Then, all instances whose predicted label was positive are delegated to the left children node, the others to the right children node. Stopping when a predefined depth of the tree is reached, the instance of the leftmost leaf is ranked highest and so far, so the rightmost leaf indicates the bottom instance.

Remark 5.5.1 (**Interpretability**). *Note that single trees in general lack the properties of stability and robustness (see for example Friedman et al. [2001]). On the other hand, when aggregating trees like it has been done for trees that solve the bipartite ranking problem (Clémençon et al. [2009], Clémençon et al. [2013a]), one essentially faces partial rankings predicted by each single tree. To aggregate them, they compute a so-called consensus ranking (see section 13.1 for more details) from these partial rankings and a median scoring rule. As for RandomForests, it would be very hard to interpret this combined model in the sense of quantifying the impact of a single predictor on the consensus ranking.*

Remark 5.5.2 (High-dimensional data). *Furthermore, it is not evident if the tree-type algorithms provided by Clémençon and coauthors are designed for very high-dimensional data, i.e., for very large p. In Clémençon et al. [2013b] where different ranking algorithms have been applied to real data sets, the number of variables is never larger than 34. We note that in Dhanjal and Clémençon [2014], there indeed has been analyzed high-dimensional data by using forest-type algorithms, but with a fairly large number of observations.*

5.6 Ranking vs. ordinal regression

Ordinal regression problems are indeed very closely related to ranking problems. As already pointed out in Robbiano [2013], especially multipartite ranking problems (Clémençon et al. [2013c]) share the main ingredient, i.e., the computation of a scoring function that should provide pseudo-responses with a suitable ordering. However, the main difference is that the multipartite ranking problem is already solved once the ordering of the pseudo-responses is correct while the ordinal regression problem still needs thresholds such that a discretization of the pseudo-responses into the d classes of the original responses is correct.

Note that due to the discretization, ordinal regression problems can also be perfectly solved even if the rankings provided by the scoring function are not perfect. For example, consider observations with indices $i_1, ..., i_{m_k}$ that belong to class k. If for a scoring rule s we had the predicted ordering $s(X_{i_1}) < s(X_{i_2}) < ... < s(X_{i_{m_k}})$ but the true ordering is different, then we can still choose thresholds such that all m_k instances that belong to class k (and no other instance) are classified into this class, provided that $s(X_i) \notin [s(X_{i_1}), s(X_{i_{m_k}})] \ \forall i \notin \{i_1, ..., i_{m_k}\}$. Though, as Robbiano [2013] already pointed out, the ordinal regression is based on another loss function.

Concerning informativity, one can state that multipartite ranking problems are more informative than ordinal regression problems due to the chunking that is done in the latter ones. But in fact, in an intermediary step, i.e., when having computed the scoring function, the ordinal regression problem is as informative as multipartite ranking problems. This is also true for standard logit or probit models (the two classes generally are not ordered, but when artificially replacing the true labels by -1 and $+1$ where the particular assignment does not affect the quality of the models, they can indeed be treated as ordinal regression models) where the real-valued pseudo-responses computed by the scoring function are discretized at the end to have again two classes.

The continuous ranking problem can be treated as a special case where no pseudo-responses are needed since the original responses are already real-valued, but again, instead of optimizing some regression loss function, the goal is actually to optimize a ranking loss function.

For further discussions on the relation of ranking and ordinal regression (also called ”ordinal classification” and ”ordinal ranking” in the reference), see Lin [2008].

From this point of view, the three combined problems for the continuous case, i.e., weak, hard and localized continuous ranking problems, are easy to distinguish and are all meaningful. Hard bipartite and hard $d-$partite ranking problems are essentially optimized by the corresponding algorithms that we listed earlier in section 5.3 and localized bipartite ranking problems can be solved using the tree-type algorithms of Clémençon as pointed out for instance in Clémençon et al. [2013b].

Clearly, these localized bipartite problems directly reflect the motivation from risk-based auditing or document retrieval. It has been mentioned in Clémençon and Robbiano [2015b] that their tree-type algorithm is not able to optimize the VUS locally. To the best of our knowledge, this has not been achieved until now. But indeed, localized $d-$partite ranking problems can also be interesting in document retrieval settings where the classes represent different degrees of relevance. Then it would be interesting for example to just recover the correct ranking of the relevant instances, i.e., the ones from the ”best” $(d-1)$ classes.

As mentioned earlier, weak ranking problems can be identified with binary classification with a mass constraint. In the case of weak bipartite ranking problems, it may sounds strange to essentially mix up two classification paradigms, but one can think of performing binary classification by computing a scoring function and by predicting each instance as element of class 1 whose score exceeds some threshold, as it is done for example in logit or probit models. One can think of choosing the threshold such that there are exactly K instances classified into class 1 instead of optimizing the AUC or some misclassification rate.

The only combination that does not seem to be meaningful at all would be weak $d-$partite ranking problems. By its inherent nature, a weak ranking problem imposes are binarity which cannot be reasonably given for the $d-$partite case. Even in the document retrieval setting, a weak $d-$partite ranking problem may be thought of trying to find the K most important documents which implied that the information that is already given by the d classes would be boiled down to essentially two classes, so this combination is not reasonable.

Chapter 6

Some properties of ranking

Note that a naïve evaluation of the hard ranking loss requires $\mathcal{O}(n^2)$ comparisons. This will surely become infeasible for data sets with many observations. We provide a solution so that the number of necessary evaluations will boil down to $\mathcal{O}(n\ln(n))$.

In the second section, we gather some small results on properties of ranking. First, we propose an influence function for a general ranking functional by approximating the ranking by a special generalized linear model (GLM). For a suitable link function, we will transfer the existing methods of quantitative robustness for GLMs to ranking. Therefore, we recap results on extensions of L_2-differentiability of parametric linear regression models (cf. Rieder [1994], section 4.4) from Pupashenko [2015], Pupashenko et al. [2015] for GLMs where the distribution of the error term does not necessarily stem from an exponential family.

6.1 Fast computation of the hard ranking loss

A key question concerning the implementation of ranking algorithms arises when the hard ranking loss of 5.2.3, equivalently formulated for predictions $\hat{Y}_i$ instead of a ranking rule r by

$$L_n^{hard}(Y,\hat{Y}) = \frac{1}{n(n-1)} \sum\sum_{i \neq j} I((Y_i - Y_j)(\hat{Y}_i - \hat{Y}_j) < 0),$$

needs to be evaluated several times which indeed will be required later in this thesis.

The naïve way is to compute all pair-wise differences, separately for the vectors Y and $\hat{Y}$, by using the R$-$command `outer`. Then all cells where the signs of the differences do not coincide are counted as one misranking, leading to the following code for the case that neither

Y nor $\hat{Y}$ have ties:

```
loss=function(y,yhat){
        m←length(y)
        oopt←outer(y,y,function(x,z)z−x)
        ohat←outer(yhat,yhat,function(x,y)y−x)
        return(sum(sign(oopt)−sign(ohat)!=0)/m/(m−1))
}
```

Unfortunately, the computation is very time-consuming for high numbers n of observations, making this approach noncompetitive.

We take a look at the concordance measure

$$\tau(Y,\hat{Y}) := \frac{1}{n(n-1)}\sum_{i\neq j}\sum \operatorname{sign}((Y_i-Y_j)(\hat{Y}_i-\hat{Y}_j))$$

called **Kendall's Tau**. Unlike the ranking loss which is high if there are many misrankings and which is $[0,1]-$valued, the Kendall's Tau is high if many pairs are concordant, i.e., if the pair-wise ranking is correct in most cases and takes values in $[-1,1]$.

This leads to a bijection between these two quantities if we do not face ties.

Lemma 6.1.1 (**Hard ranking loss and Kendall's Tau**). *Assume the vectors x and y have the same length n and do not contain ties. Then it holds that*

$$L_n^{hard}(x,y) = \frac{1-\tau(x,y)}{2}.$$

Proof. *In the case of a perfect concordance, the ranking loss function is zero whereas Kendall's τ takes the value 1. If we produce one misranking, w.l.o.g. by swapping the largest and the second largest entry x_{j_1} resp. x_{j_2} of x, the indicator function in the ranking loss jumps from zero to 1 for $(i,j)\in\{(j_1,j_2),(j_2,j_1)\}$, increasing the total ranking loss by $\frac{2}{n(n-1)}$. The same manipulation results in the summands for the same indices in the Kendall's τ changing from 1 to -1, decreasing it by $\frac{4}{n(n-1)}$.*

By induction, the claimed formula is valid.

□

Luckily, there exists an R−command that provides fast computation of Kendall's τ, namely the command `cor.fk` from the package `pcaPP` (Filzmoser et al. [2018]). The algorithm essentially goes back to Knight (Knight [1966]) and relies on the idea of fast ordering algorithms. So in fact, we first compute Kendall's Tau using `cor.fk` and then, we use the bijection to compute the hard ranking loss which results in the number of calculations necessary for the computation of the hard ranking loss decreasing from $\mathcal{O}(n^2)$ in the naïve implementation to $\mathcal{O}(n\ln(n))$. Thus, the code for evaluating the hard ranking loss just looks like this:

```
require(pcaPP)
loss=function(y,f){
      return((1-cor.fk(y,f))/2)
   }
```

In figure 6.1, we have data sets with n observations and p regressor columns and compute several localized ranking losses, including the hard ranking loss as limit case, for the predicted response vector computed by simple linear models, separately for each column, w.r.t. the original responses. This is already called "singular step" in the title of the graphic. For more details, see part III.

It is expected that the cost grows linearly with p which results in the concave-shaped parts as we see in the figure. Furthermore, for different n we expect that the cost satisfies the complexity $\mathcal{O}(n^2)$ for the naïve computation (black) and $\mathcal{O}(n\ln(n))$ otherwise which does not exactly in the figure due to the very low times for these computations.

Remark 6.1.1. *By the same arguments, one can show that if either the sorted vector x or the sorted vector y has k consecutive ties, then we get the mapping*

$$L_n^{hard}(x,y) = \frac{1-\tau(x,y)}{2} - \frac{k(k-1)}{2n(n-1)}.$$

This is true since any such tie produces a zero difference in two summands, hence not affecting the ranking loss but indeed Kendall's τ. If a sequence of k equal entries of x or y occurs, one gets $k(k-1)$ such defects. By halving, there is still a discrepance of $\frac{k(k-1)}{2n(n-1)}$ which must be subtracted to get the ranking loss.

Although the implementation of `cor.fk` can also handle the case of ties, we restrict ourselves to the case without ties because finding a bijection in the case where both vectors contain ties or where the ties are not consecutive gets much more difficult and since we are interested in the continuous ranking problem anyways.

For the localized ranking loss 5.2.5, which includes the computation of the hard ranking loss restricted on the estimated set of best instances $\widehat{Best_K}$ (though differently standardized), we can use this function again. However, since K is usually small in comparison with n, the benefit of the accelerated computation may be smaller. See also figure 6.1.

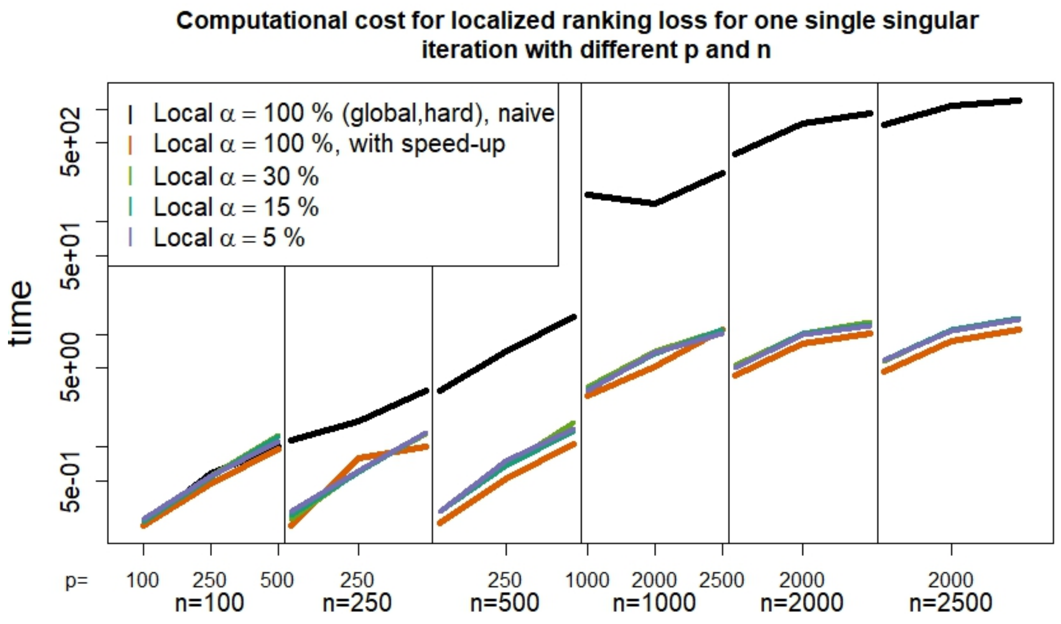


Figure 6.1: Time consumption: Naïve computation of the hard ranking loss and our computations of the hard and some localized ranking losses, exemplary on one data set for each allocation

6.2 Quantitative robustness of ranking

In this section, we try to derive influence functions for the ranking problems that have been introduced in the previous section.
After defining L_2-differentiability of GLMs, we compute the score of the ordered logit model and show how it can be used to get influence curves for ranking problems.

6.2.1 L_2-differentiability of generalized linear regression models

In the following subsection, we try to get influence functions for ranking problems through ordered logit models. Since we always assume L_2-differentiability of the underlying model when concerning about influence functions, we briefly recap results of Pupashenko et al. [2015] on L_2-differentiability of generalized linear model families even if the error distribution is not part of an exponential family, for example, if the error term follows a generalized extreme value distribution (GEVD) or a generalized Pareto distribution (GPD) as in Pupashenko [2015]. In fact, Pupashenko et al. [2015] extended the results of Rieder that we needed in section 4.4 to the case of generalized linear models with an arbitrary error distribution.

Suppressing the subscript, let $x \in \mathbb{R}^{1 \times p}$ always be a row of the regressor matrix and let

$\beta \in \Theta \subset \mathbb{R}^p$. The rest of the notation is as before in 4.4. Pupashenko et al. [2015] respected the case that the parameter β may be partitioned into d blocks of length p_h, $h = 1, ..., d$, and introduced the maps

$$T_\pi : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^d, \ T_\pi(a, b) := \left(\sum_j a_{h,j} b_{h,j} \right)_{h=1,\ldots,d},$$

$$\rho_\pi : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^p, \ \rho_\pi(c, a) := (c_h a_{h,j})_{h=1,\ldots,d, j=1,\ldots,p_h}$$

$$M_\pi : \mathbb{R}^{d\times d} \times \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^{p\times p}, \ M_\pi(C, a, b) := (C_{h_1,h_2} a_{h_1,j_1} b_{h_2,j_2})_{h_1,h_2=1,\ldots,d; j_1,j_2=1,\ldots,p_h}$$

to elegantly describe matrix and vector multiplications with such partitions π.

Essentially, the goal is to estimate some parameter $\vartheta \in \Upsilon \subset \mathbb{R}^d$. Defining the parametric model

$$\mathcal{Q} := \{Q_\vartheta \mid \vartheta \in \Upsilon\} \subset \mathcal{M}_1(\mathcal{A})$$

and a continuously differentiable link function $\ell : \mathbb{R}^d \to \Upsilon$ such that

$$\vartheta = \ell(\theta) := \ell(T_\pi(x, \beta)) \tag{6.2.1}$$

where in the standard case $T_\pi(x, \beta) = x\beta$, the parametric generalized linear regression model is given by

$$\mathcal{P} := \{P_\beta(dx, dy) := Q_{\ell(\theta)}(dy|x)F(dx) \mid \beta \in \Theta, Q_\vartheta \in \mathcal{Q}\}. \tag{6.2.2}$$

By the chain rule, the score function w.r.t. β is given by

$$\Lambda_\beta^{\mathcal{P}}(x|y) = \partial_\beta \ell(\theta) \partial_\vartheta Q_\vartheta(y|x) = \partial_\theta \ell(\theta) x \Lambda_\vartheta^{\mathcal{Q}}(y)$$

which in the case of a block partition can be rewritten as

$$\Lambda_\beta^{\mathcal{P}}(y) = \rho_\pi((\partial_\theta \ell(\theta))^T \Lambda_\vartheta^{\mathcal{Q}}(y), x).$$

Pupashenko et al. worked out conditions (see [Pupashenko, 2015, Thm. 6.1], [Pupashenko et al., 2015, Thm. 2.3]) that are sufficient for the model (6.2.2) to be continuously L_2−differentiable in some β_0 with L_2−derivative $\Lambda_\beta^{\mathcal{P}}$ using Vitali's theorem. They guaranteed the continuity by invoking the lemma of Hájek (3.2.1) where the definition A.3.5 was used for absolute continuity in more than one dimension in the case of a block partition structure.

We note that [Pupashenko et al., 2015, Thm. 2.6] also provide a result extending [Rieder, 1994, Thm. 2.4.2], i.e., the case of deterministic design of the regressor matrix, to the GLM case.

Note that there is a small mistake in [Pupashenko et al., 2015, Rem 2.4] where $|I_{\vartheta_0}^{\mathcal{P}}|$ should be replaced by $|I_{\vartheta_0}^{\mathcal{Q}}|$ in the first integrability condition.

6.2.2 The logit model and the ordered logit model

Binary-valued outcomes cannot be directly predicted by a linear model since the latter produces real-valued predictions. Therefore, one uses a GLM where the link function is the logit function logit $:]0,1[\to \mathbb{R}$, resulting in the logit model

$$\operatorname{logit}(q) = \left(\frac{\ln(q)}{\ln(1-q)}\right) = x\beta \Longleftrightarrow q = \operatorname{expit}(X\beta) = \frac{e^{X\beta}}{1+e^{X\beta}}$$

with expit $= \operatorname{logit}^{-1} : \mathbb{R} \to]0,1[$ and $q := P(Y=1|X)$.

A different situation occurs if the responses are ordinal and take w.l.o.g. values in $\{1,...,d\}$. Then a so-called **threshold model** or **ordered discrete choice model** is applicable (cf. Berridge and Crouchley [2011], Fahrmeir et al. [2007], Fahrmeir and Tutz [2013]). Defining the linear model

$$Y' = X\beta + \epsilon \tag{6.2.3}$$

for $\epsilon \sim F_\epsilon$ and a pseudo-response Y', one defines the predicted response $\hat{Y}$ by

$$\hat{Y} = r \Longleftrightarrow Y' \in]\gamma_{r-1}, \gamma_r]$$

for threshold values $-\infty =: \gamma_0 < \gamma_1 < ... < \gamma_d := \infty$. This approach can be identified with a binomial model when going over to predict the probabilities $P(Y \leq r)$ for $r \in \{1,...,d\}$. Rearranging and using the definition of the model in equation (6.2.3), it obviously holds that

$$P(Y \leq r) = P(Y' \leq \gamma_r) = P(\epsilon \leq -X\beta + \gamma_r) = F_\epsilon(\gamma_r - X\beta).$$

Assuming that the errors are logistically distributed, one has the **ordered logit model**

$$P(Y \leq r|X) = \frac{\exp(\gamma_r - X\beta)}{1+\exp(\gamma_r - X\beta)} = \operatorname{expit}(\gamma_r - X\beta) \tag{6.2.4}$$

and inverting the expit, one gets

$$\ln\left(\frac{P(Y \leq r|X)}{P(Y > r|X)}\right) = \gamma_r - X\beta.$$

With the error variance σ_ϵ^2, the likelihood is

$$L(\gamma, \sigma_\epsilon^2|Y,X) = \prod_r P(Y=r|X)^{I(Y=r)} = \prod_r [F_\epsilon(\gamma_r - X\beta) - F_\epsilon(\gamma_{r-1} - X\beta)]^{I(Y=r)}.$$

Remark 6.2.1. *By the chain rule, the score function for the logit model is given by*

$$\Lambda_\beta^{logit}(x,y) = \left(\Lambda_q^{Bin(1,q)}(y|x)\frac{e^{x\beta}x_j}{(1+e^{x\beta})^2}\right)_{j=1}^p$$

$$= \rho_\pi \left(\partial_\theta \operatorname{expit}(\theta)|_{\theta=x\beta} \Lambda_q^{Bin(1,q)}(y|x), x \right).$$

Again, x and y represent an arbitrary row of X resp. Y for the sake of simple notation. Then by theorem 3.2.1, the influence curve for the logit model is directly given.

For general cases with block partition, the link function is a mapping from $\mathbb{R}^d$ into (maybe a subset of) $\mathbb{R}^d$ and $\theta = T_\pi(x,\beta) \in \mathbb{R}^d$. Then the score, as computed in equation (2.6) of Pupashenko et al. [2015], is

$$\Lambda_\beta^{\mathcal{P}}(x,y) = \rho_\pi \left((\partial_\theta \ell(\theta))^T \Lambda_\vartheta^{\mathcal{Q}}(y|x), x \right) \tag{6.2.5}$$

since the derivative $\partial_\theta \ell$ of a $d-$dimensional function w.r.t. a $d-$dimensional variable is a $d \times d-$matrix.

We will use the notation of Pupashenko et al. to directly write down the estimation the $d-$dimensional parameter

$$\vartheta^i := \begin{pmatrix} P(Y_i = 1|X) \\ \vdots \\ P(Y_i = d|X) \end{pmatrix}$$

by an ordered logit model. Then we have

$$\begin{pmatrix} P(Y_i \leq 1|X) \\ \vdots \\ P(Y_i \leq d|X) \end{pmatrix} = \operatorname{expit} \circ^{vec} (-1) T_\pi \left(\begin{pmatrix} -1 \\ X_{i,1} \\ \vdots \\ X_{i,p} \\ \vdots \\ -1 \\ X_{i,1} \\ \vdots \\ X_{i,p} \end{pmatrix}, \begin{pmatrix} \gamma_1 \\ \beta_1 \\ \vdots \\ \beta_p \\ \vdots \\ \gamma_d \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \right), \tag{6.2.6}$$

where $\circ^{vec}$ denotes the component-wise $\circ$. We face the disadvantage that the regressor and the parameter have changed into $d(p+1)-$dimensional vectors. Therefore, we introduce the notation

$$\tilde{T}_\pi : \mathbb{R}^p \times \mathbb{R}^p \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d, \tilde{T}_\pi(X_i, \beta, 1_d, \gamma) := \begin{pmatrix} \gamma_1 - X_{i,1}\beta_1 - ... - X_{i,p}\beta_p \\ \vdots \\ \gamma_d - X_{i,1}\beta_1 - ... - X_{i,p}\beta_p \end{pmatrix}, \tag{6.2.7}$$

where 1_d denotes the vector consisting only of d ones and $\gamma := (\gamma_r)_{r=1}^d$.

We now suppress the subscript i. By (6.2.4), we see that

$$P(Y=r|X)=\text{expit}(\gamma_r-X\beta)-\text{expit}(\gamma_{r-1}-X\beta).$$

In light of (6.2.7), the estimation of ϑ can be written as

$$\vartheta=\begin{pmatrix}P(Y=1|X)\\ \vdots \\ P(Y=d|X)\end{pmatrix}:=\begin{pmatrix}p_1\\ \vdots \\ p_d\end{pmatrix}=\left(\text{expit}\circ^{vec}\tilde{T}_\pi(X,\beta,1_d,\gamma)-\text{expit}\circ^{vec}\tilde{T}_\pi(X,\beta,1_d,\tilde{\gamma})\right) \tag{6.2.8}$$

where $\tilde{\gamma}:=(\gamma_r)_{r=0}^{d-1}$.

Mimicking the score for the logit model, the $r-$th component of the score for the ordered logit model, $r=1,...,d$, is then

$$(\Lambda_p^{ordlog}(X,Y))_r=\sum_{i=1}^n\rho_\pi\left(\partial_\theta\,\text{expit}(\theta)|_{\theta=\gamma_r-X_i\beta}\Lambda_{p_r}^{Bin(1,p_r)}(Y_i|X_i),X_i\right). \tag{6.2.9}$$

Remark 6.2.2. *Again, by theorem 3.2.1, the influence curves can be computed using the scores.*

6.2.3 A proposal for the hard ranking problem

Although there already exist some work on influence curves for the AUC (LeDell et al. [2015]), there are, to the best of our knowledge, no results on influence curves for the ranking problem as stated in this work yet.

Trivially, the hard ranking task is perfectly achieved if the ordering of the instances coincides with the ordering of the predicted values. In our ordinal regression model, we want to enforce that

$$P(Y_{i:n}=i|X_{i:n})=F_\epsilon(\gamma_i-X_{i:n}\beta)-F_\epsilon(\gamma_{i-1}-X_{i:n}\beta)$$

is maximized for every i, so equivalently, we try to maximize

$$\prod_i[F_\epsilon(\gamma_i-X_{i:n}\beta)-F_\epsilon(\gamma_{i-1}-X_{i:n}\beta)]$$

which coincides with the likelihood of the ordered logit model in the case that $d=n$ and each class i is attended only by $Y_{i:n}$. Since this is just a variant of the likelihood of the ordered logit, the score of the hard ranking is computed analogously, so its $i-$th component, $i = 1, ..., n$, is

$$(\Lambda_\beta^{hardrank}(X,Y))_i = \sum_{i=1}^n \rho_\pi \left(\partial_\theta \operatorname{expit}(\theta)|_{\theta=\gamma_i - X_{i:n}\beta} \Lambda_{p_i}^{Bin(1,p_i)}(Y_{i:n}|X_{i:n}), X_{i:n} \right). \tag{6.2.10}$$

In the case of the localized ranking problem, i would just run through $\widehat{Best_K}$ instead of $\{1, ..., n\}$ and the score for the weak ranking has to be taken into account as in equation (5.2.5) (see also (6.2.13) below).

Again, given these scores, one can derive influence curves using theorem 3.2.1.

6.2.4 A proposal for weak ranking problems

We remind that the weak ranking problem with a pre-specified number K is solved if the set $Best_K$ of indices corresponding to the original K best instances coincides with its predicted counterpart $\widehat{Best_K}$. Therefore, in the notation of ordinal regression, the goal is to achieve $\hat{Y}_i \geq n - K + 1$ for every $i \in Best_K$. Using the model from the previous subsection, we can write

$$P(Y_i \geq n - K + 1|X_i) = 1 - F_\epsilon(\gamma_{n-K+1} - X_i\beta)$$

leading to the M-estimation

$$\max_\beta \left(\prod_{i \in Best_K} [1 - F_\epsilon(\gamma_{n-K+1} - X_i\beta)] \right). \tag{6.2.11}$$

Invoking the log-transformation, differentiating w.r.t. β yields the Z-equation

$$\sum_{i \in Best_K} \frac{1}{1 - F_\epsilon(\gamma_{n-K+1} - X_i\beta)} f_\epsilon(\gamma_{n-K+1} - X_i\beta) \cdot (-X_i) \overset{!}{=} 0, \tag{6.2.12}$$

and assuming that F_ϵ is the distribution function of the logistic distribution, we get the (negative) weak ranking score

$$\begin{aligned} \Lambda_\beta^{weak}(X) = \sum_{i \in Best_K} \frac{1}{1 - \frac{\exp(\gamma_{n-K+1} - X_i\beta)}{1+\exp(\gamma_{n-K+1} - X_i\beta)}} \frac{\exp(\gamma_{n-K+1} - X_i\beta)}{(1 + \exp(\gamma_{n-K+1} - X_i\beta))^2} X_i = \\ \sum_{i \in Best_K} \operatorname{expit}(\gamma_{n-K+1} - X_i\beta) X_i. \end{aligned} \tag{6.2.13}$$

Remark 6.2.3. *Of course, other choices of F_ϵ are possible which would lead to other scores and therefore to other influence functions. Block partitions can be handled analogously as before.*

Chapter 7

Elicitability: Comparing competing models

We show that the hard ranking problem is both elicitable and strongly elicitable, so there are consistent loss functions such that different models can be compared, even in the case of ties. This is vital for determining which model had performed best.

7.1 Elicitability of ranking

Fitting different forecast models leads to the question which one or which ones was resp. were the best, i.e., the forecaster is interested in the **ranking of the models** (which of course has nothing to do with the problem of ranking instances as in this work). Two situations can be distinguished:
i) A probabilistic forecast model, say, a density, is fitted;
ii) A point forecast is made, based on a set of observations.

Then the quality of the forecast has to be assessed once new observations are available.

The first situation requires some kind of distance between a density and point observations, leading to the concept of proper scoring rules like the CRPS score (Gneiting and Raftery [2007]), while the second situation needs consistent loss functions (Gneiting [2011]).

Definition 7.1.1. *Let $\mathcal{F}$ be a family of probability distributions and let A, O be some action resp. observation domain. Then a loss function $L : A \times O \to [0, \infty[$ is* ***consistent for the***

statistical functional *$T : \mathcal{F} \to A$ if*

$$\mathbb{E}_F[L(t, Y)] \leq \mathbb{E}_F[L(x, Y)] \tag{7.1.1}$$

for all $F \in \mathcal{F}$, $t \in T(F)$, $x \in A$. It is ***strictly consistent*** *if equality holds if and only if $x \in T(F)$.*

As shown in [Gneiting, 2011, Thm. 2.4], the function $S(F, Y) := L(T(F), Y)$ is a proper scoring rule if L is a consistent loss function. Both concepts have a common main property, namely that predicting the true density or the true observations provides the greatest benefit, encouraging the forecaster to make honest predictions (see Gneiting and Raftery [2007]). This motivates the importance to know whether a suitable loss function exists.

Definition 7.1.2. *A statistical functional $T : \mathcal{F} \to A$ is* ***elicitable relative to the class*** *$\mathcal{F}$ if there exists a loss function that is strictly consistent for T relative to $\mathcal{F}$.*

Trivially, each constant loss function is automatically consistent, so only the requirement of strict consistency is reasonable. It can be shown that elicitable functionals have convex level sets ([Lambert et al., 2008, Lemma 1]), but the other direction is not true since the mode functional has convex level sets without being elicitable (Heinrich [2013]). Elicitability attracted the attention of forecasters once Gneiting showed that the Conditional Value at Risk is not elicitable (see [Gneiting, 2011, Thm. 3.5]). That was the starting point for the generalization of elicitability to functionals mapping into a subset of $\mathbb{R}^k$ and calling them $k-$**elicitable** if a strictly consistent loss function exists. Among others, one of the most striking results was that the pair $(\text{VaR}, \text{CVaR})$ is $2-$elicitable despite the Conditional Value at Risk itself is not elicitable (Fissler et al. [2016], Frongillo and Kash [2015]).

We are actually interested in the question if ranking functionals are elicitable. We propose the following definition.

Definition 7.1.3. *Let $n \in \mathbb{N}$, $n \geq 2$ be fixed. We define the* ***ranking functional*** *as a mapping $T : \mathcal{F} \to \text{Perm}(1 : n)$ for any family $\mathcal{F}$ of probability distributions on the space $\mathcal{Y}^n \subset \mathbb{R}^n$ such that for $T(F) = \pi$, it holds that*

$$\pi_i < \pi_j \Longleftrightarrow P(Y_i \leq Y_j) \geq 0.5 \tag{7.1.2}$$

for $Y \sim F$.

This coincides with the setting in Clémençon et al. [2008] with the small difference that we allow Y to be discrete-valued, so the probability that some components of Y coincide is not zero. The value $T(F)$ of the ranking functional may be regarded as the **expected ranking** w.r.t. F.

Theorem 7.1.1. *Let $n \in \mathbb{N}$, $n \geq 2$ be fixed. Let $A := O := \mathrm{Perm}(1:n)$ and let $\mathcal{F}$ be any class of probability distributions on $\mathcal{Y}^n \subset \mathbb{R}^n$ with the restriction that $P(Y_i > Y_j) > 0.5$ for at least one index pair $(i,j) \in \{1,...,n\} \times \{1,...,n\}$ with $i \neq j$. Then the ranking functional $T : \mathcal{F} \to A$ is elicitable relative to $\mathcal{F}$.*

Proof. *Let $n = 2$. Let $P(Y_1 \geq Y_2) > 0.5$. Then, according to definition 7.1.3, the ranking functional takes the permutation $\pi^* = T(F) = (2,1)$. Then invoking one summand of the hard ranking loss L_n^{hard}, we have*

$$\mathbb{E}_F[I(((T(F))_1 - (T(F))_2)(Y_1 - Y_2) < 0)]$$

$$= P(((T(F))_1 - (T(F))_2)(Y_1 - Y_2) < 0) = P(Y_1 < Y_2) < 0.5$$

and analogously,

$$\mathbb{E}_F[I((\pi_1 - \pi_2)(Y_1 - Y_2) < 0)] > 0.5$$

for all $\pi \notin T(F)$ (which is just the permutation $(1,2)$).

Using the assumption that $P(Y_i > Y_j) \neq 0.5$ for at least one pair (i,j), $i \neq j$, the induction for arbitrary n is straightforward by the linearity of the expectation. So, a suitable strictly consistent loss function has been found, therefore the elicitability property is valid.

□

Remark 7.1.1. *Note that the strictly consistent loss function that leads to the elicitability of ranking is the hard ranking loss already proposed in Clémençon et al. [2008].*

Remark 7.1.2. *Clearly, we think of F in the theorem as the conditional distribution of Y given X.*

Remark 7.1.3. *Without the assumption in the last theorem, we cannot guarantee strict consistency but only consistency. This issue may occur if all components of Y are identically distributed. The assumption that the regressor matrix is designed fixed in the regression context is hence necessary but not sufficient for elicitability of the ranking functional. However, in real data analysis, the regressor matrix is unlikely to lead to an identical distribution of the responses, and discussing something like an expected ranking of identically distributed variables is also not meaningful.*

It is obvious that the finiteness of the action space beneficially affects the proof. Another advantage will be shown in the next section.

7.2 Strong elicitability of ranking

Fissler and coauthors introduced the following definitions if the best action is not unique.

Definition 7.2.1. *Let A be the action domain, O the observation domain and let $\mathcal{F}$ be a class of probability distributions. Let $\mathcal{P}(A)$ be the power set of A.*
i) *The functional $T : \mathcal{F} \to \mathcal{P}(A)$ is **weakly elicitable** if there exists a loss function $L : A \times O \to [0, \infty[$ such that*

$$\mathbb{E}_F[L(t, Y)] < \mathbb{E}_F[L(x, Y)]$$

for all $F \in \mathcal{F}$, $t \in T(F)$, $x \in A \setminus T(F)$.
ii) *The functional $T : \mathcal{F} \to \mathcal{P}(A)$ is **strongly elicitable** if there exists a loss function $L : \mathcal{P}(A) \times O \to [0, \infty[$ such that*

$$\mathbb{E}_F[L(T(F), Y)] < \mathbb{E}_F[L(x, Y)]$$

for all $F \in \mathcal{F}$, $x \in \mathcal{P}(A)$ with $x \neq T(F)$.

Acknowledgement: Tobias Fissler kindly delivered us the slides of the presentation from which we learned the definition of weak and strong elicitability.

The main difference is that weak elicitability just requires that finding one best action is possible while strong elicitability requires that the set of all best actions can be found using the loss function, not just a subset of it.

Focusing on the ranking problem, we can face the situation of multiple best actions if ties can occur with a probability greater than zero, for example when the responses are discrete-valued or when we have Bootstrap samples. We propose the following theorem for strong elicitability of ranking functionals with ties.

Theorem 7.2.1. *Let $A := O := \mathrm{Perm}(1 : n)$ for $n \geq 2$, $n \in \mathbb{N}$ be fixed. Let $\mathcal{F}$ be a class of probability distributions and let the ranking functional $T : \mathcal{F} \to A$ be elicitable relative to $\mathcal{F}$ with the loss function L which takes vales in $[0, 1]$. Then $T_P : \mathcal{F} \to \mathcal{P}(A)$ is strongly elicitable relative to $\mathcal{F}$.*

Proof. *We introduce the quantity $c(Y)$ which is the number of common entries of Y. Then set*

$$L_P : \mathcal{P}(A) \times O \to [0, \infty[, \quad L_P(B, Y) := (|B| - c(Y)!)^2 + \sum_{i=1}^{|B|} L(B_i, Y)$$

where B is the set of predicted permutations $B_i \in A$. Then strong consistency of L provides

$$\mathbb{E}_F\left[\sum_{i=1}^{|B|} L(B_i, Y)\right] > \mathbb{E}_F\left[\sum_{j=1}^{|T(F)|} L((T(F))_j), Y)\right] \tag{7.2.1}$$

for any B with $|B| = |T(F)|$. If $|B| > |T(F)|$, this part of the loss function L_P would already suffice since every summand is non-negative.

In the case that $|B| < |T(F)|$, we clearly could find B such that the inequality (7.2.1) does not hold. This issue is corrected by the first part. We have

$$\underset{|B|}{\operatorname{argmin}}(\mathbb{E}_F[(|B| - c(Y)!)^2]) = \mathbb{E}_F[c(Y)!] = |T(F)|.$$

Since the expected squared difference cannot be explicitly computed without specifying F, we establish the lower bound for the difference between these expected differences

$$\mathbb{E}_F[(|B| - c(Y)!)^2] - \mathbb{E}_F[(\mathbb{E}_F[c(Y)!] - c(Y)!)^2] \tag{7.2.2}$$

$$= |B|^2 - 2\mathbb{E}_F[|B|c(Y)!] + \mathbb{E}_F[(c(Y)!)^2] - \mathbb{E}_F[\mathbb{E}_F^2[c(Y)!]] + 2\mathbb{E}_F[c(Y)!\mathbb{E}_F[c(Y)!]] - \mathbb{E}_F[(c(Y)!)^2]$$

$$= |B|^2 - 2|B|\mathbb{E}_F[c(Y)!] + \mathbb{E}_F^2[c(Y)!] = (|B| - \mathbb{E}_F[c(Y)!])^2 = (|B| - |T(F)|)^2 \geq |T(F)| - |B|$$

for any $|B| < |T(F)|$.

Combining inequalities (7.2.2) and (7.2.1), we get for $|B| < |T(F)|$ that

$$\mathbb{E}_F[L_P(B, Y)] \geq \mathbb{E}_F\left[\sum_{i=1}^{|B|} L(B_i, Y)\right] + |T(F)| - |B|$$

$$> \mathbb{E}_F\left[\sum_{i=1}^{|B|} L(B_i, Y)\right] + \mathbb{E}_F\left[\sum_{i=|B|+1}^{|T(F)|} L((T(F))_i, Y)\right] = \mathbb{E}_F[L_P(T(F), Y)]$$

where the second inequality is true since L is $[0, 1]-$valued which proves the theorem.

□

We only considered the ranking with possible ties. Maybe there exist similar situations for which this result can be adapted.

Remark 7.2.1. *The proposed loss function highlights the necessity of the finiteness of the cartesian product of action and observation domain. Otherwise the sum would not be meaningful (besides diverging).*

Remark 7.2.2. *The very last step of the proof uses the fact that the hard ranking loss is always bounded from above by 1. The cardinality penalty would not suffice if the loss function could potentially be infinite. However, we think that loss functions on a finite set $A \times O$ should always take finite values.*

Remark 7.2.3. *A major difficulty is to avoid that a subset of the best action set is expected to be preferred over the whole best action set w.r.t. F. We only were able to correct this issue thanks to the exact knowledge of the cardinality of the actions provided that the number $c(Y)$ of common entries of Y is known. We also remark that the quadratic loss invoked for the penalty term is a strictly consistent loss function for the expectation (of the true cardinality $|T(F)|$ is this setting) although the strictness is not actually needed since it is superimposed by the strictness of the loss function L.*

Remark 7.2.4. *We restricted ourselves to distributions on the space $\mathcal{Y}^n$. Clearly, in a ranking setting with predictors, we think of conditional distributions of Y given X, but the proof would not change.*

Remark 7.2.5. *Just requiring weak elicitability in the case of ties in the responses would not reflect this issue sufficiently. A ranking rule can indeed be perfect if one of the best rankings has been predicted, but with the requirement of strong elicitability, a ranking model has to detect these ties implicitly by proposing all perfect rankings. With the results in this section and in the previous section, we are ready to reasonably compare different hard ranking models computed by competing algorithms.*

Acknowledgement: Very recently, Fissler and coauthors published their results (Fissler et al. [2019]) where the former denotation of ”weak” and ”strong” elicitability has been replaced by ”selective” resp. ”exhaustive” elicitability.

Chapter 8

Gradient Boosting for ranking problems?

Since there does not yet exist a Gradient Boosting method for the continuous ranking problem, despite there are the RankBoost and the p-Norm-Push for the binary ranking problem (Freund et al. [2003], Rudin and Schapire [2009]), we try to embed the ranking problem into the Gradient Boosting framework of Bühlmann (Bühlmann and Hothorn [2007]).

We will see that the hard ranking loss function itself does not satisfy the required regularity properties to make Gradient Boosting applicable. Motivated by standard surrogate losses from classification, we try to define some kind of pair-wise surrogate losses for the hard ranking loss and to construct three Gradient Boosting algorithms, each for another surrogate loss.

At the end, we provide heuristic arguments why Gradient Boosting algorithms of this type always suffer from the pair-wise nature of the loss functions, resulting in a very poor speed, aside from the empirically revealed fact that their ranking performance on test data is rather poor due to other issues that arise from the choice of the surrogates.

8.1 Arising problems

As introduced in equation (5.2.3), the loss function for the hard ranking problem may be rewritten as

$$\frac{1}{n(n-1)}\sum\sum_{i \neq j} L((Y_i, Y_j), f), \quad L((Y_i, Y_j), f) := I((Y_i - Y_j)(f(X_i) - f(X_j)) < 0) \quad (8.1.1)$$

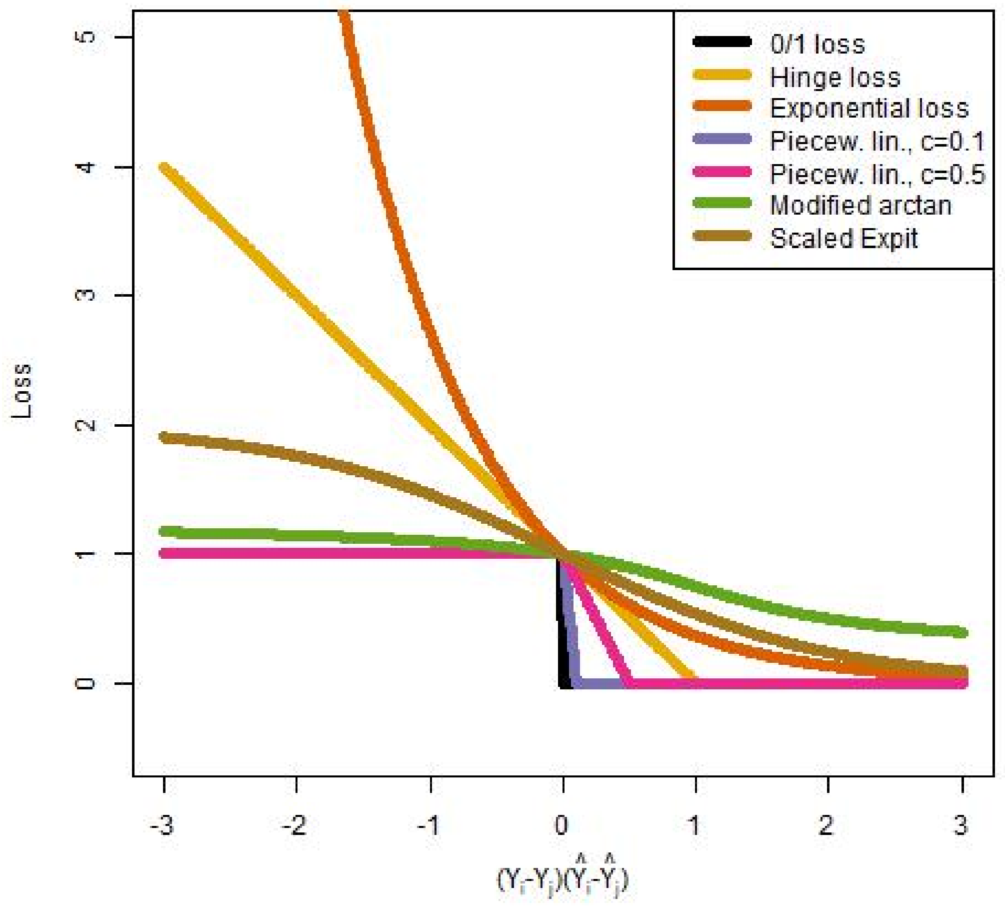


Figure 8.1: Some surrogates for the 0/1−loss

to mimic the notation of Bühlmann and Hothorn [2007]. Usually, one would compute the gradient by differentiating the loss function L with respect to f. However, as pointed out in [Bühlmann and Hothorn, 2007, Ch. 2], L has to be convex and differentiable in f which is obviously not the case. So, the hard ranking loss function is not suitable for Gradient Boosting.

In such cases, e.g. when facing the 0/1−loss in a classification setting, one commonly defines some convex surrogate loss function, for example by the exponential loss or the Hinge loss. See figure 8.1 for (convex and non-convex) surrogate losses. Invoking the exponential surrogate for the indicator function in the ranking setting has already been done in Freund et al. [2003], Rudin and Schapire [2009] which led to the RankBoost algorithm and its modification, the p-Norm-Push, which we already mentioned before. However, their algorithms (see the two former references or the survey Clémençon et al. [2013b]) are tailored to binary classification problems.

8.2 An exponential surrogate

The question that arises from the nature of the hard ranking loss function as double sum of indicator functions is if the approximation of the indicator function by suitable surrogates can lead to a Gradient Boosting procedure for the hard ranking problem. In this section, we replace the indicator functions by exponential functions. To write down the empirical risk in the form of Bühlmann and Hothorn [2007], we define

$$L_i^{exp}(Y,f) := \frac{1}{n-1}\sum_{j\neq i}\exp(-(Y_i-Y_j)(f(X_i)-f(X_j))) \tag{8.2.1}$$

so that the empirical loss is the empirical mean of all L_i over i.

Remark 8.2.1. *It does not matter if we include the factor $(n-1)^{-1}$ in the single losses L_i or not when searching the best predictor. But it scales the gradients and it seems much more reasonable to include the factor to mimic Bühlmann and Hothorn [2007] as best as possible.*

We also made experiments where we only used the exponential term, but as anticipated, the gradients exploded for certain surrogates, leading to a complete breakdown of the algorithm after even one step.

The most natural way to define the required gradients (see section A.8) is to set

$$U_i^{exp} := -\partial_f L_i^{exp}(Y,f)|_{f=\hat{f}^{(m)}} = \frac{1}{n-1}\sum_{j\neq i}(Y_i-Y_j)\exp(-(Y_i-Y_j)(\hat{f}^{(m)}(X_i-X_j))) \tag{8.2.2}$$

provided that f is a linear function, so that $f(X_i-X_j)$ equals $f(X_i)-f(X_j)$. This resembles the usual Gradient Boosting, the only difference is that all components of Y and $f(X)$ have to enter the gradient in our case.

Due to the sum structure of the losses, even each of the L_i inherits the combinatorial nature of the ranking problem, therefore we cannot determine an exact minimizer, i.e., a suitable baselearner which is tailored to the exponential surrogate. Algorithmically, we proceed as in the component-wise L_2-Boosting where a regression w.r.t. every single predictor, i.e., a simple regression, is performed and the predictor that decreased the RSS most is taken. The difference in our case is that we evaluate the exponential loss instead of the RSS and that we have to evaluate the gradient (8.2.2) instead of the least squares residual. Additionally, we are not able do define an offset as in other Boosting methods. The offset is an empirical population minimizer and usually the mean or the median of the responses in the case of L_2-

or L_1-Boosting which does clearly not make any sense in the ranking case since a constant always leads to a perfect zero ranking loss. Therefore, we decided to start which the original values of Y in the first step. Just for internal reference, we call the proposed algorithm **ExpBoost**.

```
Initialization: Data $(X,Y)$, step size $\kappa \in ]0,1]$, number $m_{iter}$ of iterations and
 parameter vector $\hat{\beta}^{(0)} = 0_{p+1}$;
Set $r^{(0)} := Y$;
for $k = 1, ..., m_{iter}$ do
    for $j = 1, ..., p$ do
        Fit the current gradients $r^{(k-1)}$ by a simple least squares regression model using
         the predictor variable $j$;
        Compute the exponential loss;
    end
    Take the variable $\hat{j}_k$ whose simple model $\hat{\beta}_{\hat{j}_k} \in \mathbb{R}^{p+1}$ provides the smallest
     exponential loss;
    Update the model via $\hat{\beta}^{(k)} = \hat{\beta}^{(k-1)} + \kappa\hat{\beta}_{\hat{j}_k}$;
    Compute the current gradients $r^{(k)} = (-\partial_f L_i^{exp}(Y,f)|_{f=\hat{\beta}^{(k)}})_{i=1}^n$ as in (8.2.2)
end
```

Algorithm 2: ExpBoost with component-wise linear baselearners

Experiments show that, as expected, it performs rather poor. The reason is the highly non-robust exponential loss function which is a really bad approximant of the indicator function for small values since in contrast to classification, its argument takes values in $\mathbb{R}$ instead of $\{\pm 1\}$. Furthermore, the function does not take a minimum in $\mathbb{R}$.

Surprisingly, the algorithm can even break down after one single iteration on a moderate data set! We also tried an expit transformation of the response values before applying the algorithm or a further scaling of the gradients by n^{-1}, but after some iterations, the gradients also diverged. Evidently, the exponential loss is definitely not appropriate for the continuous ranking problem.

Remark 8.2.2. *This is no contradiction to the RankBoost algorithm since RankBoost proceeds in the same manner as AdaBoost, so the iterations compute weights assigned to the observations (more precisely, a distribution on the pairs of the observations is updated). The gradients are not evaluated explicitly.*

8.3 A Hinge surrogate

Next, we try to use the Hinge loss that at most grows linearly for decreasing values. In the same manner as before, we propose

$$L_i^{hinge}(Y,f) := \frac{1}{n-1}\sum_{j\neq i}\max(0, 1-(Y_i-Y_j)(f(X_i)-f(X_j))). \tag{8.3.1}$$

The gradient is given by

$$U_i^{hinge} := -\partial_f L_i^{hinge}(Y,f)|_{f=\hat{f}^{(m)}} = \frac{1}{n-1}\sum_{j\neq i}\begin{cases} Y_i-Y_j, & (Y_i-Y_j)(\hat{f}^{(m)}(X_i-X_j))<1 \\ 0, & \text{otherwise}\end{cases}. \tag{8.3.2}$$

We rely on the argument of [Bühlmann and Hothorn, 2007, Ch. 3.2] for the L_1-loss that the region of nondifferentiability of the loss is just a single point that has zero probability which indeed holds for the Hinge loss.

However, the running time of this algorithm that we refer to as **HingeBoost** is still very high due to the pair-wise structure of the single gradients and losses. Compared to the L_2-Boosting with component-wise linear baselearners, it is absurdly slow.

8.4 A piece-wise linear surrogate

We propose the much tighter surrogate

$$L_i^c(Y,f) := \frac{1}{n-1}\sum_{j\neq i}\begin{cases} 1, & (Y_i-Y_j)f(X_i-X_j)<0 \\ 1-\frac{(Y_i-Y_j)f(X_i-X_j)}{c}, & 0\leq (Y_i-Y_j)f(X_i-X_j)\leq c \\ 0, & (Y_i-Y_j)f(X_i-X_j)>c\end{cases},$$

so we approximate the indicator function in a piece-wise linear manner, for $c\in]0,1]$ (cf. figure 8.1). The probability of the product of the differences being located at 0 or c is zero, so the differentiability is still valid almost-everywhere. Although we violate the convexity assumption, this loss function is still piece-wise convex.

The gradient is

$$U_i^c = -\partial_f L_i^c(Y,f)|_{f=\hat{f}^{(m)}} = \frac{1}{n-1}\sum_{j\neq i}\frac{Y_i-Y_j}{c}I((Y_i-Y_j)\hat{f}^{(m)}(X_i-X_j)\in]0,c[).$$

Besides its very slow running time, this algorithm is completely worthless due to an astounding issue. In each iteration, the algorithm selects the variable which is exactly the one whose column ranking of the data matrix already is at most comparable with either the response or the negative response, i.e., it selects predictor variable j_0 with

$$j_0 = \underset{j}{\operatorname{argmin}}(\min(L_n^{hard}(Y, X_{\cdot,j}), L_n^{hard}(-Y, X_{\cdot,j}))).$$

The coefficient is positive or negative, depending on the fact if the column ranking resembles the ranking of the response vector or of the negative response vector most, respectively (which is clearly inherited from the fact that the simple regression slope estimator is a standardized covariance between the response and the respective column in the regressor matrix). It does not necessarily happen, but one issue that sometimes occurs is that this variable is again chosen in the subsequent iterations, and in this case, the whole Boosting model just contains one variable (and the intercept), so its performance on test data is likely to be very bad. Taking a closer look on it, we clearly see that the gradients that are computed are likely to be zero. Only in the small region $]0, c[$, we get a non-zero gradient. This is the reason why the algorithm sometimes ends up in just having selected one single variable.

Remark 8.4.1 (**Comparison with L_2-Boosting**). *Maybe it looks irritating that this algorithm does not perform well since L_2-Boosting selects in each step the variable which is most correlated with the current residual. If there was a column whose ranking was exactly the same as the ranking of the residual column, L_2-Boosting would clearly select this column. But one has to keep in mind that concordance and correlation are only identical in the case of perfect concordance or perfect discordance and that L_2-Boosting and our proposed Boosting algorithm proceed along very different gradients.*

We will discuss the variable selection procedure of L_2-Boosting later again in terms of correlation of the predictor columns with the current residual vector (see remark 10.4.4)

8.5 Could we speed it up?

One may ask if we can accelerate the evaluation of the Hinge loss in the HingeBoost algorithm benefitting from the constant part. An idea could be to find a partial ordering of the products of the pair-wise differences. If for some product $f(X_i - X_j)(Y_i - Y_j)$ the Hinge loss is zero, then for all other products of pair-wise differences that are larger, the Hinge loss is also zero. There exists work where such partial ordered sets, so-called ”posets”, are treated, see e.g. Daskalakis et al. [2011] and references therein, but sorting them cannot be achieved with a lower complexity than $\mathcal{O}(n\ln(n))$, to the best of our knowledge.

We can deduct that there will always be a loss in performance w.r.t. the L_2- or L_1-Boosting when having pair-wise loss functions. Since the algorithms are nearly identical, we only need to concern the cost of evaluating the loss function in and evaluating the gradient at the end of each iteration. The cost is of order $\mathcal{O}(pn)$ for Gradient Boosting with standard losses in each Boosting iteration. In the way we implemented ExpBoost and HingeBoost, it is $\mathcal{O}(pn^2)$. Even if one could invoke fast sorting algorithms for our case, the cost would be a least of order $\mathcal{O}(pn\ln(n))$. That does not look bad, but this would be an absolutely idealistic situation.

Assume that we have a poset P of the products of pair-wise differences as above, in an ascending order. In fact, the poset contains all quadruples (X_i, X_j, Y_i, Y_j). To benefit from the idea, we would need to determine the element P_{i_0} of the poset which leads to the smallest product of pair-wise differences that exceeds one, so we would need to find this element in a set of cardinality $\binom{n}{2}$. Furthermore, the usage of a partial order in this HingeBoost setting is only meaningful in the area in which the loss function is constant. But for the Hinge loss, we still have to evaluate the loss for any element P_k, $k < i_0$, of the poset.

Remark 8.5.1. *Therefore, we expect that no Gradient Boosting algorithm with a pair-wise loss function could be competitive with Gradient Boosting with standard losses in terms of time consumption.*

8.6 Conclusion

We can conclude that a Gradient Boosting with a pair-wise loss to solve the continuous ranking problem is not feasible.

The first great issue is the choice of a suitable loss function. Loss functions that are too ”loose” like the exponential or the Hinge loss do not lead to acceptable ranking performances. Experiments have revealed that the Boosting models from ExpBoost or HingeBoost do not imply small ranking losses.

The $0/1-$loss itself or a very close surrogate lead to models that are not reasonable since only one predictor is likely to be selected.

Robust regular losses as for example a sign-reversed, shifted and tied arcus tangens or a sigmoid function (see figure 8.1) are more complicated than the piece-wise linear loss (though not even convex), so the time consumption would be even higher. Moreover, those loss functions are also nearly flat outside a neighborhood of zero, a due to the fact the we have real-valued and not $\{\pm 1\}-$valued responses, the product of pair-wise differences is likely to take values in that region, so there will be problems when trying to fit the new negative gradients. On the other hand, we must not allow the loss function to be unbounded.

However, if we had such pair-wise loss functions, each iteration of Boosting required p times the computation of this loss function that needs $\mathcal{O}(n^2)$ operations, and at the end, the evaluation of the pair-wise gradient whose computation time is also of order $\mathcal{O}(n^2)$ has to be performed. This indeed is the second great issue of Gradient Boosting with pair-wise losses. The desirable case to have a data set with many observations turns out to be clearly disadvantageous for the running time.

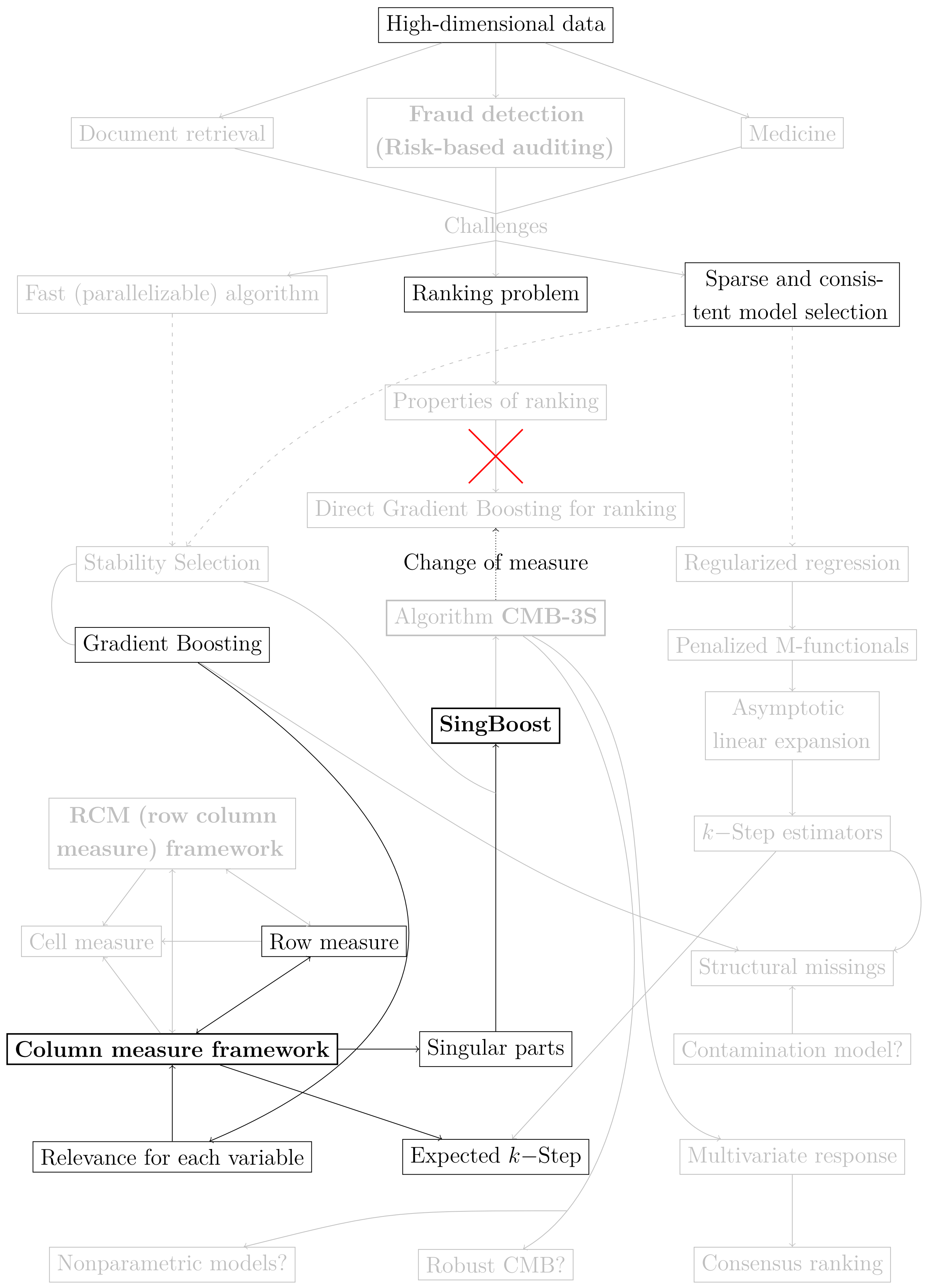
Even some sort of subgradient descent (see e.g. Shor [2012]) is not possible due to the non-convexity of the ranking losses.

These issues motivate us to solve the following problem:

Is there another, more efficient way to find a Gradient Boosting procedure that solves an approximated problem tailored to the ranking problem?

Part III

The row measure and the column measure on a data matrix


High-dimensional data
Document retrieval
Fraud detection (Risk-based auditing)
Medicine
Challenges
Fast (parallelizable) algorithm
Ranking problem
Sparse and consistent model selection
Properties of ranking
Direct Gradient Boosting for ranking
Change of measure
Stability Selection
Regularized regression
Algorithm CMB-3S
Gradient Boosting
Penalized M-functionals
Asymptotic linear expansion
SingBoost
RCM (row column measure) framework
k−Step estimators
Cell measure
Row measure
Structural missings
Column measure framework
Singular parts
Contamination model?
Relevance for each variable
Expected k−Step
Multivariate response
Nonparametric models?
Robust CMB?
Consensus ranking

In this conceptual part, we formalize resampling and model selection strategies in the language of measures.

We identify resampling procedures that essentially assign weights to each row of a data matrix with the concept of a "row measure". Based on this definition, we define a similar measure which we refer to as "column measure" to describe variable selection, i.e., column selection, w.r.t. some loss function L as a realization of a such a measure.

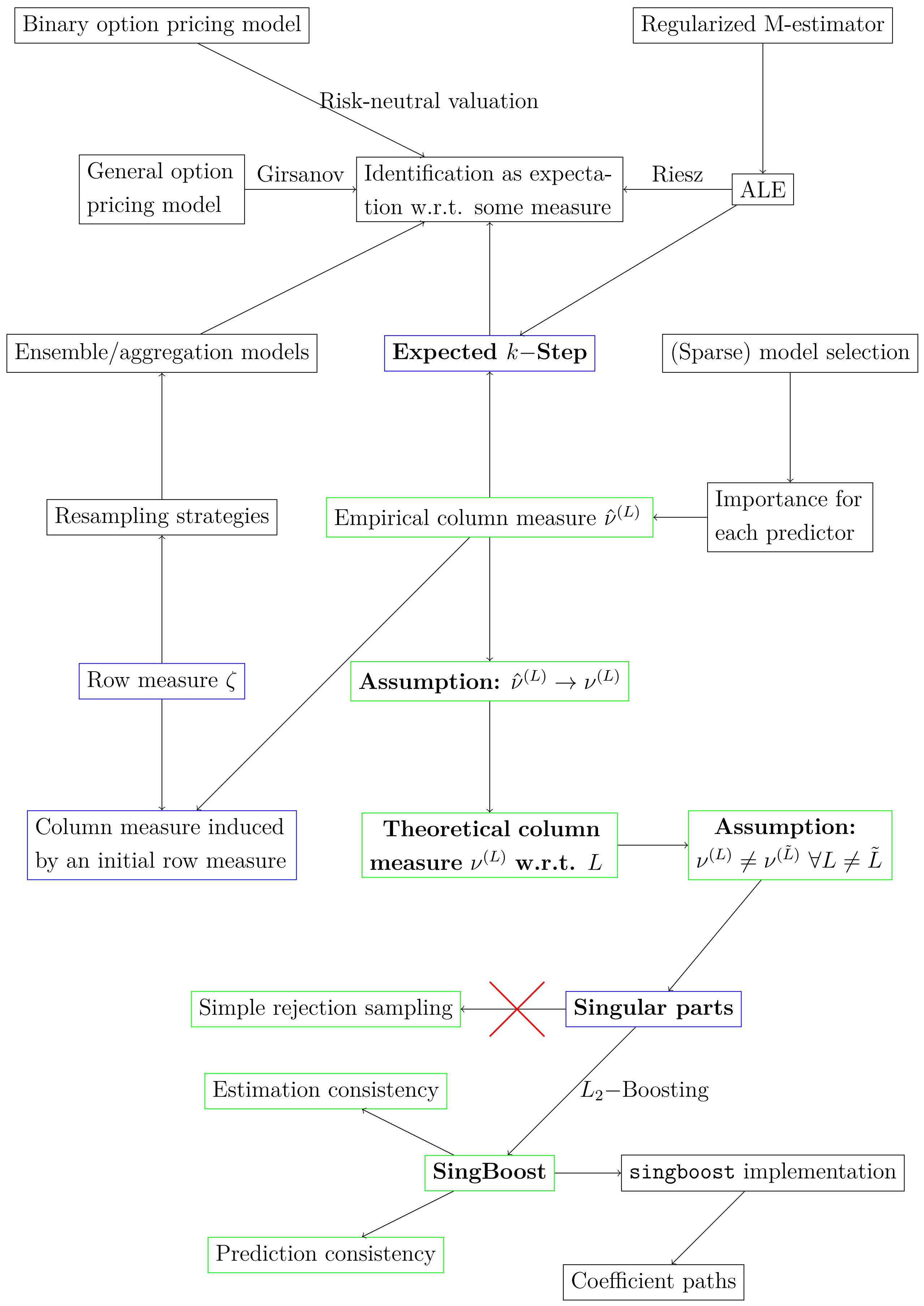
Since the true column measure w.r.t. some loss function is not known, we will make key assumptions for the rest of the thesis on its nature and of approximating properties of suitable model selection algorithms.

The column measure defines a whole framework in which all learning procedures that somehow invoke columns of a data matrix can be embedded. This framework can be extended to the case of time-dependent column measures. Moreover, we connect $k-$Step estimators with the empirical column measure and define a new estimator, the Expected $k-$Step.

Another key assumption that real applications directly motivate is that true column measures for different loss functions are not identical. This shows that a simple rejection sampling strategy where optimization of a complicated loss function $\tilde{L}$ should be replaced by the optimization of a simple loss function in the sense of comparing different solutions by their performance w.r.t. $\tilde{L}$ is not recommended and potentially leads to mis-specified models. In fact, it becomes inevitable to perform some kind of rejection step already during the Boosting algorithm.

We close this part with an extension of Gradient Boosting to cases where either the gradients are computationally very intensive or where they do not even exist. We introduce a "gradient-free Gradient Boosting" algorithm which we call SingBoost which is a first approach to combine L_2-Boosting with rejection sampling w.r.t. $\tilde{L}$, respecting possible singular parts. We can show that under a Corr-min condition that ensures that each chosen variable is sufficiently correlated with the current residual, SingBoost enjoys estimation and prediction consistency properties.

Further extensions of this algorithm concerning stable model selection, the computation of final coefficients w.r.t. a stable model and especially approaches to approximate the column measure w.r.t. $\tilde{L}$ are discussed in the next part.


Binary option pricing model
Regularized M-estimator
Risk-neutral valuation
General option pricing model
Girsanov
Identification as expectation w.r.t. some measure
Riesz
ALE
Ensemble/aggregation models
Expected $k-$Step
(Sparse) model selection
Resampling strategies
Empirical column measure $\hat{\nu}^{(L)}$
Importance for each predictor
Row measure ζ
Assumption: $\hat{\nu}^{(L)} \to \nu^{(L)}$
Column measure induced by an initial row measure
Theoretical column measure $\nu^{(L)}$ **w.r.t.** L
Assumption: $\nu^{(L)} \neq \nu^{(\tilde{L})}\ \forall L \neq \tilde{L}$
Simple rejection sampling
Singular parts
Estimation consistency
L_2-Boosting
SingBoost
`singboost` implementation
Prediction consistency
Coefficient paths

Chapter 9

The row measure and the column measure

We already have seen that suitable model selection is a crucial concept in machine learning. Regardless of the regularization procedure, the final model is always based on a "hard" decision whether a certain variable enters it or not. One may ask if there is any possibility to loose these constraints. This rather conceptual part is organized as follows.

After proving a theoretical result that identifies regularized M-estimation with integration of the regularized score function Z_n^{λ} w.r.t some vector measure, we describe a simple rejection sampling strategy and point out the difficulties that arise when trying to apply Gradient Boosting to ranking problems in the spirit of such a simple rejection sampling.

We see that aggregation procedures that are based on subsampled or bootstrapped data essentially depend on some row measure through the selected samples. This can be identified with usual weights that are given to rows when resampling. The question whether a similar concept may be thought of for the columns leads to the definition of a measure that indicates the relevance of each predictor variable which we will call "column measure".

We will identify all existing learning procedures which incorporate columns of a data matrix, including Online Learning algorithms, as limiting case of our column measure framework and restate some popular model selection strategies in the language of the column measure.

The main theoretical contribution will be the introduction of the "Expected One-Step estimator" which is proven to be a natural aggregated form of usual One-Step estimators when fitting models invoking an empirical column measure.

9.1 The induction of randomness: Option pricing with the binary model

Model selection is a crucial concept in this thesis. However, model selection is done by optimization and hence there is no randomness. But at the second glance, we can indeed treat the model selection afterwards as it had been randomly, especially in Boosting models. This will be discussed in the sequel. For now, we recapitulate the binary option pricing model where the stock movement and therefore the fair option price can also be written as an expectation w.r.t. some suitable measure to illustrate this principle.

Consider the well-known one-period binary model for the determination of a fair option price. One has the current stock price S_0 and assumes that the price can either rise to S_u or decrease to S_d in the next time step. One strategy is to buy x stocks and to save an amount of y, where the other strategy is to buy a call option on the respective stock. Since there should not be any arbitrage opportunity, the call price C_0 has to be determined to be ”fair”, i.e., the payoff after the first period has to be the same for both strategies, regardless in which direction the stock price will move. Denoting the value of the call by V_u resp. V_d, the equations

$$xS_u + y = V_u, \quad xS_d + y = V_d$$

must hold. Solving this system, one gets

$$x = \frac{V_u - V_d}{S_u - S_d}, \quad y = \frac{S_u V_d - S_d V_u}{S_u - S_d}.$$

Since the first strategy requires an amount of $xS_0 + y$ of money, the fair call price C_0 must equal this quantity. Inserting the solutions for x and y, one gets

$$C_0 = \frac{V_u(S_0 - S_d) + V_d(S_u - S_0)}{S_u - S_d}.$$

But when defining

$$q := \frac{S_0 - S_d}{S_u - S_d},$$

the call price C_0 can be rewritten as the expectation $\mathbb{E}_Q[V]$ w.r.t. the Bernoulli measure Q corresponding to q, where V is the random variable modelling the value of the call. For further details, see any reference on financial mathematics or option pricing, e.g. Björk [2009].

Much more sophisticated techniques concerning a risk-neutral measure invoke the Girsanov theorem ([Björk, 2009, Thm. 11.3]) that allows a change of measure from the real to the risk-neutral measure by a Radon-Nikodym derivative. Special cases arise in incomplete markets where the derivative is irreplicable by other underlyings. Standard examples are weather or catastrophy indices. To compute the price of those objects, Girsanov's or related theorems are applied to derive a risk-neutral measure (see e.g. Alexandridis and Zapranis [2013], Härdle et al. [2012], Cabrera et al. [2013]).

Why do we refer to option pricing in this thesis? The key idea, namely to identify some quantity as an expectation value w.r.t. a certain measure, will be applied when performing model selection with Boosting-type algorithms, leading to a measure on the columns of the regressor matrix.

9.2 A simple observation

The prediction step in a linear regression model is simply

$$\hat{Y} = X\hat{\beta}_n$$

for the estimated parameter $\hat{\beta}_n$. Assume that we have an asymptotically linear estimator (3.2.4). Then it holds that

$$\hat{Y} = X\hat{\beta}_n + \frac{1}{n}\sum_i X\psi_{\hat{\beta}_n}(X_i) \tag{9.2.1}$$

with influence curve $\psi_{\hat{\beta}_n}$ w.r.t. the model corresponding to $\hat{\beta}_n$. For the size n of the training data tending to infinity, the prediction asymptotically tends to

$$\hat{Y} = X\hat{\beta}_n + \int_{\mathcal{X}} x\psi_{\hat{\beta}_n}(x)dF_X(x)$$

where the distribution from which the predictors arise is denoted by F_X as in the previous parts. So, taking a closer look on this expression, the prediction reveals itself to be essentially an expectation value.

In the following subsection, we present a result that goes into a different direction by using a Riesz representation theorem.

9.3 A Riesz representation result

We now try to apply an appropriate Riesz representation theorem to identify the regularized M- and Z-functionals from section 4.4 with certain vector measures. An absolutely necessary requirement for all Riesz-type theorems is the linearity of the functional. In fact, asymptotically linear M-functionals are not linear themselves due to the summand θ_0, but the "centered" or "shifted" version $\check{T} := T - \theta_0$ is linear as the following theorem shows.

Lemma 9.3.1. *Let Θ be a normed real vector space, let $T : \mathcal{C}^p(\Theta) \to \Theta$ and define*

$$\mathcal{Z} := \{Z_n((X_1,Y_1),...,(X_n,Y_n),\cdot) \mid n \in \mathbb{N}, (X_1,Y_1),...,(X_n,Y_n) \overset{i.id.}{\sim} F\}.$$

Let $S_n = T \circ Z_n$ be an asymptotically linear (regularized) M-estimator. Then $\check{T} := T - \theta_0$ is linear on the restriction of $\mathcal{C}^p(\Theta)$ on the set $\mathcal{Z}$ for fixed $\varphi : \mathbb{R}^p \times \mathbb{R} \times \Theta \to \mathbb{R}^p$ and Z_n in the sense of (4.3.1).

Proof. *The asymptotic linear expansion of $\check{T}$ restricted on $\mathcal{Z}$ is given by*

$$\check{T}(Z_n) = \frac{1}{n}\sum_{i=1}^{n} \psi_{\theta_0}(X_i,Y_i) + o_{P^n_{\theta_0}}(n^{-1/2})$$

for any $Z_n \in \mathcal{Z}$. Additionally, by restriction on $\mathcal{Z}$, we ensure that the limiting function of each element of this family for n tending to infinity equals $\eta = \mathbb{E}_F[\varphi((X,Y),\cdot)]$, so the optimal parameter $\theta_0 = T\eta$ and therefore the shifting does not change over functions in $\mathcal{Z}$. Then the equation $\check{T}(af+bg) = a\check{T}f + b\check{T}g$ is true for any $f,g \in \mathcal{Z}$ and $a,b \in \mathbb{R}$ since the ψ_{θ_0} is an influence curve, so in fact a special Gâteaux derivative which itself is just a special $\mathcal{R}-$derivative. By definition 3.1.1, the map $d_{\mathcal{R}}T : \mathcal{C}^p(\Theta) \to \Theta$ is linear in the direction of the derivation, hence $\check{T}$ is linear since the $o-$classes are of course linear.

□

The restriction on the family of the empirical counterparts Z_n of η for a fixed score φ is no problem since we want to find the measure corresponding to the regularized risk corresponding to a certain structural risk minimization problem, so combining different loss or penalty functions would not be reasonable for our purposes.

Remark 9.3.1. *We mention that in addition to the vital linearity, one-dimensional Riesz representation theorems sometimes require positivity (definition A.5.1) or boundedness of the functional.*

Even for functionals that are not positive, there exist results by invoking a minimal decomposition ("Minimalzerlegung", see [Elstrodt, 2006, Satz 2.25]) of the functional that corresponds to a Jordan decomposition of the related signed measure (definition A.3.1). We do not need this positivity for our work, but in the case of asymptotically linear estimators, one could always separately handle the positive and negative parts of the influence curves.

Boundedness is not required for this work, too, but since we restrict ourselves to compacts $\Theta \subset\subset \mathbb{R}^p$, the boundedness of T resp. $\check{T}$ mapping onto Θ is always valid.

The main assertion of this subsection is the following theorem.

Theorem 9.3.1. *With the notation as before, let $\mathcal{Z}^\lambda$ be the analogue to $\mathcal{Z}$ from lemma 9.3.1 where a fixed penalty term $J_\lambda : \Theta \to \mathbb{R}$ is invoked with $L \not\equiv 0 \not\equiv J$. Let $S_n^\lambda = T \circ Z_n^\lambda$ be an asymptotically linear regularized M-estimator for $Z_n^\lambda \in \mathcal{Z}^\lambda$. Then there exists a unique, weakly regular set function $\mu : \mathbb{B}^p \cap \Theta \to \mathcal{C}_f(\mathbb{R}^p, \mathbb{R}^p)$ (see also definitions A.3.2, A.3.4 A.4.3) with compact support such that*

$$S_n^\lambda - \theta_0 = \check{T}(Z_n^\lambda) = T(Z_n^\lambda) - \theta_0 = \int_\Theta Z_n^\lambda d\mu = \int_\Theta \left[\frac{1}{n}\sum_{i=1}^n \varphi(X_i, Y_i, \theta) + J_\lambda'(\theta)\right] d\mu \tag{9.3.1}$$

for any $Z_n^\lambda \in \mathcal{Z}^\lambda$.

Proof. *Trivially, $Z := \Theta$ is a locally compact Hausdorff space (since it is non-empty by assumption (A1), see definition A.4.1). Then Z is an S-space (see e.g. [Reiter, 1972, Ex. 1.2+Thm. 2.1]). W.l.o.g., we set $E := F := \mathbb{R}^p$, so E and F are Banach spaces.*

The shifted functional $\check{T} : \mathcal{C}(Z, E) \to F$ is linear on $\mathcal{Z}^\lambda$ by lemma 9.3.1 since it corresponds to an asymptotically linear regularized M-estimator. Moreover, Θ is compact, hence $\check{T}$ is bounded (see also remark 9.3.1), so $||\check{T}f||_F < \infty$ for any $f \in \mathcal{Z}^\lambda$. Thus, there exists a constant $M > 0$ and a compact $K \subset\subset E$ providing $||\check{T}f||_F \leq M||f||_K$ for any $f \in \mathcal{Z}^\lambda$. Grace to the correspondence of the family $\mathcal{Z}^\lambda$ to a penalized loss function, we conclude that $0 \notin \mathcal{Z}^\lambda$, otherwise the sum of loss and penalty would have to be perfectly flat, so by the assumption that $J(0) = L(0)$ and the non-negativity, they would both equal zero everywhere which definitely does not make sense and has been excluded by assumption. Hence, $\check{T}$ is continuous with respect to the compact-open topology on $\mathcal{C}(Z, E)$ (see definition A.4.2).

Hence, theorem A.5.2 provides such a representing set function $\mu : \mathbb{B}^p \to \mathcal{C}_f(E, F'')$. Since the dimension p is finite, the space $\mathbb{R}^p$ is reflexive (see e.g. [Werner, 2006, Ch. III.3]), so $F'' = F = \mathbb{R}^p$ and the theorem is proven.

□

Remark 9.3.2. *We have seen in the proof of the last theorem that the zero function could be an issue. We argued that this case does not make sense. Alternatively, one could define that the M-estimator for an empirical Z-function $Z_n \equiv 0$ provides a coefficient vector that only contains zeroes itself. Then the continuity is still valid.*

Theorem 9.3.1 theoretically replaces an optimization step by an integration step over the whole parameter space. Nicely, the integrand is just the empirical regularized Z-function but unfortunately, we cannot explicitly get the representing measure μ which differs for different function classes $\mathcal{Z}^\lambda$, i.e., for different losses and penalties. But, the heuristic approaches in this subsection motivate that an ALE can indeed be identified with an expectation w.r.t. a certain measure.

In the subsequent sections, we will provide another measure which is directly connected to model selection and which provides a framework for all existing model selection techniques.

9.4 Rejection sampling

Consider the easiest kind of a rejection sampling situation: One has a density f which is known and which has a bounded support, but drawing samples according to it is difficult. If $\sup_{x \in \mathrm{supp}(f)}(f(x)) = M < \infty$, then one can draw samples x and u from $X \sim U(\mathrm{supp}(f))$ and $U \sim U([0, M])$. The realization x is accepted if $u \leq f(x)$, otherwise the sample is rejected. Since the area under f has the size 1, the acceptance rate, i.e., the expected rate of accepted realizations, is obviously $(M|\mathrm{supp}(f)|)^{-1}$.

A more sophisticated strategy to increase the acceptance rate, especially if the density f is highly non-uniform, is to find a suitable density g with $\mathrm{supp}(g) \supset \mathrm{supp}(f)$ and from whose cumulative distribution function one can easily draw samples. If again $\sup_{x \in \mathrm{supp}(f)}(f(x)) < \infty$, then there exists $c > 1$ such that $f(x) \leq cg(x)$ for all $x \in \mathrm{supp}(f)$. In fact, f is dominated by cg. The rejection sampling strategy manifests itself in sampling a pair (z, v) where z is a realization of $Z \sim U([0, 1])$ and v is a realization of $V \sim G$ with the cumulative distribution function G corresponding to g. The realization x is accepted if

$$z \leq \frac{f(v)}{cg(v)}.$$

The acceptance rate in this case is c^{-1}.

Remark 9.4.1. *Note that in both cases, although we do not directly sample according to it, the (complicated) density f at least has to be evaluated.*

Example 9.4.1 (Best Subset Selection). *Certain variable selection procedures may also be identified with a rejection sampling. The naïve brute-force best subset selection (see p.e. Friedman et al. [2001]) indeed proposes all 2^p possible subsets of predictors and chooses the best one by evaluating a complexity-penalized goodness function like the Akaike Information Criterion or the Bayes Information Criterion. Therefore, from another point of view, the* *best subset selection can be seen as a rejection sampling with deterministic acceptance ratio 2^{-p} which is very poor.*

Inspired by this reasoning, a naïve rejection sampling strategy to solve the hard continuous ranking problem can be described by the following algorithm:

```
Initialization: Data $\mathcal{D}$, step size $\kappa \in ]0,1]$, number $m_{iter}$ of iterations, number $B$ of
  subsamples, size $n_{train}$ of the training sets;
for $b = 1, ..., B$ do
    Draw a subsample $\mathcal{D}^{(b,train)}$ of size $n_{train}$ from the data. The non-selected rows
      form the training set $\mathcal{D}^{(b,test)}$;
    Perform $L_2-$Boosting with parameters $m_{iter}$ and $\kappa$ on $\mathcal{D}^{(b,train)}$;
    Evaluate the model on $\mathcal{D}^{(b,test)}$ by computing the hard ranking loss as given in
      (5.2.3);
end
Take the model whose empirical ranking loss was minimal;
```

Algorithm 3: A naïve rejection sampling strategy for the hard ranking problem

Remark 9.4.2. *By replacing the hard ranking loss function by either the weak or the localized ranking loss function, one would get a similar strategy for solving them relying on L_2-Boosting. In fact, any loss function $\tilde{L}$ would be allowed for evaluation. Obviously, the acceptance rate is B^{-1}.*

Remark 9.4.3. *Obviously, one can also invoke a Stability Selection and take the model whose out-of-sample ranking loss based on a stable predictor set is minimal.*

Remark 9.4.4. *Of course, this method is trivial and leads to justified criticism. The ranking loss itself is not optimized in any sense and it is hard to decide whether the best L_2-Boosting solution is "near" to the optimal linear model that really would minimize the ranking loss. The only aspect that encourages this strategy is that the regression function $\mathbb{E}[Y|X = x]$ that is approximated by L_2-Boosting has been shown to be an optimal scoring rule for the continuous ranking problem ([Clémençon and Achab, 2017, Prop. 1]).*

Remark 9.4.5. *One aspect that we want to highlight is that subsampling (and also Bootstrapping) of rows of a data set in combination with a model selection procedure* ***essentially randomizes column selection*** *to some extent. More precisely, the immanent randomness of sampling rows is transferred to the selection of columns through the model selection algorithm.*

9.5 The row measure

When sampling rows of data sets, there exists a vast variety of sampling procedures, for example when up- or downsampling in imbalanced data sets (for a survey, see e.g. More [2016]). This can be identified with some kind of **row measure** that is sampled from.

Definition 9.5.1. *Suppose we have n observations $X_1, ..., X_n \in \mathcal{X}$ where $\mathcal{X}$ is some space. W.l.o.g., we think of $X := (X_1, ..., X_n)$ as an $(n \times 1)-$dimensional column vector or, in the case of multivariate observations, as matrix with n rows. Then we define the **row measure** as the map*

$$\zeta : (\{1, ..., n\}, \mathcal{P}(\{1, ..., n\})) \to ([0, 1], \mathbb{B} \cap [0, 1]),$$

$$\zeta : \{i\} \mapsto \zeta(\{i\}) =: \zeta_i \in [0, 1] \ \forall i \in \{1, ..., n\}$$

which assigns a weight to each row.

In other words, special sampling techniques replace the uniform distribution on the row indices, i.e., the **uniform row measure** to some other measure which is tailored to those data sets.

Example 9.5.1. *For example, imbalanced data, i.e., classification data where the relative class occurences in the training data are far away from being considered as being equal, lead to the problem of giving too low attention to the classes which are underrepresented. Therefore, one considers up- and downsampling approaches where one takes large Bootstrap samples from the underrepresented classes resp. only takes subsamples from the over-represented classes in order to force nearly equal relative frequencies. In each class, one may still perform sampling from a uniform row measure, but from the view of the whole data, the resulting complete resample can be seen as a realization from a non-uniform row measure. More sophisticated approaches like over-sampling from the underrepresented classes may also be identified with some empirical row measure, fitted implicitly by learning algorithms, see SMOTE (Chawla et al. [2002]) and extensions (Elrahman and Abraham [2013] and references therein).*

Example 9.5.2. *A natural algorithm based on non-uniform row measures is the IRWLS algorithm which is well-known for fitting generalized linear models. Furthermore, the IRWLS algorithm is used to fit robust M- and S-estimators (see Maronna et al. [2006]) whereas for example a similar strategy, an iteratively reweighted Ridge regression algorithm, is used for performing robust Ridge regression based on an MM-estimator (Maronna [2011], originally introduced in Yohai et al. [1987]).*

Example 9.5.3. *A combination of both concepts of interpreting the row measure, i.e., as sampling weights or as fitting weights, manifests itself in fast and robust bootstrap (FRB), cf. Salibián-Barrera et al. [2002] and Salibián-Barrera et al. [2008]. The idea behind this algorithm is to avoid the expensive computation of robust estimators, therefore the estimators are being updated iteratively according to robust weights.*

Example 9.5.4. *Since the coefficients computed by support vector machines (Vapnik [1998]) only depend on the support vectors, one can interpret the selection of these support vectors also as a* $\{0,1\}-$*valued empirical row measure.*

Remark 9.5.1. *Note that the row measures that we mentioned in these examples are data-based, therefore we need to interpret them as* ***(robust) empirical row measures*** *whereas the uniform row measure can be regarded as a* ***prior*** *(non-empirical, though depending clearly on n) row measure* ζ^{init}.

This can be translated to a similar concept for the columns of a data matrix, although far less intuitive. We try to investigate whether it is possible to develop similar strategies when sampling columns.

9.6 The definition of a column measure

Definition 9.6.1. *Suppose we have a data set* $\mathcal{D} := (X, Y) \in \mathbb{R}^{n \times (p+1)}$ *and a loss function* $L : (\mathcal{Y} \times \mathcal{Y}) \to \mathbb{R}_{\geq 0}$ *that should be minimized empirically. Then the* ***column measure w.r.t. L*** *is defined as the map*

$$\nu^{(L)} : (\{1,...,p\}, \mathcal{P}(\{1,...,p\})) \to ([0,p], \mathbb{B} \cap [0,p]),$$

$$\nu^{(L)} : \{j\} \mapsto \nu^{(L)}(\{j\}) =: \nu_j^{(L)} \in [0,1] \ \forall j \in \{1,...,p\}$$

and assigns an importance to each predictor.

Remark 9.6.1. *Note that the column measure is not a probability measure. However, for each singleton* $\{j\} \in \{1,...,p\}$, *the quantity may be seen as a probability for the corresponding predictor to be chosen. Of course, in the case that* $p \to \infty$, *we do no longer have a finite measure but still a sigma-finite measure.*

Remark 9.6.2. *We used this definition primarily because of the easy interpretation that the value assigned to each singleton can be interpreted as a probability. For theoretical purposes,* *it may be handy to normalize the column measure to get a probability measure for finite p. Thus, one would just divide each $\nu^{(L)}(\{j\})$ by $\nu^{(L)}(\{1,...,p\})$. A third variant would map the index set $\{1,...,p\}$ into the $p-$dimensional hypercube $[0,1]^p$, but we do not see any advantages of this vector-wise definition over the proposed one.*

Remark 9.6.3. *By invoking the superscript on the column measure, we would like to emphasize that the column measure may depend on the loss function. This will be further discussed in the subsequent chapters. In fact, the column measure clearly depends on the data set itself, but since it only makes sense to compare different empirical column measures for the same data set but for different loss functions, we do not include the data set in the notation. The superscript will be suppressed if we concern about a column measure w.r.t. some loss function which is not further specified. Additionally, we suppress the dependence of empirical column measures on the number n of training samples if it is not explicitly required.*

Definition 9.6.2. *We call the column measure ν* ***sparse*** *if $\nu(\{j\}) > 0$ for only a few $j = 1,...,p$.*

Of course, we do not know the true column measure ν for any loss function. But we will make the following assumption throughout this thesis.

Assumption 9.6.1. *Let L be a loss function as before and let $\hat{\nu}_n^{(L)}$ (with $\hat{\nu}_{n,j}^{(L)} := \hat{\nu}_n^{(L)}(\{j\})$) be the empirical column measure that has been computed by a variable selection consistent, $L-$adapted learning procedure. Then we assume that*

$$\hat{\nu}_n^{(L)} \xrightarrow{w} \nu^{(L)}.$$

Remark 9.6.4 (Weak convergence). *Note that by the Portmanteau theorem (see e.g. [Elstrodt, 2006, Thm. 4.10]), weak convergence stated in 9.6.1 is equivalent to the condition*

$$\hat{\nu}(B) \longrightarrow \nu(B)$$

for all $\nu-$continuity sets B (see definition A.3.6). Therefore, when concerning about column selection, we can think of the simpler condition

$$\hat{\nu}_{n,j}^{(L)} \overset{n\to\infty}{\longrightarrow} \nu_j^{(L)} \quad \forall j \in \{1,...,p\}.$$

Remark 9.6.5. *Assumption 9.6.1 is no contradiction to variable selection consistency since this property only means that the true model is chosen asymptotically. This has to be understood in the language of column measures that asymptotically, we have*

$$\begin{aligned} \hat{\nu}_{n,j}^{(L)} &\longrightarrow 0 \ \forall j \notin S^0 \\ \hat{\nu}_{n,j}^{(L)} &\longrightarrow c_j^{(L)} > 0 \ \forall j \in S^0 \end{aligned} \tag{9.6.1}$$

for $n \to \infty$.

Remark 9.6.6. *By the assumption of weak convergence in 9.6.1, we potentially could extend the definition of column measures on subsets of* $\mathbb{N}$ *to "column measures" on subsets of* $\mathbb{R}$ *or other uncountable sets.*

For instance, assume that we had time-dependent measurements where the time index t *can be treated as element of the uncountable interval* $[0, T]$ *for some* $T \in \mathbb{R}_{>0}$ *and the goal is to represent this sequence of measurements by a sparser sequence. Then we essentially had to work with "column measures" on* $\mathbb{R}$*.*

9.7 The column measure framework

All classical existing data analysis methods where columns of a data matrix play a role can be identified as limit cases of the column measure framework. One can also define the column measure for these methods, always leading to an empirical column measure $\hat{\nu}$ taking values in $\{0, 1\}$ instead of $[0, 1]$ in each component.

Example 9.7.1. *Non-sparse models like ordinary linear regression simply correspond to* $\hat{\nu}(\{j\}) = 1 \ \forall j \in \{1, ..., p\}$.

Example 9.7.2. *Non-aggregated variable selection procedures like the Lasso, SVMs with variable selection (e.g. Zhu et al. [2004], Wang et al. [2006], Zhang et al. [2016]) or* l_1-*penalized maximum likelihood methods (cf. Park and Hastie [2007]) result in a concrete predictor set, i.e., all columns that are not contained in it are ignored. The column measure is zero for the ignored columns, 1 otherwise.*

Example 9.7.3. *Since the definition of the column measure does not require the existence of a response column, clustering algorithms also fall into this framework. Classical clustering algorithms like* $k-$*means again provide the column measure* $\hat{\nu}(\{j\}) = 1 \ \forall j \in \{1, ..., p\}$ *whereas the column measure given by clustering methods with variable selection (see Witten and Tibshirani [2010] and references therein) again is* $\{0, 1\}-$*valued.*

The identification of a predicted set of variables with an empirical column measure is not restricted to non-ensemble methods like in the examples above. Stability Selection with Lasso models can be directly embedded into the column measure framework. Let $L_\lambda := L + J_\lambda$ be the penalized loss function used in algorithm 4.

In the same manner, the selection frequencies of a Boosting algorithm are given and of course, Stability Selection with Boosting as an ensemble (of Boosting models) of ensembles (of weak learners) is also part of the column measure framework. For completeness, we express the Stability Selection with Boosting proposed by Hofner (Hofner et al. [2015]) in terms of column measures.

```
Initialization: Data $\mathcal{D}$, number $B$ of subsamples, row measure $\zeta^{init}$, size $n_{train} < n$
  of the subsamples, discretization $\Lambda^*$ of $\Lambda$, threshold $\pi_{thr}$;
for $b = 1, ..., B$ do
    Get a subsample $\mathcal{D}^{(b,train)}$ of size $n_{train}$ from the data according to $\zeta^{init}$;
    for $\lambda \in \Lambda^*$ do
        Fit a Lasso model with loss function $L_\lambda$ and get the set $S_{L_\lambda}^{(b)}$;
        Get the raw column measure $(\hat{\nu}^{(L_\lambda)})^{(b)} = (I(j \in \hat{S}_{L_2}))_{j=1}^p$;
    end
end
Get the averaged column measure
    $$\hat{\nu}^{(L_\lambda)} = \left(\frac{1}{B}\sum_b (\hat{\nu}_j^{(L_\lambda)})^{(b)}\right)_{j=1}^p$$
Get the stable column measure
    $$(\hat{\nu}^{(L_\lambda)})^{stab} = (I(\max_{\lambda \in \Lambda^*}(\hat{\nu}_j^{L_\lambda}) \geq \pi_{thr}))_{j=1}^p$$
```

Algorithm 4: Lasso Stability Selection in the language of the column measure

```
Initialization: Data $\mathcal{D}$, number $B$ of subsamples, row measure $\zeta^{init}$, step size
  $\kappa \in ]0,1]$, size $n_{train} < n$ of the subsamples, number $q$, threshold $\pi_{thr}$ ;
for $b = 1, ..., B$ do
    Get a subsample $\mathcal{D}^{(b,train)}$ of size $n_{train}$ from the data according to $\zeta^{init}$ ;
    Fit a Boosting model with early stopping with learning rate $\kappa$ once $q$ variables
      with indices $j_1^{(b)} < ... < j_q^{(b)}$ are chosen. Get the raw column measure
        $$(\hat{\nu}^{(L)})^{(b)} = (I(j \in \{j_1^{(b)}, ..., j_q^{(b)}\}))_{j=1}^p$$
end
Get the averaged column measure
    $$\hat{\nu}^{(L)} = \left(\frac{1}{B}\sum_b (\hat{\nu}_j^{(L)})^{(b)}\right)_{j=1}^p$$
Get the stable column measure
    $$(\hat{\nu}^{(L)})^{stab} = (I(\hat{\nu}_j^{(L)} \geq \pi_{thr}))_{j=1}^p$$
```

Algorithm 5: Boosting Stability Selection in the language of the column measure

Remark 9.7.1. *Per default, the initial row measure ζ^{init} is the uniform measure on $\{1, ..., n\}$, but this can be altered in* `stabs`*.*

Remark 9.7.2 (Convergence of column measures?). *Asymptotic results on variable selection algorithms are often spelled out in terms of the screening property and the variable selection property, see definition 2.3.1.*

Additionally, Meinshausen and Bühlmann [2010] and Hofner et al. [2015], among others, provide results where type I error control, that is, where the control of the probability that a variable with low selection probability is included in the stable predictor set, is possible.

Note that these achievements refer to the "hard decision case" where variables either enter or do not enter the final model. The situation here is different and it remains unclear if one could provide results to show weak convergence of the empirical to the real column measure for $B \to \infty$. Since the measure is discrete, this would be equivalent to show that $\hat{\nu}^{(L)}(\{j\}) \to \nu^{(L)}(\{j\})$ for every $j \in \{1, ..., p\}$.

However, treating the appearance of each single variable in the final model as a Bernoulli random variable, it is the most natural way to estimate the selection probability by computing the relative number of occurences as in algorithms 4 and 5.

Let us come back to the question how row measures and column measures are related. As we have seen, **empirical column measures that are able to assign values to singletons in the whole interval $[0, 1]$ can only be computed by aggregation of different models**. If these different models were fitted on different subsamples or Bootstrap samples of the data, then we actually induced an empirical column measure in the sense of the following definition.

Definition 9.7.1. *Let $\mathcal{D} := (X, Y) \in \mathbb{R}^{n\times(p+1)}$ be a data matrix and let ζ be a row measure on the row indices. Assume that a fitting procedure that optimizes some empirical loss corresponding to a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is given. Let B be the number of samples drawn from ζ. Let $(\hat{\nu}^{(L)})^{(b)}$ be the empirical column measure fitted on the data set reduced to the rows given by the $b-th$ realization of ζ. Then we call the resulting empirical column measure $\hat{\nu}^{(L)}(\zeta)$ with*

$$\hat{\nu}^{(L)}(\zeta)(\{j\}) := \frac{1}{B}\sum_b (\hat{\nu}^{(L)})^{(b)}(\{j\})$$

*the **empirical column measure induced by the row measure** ζ.*

Remark 9.7.3. *In fact, aggregating over all b autonomizes against the concrete realizations* *of the row measure ζ. With a little abuse of notation, one can see this* ***empirical column measure induced by*** ζ ***as empirical expectation of*** $\hat{\nu}^{(L)}$ ***w.r.t.*** ζ.

Remark 9.7.4. *Without calling them measures, Meinshausen and Bühlmann [2010] implicitly spoke of induced column measures when emphasizing that for a given penalty parameter* λ*, the set of selected predictors w.r.t.* λ *also depends on the subsample.*

Note that not all variable selection strategies produce such induced empirical column measures. There already exist strategies that perform **sampling from a column measure** without anyone ever had it called like this, to the best of our knowledge.

Example 9.7.4. *In component-wise Boosting, each iteration performs sampling from p* ***column measures on the set*** $\{1, ..., p\}$*, more precisely, column measure* ν^j *assigns mass 1 to column j and mass zero to each other column, so "sampling" once from each of them ensures that each column will be selected once.*

Example 9.7.5. *Again, naïve best subset selection can be seen as sampling from* ***generalized column measures on the power set*** $\mathcal{P}(\{1, ..., p\})$ *in the same manner as in the example above.*

Remark 9.7.5. *Note that the column measures that are sampled from in the two examples above are given in advance without further knowledge. They have to be treated as* ***initial or prior column measures*** ν^{init}.

Example 9.7.6. *In mixed models, we have again a regressor matrix X of fixed effects but also a matrix Z of random effects. The linear mixed model is given by*

$$Y = X\beta + Z\gamma + \epsilon$$

where $Z \in \mathbb{R}^{n \times q}$*,* $\gamma \in \mathbb{R}^q$ *and where all* ϵ_i *are independent and also independent from the* γ_i*. Then a GLMM-Lasso (Schelldorfer et al. [2014]) provides simultaneously an empirical* $\{0, 1\}-$*valued column measure* $\hat{\nu}^{(fixed)}$ *for the fixed effects as well as an empirical* $\{0, 1\}-$*valued column measure* $\hat{\nu}^{(rand)}$ *for the random effects. Similarly, Boosting for GLMMs (bGLMM, see Tutz and Groll [2010]) provides empirical column measures* $\hat{\nu}^{(fixed)}$, $\hat{\nu}^{(rand)}$ *which generally can take more different values in* $[0, 1]$ *than just zero and one.*

Remark 9.7.6. *There also exist algorithms that combine sampling from column measures and row measures. We postpone them to parts IV and V.*

So, the key question that arises is the following:

Can we find strategies to sample from reasonable empirical column measures to mirror sophisticated sampling techniques from row measures?

9.8 Connecting the column measure with $k-$Step estimators

The main difficulty up to this point is the question how empirical column measures can be incorporated into model fitting and prediction. As their corresponding empirical column measure is the "0/1−limit case", any existing model selection method requires at least the decision if a predictor variable enters the final model or not. Here, we try to explicitly account for the relative importance of each variable.

Therefore, we present an estimator that is based on $k-$Step estimators (see definition 3.5.2) on whose improvement property we rely.

Observe the following fact: We have done model selection and have gotten a subset $J \subset \{1, ..., p\}$ of relevant variables. In the "0/1−setting", this corresponds to the column measure $\hat{\nu}^{(L)}$ with $\hat{\nu}^{(L)}(\{j\}) = 1$ for all $j \in J$ and zero otherwise. Due to equation (3.5.2), the One-Step S_n^1 is asymptotically equal to

$$\mathbb{E}_{\hat{\nu}^{(L)}}\left[\hat{\theta}_n + \frac{1}{n}\sum_i \psi_{\hat{\theta}_n}(X_i, Y_i)\right] := \left(\left((\hat{\theta}_n)_j + \frac{1}{n}\sum_i (\psi_{\hat{\theta}_n}(X_i, Y_i))_j\right) I(j \in J)\right)_{j=1}^p ,$$

where $\hat{\theta}_n$ is the estimated coefficient on the complete data set. This is true since the starting estimator is $\sqrt{n}-$consistent, so the difference in the components of the parameter corresponding to J gets negligible. The same is true for the influence curve by property i) in theorem 3.5.1. The components corresponding to J^c are zero in both cases. Implicitly, the One-Step estimator that is computed after model selection is asymptotically an expectation of the "original" One-Step estimator (i.e., without model selection) w.r.t. a suitable column measure taking values in $\{0, 1\}$. Note again that this leads to partial influence curves (see remark 3.5.2).

Remark 9.8.1. *Similar as in the option pricing case in section 9.1, we wrote our quantity as an expectation w.r.t. a certain measure.*

We now investigate the following problem:

Can we combine the concept of $k-$Step estimators with any column measure?

Assume that we already have drawn B samples from the data and computed the One-Step estimator $\hat{\theta}^{(b)}$ for each $b = 1, ..., B$. Then averaging leads to the estimator

$$\hat{\theta}^{(avg)} := \frac{1}{B}\sum_b \hat{\theta}^{(b)} + \frac{1}{n}\sum_i \frac{1}{B}\sum_b \psi_{\hat{\theta}^{(b)}}(X_i, Y_i). \tag{9.8.1}$$

What does actually happen? Assume there is a subset $K \subset \{1, ..., B\}$ such that a variable j_0 is chosen in all iterations $b \in K$. Then the j_0-th component of the estimated coefficient in equation (9.8.1) would be

$$\hat{\theta}_{j_0}^{(avg)} = \frac{1}{B}\sum_{b \in K} \hat{\theta}_{j_0}^{(b)} + \frac{1}{n}\sum_{i=1}^{n}\sum_{b \in K}(\psi_{\hat{\theta}^{(b)}}(X_i, Y_i))_{j_0}. \tag{9.8.2}$$

Note that the empirical column measure is $\hat{\nu}(\{j_0\}) = \frac{|K|}{n}$ in this case. This leads to the following proposal.

Definition 9.8.1. *Let $\mathcal{D} := (X, Y) \in \mathbb{R}^{n \times (p+1)}$ be a data matrix. Let $\hat{\nu}_n$ be some empirical column measure on the set $\{1, ..., p\}$ and let S_n^1 be a One-Step estimator based on n observations. Then we define the* ***Expected One-Step estimator***

$$\mathbb{E}_{\hat{\nu}_n}[S_n^1] := \hat{\nu}_{n,vec} \circ_H S_n^1 \tag{9.8.3}$$

where $\circ_H$ is the component-wise (Hadamard) product and

$$\hat{\nu}_{n,vec} := (\hat{\nu}_n(\{j\}))_{j=1}^p.$$

However, note that stochastic convergence is not convex in the sense that for a triangular scheme $(X_{i;n})_{i=1,...,n;\, n \in \mathbb{N}}$, one may encounter the situation that

$$\text{mean}_{i=1,...,n}(X_{i;n}) \not\longrightarrow 0$$

for $n \to \infty$ while it still holds that $X_{i;n} \to 0$ stochastically for $n \to \infty$ and for each i. To deal with such cases, we have to sharpen the assumptions on the remainder term in the asymptotically linear expansion that the one-step construction principle leads to (cf. theorem 3.5.1). If the remainder term is denoted by R_n, we now have to assume that

$$\sqrt{n} R_n \longrightarrow 0 \tag{9.8.4}$$

in $L_1(P_{\theta_0})$. In fact, this is a matter of uniform integrability and can be warranted in many cases by suitable exponential concentration bounds (see [Ruckdeschel, 2010a, Prop. 2.1b)+Thm. 4.1]).

Theorem 9.8.1. *Assume that the variable selection procedure that leads to the empirical column measure $\hat{\nu}_n$ is variable selection consistent and that the learning procedure that leads to* *the estimated coefficients is $\sqrt{n}-$consistent. Then the Expected One-Step estimator is variable selection consistent and is $\sqrt{n}-$estimation consistent for the coefficients j with $\hat{\nu}_n(\{j\}) = 1 \ \forall n$ assuming (9.8.4).*

Proof. ***a)*** *By assumption, the variable selection procedure is variable selection consistent (see definition 2.3.1), so $\hat{S} = S^0$ asymptotically. Then the empirical column measure is, asymptotically, zero in all entries $j \notin S^0$, so the Expected One-Step is already correct in these components.*

The learning procedure is $\sqrt{n}-$consistent, so any estimated coefficient $\hat{\theta}^{(b)}$ falls into an $o_P(n^{-1/2})-$neighborhood of the true coefficient, hence the arithmetic mean over all b also does. That implies, as for the initial estimator, it does not depend if it is chosen as arithmetic mean over all $\hat{\theta}^{(b)}$ or if it is computed once since its distance to the true coefficient gets negligible for $n \to \infty$. The same holds for the arithmetic mean of influence curves due to property i) in theorem 3.5.1, provided that the variable is chosen with an empirical probability of one as assumed.

b) *By the variable selection consistency, a growing number of observations guards against false negatives. Thus, asymptotically this case does not have to be treated. The same is true for false positives.*

□

Remark 9.8.2. *The Expected One-Step can be thought of weighting each component of the coefficient by its empirically estimated relevance. Since these weights only take values in $[0, 1]$, we essentially perform a* ***shrinkage*** *of the coefficients that are not chosen with an empirical probability of one.*

Remark 9.8.3. *This technique generalizes the former concept of combining One-Steps with variable selection. In fact, the One-Step estimator with variable selection that we considered in section 3.5 is the special case for a $0/1-$column measure.*

Remark 9.8.4. *The usual One-Step is constructed to get an estimator with a given influence curve. The Expected One-Step can be seen as an extension in the sense that the resulting estimator both has the desired (partial) influence curve and is based on a given column measure.*

Remark 9.8.5. *The extension of the Expected One-Step to the Expected $k-$Step $\mathbb{E}_{\hat{\nu}_n}[S_n^k]$ is straightforward.*

9.9 Time-dependent row and column measures

In the Online Learning context (see for instance Rakhlin et al. [2011]), a new regressor X_t is consecutively provided and a forecast for Y_t has to be computed, based on the information given by $(X_0, Y_0), ..., (X_{t-1}, Y_{t-1})$ is the most simple case.

This Online Learning context has two interesting features that directly correspond to time-dependent versions of row and column measures, i.e., adaptively forgetting observations and updating the model selection.

In contrast to the standard Batch Learning where n instances are observed and where a model is computed based on these observations, the mechanism of Online Learning to repeatedly get new regressors and with some delay, also the corresponding responses, directly leads to a **sequence** $(\zeta_t^{init})_t$ **of initial row measures** where the row measure ζ_t^{init} is a measure on the set $\{1, ..., t\}$.
As for example discussed in Yu et al. [2007] in the context of financial time series, recent information is generally more important than information which is rather old, so an Online Learning algorithm needs to adaptively forget information to handle structural changes in the data which clearly is a usual issue in financial time series. However, they proposed a weighted loss criterion instead of adaptively chosen row measures.

Forgetting factors have for example also been used for Online LDA and Online QDA which essentially rely on recursive and computationally efficient updates of mean, covariance, prior probabilities etc. (Anagnostopoulos et al. [2012]).

Another strategy has been proposed in Duchi et al. [2011] where the informativity of the observations is used to adjust the learning rates in a subgradient descent algorithm. Ribeiro and Cano [2014] discussed several approaches and decided to drop the observations with the smallest learning rates proposed by the procedure of Duchi et al. [2011], i.e., where some kind of **row selection** is performed, or in other words, where **empirical row measures** $\hat{\zeta}_t$ with zero entries are generated.

Even more suitable for our framework are for example Streaming Lassos as discussed in Monti et al. [2016]. The aim is to adaptively update the regularization parameter once a new observation appears which clearly leads to a **sequence of empirical** $\{0,1\}-$**valued column measures** $(\hat{\nu}_t^{(L_2)})_t$. Another online variable selection procedure, namely the adaptive random forest (Gomes et al. [2017]), also provides sequences of empirical column measures which are not necessarily $\{0,1\}-$valued.

9.10 Conclusion

Starting with connecting resampling strategies with a row measure, we proceeded in defining a similar measure, a so-called column measure, on the columns of a data matrix. This column measure has the natural interpretation of assigning relevance to each predictor in the same manner as a row measure assigns a weight to each observation. We have seen that the aggregation of model selection procedures based on resampling essentially leads to empirical column measures that are induced by the underlying row measure.

The true column measure is clearly unknown. However, a key assumption throughout the thesis is that suitable model selection algorithms tend to approximate the true column measure for the number of observations growing to infinity.
The major difficulty when working with column measures is the dependence of the column measure on the underlying loss function. In subsequent chapters, we will investigate how to work with column measures for different loss functions.

A theoretical result that connected $k-$Step estimators with the column measure has been given by the introduction of the Expected One-Step which can easily be extended to Expected $k-$Steps.

Online learning algorithms lead to time-dependent row and column measures.

Acknowledgement: The concept of the column measure and its relation to rejection sampling and to the Lebesgue decomposition were suggested by the supervisor of this thesis, P. Ruckdeschel. These concepts however were worked out in detail and much extended independently by the present author.

Chapter 10

Singular parts of column measures

Whenever we have two or more measures on the same measurable space, the question arises if they are equivalent or if there exist at least singular parts of one measure w.r.t. another one. Singular parts are an issue if changes of measures have to be performed.

In the first section, we briefly apply those standard concepts on the column measure, more precisely, we define singular parts of one column measure w.r.t. a loss function L w.r.t another column measure w.r.t. a different loss function $\tilde{L}$. This is especially relevant for sparse model selection algorithms that usually select a subset of the true set S^0 of relevant variables.

The rest of this chapter occupies itself with the consequences to the already proposed rejection sampling strategy. We will see that some change of measure needs to be performed already during the Boosting procedure itself. At the end, a new algorithm which we call SingBoost which is a first approach to handle situations where singular parts may occur is provided. We list the requirements that SingBoost imposes as well as the main computational issues that we faced and solved and discuss the updating scheme for L_2-Boosting and SingBoost.

We rely on existing results on estimation and prediction consistency of L_2-Boosting (see [Bühlmann, 2006, Thm. 1] resp. [Bühlmann and Van De Geer, 2011, Thm. 12.2]) to show that SingBoost also has these asymptotic properties if a so-called Corr-min condition holds.

10.1 Singular parts and domination

When performing Gradient Boosting w.r.t. different loss functions, it becomes evident that the selection frequencies of the variables differ from loss function to loss function. Therefore, we cannot assume that the column measure $\hat{\nu}^{(L)}$ is identical to the column measure $\hat{\nu}^{(\tilde{L})}$ for two different loss functions L and $\tilde{L}$. So why should equality be even true for the theoretical column measures $\nu^{(L)}$ and $\nu^{(\tilde{L})}$? Ideas related to the rejection sampling algorithm (alg. 3) may still work if these two measures still coincide in terms of which variables are important and only differ in the concrete selection probabilities.

An issue would appear if one applied Gradient Boosting w.r.t loss function L and evaluated the performance in the loss function $\tilde{L}$, but if some relevant variables for loss function $\tilde{L}$ are not selected by Gradient Boosting w.r.t. L. In this case, rejection sampling can never succeed in finding all relevant variables for loss function $\tilde{L}$. The leads to the following question:

Is it realistic to assume that suitable model selection algorithms for different loss functions select different variables?

Remark 10.1.1 (Empirical column measures). *Let S^0 be the true set of relevant variables. Then an algorithm with the screening property (see definition 2.3.1 a)) tends to select a superset of S^0, so it is prone to overfitting. If we compared two algorithms for different loss functions that both have the screening property, it asymptotically may happen that they select* ***different noise variables****. This would not be an issue as long as S^0 is contained in the selected sets of variables for both algorithms. If even the variable selection property (definition 2.3.1 b)) is valid, then the true set S^0 is chosen asymptotically.*

But once real data analysis is performed, i.e., we only have a finite and usually small number of observations, these asymptotic properties clearly do not hold. One can easily run sophisticated algorithms (see example 10.2.1), for example L_2-Boosting and quantile Boosting with possible Stability Selection on either real or simulated data, resulting in different variables being chosen at the end. In the case of simulated data, one instantaneously can find examples where different subsets of S^0 are being selected.

On may object that the L_2- resp. $L_\tau-$Boosting models are not stable, so it would not be fair to compare them. This is true and it happens that Stability Selection applied to both on the same data set leads to the same stable set of coefficients, but this cannot be generalized to arbitrary data. Additionally, for those loss functions, there exist Boosting algorithms. In the case of continuous ranking, we have no chance to check if a stable sparse ranking algorithm would select the same predictors as L_2-Boosting with Stability Selection does.

In view of the fact that one can find data sets such that even L_1- and L_2-Boosting with Stability Selection choose different variables at the end, one should not dare to assume that the empirical column measures w.r.t. L_2 and w.r.t. some ranking loss were equivalent, not *even that this would hold for the theoretical column measures.*

Assumption 10.1.1. *We assume that even the true column measures for different loss functions L and $\tilde{L}$ differ, i.e.,*

$$\nu^{(L)} \neq \nu^{(\tilde{L})}.$$

This motivates to adapt the definition of domination and singular parts for our column measure framework.

Definition 10.1.1. *Let L, $\tilde{L}$ be loss functions as before and let $\mathcal{D} := (X, Y) \in \mathbb{R}^{n \times (p+1)}$ be a data set.*
a) *We call the column measure $\nu^{(L)}$* ***dominated by*** *$\nu^{(\tilde{L})}$, written $\nu^{(\tilde{L})} \gg \nu^{(L)}$ if*

$$\nu^{(\tilde{L})}(\{j\}) = 0 \Longrightarrow \nu^{(L)}(\{j\}) = 0 \ \forall j \in \{1, .., p\}.$$

b) *We say that $\nu^{(L)}$ and $\nu^{(\tilde{L})}$ are* ***equivalent****, $\nu^{(L)} \sim \nu^{(\tilde{L})}$, if*

$$\nu^{(L)} \gg \nu^{(\tilde{L})} \text{ and } \nu^{(\tilde{L})} \gg \nu^{(L)},$$

or just written differently, if

$$\nu^{(L)}(\{j\}) = 0 \Longleftrightarrow \nu^{(\tilde{L})}(\{j\}) = 0.$$

c) *If $\nu^{(\tilde{L})} \gg \nu^{(L)}$ and if the measures are not equivalent, then we call the greatest set $J_{\tilde{L}}^{L} \subset \{1, ..., p\}$ such that*

$$\nu^{(L)}(\{j\}) = 0 \neq \nu^{(\tilde{L})}(\{j\}) \ \forall j \in J_{\tilde{L}}^{L}$$

the ***singular part of*** *$\nu^{(\tilde{L})}$* ***w.r.t.*** *$\nu^{(L)}$.*

Remark 10.1.2. *One could extend the definition of the column measure by invoking the learning procedure that is used to perform risk minimization.*

It certainly will make a difference if the quadratic loss is minimized by ordinary least squares or by L_2-Boosting where the latter will generally result in a much sparser model, generating a singular part, too.

We do not think that this extension would be meaningful. If one is interested in the sparsity of the models computed by different algorithms for the same loss function, one does not really need the column measure. On the other side, of course the empirical column measure from the ordinary least squares model would "dominate" every other model's empirical column measure, but if every variable is selected anyways, thinking of a column measure is superfluous, leading every singleton $\{j\}$ to get the probability 1.

Assumption 10.1.2. *If we compare the relevant sets of predictors for different loss functions, we always consider that a suitable learning algorithm performing (sparse) variable selection has been applied without accounting for it specifically.*

Since we face discrete measures on the same finite measurable space, we can easily find the Lebesgue decomposition (see theorem A.3.3) of either measure if none of them dominates the other one.

Example 10.1.1 (Lebesgue decomposition). *Assume we have a set J where both measures are nonzero for every singleton contained in J. Additionally, we have singular parts $J_{\tilde{L}}^{L}$, $J_{L}^{\tilde{L}}$, so the union of all three sets is $\{1, ..., p\}$. Then we trivially get the Lebesgue decompositions*

$$\nu^{(\tilde{L})} = \nu^{(\tilde{L})}|_{J_L^{\tilde{L}} \cup J} + \nu^{(\tilde{L})}|_{J_{\tilde{L}}^{L}},$$
$$\nu^{L} = \nu^{L}|_{J_{\tilde{L}}^{L} \cup J} + \nu^{(L)}|_{J_L^{\tilde{L}}}.$$

Then $\nu^{(\tilde{L})}|_{J_{\tilde{L}}^{L}} \perp \nu^{(L)}$ and $\nu^{L}|_{J_L^{\tilde{L}}} \perp \nu^{(\tilde{L})}$.

Assumption 10.1.3. *We even go further as in assumption 10.1.1 and assume that the column measures $\nu^{(L)}$ and $\nu^{(\tilde{L})}$ already differ on the set S^0 while not necessarily having singular parts, so we assume*

$$\nu^{(L)}|_{S^0} \neq \nu^{(\tilde{L})}|_{S^0}, \quad i.e., \quad \exists j \in S^0 : \nu_j^{(L)} \neq \nu_j^{(\tilde{L})}.$$

We always keep in mind that we generally assume sparsity of the underlying model, see assumption 2.2.1.

10.2 Consequences to model selection

With the definition of the column measure w.r.t. different loss functions and their potential singular parts, we once more take a look at the rejection sampling idea.

Each proposed model (e.g. a Boosting model or a model determined by Stability Selection with Boosting or with the Lasso) provided a "raw" empirical column measure, though not necessarily stable. However, this measure corresponds to the selected loss function, i.e., the loss L. Performing rejection sampling by evaluating the model w.r.t. the loss function $\tilde{L}$

effectively picks one of these proposed raw column measures. The resulting column measure thus differs from the one that we would have gotten for the original target loss $\tilde{L}$. Mathematically spoken, through rejection sampling, we implicitly made a **change of measure**.

In the same way as a change of measure requires that the initial measure dominates the target measure, we get a similar requirement for our rejection sampling strategy.

Remark 10.2.1 (Rejection sampling and domination). *A rejection sampling strategy as algorithm 3 where models are fitted w.r.t. a loss function L and evaluated in another loss function $\tilde{L}$ is inadequate if*

$$J_{\tilde{L}}^{L} \neq \emptyset, \tag{10.2.1}$$

or in other words, if the measure $\nu^{(\tilde{L})}$ has a singular part w.r.t. $\nu^{(L)}$.

In turn, our strategy can only be meaningful if

$$\nu^{(L)} \gg \nu^{(\tilde{L})}. \tag{10.2.2}$$

Evidently, singular parts of $\nu^{(L)}$ w.r.t. $\nu^{(\tilde{L})}$ can lead to false positives (w.r.t. $\tilde{L}$), but the rejection sampling strategy can still work if these false positives can be appropriately neglected in subsequent stabilization steps.

Remark 10.2.2 (Domination and variable selection consistency). *Even if there exist algorithms for L resp. $\tilde{L}$ that both have the variable selection consistency property (definition 2.3.1), there would be no guarantee that there would be no empirical singular parts in real applications, i.e., **for finite** n.*

Remark 10.2.3 (Surrogate losses and singular parts). *Although the usage of surrogate loss functions is common for example in classification settings, we are not sure if a "wrong" surrogate may even lead to singular parts of the column measure corresponding to the original loss (e.g. the $0/1-$loss) w.r.t. the column measure corresponding to the surrogate loss.*

Remark 10.2.4 (Time-dependent singular parts). *In the Online Learning context (see section 9.9), one usually is provided with a new regressor X_t at time t with the goal to predict the corresponding response Y_t which is revealed afterwards. Clearly, if we try to optimize a loss w.r.t. some loss function $\tilde{L}$ using an $L-$adapted algorithm, we may get a singular part $(J_{\tilde{L}}^{L})_t$. In the case of facing structural changes in the data, for example when having financial data (see Yu et al. [2007]), the relevance of the predictors w.r.t. the losses L and $\tilde{L}$ may also vary over time, leading to a **sequence of true column measures** $(\nu_t^{(L)})_t$ and $(\nu_t^{\tilde{L}})_t$.*

We cannot exclude that even the singular parts vary over time due to the following two reasons. First, if new observations let predictors which did not yet appear to be relevant for L indeed become relevant, then the cardinality of the singular part can decrease over time. On the other hand, a particular forgetting scheme can also cause the converse direction if predictors which were relevant for L are no longer relevant for the newer observations. But if these predictors are still relevant for $\tilde{L}$, the cardinality of the singular part would increase. Therefore, one has to be aware of potentially encountering a ***sequence of singular parts*** *$((J_{\tilde{L}}^L)_t)_t$ which does not need to be constant.*

Remark 10.2.5. *Note that singular parts between column measures lead to a deficiency in the sense of definition A.3.7 when concerning about model selection. Without model selection, i.e., having non-zero coefficients for all predictors, we may find a suitable transition which translates the coefficients w.r.t. L into coefficients w.r.t. $\tilde{L}$. Once model selection is performed, the presence of a singular part $J_{\tilde{L}}^L$ does not allow such a transition.*

Remark 10.2.6. *Clearly, any learning procedure can produce false positives, so it is not impossible to select variables from the singular part of $\nu^{(\tilde{L})}$ w.r.t. $\nu^{(L)}$ in practice. However, theoretically (say, asymptotically) this kind of "useful mistake" would become more unlikely the more n grows, apart from the fact that no model should ever rely on errors.*

Example 10.2.1. *Not concerning about singular parts for the moment, let us take a look at the following simple example where we apply the Stability Selection of Hofner et al. [2015] implemented in* `stabs` *to Boosting models with different loss functions:*

```
require(mboost)
require(stabs)
require(pcaPP)

load('StabSelEx.RData')
D1←Data$D
resl2←glmboost(Y~.,D1)
reshub←glmboost(Y~.,D1,family=Huber())
resl1←glmboost(Y~.,D1,family=Laplace())
restau←glmboost(Y~.,D1,family=QuantReg(tau=0.75))
attributes(varimp(resl2))$self
```

```
 [1] 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.17 0.04 0.00 0.00
[15] 0.00 0.00 0.00 0.08 0.02 0.00 0.00 0.00 0.00 0.00 0.00 0.02 0.05 0.12
[29] 0.05 0.00 0.00 0.00 0.06 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.15
[43] 0.05 0.00 0.00 0.13 0.03 0.00 0.00 0.02 0.01
```

```
print("L2-Boosting")
```

```
[1] "L2-Boosting"
```

```
set.seed(7983216)
stabsel(resl2,q=8,PFER=1)$sel
```

```
X10 X27 X41 X45
 11  28  42  46
```

```
print("Huber-Boosting")
```

```
[1] "Huber-Boosting"
```

```
set.seed(7983216)
stabsel(reshub,q=8,PFER=1)$sel
```

```
Warning in stabsel.mboost(reshub, q = 8, PFER = 1): 'mstop' too small in 1
     of the 100 subsampling replicates to select 'q' base-learners;
    Increase 'mstop' bevor applying 'stabsel'
```

```
X10 X41 X45
 11  42  46
```

```
print("L1-Boosting")
```

```
[1] "L1-Boosting"
```

```
set.seed(7983216)
stabsel(resl1,q=8,PFER=1)$sel
```

```
Warning in stabsel.mboost(resl1, q = 8, PFER = 1): 'mstop' too small in 75
     of the 100 subsampling replicates to select 'q' base-learners;
    Increase 'mstop' bevor applying 'stabsel'
```

```
X10 X41 X45
 11  42  46
```

```
print("Ltau-Boosting, tau=0.75")
```

```
[1] "Ltau-Boosting, tau=0.75"
```

```
set.seed(7983216)
stabsel(restau,q=8,PFER=1)$sel
```

```
Warning in stabsel.mboost(restau, q = 8, PFER = 1): 'mstop' too small in
    99 of the 100 subsampling replicates to select 'q' base-learners;
    Increase 'mstop' bevor applying 'stabsel'
```

```
(Intercept)
          1
```

We see that even for this unspectacular data set with $n = 100$, $p = 50$ and $s^0 = 10$, there is no consensus concerning a stable model for these four loss functions. Note that the warning appeared in all cases except for the L_2-Boosting-based Stability Selection, meaning that m_{iter} should be increased since even after 100 iterations since some Boosting algorithms did not select q predictors. However, since this issue just happened one time for Huber Boosting, we can ignore it. One the other hand, the issue that the major fraction of Boosting models for the L_1- and the $L_\tau-$loss for $\tau = 0.75$ did not select q predictors indicates the low relevance of the non-selected predictors for these loss functions. Especially for $L_\tau-$Boosting, we do not see any reasonability to thoughtlessly just increase the number of iterations as long as the warning does not appear anymore.

However, we note that the ordering of the relevances of the four selected stable variables w.r.t. the squared loss is identical to the respective ordering w.r.t. the other losses, disregarding the intercept as leading variable for quantile Boosting. So, when trying different cutoffs, one may would get identical stable models. But this is a contradiction to the statement of Meinshausen and Bühlmann [2010] and Hofner et al. [2015] that the cutoff is of minor importance. We conclude that for example for the hard ranking loss, we had no chance to get a suitable stable model but to try different models of the stated ones above since although the quantity $\mathbb{E}[Y|X]$ that L_2-Boosting approximates is an optimal ranking rule, there is no evidence that a stable L_2-based predictor set would be more suitable to the ranking problem than some stable L_1- or $L_\tau-$based predictor set.

Summing up, a vital question when trying to solve the problem of sparse empirical minimization of $\tilde{L}$ by such a rejection sampling scheme is the following: How can we control for singular parts of the (unknown) column measure $\nu^{(\tilde{L})}$ w.r.t. the (unknown) column measure $\nu^{(L)}$?

10.3 SingBoost: Boosting with singular parts for any target loss

The relevance of a certain predictor in a Boosting model w.r.t. a loss L is simply determined by dividing the number of iterations in which this variable has been selected by the number of Boosting iterations.

Remark 10.3.1 (Boosting and rejection sampling). *More precisely,* ***each Boosting it-***

eration is a rejection sampling step *itself, comparing weak learners w.r.t. each predictor and selecting the one that performs best, evaluated by the loss L. For example in L_2-Boosting, all p predictors are chosen and the RSS of the corresponding simple least squares regression fit is computed, thus only accepting the single predictor with the lowest resulting RSS. The acceptance ratio is clearly p^{-1}.*

Remark 10.3.2 (**Elicitability**). *Note that although L_2-Boosting also compares different (simple) models in each iteration, this concept is not directly related to the strict consistency of the squared loss (cf. Gneiting [2011]). The basic idea behind consistent loss functions and therefore elicitability (chapter 7) is that the best forecast can be identified, so honest predictions are rewarded (see also Gneiting and Raftery [2007]).*

In a Boosting algorithm, we never expect a single column to perfectly model the current residuals. However, elicitability is usually related to a ranking of forecasts (see e.g. Fissler et al. [2015], Ziegel [2016]) which indeed can be transferred to Boosting when treating the predicted values of the simple linear models as "forecasts" for the current residual vector.

Since trying to control for singular parts on the level of whole Boosting models is inappropriate in the presence of singular parts, we need to interfere deeper into the Boosting algorithm itself and force that $\tilde{L}$ is already respected there.

This motivates the following idea to perform a singular Boosting step w.r.t $\tilde{L}$ where $\tilde{L}$ may do not even have a (unique) gradient: ***Fit simple least squares models as in L_2-Boosting and evaluate them in the target loss $\tilde{L}$ in each $M-$th iteration.***

Heuristically, we are getting a chance to have variables that correspond to a singular part of $\nu^{(\tilde{L})}$ w.r.t $\nu^{(L)}$ being detected as well as keeping the efficient structure of component-wise L_2-Boosting.

Remark 10.3.3 (**Change of measure**). *Note that we essentially already perform a change of measure in the innermost level of the whole resulting learning algorithm. In terms of column measures, if in such a singular iteration k a variable $\tilde{j}_k$ has been selected, we can think of the "empirical column measure" $e_{\tilde{j}_k}$, i.e., the $\tilde{j}_k-$th $p-$dimensional unit vector. But if we had performed a usual Boosting iteration, we had gotten variable $\hat{j}_k$, i.e., by performing a rejection step in the singular iteration, we enforced a change of measure from the "empirical column measure" $e_{\hat{j}_k}$ to the "empirical column measure" $e_{\tilde{j}_k}$.*

We will call this algorithm **SingBoost**:

```
Initialization: Data $\mathcal{D}^{sing}$, step size $\kappa \in ]0,1]$, number $m_{iter}$ of iterations, number
  $M \leq m_{iter}$ (each $M-$th iteration is a singular iteration), target loss $\tilde{L}$ (as part of a
  family object singfamily) and binary variable LS;
Set
    $$\text{runs} = \left\lfloor \frac{m_{iter}}{M} \right\rfloor$$
  Set $f^{(0)} := 0$;
for $k = 1, ..., \text{runs}$ do
    if LS==F then
        Perform a single step of Gradient Boosting w.r.t. $\tilde{L}$ on the residuals of model
          $\hat{f}^{((k-1)M)}$;
    else
        Evaluate all simple least squares fits on the residuals of model $\hat{f}^{((k-1)M)}$ w.r.t.
          $\tilde{L}$ and take the best one;
    end
    Get the weak model $\hat{g}^{((k-1)M+1)}$ and update the model via
    $\hat{f}^{((k-1)M+1)} = \hat{f}^{((k-1)M)} + \kappa\hat{g}^{((k-1)M+1)}$;
    Perform $(M-1)$ steps of $L_2-$Boosting starting with the residuals w.r.t. the
      model $\hat{f}^{((k-1)M+1)}$;
    Get the updated model $\hat{f}^{(kM)}$
end
```

Algorithm 6: SingBoost

Remark 10.3.4. *The argument* **`LS=F`** *which performs a single Gradient Boosting step w.r.t. $\tilde{L}$ is only meaningful in simulation settings if one wants to compare the performance of different baselearners. In general (and when we actually need SingBoost), $\tilde{L}$ does not have a corresponding "pure" Boosting algorithm. Then setting* **`LS=T`** *uses simple least squares baselearners in the singular steps, but compares them according to $\tilde{L}$.*

Remark 10.3.5. *We will see at the end that the whole cross-validated CMB-3S algorithm that we propose (see subsection 12.4) includes different training, validation and test sets on each level. Therefore, we decided that tedious notation, here starting with the superscript in $\mathcal{D}^{sing}$, is indeed indispensable to avoid confusion.*

Remark 10.3.6. *The only property of $\tilde{L}$ that we require to get a computationally tractable algorithm is that $\tilde{L}$ is* ***easy to evaluate****. Note that no (local) differentiability property, not even continuity of $\tilde{L}$ is required, so especially the existence of a (sub)gradient is needless! This is the reason why we think of SingBoost as* ***gradient-free Gradient Boosting****.*

Remark 10.3.7. *Again, note the strong similarity to performing rejection sampling in order* *to generate samples according to a density f (section 9.4). Although one does not directly sample from the cumulative distribution function corresponding to f, one at least has to evaluate the density in the random number drawn according to the density g as we evaluate the loss function $\tilde{L}$ at the baselearners, e.g., the "samples".*

Remark 10.3.8. *The hard or the localized ranking loss are such complicated loss functions that are not even continuous, but can be evaluated in $\mathcal{O}(n\ln(n))$ steps thanks to lemma 6.1.1.*

Remark 10.3.9 (Why L_2−Boosting?). *So far, it is not evident why we take L_2−Boosting as basis algorithm. Of course, just the argument that L_2−Boosting approximates $\mathbb{E}[Y|X]$ and that this is already an optimal scoring rule for ranking is too weak since we do not restrict SingBoost to ranking applications and since this conditional expectation gets approximated by a subset of the column space of the regressor matrix which potentially provokes that the selected variables, despite being relevant, are not the best relevant variables for the ranking problem or generally for loss $\tilde{L}$.*

The main initial reason why we used L_2−Boosting is its excellent implementation which is very efficient and extremely fast. However, we will see at the end of this chapter (section 10.5) that there is indeed a solid theoretical foundation of this $L_2-\tilde{L}$−hybrid algorithm.

Remark 10.3.10 (Sparse Boosting). *Although not intended to detect potential singular parts by tailoring the model selection to another loss function, the Sparse Boosting algorithm (Bühlmann and Yu [2006]) can be regarded as related work. The aim of this algorithm is to provide sparser models by trying to minimize the out-of-sample prediction error. Since this is not directly possible, they use the degrees of freedom defined as $\mathrm{tr}(\mathcal{B}_m)$ for the Boosting operator*

$$\mathcal{B}_m = I - (I - \kappa\mathcal{H}_{\hat{j}_m})\ldots(I - \kappa\mathcal{H}_{\hat{j}_1})$$

to estimate the model complexity where $\mathcal{H}_{\hat{j}_k}$ is the hat matrix corresponding to the weak model which is based on variable $\hat{j}_k$ (see also Bühlmann and Yu [2003] for further details on the Boosting operator). In fact, they choose the variable that minimizes the residual sum of squares, penalized by the model complexity, and then proceed as in L_2−Boosting. As they pointed out, this leads to ***possibly different choices of the best variable*** *in each iteration which in some sense can also be understood as some kind of SingBoost with a* ***sequence of target loss functions*** *$\tilde{L}^{(k)}$ that equal their penalized criterion in every iteration $k = 1, ..., m$.*

Remark 10.3.11. *The analogy to Sparse Boosting leads to the question if there are cases where working with varying loss functions in SingBoost is meaningful when concerning about detecting potential singular parts. We postpone this discussion to section 12.3.*

Now, we briefly list the main implementation steps and issues:

1.) We include the function argument `singfamily` to provide flexibility. For standard families that are implemented in the `R`−package `mboost`, the singular step is one iteration of `glmboost` with the corresponding loss function using the commands `LS=F` and `boost_control` to set the number of Gradient Boosting iterations for this loss function to one, otherwise the component-wise least squares approach is applied.

2.) If the target loss $\tilde{L}$ is discrete-valued, it can happen that more than one simple least squares model lead to the same loss, so the optimal predictor cannot be uniquely determined. We use the command `sample` to randomly choose one of the considered columns with equal probability.

3.) After the singular steps, we save the coefficients but only want to insert the residuals in the `glmboost` function. This can be an issue if they are not approximately mean-centered. Therefore, we mean-center them manually so that `glmboost` skips the computation of the offset in the first iteration. After the $(M-1)$ iterations, we again add the mean to the intercept. This is not a problem since this shifting does neither change the selected variables nor their non-intercept coefficients.

4.) We count the overall selection frequencies of all variables. The ones in the non-singular steps are internally saved and can be retrieved with the command

```
attributes(varimp(...))$selfreqs
```

where the `mboost` object needs to be inserted into the brackets. We also display the selected variables as output of `singboost`.

5.) We insert a value for the step size κ at the beginning. As suggested p.e. by Friedman (Friedman [2001]), one can even perform a line search by taking the value for κ for which the loss of the resulting combined model is minimal. Bühlmann and Hothorn [2007] argued that this technique is not needed when optimizing the squared loss. However, one can think about a line search when invoking singular iterations w.r.t. a loss $\tilde{L}$. The reasonability of this step depends on $\tilde{L}$ because with a loss which is still rather expensive to evaluate like a ranking loss, the line search would again decelerate the whole algorithm and should not be recommended. Therefore, a line search is not contained in our implementation.

6.) For details concerning the selection of the best variable in the singular steps, see below (section 10.4).

7.) We call the number of iterations m_{iter} to distinguish between the number m_{stop} in L_2-Boosting where indeed m_{stop} iterations according to L_2 are performed in contrast to the respective number $M\lfloor m_{iter}/M \rfloor$ – runs in SingBoost. It is hard to decide when one should stop the algorithm. Approaches for early stopping standard Boosting algorithms do not work here since they are either in general loss-based (convergence criteria) or based on an AIC (Bühlmann and Yu [2003], Bühlmann [2006], Mayr et al. [2012b], see also section 2.4). But the latter penalizes the model complexity which contradicts the intention to find more variables, i.e., those that form the singular part $J_{\tilde{L}}^{L}$. We are sure that the optimal number of iterations is not only dependent from κ but also from the number M for which we also do not make universal recommendations. In our applications, we therefore just try different numbers of m_{iter} and M and leave this problem as an open question for future research.

8.) In its current implementation, a data frame has to be inserted as `singboost` input. Then, the last column of the data frame is automatically treated as response column and a regression of this column based on the rest of the data frame is performed. If the regression should only be done w.r.t. several selected columns, one just has to insert the reduced data frame.

9.) SingBoost can easily handle categorical variables, interactions and basis functions. One just has to insert the model matrix where extra columns or transformed columns already have been added to represent those objects. Note that the column corresponding to the intercept containing only ones which is generated by `model.matrix` needs to be deleted before running SingBoost since this column is generated automatically in `singboost`. See also part VI for a demonstration of our implemented functions.

10.) We do not provide any weighted version of SingBoost (we discuss it again later, see remark 12.6.3). It would not be that clear if the same weights for two different loss functions would be appropriate. However, the next part will reveal that our final algorithm indeed can handle weights, but in an adapted fashion in the sense that some empirical row measure will be computed. If there are rows that appear to be outliers, either delete them in advance or use a possible robust extension of SingBoost (see section 18.2 for further details and ideas).

11.) To provide a quick overview on the relevant input arguments, see the following command line that is used to apply `singboost` with $M = 10$, $m_{iter} = 100$ and $\kappa = 0.1$ (all defaults) with hard ranking singular steps:

```
singboost(D,M,m_iter,kap,singfamily=Rank(),LS=T)
```

Remark 10.3.12 (Grouped variables). *To the best of our knowledge, the BlockBoost algorithm (Gertheiss et al. [2011]) has not been implemented under this name in* R. *But in fact, Hofner et al. [2014] describe that one can assign multiple variables to the baselearner when using the function* `gamboost`. *These variables are treated as a group, so the baselearner is built on all these variables together and either all corresponding coefficients are updated in the Boosting iteration or none of them. The same is true for the function* `mboost`. *If a categorical variable appears, both functions automatically treat all corresponding columns as group and perform block-wise updates. So in the case of group-wise variables that enter SingBoost, it is rather a programming task to automatically flag variables as groups. It may be a topic for future work to implement it in a way that it runs reasonably fast for SingBoost which would make our algorithm capable to handle block-wise updates for categorical variables and even grouped variables.*

Remark 10.3.13. *Of course, if the singular family is set to* `Gaussian()` *which corresponds to the squared loss, SingBoost delivers exactly the same coefficients and selection frequencies as L_2-Boosting with the same number of iterations. Note that the reported intercept indeed differs, but just because* `glmboost` *performs a centering step in advance which results in a so-called offset value (see algorithms 1 and 23). For unknown reasons, this function reports the offset value and the intercept separately in the sense that the true intercept which is used for prediction is the sum of both. This sum is reported by* `singboost`.

Remark 10.3.14. *The SingBoost algorithm is implemented in* R *as* `singboost`. *In its current implementation, it consumes much more computational time than* `glmboost`, *depending on M. Maybe a neat memory handling or outsourcing code chunks using* `C` *or* `FORTRAN` *as done for* `mboost` *and* `glmboost` *could improve the running time.*

However, SingBoost is significantly faster than our first approach using surrogate losses in a pair-wise fashion. Figure 10.1 shows the computational performance of SingBoost in comparison with HingeBoost.

We want to be honest and provide a similar plot (figure 10.2) that compares the time consumption for L_2-Boosting as implemented as `glmboost` *and SingBoost for L_1- and ranking singular iterations as well as for the case where we artificially invoke Gaussian "singular" iterations, i.e., we use* `glmboost` *in each iteration. Clearly, there is no chance to beat* `glmboost`.

As for the complexity of SingBoost, see the following lemma.

Lemma 10.3.1 (Complexity of SingBoost). *Assume that the evaluation of the target loss function $\tilde{L}$ requires $\mathcal{O}(c_n)$ operations for some $c_n \geq n \ \forall n$. Then the complexity of SingBoost is $\mathcal{O}(mc_np)$.*

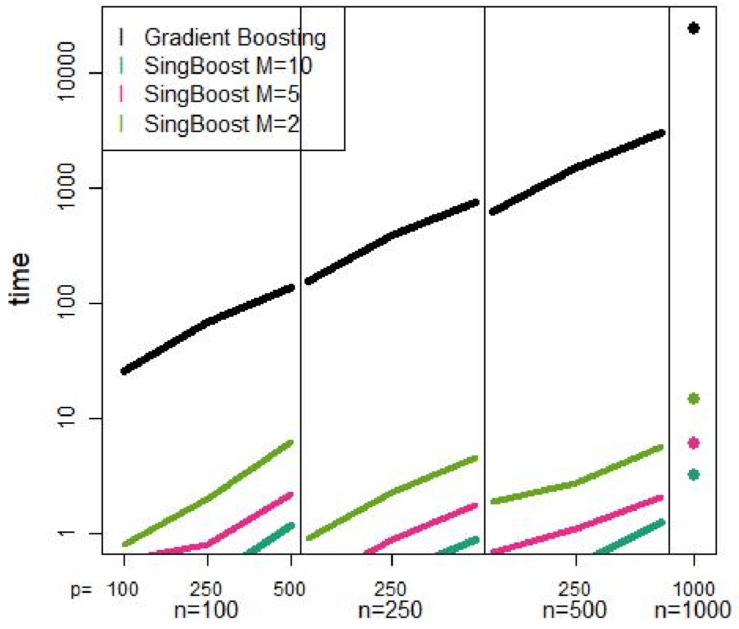


Figure 10.1: Time consumption: SingBoost with different values of M vs. HingeBoost on several allocations of (p, n) and only $m_{iter} = 10$

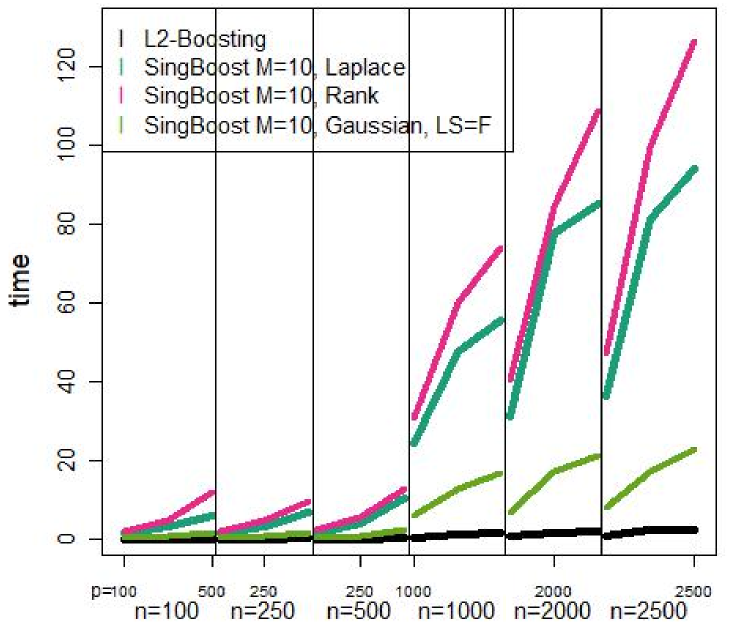


Figure 10.2: Time consumption: SingBoost with different singular families and L_2-Boosting, $m_{iter} = 100$

Remark 10.3.15. *Since L_2-Boosting is of complexity $\mathcal{O}(mnp)$, the loss in performance (assuming an equally excellent implementation) for SingBoost compared to L_2-Boosting gets negligible if $c_n = \mathcal{O}(n)$.*

In the case of the hard ranking loss, we have $c_n = \mathcal{O}(n\ln(n))$ due to lemma 6.1.1, so for real-data applications where n is usually rather small, the complexity of SingBoost is comparable with that of L_2-Boosting, also reminding that not each iteration is a singular iteration.

10.4 Variable selection and updating

Essentially, each singular step of SingBoost is still a Gradient Boosting step w.r.t. the squared loss. This is true since fitting simple least squares models and evaluating them in $\tilde{L}$ does not contradict the requirements on the base procedure from Bühlmann [2006]. Indeed, we have a real-valued baselearner that fits the residuals, i.e., the negative functional gradient of the squared loss.

Of course, according to Bühlmann [2006], a base procedure has to be specified before running the algorithm so that the base procedure performed in each iteration is the same which is not true for SingBoost due to the different evaluations, either in L or in $\tilde{L}$. Therefore, the theoretical results that already have been established for L_2-Boosting in Bühlmann and Yu [2003] and Bühlmann [2006] are not one-to-one transmissible to SingBoost. However, we establish asymptotic results in the next section.

Let us illustrate more computational details of SingBoost. We did not yet really specify the term "taking the best one" in algorithm 6. In standard L_2-Boosting with component-wise least squares, we select column $\hat{j}_k$ with

$$\hat{j}_k = \underset{j}{\operatorname{argmin}} \left(\sum_i (r_i^{(k-1)} - X_{i,j}\hat{\beta}(j))^2 \right) \tag{10.4.1}$$

in the $k-$th iteration where $\hat{\beta}(j)$ is the baselearner w.r.t. column j. This is only valid because the squared loss L_2 just depends on the residuals, i.e.

$$L_2(y, f) =: L_2(y - f) = L_2(|r|). \tag{10.4.2}$$

Note that by the linearity of our model, it does not matter if we compute the residual sum of squares by comparing the predicted responses from the baselearner with the current residuals or the predicted responses from the whole model, including the new base model, with weight 1 instead of κ, with the original responses if the loss function has a structure as in (10.4.2). After the $(k-1)-$th iteration, the resulting residual vector is

$$r^{(k-1)} = Y - X\hat{\beta}^{(k-1)}.$$

Let $\hat{j}_k$ be the selected predictor in the $k-$th iteration, let $g^{(k)}$ be the base model and $\hat{\beta}$ be the corresponding coefficient. Then, obviously, we have

$$\begin{aligned} r_i^{(k-1)} - g^{(k)}(X_i) &= Y_i - X_i\hat{\beta}^{(k-1)} - (\kappa(\hat{\beta}(\hat{j}_k))_0 + \kappa(\hat{\beta}(\hat{j}_k))_{\hat{j}_k} X_{i,\hat{j}_k}) \\ &= Y_i - X_i(\hat{\beta}^{(k-1)} + \kappa\hat{\beta}(\hat{j}_k)) \end{aligned}$$

where the sum of the coefficients in the brackets is what we meant by "whole model". Therefore, we can just compare the simple models w.r.t. each column by quantifying their prediction performance w.r.t. the current residuals which is computationally easier and faster since no large matrix enters the matrix-vector-multiplication. But nevertheless, the selection scheme in 10.4.1 can be equivalently written as

$$\hat{j}_k = \underset{j}{\operatorname{argmin}} \left(\sum_i (Y_i - X_i(\hat{\beta}^{(k-1)} + \kappa\hat{\beta}(j)))^2 \right) \tag{10.4.3}$$

which is also faced in literature, but we think that 10.4.1 reflects even more the principle of Boosting.

Remark 10.4.1 (Maximal absolute correlation with the residual). *Note that, due to the learning rate κ, it is not evident at the very first glance that choosing $\hat{j}_k$ such that the loss*

$$\sum_i (r_i^{(k-1)} - X_i\hat{\beta}(\hat{j}_k))^2$$

is minimized w.r.t. j implies that the loss

$$\sum_i (Y_i - X_i(\hat{\beta}^{(k-1)} + \kappa\hat{\beta}(\hat{j}_k))))^2$$

from the aggregated model w.r.t. κ is also minimal compared to the losses that we had gotten for any other $j \neq \hat{j}_k$ where again the notation $\hat{\beta}(\hat{j}_k)$ just artificially highlights the dependence of the coefficient on column $\hat{j}_k$ (and possibly the intercept column).

But indeed, this is true due to the fact that for L_2-Boosting, ***the optimal variable is always the variable with the maximal absolute correlation with the current residual****. This has been proven in an old version of a work of Zhao and Yu (Zhao and Yu [2004]), but was not published in the final version Zhao and Yu [2007]. Therefore, we recapitulate* *the proof using our notation. As in Zhao and Yu [2004], we assume for simplicity that each column of X is normalized, i.e.,*

$$\sum_i X_{ij} = 0, \quad \sum_i X_{ij}^2 = 1,$$

and that we do not include intercepts in the baselearners. Then we take a look at the L_2-loss difference of two subsequent iterations:

$$\sum_i (Y_i - \hat{f}^{(k+1)}(X_i))^2 - \sum_i (Y_i - f^{(k)}(X_i))^2$$

$$= \sum_i (Y_i - X_i(\hat{\beta}^{(k)} + \kappa e_j \hat{\beta}(j)))^2 - (Y_i - X_i \hat{\beta}^{(k)})^2$$

$$= \sum_i (r_i^{(k)} - \kappa e_j X_i \hat{\beta}(j))^2 - (r_i^{(k)})^2 = \sum_i (\kappa e_j X_i \hat{\beta}(j))^2 - 2\kappa \hat{\beta}(j) r_i^{(k)} e_j X_i$$

$$= \sum_i (\kappa X_{ij} \hat{\beta}(j))^2 - 2\kappa \hat{\beta}(j) r_i^{(k)} X_{ij}$$

where the last summand clearly depends on the correlation of $X_{\cdot,j}$ and $r^{(k)}$ since it holds that

$$\hat{\beta}(j) = \mathrm{Corr}(X_{\cdot,j}, r^{(k)}) \sqrt{\mathrm{Var}(r^{(k)})} = \mathrm{Cov}(X_{\cdot,j}, r^{(k)})$$

due to the assumption that the columns of X are normalized, so the sign of the last summand stays negative. More precisely, we can write the last line of the previous display as

$$\kappa^2 \hat{\beta}(j)^2 \sum_i X_{ij}^2 - 2\kappa \hat{\beta}(j) \sum_i r_i^{(k)} X_{ij}$$

$$= \kappa^2 (\mathrm{Corr}(X_{\cdot,j}, r^{(k)}) \sqrt{\mathrm{Var}(r^{(k)})})^2 - 2\kappa \, \mathrm{Corr}(X_{\cdot,j}, r^{(k)}) \sqrt{\mathrm{Var}(r^{(k)})} \, \mathrm{Cov}(X_{\cdot,j}, r^{(k)})$$

$$= \kappa^2 (\mathrm{Corr}(X_{\cdot,j}, r^{(k)}) \sqrt{\mathrm{Var}(r^{(k)})})^2 - 2\kappa (\mathrm{Corr}(X_{\cdot,j}, r^{(k)}) \sqrt{\mathrm{Var}(r^{(k)})})^2$$

$$= (\kappa^2 - 2\kappa)(\mathrm{Corr}(X_{\cdot,j}, r^{(k)}) \sqrt{\mathrm{Var}(r^{(k)})})^2$$

which is non-positive since $\kappa \in]0,1]$. Even more, this again shows why a larger step size leads to faster convergence and indeed that L_2-Boosting would not be able to reduce the RSS any more once all columns had a perfect correlation of zero with the current residual (which clearly would never happen in practice).

This proves why for L_2-Boosting, the variable with the most absolute correlation with the current residual is selected and why L_2-Boosting gradually decreases the training error. In fact, the consistency result of L_2-Boosting in [Bühlmann, 2006, Thm. 1] does not incorporate the learning rate κ since it can be bounded from above by 1 as shown in [Bühlmann, 2006, Sec. 6.3]. Further results on the decreasing behaviour of the training error in L_2-Boosting are also given in [Freund et al., 2017, Thm. 2.1].

Remark 10.4.2 (**Fast update formula**). *As stated in Zhao and Yu [2007], in the case of L_2-Boosting where the best variable is the one with the highest absolute correlation with the current residual, the computational time can be significantly decreased by using the update formula*

$$(r^{(k+1)})^T X_{\cdot,j} = (r^{(k)} - \kappa\hat{\beta}(\hat{j}_k) X_{\cdot,\hat{j}_k})^T X_{\cdot,j} = (r^{(k)})^T X_{\cdot,j} - \kappa\hat{\beta}(\hat{j}_k) X_{\cdot,\hat{j}_k}^T X_{\cdot,j}$$

for the correlations (again assuming normalized columns).

We cannot proceed in the same manner if $\tilde{L}$ is a loss function that for example belongs to the hard or a localized ranking problem. Why should a good ranking performance of the baselearner with the residuals as responses be related to a good ranking performance of the strong model for the original response values?

Thus, for any loss function $\tilde{L}$ which has a structure as in (10.4.2) such that $\tilde{L}(|r|)$ is monotonically increasing, we can transfer the selection criterion from component-wise least squares Boosting to SingBoost by taking

$$\hat{j}_k = \operatorname*{argmin}_j \left(\sum_i \tilde{L}(r_i^{(k-1)} - X_{i,j}\hat{\beta}(j)) \right).$$

For losses that cannot be written as in (10.4.2) like ranking losses, we propose the following criterion:

$$\hat{j}_k = \operatorname*{argmin}_j \left(\sum_i \tilde{L}(Y_i, X_i\hat{\beta}^{(k-1)} + \kappa\hat{\beta}(j)) \right), \tag{10.4.4}$$

in other words, we directly evaluate the ranking performance of the resulting strong model respecting the learning rate κ. This criterion directly corresponds to some kind of **secant approximation** of $\nabla\tilde{L}$.

Remark 10.4.3. *An approach where secants are used in literature may be found in Gulcehre et al. [2017], but in a different setting. They used a secant approximation of gradients to extend a stochastic gradient descent to their* `AdaSecant` *algorithm. Thus, the two main differences to our work are that their target function has a unique gradient and that the secants are used to approximate it whereas we do not face unique gradients and therefore the term "secant" should only be used implicitly.*

Remark 10.4.4 (**Concordance and correlation**). *Let us once more concern the Gradient Boosting algorithm with the piece-wise linear surrogate, see section 8.4. The algorithm ended up selecting the column which itself or whose negative is most concordant with the current* *residual. Of course, if a noise variable by chance has the property that its order or its negative order perfectly resembles that of the current residual, it would be chosen by the Gradient Boosting algorithm. But in fact, in this case, the absolute value of the correlation of the current residual and the respective column is 1, so L_2-**Boosting would also select this variable**!*

However, this clearly does not disprove either SingBoost or L_2-Boosting since this case is an extreme one which is unlikely to happen. But it is important to note the strong relation of correlation- and concordance-based variable selection. Coming back to the proposed strategy in (10.4.4), it is not meaningless since firstly, we do not restrict ourselves to ranking losses but allow $\tilde{L}$ to be an arbitrary loss function and secondly, experiments showed that L_2-Boosting, SingBoost with the variable selection as in (10.4.4) and SingBoost where we indeed compared the least squares baselearner models based on each single column with the current residual actually ***do not lead to the same empirical column measures****.*

This may be constituted since correlation and concordance, although being closely related, are not the same statistical measures of agreement.

10.5 Asymptotic properties of SingBoost

Although SingBoost seems to be quite different from standard L_2-Boosting, we are able to modify the theorems of estimation and prediction consistency (see [Bühlmann, 2006, Thm. 1] resp. [Bühlmann and Van De Geer, 2011, Thm. 12.2]) by adding a so-called **Corr-min condition**.

[Bühlmann, 2006, Thm. 1] use the following scheme going back to Temlyakov (Temlyakov [2000]): For

$$\langle f, g \rangle_{(n)} := \frac{1}{n} \sum_i f(X_i) g(X_i),$$

i.e., an empirical version of the inner product $\langle f, g \rangle$ on the space L_2. Bühlmann identifies L_2-Boosting as the iterative scheme

$$\begin{aligned} \hat{R}_n^0 f = f, \quad \hat{R}_n^1 f = f - \langle Y, g_{\widehat{j_1}} \rangle_{(n)} g_{\widehat{j_1}}, \\ \hat{R}_n^m f = \hat{R}_n^{m-1} f - \langle \hat{R}_n^{m-1} f, g_{\widehat{j_m}} \rangle_{(n)} g_{\widehat{j_m}} \ \forall m \geq 2, \end{aligned} \tag{10.5.1}$$

where g_j represents the baselearner w.r.t. variable j (it is assumed that $\langle g_j, g_j \rangle_{(n)} = 1$ for all

j) and $\hat{R}_n^m$ is the empirical residual after the $m-$th iteration.

For the special case of L_2-Boosting, the selected variables are, as described in Bühlmann [2006],

$$\widehat{j_1} = \underset{j}{\operatorname{argmax}}(|\langle Y, g_j\rangle_{(n)}|), \quad \widehat{j_m} = \underset{j}{\operatorname{argmax}}(|\langle \hat{R}_n^{m-1} f, g_j\rangle_{(n)}|) \ \forall m \geq 2. \tag{10.5.2}$$

So, the only difference between L_2-Boosting in SingBoost in the language of the Temlyakov scheme is the variable selection in the singular iterations, leading to other residuals according to (10.5.1). We mimic [Bühlmann, 2006, Thm. 1] and propose the following result for **random design of the regressor matrix**:

Theorem 10.5.1 (**Estimation consistency of SingBoost**). *Let us define the model*

$$Y_i = f_n(X_i) + \epsilon_i = \sum_{j=1}^{p_n} \beta_{n,j} X_{i,j} + \epsilon_i$$

for $X_i \in \mathbb{R}^{p_n}$ i.id. with $I\!E[||X_{\cdot,j}||^2] = 1$ for all $j = 1, ..., p_n$ and error terms $\epsilon_1, ..., \epsilon_n$ that are i.id., independent from all X_i and mean-centered. Let X denote a new observation which follows the same distribution as the X_i, independently from all X_i. Assume

(E1) $\exists \xi \in]0,1[, C \in]0,\infty[$ *such that* $p_n = \mathcal{O}(\exp(Cn^{1-\xi}))$ *for* $n \to \infty$,

(E2) $\sup_n \left(\sum_{j=1}^{p_n} |\beta_{n,j}|\right) < \infty$,

(E3) $\sup_{j,n}(||X_{\cdot,j}||_\infty) < \infty$,

(E4) $I\!E[||\epsilon||^\delta] < \infty$ *for* $\delta > \frac{4}{\xi}$ *with* ξ *from (E1),*

(E5) $\exists a > 0 \ \forall m, n \ \exists \tilde{C} \geq a$:

$$|\langle \hat{R}_n^{m-1} f, g_{\widehat{j_m}}\rangle_{(n)}| \geq \tilde{C} \sup_j(|\langle \hat{R}_n^{m-1} f, g_j\rangle_{(n)}|). \tag{10.5.3}$$

Then, denoting the SingBoost model after the $m-$th iteration based on n observations by $\hat{f}_n^{(m)}$, it holds that

$$I\!E_X[||\hat{f}_n^{m_n}(X) - f_n(X)||^2] = o_P(n^0)$$

for $n \to \infty$ provided that $(m_n)_n$ satisfies that $m_n \to \infty$ sufficiently slowly for $n \to \infty$.

Proof. *The proof follows the same steps as the proof of [Bühlmann, 2006, Thm. 1]. Lemma 1 of Bühlmann [2006] does not include variable selection and indeed holds for our case. [Bühlmann, 2006, Lem. 2] also holds since an upper bound of the expression* $\langle g_{\widehat{j_m}}, g_j\rangle_{(n)}$ *is used which is independent of* $\widehat{j_m}$.

Using (E5), equation (6.13) in Bühlmann [2006] changes to

$$|\langle \tilde{R}_n^m f, g_{\widehat{j_{m+1}}}\rangle| \geq \tilde{C}\sup_j(|\langle \tilde{R}_n^m f, g_j\rangle|) - (1+\tilde{C})(2.5)^m\zeta_n C_* \tag{10.5.4}$$

with ζ_n, C_* *from [Bühlmann, 2006, Lem. 2], on the set* $A_n := \{\omega \mid |\zeta_n(\omega)| < 0.5\}$. *Now, we have to consider an analog of the set* B_n *defined on p. 580 in Bühlmann [2006], namely*

$$\tilde{B}_n := \{\omega \mid \sup_j(|\langle \tilde{R}_n^m f, g_j\rangle|) > 2(\tilde{C})^{-1}(1+\tilde{C})(2.5)^m\zeta_n C_*\}, \tag{10.5.5}$$

and can conclude that, using (10.5.4), it holds that

$$|\langle \tilde{R}_n^m f, g_{\widehat{j_m}}\rangle| \geq 0.5\tilde{C}\sup_j(|\langle \tilde{R}_n^m f, g_j\rangle|)$$

on $A_n \cap \tilde{B}_n$. *That means, we have* $b = 0.5\tilde{C} \geq 0.5a > 0$ *in equation (6.2) of Bühlmann [2006]. This is less than the constant* 0.5 *established in Bühlmann [2006], but it does not matter as long as it is bounded away from zero. Then, we get an analog of (6.15) of Bühlmann [2006] which is*

$$||\tilde{R}_n^m f|| \leq B(1+mb^2)^{-b/2(2+b)} = o(n^0)$$

for $n \to \infty$ *provided that* $m_n \to \infty$ *as assumed.*

On $\tilde{B}_n^c$, *we proceed in the same manner as Bühlmann [2006] since* $\tilde{B}_n$ *can be identified with the set notated as* $\mathcal{A}(\mathcal{D}, b_m)$ *in Temlyakov [2000] which supplies a recursive scheme, leading to the same upper bound as in (6.16) in Bühlmann [2006] since the variable selection is absorbed when applying the "norm-reducing property" given in (6.3) in Bühlmann [2006].*

The rest of the proof follows exactly the same steps as the proof in Bühlmann [2006] since $\widehat{j_m}$ *does only appear again when bounding the quantity* $A_n(m) = ||\hat{R}_n^m f - \tilde{R}_n^m f||$, *but* $||g_{\widehat{j_m}}||$ *is bounded by one and [Bühlmann, 2006, Lem. 2] holds anyway, delivering the last inequality on page 580 in Bühlmann [2006].*

□

Note that the equality in the second last display on page 579 in Bühlmann [2006] is not correct and would imply that the residual after the $m-$th iteration would be most correlated with the column selected in this iteration (which clearly had the consequence that L_2-Boosting selected the same variable in each iteration). Replacing $\widehat{j_m}$ by $\widehat{j_{m+1}}$, inequality (6.13) in Bühlmann [2006] also slightly changes as well as (6.14) which then proves inequality (6.2) in Bühlmann [2006].

Remark 10.5.1 (Step size). *The proof implicitly assumed the learning rate $\kappa = 1$. Of course, we can follow the same lines as in [Bühlmann, 2006, Sec. 6.3] to extend the main result to the case of learning rates $\kappa < 1$.*

Remark 10.5.2. *We did not explicitly use the fact that only each $M-th$ iteration is a singular iteration, i.e., for all other m, nothing changes compared to [Bühlmann, 2006, Thm. 1]. Indeed, asymptotically, the statement of the theorem does not change, only the bound for finite n would be affected, but we do not see an advantage in a tedious case-by-case-analysis.*

The constant $a > 0$ from the Corr-min condition (E5) was absorbed during the proof when one forced the sequence $(m_n)_n$ to grow sufficiently slowly to get an upper bound of order $o_P(n^0)$ for $||\tilde{R}_n^m f||$. However, in the following theorem that is based on [Bühlmann and Van De Geer, 2011, Thm 12.2], a bit more work is necessary to adapt it to SingBoost, directly appearing in the convergence rate at the end.

Theorem 10.5.2 (Prediction consistency of SingBoost). *Let us define the model*

$$Y_i = f_n(X_{n;i}) + \epsilon_i = \sum_{j=1}^{p_n} \beta_{n,j} X_{n;i,j} + \epsilon_i$$

*for **fixed design** of the regressor matrix and error terms $\epsilon_1, ..., \epsilon_n$ that are i.id., independent from all X_i and mean-centered. Let X denote a new observation which is independent from all X_i. Assume*

(P1) *The number p_n satisfies $\frac{\ln(p_n)}{n} \to 0$ for $n \to \infty$,*

(P2) *The true coefficient vector is sparse in terms of the l_1-norm, i.e.,*

$$||\beta_n||_1 = \sum_j |\beta_{n,j}| = o\left(\sqrt{\frac{n}{\ln(p_n)}}\right)$$

for $n \to \infty$,

(P3) *It holds that*

$$\frac{1}{n} \sum_i X_{n;i,j}^2 = 1 \; \forall j = 1, ..., p_n$$

and

$$\frac{||X\beta||_2^2}{n} = \frac{1}{n}\sum_i f_n^2(X_{n;i}) \leq C < \infty$$

for all $n \in \mathbb{N}$,

(P4) *The errors are i.id. $\mathcal{N}(0,\sigma^2)-$distributed for all $n \in \mathbb{N}$*

and the Corr-min condition (E5) of the previous theorem. Then for

$$m_n \longrightarrow \infty, \quad m_n = o\left(\sqrt{\frac{n}{\ln(p_n)}}\right)$$

for $n \to \infty$, it holds that

$$\frac{||X(\hat{\beta}_n^{(m_n)} - \beta_n)||_2^2}{n} = o_P(n^0)$$

as $n \to \infty$ where $\hat{\beta}_n^{(m_n)}$ is the coefficient vector based on n observations and after the m_n-th SingBoost iteration.

Proof. *We follow the same steps as in the proof of [Bühlmann and Van De Geer, 2011, Thm. 12.2] which again is based on a Temlyakov scheme as before. First, [Bühlmann and Van De Geer, 2011, Lem. 12.1] needs to be modified to:*

Lemma 10.5.1. *If there exists $0 < \phi < 0.5$ such that*

$$\max_j(|\langle \hat{R}^{m-1}f, X_{\cdot,j}\rangle_{(n)}|) \geq 2\Delta_n\phi^{-1}(\tilde{C})^{-1},$$

then it holds that

$$|\langle \hat{R}^{m-1}f, X_{\cdot,\widehat{j_m}}\rangle_{(n)}| \geq \tilde{C}(1-\phi)\max_j(|\langle \hat{R}^{m-1}f, X_{\cdot,j}\rangle_{(n)}|),$$
$$|\langle Y - \hat{f}^{(m-1)}, X_{\cdot,\widehat{j_m}}\rangle_{(n)}| \geq \tilde{C}(1-\phi/2)\max_j(|\langle \hat{R}^{m-1}f, X_{\cdot,j}\rangle_{(n)}|).$$

The proof just modifies the proof of [Bühlmann and Van De Geer, 2011, Lem. 12.1] at obvious places.

The quantities a_k and d_k defined on page 422 in Bühlmann and Van De Geer [2011] indeed depend on the selected column, but the inequality in (12.28) in Bühlmann and Van De Geer [2011] holds as well and since the relationship of a_k and d_k does not change with the concrete column selection and since both are bounded from above at the end, the singular iterations do not affect the proof structure when facing a_k and d_k.

[Bühlmann and Van De Geer, 2011, Lem. 12.2] has to be modified to:

Lemma 10.5.2. *If there exists $0 < \phi < 1/2$ such that*

$$\max_j(|\langle \hat{R}^{m-1}f, X_{\cdot,j}\rangle_{(n)}|) \geq 2\Delta_n \phi^{-1}(1-\phi/2)^{-1}(\tilde{C})^{-1},$$

then it holds that

$$a_m \leq a_{m-1} - (1-\phi)d_m^2.$$

The proof also follows the same lines as the proof of [Bühlmann and Van De Geer, 2011, Lem. 12.2]. As we see in the following steps, it would not be meaningful to absorb $\tilde{C}$ (or a) already in ϕ.

(12.33) in Bühlmann and Van De Geer [2011] changes to

$$d_k \geq \frac{(1-\phi/2)a_{k-1}\tilde{C}}{b_{k-1}}$$

and instead of (12.34) in Bühlmann and Van De Geer [2011], we get

$$a_k b_k^{-2} \leq a_{k-1}b_{k-1}^{-2}(1 - C_\phi^2 a_{k-1}b_{k-1}^{-2}\tilde{C})$$

with C_ϕ as in Bühlmann and Van De Geer [2011]. Instead of $B_n(m)$ in (12.36), we define

$$\tilde{B}_n(m) := \bigcap_{k=1}^{m}\{\max_j(|\langle \hat{R}^{k-1}f^0, X_{\cdot,j}\rangle_{(n)}|) \geq 2(\tilde{C})^{-1}\phi^{-1}(1-\phi/2)^{-1}\Delta_n\}$$

and, analogously to (12.37) in Bühlmann and Van De Geer [2011], we conclude that

$$a_m b_m^{-2} \leq ||\beta_n^0||_1^{-2}(1 + C_\phi^2 m\tilde{C})^{-1}.$$

Inequality (12.38) in Bühlmann and Van De Geer [2011] is modified to

$$a_m \leq a_{m-1}\left(1 - \frac{D_\phi d_m \tilde{C}}{b_{m-1}}\right)$$

where we define

$$\tilde{D}_\phi := (1-\phi)(1-\phi/2)\tilde{C}$$

so that $\tilde{C}$ gets absorbed. One may ask if one could proceed also with D_ϕ, but then the third-last display on page 424 in Bühlmann and Van De Geer [2011] would require

$$1 + D_\phi d_m/b_{m-1} - D_\phi d_m\tilde{C}/b_{m-1} - D_\phi^2 d_m^2\tilde{C}/b_{m-1}^2 \leq 1$$

which is not evident (note that the same recursion for the b_m as in [Bühlmann and Van De Geer, 2011, p. 424] is valid). Maybe one could distinguish between the case where the inequality holds, i.e., where

$$\tilde{C} \geq \frac{1}{1 + D_\phi d_m b_{m-1}}$$

holds and the other case, but we do not see any advantage.

However, we can also apply Temlyakov's lemma [Bühlmann and Van De Geer, 2011, Lem. 12.3] and get

$$a_m^{2+D_\phi} \leq (1 + C_\phi^2 m\tilde{C})^{-\tilde{D}_\phi}$$

as analog to the last inequality on page 424 in Bühlmann and Van De Geer [2011]. The bound $C_\phi^{-2\tilde{D}_\phi} \leq 2$ *is still valid and we get the inequality*

$$a_m^{2+\tilde{D}_\phi} \leq 2m^{-\tilde{D}_\phi}$$

on the set $\tilde{B}_n(m)$.

On the complement of $\tilde{B}_n(m)$, *we follow exactly the same steps as in Bühlmann and Van De Geer [2011], resulting in the factor* $(\tilde{C})^{-1}$ *before the first summand of the right hand side in the last display on page 425.*

Finally, we get

$$||\hat{R}^m f||_{(n)}^2 \leq \max\left(2m^{-\frac{(1-\phi)(1-\phi/2)\tilde{C}}{2+(1-\phi)(1-\phi/2)\tilde{C}}}, 2(\tilde{C})^{-1}\phi^{-1}(1-\phi/2)^{-1}\Delta_n(||\beta_n||_1 + m\gamma_n) + m\Delta_n\right)$$

with γ_n *as in (12.30) in Bühlmann and Van De Geer [2011].*

Using the fact that ϕ *and* D_ϕ *are fixed, that* $\gamma_n = \mathcal{O}_P(n^0)$ *and* $\Delta_n = \mathcal{O}_P\left(\sqrt{\ln(p_n)/n}\right)$ *and invoking (P2) and the assumptions on* $(m_n)_n$, *we asymptotically conclude that*

$$||\hat{R}^m f||_{(n)}^2 = o_P(n^0).$$

□

Remark 10.5.3. *The last line in the proof of theorem 10.5.2 means that although the convergence rate is indeed slower for a fixed n due to the constant* $\tilde{C}$ *that enters the exponent resp. that enters as factor in the arguments of the maximum,* ***we asymptotically do not lose anything compared to*** L_2-***Boosting when performing SingBoost!***

Remark 10.5.4. *Again, we did not perform a case-by-case-analysis w.r.t. m using that only each M−th iteration is a singular iteration. However, since we get a bound for each* $||R^m f||_{(n)}^2$, *we do not see any advantage in distinguishing cases w.r.t. m here.*

Remark 10.5.5 (Discussion of the Corr-min condition (E5)). *We admit that we do not know if the Corr-min condition excludes some loss functions to be used as target loss* $\tilde{L}$. *Assuming for the moment that we had a Gradient Boosting algorithm for* $\tilde{L}$ *or maybe even* *another learning strategy that selects variables which lead to a good (still training!) performance w.r.t. $\tilde{L}$, then we generally cannot guarantee that these variables, although improving the performance, are strongly enough correlated with the current residual to satisfy the Corr-min condition.*

This is already true if one concerns quantile Boosting. If one uses the implementation of quantile Boosting in the package `mboost` *by using either* `QuantReg()` *or, in the case of L_1-Boosting,* `Laplace()` *as family object, then one faces the issue that only the intercept is being selected in some (or even in all) iterations. Therefore, if we based SingBoost on quantile Boosting, there would have been no chance to justify the Corr-min condition as long as selecting only the intercept is allowed.*

But in fact, the SingBoost algorithm always uses simple least squares models as baselearners which are directly based on the correlation of the current residual with each single variable. Therefore, if we have variables that are (nearly) perfectly uncorrelated with the current residual, then the resulting coefficient would be (in a very close neighborhood of) zero. Although we cannot exclude this case, it would be very unlikely that such a quasi-zero-baselearner would pass the rejection step and enter the SingBoost model.

Remark 10.5.6. *Until here, we motivated the application of L_2-Boosting as basis algorithm for SingBoost with the computational performance, see 10.3.9. Now, we see that L_2-Boosting is indeed a very good candidate algorithm to serve as the underlying algorithm in SingBoost to get a solid theoretical foundation.*

Remark 10.5.7 (Variable selection consistency). *In contrast to the Lasso for which there exist results on variable selection consistency requiring a beta-min condition and an irrepresentability condition (see Bühlmann and Van De Geer [2011]), Vogt [2018] recently showed that even if the regressor matrix satisfies the restricted nullspace property, i.e., the intersection set of the nullspace of X and the cone*

$$C(S,L) := \{\beta \in \mathbb{R}^p \mid ||\beta_{(S^0)^c}||_1 \leq L||\beta_{S^0}||_1\}$$

*is just the element 0_p, L_2-**Boosting is not variable selection consistent**. They provided an example for a regressor matrix with the restricted nullspace condition but where L_2-Boosting reliably selects the wrong columns.*

However, the proof is based on the fact that L_2-Boosting always selects the variable which is most correlated with the current residual vector. It is not evident that SingBoost would always fail on their example for each loss function $\tilde{L}$.

10.6 Coefficient paths for SingBoost

Implementations of sparse learning algorithms like the Lasso or the elastic net are capable to supply the user with a regularization path of the coefficients, like in the package `glmnet` (Friedman et al. [2010]). Those paths show the evolution of the coefficients for different values of the regularization parameter λ, more precisely, in the cited package, one can (amongst other options) let the values of the non-zero coefficients be plotted against the natural logarithm of the sequence of regularization parameters (in an ascending order, i.e., the paths evolve from the right to the left). We already mentioned the concept of stability paths for Stability Selection in the R−package `stabs` (Hofner and Hothorn [2017], Hofner et al. [2015], Thomas et al. [2018]) in section 2.3.

For Boosting, similar coefficient paths are available in the package `mboost`. Since Boosting does not incorporate a regularization parameter, those Boosting coefficient paths are given by the values of the (non-zero) coefficients vs. the current Boosting iteration. Clearly, since only one variable, not counting the intercept, changes in one Boosting iteration, the coefficient paths, except one, are flat when jumping from on iteration to the next.

Since coefficient or stability paths are a very useful graphical tool for visualizing the behaviour of the model, it is desirable to plot the coefficient paths for SingBoost in a similar fashion. The difficulty in the implementation is not how to handle the coefficients in the singular iterations but how to extract the evolution of the coefficients from the output of `glmboost` when performing the L_2−Boosting iterations.

We briefly list some details:

1.) We start with an empty matrix `coeffpath` with two rows and columns as many as the Boosting iterations. In each iteration, we want to insert the fitted coefficient in the first row in the respective column and in the second row, the variable name has to be inserted as a dictionary for later plotting.

2.) We use the command `coef(res,aggregate='none')` after running `glmboost` to get a list with as many elements as non-zero coefficients, including the intercept. Each list element is a vector having as many entries as the number of iterations. For each iteration, the computed variable enters as the corresponding component in the vector in the respective list, so most entries of these vectors are zero. In other words, the m−th components is zero for all but one list (not counting the intercept list), for all $m = 1, ..., m_{iter}$. We extract the relevant entries corresponding to the $(M-1)$ Boosting iterations of `glmboost` as well as the variables that have been chosen in each iteration.

3.) We implemented the function `path.singboost` which is just SingBoost but where the coefficients paths are saved. The reason why we essentially copied the `singboost` implementation is that we will perform a Stability Selection later, so the coefficient paths of the single SingBoost models are useless in this context, but the computation (though just consuming very little time) cannot be easily avoided.

4.) Getting the matrix `coeffpath` as part of the `path.singboost` output, we can apply the function `singboost.plot` to obtain the plotted coefficient paths. In the latter function, we determine the number of different non-zero coefficients, not counting the intercept, generating a list containing as many empty lists as the respective number, each indicated with one of the variable names. For each variable, we fill the components of the respective matrix corresponding to the iteration number in which the coefficient has been updated with the computed coefficient, resulting in a similar structure as in the output of `glmboost`.

5.) We let `path.singboost` also report the intercept path, but do not intend to plot it since it may differ from the one of L_2-Boosting even if we perform L_2-Boosting using `singboost`. By default, using the `plot` command for an `mboost` object draws the path of the intercept disregarding the offset value which indeed can be avoided using the extra argument `off2int=TRUE` which results in adding the offset value to the intercept. However, there is no sense in defining a cutoff for SingBoost, so the cutoff computed by `glmboost` is already contained in the SingBoost intercept, therefore we avoid confusion by not plotting the intercept path when running `singboost.plot`, besides that we do not think the intercept path to be interesting at all. Nonetheless, we think that reporting the fitted intercept is more reasonable than the intercept in the `glmboost` output which is not the intercept used for prediction, but differs from it by the offset value. See again 10.3.13.

6.) We essentially just need three input arguments (`mod` refers to a `path.singboost` model) :

```
singboost.plot(mod,M,m_iter)
```

The coefficient paths in figure 10.3 have been drawn using the `plot` command in the package `glmboost`. The coefficient paths in figure 10.4 were generated by our function `singboost.plot` on the same data set and with the same configurations as in `glmboost` (especially setting `singfamily=Gaussian()`).

Remark 10.6.1 (Behaviour of coefficient paths). *One may ask if the coefficient paths, not regarding the intercept path, are always monotonic. This is not true. One can easily get non-monotonic coefficient paths when using a high step size in L_2-Boosting. The reason is*

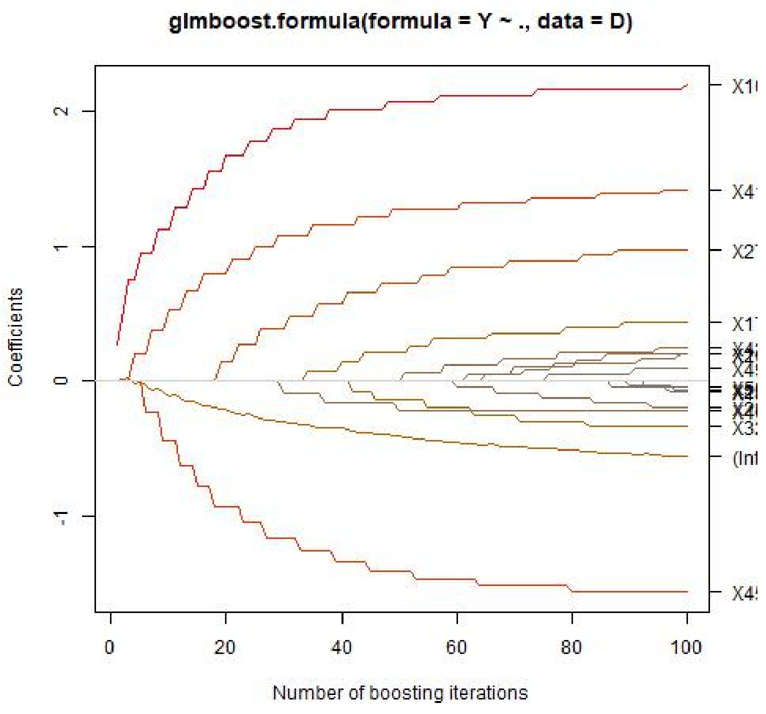


Figure 10.3: Coefficient paths for L_2-Boosting

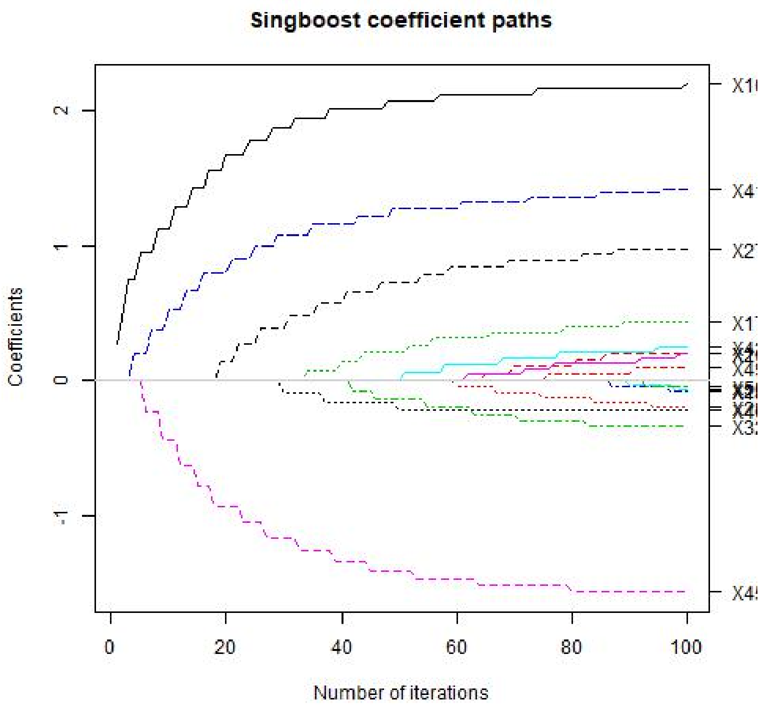


Figure 10.4: Coefficient paths for SingBoost

that large step sizes lead to a fast de-correlation of the variables with the residuals. Once the correlations are very close to zero, a sign change is possible and if the variable whose sign has changed becomes the most correlated variable in an absolute sense with the current residual, its coefficient path will lose the monotonic behaviour. There is some literature where the (non-)monotonicity of coefficient paths for stage-wise procedures has been discussed (see Efron et al. [2004], Tibshirani [2015], Ehrlinger et al. [2012]). As for the effect of the step size, Rosset et al. [2004] emphasize that a smaller step size just causes the non-monotonicity to occur later than with a low learning rate.

Remark 10.6.2. *Since SingBoost just potentially selects different variables but the computation of the coefficients is the same as in L_2-Boosting, SingBoost coefficient paths can be non-monotone due to the same reasons, and the non-monotonicity can also occur in singular iterations.*

10.7 Conclusion

This chapter provided deeper insights into (sparse) model selection, especially for the case that the solutions of algorithms based on different loss functions have to be compared. We have seen that domination and singular parts similarly can be defined for the corresponding empirical column measures. Moreover, we provided arguments that this phenomenon is a standard issue when applying sparse model selection procedures.

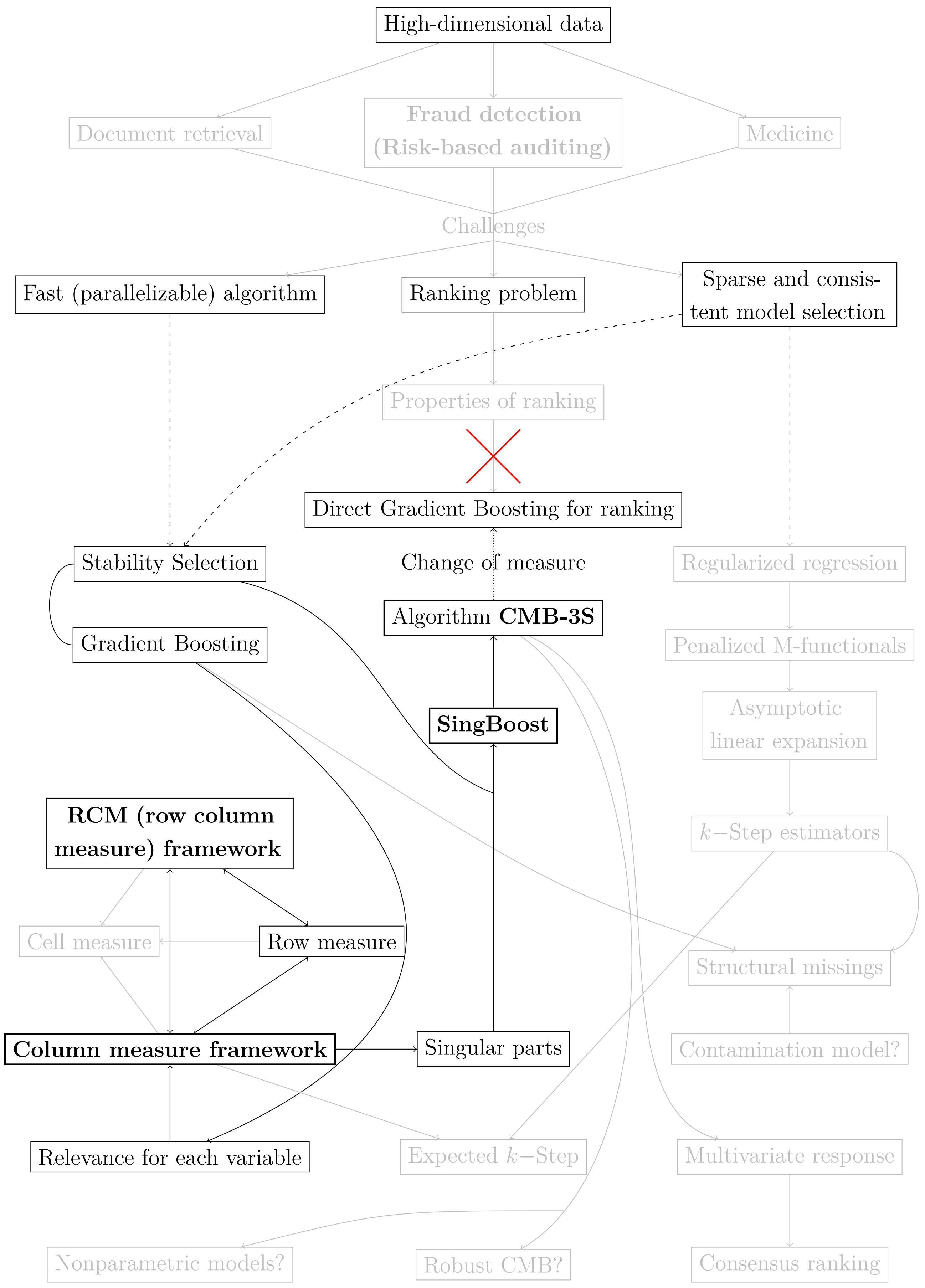
We provided and implemented the ”gradient-free Gradient Boosting” algorithm SingBoost that combines standard L_2-Boosting and rejection sampling to account for loss functions $\tilde{L} \neq L_2$. We showed that the SingBoost algorithm is also applicable for ranking problems although the loss functions do not satisfy the regularity requirements that are needed for Boosting and provided coefficient paths for SingBoost and discussed how the coefficient updates of L_2-Boosting and SingBoost are performed, including the case where the loss function is not a strictly monotonically increasing function of the absolute value of the residual.

As a theoretical foundation of SingBoost, we provided asymptotic results on estimation and prediction consistency even in very high dimensions (the number of predictors is allowed to grow (nearly) as fast as $\exp(n)$) that are based on similar results provided by Bühlmann for L_2-Boosting. We have seen that these results require an additional Corr-min condition for SingBoost which we think is not too strict for our L_2-Boosting-based algorithm due to the direct dependence of the coefficients on the correlation with the residual.

It remains to investigate how the models proposed by SingBoost have to be worked with.

Part IV

Column Measure Boosting, its variants and applications


High-dimensional data
Document retrieval
Fraud detection (Risk-based auditing)
Medicine
Challenges
Fast (parallelizable) algorithm
Ranking problem
Sparse and consistent model selection
Properties of ranking
Direct Gradient Boosting for ranking
Stability Selection
Change of measure
Regularized regression
Gradient Boosting
Algorithm CMB-3S
Penalized M-functionals
SingBoost
Asymptotic linear expansion
RCM (row column measure) framework
k−Step estimators
Cell measure
Row measure
Structural missings
Column measure framework
Singular parts
Contamination model?
Relevance for each variable
Expected k−Step
Multivariate response
Nonparametric models?
Robust CMB?
Consensus ranking

This both conceptual and theoretical part combines SingBoost with Stability Selection and, more abstractly, the concept of row measures and column measures.

So far, we developed the algorithm SingBoost that may select variables that L_2-Boosting does ignore. But just applying one SingBoost model and taking the computed empirical column measure does not suffice since this empirical column measure is not yet sufficiently adapted to the target loss $\tilde{L}$, not yet sparse enough and - as Boosting or Lasso models generally are - highly unstable.

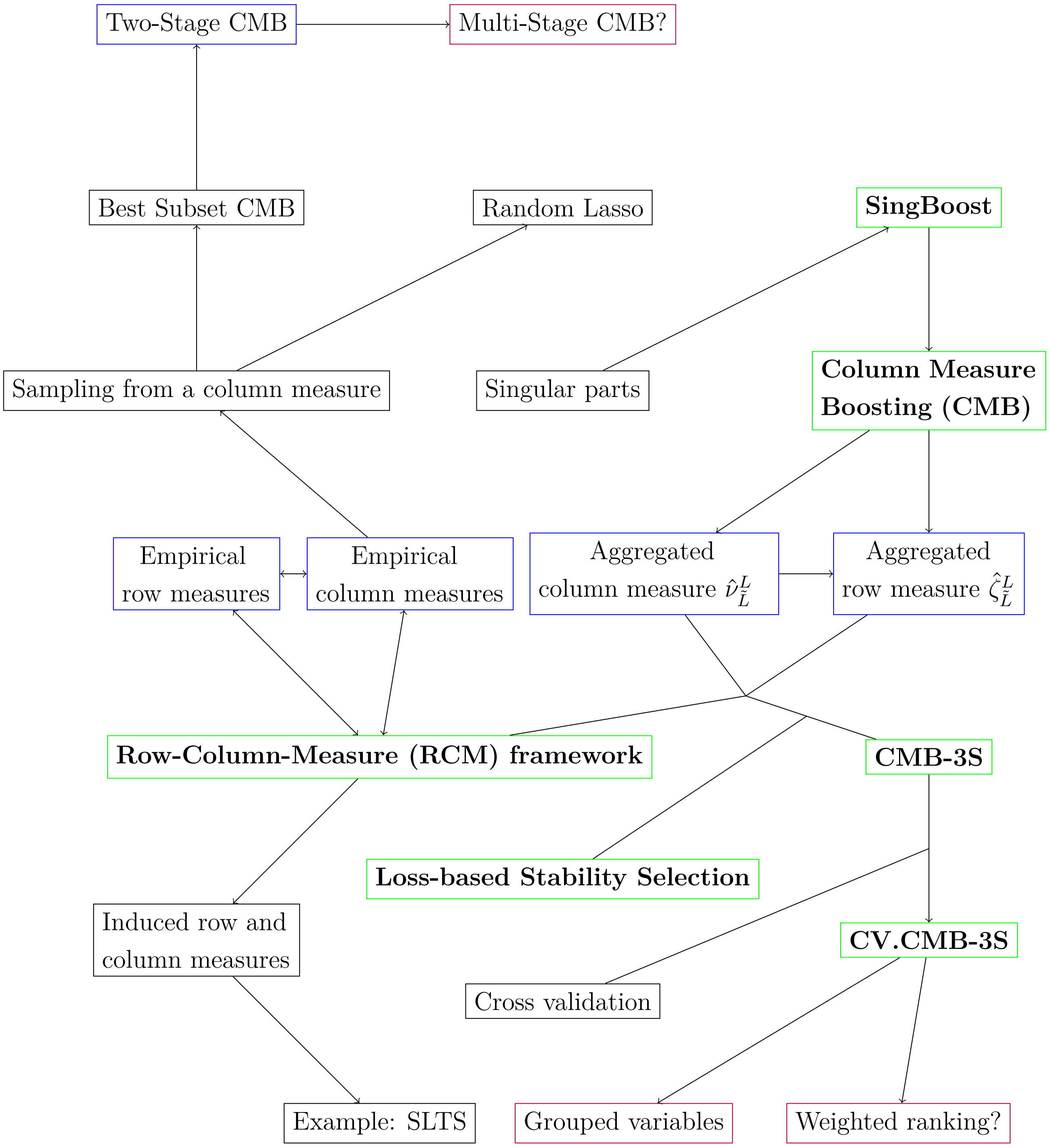
To fix the first problem, we aggregate several empirical SingBoost column measures in a special manner to better account for $\tilde{L}$ and to stabilize the empirical singular part. In fact, this "Column Measure Boosting" (CMB) reweights the priority of the selected variables, so we essentially perform a change of measure.

For cases in which the target loss $\tilde{L}$ is expensive to evaluate, we provide an idea for multi-step approaches to combine sampling from empirical column measures and optimization.

After reweighting, our aggregated column measure will be sparsified and stabilized by performing a modified Stability Selection where either the threshold for the selection frequencies or the number of final variables is computed by a grid search and optimization w.r.t. $\tilde{L}$. The whole algorithm will be called CMB-3S. We connect the column measure with the row measure which leads to a so-called "Row column measure (RCM) framework" in which we can exactly embed all learning algorithms with univariate response or without response. We propose a systematic overview of learning algorithms which can be related to sparsity, stability and robustness properties.

To assess the quality of CMB-3S which itself runs on a single subset of the data, one clearly needs to compute some cross-validated error. Mimicking the notation in the `R`$-$package `glmnet`, we will refer to the combination of cross validation and CMB-3S as CV.CMB-3S.

At the end, we list some directions for future research concerning CMB-3S.


Two-Stage CMB
Multi-Stage CMB?
Best Subset CMB
Random Lasso
SingBoost
Sampling from a column measure
Singular parts
Column Measure Boosting (CMB)
Empirical row measures
Empirical column measures
Aggregated column measure $\hat{\nu}_{\tilde{L}}^{L}$
Aggregated row measure $\hat{\zeta}_{\tilde{L}}^{L}$
Row-Column-Measure (RCM) framework
CMB-3S
Loss-based Stability Selection
Induced row and column measures
CV.CMB-3S
Cross validation
Example: SLTS
Grouped variables
Weighted ranking?

Chapter 11

Aggregating SingBoost models

This chapter starts with the algorithms called Random Lasso (Wang et al. [2011]) and Block-Forest (Hornung and Wright [2018]) which already follow the paradigm of sampling from empirical column measures.

As a conceptual contribution, we show that a single SingBoost model is not sufficient since although the singular steps are already based on $\tilde{L}$, a subsequent Stability Selection may delete possibly detected variables from singular parts. Therefore, we essentially need some reweighting according to the loss $\tilde{L}$ which leads to a change of measure, based on special weights that we use to compute an aggregated empirical column measure. This algorithm will be called "Column Measure Boosting" (CMB).

We use the idea of sampling from an empirical column measure at the end of this chapter and propose a two-stage strategy to accelerate CMB if the evaluation of the loss $\tilde{L}$ is computationally expensive which may be extended to a multi-stage procedure.

11.1 The Random Lasso

Stable variable selection in a high-dimensional setting has been considered in Wang et al. [2011]. Their intention was to fix two major weaknesses of the Lasso, namely its property to choose in general at most one predictor when facing highly correlated variables which may not be appropriate in microarray data. Secondly, in such very high-dimensional data with only a few observations, the limitation of the Lasso to select at most n predictors may lead to models that are too sparse for this application area. They proposed an algorithm that overcomes these issues.

In fact, Wang et al. [2011] introduced a two-stage procedure which they called the **Random Lasso**. In the first stage, there are B Bootstrap samples generated from the data (according to the row measure $\zeta^{init} = U(\{1, ..., n\})$) and for each of the samples, one randomly picks some of the columns (say, from a prior uniform column measure ν^{init}). Then, a Lasso is applied, providing coefficients $\hat{\beta}_j^{(b,init)}$. These coefficients are used to compute a **measure of importance for each variable** which in this case is

$$\hat{\nu}_j^{(L_2)} = \frac{1}{B}\left|\sum_b \hat{\beta}_j^{(b,init)}\right|.$$

Note that $\hat{\nu}^{(L_2)}$ can be seen as an **empirical column measure corresponding to the squared loss**. The intuition behind the definition of this measure is that relevant predictors usually get coefficients bounded away from zero in each run where the corresponding columns have been chosen. On the other hand, noise variables get rather low coefficients or even coefficients with different signs on different Bootstrap samples so that their importance measure will be low, especially when they nearly compensate each other in the sum.

The second stage again starts by generating B Bootstrap samples from the data according to the same row measure ζ^{init}, but instead of using some uniform prior column measure, **one draws columns from the column measure $\hat{\nu}^{(L_2)}$ computed in the first stage** and applies the Lasso to the reduced data! Referring to those coefficients as $\hat{\beta}^{(b)}$, the final coefficients are computed by bagging (Breiman [1996]), i.e.,

$$\hat{\beta}_j = \frac{1}{B}\sum_b \hat{\beta}_j^{(b)}.$$

According to Wang et al. [2011], one may replace the Lasso in the second stage by an Adaptive Lasso.

Experiments on microarray data in Wang et al. [2011] show that this Random Lasso can both select more than n predictors and that it can simultaneously choose different highly correlated variables, but assigning appropriately different coefficients to them in contrast to algorithms like the elastic net (Zou and Hastie [2005]).

We referred to this related work because the Random Lasso **explicitly works with an empirical column measure** and therefore perfectly fits into our framework. Additionally, sampling columns is a technique that we will use in section 11.5.

On the other hand, the Random Lasso is adapted to the squared loss and thus not applicable to some other loss $\tilde{L}$. Obviously, Wang et al. [2011] could not use the relative selection frequency as importance criterion like in Stability Selection (Meinshausen and Bühlmann [2010]) since it depends on the random sampling scheme of the columns.

Remark 11.1.1. *Taking the averaged coefficient as importance measure seems to be promising and clever. However, we do not see any possibility to adapt this criterion such that it depends on the loss function $\tilde{L}$ and therefore, we do not see how we could generalize the Random Lasso to other losses like the hard ranking loss.*

11.2 Block forests

Note that reducing data by drawing columns is also done by Random Forests (Breiman [2001]) where for each split, the optimal splitting variable is chosen to be optimal from only a few randomly selected predictors instead of all predictors. The fitted trees on different Bootstrap samples are then aggregated by bagging (Breiman [1996]), i.e., by averaging the predictions for regression or by following the majority vote in classification settings. In fact, the random selection of a subset of columns can be identified with sampling from a prior uniform column measure without replacement.

Hornung and Wright [2018] compared different RandomForest-type learning algorithms for so-called multi-omics data. A priority-based approach is to distribute block-specific weights or even different weights within the blocks for variables which are randomly chosen in advance for each tree to account for the difference in the importance of information of the blocks. The variant **RandomBlock** selects one block and variables from only this block to enforce that the different trees are based on different variables in order to profit from the randomization and to get de-correlated trees. The combination of all approaches is referred to as **BlockForest**, i.e., let every block enter with probability 0.5 each (if no block is chosen, repeat) and then sample only variables from the selected blocks to compute a tree on the reduced data.

A major difference of the Random Lasso and Block Forests is that the column measure from which variables are randomly chosen in the second stage of the Random Lasso is already an empirical column measure which has been computed by aggregating Lasso models. Block Forests do not include such a step, so the column measures from which samples are drawn need to be chosen by cross validation (or maybe even by incorporating expert knowledge).

In fact, let L be the number of blocks (or groups) where each block contains p_l variables. Note that in algorithms that respect a block-wise structure as the Group Lasso (Yuan and Lin [2006]), the Sparse-Group Lasso (Simon et al. [2013]) or BlockForest, a block-wise structure of the regressor matrix as well as of the coefficient vector is given, i.e., we have

$$\beta = \bigotimes_{l=1}^{L} \beta^{(l)}, \quad X = (X^{(1)}, ..., X^{(L)}),$$

with $\beta^{(l)} \in \mathbb{R}^{p_l}$ and $X^{(l)} \in \mathbb{R}^{n \times p_l}$ for $l = 1, ..., L$, see also section 6.2.1 for block structures.

However, as the sampling scheme for BlockForeset is indeed as two-stage procedure, a simple block-wise structure of some column measure does not suffice here. More precisely, we have to work with a **block column measure** which is essentially the same as a column measure but which works with blocks in the same sense as the column measure works with columns. In our notation, we have

$$\nu^{block} : (\{1, ..., L\}, \mathcal{P}(\{1, ..., L\})) \to ([0, L], \mathbb{B} \cap [0, L]),$$

i.e., ν_l^{block} is the probability of selecting block l, that is, all columns that correspond to block l. After having sampled blocks in BlockForest, one proceeds by sampling from usual initial column measures $\nu^{init,l}$, $l = 1, ..., L$.

Remark 11.2.1 (Block-wise column and row measures). *The partition into different block-specific column measures and block measures is necessary due to the sampling scheme in BlockForest. In the "usual" setting where predictors may appear as groups like in the case of categorical predictors, we can similarly define column measures with a block-wise structure provided that either a whole group is selected or not by setting*

$$\nu = \bigotimes_{l=1}^{L} \nu^{(l)} \tag{11.2.1}$$

where the $\nu^{(l)}$ are based on the sets $\{j_1^{(l)}, ..., j_{p_l}^{(l)}\}$ and where $j_k^{(l)}$ represents the column indices of the $l-$th group, so $j_k^{(l)} \neq j_h^{(m)}$ $\forall k \neq h$ or $l \neq m$.

But this notation clearly has the disadvantage that the $\nu^{(l)}$ are no longer measures for $p_l > 1$ since the additivity does not hold since we essentially have a relation like

$$\nu^{(l)}(\{j_1^{(l)}, ..., j_{p_l}^{(l)}\}) = \nu^{(l)}(J)$$

for any $J \subset \{j_1^{(l)}, ..., j_{p_l}^{(l)}\}$ which is true due to the perfect dependency that either all variables from a group are selected or none.

This issue does not appear in BlockForest since the selection probabilities within the blocks are independent.

The block measure that we introduced above exactly accounts for this dependency since it either selects a whole block or not.

The same concept would be necessary when working with ***block row measures*** ζ^{block}*, for example when performing some kind of cluster sampling. It is essentially the same strategy as for blocks of columns, i.e., we need block row measures that select a whole block of rows and then we maybe need block-specific row measures to select observations from the blocks, respectively.*

Finding a suitable theory for working with block-wise column measures and block-wise row measures may be a topic for future research.

11.3 Has $\tilde{L}$ already been respected appropriately?

As briefly shown in sections 2.3 and 2.4, Stability Selection is a sophisticated strategy to reduce the number of false positives that have been chosen either by Boosting or by a Lasso-type algorithm. In fact, one generates a number B of subsamples of the data according to some row measure ζ^{init}, fits a Boosting or Lasso-type model on each of them and defines an appropriate aggregation of the selected models computed on each subsample.

A naïve way to derive this would be to exactly copy the Stability Selection, i.e., to draw subsamples, to compute a SingBoost model on each of them (perhaps until q variables are chosen in each model) and to use the same variable aggregation procedure as for the original Stability Selection by replacing the empirical column measures with $0/1-$column measures and computing their empirical mean.

But this would not reflect our goal appropriately. Firstly, setting the number of variables selected in each Boosting model to q would be counterproductive when trying to find additional predictors in the singular steps and would need some balancing of the frequency M of singular iterations, the number m_{iter} of iterations and q. Secondly, we have a Boosting procedure for the loss L_2, but we want to **solve a structural risk minimization problem w.r.t. the target loss** $\tilde{L}$. This loss function $\tilde{L}$ can easily (or at least in a reasonable time) be evaluated, but fails to have a corresponding Boosting algorithm, e.g. due to the lack of differentiability. Up to now, $\tilde{L}$ only appears in the singular iterations whereas the rest of the Boosting is still an L_2-Boosting. But this does not guard against cases where the singular iterations did not provide (new) useful variables.

Motivated by this issue, we need to modify SingBoost by evaluating the whole model again w.r.t. $\tilde{L}$. In other words, a score needs to be assigned to each model that has been computed on a subsample and our final variable selection must rely on these scores. The next section addresses to the following question:

How can we suitably aggregate the models to a final model using these scores?

11.4 Column Measure Boosting: SingBoost aggregation

Definition 11.4.1. *To emphasize that SingBoost intends to respect the singular parts $J_{\tilde{L}}^{L_2}$, we will denote the empirical column measure fitted by SingBoost on the subsamples $\mathcal{D}^{(sing,b,train)}$ by $(\hat{\nu}_{\tilde{L}}^{L_2})^{(b)}$, respectively for each $b = 1, ..., B^{sing}$.*

The question remains how to evaluate these models. We already denoted the subsamples by $\mathcal{D}^{(sing,b,train)}$, insisting that there will be B^{sing} test sets as well. We think of these test sets $\mathcal{D}^{(sing,b,test)}$ as the rows that are not contained in $\mathcal{D}^{(sing,b,train)}$. This gives us an opportunity to test the **out-of-sample performance** of each of the SingBoost models w.r.t. $\tilde{L}$.

Let $\hat{Y}^{(sing,b,test)}$ be the prediction of the $Y-$column of the test set $\mathcal{D}^{(sing,b,test)}$ based on the model $\hat{f}^{(b)}$ which has been trained on the respective training set $\mathcal{D}^{(sing,b,train)}$. Then, we define the scores assigned to each model as

$$\hat{s}^{(b)} := \tilde{L}(Y^{(sing,b,test)}, \hat{Y}^{(sing,b,test)}). \tag{11.4.1}$$

These scores are used to aggregate the selection frequencies of the variables represented by the empirical column measures $(\hat{\nu}_{\tilde{L}}^{L_2})^{(b)}$. Of course, models that have performed better w.r.t. $\tilde{L}$ should get a higher weight than those which have a rather bad out-of-sample performance. One may think of two strategies to combine the empirical column measures:

i) Take the best relative part of around α of the models and assign equal weight to all models in this quantile. Then, we get weights

$$\hat{w}^{(b)} = I(\hat{s}^{(b)} \leq (\hat{s}^{(b)})_{\lceil \alpha B^{sing} \rceil : B^{sing}}). \tag{11.4.2}$$

As for the notation, writing $x_{i:n}$ for some vector x of length n indicates that we refer to the $i-$th smallest component.

ii) Assign a weight to each of the best models which is anti-proportional to the respective losses, i.e.,

$$\hat{w}^{(b)} := \lceil \alpha B^{sing} \rceil \frac{(\hat{s}^{(b)})^{-1} I(\hat{s}^{(b)} \leq (\hat{s}^{(b)})_{\lceil \alpha B^{sing} \rceil : B^{sing}})}{\sum_b (\hat{s}^{(b)})^{-1} I(\hat{s}^{(b)} \leq (\hat{s}^{(b)})_{\lceil \alpha B^{sing} \rceil : B^{sing}})}. \tag{11.4.3}$$

Remark 11.4.1. *By assumption 2.1.1, it is unlikely to face zero losses in practical applications. However, discrete-valued losses like the hard ranking loss may be zero with a probability greater than zero. Therefore, we recommend to use the weights in (11.4.2) if such a loss is chosen as target loss $\tilde{L}$.*

In the last step, we only need to aggregate the empirical column measures according to the weights, so we get an **aggregated column measure**

$$\hat{\nu}_{\tilde{L}}^{L_2} = \left(\frac{1}{\lceil \alpha B^{sing} \rceil} \sum_b \hat{w}^{(b)} I((\hat{\nu}_{\tilde{L}}^{L})_j^{(b)} > 0) \right)_{j=1}^{p}. \tag{11.4.4}$$

Actually, this is nothing but the relative part of the best models in which the $j-$th variable is contained, respectively. We use this aggregation instead of a simple weighted sum because the overall selection frequency of variables in potential singular parts is clearly limited from above where the bound depends on M. Thus, a simple weighted sum would not reflect the importance of predictors that are only chosen in the singular steps. Note that we indeed again used 0/1-column measures in the aggregation step.

We put all these ideas into the following algorithm. Since the computed model at the end is still unstable but since we can interpret the aggregation as some sort of Boosting iteration (where not necessarily the best weak model but a set of best models is being selected, see also part V for further details on that approach) to get an improved approximant of the column measure $\nu^{(\tilde{L})}$ (roughly speaking, the raw SingBoost models may be regarded as "weak models"), we call this procedure **Column Measure Boosting (CMB)** (algorithm 7).

Remark 11.4.2. *If there already exists a $\tilde{L}-$Boosting, we replace the step (SING) by a step (GLMBOOST) which is just standard $\tilde{L}-$Boosting instead of SingBoost to benefit from the faster `glmboost` implementation. We will later see that this enables us to use our Stability Selection for standard Boosting models. Since neither Meinshausen and Bühlmann [2010] nor Hofner et al. [2015] mention how to compute the final coefficients after having performed the Stability Selection, we will provide an own strategy in the next chapter. With our implementation, we can directly apply our Stability Selection to existing Boosting methods as well as for SingBoost models by altering the `sing` argument. More details about the flexibility of our implementation can be found in part VI.*

Remark 11.4.3. *As already announced in section 2.4, we do not choose the learning rate adaptively. We usually set $\kappa := 0.1$.*

```
Initialization: Data $\mathcal{D}^{CMB} \in \mathbb{R}^{n_{cmb} \times (p+1)}$, number $B^{sing}$ of subsamples, initial row
  measure $\zeta^{init} := U(\{1,...,n\})$, number $n_{sing} < n_{cmb}$ of training instances used for the
  subsamples, binary variable sing, variable wagg for the type of weight aggregation,
  level $\alpha$, binary variable robagg and lower bound $0 \leq$ lower; other hyperparameters
  as in SingBoost (algorithm 6);
for $b = 1,...,B^{sing}$ do
    Draw a subsample $\mathcal{D}^{(sing,b,train)} \in \mathbb{R}^{n_{sing} \times (p+1)}$ from the data according to $\zeta^{init}$.
     The non-selected rows form the test data $\mathcal{D}^{(sing,b,test)}$;
    if $sing==TRUE$ then
        Step (SING): Perform SingBoost on $\mathcal{D}^{(sing,b,train)}$ with the given input
          parameters and get model $\hat{f}^{(b)}$ and empirical column measure $(\hat{\nu}_{\tilde{L}}^{L_2})^{(b)}$;
    else
        Step (GLMBOOST): Perform standard Gradient Boosting on $\mathcal{D}^{(sing,b,train)}$
          w.r.t. $\tilde{L}$ with the given input parameters and get model $\hat{f}^{(b)}$ and empirical
          column measure $(\hat{\nu}^{(\tilde{L})})^{(b)}$;
    end
    Step (CoM): Compute the scores $\hat{s}^{(b)}$ as in (11.4.1);
    Compute the weights $\hat{w}^{(b)}$ either as in (11.4.2) or (11.4.3)
end
if robagg==T then
    Step (ROB): Winsorize score vector $\hat{s}$ according to lower
end
Step (W-AGG): Compute the aggregated column measure $\hat{\nu}_{\tilde{L}}^{L_2}$ as in (11.4.4);
```
Algorithm 7: Column Measure Boosting

Remark 11.4.4 (**Stabilization of the singular part**). *CMB mimics in some sense the Stability Selection since models based on B^{sing} subsamples from the data are aggregated. However, the resulting model is clearly neither sparse nor stable yet, but one can think of CMB as an $\tilde{L}-$adapted* ***stabilization of the singular part*** *$J_{\tilde{L}}^{L_2}$. We recommend to use $B^{sing} < B$ subsamples for reasons of computational feasibility. However, it may not be necessary to set $B^{sing} > 1$ if we 0/1-transform the computed empirical column measure as in (11.4.4), but we will see in the next chapter that using Bootstrap samples generally may be beneficial when encountering contamination.*

Remark 11.4.5 (**Inliers**). *The step (ROB) should partially guard against contamination, i.e., if there are inliers in the data that let a model appear as being too well-suited and* *therefore get a very low score which leads to a very high weight when using the weights as in 11.4.3. An upper bound for winsorization is obsolete as we only focus on the $\lceil \alpha B^{sing} \rceil$ models with the smallest scores.*

Remark 11.4.6 (Bootstrapping vs. subsampling). *We used subsamples of the data. Algorithmically, one could also think about using Bootstrap samples, i.e., each row can be selected more than once. Our motivation for using subsamples arises from the goal to solve the hard ranking problem. As we have seen in remark 6.1.1, the computation of the hard ranking loss gets difficult if our response vector contains ties which would clearly be the very likely in Bootstrap samples. Aside from this issue, we want to let our algorithm resemble standard Stability Selection that also uses subsamples instead of Bootstrap samples.*

Remark 11.4.7. *To better understand which input arguments have to be inserted , see the following command line to apply CMB on a data set $\mathcal{D}^{CMB}$ where the best 3 of 10 SingBoost models with hard ranking singular iterations that are computed on subsamples containing 80 observations of $\mathcal{D}^{CMB}$ are aggregated, per default as in 11.4.2 and without winsorizing:*

```
CMB(D, nsing=80, Bsing=10, alpha=0.3 , singfam=Rank() , evalfam=Rank() , sing=T, LS=T)
```

For further details on the meaning of `evalfam`*, see part VI.*

Remark 11.4.8. *We admit that it may look strange to use just one test set resp. just one partition of the data into a training and a test set for the SingBoost models in Column Measure Boosting. We will see later that this issue gets overlain by an overriding partitioning scheme (see especially section 12.4).*

Remark 11.4.9 (Parallelization). *We admit that this algorithm can be time-consuming (especially as long as* `singboost` *is not implemented sophistically, see section 10.3), depending clearly on the number B^{sing}. However, since all models are computed separately on different subsamples, we could* ***parallelize*** *it, i.e., distribute these calculations on different independent computational cores and gather the results. The same is true for standard Stability Selection which has been implemented by Hofner in the* R*−package* `stabs` *in a parallelized manner. Exemplary for a parallelized algorithm, we provide the pseudocode for the parallelized Column Measure Boosting (without referring again to the extra features like the (ROB) step) in algorithm 8.*

Note that the distribution of the index sets requires some kind of dictionary so that each score reported by each core can be uniquely identified with one of the index sets. Of course, the respective packages in R *satisfy these requirements.*

```
Initialization: Data $\mathcal{D}^{CMB} \in \mathbb{R}^{n_{cmb}\times(p+1)}$, number $B^{sing}$ of subsamples, initial row
 measure $\zeta^{init} := U(\{1,...,n\})$, number $n_{sing} < n_{cmb}$ of training instances used for the
 subsamples, binary variable sing, variable wagg for the type of weight aggregation
 and level $\alpha$; other hyperparameters as in SingBoost (algorithm 6);
for $b = 1,...,B^{sing}$ do
    Draw a subsample $i^{(sing,b,train)}$ from the row indices $\{1,...,n\}$ according to $\zeta^{init}$;
end
(Step Prll): Distribute the index sets to cores $C_1,...,C_z$ such that core $k$ gets $b_k$
 index sets, $\sum_k b_k = B^{sing}$;
for $k = 1,...,z$ do
    for $l = 1,...,b_k$ do
        Form the sets $\mathcal{D}^{(sing,b,train)}$ and $\mathcal{D}^{(sing,b,test)}$ according to the corresponding
         index set;
        Apply SingBoost;
        Get the empirical column measure $(\hat{\nu}^L_{\tilde{L}})^{(l,k)}$;
        Compute the score $\hat{s}^{(l,k)}$
    end
    Gather all scores as in (11.4.1) and report the score vector $\hat{s}^{(k)} := (\hat{s}^{(l,k)})_{l=1}^{b_k}$;
end
Gather the scores from each core;
Compute the weights $\hat{w}^{(b)}$ either as in (11.4.2) or (11.4.3);
Step (W-AGG): Compute the aggregated column measure $\hat{\nu}^{L_2}_{\tilde{L}}$ as in 11.4.4;
```

Algorithm 8: Parallelized Column Measure Boosting

11.5 If the loss is very expensive to evaluate...

A key assumption to keep the computational complexity of SingBoost as stated in lemma 10.3.1 and remark 10.3.15 was that the loss function $\tilde{L}$ is easy to evaluate. Each $M-$th SingBoost iteration requires evaluating the target loss p times, letting the combined costs may get very high.

Can one think of another way to get the singular parts by avoiding cumbersome evaluation?

In other words, we still want to use L_2-Boosting, but somehow need to include variables that this algorithm does not select. Having the techniques of Random Forests (Breiman [2001]) and Random Lasso (Wang et al. [2011]) in mind, this leads to the following sampling idea: In each singular iteration, get a realization u_i, $i = 1, ..., M$, from the initial column measure $\nu^{init} \sim U(\{1, ..., p\})$ and fit the baselearner w.r.t. the corresponding variable.

So we replaced the rejection sampling step from the original SingBoost where we took the variable whose baselearner granted the best improvement, evaluated in $\tilde{L}$, with a simple sampling step where we randomly choose one variable in each singular iteration.

The heuristic behind this idea is that in the cases where we have chosen variables which are relevant for the loss $\tilde{L}$, the resulting performance on the test set is well enough for the whole model being selected by the aggregation step in CMB. But in fact, this approach is not yet meaningful.

Since the probability for each variable to be chosen is always p^{-1}, the expected number of singleton samples that we need until every variable has been chosen at least once is pH_p where H_p is the harmonic series up to summand p since we face a **coupon collector problem**. More precisely, let s^0 be the number of true nonzero coefficients. Then the probability that this strategy yields a model where at least one of those variables is included is clearly

$$1 - \left(\frac{p - s^0}{p}\right)^M.$$

But on the other side, the probability that we do not include any noise variable in the model is just

$$\tilde{p} = \left(\frac{s^0}{p}\right)^M$$

which is extremely small even with standard parameters. For example, this quantity is around 10^{-13} for $p = 200$, $s^0 = 10$, $M = 10$. Therefore, this approach implies the risk that selecting noise variables nullifies possible advantages in out-of-sample performance when including relevant variables (even from the assumed singular part $J_{\tilde{L}}^{L_2}$) in other singular iterations.

These issues can be partially fixed if one generates a realization u from $U \sim U(\{1, ..., p\})$ in advance and then fits a baselearner w.r.t. the column u in each singular iteration.

Then we are in the coupon collector setting regarding the expected number B of subsamples needed. But again, there is a great issue. Assume for example that $p = 1000$, $B^{sing} = 10000$ (for this sampling strategy, B^{sing} must clearly be chosen much higher than we would require for CMB with standard SingBoost) and $\alpha = 0.01$. If there is a variable which is contained in the singular part $J_{\tilde{L}}^{L_2}$ such that it cannot be selected by L_2-Boosting, then we will only get this variable in the cases where we draw the respective column. But in fact, with a probability of 99.5%, we did not draw it in more than 19 cases, thus the variable would at most get a relative importance of 0.19 and therefore it ultimately gets deleted due to the cutoff in the subsequent Stability Selection, see next section.

Our final proposal is therefore the following algorithm:

```
Initialization: Data $\mathcal{D}^{sing}$, step size $\kappa \in ]0,1]$, number $m_{iter}$ of iterations, number
  $M \leq m_{iter}$ (each $M-$th iteration is a singular iteration), target loss $\tilde{L}$ (as part of a
  family object singfamily), binary variable LS and column index vector $u$ ;
Set
  $$\text{runs} = \left\lfloor \frac{m_{iter}}{M} \right\rfloor$$
  Set $\hat{f}^{(0)} := 0$;
for $k = 1, ..., \text{runs}$ do
    Fit the baselearner on the residuals of model $\hat{f}^{((k-1)M)}$ w.r.t. the variable
      corresponding to column $u_{\lceil kM^{-1} \rceil}$;
    Get the weak model $\hat{g}^{((k-1)M+1)}$ and update the model via
    $\hat{f}^{((k-1)M+1)} = \hat{f}^{((k-1)M)} + \kappa \hat{g}^{((k-1)M+1)}$;
    Perform $(M-1)$ steps of $L_2-$Boosting starting with the residuals w.r.t. the
      model $\hat{f}^{((k-1)M+1)}$;
    Get the updated model $\hat{f}^{(kM)}$;
end
```

Algorithm 9: SingBoost-u

So, we directly get the following **Best Subset CMB (BSCMB)** algorithm:

```
Initialization: Same as in CMB (algorithm 7);
Same as CMB, but replace step (SING) with
Step (SING-u): Get $u$ and perform SingBoost-u on $\mathcal{D}^{(sing,b,train)}$ with the given
  input parameters and get the model $\hat{f}^{(b)}$ and the empirical column measure $(\hat{\nu}_{\tilde{L}}^{L_2})^{(b)}$;
```

Algorithm 10: Best Subset Column Measure Boosting (BSCMB)

What have we done? We pre-specified a vector u for the columns w.r.t. which the baselearners in the singular iterations should be computed. As already indicated by the name of the latter algorithm, we may combine some sort of Best Subset Selection with SingBoost. If for example we select columns $j_1 < j_2$ and set $u_k = j_{k(mod2)+1}$, we may choose all $p(p-1)/2$ possible pairs of columns and run SingBoost with the suitable singular iterations. The idea behind this procedure is that we need to resolve the issue that a relevant variable may not pass the cutoff in a subsequent Stability Selection. With this pair-wise strategy and with α

low enough, even singular variables may enter the final model of Best Subset CMB.

Clearly, this approach is only meaningful if $\tilde{L}$ is really expensive because the number of different models needs to be drastically increased to some $\tilde{B}^{sing} \gg B^{sing}$, so that Best Subset CMB is only faster than CMB if the additional computation time that arises due to the large $\tilde{B}^{sing}$ is smaller than the computation time that was needed for evaluating all p simple least squares baselearners w.r.t $\tilde{L}$ in CMB.

Let us take a closer look on the computational cost of standard CMB and Best Subset CMB. Let us denote by $C(\cdot)$ a cost operator which represents the computational cost of the operation in the brackets. Let

$$\tilde{M} := \left\lfloor \frac{m_{iter}}{M} \right\rfloor$$

be the number of performed singular iterations. Then we get

$$C(\text{SingBoost}) \sim \tilde{M}(pC(bl) + pC(eval\tilde{L}(n_{sing}))) + (m_{iter} - \tilde{M})C(\texttt{glmboost}(M-1)),$$

$$C(\text{CMB}) \sim B^{sing}C(\text{SingBoost}) + B^{sing}C(eval\tilde{L}(n_{cmb} - n_{sing}))$$

where $\texttt{glmboost}(M-1)$ represents $(M-1)$ steps of L_2-Boosting, $C(bl)$ is the cost of fitting a baselearner and $eval\tilde{L}(n)$ emphasizes the cost of evaluating the loss $\tilde{L}$ where both input vectors have the length n. Minor computations like getting the final weights or the residuals, generating the subsamples, finding the index of the minimal loss or doing simple assignments are ignored here.

Similarly, we get

$$C(\text{SingBoost} - u) \sim \tilde{M}C(bl) + (m_{iter} - \tilde{M})C(\texttt{glmboost}(M-1)),$$

$$C(\text{BSCMB}) \sim \tilde{B}^{sing}C(\text{SingBoost} - u) + \tilde{B}^{sing}C(eval\tilde{L}(n_{cmb} - n_{sing}))$$

so that we get

$$C(\text{CMB}) - C(\text{BSCMB}) \sim \underbrace{[pB^{sing} - \tilde{B}^{sing}]\tilde{M}C(bl)}_{①}$$

$$+ \underbrace{(B^{sing} - \tilde{B}^{sing})(m_{iter} - \tilde{M})C(\texttt{glmboost}(M-1))}_{②}$$

$$+ \underbrace{(B^{sing} - \tilde{B}^{sing})C(eval\tilde{L}(n_{cmb} - n_{sing}))}_{③} + \underbrace{B^{sing}\tilde{M}pC(eval\tilde{L}(n_{cmb} - n_{sing}))}_{④}.$$

We proceed analyzing each component of the cost gap.

The sign of ① is undetermined. If one sets $\tilde{B}^{sing}$ to $p(p-1)/2$, then the summand is negative for large p. Other sampling strategies can also lead the summand to be positive.

② is clearly negative and represents the extra costs of Best Subset CMB due to the increased number of baselearners to be computed in the non-singular iterations.

③ is clearly negative, so Best Subset CMB has extra costs concerning the computation of the loss $\tilde{L}$ on the test set.

④ is, especially for losses that are expensive to compute, the main trigger of computational costs that are additionally necessary for CMB when performing a rejection step w.r.t. $\tilde{L}$ in the singular iterations. Due to the factors $\tilde{M}$ and p, its amount can be immense on large data sets and when performing many singular steps.

Note also that in SingBoost we need to compute $\tilde{M}p$ times the baselearner which may be expensive if the baselearner is costly to compute (which may be the case when fitting non-linear baselearners, see section 18.1). But on the other hand, due to its computation in each non-singular iteration, this rather is a disadvantage of Best Subset CMB than of CMB.

To get further insights into which algorithm would be faster, let us assume for the moment that we can write

$$C(\texttt{glmboost}(M-1)) \sim (M-1)pC(bl),$$

i.e., we ignore any other computational costs and only concentrate on the costs for computing the baselearners. Now one can think of different scenarios concerning the costs of the baselearners and of the evaluation of $\tilde{L}$.

Example 11.5.1. *Assume that the cost of fitting a baselearner on a column with n entries is $C(bl) \sim n$ and that the evaluation of $\tilde{L}$ based on two vectors of length n is also proportional to n, so $C(eval\tilde{L}(n)) \sim n$. Then we get*

$$C(\text{CMB}) - C(\text{BSCMB}) \sim [B^{sing}\tilde{M}p - \tilde{B}^{sing}\tilde{M}]n_{sing}$$

$$+(B^{sing} - \tilde{B}^{sing})(M-1)(m_{iter} - \tilde{M})pn_{sing} + (B^{sing} - \tilde{B}^{sing})(n_{cmb} - n_{sing})$$

$$+B^{sing}\tilde{M}p(n_{cmb} - n_{sing})$$

$$= n_{sing}B^{sing}[(M-1)(m_{iter} - \tilde{M})p + \tilde{M}p] + B^{sing}(n_{cmb} - n_{sing}) + B^{sing}\tilde{M}p(n_{cmb} - n_{sing})$$

$$-n_{sing}\tilde{B}^{sing}[(M-1)(m_{iter} - \tilde{M})p + \tilde{M}] - \tilde{B}^{sing}(n_{cmb} - n_{sing})$$

$$\sim n_{sing}pB^{sing}[(M-1)(m_{iter}-\tilde{M})+2\tilde{M}]-n_{sing}p\tilde{B}sing[(M-1)(m_{iter}-\tilde{M})]$$

since the difference $(n_{cmb}-n_{sing})$ can be considered to be low in comparison to p, hence we see that in general, i.e., for $\tilde{B}^{sing} \gg B^{sing}$, this quantity is negative.

Example 11.5.2. *Let everything be as in the previous example except that we assume here that $C(eval\tilde{L}(n)) \sim n^2$. Then we similarly get a cost difference proportional to*

$$n_{sing}pBsing\left[(M-1)(m_{iter}-\tilde{M})+\tilde{M}+\tilde{M}\frac{(n_{cmb}-n_{sing})^2}{n_{sing}}\right]$$
$$-n_{sing}p\tilde{B}sing[(M-1)(m_{iter}-\tilde{M})]$$

which may be positive since the last summand in the first angular brackets can be the leading term for large n_{cmb} and therefore a considerable difference $(n_{cmb}-n_{sing})$. So, if we perform many singular steps and if we have a data set where $n_{sing} > p$, the additional costs when letting the algorithm run $\tilde{B}^{sing} \gg B^{sing}$ times is more than compensated due to the expensive loss $\tilde{L}$.

Remark 11.5.1. *In the two scenarios like in the examples but where we assume that computing a baselearner causes costs proportional to n^2, the loss differences are negative in general. That means that even in the case where the loss function $\tilde{L}$ is expensive to evaluate, the additional cost for computing the baselearners in the non-singular steps for $\tilde{B}^{sing} \gg B^{sing}$ in Best Subset CMB is more relevant.*

Remark 11.5.2. *In Best Subset CMB, it maybe suffices if one samples sufficiently often from the set*

$$\{(j_1,j_2) \mid j_1,\ j_2 \in \{1,...,p\},\ j_1 < j_2\}$$

to avoid vast computational costs for really high-dimensional data.

One may ask if one could apply some two-stage procedure to avoid sampling from the uniform distribution in order to respect the column measure itself like the Random Lasso of Wang et al. [2011]. A two-stage algorithm can indeed be a method to decrease the computational costs for expensive evaluations of $\tilde{L}$. The former may not be realizable since sampling from the empirical column measure in a second stage would require to approximate it in the first stage. Therefore, in order to account for $\tilde{L}$, one had to apply CMB less than B times, to compute the empirical column measure to sample column w.r.t. to this measure in a second stage and to apply CMB again. Clearly, the computational costs of the later runs are far lower than for the runs in the first stage due to the reduced number of columns, but since we need enough statistical evidence in the first stage, B cannot be significantly reduced. Moreover, this requires two parameters that control the fraction of best models to account for $\tilde{L}$, say, $\alpha^{(1)}$ and $\alpha^{(2)}$, that enter as tuning parameters as well as the proportion of subsamples in the first and second stage. Additionally, since the second stage performs sampling in the singular iterations instead of optimization w.r.t. $\tilde{L}$, we suspect that we lose quality compared to the original CMB.

As for the question if any two-stage procedure could be possible, we may turn the order of the stages discussed above. Our motivation for preferring the Best 2-Subset Selection over a Best Singleton Selection was the issue that the final aggregation step in a Stability Selection would nevertheless delete all relevant variables again that were not chosen by L_2-Boosting.

We consider the following algorithm:

```
Initialization: Same as in CMB (algorithm 7), but with levels $\alpha^{(1)}$, $\alpha^{(2)}$;
for $b_1 = 1, ..., p$ do
    Same as in CMB, but replace step (SING) with step (SING-u) with $u := b_1$;
    Step (CoM): Compute the scores $\hat{s}^{(b_1)}$ as in (11.4.1);
    Compute the weights analogously as in $\hat{w}^{(b_1)}$ as in (11.4.2) w.r.t. $\alpha^{(1)}$
end
if robagg==T then
    Step (ROB): Winsorize score vector $\hat{s}$ according to lower
end
Step (W-AGG): Compute the aggregated column measure $\check{\nu}_{\tilde{L}}^{L_2}$ in the sense of
 (11.4.4);
for $b_2 = 1, ..., B^{sing}$ do
    Reduce the data to the support of the column measure $\check{\nu}_{\tilde{L}}^{L_2}$;
    Draw a subsample $\tilde{\mathcal{D}}^{(sing,b_2,train)}$ from the reduced data according to the support of
     $\check{\nu}_{\tilde{L}}^{L_2}$. The non-selected rows form the test data $\tilde{\mathcal{D}}^{(sing,b_2,test)}$;
    Step (SING): Perform SingBoost on $\tilde{\mathcal{D}}^{(sing,b_2,train)}$ with the given input
     parameters and get model $\hat{f}^{(b_2)}$ and empirical column measure $(\hat{\nu}_{\tilde{L}}^{L_2})^{(b_2)}$;
    Step (CoM): Compute the scores $\hat{s}^{(b_2)}$ as in (11.4.1);
    Compute the weights $\hat{w}^{(b_2)}$ either as in (11.4.2) or (11.4.3) using $\alpha^{(2)}$
end
if robagg==T then
    Step (ROB): Winsorize score vector $\hat{s}$ according to lower
end
Step (W-AGG): Compute the aggregated column measure $\hat{\nu}_{\tilde{L}}^{L_2}$ as in (11.4.4)
```

Algorithm 11: Two Stage Column Measure Boosting (TSCMB)

Similarly, we can calculate the computational costs in terms of their major triggers. Let $p^{(2)}$ be the number of columns that have been selected in the second stage, i.e.,

$$p^{(2)} = \#\{j \mid (\check{\nu}_{\tilde{L}}^{L_2})_j > 0\}.$$

Then the costs for TSCMB are of order

$$C(\text{TSCMB}) \sim p[MC(bl) + (m_{iter} - M)C(\texttt{glmboost}(M-1))] + pC(eval\tilde{L}(n_{cmb} - n_{sing}))$$
$$+B^{sing}[Mp^{(2)}C(bl) + Mp^{(2)}C(eval\tilde{L}(n_{sing})) + (m_{iter} - M)C(\texttt{glmboost}(M-1))]$$
$$+B^{sing}C(eval\tilde{L}(n_{cmb} - n_{sing})).$$

If one compares these costs with the costs of CMB and BSCMB with the same parameters, one can conclude that, in general, TSCMB is cheaper than BSCMB if the number of SingBoost runs in BSCMB is very large compared to p which may be the case for high values of p. It is hard to determine whether CMB or TSCMB is more expensive if the computational costs of fitting the baselearner and evaluating the loss are both of order $\mathcal{O}(n)$. But in fact, if the loss is expensive to evaluate, say the costs are of order $\mathcal{O}(n^2)$, then one can easily find realistic parameter specifications such that each of them may be superior to the other, for small values for p as well as for large values of p. We skip tedious calculations here.

Remark 11.5.3 (**A multilevel procedure?**). *We already have proposed a two-stage procedure that is able to reduce the computational costs of Column Measure Boosting in certain settings. This may be a starting point to think about another extension. In TSCMB, we perform two different approaches on each stage, since an optimization w.r.t. $\tilde{L}$ in the singular steps is only done in the second stage whereas the first stage only respects the target loss function when finding the best whole models.*

In contrast to the relation of precision in the sense of a reduced set of columns and computational costs where increased precision reduces the cost, the number of singular iterations is proportionally related to the computational costs. It is evident that a high number of singular iterations may reflect the structure of $\tilde{L}$ better than a low number $\tilde{M}$. One faces a related problem for example in numerical integration where a fine grid dramatically increases the costs. In this setting, one can strikingly increase the efficiency if one uses a Multilevel Monte Carlo (MLMC) approximation (see Heinrich [2001]). The idea behind this method is to combine different degrees of precision (here in the sense of how many nodes are chosen for interpolation) in order to find an optimal tradeoff between costs and variance.

Transferring this idea to CMB, this would imply to increase the number of singular iterations in each level which produces an empirical column measure through aggregation and therefore *reduces the number of columns for the next level. However, apart from the practical problem how to choose the numbers* B_l^{sing} *of subsamples and* $\tilde{M}_l$ *of singular iterations for each level* $l = 1, ..., L$*, it is debatable if this approach would yield the risk of discarding columns in the earlier levels due to too few singular iterations that would have been selected if one ran the algorithm with more singular steps. We leave this open as a direction for future research.*

Chapter 12

Interplay of row and column measures

So far, our CMB algorithm delivers us the aggregated relevance of each variable, i.e., an aggregated empirical column measure $\hat{\nu}_{\tilde{L}}^{L_2}$. Choosing a threshold π_{thr} or a number q of variables for our final model, we could get a sparse set of selected variables. But since the computed coefficients clearly got lost during the aggregation process, it remains to investigate how we can get the coefficients for the (few) variables in our final model.

This model is of course not stable since it has been trained on one single subset $\mathcal{D}^{CMB}$ of the data. Thus, we need a modified Stability Selection that finds a stable set of predictors according to B different CMB models. More precisely, we do not only have to account for the reported empirical column measures of CMB but also need to tailor the selection to the target loss $\tilde{L}$.

Traditionally, one first performs model selection, resulting in a set $\hat{S}$ of relevant predictors. Then, standard linear regression on the reduced data $\mathcal{D}_{\hat{S}} := (X_{\cdot,\hat{S}}, Y)$ is used to estimate the corresponding coefficients. For example, this procedure is applied when using best subset selection (see [Friedman et al., 2001, Ch. 3.3]). However, modern model selection algorithms like Lasso or Boosting already provide reliable coefficients which can be directly used. Another method is averaging the coefficients fitted on different Bootstrap samples as done in the Random Lasso (Wang et al. [2011]).

Concerning models that have been derived by a Stability Selection, it is no longer that evident how the final coefficients for the predictors in the stable resulting model should be computed. Note that neither Meinshausen and Bühlmann [2010] nor Hofner et al. [2015] provide any recommendations for that. Averaging coefficients does not make any sense since the coefficients were fit for generally more than q variables, if q represents the number of selected predictors after the Stability Selection. We assume that one just applies the traditional algorithm corresponding to the respective loss function on the reduced data set, i.e., using all rows but only the columns corresponding to the stable predictor set.

For our setting, it is again more difficult since we do not even have a simple standard algorithm to compute a model according to $\tilde{L}$ on the reduced data. One may ask if Column Measure Boosting could be seen as a pre-processing step to apply the `CRank` algorithm of Clémençon and Achab [2017] on the reduced data. This has to be negated since the intention of this work is to remedy some of the weaknesses of tree-type algorithms that we already discussed in section 5.5. Furthermore, especially if we assume a linear underlying model, there is no need to use trees as they would only cause that effects of single predictors to be less interpretable than in a linear regression model.

We will see in this chapter that we even got more information from Column Measure Boosting than we concerned about yet. We present the algorithm CMB-3S which takes $\tilde{L}-$adapted empirical row and column measures into account and produces a stable predictor set as well as final coefficients.

As a computational tool, we also provide an algorithm (which we call CV.CMB-3S) that computes the cross-validated error according to the target loss and an "ultra-stable" empirical column measure.

In light of our RCM framework, we try to find a systematization of sparse learning algorithms in order to get deeper insights into the relation of sparsity, stability and robustness.

At the end, we show some possible extensions of our CMB-3S algorithm for particular situations that we did not yet treated.

12.1 A modified, loss-based Stability Selection

Note that CMB still leads to a **possibly non-sparse and unstable** empirical column measure. The first step is to randomly split the data set into a training set $\mathcal{D}^{train}$ and a validation set $\mathcal{D}^{valid}$. Then, we draw B subsamples $\mathcal{D}^{(CMB,b)}$ from $\mathcal{D}^{train}$ and perform Column Measure Boosting on them, resulting in empirical aggregated column measures. These measures are averaged and form an empirical column measure $(\hat{\nu}_{\tilde{L}}^{L})^{CMB}$ with a little abuse of notation (see the pseudocode 16 of our final algorithm where we again spell out each step with appropriate sub- and superscripts). This aggregated column measure is the analog of the aggregated column measure $\hat{\nu}^{(L)}$ in Hofner's Stability Selection (algorithm 5).

We propose the following strategy: We either only take all variables whose corresponding component of $(\hat{\nu}_{\tilde{L}}^{L_2})^{CMB}$ exceeds some threshold π_{thr}, or we rank the components in a descending order and take the first q ones. Thus, we produce one of the final sets of selected predictors

$$(\hat{S}_{\tilde{L}}^{L_2})^{stab}(q) := \{j \mid (\hat{\nu}_{\tilde{L}}^{L_2})^{CMB}(\{j\}) \geq (\hat{\nu}_{\tilde{L}}^{L_2})^{CMB}_{(p-q+1):p}\} \tag{12.1.1}$$

$$(\hat{S}_{\tilde{L}}^{L_2})^{stab}(\pi_{thr}) := \{j \mid (\hat{\nu}_{\tilde{L}}^{L_2})^{CMB}(\{j\}) \geq \pi_{thr}\} \tag{12.1.2}$$

where the notation with the sub- and superscript already emphasizes that these sets are produced by our algorithm, cf. the notation in Meinshausen and Bühlmann [2010].

Remark 12.1.1 (**Discussion of** q)**.** *Note that the usage of q is quite different in our context than in Hofner's Stability Selection. While the latter lets each Boosting algorithm run until q variables have been selected, we think of q as the number of final variables for the following two reasons. If we run SingBoost until it selects q variables, the variable set reported by CMB has at least q variables, so we could not control the number of variables that enter the Stability Selection. Secondly, due to the rather expensive computation, we need to avoid the case that our Stability Selection just ends up in having chosen no variable at all costs. This can indeed be controlled by treating q as the final number of variables, see also part VI for further details and issues that the Stability Selection from `stabs` can face.*

Until now, we did not specify the parameters q or π_{thr}. Although Meinshausen and Bühlmann [2010] and Hofner et al. [2015] recommend not to give too much attention to the cutoff π_{thr} as long as it falls into a reasonable interval, we propose to ensure that the Stability Selection that we perform is adapted to the loss function $\tilde{L}$.

We choose either q or π_{thr} by cross validation, or more precisely, by a grid search, by using a validation data set $\mathcal{D}^{valid} \in \mathbb{R}^{n_{val} \times (p+1)}$ that is drawn in advance from the given data set $\mathcal{D}$ to adjust either the parameter q or π_{thr} by comparing the out-of-sample performance w.r.t. $\tilde{L}$ of the respective models.

In the case of adjusting q, we need a reasonable subset of $\mathbb{N}$ which clearly satisfies

$$q_{grid} \subset \{1, 2, ..., \#\{j \mid (\hat{\nu}_{\tilde{L}}^{L_2})^{CMB}_j > 0\}\}. \tag{12.1.3}$$

In the case of adjusting π_{thr}, we discretize a reasonable interval (w.l.o.g. the one suggested in Hofner et al. [2015], but with the boundaries, i.e. $[0.5, 1]$) according to some mesh size

$\Delta > 0$, so we get the grid

$$\pi_{grid} = \{0.5, 0.5 + \Delta, ..., 1 - \Delta, 1\}. \tag{12.1.4}$$

Remark 12.1.2. *Setting $\Delta = 0.05$ seems to be a reasonable compromise between the fineness of the grid and the computational time. However, maybe it is possible to choose Δ in an adaptive fashion.*

Remark 12.1.3 (Another Stability Selection variant?). *The Stability Selection of Hofner (Hofner et al. [2015]) is based on Boosting models that are iterated until q predictors are chosen and counting just the number of Boosting models in which a particular variable has been selected. The intention behind this strategy is that very relevant variables tend to be selected early, so iterating too long will cause noise variables to enter the Boosting models.*

On the other hand, this Stability Selection ***ignores how often a variable has been chosen in the Boosting models****. Assume that each Boosting algorithm has m_{iter} iterations. Then the Boosting models are prone to include noise variables, but their relative selection frequency can be considered to be rather low. Therefore, simply taking the arithmetic mean of the selection frequencies over all Boosting models can indeed lead to a similar empirical column measure such that a suitable cutoff defines a stable model which is comparable to the stable models that are computed by Hofner's Stability Selection.*

As for the computational time, Hofner's Stability Selection is clearly better in the case of strong signals, i.e., high signal-to-noise ratios, since the relevant variables generally enter the models quickly. As we will see in part VI (and have already seen in example 10.2.1), a low signal-to-noise ratio can severely irritate the Boosting models, especially for other losses than the squared loss, so Hofner's Stability Selection fails and requires to even increase the iteration number until the Boosting models finally indeed select q variables. Therefore, this strategy to enforce a selection of q variables can be a restriction in both directions, i.e., that very few Boosting iterations are performed as well as the number of iterations has to be manually modified.

However, when concerning SingBoost models, only counting the relative selection frequency would be disadvantageous for variables which can only be detected in singular iterations since their selection frequency is bounded from above in advance according to the proportion of the number of singular iterations and m_{iter}. Therefore, it is necessary to find a way to adapt the Stability Selection to SingBoost.

Since we assume that this kind of Stability Selection can be promising but since it may be more *suitable to be combined with standard Boosting models rather than with SingBoost models, we do not pursue this idea any further in this work but leave this open for future work.*

Our ideas from above result in the following algorithm which we call **CMB-3S** as acronym for "**C**olumn **M**easure **B**oosting with **S**ingBoost and **S**tability **S**election":

```
Initialization: Data $\mathcal{D}^{train}$, $\mathcal{D}^{valid}$, size $n_{cmb}$ of subsamples, binary variable
 gridtype, grid, hyperparameters as in algorithm 7;
for $b=1,...,B$ do
    Draw a subsample $\mathcal{D}^{CMB} \in \mathbb{R}^{n_{cmb} \times (p+1)}$;
    Perform Column Measure Boosting on $\mathcal{D}^{CMB}$ and get an aggregated empirical
     column measure;
end
Step (SUB-AGG): Average the $B$ column measures and get the aggregated
 empirical column measure $(\hat{\nu}_{\tilde{L}}^{L_2})^{CMB}$;
Step (STAB):
if gridtype=='qgrid' then
    for $k = 1, ..., |q_{grid}|$ do
        Get the stable model $(\hat{S}_{\tilde{L}}^{L_2})^{stab}((q_{grid})_k)$ according to 12.1.1;
        Compute the coefficients on the reduced data
            $$\mathcal{D}^{(train,k)} = \mathcal{D}^{train}_{(\hat{S}_{\tilde{L}}^{L_2})^{stab}((q_{grid})_k)}$$
        and get the loss $(\tilde{L}_n)^{(valid,k)}$ on the validation data $\mathcal{D}^{valid}$
    end
else
    for $k = 1, ..., |\pi_{grid}|$ do
        Get the stable model $(\hat{S}_{\tilde{L}}^{L_2})^{stab}((\pi_{grid})_k)$ according to 12.1.2;
        Compute the coefficients on the reduced data
            $$\mathcal{D}^{(train,k)} = \mathcal{D}^{train}_{(\hat{S}_{\tilde{L}}^{L_2})^{stab}((\pi_{grid})_k)}$$
        and get the loss $(\tilde{L}_n)^{(valid,k)}$ on the validation data $\mathcal{D}^{valid}$
    end
end
Choose the model corresponding to $k_{opt} = \text{argmin}_k((\tilde{L}_n)^{(valid,k)})$;
Step (COEF): Get final coefficients
```

Algorithm 12: CMB-3S

Remark 12.1.4. *According to Hofner et al. [2015], it suffices to set B to 100 as it has* *been done in Meinshausen and Bühlmann [2010]. This seems to be also appropriate for our algorithm. So, we mimic the implementation in the package* `stabs` *(Hofner and Hothorn [2017], Hofner et al. [2015], Thomas et al. [2018]) where this is referred to as "Meinshausen-Bühlmann sampling scheme" which is the default* (`sampling.type='MB'`)*. We do not regard sampling disjoint subsamples as proposed by Shah and Samworth [2013] which nevertheless is available in* `stabs` (`sampling.type='SS'`)*.*

Remark 12.1.5. *The major difficulty and the main difference between our Stability Selection and the Stability Selection proposed in Meinshausen and Bühlmann [2010] or Hofner et al. [2015] is that we perform an optimization of either q or π_{thr} whereas the cited Stability Selections choose π_{thr} in advance. Since we actually want to optimize $\tilde{L}$, it is most natural to compare the out-of-sample performances w.r.t. $\tilde{L}$. This is the reason why we had to separate the set $\mathcal{D}^{valid}$ in advance. The procedure in the stability iterations is basically the same as in the stability iterations in the standard Stability Selections. We also compute our (CMB) model on a subsample of the data and get an (in our case aggregated) empirical column measure which is aggregated at the end. The non-selected rows in each stability iteration are ignored, as in the cited references.*

Remark 12.1.6 (Singlar parts revisited). *It is very important to remind that even if the true column measures $\nu^{(L)}$ and $\nu^{(\tilde{L})}$ were equivalent, i.e., there do not exist singular parts, and even if the empirical column measures $\hat{\nu}_n^{(L)}$ and $\hat{\nu}_n^{(\tilde{L})}$ were equivalent, there would be no guarantee that the stable sets were the same due to the different mass distributed to the relevant predictors. That means that even the selection of the true coefficients in real applications, i.e., for finite n,* ***is insufficient for the equality of stable predictor sets****.*

12.2 Final coefficients and CV.CMB-3S

We did not yet reveal how we compute the coefficients. There are several reasonable approaches, but one of them will require more theoretical background which we discuss in this section.

12.2.1 Induced empirical row measures

In definition 9.7.1, we emphasized that the empirical column measure that we get from our algorithm essentially depends on the initial row measure ζ^{init} which has been used for subsampling. In our former examples concerning empirical row measures, especially in the context of robust statistics (examples 9.5.2, 9.5.3), they solely depended on some empirical loss, i.e., the row measures may be thought of being induced by the coefficients.

This is not the whole truth yet for robust learning procedures with model selection from which kind we shortly analyze the already mentioned algorithms FRB (Salibián-Barrera and Van Aelst [2008]), see also example 9.5.3, and SLTS (Alfons et al. [2013]). FRB with model selection effectively performs Best Subset Selection combined with FRB. In other words, one starts with an **initial 0/1-column measure** ν^{init} and applies the FRB algorithm as usual, but on the reduced data, resulting in an empirical row measure $\hat{\zeta}$ for each choice of ν^{init}. Therefore, $\hat{\zeta}$ is an empirical row measure **induced by a column measure**, so one may write $\hat{\zeta}(\nu^{init})$ which we will suppress if possible. Note that Salibián-Barrera and Van Aelst [2008] also provided a backward subset selection strategy which may be identified with adaptively choosing initial column measures such that the number of columns to which a zero is assigned increases by one in each step.

Even more interesting for our **row column measure (RCM) framework** is the SLTS algorithm which is a sparse variant of the usual LTS (”Least Trimmed Squares”) going back to Rousseeuw (Rousseeuw [1984]). While LTS searches for the best ”clean” subset $H \subset \{1, ..., n\}$ such that the least squares model based on the rows corresponding to H and all columns provides the smallest L_2-loss, the SLTS algorithm replaces the least squares fits with Lasso fits. Since the algorithm works iteratively, one can state it in our language in the following form:

```
Initialization: Data $\mathcal{D}$, hyperparameters $\lambda$, $h$;
Set $k = 0$;
Get set $H_0$ and the corresponding empirical row measure $\hat{\zeta}^{(0)}$;
while not converged do
    Compute a Lasso model on the rows defined by $\hat{\zeta}^{(k-1)}$ and get coefficients $\hat{\beta}^{(k)}$ and
     the corresponding empirical column measure $\hat{\nu}^{(k)}$;
    Find the set $H_k$ of size $h$ to which the rows resulting in the smallest $h$
     (unpenalized) squared residuals belong. Get the corresponding empirical row
     measure $\hat{\zeta}^{(k)}$;
end
```

Algorithm 13: SLTS

Note that SLTS starts with different initial subsets H_0 which are based on the rows belonging to the smallest h residuals of an initial Lasso fit on three randomly selected (from a uniform initial row measure ζ^{init}) instances. Despite SLTS can be identified as a limit case of the general RCM framework, it is an exemplar of sophisticatedly constructing a learning model based on an adapted interplay of empirical row measures and column measures.

Remark 12.2.1 (RCM pairs). *In general, we want to construct learning algorithms that provide* ***RCM pairs*** *resulting in mappings of the form*

$$(i,j) \mapsto (\hat{\zeta}_i, \hat{\nu}_j), \quad i=1,...,n,\ j=1,...,p. \tag{12.2.1}$$

SLTS is a limit case in the sense that both the final empirical row measure as well as the final empirical column meausure are $0/1-$*valued which means a hard decision in the sense that a row or column enters the model or not.*

12.2.2 Computing the coefficients of CMB-3S

Inspired by the RCM framework, we take a closer look at CMB and reveal that we **indeed have empirical row measures that are induced by empirical column measures**.

In CMB, we compute the weights in the sense of (11.4.2) or (11.4.3) for each fitted model. Since the model is clearly represented by its coefficients and the empirical column measure, it implicitly also depends on the selected rows. More precisely, the $b-$th model is based on the set

$$I^{(b)} = \{i_1^{(b)}, ..., i_{n_{sing}}^{(b)}\}$$

and we assign a weight $\hat{w}^{(b)}$ based on the score $\hat{s}^{(b)}$ according to (11.4.1) to it. Similarly to the empirical column measure $\hat{\nu}_{\tilde{L}}^{L_2}$ from CMB as in (11.4.4), we can compute an **aggregated empirical row measure** by

$$\hat{\zeta}_{\tilde{L}}^{L_2}(\{i\}) := \frac{1}{\lceil \alpha B^{sing} \rceil} \sum_b \hat{w}^{(b)} I(i \in I^{(b)}) \tag{12.2.2}$$

so that $\hat{\zeta}_{\tilde{L}}^{L_2} \in \mathbb{R}^{n_{cmb}}$ where we emphasize again with the notation that it is based on L_2, but $\tilde{L}-$adapted.

Therefore, we propose the following modification of CMB:

```
Same as Column Measure Boosting (algorithm 7);
Step (W-AGG): Compute the aggregated column measure $\hat{\nu}_{\tilde{L}}^{L_2}$ as in (11.4.4) and
the aggregated row measure $\hat{\zeta}_{\tilde{L}}^{L_2}$ as in (12.2.2)
```

Algorithm 14: CMB with empirical row measures

Remark 12.2.2. *Since the empirical column measures essentially depend on the initial row measure* ζ^{init}*, an accurate notation would be* $\hat{\zeta}_{\tilde{L}}^{L_2}(\hat{\nu}_{\tilde{L}}^{L_2}(\zeta^{init}))$ *which we will suppress in the rest of the thesis whenever it is not absolutely necessary.*

Thus, we already have $\tilde{L}-$adapted row weights in the sense that a high row weight indicates that the computed coefficients based on these rows are well-suited w.r.t. $\tilde{L}$. Therefore, one of our recommendations of how to compute the coefficients of CMB-3S is **weighted least squares** with the weights according to the aggregated row measure $(\hat{\zeta}_{\tilde{L}}^{L_2})^{CMB}$ averaged by the B CMB models on each subsample of $\mathcal{D}^{train}$.

More formally, let

$$p_k := |(\hat{S}_{\tilde{L}}^{L_2})^{stab}(*_k)|$$

for $* \in \{q_{grid}, \pi_{grid}\}$. Then, for any k ranging from 1 to the cardinality of the grid, we compute

$$\hat{\beta}^{WLS,k} := [(X^{train}_{\cdot,(\hat{S}_{\tilde{L}}^{L_2})^{stab}(*_k)})^T (\widehat{W_{\tilde{L}}^{L_2}})^{-1} X^{train}_{\cdot,(\hat{S}_{\tilde{L}}^{L_2})^{stab}(*_k)}]^{-1} (X^{train}_{\cdot,(\hat{S}_{\tilde{L}}^{L_2})^{stab}(*_k)})^T (\widehat{W_{\tilde{L}}^{L_2}})^{-1} Y^{train} \quad (12.2.3)$$

where

$$\widehat{W_{\tilde{L}}^{L_2}} := \text{diag}((\hat{\zeta}_{\tilde{L}}^{L_2})_i).$$

We also keep the option of computing unweighted least squares models. Then we just have

$$\hat{\beta}^{LS,k} := [(X^{train}_{\cdot,(\hat{S}_{\tilde{L}}^{L_2})^{stab}(*_k)})^T X^{train}_{\cdot,(\hat{S}_{\tilde{L}}^{L_2})^{stab}(*_k)}]^{-1} (X^{train}_{\cdot,(\hat{S}_{\tilde{L}}^{L_2})^{stab}(*_k)})^T Y^{train} \quad (12.2.4)$$

Since the first variant is just $\tilde{L}-$adapted in the sense of a row measure, we propose a third variant which is even more adapted to the target loss $\tilde{L}$, i.e., we apply SingBoost again on the reduced data, i.e.,

$$(X^{train}_{\cdot,(\hat{S}_{\tilde{L}}^{L_2})^{stab}(*_k)}, Y^{train}) \overset{SingBoost}{\longmapsto} \hat{\beta}^{sing,k} \quad (12.2.5)$$

Since the reduced data is usually rather small, the costs of this last SingBoost step are low, so we propose to use $M^{final} < M$ for the singular iteration frequency.

Finally, the decrypted CMB-3S algorithm looks as in algorithm 15.

12.3 The choice of M

So far, we did not make any recommendation for the choice of M. As we have seen, a single SingBoost model is in general neither stable nor sparse, so approaches like cross validation do not seem reasonable at that stage. Moreover, since CMB only reports an empirical column measure but no coefficients, there is no chance to perform cross validation for CMB.

The only chance that we have is to compute a cross-validated performance of the CMB-3S models. Maybe it would already be suitable to perform the cross validation on a CMB-3S model based on one single partition of the data (see section 12.4 for more details) to get an intuition if the data set with the particular choice of $\tilde{L}$ requires many singular iterations or not to save computational time compared to a cross validation with whole CV.CMB-3S procedures.

```
Initialization: Same as in 12, additional binary variables useZeta and singcoef,
 frequency $M^{final}$;
Same as in CMB-3S until step (STAB);
if singcoef==TRUE then
    Compute the coefficients in step (STAB) according to (12.2.5)
else
    if useZeta==TRUE then
        Compute the coefficients in step (STAB) according to (12.2.3)
    else
        Compute the coefficients in step (STAB) according to (12.2.4)
    end
end
Step (COEF): if singcoef==TRUE then
    Compute the final coefficients in the sense of (12.2.5) on $\mathcal{D}^{(train,k_{opt})}$
else
    if useZeta==TRUE then
        Compute the final coefficients in the sense of (12.2.3) on $\mathcal{D}^{(train,k_{opt})}$
    else
        Compute the final coefficients in the sense of (12.2.4) on $\mathcal{D}^{(train,k_{opt})}$
    end
end
```

Algorithm 15: CMB-3S with coefficients

Meinshausen and Bühlmann [2010] already introduced stability paths for the Lasso Stability Selection. These stability paths are understood, similarly as coefficient paths, as the (empirically estimated) probability that a variable j is selected by the Lasso, in dependence of the regularization parameter λ. These stability paths provide a useful visualization that directly clarifies which variables will be chosen for which cutoff. Note that in the case of Lasso Stability Selection, a variable is included in the stable set if for any λ, the corresponding empirical selection frequency exceeds the threshold, i.e., if the stability path intersects with the graph of $f(\lambda) \equiv \pi_{thr}$ anywhere or if it is even located above this graph everywhere, i.e., for all λ in the grid Λ.

For Boosting with Stability Selection (Hofner et al. [2015]), there also exist stability paths which describe the relative selection frequency in dependence of the number m_{iter} of Boosting iterations. More precisely, for each $m = 1, ..., m_{iter}$, the value of the stability path of variable j corresponding to m is the relative selection frequency of variable j, averaged over all B models, if the respective Boosting models were artificially truncated after the $m-$th iteration.

For SingBoost, stability paths are problematic since they make no difference between a standard iteration and a singular iteration. Moreover, the stability paths of Meinshausen and Bühlmann [2010] and Hofner et al. [2015] illustrate, in our language, the empirical column measure which enters the Stability Selection which would not be true for CMB-3S since a CMB column measure enters the Stability Selection and no SingBoost column measure. Therefore, an analog to stability paths for CMB-3S would be the relative selection frequencies of the variables after CMB which indeed enter the Stability Selection.

Since the selection of variables of a potential singular part itself depends on the frequency M of singular iterations performed in SingBoost, our **CMB stability paths** must depend on M, so we need some grid M_{grid}. We propose indeed to plot the components of $|M_{grid}|$ different empirical column measures provided by CMB on the same subsamples but with different $M \in M_{grid}$ against the elements of M_{grid}.

Admittedly, these plots do not look nice since the CMB column measures are in general not yet sparse. The heuristic intuition behind the singular stability paths is that for a high frequency of singular parts, i.e., for decreasing M, we get more chances to detect variables from singular parts, i.e., there may exist some paths in the plot that evolve from the right to the left.

To save computational time, it may indeed suffice to think of a **compromise between stability paths and coefficient paths**. Since we are primarily interested in detecting singular parts, we propose to draw B subsamples from the data and to perform several SingBoost models on each subsample, with different M, respectively. We define **SingBoost stability paths** in the sense that we plot the aggregated relative selection frequencies against the frequencies of the singular steps. Although the aggregated selection probabilities are not $0/1-$normed and there is no chance to distinguish between noise variables and relevant variables (which we tried to partially achieve in CMB by taking only the variables from the best models w.r.t. their out-of-sample performance), the shape of these SingBoost stability paths can be a useful indicator for singular parts (when some paths increase from the right to the left) and is computationally rather cheap, compared to the CMB stability paths.

See figures 12.1 and 12.2 for an example with $\tilde{L} = L_1$.

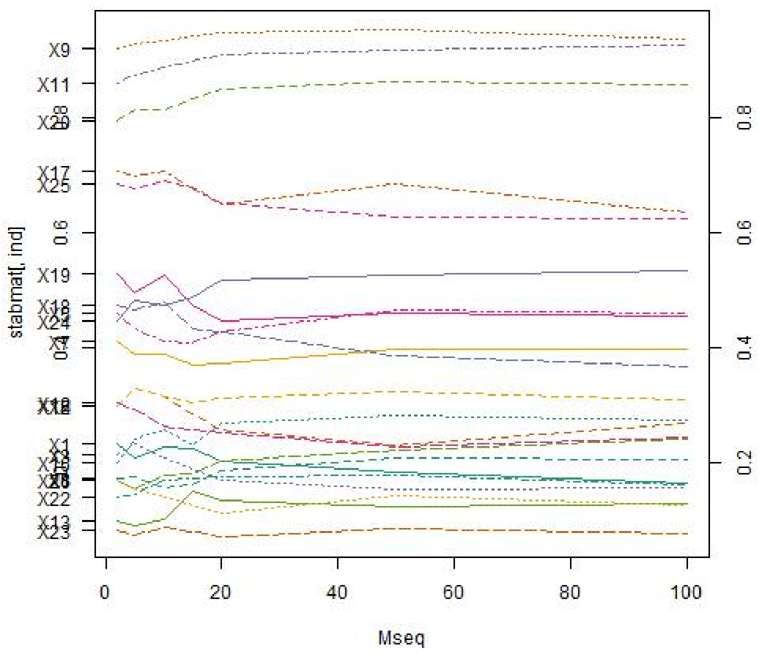


Figure 12.1: CMB stability paths on a data set with $p = 25$, $n = 150$ and $\tilde{L} = L_1$

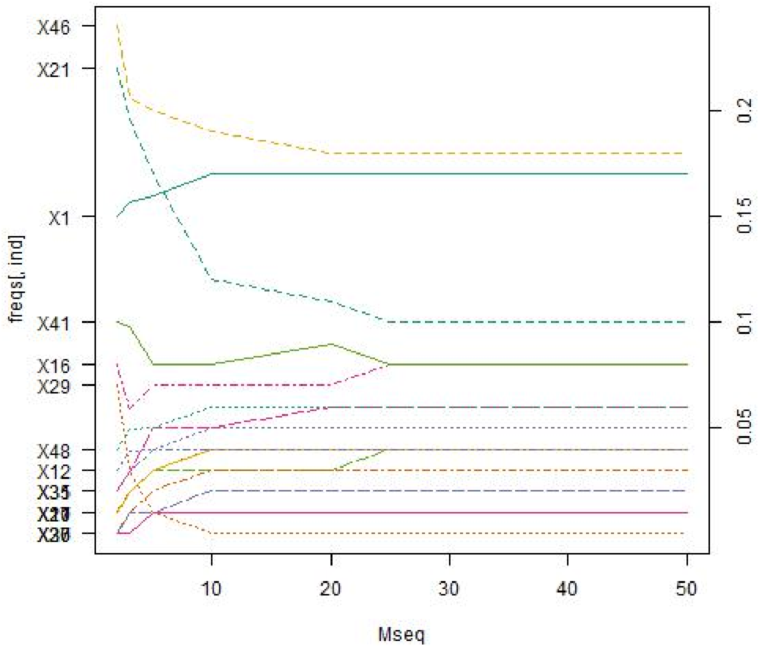


Figure 12.2: SingBoost stability paths on a data set with $p = 50$, $n = 250$ and $\tilde{L} = L_1$

Remark 12.3.1 (**Monotonicity**)**.** *Note that our stability paths are non-monotonic since they describe the evolvement of the relative selection frequencies for sequences of values of M. In contrast, the stability paths for Boosting (see Hofner et al. [2015]) are monotonically increasing since the Boosting-specific relative selection frequency is not accounted for but only the indicator if a variable is contained in the model is used. More precisely, if a variable had been chosen by l of the Boosting models in the first $k-th$ iterations, it clearly has also been chosen by at least l of the Boosting models in the first $\tilde{k} > k$ iterations.*

Remark 12.3.2 (**Measuring a similarity of loss functions**)**.** *Throughout this thesis, we assumed that there are potential singular parts between column measures of loss functions L and $\tilde{L}$ in the sense that the corresponding true column measures are not equivalent.*

We postulate that the approximation of a measure $\nu^{(\tilde{L})}$ that is "more similar" to $\nu^{(L_2)}$ than $\nu^{(L)}$ requires less singular iterations in SingBoost. The CMB stability paths, the SingBoost stability paths and the cross validation w.r.t. M are initial heuristic methods to get a closer look on the necessary degree of adaption to the target loss function in order to reasonably perform the change of measure.

If we were able to determine this kind of similarity between loss functions through the similarity of their corresponding column measures in advance (probably given a particular data set), we could optimize $\tilde{L}$ by using only very few singular steps, i.e., the loss in performance w.r.t. L_2-Boosting would be rather low. If one even knew that the column measure w.r.t. a standard loss function L dominates the column measure w.r.t. $\tilde{L}$, we could replace SingBoost by $L-$Boosting and only perform the change of measure in CMB and the $\tilde{L}-$adapted Stability Selection without having to perform any singular iteration.

*Once similarity quantifications were available, one could even think of **clustering loss functions** in the sense that the loss functions from a cluster have no or negligible singular mass. Then one could define some representant or centroid loss function which is the computationally cheapest.*

Remark 12.3.3 (**Sequences of loss functions**)**.** *We already mentioned Sparse Boosting (Bühlmann and Yu [2006]) where the best variable in each iteration is determined by a penalized L_2-loss, leading to possibly different choices. Let us assume the case that $\tilde{L}$ is very difficult to evaluate, but that there is some loss function $\check{L}$ which is close to $\tilde{L}$ in the sense of the previous remark. Then it could be beneficial to use $\check{L}$ in the most singular iterations, but to also use $\tilde{L}$ itself in a very few iterations to even respect minor singular mass of $\nu^{(\tilde{L})}$ w.r.t. $\nu^{(\check{L})}$. An extension would be to define a whole sequence of loss functions that approximate $\tilde{L}$.*

12.4 Stabilized Stability Selection and cross validation

The CMB-3S algorithm is already completed by the techniques stated in the previous subsection and the previous section. But the whole algorithm is still based on one partition into $\mathcal{D}^{train}$, $\mathcal{D}^{valid}$ and $\mathcal{D}^{test}$ of the data $\mathcal{D}$.

In order to reasonably compare the performance of CMB-3S to the performance of a competitor algorithm, we need numerous different partitions of the data in the sense of cross validation. This leads to the following CV.CMB-3S algorithm. To make it perfectly clear which objects we have at each step, we spell out all sub- and superscripts that we previously suppressed (see algorithm 16 and figure 12.3). Due to space constraints, we skip the (ROB) step.

Remark 12.4.1. *Computing the cross-validated hard ranking loss of a final model which is based on the hard ranking loss with already specified parameters n_{sing}, n_{cmb}, B^{sing}, B and α, with defaults for M, m_{iter} and κ and with a Stability Selection based on a $\pi-$grid with a pre-specified grid* `grid` *and a given allocation of partitions into training, validation and test data represented by an object* `CVind` *and with final coefficients computed by SingBoost where each second iteration is a singular iteration would be done by the following command line:*

```
CV.CMB3S(D, nsing=nsing , Bsing=Bsing ,B=B, alpha=alpha , singfam=Rank() , evalfam=
    Rank() , sing=T, LS=F, wagg=wagg , gridtype='pigrid ' , grid=grid ,ncmb=ncmb , CVind=
    CVind, targetfam=Rank() , useZeta=F, singcoef=T, Mfinal=2)
```

For further details on its flexibility and the meaning of `evalfam` *and* `targetfam`*, see part VI.*

Remark 12.4.2 (Parallelization). *When facing a rather complicated loss function $\tilde{L}$ like a ranking loss, it requires parallelized architecture to run CV.CMB-3S. A parallelization is evidently possible at each stage, i.e., the inner CMB can be parallelized as already spelled out in algorithm 8, but also the CMB-3S and the outer cross validation can be parallelized.*

Remark 12.4.3 ($V-$fold cross validation). *As usual, the partition into training, validation and test data does not have to be randomly (though it is recommended to do so to guard against peculiarities) but can also be done in the sense of $V-$fold validation where the $v-$th fold forms the test data and the remaining rows are similarly split up into folds, thus for every fixed v, we essentially may have more than one partition into validation and training data.*

```
**Initialization:** Hyperparameters as in algorithm 15, number $V$ of cross validation steps;
**for** *v=1,...,V* **do**
    Get $\mathcal{D}^{(v,train)} \in \mathbb{R}^{n_{train} \times p}$, $\mathcal{D}^{(v,valid)} \in \mathbb{R}^{n_{val} \times p}$ and $\mathcal{D}^{(v,test)} \in \mathbb{R}^{n_{test} \times p}$;
    **for** $b = 1, ..., B$ **do**
        Draw a subsample $\mathcal{D}^{(v,b,CMB)} \in \mathbb{R}^{n_{cmb} \times (p+1)}$ from the data according to $\zeta^{init}$;
        **for** $b^{sing} = 1, ..., B^{sing}$ **do**
            Draw a subsample $\mathcal{D}^{(v,sing,b,train)} \in \mathbb{R}^{n_{sing} \times (p+1)}$ from $\mathcal{D}^{(v,b,CMB)}$ according to $\zeta^{init}$. The
              non-selected rows form the set $\mathcal{D}^{(v,sing,b,test)}$ ;
            **if** *sing==TRUE* **then**
                **Step (SING):** Perform SingBoost on $\mathcal{D}^{(v,sing,b,train)}$ with the given input parameters
                  and get model $\hat{f}^{(v,b,b^{sing})}$ and empirical column measure $(\hat{\nu}_{\tilde{L}}^{L_2})^{(v,b,b^{sing})}$;
            **else**
                **Step (GLMBOOST):** Perform standard Gradient Boosting on $\mathcal{D}^{(sing,b,train)}$ w.r.t.
                  $\tilde{L}$ with the given input parameters and get model $\hat{f}^{(v,b,b^{sing})}$ and empirical column
                  measure $(\hat{\nu}^{(\tilde{L})})^{(v,b,b^{sing})}$;
            **end**
            **Step (CoM):** Compute the scores $\hat{s}^{(v,b,b^{sing})}$ in the sense of (11.4.1);
            Compute the weights $\hat{w}^{(v,b,b^{sing})}$ either in the sense of (11.4.2) or (11.4.3)
        **end**
        **Step (W-AGG):** Compute the aggregated column measure $(\hat{\nu}_{\tilde{L}}^{L_2})^{(v,b)}$ in the sense of (11.4.4)
          and the aggregated row measure $(\hat{\zeta}_{\tilde{L}}^{L_2})^{(v,b)}$ in the sense of (12.2.2);
    **end**
    **Step (SUB-AGG):** Average the $B$ column measures and get the aggregated empirical column
      measure $(\hat{\nu}_{\tilde{L}}^{L_2})^{(v,CMB)}$ and the aggregated empirical row measure $(\hat{\zeta}_{\tilde{L}}^{L_2})^{(v,CMB)}$;
    **Step (STAB): if** gridtype=='qgrid' **then**
        **for** $k = 1, ..., |q_{grid}|$ **do**
            Get the stable model $(\hat{S}_{\tilde{L}}^{L_2})^{(v,stab)}((q_{grid})_k)$ according to (12.1.1);
            Compute the coefficients on the reduced data $\mathcal{D}^{(v,train,k)} = \mathcal{D}^{(v,train)}_{\cdot,(\hat{S}_{\tilde{L}}^{L_2})^{(v,stab)}((q_{grid})_k}$ in the
              sense of (12.2.3) , (12.2.4) or (12.2.5) depending on useZeta and singcoef and get the
              loss $(\tilde{L}_n)^{(v,valid,k)}$ on the validation data $\mathcal{D}^{(v,valid)}$
        **end**
    **else**
        **for** $k = 1, ..., |\pi_{grid}|$ **do**
            Get the stable model $(\hat{S}_{\tilde{L}}^{L_2})^{(v,stab)}((\pi_{grid})_k)$ according to (12.1.2);
            Compute the coefficients on the reduced data $\mathcal{D}^{(v,train,k)} = \mathcal{D}^{(v,train)}_{\cdot,(\hat{S}_{\tilde{L}}^{L_2})^{(v,stab)}((\pi_{grid})_k)}$ in the
              sense of (12.2.3), (12.2.4) or (12.2.5) depending on useZeta and singcoef and get the
              loss $(\tilde{L}_n)^{(v,valid,k)}$ on the validation data $\mathcal{D}^{(v,valid)}$
        **end**
    **end**
    Choose the model corresponding to $k_{opt} = \mathrm{argmin}_k((\tilde{L}_n)^{(v,valid,k)})$;
    **Step (COEF):** Compute the final coefficients on $\mathcal{D}^{(v,train,k_{opt})}$ according to (12.2.3), (12.2.4) or
      (12.2.5) with $M = M^{final}$ depending on useZeta and singcoef. Call this final model $\hat{f}^{(v)}$;
    Get the test loss $\tilde{L}_n^{(v,test)}$ of model $\hat{f}^{(v)}$ w.r.t. $\tilde{L}$ on $\mathcal{D}^{(v,test)}$;
**end**
Get the cross-validated loss and the ultra-stable column measure according to (12.4.2), (12.4.1)
```

Algorithm 16: CV.CMB-3S

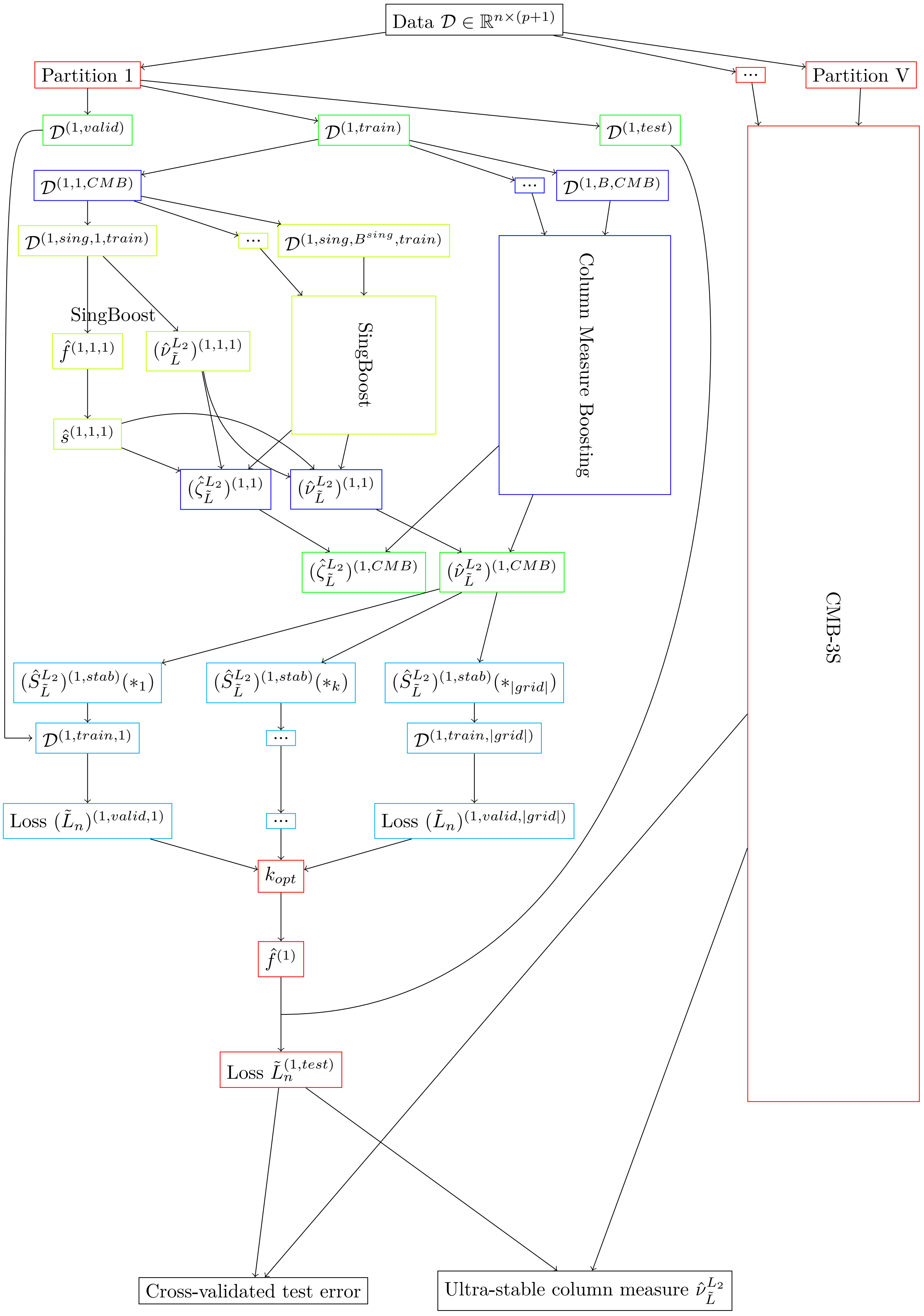

Data $\mathcal{D} \in \mathbb{R}^{n\times(p+1)}$
Partition 1
$\cdots$
Partition V
$\mathcal{D}^{(1,valid)}$
$\mathcal{D}^{(1,train)}$
$\mathcal{D}^{(1,test)}$
$\mathcal{D}^{(1,1,CMB)}$
$\cdots$
$\mathcal{D}^{(1,B,CMB)}$
$\mathcal{D}^{(1,sing,1,train)}$
$\cdots$
$\mathcal{D}^{(1,sing,B^{sing},train)}$
Column Measure Boosting
CMB-3S
SingBoost
$\hat{f}^{(1,1,1)}$
$(\hat{\nu}_{\tilde{L}}^{L_2})^{(1,1,1)}$
SingBoost
$\hat{s}^{(1,1,1)}$
$(\hat{\zeta}_{\tilde{L}}^{L_2})^{(1,1)}$
$(\hat{\nu}_{\tilde{L}}^{L_2})^{(1,1)}$
$(\hat{\zeta}_{\tilde{L}}^{L_2})^{(1,CMB)}$
$(\hat{\nu}_{\tilde{L}}^{L_2})^{(1,CMB)}$
$(\hat{S}_{\tilde{L}}^{L_2})^{(1,stab)}(*_1)$
$(\hat{S}_{\tilde{L}}^{L_2})^{(1,stab)}(*_k)$
$(\hat{S}_{\tilde{L}}^{L_2})^{(1,stab)}(*_{|grid|})$
$\mathcal{D}^{(1,train,1)}$
$\cdots$
$\mathcal{D}^{(1,train,|grid|)}$
Loss $(\tilde{L}_n)^{(1,valid,1)}$
$\cdots$
Loss $(\tilde{L}_n)^{(1,valid,|grid|)}$
k_{opt}
$\hat{f}^{(1)}$
Loss $\tilde{L}_n^{(1,test)}$
Cross-validated test error
Ultra-stable column measure $\hat{\nu}_{\tilde{L}}^{L_2}$

Figure 12.3: CV.CMB-3S diagram

Remark 12.4.4 (Stabilizing the Stability Selection). *As already mentioned, the final model and the final coefficients still depend on the partition of the data set $\mathcal{D}$ into training, validation and test data. If one really wants to stabilize even the Stability Selection, one can use the computed test losses from CV.CMB-3S as scores for one final aggregation step.*

We already have computed the sets

$$(\hat{S}_{\tilde{L}}^{L_2})^{(v,stab)}(*_{k_{opt}^{(v)}})$$

*of cardinalities $p_{k_{opt}}$ for $* \in \{\pi_{grid}, q_{grid}\}$ where we denote the optimal element index of the grid in the $v-th$ loop by $k_{opt}^{(v)}$. Combining this with the computed test losses $\tilde{L}_n^{(v,test)}$, we would finally get an **ultra-stable $\tilde{L}-$adapted empirical column measure** via*

$$(\hat{\nu}_{\tilde{L}}^{L_2})^{ultrastable} := \sum_v \frac{\frac{1}{\tilde{L}_n^{(v,test)}}}{\sum_w \frac{1}{\tilde{L}_n^{(w,test)}}} (I(j \in \hat{S}_{\tilde{L}}^{L_2})^{(v,stab)}(*_{k_{opt}^{(v)}}))_{j=1}^p \qquad (12.4.1)$$

which generally is not a 0/1-measure (we may alternatively also take equal weights). Obviously, the cross-validated test loss w.r.t. $\tilde{L}$ is given by

$$\tilde{L}_n^{test} = \frac{1}{V}\sum_v (\tilde{L}_n)^{(v,test)}. \qquad (12.4.2)$$

12.5 A systematic view of $\nu^{(L_2)}$−approximating techniques

This section again gives an overview of several strategies to approximate the true L_2−column measure $\nu^{(L_2)}$.

Since we assume thoughout this thesis that the true underlying model is sparse, traditional models that do not perform variable selection need not to be mentioned here.

12.5.1 Column measure framework $\rightsquigarrow$ Group (CM)

Algorithms that fall into this class are those which propose empirical column measures without any resampling of rows. In fact, for the loss L_2, we have for example:
i) Best Subset Selection,
ii) Lasso / LAD-Lasso / Huberized Lasso,
iii) Group Lasso,

iv) Sparse-Group Lasso,
v) L_2-Boosting and SingBoost,
vi) Clustering with model selection (with an L_2-distance).

This of course does not give any further insights into the similarity of Lasso and L_2-Boosting.

12.5.2 Row measure framework $\rightsquigarrow$ Group (RM)

Algorithms that fall into the row measure framework need some initial row measure ζ^{init} to sample from. Algorithms of this kind surely are able to provide aggregated column measures that are no longer necessarily 0/1-measures. Examples are:
i) Lasso with Stability Selection,
ii) L_2-Boosting with Stability Selection.

It is not surprising that aggregating several models leads to a stabilization of the set of selected variables, but we highlight the fact that **stabilizing a column measure is only possible through row measures**.

12.5.3 Row measure framework with random sampling from column measures $\rightsquigarrow$ Group (sRM)

Those algorithms form the important subset of (RM)-algorithms that also incorporate random sampling from column measures. With the terminology ”random sampling”, we exclude algorithms like L_2-Boosting which performs some kind of ”deterministic sampling” in each iteration by checking the performance of all p baselearners. Examples are:
i) Random forests,
ii) Block forests,
iii) Random Lasso.

12.5.4 Row column measure framework $\rightsquigarrow$ Group (RCM)

This type of algorithm does not only compute several column measures but also provides some kind of empirical row measures. Examples are:

i) Sparse Least Trimmed Squares,
ii) FRB,
iii) Tukey-Lasso,
iv) CMB-3S.

Those empirical row measures are not only useful for weighted sampling but also provide outlyingness evidence or even more precisely, **outlyingness w.r.t. some loss function**. Classical outlier detection procedures generally compute some robust distance of each row to some (robust) centroid in order to detect observations whose robust distance exceeds some threshold, indicating outlyingness. But for example in linear regression settings, one distinguishes between ”good” and ”bad” leverage points, so the first type of rows may have a large robust distance w.r.t. the remaining rows or some ”clean” subset, but they fit to the regression hyperplane anyway, so outlyingness w.r.t. the squared loss would not be given. See part V for further insights into robust procedures and their relation to variable selection strategies.

Modern approaches like FRB and SLTS indeed compare predictions and true responses, letting their empirical row measures get L_2−based.

Remark 12.5.1. *Even more precisely, the cited robust approaches that are based on residuals are essentially L−adapted for any loss function L that satisfies*

$$L(y, \hat{y}) = f(|y - \hat{y}|) = f(|r|)$$

for a monotonically increasing function f. As the ranking loss function does not belong to that class of loss functions (see section 10.4), we cannot anticipate that those techniques from robust statistics are tailored to ranking losses.

We want to emphasize that it is a quite surprising fact that **computing an empirical row measure requires the induction of this measure by a column measure**.

12.5.5 Adding noise to the regressor matrix $\rightsquigarrow$ Group (NX)

We would like to mention two techniques that also perform sparse model selection for the squared loss and therefore propose some column measure, but which manipulate the regressor matrix itself internally.

The first algorithm is the **Lasso-Zero (lass0)** from Descloux and Sardy [2018] that enriches the regressor matrix with additional noise columns that are meant to fit the error term ϵ. They work with the model

$$Y = X\beta + G\gamma$$

where G is a random matrix. The goal is to solve the Lasso-type problem

$$\min(||\beta||_1 + ||\gamma||_1) \quad s.t. \quad Y = X\beta + G\gamma.$$

The random matrix G is generated randomly several times, so a bunch of Lasso-Zero solutions is provided. Since the noise variables that are presumed to model the error term should have small coefficients in contrast to the true variables that model the response, the median of the coefficients is taken and some threshold function is applied to exclude variables whose median coefficient is too small. Their algorithm provides a 0/1-column measure.

The major difference of Lasso-Zero compared to the algorithms that we already listed is that the randomness does neither arise from sampling from some row measure nor from sampling from some column measure but in adding q columns with random entries to the regressor matrix.

A somewhat different approach is given in Davies and Duembgen [2018] and Davies [2018]. They define

$$SS_J := ||Y - \hat{Y}_J|| = ||Y||^2 - ||\hat{Y}_J||^2$$

where $J \subset \{1, ..., p\}$ and where $\hat{Y}_J$ is the orthogonal projection of Y onto the column space of the reduced matrix. Without assuming any distribution for Y, they derived a formula which allows to compute a $p-$value for the null hypothesis that all coefficients w.r.t. the columns in J^c are zero by evaluating the $F-$distribution.

This is used to quantify the importance of each variable. In fact, the authors propose a Forward Step-wise procedure where the residual sum of squares of the model including all still chosen variables and one of the remaining variables, say X_{j_0}, is compared with the residual sum of squares obtained by replacing all columns corresponding to the yet inactive variables with white noise and to use the already chosen variables and all noise variables to determine if the variable X_{j_0} has more predictive power than the white noise, i.e., if the null hypothesis that the true corresponding coefficient β_{j_0} is zero can be rejected or not.

12.5.6 Concluding remarks

Remark 12.5.2. *It is obvious that the inclusions (CM) $\subset$ (RM) and (sRM) $\subset$ (RM) hold. The group (CM) can be identified as a special case of (RM) where all rows are treated as "sample". The major difference of (sRM) and (RM)$\setminus$ (sRM) is that algorithms of type (sRM) always compute models on data sets whose columns have been reduced, potentially including a prioritization of certain columns according to extra knowledge which has been achieved by aggregation of Lasso models in the case of the Random Lasso or by using cross validation as in the case of Block forests.*

Remark 12.5.3 (**Stability vs. robustness**)**.** *We have seen that "stability" is essentially a property of empirical column measures which cannot be achieved but by computing models on subsampled or bootstrapped data according to some (prior) row measure. More precisely, stability of an empirical column measure means that its variability is low concerning other realizations from the same distribution which the pairs (X_i, Y_i) follow. Just regarding the terminology, we face the word "stabilization" in asymptotic statistics when computing a so-called variance-stabilizing transformation (using the delta method) of an estimator such that the asymptotic variance is independent from the estimated value, hence it is stabilized.*

In contrast to stability, "robustness" of a column measure in fact means that its variability is low even concerning realizations from other distributions. For example in regression settings, the quantification of outlyingness always requires the knowledge of residuals, so an interplay of row and column measures is inevitable when trying to robustify a supervised learning procedure. This is the reason why robustness of regression models is sometimes thought of being a property of the loss function which is not yet the whole truth since a "robust loss function" just truncates the residuals (to some extent) beforehand while concepts of regression MM-estimators or FRB iteratively propose weighted estimators, resulting in new residual-based weights (though also depending on robust loss functions). Note again that the Huberized Lasso just replaces the squared loss with the "robust" Huber loss function whereas the Tukey Lasso (see section 3.4) is a stage-wise algorithm that first computes an MM-estimator that produces weights, so it indeed internally computed empirical row measures!

We can summarize that robustness is a stricter property than stability which has been clarified by our framework. In fact, the group (RCM) is clearly a subset of the group (RM), but other inclusions do not hold.

Remark 12.5.4 (**Robustness and sparsity**)**.** *Our systematic view also provides answers on how robustness and sparsity are related. Again, sparsity is clearly a weaker property of* *an empirical column measure than stability or robustness since even the models from group (CM) already are able to provide sparse column measures, not accounting for stability nor robustness.*

*We can even go deeper into detail: From the perspective of robust statistics, the procedure proposed by Davies (Davies and Duembgen [2018], Davies [2018]) which replaces columns by white noise columns can be seen as invoking a **dependent contamination model** (see 3.3), but where the dependency does not mean that whole rows are either contaminated or not as in the classical contamination setting but in the sense that a whole column is contaminated, so it is evidently a special case of cell-wise contamination.*

Therefore, we can state that the cited procedure works well since it implicitly is based on the relation of sparsity and robustness, i.e., when assuming a sparse underlying model, contamination of cells of the regressor matrix which are not related to the response at all should not affect an estimator. For further insights, see part V.

12.6 Some possible extensions for future work

Remark 12.6.1 (False positives). *We do not provide theoretical results that connect the number of the variables that are selected in each Boosting model and the threshold π_{thr} with some quality criterion of the selected set of predictors as Meinshausen and Bühlmann [2010], Shah and Samworth [2013] or Hofner et al. [2015] do. While the true coefficients are known in a simulation setting, this is not the case for potential singular parts $J_{\tilde{L}}^{L}$ which are completely unknown in advance. Therefore, finding an appropriate criterion can be a topic for future research. However, we clearly also want to select a subset of the true set S^0 of variables, but the subset that better fits to $\tilde{L}$ instead of L_2. Therefore, false positives are the same in our setting, but the main difference is that the number q does refer to the final variable number in CMB-3S, so we cannot transfer the results from Hofner et al. [2015] to our setting that easily.*

Remark 12.6.2. *SingBoost and therefore CMB-3S is not yet capable to handle grouped variables. We already mentioned the Boosting variant BlockBoost (see section 2.5) which has to be transferred to SingBoost. Clearly, the structure is the same, i.e., in the singular steps, we fit a least squares model (or some other baselearner) which is based on all variables of a group, resulting in either selecting every variable in the group for a joint update or none of them.*

Taking a closer look on the resulting empirical column measure, we can conclude that all variables that form a group get the same selection probabilities, therefore the final stable model either includes the whole group of variables or the group is not represented at all.

Since we were able to represent the column measure on p columns essentially as $p-$dimensional vector, incorporating grouped variables leads to some block-wise structure, considering each non-grouped variable as single block, leading to the situation that we already described in remark 11.2.1.

Remark 12.6.3 (Weighted SingBoost?). *So far, we considered the initial row measure ζ^{init} as vector of sampling weights. Clearly, if we knew that some instances may be contaminated, it is reasonable to downweight them. If those instances are furthermore bad predictors for the ranking structure, the corresponding empirical row measure $\hat{\zeta}_{\tilde{L}}^{L}$ would be very low for those instances either. Similarly, imbalanced data may require better initial row measures than the uniform measure for resampling. But in fact, we did not directly incorporate those weights in fitting SingBoost models which we also avoided when fitting the final coefficients with a weighted least squares model according to this empirical row measure.*

If one would like to incorporate weights in SingBoost, one should first concern about the question if weighting is reasonable for ranking losses. Indeed, we propose to define weights $w_1, ..., w_n$ (i.e., an initial row measure) with

$$w_i \geq 0 \ \forall i, \quad \sum\sum_{i \neq j} w_i w_j = n(n-1)$$

where the last condition should guarantee that the ***weighted empirical hard ranking loss***

$$L_n^{w,hard} = \frac{1}{n(n-1)} \sum\sum_{i \neq j} w_i w_j I((\hat{Y}_i - Y_j)(Y_i - Y_j) < 0)$$

is $[0,1]-$standardized. The interpretation is straightforward, i.e., a misranking of two less important instances does not hurt much.

As for the localized ranking problem, note that the second summand in the localized ranking loss (see (5.2.5)) already distributed weights $w_i = I(i \in Best_K)$. However, we may face a conceptual problem: Assuming that some of the best instances actually have low weights, then it is not evident how one should proceed when fitting a weak or a localized ranking model. This emphasizes the problem which we face for the continuous localized ranking problem when we actually want to predict which responses are the highest ones but where we have to be aware that these responses are not just outliers.

Generally, one could also pose the question if the weights that could be used to provide weighted $\tilde{L}-$losses in the singular steps should be the same as the ones that are used in the L_2-iterations.

Remark 12.6.4 (The localized ranking problem). *Of course, it would be algorithmically possible to insert the localized ranking loss into CMB-3S. As hyperparameter, we do not insert a fixed K but the proportion of the best instances, i.e., K/n. But since we were forced to use this fixed proportion also on the subsamples and especially on the internal test sets, the whole models would rely on potentially very few observations which destabilizes the algorithm.*

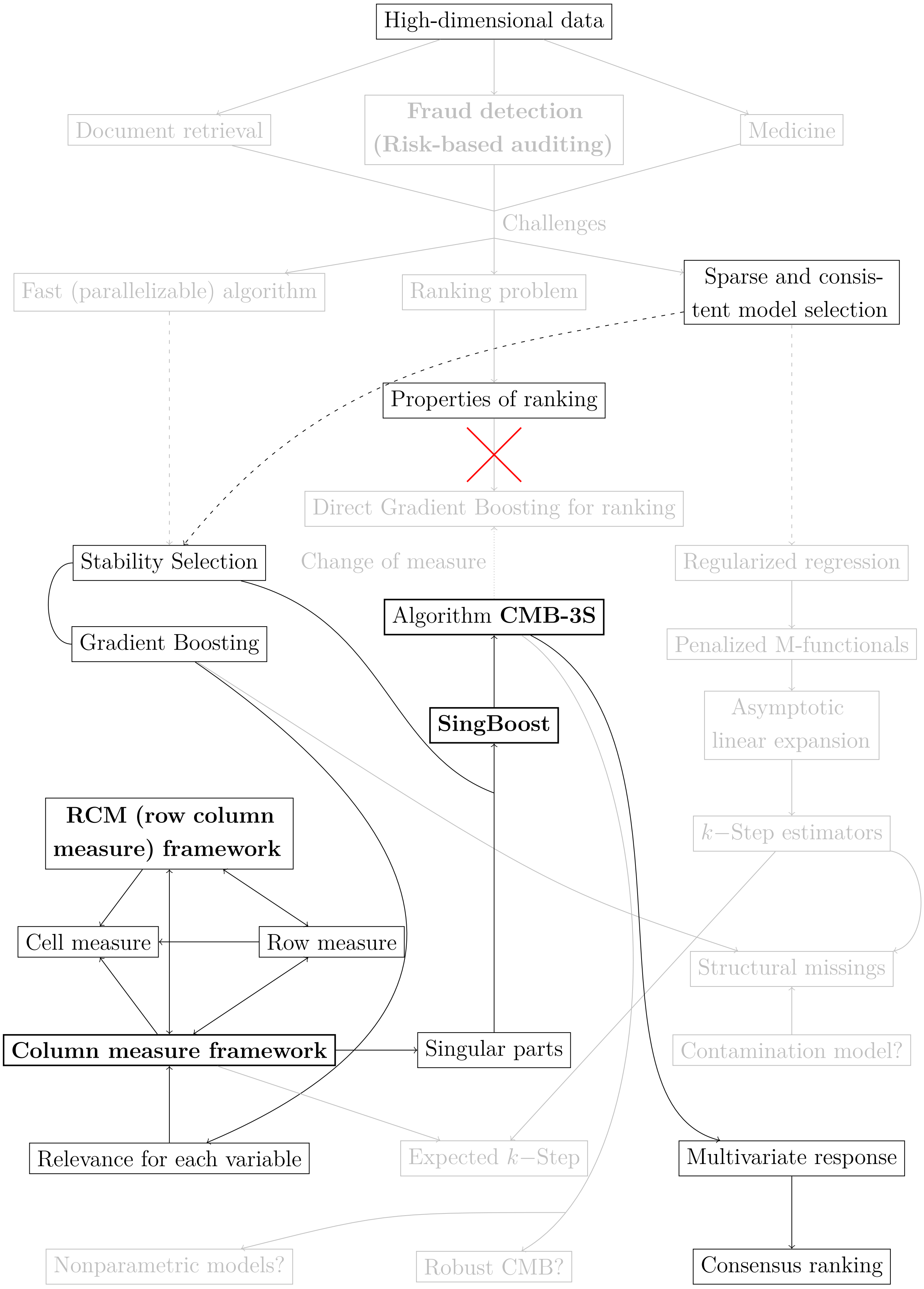
To overcome this issue, we recommend for now to solve the hard ranking problem. For comparison of those hard ranking models according to some localized ranking loss, one can use different family objects for `singfam`*,* `evalfam` *and* `targetfam`*, see part VI for details concerning the flexibility of our algorithms.*

Another idea can be to use some sort of weighted hard ranking loss as in remark 12.6.3 where for $Y_{(1)} \geq ... \geq Y_{(n)}$, we would distribute weights satisfying $w_{(1)} \geq ... \geq w_{(n)} > 0$ such that we still use all observations for fitting. This can be seen as a compromise between the theoretical results of Clémençon (Clémençon and Vayatis [2007]) for the localized ranking problem and the hard ranking problem itself. The generalized localized ranking loss would be the same as the localized ranking loss except for the fact that the hard ranking loss in the first summand would be replaced by some weighted hard ranking loss.

This would nullify the issue that we do not want to include weights in the L_2-iterations of SingBoost since the weights would only be used for computing the respective ranking losses.

Part V

Extensions of the RCM framework


High-dimensional data
Document retrieval
Fraud detection (Risk-based auditing)
Medicine
Challenges
Fast (parallelizable) algorithm
Ranking problem
Sparse and consistent model selection
Properties of ranking
Direct Gradient Boosting for ranking
Change of measure
Stability Selection
Gradient Boosting
Algorithm CMB-3S
Regularized regression
Penalized M-functionals
SingBoost
Asymptotic linear expansion
RCM (row column measure) framework
$k-$Step estimators
Cell measure
Row measure
Structural missings
Column measure framework
Singular parts
Contamination model?
Relevance for each variable
Expected $k-$Step
Multivariate response
Nonparametric models?
Robust CMB?
Consensus ranking

High-dimensional ranking and regression is not restricted to the case of univariate responses which we treated until here. For example, multivariate responses are not uncommon in biological studies (Wille et al. [2004], Kriegel et al. [2004]). This setting is indeed very interesting from our column measure perspective since the response columns may have different relevant variables, leading to singular parts between response-column-specific column measures.

The first chapter transfers the concept of singular parts and Stability Selection to the case of multivariate responses. We will see that ranking problems lead to a so-called consensus ranking which can be seen as an aggregation of partial rankings. We provide ideas how to suitably aggregate the empirical column measures in order to get a stable aggregated column measure representing the importances of the variables w.r.t. the consensus ranking.

We furthermore investigate the multivariate L_2-Boosting algorithm of Lutz and Bühlmann [2006b] and adapt the consistency proof to a prototype multivariate SingBoost variant.

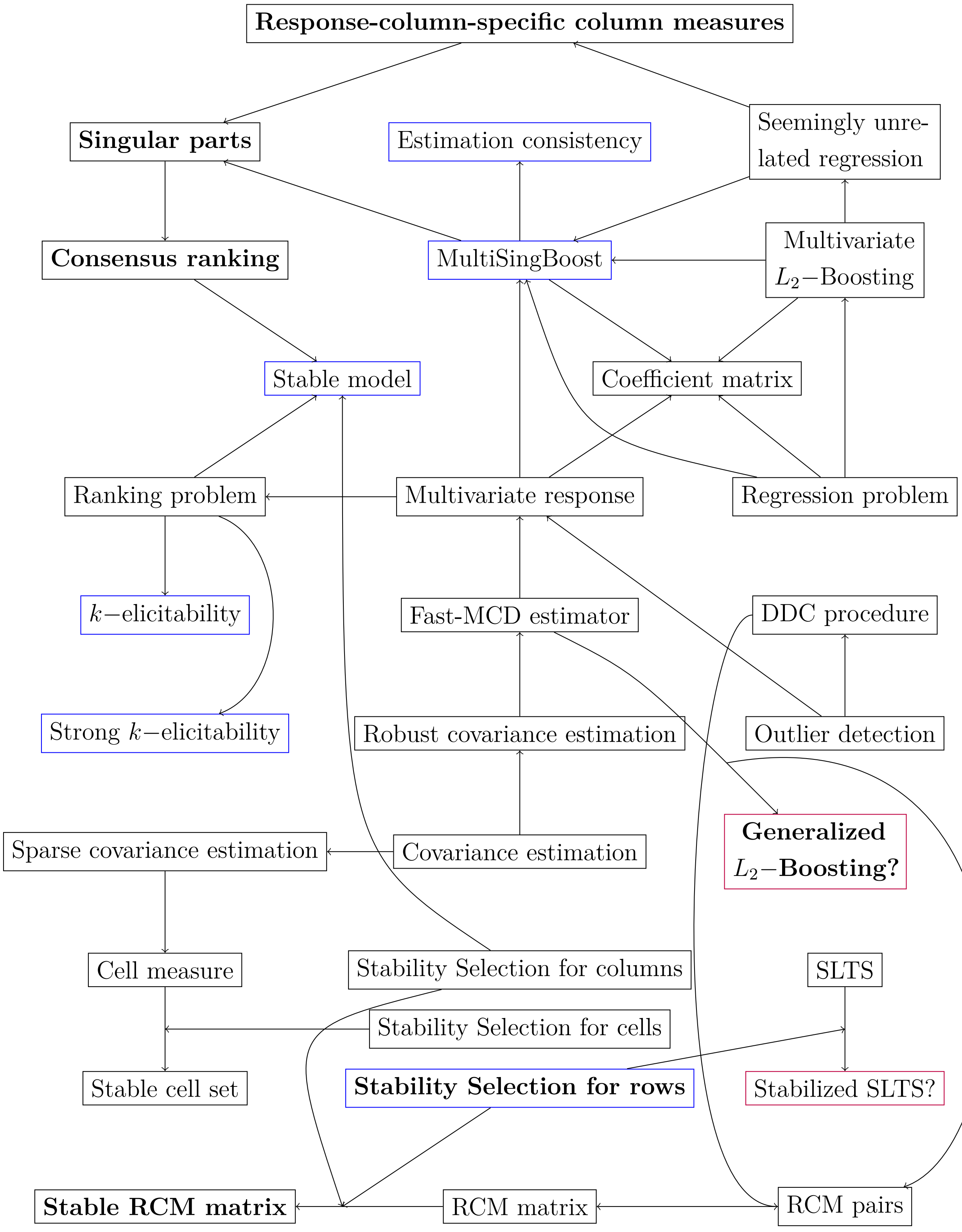
The chapter ends with two lemmas that show that multivariate hard ranking functionals are $k-$elicitable and strongly $k-$elicitable if the response columns are uncorrelated.

The second chapter introduces the cell measure which becomes relevant when performing sparse covariance or precision matrix estimation.

We continue to show that even outlier detecting algorithms like the DDC procedure and the Fast-MCD estimator for robust covariance estimation can be embedded into our RCM framework.

We furthermore identify the Fast-MCD estimator, a classical outlier detecting procedure as well as the SLTS algorithm as special instances of some Generalized L_2-Boosting algorithm which we propose at the end of the third section.

The last section provides an idea for a stabilization of SLTS in the sense of performing a Stability Selection for the columns and for the rows. We discuss this Stability Selection for the rows in the general case and make suggestions how one could combine it with outlier detecting algorithms or Generalized L_2-Boosting.


Response-column-specific column measures
Singular parts
Estimation consistency
Seemingly unrelated regression
Consensus ranking
MultiSingBoost
Multivariate L_2-Boosting
Stable model
Coefficient matrix
Ranking problem
Multivariate response
Regression problem
$k-$elicitability
Fast-MCD estimator
DDC procedure
Strong $k-$elicitability
Robust covariance estimation
Outlier detection
Generalized L_2-Boosting?
Sparse covariance estimation
Covariance estimation
Cell measure
Stability Selection for columns
SLTS
Stability Selection for cells
Stable cell set
Stability Selection for rows
Stabilized SLTS?
Stable RCM matrix
RCM matrix
RCM pairs

Chapter 13

Multivariate ranking and regression

The first section of this chapter is devoted to the so-called consensus ranking problem where the task is to find an overall ranking based on partial rankings. We address to the question how one could reasonably select an overall set of relevant predictors using our CMB-3S algorithm and takes column measures into account. We will see that the multivariate responses lead to a new potential source of singular parts between column measures when concerning column-specific column measures, separately for each response column.

The second section starts with a short recapitulation of the multivariate L_2-Boosting algorithm from Lutz and Bühlmann [2006b] which is also capable to handle the case of correlated responses. We propose a first idea how to construct a multivariate SingBoost algorithm and show that this algorithm would be consistent even in very high dimensions.

The chapter ends with a very brief review on higher-order elicitability. We show that in the case of uncorrelated responses, multivariate ranking functionals indeed satisfy the properties of $k-$elicitability and strong $k-$elicitability.

13.1 Consensus ranking

There are many situations where we have partial rankings and want to get a suitable combined ranking based on these partial rankings. Such situations range from the ranking of websites by different search engines (Dwork et al. [2001]) to the combination of judge grades in competitions (Davenport and Lovell [2005]) and even to applications in nanotoxicology (Patel et al. [2013]). The aggregation of the partial rankings gets even more difficult if the quality of the partial rankers is different (Deng et al. [2014]).

Therefore, suppose that the response variable is multivariate, i.e., in this section we have $\mathcal{Y} \subset \mathbb{R}^k$, $k \geq 2$. Then we can clearly get partial rankings which are rankings for each column of Y separately. However, since we are actually interested in the ranking of the rows X_i which in the case of univariate responses just equals the ranking of the Y_i, it remains to find an overall ranking for the X_i in the case that each response column corresponds potentially to a different ranking. A standard technique to get a so-called **consensus ranking** is the Kemeny ranking aggregation (Kemeny [1959]). Having predicted permutations $\pi^{(r)}$, $r = 1, ..., k$, for each column of Y, the goal is to find a overall ranking via

$$\pi^* = \underset{\pi \in \mathrm{Perm}(1:n)}{\operatorname{argmin}} \left(\sum_{r=1}^{k} L_n^{hard}(\pi^{(r)}, \pi) \right).$$

Korba et al. [2017] identified this problem as statistical learning problem since the term in the brackets is an empirical loss (they even allowed to replace the hard ranking loss by other distance measures d between permutations). We do not see any reason why a localized or even weak ranking losses as distance measure should not be allowed which would imply that the consensus ranking is the permutation that resembles the partial rankings most only concentrating on the top of the list. However, there does not seem to be an algorithm yet that can compute the consensus ranking for arbitrary distance measures between permutations.

Note that Korba et al. [2017] worked with an i.id. assumption of the $\pi^{(r)}$ which in the case of a multi-response problem obviously does not need to be true, neither our following assumption that we state for simplicity.

Assumption 13.1.1. *Throughout this section, we assume that the response columns are uncorrelated.*

Although there exist algorithms that are able to compute such a consensus ranking, all the information provided by the predictive models that led to each of the partial rankings, respectively, is lost, i.e., there is no coefficient vector that models the consensus ranking and no empirical column measure for the consensus ranking. We already referred to this weakness in the context of ranking forests which required to compute a consensus ranking based on the partial rankings provided by each ranking tree (see section 5.5).

Even worse, if our predictive models are sparse, they do not necessarily have selected the same variables. But if the ranking model r has selected the predictor set $\hat{S}^{(r)}$, $r = 1, ..., k$, we cannot directly determine a set $\hat{S}^{cons}$ which corresponds to the consensus ranking. Taking the union of all sets $\hat{S}^{(r)}$ would be problematic especially in cases like that there exists a variable j^* which appears in only one of these predictor sets. Finally, a simple computation of selection frequencies in the sense of Stability Selection, i.e., the relative number of the subsets $\hat{S}^{(r)}$ in which a particular variable is contained, would delete all the information on the degree of importance of that variable in the models in which it is contained.

We provide the following approach to compute a **sparse and stable parametric consensus ranking model** by using our CMB-3S algorithm. Given training data $\mathcal{D}^{train}$ where the rankings for each column of the response vector are already known, we can directly compute a consensus ranking using the R$-$package `ConsRank` (D'Ambrosio et al. [2017]). One possible way to proceed would be to treat the $n-$dimensional vector π^* as pseudo-response and to compute an ordinal regression model like Ordered logit or (more likely) an Ordinal Boosting model (which is implemented in `mboost` and corresponds to the `family` object `propOdds()`). This would be a naïve approach since the original continuous responses would be replaced by discrete-valued responses in the last stage, so there is no reason to assume that even the model selection performed by the Boosting model would still be appropriate for the original responses. In other words, the task is to **find a suitable pseudo-response for the consensus ranking**.

A first approach to address to this problem is to determine

$$r^* := \underset{r}{\operatorname{argmin}}(d(\pi^{(r)}, \pi^*)),$$

so we search for the permutation which has the smallest distance d to the consensus ranking. Then, we only use the response column $Y_{\cdot,r^*}$ and proceed with computing a CMB-3S ranking model using only $Y_{\cdot,r^*}$, so we are again in the usual case with one-dimensional responses. This could be seen as some kind of dimension reduction of the response matrix Y. Since we are in the position that we have distances, we can identify our problem as a variant of a clustering problem in the sense that $Y_{\cdot,r^*}$ is the **centroid or representant for the whole response** Y in terms of hard ranking. This approach would directly lead to a new issue. If there are relevant variables for the ranking loss $\tilde{L}$ corresponding to response column r_1 which is not relevant when concerning response column $r_2 \neq r_1$, we face **a new type of singular parts between column measures**, this time not in dependence of some loss function but w.r.t. components of the response. We call our column measures $\nu^{(r)}$ in this section (we suppress the usual sub- and superscript since we always use the hard ranking loss as loss $\tilde{L}$).

We propose to perform an aggregation of ultra-stable column measures. First, we can easily use CV.CMB-3S separately for each response column $Y_{\cdot,r}$, $r = 1, ..., k$, to get coefficients $\hat{\beta}_j^{(r)}$ and ultra-stable empirical column measures $(\hat{\nu}^{(r)})^{ultrastable}$. With the distances $d_r := d(\pi^{(r)}, \pi^*)$, we define the weights

$$w_r := \frac{\frac{1}{d_r}}{\sum_{s=1}^k \frac{1}{d_s}}$$

that lead to the **consensus column measure**

$$\hat{\nu}^{cons}(\{j\}) := \sum_r w_r (\hat{\nu}^{(r)})^{ultrastable}(\{j\}). \tag{13.1.1}$$

Then we again perform a Stability Selection for $\hat{\nu}^{cons}$ and get a stable set $\hat{S}^{cons,stab}$. The final model is a least squares or SingBoost model with ranking singular steps using only the data $(X_{\cdot,\hat{S}^{cons,stab}}, Y_{r^*})$.

Remark 13.1.1. *Since Kendall's τ and the hard ranking loss are closely related in the sense of lemma 6.1.1, the distance-based aggregation of empirical column measures can still be seen as loss-based aggregation as it has been done in CMB with weights as in (11.4.3).*

Remark 13.1.2. *To save computational time, it may also be reasonable to aggregate CMB column measures $(\hat{\nu}^{(r)})^{CMB}$ instead of ultra-stable column measures in (13.1.1). Then the Stability Selection either would be similar or we maybe even would use **different cutoffs** $\pi_{thr}^{(r)}$ for each CMB column measure. Note that for a reasonable optimization, the second strategy would cause an immense grid search since we must visit all elements of $\times_r \pi_{thr}^{(r)}$. However, this approach would still be cheaper than computing ultra-stable column measures for each response column if p and k are large.*

Remark 13.1.3. *So far, we considered the computation of a model that represents the important variables for the consensus ranking on the training set. On a test set, we would apply this model to instantaneously get a predicted consensus ranking there. This strategy is a direct consequence of the nature of ranking problems since we are not interested in predicting some sort of aggregated response value but only an aggregated ranking. Note that a combination of the true values of the response columns would not be reasonable at all since different scalings of them would lead to the issue that the values of a single column can dominate all other values due to a larger scale.*

*Apart from the already presented strategy, we think that there can be an alternative way to work on test sets. We may apply the models that we computed based on each column of the response matrix separately, represented by the coefficient vectors $\hat{\beta}^{(r)}$, to predict the k response columns of the test data. Then, we proceed as we did on the training set by finding the **empirical** response column whose ranking has the smallest distance to the (empirical!) consensus ranking on the test set which we also compute using `ConsRank`. Combining the empirical column measures and generating a model for the consensus ranking works analogously as on the training set. The disadvantage of the second approach is that, unless on the training data where the consensus ranking can be treated as known if we rely on `ConsRank`, the consensus model cannot be validated since neither the partial rankings nor the consensus ranking are known on the test set. Furthermore, the prediction errors from the models on the* *training set may accumulate when computing the consensus model.*

Remark 13.1.4 (Ranking and hierarchical clustering?). *The main difficulty was to find a suitable centroid w.r.t. the ranking of the response columns. However, if k is rather large or if no column of the response vector has a reasonable small distance to the consensus ranking, this approach may get unreliable.*

Whenever we have (pair-wise) distances, we can think of applying clustering methods (see e.g. Friedman et al. [2001]). Maybe we can proceed as in hierarchical clustering in an agglomerative manner to cluster the response columns by means of the pair-wise distances of their rankings. Then we may compute models on response columns corresponding to the cluster of rankings in which the consensus ranking is contained. Since this cluster may in general do not correspond to enough different response columns, we think of convex combinations of the respective columns in order to artificially construct more pseudo-response columns on which we compute a CMB(-3S) model (w.r.t. an aggregated, stable column measure) in order to aggregate these models to get final coefficients. A concrete recommendation of which distance one should use, how many clusters one should compute (i.e., which evolution step of the hierarchical clustering has to be chosen) and how to generate the pseudo-responses can be a topic for future work.

13.2 Multivariate Boosting

13.2.1 Multivariate L_2-Boosting

The multivariate L_2-Boosting algorithm goes back to Lutz and Bühlmann [2006b] and can be seen as the extension of L_2-Boosting to multivariate response vectors, i.e., response matrices, which is more sophisticated than just applying L_2-Boosting on each response column separately.

We mainly follow the notation as in Lutz and Bühlmann [2006b]: A linear model for this case is given by

$$Y = X\beth + E$$

for $Y \in \mathbb{R}^{n \times k}$, $X \in \mathbb{R}^{n \times p}$, a coefficient matrix $\beth \in \mathbb{R}^{p \times k}$ with entries $\beta_{j,r}$ and the error matrix $E \in \mathbb{R}^{n \times k}$ which satisfies $\mathbb{E}[E_i] = 0_k$ and $\mathrm{Cov}(E_i) = \Sigma$ where the unknown quantity Σ is replaced by a suitable proxy Γ for example when evaluating the loss function (13.2.1).

The multivariate L_2-Boosting algorithm iteratively selects the **cell of the coefficient matrix** whose least squares fit w.r.t. the corresponding predictor improves the current fit most, quantified by a suitable multivariate analog of the squared loss of the form

$$\frac{1}{n}\sum_i L(Y_i, \hat{Y}_i) = \frac{1}{n}\sum_i (Y_i - \hat{Y}_i)^T \Gamma^{-1}(Y_i - \hat{Y}_i), \tag{13.2.1}$$

so one compares the differences of the $(1 \times k)-$dimensional rows of the response matrix and their predictions $\hat{Y}_i = X_i\hat{\beth} \in \mathbb{R}^{1\times k}$, respecting a potential correlation structure of the response columns. After having detected the best cell $(\hat{j}, \hat{r})$ in the coefficient matrix which means that the whole $\hat{r}-$th predicted response or residual column was fitted by a simple model based exclusively on the $\hat{j}-$th regressor column, the coefficient matrix is updated by adding the fitted coefficient

$$\hat{\beta}_{\hat{j},\hat{r}} = \frac{\sum_{r=1}^k R_{\cdot,r}^T X_{\cdot,j}\Gamma_{r\hat{r}}^{-1}}{X_{\cdot,j}^T X_{\cdot,j}\Gamma_{\hat{r}\hat{r}}^{-1}} \tag{13.2.2}$$

to the component in the respective cell of the current coefficient matrix, maybe respecting some learning rate κ. The residual matrix R is given by $R = Y - X\hat{\beth}$ with the current coefficient matrix $\hat{\beth}$. After updating the coefficient, the $\hat{r}-$th column of the residual matrix is updated and the algorithm proceeds until some stopping criterion gets valid which is considered to be a corrected AIC criterion. For further details on updating and how $(\hat{j}, \hat{r})$ is determined, see Lutz and Bühlmann [2006b].

As a variant of multivariate L_2-Boosting, they additionally propose an idea for a so-called **"row-Boosting"** which updates rows of $\hat{\beth}$ and not just cells which means that the whole predicted response matrix and therefore the whole residual matrix is updated after each iteration but that the baselearner is still computed using one single regressor column.

As pointed out in Lutz and Bühlmann [2006b], multivariate L_2-Boosting with updating cells is suitable for the case of **seemingly unrelated regression** going back to Zellner [1962] which implies that the different response columns are potentially based on different predictors. This is obvious since a row-wise update of the coefficient matrix means that a predictor variable has been selected simultaneously for all response columns. The concept of seemingly unrelated regression opens the door for the column measure framework, in which this phenomenon results in **singular parts of column measures w.r.t. different response columns** as in the previous section where the partial rankings may depend on different variables, respectively.

13.2.2 SingBoost for multivariate responses?

As a first approach, a SingBoost variant for multi-response data (**MultiSingBoost**) can be the straightforward extension of SingBoost for univariate data, i.e., we similarly determine the baselearners w.r.t. each cell of the coefficient matrix as multivariate L_2-Boosting does, but we do not evaluate the fit in a multivariate analog of the squared loss but we take another loss function into account.

Clearly, the computational time in the singular steps when updating cells would be enormous, especially if k is also very large. Row-wise updates can only be useful if one knew in advance that there are no singular parts or at least very few singular mass between the column measures w.r.t. each response column. It remains to investigate how one could test for it without wasting too much computational resources. The question if one could perform row-wise updates in a ranking setting obviously directly corresponds to the rank aggregation problem from the previous section since it is not evident if one should just sum up the ranking losses for all $r=1,...,k$ or if an overall criterion should be invoked.

However, if a Corr-min condition as in section 10.5 is satisfied, we can conclude that a MultiSingBoost is consistent even for diverging dimensions p_n and k_n (i.e., for $Y_{i;n}$, $E_{i;n} \in \mathbb{R}^{k_n}$, $\beth_n \in \mathbb{R}^{p_n \times k_n}$ and $X_{i;n} \in \mathbb{R}^{p_n}$ with i.id. rows) as the following theorem that we adapt from [Lutz and Bühlmann, 2006b, Thm. 1] states.

Theorem 13.2.1. *If the assumptions*

(M1) $\exists 0 < \xi < 1,\ 0 < C < \infty : p_n = \mathcal{O}(\exp(Cn^{1-\xi})),\ k_n = \mathcal{O}(\exp(Cn^{1-\xi}))$ *for* $n \to \infty$*,*

(M2) $\sup_n \left(\sum_{j=1}^{p_n} \sum_{r=1}^{k} |\beth_{j,r;n}|\right) < \infty$*,*

(M3) *The proxy* Γ *for the error covariance satisfies*

$$\sup_{n,r}\left(\sum_{l=1}^{k_n} |\Gamma_{r,l;n}^{-1}|\right) < \infty, \quad \inf_{n,r}(\Gamma_{r,r;n}^{-1}) > 0,$$

(M4) $\sup_j(||X_{1,j;n}||_{\infty}) < \infty$*,*

(M5) $\sup_r(\mathbb{E}[|E_{1,r;n}|^s]) < \infty \ \exists s > \frac{2}{\xi}$ *with* ξ *from (M1),*

(M6) $\exists a > 0 \ \ \forall m, n \ \exists \tilde{C} \geq a :$

$$|\langle \hat{R}_n^{m-1} f, g_{\widehat{j_m},\widehat{r_m}} \rangle_{(n)}| \geq \tilde{C} \sup_{j,r} (|\langle \hat{R}_n^{m-1} f, g_{j,r} \rangle_{(n)}|)$$

for the baselearner $g_{j,r}$ corresponding to the cell (j,r) of the coefficient matrix and for the empirical counterpart $\langle \cdot, \cdot \rangle_{(n)}$ of the inner product

$$\langle f, g \rangle = \int f^T(x) \Gamma^{-1} g(x) dP(x),$$

then for the estimated regression function $\hat{f}^{(m_n)}$ from the MultiSingBoost algorithm, it holds that

$$\mathbb{E}_X[||\hat{f}^{(m_n)} - f||^2] = \mathbb{E}_X[(\hat{f}^{(m_n)}(x) - f(x))^T \Gamma^{-1} (\hat{f}^{(m_n)}(x) - f(x))] = o_P(n^0)$$

provided that $(m_n)_n$ is a sequence with $m_n \to \infty$ for $n \to \infty$ sufficiently slowly and for a new observation x that follows the same distribution as the X_i, independently from them.

Proof. *[Lutz and Bühlmann, 2006b, Lem. 1] does not depend on a specified cell selection, so it holds anyway for a MultiSingBoost. According to the proof of [Lutz and Bühlmann, 2006b, Thm. 1], the rest of the proof for their theorem follows exactly the same steps as the proof of [Bühlmann, 2006, Thm. 1], so the Corr-min condition (M6) guarantees that the selection performed by MultiSingBoost is still well enough in terms of correlation with the current response which it does not alter the statement of the theorem asymptotically.*

□

13.3 Higher-order elicitability

The first work which extended the definition of elicitability to multivariate cases was, to the best of our knowledge, Lambert et al. [2008] who defined the property of $k-$elicitability. This concept became indeed popular once [Fissler et al., 2016, Cor. 5.5] showed that despite the Expected Shortfall is not elicitable (see [Gneiting, 2011, Thm. 11]), the pair $(\mathrm{VaR}_\alpha, \mathrm{ES}_\alpha)$ is indeed elicitable, more precisely $2-$elicitable.

In the language of Lambert et al. [2008] or Frongillo and Kash [2015], the elicitation complexity of the expected shortfall is 2 which means that it is $2-$elicitable but not "$1-$elicitable" (which corresponds to standard elicitability).

One result that we can directly apply for our context is [Fissler et al., 2016, Lemma 2.6] which we recapitulate for illustration using their notation.

Lemma 13.3.1. *Let $T_m : \mathcal{F} \to A_m$ be k_m-elicitable functionals relative to the class $\mathcal{F}$ with $A_m \subset \mathbb{R}^{k_m}$ and $k_m \geq 1$ for all $m = 1, ..., l$. Let $T = (T_1, ..., T_l) : \mathcal{F} \to A$ with $A = \bigotimes_m A_m \subset \mathbb{R}^k$ for $k = k_1 + k_2 + ... + k_l$. Then T is $k-$elicitable relative to the class $\mathcal{F}$.*

The proof is based on the linearity of the expectation, letting all loss functions $\sum_m a_i L_i$ for $a_i > 0 \; \forall i$ be strictly consistent for T if L_i is strictly consistent for T_i, respectively.

For multi-response ranking problems, we state the following results under assumption 13.1.1.

Lemma 13.3.2. *Let $n \in \mathbb{N}$, $n \geq 2$ be fixed. Let $T_1, ..., T_k : \mathcal{F} \to \mathrm{Perm}(1:n)$ be ranking functionals in the sense of definition 7.1.3. Then the functional $T = (T_1, ..., T_k) : \mathcal{F} \to \bigotimes_m \mathrm{Perm}(1:n)$ corresponding to the multi-response ranking is $k-$elicitable relative to the class $\mathcal{F}$.*

Lemma 13.3.3. *Let $n \in \mathbb{N}$, $n \geq 2$ be fixed and let $T_1, ..., T_k : \mathcal{F} \to A$ for $A = \mathrm{Perm}(1:n)$ be ranking functionals in the sense of definition 7.1.3. Let $(T_m)_P : \mathcal{F} \to \mathcal{P}(A)$ for all $m = 1, ..., k$. Then the functional $T_P := ((T_1)_P, ..., (T_k)_P) : \mathcal{F} \to \bigotimes_m \mathcal{P}(A)$ is strongly $k-$elicitable relative to the class $\mathcal{F}$.*

The proofs just apply lemma 13.3.1 in combination with our own results on elicitability and strong elicitability of ranking functionals (theorems 7.1.1 amd 7.2.1).

Remark 13.3.1. *Note that these results basically rely on the fact the we assume the partial rankings to be uncorrelated (see assumption 13.1.1). Therefore, the estimation of a multivariate response boils down to the separate estimation of each component. Under the stated assumption, one clearly can also generalize results on elicitability of regression functionals to the multi-response case by using the same arguments.*

However, it is not evident if one could prove a general result on $k-$elicitability of regression or ranking functionals if there is a correlation structure that has to be accounted for. Maybe one could also use the concepts of Maume-Deschamps et al. [2017] who proposed a general definition of multivariate elicitability of vector-valued functionals. This would exceed the scope of this work, but seems to be a promising direction for future research.

Chapter 14

Connecting robustness, stability and sparsity

The first section introduces the cell measure which is related to sparse estimation of covariance or precision matrices.

The second section restates sophisticated algorithms for outlier detection, robust estimation of covariance matrices and robust regression procedures, even in the case of cell-wise contamination. We show that these algorithms also fall into our RCM framework.

The third section is an approach to connect all three paradigms of stability, robustness and sparsity. We start with identifying classical outlier detection and even the MCD estimator as special multivariate L_2-Boosting algorithms. Additionally, we restate the SLTS algorithm as special L_2-Boosting algorithm in two different ways to motivate some Generalized L_2-Boosting which extends the variable selection and updating scheme of standard L_2-Boosting and which includes forgetting factors and a concentration step to guard against case-wise outliers. We go even further and provide initial ideas for Stability Selection for the columns selected by the Generalized L_2-Boosting and even for some kind of Stability Selection for the rows.

14.1 The cell measure

The row measure and the column measure are useful to represent the importance of observations resp. predictors, especially in regression settings, but as we already emphasized, the row measure is in particular useful whenever classical (row-wise) contamination appears and the column measure is also used in the clustering context when performing feature selection there. We also mentioned that multivariate responses lead to new potential singular parts since each response column can have its own column measure.

So far, we have not concerned about (the estimation of) covariance matrices, but it is obvious to ask if the RCM framework is also suitable in this context. At the first glance, this has to be negated since the partition of a covariance matrix into cells and rows is not meaningful due to its inherent structure which can be regarded as either cell- or diagonal-based. More precisely, reducing the number of observations does not affect the dimension of the estimated covariance or precision matrix, but reducing the number of columns clearly leads to smaller matrices. But in fact, we will see in this section that the RCM framework is still important when performing variable selection first or when contamination or missings are present.

Nevertheless, covariance matrices are related to cell measures in the sense of the following definition.

Definition 14.1.1. *Suppose we have a matrix $\Sigma \in \mathbb{R}^{p \times p}$. Then the* ***cell measure*** *is defined as the map*

$$\aleph : (\{1,...,p\}^2, \mathcal{P}(\{1,...,p\}^2)) \to ([0,p^2], \mathbb{B} \cap [0,p^2]),$$

$$\aleph : \{(i,j)\} \mapsto \aleph(\{(i,j)\}) =: \aleph_{(i,j)} \in [0,1] \ \forall (i,j) \in \{1,...,p\}^2$$

and assigns an importance to each cell.

The cell measure is specifically interesting when Σ is an estimated (sparse) covariance or precision matrix. Straightforward examples for the cell measure are the following ones.

Example 14.1.1. *Let $\hat{\Sigma} \in \mathbb{R}^{p \times p}$ be a classical estimator of a covariance matrix, based on a data set $\mathcal{D} = X \in \mathbb{R}^{n \times p}$. Then the cell measure is given by $\hat{\aleph}_{(i,j)} = 1$ for all $(i,j) \in \{1,...,p\}^2$.*

Example 14.1.2. *Let $\hat{\Sigma} \in \mathbb{R}^{p \times p}$ be some sparse covariance or precision matrix, e.g., by having applied the Graphical Lasso (see Banerjee et al. [2008] and Friedman et al. [2008]). Then, $\hat{\aleph}_{(i,j)} = 0$ for all $(i,j) \in \{1,...,p\}^2$ where $\hat{\Sigma}_{i,j} = 0$ and $\hat{\aleph}_{(i,j)} = 1$ otherwise.*

Example 14.1.3 (Initial cell measures). *Of course, if a matrix is only partially observed which is a standard issue for example in matrix completion problems that are especially imporant in image recognition applications (see for instance Candès and Recht [2009], Candès et al. [2011]), one can think of an initial cell measure $\aleph^{init}$ which is zero where the cell is missing and one otherwise.*

Example 14.1.4 (**Stable cell measures**). *Meinshausen and Bühlmann [2010] applied their Stability Selection to Graphical Lassos, i.e., with the* ***cell set*** $\hat{S}^{cell,\lambda} = \{(i,j) \mid (\hat{\Sigma}_{ij}^{-1})^{\lambda} \neq 0\}$ *where* $(\hat{\Sigma}^{-1})^{\lambda}$ *denotes the estimated precision matrix provided by the Graphical Lasso with regularization parameter* λ*. Clearly, averaging the corresponding cell measures w.r.t. each subsample leads to a cell measure which can take more values in* $[0,1]$ *than* $\{0,1\}$ *and the Stability Selection provides a stable* $\{0,1\}-$*valued cell measure* $\hat{\aleph}^{stab}$ *and a corresponding stable empirical cell set* $\hat{S}^{cell,stab}$.

Until here, the concept of cell measures seems to be just a trivial extension of the former concepts without any relevance, specifically the third example is nothing but the description of a missingness pattern. But in fact, the cell measure framework in the context of covariance or precision matrix estimation is closely connected to the RCM framework and can be thought of a second stage.

Let us take a closer look on the examples, not counting the third one. Clearly, the cell measures depend on the empirical row and column measures, so we can identify the (empirical) cell measures as being **induced by RCM pairs**, i.e.,

$$\hat{\aleph} = \hat{\aleph}(\hat{\zeta}, \hat{\nu}) \text{ i.e., } \hat{\aleph}_{(i,j)} = \hat{\aleph}_{(i,j)}(\hat{\zeta}_i, \hat{\nu}_j).$$

For the case that there is done variable selection in the first stage, leading to $\hat{s}$ selected variables, represented by the set $\hat{S}$, an estimated covariance or precision matrix on the reduced data would clearly be a matrix in $\mathbb{R}^{\hat{s}\times\hat{s}}$. We think that it would not be fair to also reduce the space $\{1,...,p\}^2$ to the subspace corresponding to the selected columns since it would not reflect the variable selection in the cell measures. In this case, we prefer to define the cell measure in the sense

$$\hat{\aleph}_{(i,j)} = I(\{i \in \hat{S}\} \cap \{j \in \hat{S}\})$$

with the corresponding cell set

$$\hat{S}^{cell} = \{(i,j) \mid i \in \hat{S} \wedge j \in \hat{S}\}.$$

One can interpret this in artificially setting all components of the matrix where the cell measure is zero to NA. Note the close connection to cell-wise outliers and structural missings, see definition 3.3.3 and chapter 17.

Remark 14.1.1. *The definition of a cell measure is not reasonable for a coefficient matrix* $\beth$ *as for example in multivariate regression as in the previous chapter. To rationally describe the pattern of such a coefficient matrix, one has to work with* ***column-specific column***

***measures**, i.e., for each response column $Y_{.,r}$, $r = 1, ..., k$, one has to compute an empirical column measure $\hat{\nu}^{(r)}$, especially in the case of seemingly unrelated regression (Zellner [1962]).*

Since the column measure $\hat{\nu}^{(r)}$ corresponds to the $r-th$ column of the coefficient matrix $\beth$ and since an overall empirical row measure $\hat{\zeta}$ would affect whole rows of $\beth$, cell measures for a coefficient matrix would be obsolete.

*For the case that we have cell-wise outliers in the response matrix which would lead to **column-specific row measures** and **column-specific column measures**, we refer to the following sections.*

Remark 14.1.2. *Note that sparse covariance or sparse precision matrix estimation is a concept related to the sparseness of cell measures and in general not to the sparseness of column measures!*

Remark 14.1.3 (Time-dependent cell measures). *In Monti et al. [2014], the SINGLE algorithm has been proposed and has been adapted in Monti et al. [2015] to the inclusion of adaptively chosen forgetting factors which can be seen as online variant of the Graphical Lasso, providing sparse estimates of the precision matrix in order to estimate brain connectivity structures for fMRI data.*

This is indeed a very interesting algorithm which perfectly fits into our framework. The forgetting factors, no matter if they are chosen adaptively or not, can be seen as an empirical row measure, more precisely, in each time step t, one proposes an empirical row measure $\hat{\zeta}_t$, cf. section 9.9. In the special case of using sliding windows, the corresponding empirical row measure is $\{0, 1\}-$valued which is not true for row measures resulting from the usage of forgetting factors or when using soft kernels as described in Monti et al. [2014].

Concerning cell measures, [Monti et al., 2015, Sec. 3.2] describe that in each time step t, there is a set of true connections in the graph corresponding to true non-zero cells of the precision matrix, i.e.,

$$S_t^{cell,0} = \{(i,j) \mid \Sigma_{i,j}^{-1} \neq 0\}$$

*for each $t = 1, ..., T$. This clearly corresponds to a **sequence** $(\aleph_t)_t$ **of true cell measures** $\aleph_t$. The application of the SINGLE procedure provides a sequence $(\hat{\aleph}_t)_t$ of empirical cell measures $\hat{\aleph}_t$ where each empirical cell measure is induced by the empirical row measure $\hat{\zeta}_t$, respectively.*

14.2 Cell-wise outliers and missings

If one had independent response columns, one easily could replace the column-specific univariate linear regression models by suitable robust counterparts. The same is true for missings, since a missing in a response column would usually result in deleting the respective row from the training set which can be done separately for each univariate regression model if the response matrix has non-structural missings.

But once the response columns are dependent, more sophisticated approaches have to be found, as the multivariate L_2-Boosting from Lutz and Bühlmann [2006b] which is still non-robust since the linear regression coefficients directly depend on the columns of the regressor matrix and the columns of the response matrix which potentially both contain outliers. Additionally, the estimated covariance matrix would need to be robustified.

Robust estimation strategies for covariance matrices include first fruitful approaches like the OGK estimator (Gnanadesikan and Kettenring [1972]) that reduces the robust covariance estimation to robust variance estimation combined with an orthogonalization step, the MVE estimator from Rousseeuw [1985] (first mentioned in Rousseeuw [1984]) where first the ellipsoid with minimal volume that covers at least the half of the data points is determined, so that the covariance of these data points is the final MVE estimator, or the MCD estimator (Rousseeuw [1984], Rousseeuw [1985]) which is the covariance of a subset of observations that also covers at least the half of the total observations such that the determinant of the covariance matrix is minimal.

However, these techniques are tailored to row-wise contamination (cf. definition 3.3.1). In the presence of potential cell-wise outliers (see definition 3.3.3), it would be more appropriate to apply recent methods like in Agostinelli et al. [2015] or Rousseeuw and Van Den Bossche [2018]. We describe the algorithms a little bit more detailed in the examples 14.2.2 and 14.2.3. Essentially, they rely on suitable filtering steps to detect cell-wise (and case-wise) outliers. The outliers are flagged as NA, creating missings which are imputed in the DDC procedure. For the estimation of covariance matrices based on the data containing NA's, Agostinelli et al. [2015] invoke a so-called GSE estimator.

Remark 14.2.1 (Generalized S-estimator). *Note that Danilov et al. [2012] provided a generalized S-estimator (GSE) which is based on partial Mahalanobis distances that are the Mahalanobis distances computed on the reduced data set using the corresponding mean vectors and reduced covariance matrices. For each observation i, they introduced a missingness vector u_i which has a zero where an NA appears and a one otherwise. Then, the score* *function on the left hand side of the S-equation, apart from the scale which has to be optimized and a standardization, depends on these partial Mahalanobis distances according to u_i.*

Evidently, no imputation is done anywhere, but nevertheless, Danilov et al. [2012] assume a MCAR missing scheme (see chapter 17). Additionally, Rousseeuw and Van Den Bossche [2018] pointed out that the GSE requires at least $n > p$ and Leung et al. [2017] mentioned that this estimator can no longer considered to be robust for $p > 10$.

Leung et al. [2017] replace this GSE estimator with a generalized Rocke-S-estimator to keep the robustness for larger p. Although this estimator and the DDC procedure (which can be a pre-processing step for covariance estimation as described in Rousseeuw and Van Den Bossche [2018]) are also applicable for high-dimensional data, the covariance estimators are not sparse in the sense that many entries are zero. Note that there already exist a lot of work on sparse estimation of precision matrices like the Graphical Lasso (see Banerjee et al. [2008] and Friedman et al. [2008]), CLIME (Cai et al. [2011]) or MissGLasso (Städler and Bühlmann [2012]) as well as sparse estimation techniques for covariance matrices, e.g. Bickel and Levina [2008]. See Fan et al. [2016] for an overview of robust or sparse estimation techniques for covariance and precision matrices. Also note that Öllerer and Croux [2015] proposed a robustification of the Graphical Lasso by replacing the classical covariance matrix in the target function by a robust counterpart which is computed cell-wisely by using $\text{Cov}(X, Y) = \text{sd}(X)\,\text{sd}(Y)\,\text{Corr}(X, Y)$ where each of the three factors is replaced by a robust counterpart.

When concerning missing imputation, response matrices lead to additional issues. As described in Bárcena Ruiz and Tusell Palmer [2002], imputing missings in multivariate responses is problematic once the imputation would be done separately for all cells. They recommend to perform row-wise imputations to address for the potential correlations of the response columns. Indeed, the DDC procedure of Rousseeuw and Van Den Bossche [2018] predicts each cell using the available cells in the same row that are not flagged as outliers and whose columns are sufficiently correlated with the respective column (see example 14.2.3 for more details). It would be interesting to see if their procedure also works well for response matrices whose correlation structure is inherited from both the correlation structure of the regressors X_i and of the errors E_i.

Concerning multivariate L_2-Boosting, a robustification cannot be done by just replacing the covariance matrix in (13.2.2) by a suitable robust counterpart. More precisely, the whole regression procedure has to be robustified. There are already approaches for robust regression with multivariate responses, for example robust estimation via FRB based on S-estimators

(Pison et al. [2003], Van Aelst and Willems [2005]) or by replacing the means resp. the scatter

$$\mu = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix}$$

by suitable robust counterparts (Rousseeuw et al. [2004], Hubert et al. [2008]). Maybe it is possible to combine the algorithms from Leung et al. [2017] or Rousseeuw and Van Den Bossche [2018] with this strategy in order to get a robust regression with multivariate responses even in the presence of cell-wise contamination and missings in both the regressor and the response matrix. Note that Leung et al. [2016] indeed proceeded in this manner with their "three-step regression", but only for univariate responses.

However, even if one would replace the cell-wise simple least squares fit in multivariate L_2-Boosting as in (equation 13.2.2) by a robust counterpart in the sense of Leung et al. [2016], we clearly would get problems concerning computational feasibility since the robust estimates had to be updated due to the updating step for the current residuals and since only one single cell would be updated in each iteration, so the number of required iterations would be very high. Maybe a one-step approach based on optimally-robust influence curves as we will discuss in section 18.2 or an FRB-type strategy will turn out to be promising in the future.

Remark 14.2.2. *It does not seem that there already exists a covariance/precision matrix estimation or multivariate regression method which provides sparsity, robustness against cell-wise contamination and missing handling simultaneously. In the context of covariance or precision matrix estimation, one has to check if one could just combine the covariance estimation of Rousseeuw and Van Den Bossche [2018] or Leung et al. [2016] with the Graphical Lasso. For regression, one could just apply variable selection algorithms after a pre-processing step like the DDC procedure from Rousseeuw and Van Den Bossche [2018]. But as multivariate regression would also require an analogous pre-processing step for the response matrix, it needs to be validated if the DDC procedure can also be applied to response matrices.*

Remark 14.2.3 (Structural missings in multivariate responses). *We cannot exclude the case that even a response matrix can have structural missings. We indeed think that this is a rather usual case for example in questionnaires. Assume that we have a questionnaire concerning medical data with the structural missings that we will discuss in chapter 17. But if there exist response columns that are only based on the predictors that may lead to structural missings, then the response for the respective column has indeed a structural missing in all rows where a structural missing in the regressor matrix appeared.*

Then, we are of course in the context of seemingly unrelated regression since each response column is based on different predictors. Structural missings is responses should never appear if we are not in this context, otherwise we would just have the information that the questionnaire was awfully designed.

Even more interesting for the relation of the cell measure framework and the RCM framework is therefore the robust estimation of covariance matrices.

Example 14.2.1 (**MCD/MVE**). *Both the MCD and the MVE covariance estimators try to find a subset H of h "clean" observations and compute the covariance of these data points. Therefore, we can identify these strategies with a procedure providing an empirical $\{0,1\}-$valued row measure in the first stage, followed by a standard covariance estimation in the second stage, providing a full cell measure on the subset H, i.e., $\hat{\aleph}_{(i,j)} = 1$ for all i, j.*

Astoundingly, the computation of the C-step (Rousseeuw and Van Driessen [1999]) for the MCD estimator can be ***identified with the RCM framework although there are no responses****! This is true since the original observations $X_i \in \mathbb{R}^p$ are taken as pseudo-responses while the estimated mean vector and covariance matrix based on the subset of cardinality h are treated as "fitted values". The C-step then iteratively chooses the h rows corresponding to the smallest Mahalanobis distances. In this sense,* ***the Mahalanobis distance can be seen as a loss function itself****. We will investigate this relation again in the next section.*

Example 14.2.2. *The filtering step in Agostinelli et al. [2015] can also be embedded into our framework. The detection of cell-wise outliers is performed column-wisely, i.e., for each fixed column j, the observations are robustly standardized. The vector of standardized values is approximated with a standard normal distribution, leading to an adaptive choice for the cutoff so that each observation is treated as cell-wise outlier if the absolute value of its standardized value is greater than the cutoff. In our language, they compute* ***column-specific empirical row measures*** *$\hat{\zeta}^{(j)}$ for each column $j = 1, ..., p$.*

In the second step, the GSE estimator that we already discussed (remark 14.2.1) is applied, based on the modified data set where cell-wise outliers have been replaced by NA's. The GSE is actually computed by an iterative procedure which provides weights, i.e., empirical row measures, in each iteration.

Example 14.2.3 (**DDC**). *The DDC procedure (Rousseeuw and Van Den Bossche [2018]) is even more interesting since it is actually based on a very sophisticated combination of different row and column measures.*

The first step of DDC includes a column-wise standardization and cutoffs based on a χ_1^2-*distribution to flag cells as outliers, providing column-specific empirical row measures* $\hat{\zeta}^{(j)}$ *in a similar manner as in Agostinelli et al. [2015].*

To overcome the issue that cell-wise outliers may not be detectable by investigating each column separately, an additional row-based detection step is performed. More precisely, they first compute a robust correlation for all variable pairs, based on the non-flagged components. Afterwards, a predicted value for each cell based on an intercept-free robust regression is performed in the sense that a cell is predicted by all cells in the same row that have not been flagged as cell-wise outliers (including the cell itself), but only from the columns which are sufficiently correlated with the column of the respective cell. In other words, the prediction step invokes ***variable selection by column-specific empirical column measures*** $\hat{\nu}^{(j)}$ *and also the column-specific empirical row measures* $\hat{\zeta}^{(j)}$ *(which indicated the NA's). After computing suitably robustly standardized residuals, all cells whose absolute value of the standardized residual is too large are additionally treated as cell-wise outliers and replaced by NA'S . At the end, all NA's are replaced by the predicted values whereas the other cells remain unchanged.*

To determine row-wise outliers, the distribution of the standardized residuals is approximated by a χ_1^2-*distribution which leads to a similar cutoff as in the first step. More precisely, they average the values of the distribution function of the* χ_1^2-*distribution at the residuals, separately for each row, and treat a row as row-wise outlier if this empirical mean is too large. This delivers finally an empirical* $\{0,1\}-$*valued row measure.*

14.3 Connecting robustness and sparsity?

The intention behind sparse model selection is to find a small fraction of predictors (columns of the data matrix) which already contain most of the relevant information to provide well-generalizing models that are less likely to include noise variables. The goal of outlier detection is to find a rather large fraction of observations (rows of the data matrix) that represent the "clean" data such that the deleted rows are most likely to be outliers.

In this section, we discuss an idea that the row selection performed by outlier detecting algorithms can be regarded as modified (L_2-)Boosting. We start with a classical outlier detection procedure (i.e., which is designed for row-wise contamination) proposed in Rocke and Woodruff [1996].

Example 14.3.1 (Classical outlier detection). *Given data $X_1, ..., X_n \in \mathbb{R}^{1 \times p}$, the first "phase" of the outlier detection procedure consists of determining robust estimates for the mean and the covariance. In the second "phase", after a suitable rescaling of the covariance estimator to get an estimator $\hat{\Sigma} \in \mathbb{R}^{p \times p}$ and after updating the estimator for the mean to get $\hat{\mu} \in \mathbb{R}^{1 \times p}$, one computes robust Mahalanobis distances*

$$L_i = \sqrt{(X_i - \hat{\mu})^T (\hat{\Sigma})^{-1} (X_i - \hat{\mu})}$$

and only keeps the observations whose distance does not exceed some cutoff.

In fact, the distances can be thought of an (robust) L_2-loss of the same type as (13.2.1) since we indeed compute suitable differences between the rows X_i and the "row" $\hat{\mu}$. We can rewrite the problem such that the similarities to Boosting and variable selection are easier to see. Let $\check{\mathcal{D}} \in \mathbb{R}^{p \times 2n}$ be the model matrix

$$\left(\begin{array}{ccc|ccc} X_{1,1} & \dots & X_{n,1} & X_{1,1} & \dots & X_{n,1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ X_{1,p} & \dots & X_{n,p} & X_{1,p} & \dots & X_{n,p} \end{array} \right) = (X_1^T, ..., X_n^T | X_1^T, ..., X_n^T) = (X^T | X^T). \quad (14.3.1)$$

This can be interpreted as to explain the rows X_i using the rows X_i. Since this would just imply the optimal solution for a coefficient matrix $\beth$ in a linear model to be the identity matrix which addressed neither sparsity nor robustness, one replaces the $(p \times n)-$dimensional "response matrix" by n copies of the $(p \times 1)-$dimensional initial "residual" $r^{(0)}$ which admittedly abuses the notation of Boosting. However, we can think of this matrix as residual matrix $R \in \mathbb{R}^{p \times n}$. This "residual" $r^{(0)}$ is a robust estimator for the mean. Nevertheless, if we use the robust estimate $\hat{\mu}$ as residual vector, we get the updated model matrix

$$(X_1^T, ..., X_n^T | r^{(0)}, ..., r^{(0)}) = (X^T | R) \quad (14.3.2)$$

*so that the task is now to explain each residual vector by the corresponding regressor column $X_i = X_{\cdot,i}^T$. The outlier detection procedure of Rocke and Woodruff [1996] is the special case where one computes **a robust** L_2-**loss separately for each column**, i.e., we have*

$$L_i = L(X_i, r^{(0)}) = \sqrt{(X_i - r^{(0)})^T (\hat{\Sigma})^{-1} (X_i - r^{(0)})},$$

*but since $\sqrt{\cdot}$ is strictly increasing on the positive halfaxis and since we are only interested in the **ordering of the losses**, we can also use the multivariate L_2-loss (13.2.1) proposed by Bühlmann with $\Gamma = \hat{\Sigma}$ to compute the losses. The last step can be treated as **rejection step** in the sense that we only select the columns whose loss does not exceed some threshold. In* *the language of Boosting, this would correspond to a* ***generalized component-wise base procedure*** *where not only the best column is selected in each iteration but in general the "best majority" of columns according to the multivariate L_2-loss (or equivalently, to the Mahalanobis distance). Although the outlier detection does not need coefficients, we can think of getting* ***rows containing only zeroes in the coefficient matrix where a column is not selected****, so the artificial coefficient matrix $\gimel \in \mathbb{R}^{n \times n}$ has rows containing only zeroes and all other rows contain also only zeroes except for the cell corresponding to the diagonal which takes the value one.*

Summarizing, the outlier detection in Rocke and Woodruff [1996] may be seen as multivariate L_2-Boosting with only one Boosting iteration and without reasonable coefficients where the robust mean serves as the residual in the first and only iteration.

Remark 14.3.1. *Other classical outlier detection algorithms which are also capable to handle high dimensions like the PCDIST algorithm from Shieh and Hung [2009] may include additional steps, in this case a dimension reduction via principal component analysis, but the rationale behind it essentially stays the same as in the procedure we discussed above.*

Although it is not exactly L_2-Boosting, it is surprising that we can rewrite a robust (row selection) procedure in a way that we are basically in the same setting as usual with mutlivariate responses. Even more astoundingly, Rousseeuw's Fast-MCD estimator can indeed also be treated as a special L_2-Boosting!

Example 14.3.2 (Fast-MCD estimator). *We propose the same initial model matrix $\check{\mathcal{D}}$ as in the previous example (equation (14.3.1)). The construction of the initial subset H_0 of size h (see for example Hubert and Debruyne [2010]) can be seen as offset procedure to get initial "residuals". Since the robust estimates in each iteration are in truth the classical mean and covariance estimators based on the current subset, we indeed have* ***coefficients****. More precisely, if the initial subset H_0 of size h is chosen, one implicitly has the coefficient matrix $\hat{\gimel}^{(0)} \in \mathbb{R}^{n \times n}$ that only contains zero rows and rows where each entry is h^{-1}. The zero rows are exactly those rows whose row indices are not contained the set H_0.*

Starting from this initialization, we compute the current "residual" $r^{(0)} \in \mathbb{R}^p$ which is the mean of the X_i with $i \in H_0$, i.e.,

$$X^T \hat{\gimel}^{(0)} = \begin{pmatrix} \bar{X}_{H_0,1} & \dots & \bar{X}_{H_0,1} \\ \vdots & \ddots & \vdots \\ \bar{X}_{H_0,p} & \dots & \bar{X}_{H_0,p} \end{pmatrix} =: (r^{(0)}, ..., r^{(0)}) \tag{14.3.3}$$

where $\bar{X}_{H_0,j}$ is the mean of the $j-$th component of the X_i with i corresponding to the set H_0.

Additionally, we compute the covariance matrix $\hat{\Sigma}^{(0)} \in \mathbb{R}^{p \times p}$ based on H_0. Then, as in the previous example, we compute the losses

$$L_i = L(X_i, r^{(0)}) = \sqrt{(X_i - r^{(0)})^T (\hat{\Sigma}^{(0)})^{-1} (X_i - r^{(0)})},$$

leading to a new set H_1 consisting of the h indices corresponding to the h smallest losses. The set H_1 directly leads to a "base model" $\hat{\daleth}$ which consists of zero rows and rows where each entry is h^{-1} and where the latter type of rows exactly correspond to the indices contained in H_1. The coefficient update may be thought of being a special case of a generalized update scheme of the form

$$\hat{\beth}^{(1)} = (1 - \chi)\hat{\beth}^{(0)} + \kappa \hat{\daleth} \tag{14.3.4}$$

where κ is a learning rate and χ is a ***forgetting factor*** *in the sense that, in contrast to Boosting, the current "strong model" does not get weight 1. Clearly, for the Fast-MCD estimator, we have to set $\kappa = 1 = \chi$.*

Converting this prosaic description into an algorithm, we propose the following pseudocode for the Fast-MCD algorithm in Boosting form:

```
Initialization: Observations $X_1, ..., X_n \in \mathbb{R}^p$, cardinality $h$ of best subsets, learning
 rate $\kappa = 1$ and forgetting factor $\chi = 1$;
Compute an initial set $H_0$ of size $h$;
Compute the mean $r^{(0)}$ and the covariance $\hat{\Sigma}^{(0)}$ based on $H_0$;
Compute $\det(\hat{\Sigma}^{(0)})$;
Get the initial coefficient matrix $\hat{\beth}^{(0)}$;
Set $k = 0$;
while not converged do
    Set $k = k + 1$;
    Compute the $L_2$-losses $L_i$ based on $X_1, ..., X_n$, $r^{(k-1)}$ and $\hat{\Sigma}^{(k-1)}$;
    Compute $H_k$ containing the indices corresponding to the $h$ lowest losses;
    Get the base model $\hat{\daleth}$ corresponding to $H_k$;
    Compute $r^{(k)}$ and $\hat{\Sigma}^{(k)}$;
    Compute $\det(\hat{\Sigma}^{(k)})$;
    Update the coefficients via
        $\hat{\beth}^{(k)} = (1 - \chi)\hat{\beth}^{(k-1)} + \kappa \hat{\daleth}$.
end
```

Algorithm 17: Fast-MCD in L_2−Boosting form

Convergence occurs once the determinant of the current covariance matrix is the same as the determinant of the previous one. Of course, we note that one usually runs the whole algorithm *several times with different initial subsets to get a chance to reach a global minimum. This can again be seen as some kind of* ***rejection step*** *itself, only accepting the subset of cardinality h delivered by the different runs of the algorithm which leads to the minimal covariance determinant.*

The LTS estimator which can also be computed using concentration steps (see Rousseeuw and Van Driessen [2000]) can be rewritten as a generalized L_2-Boosting algorithm as well which we do not spell out here. We rather show one more example where sparsity and robustness are already connected. This is again the already mentioned SLTS algorithm (see algorithm 13).

Example 14.3.3 (SLTS). *Two major differences between the Fast-MCD algorithm and the Fast-SLTS algorithm are that the latter includes response variables which remain stationary throughout the whole process and that the target loss criterion changes from a multivariate L_2-loss to*

$$L(Y_i, \hat{Y}_i) = (Y_i - X_i\hat{\beta})^2 + \lambda \sum_j |\hat{\beta}_j|.$$

```
Initialization: Data $\mathcal{D} = (X, Y) \in \mathbb{R}^{n \times (p+1)}$, cardinality $h$ of best subsets, learning
 rate $\kappa$ and forgetting factor $\chi$;
Compute an initial subset $H_0$;
Compute the Lasso estimator $\hat{\beta}$ based on $H_0$;
Set $k = 0$;
while not converged do
    Set $k = k + 1$;
    Compute the usual squared losses for each of the $n$ response columns separately;
    Compute $H_k$ containing the indices corresponding to the $h$ smallest squared losses;
    Get the base model $\hat{\daleth}$ by applying the Lasso on the set $H_k$;
    Update the coefficients via
        $\hat{\beth}^{(k)} = (1 - \chi)\hat{\beth}^{(k-1)} + \kappa\hat{\daleth}.$
end
```
Algorithm 18: SLTS in L_2-Boosting form

In each iteration, one computes a Lasso model based on the current subset of h rows. The new subset consists of the indices of the h rows corresponding to the minimal unregularized squared losses. This can indeed again be artificially understood as a multivariate Boosting where on uses the model matrix

$$\left(\begin{array}{ccc|ccc} X_{1,1} & \dots & X_{n,1} & Y_1 & \dots & Y_n \\ \vdots & \ddots & \dots & \vdots & \dots & \vdots \\ X_{1,p} & \dots & X_{n,p} & Y_1 & \dots & Y_n \end{array}\right)$$

so that we again compute the losses for each column separately, but with the difference that our model is "transposed" in the sense that we compute $\daleth X^T$ for $\daleth \in \mathbb{R}^{n\times p}$. In Boosting language, the pseudocode is given in algorithm 18.

Example 14.3.4 (SLTS again). *The last example may look too artificial. However, we can also use the usual data matrix $\mathcal{D} = (X, Y)$ and write down the SLTS algorithm in the following Boosting-type form:*

```
Initialization: Data $\mathcal{D} = (X,Y) \in \mathbb{R}^{n\times(p+1)}$, cardinality $h$ of best subsets, learning
 rate $\kappa$ and forgetting factor $\chi$;
Compute an initial subset $H_0$;
Compute the Lasso estimator $\hat{f}^{(0)}$ based on $H_0$;
Set $k = 0$;
while not converged do
    Set $k = k + 1$;
    Compute the usual squared losses;
    Compute $H_k$ containing the indices corresponding to the $h$ smallest residuals;
    Get the base model $\hat{g}^{(k)}$ by applying the Lasso on the set $H_k$;
    Update the coefficients via
        $\hat{f}^{(k)} = (1-\chi)\hat{f}^{(k-1)} + \kappa\hat{g}^{(k)}$.
end
```

Algorithm 19: SLTS in L_2-Boosting form, second version

Note again that we never replace the response vector Y in SLTS.

Remark 14.3.2 (Robustness and sparsity). *One could ask why we stated two Boosting-type versions which essentially just rewrite the SLTS algorithm. Both versions motivate a **generalized Boosting algorithm that combines both sparse model selection and robustness in terms of "clean" subsets of observations**.*

The first extension of standard L_2-Boosting is motivated by algorithm 18 which artificially treats the regressor columns as rows and vice versa. Although this rewriting does not change the SLTS procedure which still automatically selects predictors (i.e., rows) by the Lasso and selects observations (i.e., columns) by the concentration step, one may think of extending L_2-Boosting in the sense that not only the best variable is selected in each iteration but the best l variables for some $l \geq 1$ or even depending on the iteration by taking the l_k best ones *in iteration k.*

The second extension is motivated by both algorithms 18 and 19 concerning the updating step. In standard L_2-Boosting, it is not possible to forget coefficients computed in earlier iterations. This is related to the statement of Zhao and Yu [2007] who pointed out that forward selection procedures like Boosting cannot correct mistakes which they already made. Writing down SLTS as Boosting-type algorithms, we used a forgetting factor since the coefficients computed in any but the last iteration are deleted afterwards.

The third and most important extension is motivated by algorithm 19 which performs additional concentration steps within the Boosting algorithm. Indeed, this algorithm can be seen as ***special case of a generalized Boosting algorithm*** *in the sense that it deletes all information which has been derived in previous steps which needs not to be the true for a Generalized Boosting.*

Summing up, our proposal for a ***Generalized*** *L_2-***Boosting** *would be the following algorithm.*

```
Initialization: Data $(X,Y)$, step sizes $\kappa_k \in ]0,1]$, forgetting factors $\chi_k \in [0,1]$,
  number $m_{iter}$ of iterations, cardinality $h$ of best subsets, numbers $p_k$ of chosen
  predictors;
Compute an offset $\hat{f}^{(0)}$;
Compute the current residual vector $r^{(0)}$;
Compute an initial subset $H_0$;
for $k = 1, ..., m_{iter}$ do
    Compute a generalized baselearner $\hat{g}$ based on subset $H_{k-1}$ that selects the best $p_k$
      predictors;
    Compute the gradient vector $r^{(k)}$;
    Update the model via
        $\hat{f}^{(k)} = (1-\chi_k)\hat{f}^{(k-1)} + \kappa_k \hat{g}$
    Compute the squared losses w.r.t. model $\hat{f}^{(k)}$;
    Compute the subset $H_k$ corresponding to the smallest $h$ residuals;
end
```

Algorithm 20: Generalized L_2-Boosting

Remark 14.3.3. *Clearly, this generalized L_2-Boosting algorithm leads to a bunch of questions, for example concerning the choice of the p_k, the generalized base-procedure or the choice of the forgetting factors. As for the base procedures, one may either take standard base procedures like simple least squares and select the best p_k predictors. On the other hand, SLTS* *uses Lasso baselearners which seems to be reasonable as well in this context.*

The main question that arises seems to be the paradigm which computation has to be performed when. We suggested to update the model first and to determine which rows belong to the new clean subset by the resulting losses w.r.t. the combined model. Note the similarity to the updating step in SingBoost for losses like the hard ranking loss (see section 10.4). It could also be possible to use the losses of the baselearner to make this decision, but we think that this gets unreliable, especially in later iterations where the correlation of the predictors to the current residual vector gets rather low.

14.4 Stability Selection for rows

In this thesis, we mainly concentrated on Stability Selection for the columns and mentioned that there also exists a Stability Selection for cells (see example 14.1.4). But to the best of our knowledge, there does not yet seem to be some kind of Stability Selection for the rows.

We start with the following motivating example.

Example 14.4.1 (**Stabilized SLTS?**). *When concerning about stable predictor selection and SLTS, it would be straightforward to apply Stability Selection to SLTS by drawing subsamples and performing SLTS on each subsample and by aggregating the selection frequencies.*

However, SLTS in its current form loses information since it does never account for the variables that already have been chosen by the Lasso in the previous steps, i.e., for the ***iteration-specific empirical column measures w.r.t.*** L_2 ***provided by the Lasso****. Therefore, a simple modification where the selection frequencies are saved in each iteration of SLTS may lead to a better final predictor selection. However, note that for the Fast-LTS algorithm, 10 iterations are usually sufficient, see Rousseeuw and Van Driessen [2006], and Alfons et al. [2013] only perform two C-steps (as also suggested in Rousseeuw and Van Driessen [2006]) for the most initial subsets and proceed with further C-steps only on the best 10 resulting models, so this procedure would not lead to enough evidence for column selection. But in fact, since the algorithm converges to a local minimum, one needs a large number of different initial subsets which is 500 in Alfons et al. [2013]. Therefore, we can simply* ***average the selection frequencies over all SLTS runs and iterations****. In other words, the* *aggregation of the empirical column measures would be nothing but a* ***Stability Selection*** *since one essentially has B subsamples with h observations on which the algorithm has been applied. This can be regarded as an extension of the Stability Selection for Lasso models introduced in Meinshausen and Bühlmann [2010] in the sense that more than one Lasso model is applied to each subsample in an iterative manner.*

This directly leads to the question how the final model can be computed. At the end of the whole SLTS procedure (i.e., where we have as many models as different initial subsets), one takes the model which led to the lowest value of the objective function. Therefore, it is not clear how one could get a final stable model. If one takes the set H^* with h observations corresponding to the best SLTS model, one may perform standard least squares based on the subset $\hat{S}^{stab}$ of stable predictors and on the subset H^* of observations, but it is not evident if the restriction of the columns to $\hat{S}^{stab}$ would lead to subset H^* if one ran SLTS on the reduced data.

But so far, we only concerned about stable column selection. We indeed have more information, namely an aggregated empirical row measure which we can compute by averaging the selection frequencies of the rows. One approach would be to average the selection frequencies respecting each single SLTS iteration as we proceeded when computing the aggregated column measure. However, in this case, it seems more reasonable to average the selection frequencies based on the final SLTS subsets corresponding to the (in the case of Alfons et al. [2013], the best 10) models since the objective function, here the regularized squared loss, decreases in each iteration by the property of the C-step, so one can definitely state that the final subset is the best of all, making it useless to incorporate the former ones.

This leads to the new concept of ***Stability Selection for rows*** *which means that we only keep the rows in the final subset whose aggregated selection frequencies exceed some threshold. Maybe in the special case of SLTS which is based on subsets of cardinality h, it would suffice to take the h rows with the highest aggregated importances. Then, the final stable SLTS estimator would be weighted least squares on the selected final rows which takes the aggregated row measure into account.*

Example 14.4.2. *The Generalized L_2-Boosting algorithm (see algorithm 20) would indeed be a perfect candidate for a Stability Selection both for the rows and for the columns. This would answer the question how the final model for the Generalized L_2-Boosting algorithm would look like, i.e., it would also be a weighted least squares model based on the stable rows and the stable columns.*

We believe that such a Stability Selection for rows may turn out to be a new milestone in machine learning. Historically, one began to perform a brute-force best subset selection when trying to select the best predictors in a data set. Up to this point, it has been the best opportunity due to the lack of an ordering of the predictor columns. Concerning backward and forward step-wise selection, one began some kind of local ordering by deleting the variable with the largest $p-$value or by including the variable with the highest correlation with the response. The breakthrough in model selection has been achieved with algorithms like the Lasso that automatically perform sparse model selection and like Boosting which combine baselearners and end up with sparse models. The Stability Selection (for the columns) used the ordering of the columns in terms of selection frequencies generated by Lasso or Boosting models computed on different subsamples of the data.

A Stability Selection for cells when estimating a sparse covariance matrix follows essentially the same paradigms: Repeatedly apply an algorithm which computes a sparse covariance matrix (and therefore, a sparse empirical cell measure $\hat{\aleph}$) and use the resulting ordering in terms of selection frequencies corresponding to the cells to select a stable cell set.

When concerning a row selection, why don't we proceed in a similar manner? Obviously, we do not have an ordering of the rows initially and the most learning algorithms do not help in defining one. When pre-defining a number h of observations that should be kept, a "best subset selection" for the rows would require to take all subsets of cardinality h and to take the subset where the objective criterion is minimal. For the MCD estimator, Rousseeuw already solved this problem in a very clever way by invoking the concentration steps. However, these concentration steps themselves require an ordering of the rows which is implied by the ordering of the Mahalanobis distances w.r.t. the mean and the covariance matrix of the current subset. In other words, an analog to the multivariate L_2-loss is used, so the algorithm is essentially L_2-based.

Therefore, our proposal for a **Stability Selection for rows** is as follows. Having an ordering of the rows based on different subsamples of the data, e.g., by having computed a row-selecting learning algorithm on each subsample or by suitable concentration steps as in Generalized Boosting (cf. algorithm 20, analogously for other losses than the squared loss) or even by a loss-based aggregation of the final models computed on each subsample in the spirit of the loss-based aggregation for CMB (cf. equation (11.4.3)), we **select the h rows whose aggregated selection frequencies are the highest**. Another possibility is to invoke thresholds and proceed as in the usual Stability Selection for columns or cells, basically just by replacing the aggregated empirical column or cell measure by the aggregated empirical row measure.

When concerning a Stability Selection for rows, one has to keep in mind that in contrast to column selection which should be sparse, row selection should not be sparse since that would delete too much information. Therefore, natural questions that may be answered in future works are for example how a suitable threshold (or the analog h to our q) may be chosen. The main purpose of a Stability Selection for rows is that one wants to keep the most ”clean” observations, deleting or downweighting observations that appear to be contaminated.

Remark 14.4.1 (Elbow plot). *When performing principal component analysis or clustering, one usually draws so-called scree plots or elbow plots to choose the number of principal components or the number of clusters (as for the latter, see for example [Friedman et al., 2001, Sec. 14.3.11]).*

A simple graphical approach for robust algorithms to determine the number h of rows in a Stability Selection would be to apply the already computed stable model (based on a stable column set) and to calculate the losses for each row. Since the models are assumed to be robust, rows containing outliers would be expected to correspond to higher losses than the ”clean” rows, so the elbow in a plot that maps the ordered losses (ascending) w.r.t. h (running from some $m < n$ to n) could help to choose the final number of rows.

Remark 14.4.2. *It can sound confusing to select rows since any empirical row measure is obviously tailored to the training set and gets meaningless once new observations appear. However, the Stability Selection for the rows follows the same paradigm as the one for the columns in the sense that it is expected to improve the out-of-sample performance of the model. While the Stability Selection for the columns is a variable selection procedure, the Stability Selection for the rows improves the coefficients to avoid the case that outliers lead to meaningless coefficients.*

Remark 14.4.3 (Masking and swamping). *One may ask if it would suffice to apply some algorithm like a Generalized Boosting with a concentration step to determine the empirical row measure that is used for a Stability Selection for the rows. However, in cases where the underlying algorithm is not some sophisticated robust outlier-detecting algorithm that can handle situations with multiple outliers, one indeed needs subsamples due to masking and swamping effects of multiple outliers (see e.g. Rousseeuw and Hubert [2011]). Roughly speaking, a swamping effect of outliers lets observations which are not outliers appear to be outliers. In contrast, the masking effect lets outliers appear not to be outliers due to other outliers. For example, a vast outlier can cause a masking effect on other outliers (see [Maronna et al., 2006, Ex. 1.2]). By subsampling, we ge a chance to split up these structures in order to circumvent swamping and masking effects.*

Remark 14.4.4 (Stability is not stability!). *So far, we defined stable column measures as column measures with the property that their variability is low w.r.t. other observations from the same distribution. The same is true for stable cell measures. At the first glance, the* *definition of a stable row measure seems to be inappropriate.*

However, throughout section 14.3, we already pointed out that outlyingness is essentially determined by some loss based on either some robust mean as pseudo-response or on the true responses where the "prediction" depends on several columns of the regressor matrix. Indeed, since row measures are induced by column measures, stable row measures need to be defined as ***row measures whose variability is low w.r.t. different stable column measures****. Assume that there is a cell-wise outlier in a particular row . But if the corresponding variable is not relevant, there would be no need to downweight or even to delete this row since the outlier is ignored in model-fitting when using a sophisticated robust learning algorithm. The fact that column measures depend on the rows themselves lets this strategy appear to be unrealiable since the outlier may already have affected the column selection. But this is not true for stable column measures where the subsampling procedure already (partially) guards against contamination. We are always assuming that a significant majority of rows reduced to the columns corresponding to the stable variables do not contain outliers.*

Finally, let us formally define the stable row set.

Definition 14.4.1. *Let $\mathcal{D} \in \mathbb{R}^{n \times (p+1)}$ be a data matrix and let $\hat{\zeta}$ be some aggregated row measure. Then the* ***stable row set*** *is defined as*

$$\hat{S}^{row,stab} := \{i \mid \hat{\zeta}_i \geq \pi_{thr}\}$$

for some threshold π_{thr}.

Remark 14.4.5. *Note that when using a pre-defined number h as analog to our q in the context of Stability Selection for columns, we implicitely define a threshold which can be defined as the entry of the empirical aggregated row measure corresponding to the $h-$th most relevant row.*

Remark 14.4.6. *Note that once a Stability Selection for both the rows and the columns has been performed, one finally gets* ***stable empirical RCM pairs*** *$(\hat{\zeta}^{stab}, \hat{\nu}^{stab})$ where $\hat{\zeta}^{stab}$ is the 0/1-row measure corresponding to $\hat{S}^{row,stab}$ and the stable column measure as before, e.g., algorithm 5.*

Remark 14.4.7 (Ranking rows). *The goal of the ranking problems from part II was to (partially) recover the true ordering of the instances (X_i, Y_i) or basically the X_i which is induced by the ordering of the responses Y_i in the case of supervised learning. The ranking of rows that is done as pre-processing step for the Stability Selection for rows is motivated from another paradigm and has (in its raw form) nothing to do with the problem to rank* *instances. However, a combination seems to be both possible and reasonable if the data are contaminated. Then the (predicted) ranking itself would also be prone to be misspecified due to the contamination, i.e., a stable row set should be computed first.*

The stable empirical RCM pairs from the previous section still have a weakness. Let us first introduce the following definition.

Definition 14.4.2. *Let $\mathcal{D} \in \mathbb{R}^{n \times (p+1)}$ be a data matrix. Then we define the corresponding **RCM matrix** as*

$$\gimel = (\gimel_{ij})_{i=1,...,n,j=1,...,p} = ((\zeta_i, \nu_j))_{i,j}$$

for a row measure ζ and a column measure ν.

The weakness of stable empirical RCM pairs is that once one performed variable selection, leading to a stable empirical column measure $\hat{\nu}^{stab}$ and row selection which delivers a stable empirical row measure $\hat{\zeta}^{stab}$, the non-zero entries of the stable empirical RCM matrix $\hat{\gimel}^{stab}$ are represented by the submatrix corresponding to the selected rows resp. columns. One can think of $\gimel$ as an $(n \times p \times 2)-$dimensional array where both entries of $\gimel_{i,j,\cdot}$ are zero if one of them is zero. Transferring this to the computation of the final model, one just applies the respective learning algorithm to the corresponding subset of $\mathcal{D}$ by deleting the non-selected rows and columns.

Inspired by the column-specific row and column measures that arise in the DDC procedure (Rousseeuw and Van Den Bossche [2018], see section 14.3), it would be an interesting topic for future research to develop a cell-wise procedure that allows for arbitrary structures of the RCM matrix which results in working with heterogeneous data where the rows depend on different columns (or vice versa), so that NA's appear in the modified data matrix. For suggestions how one could proceed when facing missings, we refer to chapter 17.

Remark 14.4.8 (Ranking is not ranking!). *We essentially have faced seven different types of ranking in this work:*

1.) *The ranking of columns which we encounter in each Boosting iteration when taking the column whose baselearner improves the model most w.r.t. a given loss criterion;*

2.) *The ranking of columns again in the column measure framework when columns are ranking according to their importance (in contrast to the loss-based ranking in 1.)) which becomes* *relevant especially in Stability Selection where only the most important part of the columns is chosen;*

3.) *The ranking problem that motivated our work which is nothing but a ranking of rows induced by the ranking of the responses (in supervised learning);*

4.) *The ranking of rows for some kind of Stability Selection for rows depending on relative selection frequencies;*

5.) *The ranking of rows based on losses or distances, in particular in concentration steps and when concerning about robustness;*

6.) *The ranking of whole models which is done when selecting the best part of the models computed in CMB and when choosing the best element of the respective grid in CMB-3S. This is basically a within-ranking since we compare models based on the same algorithm, but with other data resp. other predictor sets. A cross-ranking is done once one compares for instance CMB-3S models with competitors, e.g., L_2-Boosting models with Stability Selection. Note the relation to elicitability;*

7.) *The ranking of cells when concerning about Stability Selection for the cells of estimated covariance matrices.*

14.5 Outlook

However, going deeper into detail how MultiSingBoost has to be constructed, how a Stability Selection for rows can be performed, how the RCM framework can be reasonably extended to the case of multivariate responses and how similar frameworks can be defined for other structures definitely goes beyond the scope of this work.

It would be a very interesting and promising topic for future work to get answers to the following questions:

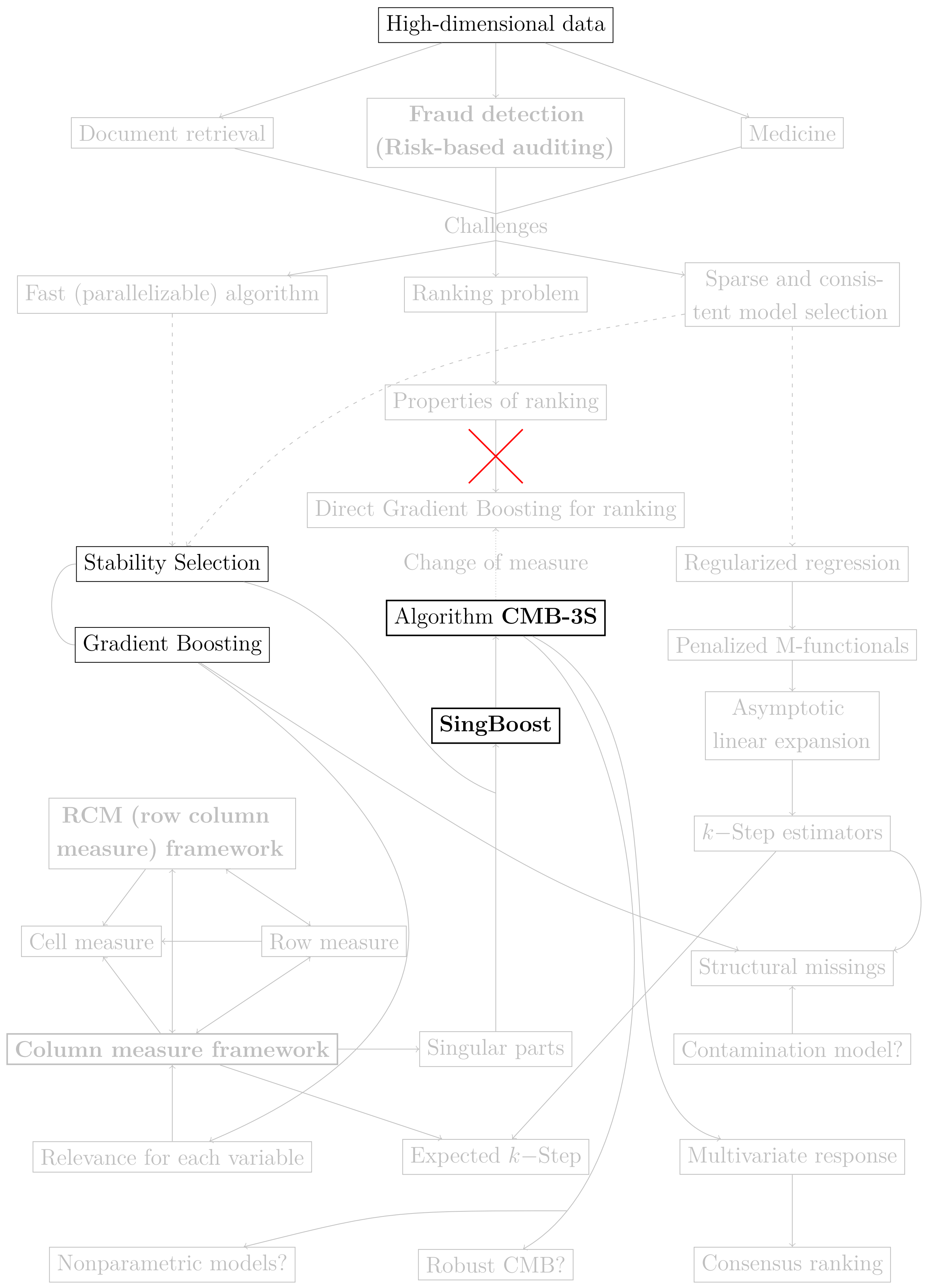
How can we extend the RCM framework to the case of multivariate responses?
Does the RCM framework help to find better algorithms when facing a highly non-regular loss function in the context of multivariate responses?
How do we account for cell-wise outliers in multivariate responses?

Can response-column-specific column measures be correlated themselves and how can we quantify this?
Do highly correlated response columns provide similar column measures?
How can we perform a Stability Selection in the case of multivariate responses?

If it was possible to get the answers, one would have a tool to unify all peculiarities like response-column-specific variable selection, case-wise and cell-wise outliers both in the variables as well as in the responses, structural missings (in both the regressor and the response matrix!) and block-wise structures into one single framework.

Part VI

Numerical demonstration


High-dimensional data
Document retrieval
Fraud detection (Risk-based auditing)
Medicine
Challenges
Fast (parallelizable) algorithm
Ranking problem
Sparse and consistent model selection
Properties of ranking
Direct Gradient Boosting for ranking
Stability Selection
Change of measure
Regularized regression
Algorithm CMB-3S
Gradient Boosting
Penalized M-functionals
SingBoost
Asymptotic linear expansion
RCM (row column measure) framework
k−Step estimators
Cell measure
Row measure
Structural missings
Column measure framework
Singular parts
Contamination model?
Relevance for each variable
Expected k−Step
Multivariate response
Nonparametric models?
Robust CMB?
Consensus ranking

This part is devoted to the implementation of the algorithms SingBoost, CMB, CMB-3S and CV.CMB-3S that have been developed during the creation of this thesis.

The first chapter demonstrates the application to real data sets. We first consider small data sets, i.e., the well-known `iris` data set and the `bodyfat` data set from the package `TH.data`. This allows us to go into detail and to show many features of our implementation. Then, we provide an example for the application to a large real data set, i.e., the `Real.2` data set from the package `PRIMsrc`. Due to the computational costs, the demonstrated features are far less here. We close the chapter with a short application to the very high-dimensional data set `riboflavin` from the package `hdi`.

It is important to note that the first chapter should just show how the algorithm works and how a user can handle it, but we do not regard any form of sorting different strategies by their performance.

In the second chapter, we indeed compare our proposed algorithms with L_2-Boosting and Hofner's Stability Selection. We will see that our CMB-3S algorithm can beat L_2-Boosting in the majority of cases when considering the hard ranking loss function after applying it to high-dimensional simulated data.

We will see that our loss-based Stability Selection outperforms Hofners's Stability Selection when facing high-dimensional noisy data. We further show that our Stability Selection can easily handle even very high-dimensional data and leads to sophisticated stable models that significantly outperform models computed by simple Boosting algorithms.

Chapter 15

Application to real data sets

In this chapter, we apply the `CV.CMB3S` function to the rather small `iris` and the `bodyfat` data set as well as to the larger `Real.2` and the `riboflavin` data set.

Since the first both data sets are very small, we can demonstrate a lot of the functionalities of this algorithm and its sub-functions. As for the large data sets, we restrict ourselves to some basic features of our algorithm due to the high computational time.

15.1 Application to the `iris` data set

The well-known `iris` data set consists of $n = 150$ observations with $p = 5$ variables where one variable is categorical with three classes, the others are metric.

15.1.1 `singboost`

In the first example, we also show the output of `glmboost`. Due to the formula argument that we inserted, this function applies L_2-Boosting (default) with the variable `Sepal.Length` as response variable with $m_{iter} = 100$ and $\kappa = 0.1$ (defaults). As output, we first get a short summary of the main input components. The reported offset value is nothing but the mean of `Sepal.Length` since the mean is the minimizer of the empirical squared error (see the definition of the offset in algorithm 23). At the bottom, the fitted non-zero coefficients and again the offset are displayed. The selection frequencies are not automatically returned and need to be called separately.

As for `singboost`, we always require that the last column of the inserted data set is the response column. We also require that the regressor matrix is already a model matrix, so that we do not need a formula argument. To use `singboost`, we first have to suitably apply the command `model.matrix` and to insert the corresponding data matrix as input argument. `glmboost` can also be used by providing a model matrix in a similar way where the response column must be entered separately. Note that we automatically create the intercept column in `singboost` so that we delete the intercept column created by `model.matrix` which is needed for `glmboost`. In the following experiment, we just use the default `singfamily=Gaussian()`, so SingBoost is nothing but standard L_2-Boosting.

As we see, the results are the same. We highlight once more that we do not compute an offset value, so we must compare our intercept with the sum of the computed intercept and the offset value reported by `glmboost`.

The fourth list element of our output is maybe the most interesting one for a user while the second list element may not be needed for visualization but for further computations in `cmb` or `CMB3S`. The selection frequencies are automatically displayed when using `singboost` and need to be called with an extra command line for `glmboost`.

Note that using `LS=T` does not change anything since we just compute the baselearners using `lm` instead of using `glmboost` which results in the same model for `singfamily=Gaussian()`.

```
glmres←glmboost(Sepal.Length~., iris)
glmres
```

```

         Generalized Linear Models Fitted via Gradient Boosting

Call:
glmboost.formula(formula = Sepal.Length ~ ., data = iris)


         Squared Error (Regression)

Loss function: (y - f)^2


Number of boosting iterations: mstop = 100
Step size:  0.1
Offset:  5.843333

```

```
Coefficients:
     (Intercept)       Sepal.Width      Petal.Length  Speciesvirginica
     -3.36560001        0.53639026        0.46090783       -0.01924627
attr(,"offset")
[1] 5.843333
```

```
attributes(varimp(glmres))$self
```

```
[1] 0.00 0.38 0.57 0.00 0.00 0.05
```

```
attributes(varimp(glmres))$var
```

```
[1] (Intercept)       Sepal.Width       Petal.Length      Petal.Width
[5] Speciesversicolor Speciesvirginica
6 Levels: (Intercept) < Petal.Width < ... < Petal.Length
```

```
firis←as.formula(Sepal.Length~.)
Xiris←model.matrix(firis,iris)
Diris←data.frame(Xiris[,−1],iris$Sepal.Length)
colnames(Diris)[6]←"Y"
coef(glmboost(Xiris,iris$Sepal.Length))
```

```
     (Intercept)       Sepal.Width      Petal.Length  Speciesvirginica
     -3.36560001        0.53639026        0.46090783       -0.01924627
attr(,"offset")
[1] 5.843333
```

```
singboost(Diris)
```

```
$`Selected variables`
[1] "Petal.Length>=Sepal.Width>=Speciesvirginica"

$Coefficients
[1]  2.47773332  0.53639026  0.46090783  0.00000000  0.00000000
   -0.01924627

$Freqs
[1] 0.00 0.38 0.57 0.00 0.00 0.05

$VarCoef
       Intercept       Sepal.Width      Petal.Length  Speciesvirginica
      2.47773332        0.53639026        0.46090783       -0.01924627
```

```
singboost(Diris,LS=T)
```

```
$`Selected variables`
[1] "Petal.Length>=Sepal.Width>=Speciesvirginica"

$Coefficients
[1]  2.47773332  0.53639026  0.46090783  0.00000000  0.00000000
   -0.01924627

$Freqs
[1] 0.00 0.38 0.57 0.00 0.00 0.05

$VarCoef
      Intercept      Sepal.Width      Petal.Length Speciesvirginica
     2.47773332       0.53639026        0.46090783      -0.01924627
```

To include interaction terms, to handle categorical variables or transformations of the regressors, one just needs to deliver the corresponding model matrix to `singboost`.

We restrict ourselves to a simple example with one interaction term.

```
glmres2←glmboost(Sepal.Length~Petal.Length+Sepal.Width:Species, iris)
finter←as.formula(Sepal.Length~Petal.Length+Sepal.Width:Species−1)
Xinter←model.matrix(finter, iris)
Dinter←data.frame(Xinter, iris$Sepal.Length)
singboost(Dinter)
```

```
$`Selected variables`
[1] "Petal.Length>=Sepal.Width.Speciessetosa>=Sepal.Width.Speciesvirginica
   "

$Coefficients
[1] 4.00831289 0.45662497 0.08896750 0.00000000 0.01751541

$Freqs
[1] 0.00 0.60 0.36 0.00 0.04

$VarCoef
                   Intercept                  Petal.Length
                  4.00831289                    0.45662497
   Sepal.Width.Speciessetosa Sepal.Width.Speciesvirginica
                  0.08896750                    0.01751541
```

```
coef(glmres2)
```

```
                  (Intercept)                   Petal.Length
                  -1.83502045                     0.45662497
   Sepal.Width:Speciessetosa  Sepal.Width:Speciesvirginica
                   0.08896750                     0.01751541
attr(,"offset")
[1] 5.843333
```

The number of Boosting iterations and the step size can be easily changed in `singboost` while `glmboost` requires to modify `boost_control`.

In this example, we set $m_{iter} = 250$ and $\kappa = 0.05$.

```
glmres3←glmboost(Xiris, iris$Sepal.Length, control=boost_control(mstop=250,nu=0
   .05))
coef(glmres3)
```

```
     (Intercept)      Sepal.Width     Petal.Length      Petal.Width
    -3.428469064      0.550913963      0.468866896     -0.002836468
Speciesvirginica
    -0.043375444
attr(,"offset")
[1] 5.843333
```

```
attributes(varimp(glmres3))$self
```

```
[1] 0.000 0.348 0.540 0.012 0.000 0.100
```

```
singboost(Diris, m_iter=250,kap=0.05)
```

```
$`Selected variables`
[1] "Petal.Length>=Sepal.Width>=Speciesvirginica>=Petal.Width"

$Coefficients
[1]  2.414864269  0.550913963  0.468866896 -0.002836468  0.000000000
[6] -0.043375444

$Freqs
[1] 0.000 0.348 0.540 0.012 0.000 0.100

$VarCoef
       Intercept      Sepal.Width     Petal.Length      Petal.Width
     2.414864269      0.550913963      0.468866896     -0.002836468
Speciesvirginica
```

```
    -0.043375444
```

```
singboost(Diris ,LS=T,m_iter=250,kap=0.05)
```

```
$`Selected variables`
[1] "Petal.Length>=Sepal.Width>=Speciesvirginica>=Petal.Width"

$Coefficients
[1]  2.414864269  0.550913963  0.468866896 -0.002836468  0.000000000
[6] -0.043375444

$Freqs
[1] 0.000 0.348 0.540 0.012 0.000 0.100

$VarCoef
       Intercept      Sepal.Width     Petal.Length      Petal.Width
     2.414864269      0.550913963      0.468866896     -0.002836468
Speciesvirginica
    -0.043375444
```

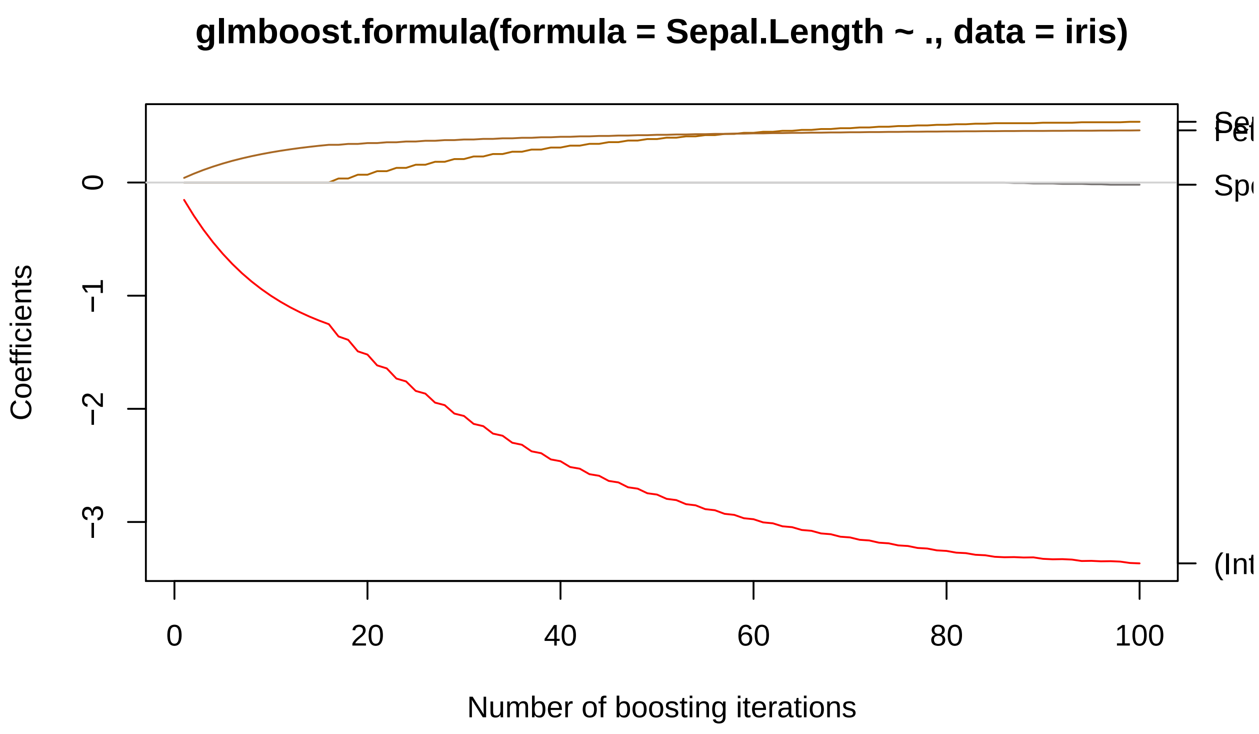
In the first of the two following examples, we show how we can plot coefficient paths for `glmboost`. We just need to apply `plot` to the `glmboost` object.

For SingBoost, we use the function `path.singboost` which exactly makes the same computations as `singboost`, but it additionally saves the path coordinates which would be wasted storage capacities for standard `singboost` since in general, this function is just called internally in `cmb` or `CMB3S`, so in these cases, saving the path coordinates would not be meaningful.

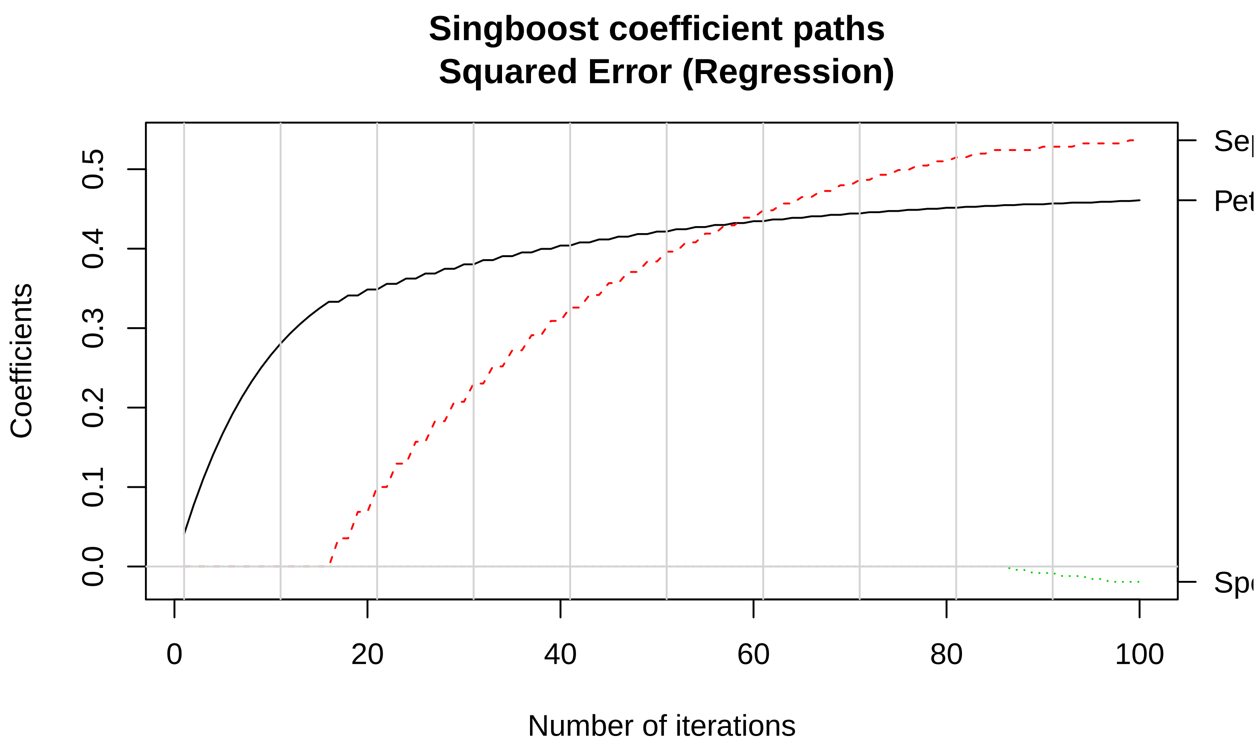
We apply `singboost.plot` to the `path.singboost` object where we also need to insert M and m_{iter} in order to activate the grid to better visualize the coefficient updates in the singular iterations.

Note that we do not include an intercept path since we do not think that it would be interesting in the context of model selection. The intercept path for `glmboost` does not include the offset per default, but it can be added to the intercept at each iteration when using `off2int=TRUE` as input argument for `plot`.

```
plot(glmres)
```



```
singpath←path.singboost(Diris)
singboost.plot(singpath,10,100,subnames=F)
```



An `mboost` family object can be handled over to `singboost` by assigning it to the input variable `singfamily`. Clearly, the results of `singboost` differ from that of `glmboost` once another family object than `Gaussian()` is used. We show the empirical column measures for `singfamily=QuantReg(tau=0.75)`, i.e., for the $L_\tau-$loss with $\tau = 0.75$ (see example 3.2.4), and the coefficients for $M = 10$ (default) and $M = 2$. Remind that we forbid that `singboost` selects the intercept when `LS=T`.

```
glmquant←glmboost(Sepal.Length~. , iris , family=QuantReg(tau=0.75))
coef(glmquant)
```

```
 (Intercept)  Sepal.Width Petal.Length
  -3.1274211    0.5191849    0.4756341
attr(,"offset")
50%
5.8
```

```
attributes(varimp(glmquant))$self
```

```
[1] 0.24 0.27 0.49 0.00 0.00 0.00
```

```
singboost(Diris , singfamily=QuantReg(tau=0.75) ,LS=T)
```

```
$‘Selected variables‘
[1] "Petal.Length>=Sepal.Width>=Speciesvirginica"

$Coefficients
[1]  2.47836568  0.53612658  0.46095525  0.00000000  0.00000000
   -0.01925953

$Freqs
[1] 0.00 0.38 0.57 0.00 0.00 0.05

$VarCoef
       Intercept      Sepal.Width     Petal.Length Speciesvirginica
      2.47836568       0.53612658       0.46095525      -0.01925953
```

```
singboost(Diris , singfamily=QuantReg(tau=0.75) ,LS=T,M=2)
```

```
$‘Selected variables‘
[1] "Petal.Length>=Sepal.Width>=Speciesvirginica"

$Coefficients
[1]  2.44900980  0.54702733  0.45925162  0.00000000  0.00000000
   -0.01196688
```

```
$Freqs
[1] 0.00 0.42 0.55 0.00 0.00 0.03

$VarCoef
       Intercept       Sepal.Width      Petal.Length Speciesvirginica
      2.44900980        0.54702733        0.45925162      -0.01196688
```

The last example is an example for the hard ranking loss. We see for the first time that the variable `Speciesversicolor` has been selected. Of course, we cannot say if the inclusion of this variable is meaningful until here. However, as the mean of `Sepal.Length` significantly differs over the three groups defined by the `Species` variable, we can assume that the `Species` variable indeed yields predictive power when concerning ranking problems.

```
singboost(Diris,singfamily=Rank(),LS=T)
```

```
$`Selected variables`
[1] "Petal.Length>=Sepal.Width>=Speciesvirginica>=Speciesversicolor"

$Coefficients
[1]  2.485989684  0.534163798  0.460436981  0.000000000  0.000344599
[6] -0.018630537

$Freqs
[1] 0.00 0.38 0.56 0.00 0.01 0.05

$VarCoef
        Intercept       Sepal.Width      Petal.Length Speciesversicolor
      2.485989684       0.534163798       0.460436981       0.000344599
 Speciesvirginica
     -0.018630537
```

```
singboost(Diris,singfamily=Rank(),LS=T,M=2)
```

```
$`Selected variables`
[1] "Petal.Length>=Sepal.Width>=Speciesversicolor>=Speciesvirginica"

$Coefficients
[1]  2.492415136  0.535364028  0.455576905  0.000000000  0.010499819
[6] -0.004278119

$Freqs
[1] 0.00 0.43 0.50 0.00 0.04 0.03
```

```
$VarCoef
        Intercept        Sepal.Width      Petal.Length Speciesversicolor
      2.492415136        0.535364028       0.455576905       0.010499819
 Speciesvirginica
     -0.004278119
```

15.1.2 cmb

In the following larger example, we demonstrate the functionalities of our `cmb` function. Again, `Sepal.Length` is our response variable. In the first example, we apply CMB to determine an aggregated empirical column measure for $B^{sing} = 50$ subsamples with $\alpha = 0.2$, so we will only use the information provided by the ten best models w.r.t. the hard ranking loss (inserted as `evalfam`) at the end. For each of the B^{sing} subsamples of size $n_{cmb} = 100$, we apply SingBoost with hard ranking singular steps, learning rate $\kappa = 0.1$ and with 100 iterations each and $M = 10$. We use the aggregation method as in (11.4.2) and do not concern about winsorizing the score vector as proposed in (11.4.5). The interesting parts of the output are the empirical aggregated column measure which clearly can only take values in $\{0, 0.1, ..., 1\}$ since $\alpha B^{sing} = 10$. The row measure reflects the frequencies of each row having been contained in the ten subsamples corresponding to the best models.

In the second example, we just changed M to 2. The third example shows the effect of using (11.4.3) which are loss-weighted, so we do no longer have a perfect 0.1 step size of the entries of the empirical aggregated column measure.

```
set.seed(19931023)
cmb1←CMB(Diris,nsing=100,Bsing=50,alpha=0.2,singfam=Rank(),evalfam=Rank(),
   sing=T,M=10,m_iter=100,kap=0.1,LS=T,wagg='weights1',robagg=F,lower=0)
cmb1
```

```
$`Column measure`
[1] 0.0 1.0 1.0 0.1 0.7 0.9

$`Selected variables`
[1] "Sepal.Width>=Petal.Length>=Speciesvirginica>=Speciesversicolor>=Petal
   .Width"

$`Variable names`
[1] "Intercept"         "Sepal.Width"       "Petal.Length"
[4] "Petal.Width"       "Speciesversicolor" "Speciesvirginica"
```

```
[7] "Y"

$`Row measure`
   [1] 0.8 0.4 0.4 0.6 0.7 0.4 1.0 0.9 0.7 0.6 0.9 0.7 0.8 0.4 0.8 0.6 1.0
  [18] 0.7 0.8 0.7 0.4 0.7 0.4 0.8 0.7 0.8 0.8 0.6 0.6 0.8 0.7 0.9 0.6 0.6
  [35] 0.4 0.8 0.7 0.5 0.7 0.6 0.4 0.4 0.6 0.6 0.4 0.5 0.5 0.6 0.6 0.6 0.6
  [52] 0.7 1.0 0.2 0.7 0.4 0.8 0.7 0.9 0.9 0.6 0.7 0.6 0.8 0.7 0.7 0.7 0.9
  [69] 0.7 0.7 0.8 0.8 0.6 0.5 0.4 0.8 0.7 0.5 0.6 0.6 0.7 0.6 0.4 0.6 0.9
  [86] 0.6 0.7 1.0 0.8 0.7 0.7 0.5 0.7 0.6 0.6 0.5 0.8 0.5 0.9 0.6 0.7 0.7
 [103] 0.6 0.6 0.7 0.9 0.8 0.6 0.6 0.6 0.9 0.5 0.6 0.8 0.5 0.8 0.7 0.7 0.7
 [120] 0.7 0.7 0.8 0.6 0.7 0.7 0.7 0.6 0.7 0.7 0.7 0.7 0.5 0.8 0.5 0.9 0.5
 [137] 0.6 0.6 0.6 0.7 0.8 0.7 0.9 0.5 0.6 0.7 0.7 0.8 0.8 0.6
```

```
set.seed(19931023)
cmb2<-CMB(Diris,nsing=100,Bsing=50,alpha=0.2,singfam=Rank(),evalfam=Rank(),
    sing=T,M=2,m_iter=100,kap=0.1,LS=T,wagg='weights1',robagg=F,lower=0)
cmb2[[1]]
```

```
[1] 0.0 1.0 1.0 0.2 1.0 0.9
```

```
set.seed(19931023)
cmb3<-CMB(Diris,nsing=100,Bsing=50,alpha=0.2,singfam=Rank(),evalfam=Rank(),
    sing=T,M=10,m_iter=100,kap=0.1,LS=T,wagg='weights2',robagg=F,lower=0)
cmb3[[1]]
```

```
[1] 0.0000000 1.0000000 1.0000000 0.1033739 0.6953329 0.8964527
```

15.1.3 `CMB3S`

In this example, we demonstrate how one applies L_2-Boosting to real data using `CMB3S`. As before, we need to compute the model matrix for the training data and of course, the same has to be done for the validation set. The first example is nothing but the application of L_2-Boosting (since `sing=F`, so `singfam` is not used anyway) to the training subsample D^{train} since $B^{sing}=1$, $B=1$ and $n_{cmb}=n_{sing}=n_{train}$, so the "subsample" is the whole training sample. Since we are forced to define the input arguments `gridtype` and `grid`, we enter the $\pi-$grid which just contains the one. Indeed, if we just apply L_2-Boosting once, we clearly get empirical selection frequencies which are either zero or one, so this special $\pi-$grid does nothing but enforcing to include all selected variables into our final model.

At the end, we inserted `singcoef=T` and `Mfinal=10`. If we had inserted `singcoef=F`, the final coefficients had been the coefficients computed by `lm` on the reduced data w.r.t. the columns chosen by L_2-Boosting. Our chosen input arguments let the final coefficients be computed by L_2-Boosting (due to `singfam=Gaussian()`) and `Mfinal` does not have any effect here since the "singular" steps are standard L_2-Boosting steps. As demonstrated below, the coefficients are exactly the same as if one had applied L_2-Boosting directly to the training set. Clearly, this example seems to be totally meaningless, but it shows how easily our Stability Selection can be applied to L_2-Boosting (and once we set $B \gg 1$, we can indeed speak of "stability").

The `CMB3S` algorithm reports the final coefficients, the aggregated empirical column measure and the selected coefficients. In our example, the aggregated empirical column measure is $\{0,1\}-$valued and the grid is $\pi_{grid} = \{0.8, 0.9, 1\}$, i.e., only the three variables to which the column measure assigns weight 1 are ultimately selected and form the stable predictor set. `CMB3S` also reports the aggregated row measure as `CMB`, but we did not show it here since it is of minor interest for now.

```
set.seed(19931023)
ind←sample(1:150,120,replace=F)
Dtrain←Diris[ind,]
Dvalid←Diris[-ind,]
set.seed(19931023)
cmb3s←CMB3S(Dtrain,nsing=120,Dvalid=Dvalid,ncmb=120,Bsing=1,B=1,alpha=1,
    singfam=Gaussian(),evalfam=Gaussian(),sing=F,M=10,m_iter=100,kap=0.1,LS=F,
    wagg='weights1',gridtype='pigrid',grid=seq(0.8,0.9,1),useZeta=F,robagg=F,
    lower=0,singcoef=T,Mfinal=10)
cmb3s$Fin
```

```
   Intercept   Sepal.Width  Petal.Length   Petal.Width
 2.441957843   0.546653819   0.462775638  -0.005221617
```

```
cmb3s$Stab
```

```
[1] 0 1 1 1 0 0
```

```
cmb3s$Sel
```

```
[1] 1 2 3
```

```
glmres4←glmboost(Sepal.Length~.,iris[ind,])
coef(glmres4)
```

```
 (Intercept)   Sepal.Width  Petal.Length   Petal.Width
-3.343875490   0.546653819   0.462775638  -0.005221617
attr(,"offset")
[1] 5.785833
```

In the next example, we apply L_2-Boosting to subsamples of the training data of size 80 for each $b = 1, ..., B = 100$ and count the selection frequencies in the best $\alpha B^{sing} = 5$ models in each stability iteration, i.e., for each subsample. These empirical column measures are aggregated and the final coefficients are L_2-Boosting coefficients, applied to the reduced data. Note that we essentially aggregate 500 ($= B\alpha B^{sing}$) different models, so the aggregated column measure using the weights proposed in (11.4.2) can take values in $\{0, 0.002, ..., 0.998, 1\}$.

```
set.seed(19931023)
cmb3s1←CMB3S(Dtrain,nsing=80,Dvalid=Dvalid,ncmb=100,Bsing=10,B=100,alpha=0.5,
    singfam=Gaussian(),evalfam=Gaussian(),sing=F,M=10,m_iter=100,kap=0.1,LS=F,
    wagg='weights1',gridtype='pigrid',grid=seq(0.8,0.9,1),useZeta=F,robagg=F,
    lower=0,singcoef=T,Mfinal=10)
cmb3s1$Fin
```

```
   Intercept  Sepal.Width Petal.Length
   2.4120982    0.5548308    0.4625010
```

```
cmb3s1$Stab
```

```
[1] 0.000 1.000 1.000 0.590 0.312 0.408
```

Clearly, using `sing=T` and `LS=T`, we apply the CMB-3S algorithm for complicated loss functions like the hard ranking loss function as in the following example. Note that the aggregated selection frequencies for the variables `Speciesversicolor` and `Speciesvirginica` are significantly different from the one in the previous example. The final coefficients are SingBoost coefficients.

Also note that the last three variables get empirical selection frequencies smaller than 0.8 which excludes them from the stable model since $\pi_{grid} = \{0.8, 0.9, 1\}$.

```
set.seed(19931023)
cmb3s2←CMB3S(Dtrain,nsing=80,Dvalid=Dvalid,ncmb=100,Bsing=10,B=100,alpha=0.5,
    singfam=Rank(),evalfam=Rank(),sing=T,M=10,m_iter=100,kap=0.1,LS=T,wagg='
    weights2',gridtype='pigrid',grid=seq(0.8,0.9,1),useZeta=F,robagg=F,lower=0,
    singcoef=T,Mfinal=10)
cmb3s2$Fin
```

```
   Intercept  Sepal.Width Petal.Length
   2.4135578    0.5543613    0.4624923
```

```
cmb3s2$Stab
```

```
[1] 0.0000000 1.0000000 1.0000000 0.5033688 0.6124791 0.5775526
```

Remark 15.1.1. *Thanks to the additional input argument `sing`, we are able to apply our loss-adapted Stability Selection to any existing Boosting algorithm for regression by just setting `sing=F` and using the respective `family` object for `singfam` and `evalfam`. Note again that the combination of `sing=F` with some `singfam` object representing loss L leads to the application of $L-$Boosting in the stability iterations.*

We present an example where we use the Huber loss function. Note again that the stable empirical column measure is somewhat different from the ones that we saw before.

```
set.seed(19931023)
cmb3s3<-CMB3S(Dtrain,nsing=80,Dvalid=Dvalid,ncmb=100,Bsing=10,B=100,alpha=0.5,
    singfam=Huber(),evalfam=Huber(),sing=F,M=10,m_iter=100,kap=0.1,LS=F,wagg='
    weights2',gridtype='pigrid',grid=seq(0.8,0.9,1),useZeta=F,robagg=F,lower=0,
    singcoef=F,Mfinal=10)
cmb3s3$Fin
```

```
   Intercept  Sepal.Width Petal.Length
   2.2575367    0.5939522    0.4722634
```

```
cmb3s3$Stab
```

```
[1] 0.070623411 1.000000000 1.000000000 0.040396108 0.002665918
   0.284898682
```

15.1.4 CV.CMB3S

The CV.CMB-3S algorithm (algorithm 16) is nothing but the application of CMB-3S to different partitions of the data into training, validation and test set to estimate the out-of-sample error of the whole algorithm. To randomly divide the data into these three parts, we wrote a simple function called `random.CVind` where we insert the number of observations, the number of training and validation observations that we require and the number V of partitions. The partitions are generated randomly and not in a $V-$fold manner.

In this first example, we just demonstrate that this is exactly what we implemented for the case of the squared loss. Due to `Bsing=1`, `B=1` and `sing=F`, we did nothing but applying standard L_2-Boosting to the training set generated by `random.CVind` since the validation set is actually not needed since we just have a one-elemental grid. The fitted model is applied to the test set (which contains $150 - 100 - 25 = 25$ observations). At the end, the cross-validated test loss as well as the ultra-stable column measure suggested in section 12.4 are provided, in this part always with equal weights.

The lines below perform the same steps and therefore produce the same loss.

```
set.seed(19931023)
CVind←random.CVind(150,100,25,1)
CV.CMB3S(D,nsing=100,Bsing=1,B=1,alpha=1,singfam=Gaussian(),evalfam=Gaussian()
    ,sing=F,M=10,m_iter=100,kap=0.1,LS=F,gridtype='pigrid',grid=1,useZeta=F,
    Dvalid=Dvalid,ncmb=100,robagg=F,lower=0,singcoef=T,Mfinal=10,CVind=CVind,
    targetfam=Gaussian(),print=F)
```

```
$`Cross-validated loss`
[1] 3.205182

$`Ultra-stable column measure`
[1] 0 1 1 1 0 0
```

```
Dtrain←D[which(CVind[[1]]=='tr'),]
Dvalid←D[which(CVind[[1]]=='v'),]
Dtest←D[which(CVind[[1]]=='te'),]
glmres←glmboost(Y~.,Dtrain)
Gaussian()@risk(predict(glmres,Dtest),Dtest$Y)
```

```
[1] 3.205182
```

Now, we perform the same experiment, but with 50 different partitions of the data, again with 100 instances used for training, 25 for validation and 25 for testing. The first run is again a simple L_2-Boosting model, the second run uses a SingBoost model with hard ranking singular iterations. As we see, there is essentially no difference in the test losses.

The third run uses 25 subsamples for each training set where on each subsample, we again use 5 subsamples and only keep the three best SingBoost models for further aggregation of the empirical column measures. The last run reveals that `evalfam` is indeed an interesting functionality since we compute L_2-Boosting models, but for the change of measure and for the selection of the optimal threshold resp. the optimal final number of variables, we use the loss $\tilde{L}$, here the hard ranking loss.

Due to the fact that the data set just contains five regressor columns (since the column corresponding to `Sepal.Length` is considered to be the response column and the variable `Species` is represented by two columns), there are no differences in the results of the first and fourth resp. of the second and third run since the Stability Selection does not reduce the sparsity compared to the L_2-Boosting model any further.

```
set.seed(19931023)
CVind←random.CVind(150,100,25,50)
set.seed(19931023)
lossglm←CV.CMB3S(D,nsing=100,ncmb=100,Bsing=1,B=1,alpha=1,singfam=Gaussian(),
    evalfam=Gaussian(),sing=F,M=10,m_iter=100,kap=0.1,LS=F,gridtype='pigrid',
    grid=1,useZeta=F,robagg=F,lower=0,singcoef=T,Mfinal=10,CVind=CVind,
    targetfam=Rank(),print=F)[[1]]
mean(lossglm)
```

```
[1] 0.1264753
```

```
set.seed(19931023)
losscmb←CV.CMB3S(D,nsing=100,ncmb=100,Bsing=1,B=1,alpha=1,singfam=Rank(),
    evalfam=Rank(),sing=T,M=10,m_iter=100,kap=0.1,LS=T,gridtype='pigrid',grid
    =1,useZeta=F,robagg=F,lower=0,singcoef=T,Mfinal=10,CVind=CVind,targetfam=
    Rank(),print=F)[[1]]
mean(losscmb)
```

```
[1] 0.1262814
```

```
set.seed(19931023)
losscmb3s←CV.CMB3S(D,nsing=100,ncmb=100,Bsing=5,B=25,alpha=0.6,singfam=Rank()
    ,evalfam=Rank(),sing=T,M=10,m_iter=100,kap=0.1,LS=T,gridtype='pigrid',grid
    =1,useZeta=F,robagg=F,lower=0,singcoef=T,Mfinal=10,CVind=CVind,targetfam=
    Rank(),print=F)[[1]]
mean(losscmb3s)
```

```
[1] 0.1262814
```

```
set.seed(19931023)
lossglmeval←CV.CMB3S(D,nsing=100,ncmb=100,Bsing=5,B=25,alpha=0.6,singfam=
    Gaussian(),evalfam=Rank(),sing=F,M=10,m_iter=100,kap=0.1,LS=F,gridtype='
    pigrid',grid=1,useZeta=F,robagg=F,lower=0,singcoef=T,Mfinal=10,CVind=CVind,
    targetfam=Rank(),print=F)[[1]]
mean(lossglmeval)
```

```
[1] 0.1264753
```

The ultra-stable column measure is indeed also computed by the function `CV.CMB3S` and is delivered to the user as second output argument which we suppressed here since there would be nothing to see.

15.2 Application to the `bodyfat` data set

The `bodyfat` data set from the R−package `TH.data` (Hothorn [2019]) contains $n = 71$ observations with $p = 10$ metric variables.

15.2.1 `singboost`

We just apply `glmboost` and `singboost` with hard ranking singular iterations to the data, treating the column `DEXfat` as response column, implying that nine variables enter as regressor variables.

```
require(TH.data)
f←as.formula(DEXfat~.,)
Xmod←model.matrix(f,bodyfat)
D←data.frame(Xmod[,−1],bodyfat$DEXfat)
colnames(D)[10]←"Y"
glmfat←glmboost(Y~.,D)
singfat←singboost(D,singfamily=Rank())
singfat2←singboost(D,singfamily=Rank(),M=2)
coef(glmfat)
```

```
 (Intercept)          age    waistcirc      hipcirc elbowbreadth
 -98.8166077    0.0136017    0.1897156    0.3516258   -0.3841399
 kneebreadth     anthro3a     anthro3b     anthro3c
   1.7365888    3.3268603    3.6565240    0.5953626
attr(,"offset")
[1] 30.78282
```

```
singfat$Var
```

```
   Intercept          age    waistcirc      hipcirc elbowbreadth
-67.79070097   0.01338179   0.19482863   0.34748391  -0.39690915
 kneebreadth     anthro3a     anthro3b     anthro3c
  1.73242999   3.32686027   3.66769186   0.55192720
```

```
singfat2$Var
```

```
   Intercept          age    waistcirc      hipcirc elbowbreadth
 -68.3973928    0.0240473    0.2064258    0.3562216   -0.2028520
 kneebreadth     anthro3a     anthro3b     anthro3c      anthro4
   1.6339704    0.5153067    2.9598198    0.8530612    1.9505952
```

```
attributes(varimp(glmfat))$self
```

```
 [1] 0.00 0.11 0.06 0.10 0.19 0.30 0.03 0.15 0.06 0.00
```

```
singfat$Freq
```

```
 [1] 0.00 0.11 0.06 0.09 0.19 0.31 0.03 0.15 0.06 0.00
```

```
singfat2$Freq
```

```
 [1] 0.00 0.20 0.11 0.09 0.10 0.22 0.05 0.18 0.01 0.04
```

15.2.2 CMB3S

Since the abovely computed models contain eight resp. nine predictors, a Stability Selection is expected to reduce the number of selected variables in the final model this time. A demonstration of `cmb` seems not to be exciting again.

The first example is essentially borrowed from the documentation of the R−package `stabs` (Hofner and Hothorn [2017]) where the functionalities of the `stabsel` function are shown. We just recapitulate it here for comparison.

```
require(stabs)
mod←glmboost(DEXfat~., data=bodyfat)
set.seed(19990904)
stab←stabsel(mod, q=3,PFER=1,sampling.type="MB")
stab
```

```
         Stability Selection without further assumptions

Selected variables:
waistcirc   hipcirc
        3         4

Selection probabilities:
 (Intercept)          age elbowbreadth  kneebreadth     anthro3c
        0.00         0.00         0.00         0.01         0.14
     anthro4     anthro3b     anthro3a    waistcirc      hipcirc
        0.19         0.23         0.47         0.96         1.00


---
Cutoff: 0.95; q: 3; PFER:  1
PFER corresponds to signif. level 0.1 (without multiplicity adjustment)
```

Again, we have to generate the model matrix in advance before applying `CMB3S`. In this example, we apply our Stability Selection to standard L_2-Boosting models. We use a grid for the number q of final variables where we include at most the five most frequently chosen variables into the final stable model.

Note that for both variables `waistcirc` and `hipcirc`, the aggregated selection frequency is one. For $q = 1$, our algorithm picks one of those variables.

Interestingly, our Stability Selection leads to the same stable model as Hofner's Stability Selection from the previous example, although the aggregated column measures are significantly different.

```
set.seed(19990904)
ind←sample(1:71,15,replace=F)
Dtrain←D[-ind,]
Dvalid←D[ind,]
set.seed(19990904)
stab2←CMB3S(Dtrain,nsing=30,Dvalid=Dvalid,ncmb=45,Bsing=10,B=100,alpha=0.5,
    singfam=Gaussian(),evalfam=Gaussian(),sing=F,M=10,m_iter=100,kap=0.1,LS=F,
    gridtype='qgrid',grid=1:5,useZeta=F,robagg=F,lower=0,singcoef=T,Mfinal=10)
stab2$Fin
```

```
  Intercept    waistcirc      hipcirc
-45.5756329    0.3630883    0.4203918
```

```
stab2$Stab
```

```
 [1] 0.000 0.914 1.000 1.000 0.932 0.844 0.680 0.676 0.992 0.740
```

In the next two examples, we apply CMB-3S with ranking singular steps and with the input argument `evalfam=Rank()`, i.e., the determination of the best models as well as the Stability Selection are based on the hard ranking loss.

The first example uses $M = 10$ and $m_{iter} = 100$ which we changed to $M = 5$ and $m_{iter} = 25$ in the second example since the data set is very small, so letting the Boosting algorithms perform 100 iterations may select too many predictors.

Indeed, the first example resulted in the three equally-stable predictors `age`, `waistcirc` and `hipcirc`. This is astounding since the predictor `age` has never been selected when performing Hofner's Stability Selection as we have seen in the example before the previous example. In the second example, we just got the stable variable `hipcirc`. Taking a look at the stable column measure, we see that the variable `waistcirc` has almost equal selection frequency, but the grid search obviously favoured the sparser stable model.

```
set.seed(19990904)
stab3<-CMB3S(Dtrain,nsing=30,Dvalid=Dvalid,ncmb=45,Bsing=10,B=100,alpha=0.5,
    singfam=Rank(),evalfam=Rank(),sing=T,M=10,m_iter=100,kap=0.1,LS=T,gridtype=
    'qgrid',grid=1:5,useZeta=F,robagg=F,lower=0,singcoef=T,Mfinal=10)
stab3$Fin
```

```
   Intercept          age    waistcirc      hipcirc
-46.86178368   0.07121778   0.34514754   0.41312129
```

```
stab3$Stab
```

```
 [1] 0.0000000 1.0000000 1.0000000 1.0000000 0.9210238 0.8566429 0.7310317
 [8] 0.7572579 0.9723810 0.7534008
```

```
set.seed(19990904)
stab4<-CMB3S(Dtrain,nsing=30,Dvalid=Dvalid,ncmb=45,Bsing=10,B=100,alpha=0.5,
    singfam=Rank(),evalfam=Rank(),sing=T,M=5,m_iter=25,kap=0.1,LS=T,gridtype='
    qgrid',grid=1:5,useZeta=F,robagg=F,lower=0,singcoef=T,Mfinal=10)
stab4$Fin
```

```
  Intercept     hipcirc
-45.9687408   0.7243226
```

```
stab4$Stab
```

```
 [1] 0.0000000 0.5979524 0.9983333 1.0000000 0.2425476 0.3130714 0.6117500
 [8] 0.7710357 0.9449048 0.6815952
```

In the final example for the `bodyfat` data set, we apply the `CV.CMB3S` function in three different ways.

The first example applies our loss-based Stability Selection to L_2-Boosting models and uses the squared loss to determine the optimal grid element and the corresponding stable model, but where the test losses are computed w.r.t. the hard ranking loss as target loss.

In the second example, we indeed use SingBoost and perform the aggregation in CMB and the Stability Selection w.r.t. the hard ranking loss. The last example is a compromise where we compute L_2-Boosting models and perform the CMB aggregation and our Stability Selection w.r.t. the hard ranking loss.

In all examples, we set $n_{train} = 45$, $n_{val} = 13$, $n_{test} = 13$, $n_{cmb} = 35$ and $n_{sing} = 25$. We draw $B = 50$ subsamples in each stability iteration and use the best 5 of 10 models computed by CMB. Note that the final coefficients are based on standard L_2-Boosting in the first and third example while we use final coefficients computed by SingBoost with hard ranking singular iterations and $M^{final} = 2$ in the second example. The variable M^{final} (and M as well) does not have any effect in the first and third example.

Due to the long computational time (of the second example), we ran these simulations on the University's HPC cluster, hence we load the data in subsequent command lines which has been saved as `TH4.RData`.

```
set.seed(19990904)
CVind←random.CVind(71,45,13,50)
set.seed(19990904)
lossglm←CV.CMB3S(D,nsing=25,ncmb=35,Bsing=10,B=50,alpha=0.5,singfam=Gaussian
    (),evalfam=Gaussian(),sing=F,M=2,m_iter=100,kap=0.1,LS=F,gridtype='qgrid',
    grid=1:5,useZeta=F,robagg=F,lower=0,singcoef=T,Mfinal=10,CVind=CVind,
    targetfam=Rank(),print=F)

set.seed(19990904)
losscmb3s←CV.CMB3S(D,nsing=25,ncmb=35,Bsing=10,B=50,alpha=0.5,singfam=Rank(),
    evalfam=Rank(),sing=T,M=2,m_iter=100,kap=0.1,LS=T,gridtype='qgrid',grid
    =1:5,useZeta=F,robagg=F,lower=0,singcoef=T,Mfinal=2,CVind=CVind,targetfam=
    Rank(),print=F)


set.seed(19990904)
lossglmeval←CV.CMB3S(D,nsing=25,ncmb=35,Bsing=10,B=50,alpha=0.5,singfam=
    Gaussian(),evalfam=Rank(),sing=F,M=2,m_iter=100,kap=0.1,LS=F,gridtype='
    qgrid',grid=1:5,useZeta=F,robagg=F,lower=0,singcoef=T,Mfinal=10,CVind=CVind
    ,targetfam=Rank(),print=F)

LTH24← list(lossglm,losscmb3s,lossglmeval)
save(LTH24,file="TH4.RData")
```

```
load("TH4.RData" )
mean(LTH24[[1]]$Cross)
```

```
[1] 0.1099664
```

```
mean(LTH24[[2]]$Cross)
```

```
[1] 0.1007495
```

```
mean(LTH24[[3]]$Cross)
```

```
[1] 0.1084214
```

```
LTH24[[1]]$Ultra
```

```
 [1] 0.00000 0.87584 0.98144 0.99760 0.88792 0.95952 0.81040 0.77168
 [9] 0.73384 0.35320
```

```
LTH24[[2]]$Ultra
```

```
 [1] 0.0000000 0.9885492 0.9978956 0.9996651 0.9292798 0.9530768 0.8505351
 [8] 0.9195646 0.8471244 0.4917235
```

```
LTH24[[3]]$Ultra
```

```
 [1] 0.0000000 0.8794121 0.9864963 0.9984119 0.8781105 0.9321749 0.7664462
 [8] 0.7999610 0.7595124 0.4001568
```

Note that if we indeed performed an "outer Stability Selection", i.e., a Stability Selection based on the ultra-stable column measures, we would again get the same model as Hofner's Stability Selection has computed when considering $q = 2$ or a suitable threshold. However, a further step like this one is definitely not necessary for a data set with $p = 10$ variables.

15.3 Application to a large genomic data set

We analyze the data set `Real.2` from the package `PRIMsrc` (Dazard et al. [2015]) which contains $n = 123$ observations and $p = 946$ variables of which five are nominal (according to the `str` command in R). Note that the nominal variables are not yet declared as factors, so we use the R−command `as.factor`.

We first apply standard L_2−Boosting using the variable `y` as response variable which represents the time until relapse after the patients were cured from lung cancer by surgery. We then apply Hofner's Stability Selection with different adjustments.

In all cases, the variable `delta` is the only stable variable. The warning that appeared indicates that the L_2−Boosting algorithm did not succeed in selecting the required number q

of predictors on 13 of the $B = 100$ subsamples since the maximal number of iterations is $m_{iter} = 100$. It is recommended to increase the number of iterations so that the Boosting models may select sufficiently enough variables. However, instead of increasing m_{iter}, we just try other input arguments here.

```
require(PRIMsrc)
dim(Real.2)
```

```
[1] 123 946
```

```
Real.2$Type←as.factor(Real.2$Type)
Real.2$delta←as.factor(Real.2$delta)
Real.2$KRAS.status←as.factor(Real.2$KRAS.status)
Real.2$EGFR.status←as.factor(Real.2$EGFR.status)
Real.2$P53.status←as.factor(Real.2$P53.status)
glmres←glmboost(y~., Real.2)
glmres
```

```

         Generalized Linear Models Fitted via Gradient Boosting

Call:
glmboost.formula(formula = y ~ ., data = Real.2)


         Squared Error (Regression)

Loss function: (y - f)^2


Number of boosting iterations: mstop = 100
Step size:  0.1
Offset:  3.246341

Coefficients:
      (Intercept)            delta1              Type2 ebv.miR.BART19.5p
      -3.87180108       -2.43275583        -0.11616139       -0.17330521
     hsa.miR.1255a      bkv.miR.B1.3p        hsa.miR.942       hsa.miR.600
      -0.18204381        0.02102917        -0.04936648        0.07823421
      hsa.miR.449b     hsa.miR.508.5p       hsa.miR.1911      hsa.miR.1226
       0.02850915        0.01861387        -0.02605721       -0.02493188
      hsa.miR.1294        hsa.miR.935        hsa.miR.593       hsa.miR.382
       0.03772344        0.17583207         0.02648411       -0.07033468
      hsa.miR.30b.        hsa.miR.564     hsa.miR.486.5p    hsa.miR.140.3p
       0.03109510        0.16999100         0.13116255        0.25070291
     hsa.miR.1269      hsa.miR.1248       hsa.miR.758    hsa.miR.339.3p
      -0.17667881        0.08496770       -0.06818442        0.01951349
    ebv.miR.BART12      hsa.miR.518e      hsa.miR.195.      hsa.miR.548m
       0.14007022        0.02829978        0.02607871       -0.05189332
attr(,"offset")
[1] 3.246341
```

```
set.seed(20111017)
stab←stabsel(glmres,cutoff=0.6,PFER=2)
```

```
Warning in stabsel.mboost(glmres, cutoff = 0.6, PFER = 2): 'mstop' too
   small in 13 of the 100 subsampling replicates to select 'q' base-
   learners; Increase 'mstop' bevor applying 'stabsel'
```

```
stab$sel
```

```
delta1
     2
```

```
set.seed(20111017)
stab2←stabsel(glmres,cutoff=0.55,PFER=2)
stab2$sel
```

```
delta1
     2
```

```
glmres←glmboost(y~.,Real.2,control=boost_control(mstop=200))
set.seed(20111017)
stab3←stabsel(glmres,cutoff=0.6,PFER=2)
stab3$sel
```

```
delta1
     2
```

```
set.seed(20111017)
stab4←stabsel(glmres,cutoff=0.6,q=8)
stab4$sel
```

```
delta1
     2
```

In the next larger example, we apply our `CV.CMB3S` function to this data set. The first application compares the cross-validated squared test loss of L_2−Boosting with the cross-validated squared test loss for CMB-3S using $n_{train} = 90$ instances for training, $n_{val} = 18$ for validation and $n_{test} = 15$ for testing. The third simulation is basically identical, but uses 100 partitions of the data according to $n_{train} = 80$, $n_{val} = 20$ and $n_{test} = 23$.

The second application is based on the same partition as the first one, but the target loss is the hard ranking loss. We compare the performance of L_2−Boosting with the performance of CMB-3S where the hard ranking loss function is also used for validation (since `evalfam`=Rank()).

```
f←as.formula(y~.)
X←model.matrix(f,Real.2)
D←data.frame(X[,−1],Real.2$y)
colnames(D)[948]←"Y"
require(mboost)
mod←glmboost(y~.,Real.2)
glmboost(X,Real.2$y)


set.seed(20111017)
CVind←random.CVind(123,90,18,100)
set.seed(20111017)
     L2Boostmod←CV.CMB3S(D,nsing=60,ncmb=75,Bsing=1,B=1,alpha=1,singfam=
        Gaussian(),evalfam=Gaussian(),sing=F,M=10,m_iter=100,kap=0.1,LS=F,
        gridtype='pigrid',grid=0.001,useZeta=F,robagg=F,lower=0,singcoef=F,
        Mfinal=10,CVind=CVind,targetfam=Gaussian(),print=T)
set.seed(20111017)
L2stabmod←CV.CMB3S(D,nsing=60,ncmb=75,Bsing=10,B=25,alpha=0.5,singfam=
   Gaussian(),evalfam=Gaussian(),sing=F,M=10,m_iter=100,kap=0.1,LS=F,gridtype=
   'qgrid',grid=1:10,useZeta=F,robagg=F,lower=0,singcoef=F,Mfinal=10,CVind=
   CVind,targetfam=Gaussian(),print=T)



L201907092←list(L2Boostmod,L2stabmod)
save(L201907092,file="20190709(2).Rdata")



set.seed(20111017)
CVind←random.CVind(123,90,18,100)
set.seed(20111017)
     L2Boostmod←CV.CMB3S(D,nsing=60,ncmb=75,Bsing=1,B=1,alpha=1,singfam=
        Gaussian(),evalfam=Gaussian(),sing=F,M=10,m_iter=100,kap=0.1,LS=F,
        gridtype='pigrid',grid=0.001,useZeta=F,robagg=F,lower=0,singcoef=F,
        Mfinal=10,CVind=CVind,targetfam=Rank(),print=F)
```

```
set.seed(20111017)
L2stabmod←CV.CMB3S(D, nsing=60,ncmb=75,Bsing=10,B=25,alpha=0.5 ,singfam=
    Gaussian() ,evalfam=Rank() ,sing=F,M=10,m_iter=100,kap=0.1 ,LS=F,gridtype='
    qgrid',grid=1:10,useZeta=F,robagg=F,lower=0,singcoef=F,Mfinal=10,CVind=
    CVind,targetfam=Rank() ,print=F)




L201907093← list(L2Boostmod,L2stabmod)
save(L201907093, file="20190709(3).Rdata")


set.seed(20111017)
CVind←random.CVind(123,80,20,100)
set.seed(20111017)
       L2Boostmod←CV.CMB3S(D, nsing=45,ncmb=60,Bsing=1,B=1,alpha=1,singfam=
           Gaussian() ,evalfam=Gaussian() ,sing=F,M=10,m_iter=100,kap=0.1 ,LS=F,
           gridtype='pigrid',grid=0.001 ,useZeta=F,robagg=F,lower=0,singcoef=F,
           Mfinal=10,CVind=CVind,targetfam=Gaussian() ,print=T)
set.seed(20111017)
       L2stabmod←CV.CMB3S(D, nsing=45,ncmb=60,Bsing=10,B=25,alpha=0.5 ,singfam=
           Gaussian() ,evalfam=Gaussian() ,sing=F,M=10,m_iter=100,kap=0.1 ,LS=F,
           gridtype='qgrid',grid=1:10,useZeta=F,robagg=F,lower=0,singcoef=F,
           Mfinal=10,CVind=CVind,targetfam=Gaussian() ,print=T)
L201907094← list(L2Boostmod,L2stabmod)
save(L201907094, file="20190709(4).Rdata")
```

```
load("20190709(2).Rdata")
mean(L201907092[[1]]$Cross)
```

```
[1] 54.22331
```

```
mean(L201907092[[2]]$Cross)
```

```
[1] 31.86205
```

```
load("20190709(3).Rdata")
mean(L201907093[[1]]$Cross)
```

```
[1] 0.2945601
```

```
mean(L201907093[[2]]$Cross)
```

```
[1] 0.1957148
```

```
load("20190709(4).Rdata")
mean(L201907094[[1]]$Cross)
```

```
[1] 80.75813
```

```
mean(L201907094[[2]]$Cross)
```

```
[1] 51.03435
```

The results are not surprising since p is large here, so the performance of a standard Boosting model is expected to be worse than that of a stable (and therefore also sparser) model. Furthermore, the performances in the first application are better than in the third one, presumably since the training sets are larger in the first example.

15.4 An ultrahigh-dimensional data set

We consider the `riboflavin` data set from the package `hdi` (Dezeure et al. [2015]) which contains $n = 71$ observations with $p = 4088$ metric predictor variables.

First, we use `glmboost` to apply standard L_2-Boosting using variable `y` which is the log-transformed riboflavin production rate as response variable. We see that a lot of predictor variables (more precisely, 32) have been chosen.

Then, we apply Hofner's Stability Selection to see which stable predictor set this algorithm provides. We try two different input arguments for the PFER and set $q = 8$. The stable predictor set just consists of one resp. two variables which seems too sparse keeping in mind that we initially had 4088 variables. Therefore, we apply `stabsel` once more with $q = 20$ and see that five resp. four variables are considered to be stable.

```
require(hdi)
data(riboflavin)
D← riboflavin
str(D)
```

```
'data.frame':   71 obs. of  2 variables:
 $ y: num  -6.64 -6.95 -7.93 -8.29 -7.31 ...
 $ x: AsIs [1:71, 1:4088] 8.49 7.64 8.09 7.89 6.81 ...
```

```
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr  "b_Fbat107PT24.CEL" "b_Fbat107PT30.CEL" "b_Fbat107PT48.CEL
     " "b_Fbat107PT52.CEL" ...
  .. ..$ : chr  "AADK_at" "AAPA_at" "ABFA_at" "ABH_at" ...
```

```
glmres←glmboost(y∼.,D)
glmres
```

```

         Generalized Linear Models Fitted via Gradient Boosting

Call:
glmboost.formula(formula = y ~ ., data = D)


         Squared Error (Regression)

Loss function: (y - f)^2


Number of boosting iterations: mstop = 100
Step size:  0.1
Offset:  -7.159432

Coefficients:
 (Intercept)     xARGF_at     xDNAJ_at     xLYSC_at     xRPLL_at
 4.720422068 -0.124673455 -0.030993712 -0.338737893 -0.018434318
 xSPOIISA_at   xSPOVAA_at     xXHLA_at     xXTRA_at     xYBFI_at
 0.045914863  0.141241666  0.180029364  0.210879579  0.248836246
    xYCGN_at     xYCKE_at     xYCLB_at     xYDDK_at     xYEBC_at
-0.029512538  0.173682086  0.083365177 -0.146928903 -0.495306082
    xYEZB_at   xYFHE_r_at     xYFIO_at     xYHCL_at   xYHDS_r_at
 0.109183594  0.049961166  0.067321325 -0.071138363  0.097810962
    xYKBA_at     xYOAB_at     xYODH_at     xYQJU_at     xYRVJ_at
 0.055341307 -0.524017879  0.026687917  0.080636690 -0.031748532
    xYURQ_at     xYUSP_at     xYWRO_at     xYXEH_at     xYXIE_at
 0.050937499  0.064700361 -0.037745645 -0.026031617 -0.009495365
    xYXLD_at     xYXLE_at     xYYDA_at
-0.229846312 -0.035332669 -0.056751692
attr(,"offset")
[1] -7.159432
```

```
stab←stabsel(glmres,q=8,PFER=2)
stab$sel
```

```
xXHLA_at xYXLD_at
```

```
    1279      4004
```

```
stab←stabsel(glmres,q=8,PFER=1)
stab$sel
```

```
xXHLA_at  xYCKE_at  xYOAB_at  xYXLD_at
    1279      1517      2565      4004
```

```
stab←stabsel(glmres,q=20,PFER=2)
stab$sel
```

```
xLYSC_at  xYCKE_at  xYOAB_at  xYXLD_at
     625      1517      2565      4004
```

```
stab←stabsel(glmres,q=20,PFER=1)
stab$sel
```

```
xLYSC_at  xXHLA_at  xYCKE_at  xYOAB_at  xYXLD_at
     625      1279      1517      2565      4004
```

Due to the large number of variables, we restrict the demonstration of our algorithms to a simple application of our Stability Selection which we apply to standard L_2-Boosting models. We try two $q-$grids here, namely $q_{grid} = \{1, 2, ..., 10\}$ and $q_{grid} = \{11, 12, ..., 20\}$.

```
f←as.formula(y~.)
X←model.matrix(f,riboflavin)
D←data.frame(X[,-1],riboflavin$y)
colnames(D)[4089]←"Y"
set.seed(20160307)
CVind←random.CVind(71,50,10,100)
set.seed(20160307)
      L2Boostcvloss←CV.CMB3S(D,nsing=30,ncmb=40,Bsing=1,B=1,alpha=1,singfam=
         Gaussian(),evalfam=Gaussian(),sing=F,M=10,m_iter=100,kap=0.1,LS=F,
         gridtype='pigrid',grid=0.001,useZeta=F,robagg=F,lower=0,singcoef=F,
         Mfinal=10,CVind=CVind,targetfam=Gaussian(),print=T)$Cross
set.seed(20160307)
      L2stabloss←CV.CMB3S(D,nsing=30,ncmb=40,Bsing=1,B=25,alpha=1,singfam=
         Gaussian(),evalfam=Gaussian(),sing=F,M=10,m_iter=100,kap=0.1,LS=F,
         gridtype='qgrid',grid=1:10,useZeta=F,robagg=F,lower=0,singcoef=F,
         Mfinal=10,CVind=CVind,targetfam=Gaussian(),print=T)$Cross

set.seed(20160307)
```

```
      L2stabloss2←CV.CMB3S(D,nsing=30,ncmb=40,Bsing=1,B=25,alpha=1,singfam=
         Gaussian(),evalfam=Gaussian(),sing=F,M=10,m_iter=100,kap=0.1,LS=F,
         gridtype='qgrid',grid=11:20,useZeta=F,robagg=F,lower=0,singcoef=F,
         Mfinal=10,CVind=CVind,targetfam=Gaussian(),print=T)$Cross

Ribolist←list(L2Boostcvloss,L2stabloss,L2stabloss2)
save(Ribolist,file="Ribo.RData")
```

```
load("Ribo.RData")
mean(Ribolist[[1]])
```

```
[1] 5.617294
```

```
mean(Ribolist[[2]])
```

```
[1] 7.85972
```

```
mean(Ribolist[[3]])
```

```
[1] 3.435842
```

We think that the performance of the stable model corresponding to the grid $q_{grid} = \{1, ..., 10\}$ is worse than that of L_2-Boosting since the number of predictors is too small compared to the very large p.

Chapter 16

Some simulation studies

This chapter starts with a comparison of the out-of-sample performance of L_2-Boosting, of L_2-Boosting combined with our loss-based Stability Selection and of CMB-3S where we use the hard ranking loss as target loss. The goal of this simulation study is to get empirical evidence that we indeed benefit from taking singular parts into account. We usually use data sets of different size and with a low signal-to-noise ratio, but we also provide two results for a high signal-to-noise ratio.

The second section is devoted to a closer look on our Stability Selection. We start with a fairly small but noisy data set and apply Hofner's Stability Selection combined with L_2-Boosting, L_2-Boosting in its pure form and our Stability Selection combined with L_2-Boosting with the squared loss as target loss. We see that Hofner's Stability Selection dramatically suffers from the low signal-to-noise ratio and that even a grid search for the best input arguments does not suffice to cope with such situations.

We provide further scenarios where we apply our Stability Selection to high-dimensional noisy data sets and compare the performance of the resulting model with L_2-Boosting when we have the squared loss and for the hard ranking loss as target loss and with Huber-Boosting when taking the Huber loss as target loss. We see that our Stability Selection is indeed able to compute reliable models which outperform standard Boosting models for low signal-to-noise ratios.

16.1 Singular parts for hard ranking

In this section, we perform the following experiment for different scenarios: We compute a standard L_2-based empirical column measure using Column Measure Boosting with standard L_2-Boosting models and a $\tilde{L}-$based empirical column measure with `singfam=Rank()` and `evalfam=Rank()`. Then, we perform our loss-based grid search and compute the performance of the model based on the reduced predictor set using standard L_2-Boosting resp. SingBoost with hard ranking singular iterations on a validation set. For simplicity, we always consider $B = 1$ and $n_{cmb} = n_{train}$, i.e., we do not combine CMB column measures but only use one CMB column measure, based on a relatively large number B^{sing} of resamples.

This is repeated for several different realizations of the data and the out-of-sample losses are compared. We furthermore compute the test losses of L_2-Boosting where the model is based on only the training set for comparability since the validation set is only used for the grid search when applying the other algorithms, so their coefficients are also based only on the training set.

More precisely, having computed two empirical column measures $\hat{\nu}^{(L_2)}$ and $(\hat{\nu}_{\tilde{L}}^{L_2})^{CMB}$, we proceed in exactly the same way as when computing our stable models with the only difference that one model is based on $\hat{\nu}^{(L_2)}$ when applying the grid search and the other one on $(\hat{\nu}_{\tilde{L}}^{L_2})^{CMB}$ for the hard ranking loss $\tilde{L}$. At the end, the coefficients for the latter model are computed by SingBoost, the other ones by standard linear regression. In other words, the models **differ by the paradigm if one accounted for singular parts or not** .

Remark 16.1.1 (CMB-2S). *To emphasize that we do not perform singular iterations but only apply our Stability Selection to standard Boosting models, we call the procedure CMB-2S (the "S" in CMB-3S corresponding to the word "singular" is missing) for the rest of this work.*

Scenario 1: We have $p = 500$, $n = 250$, $s^0 = 20$, SNR $= 0.5$ and the non-zero coefficients are $\mathcal{N}(2.5, 1)-$distributed. The rows of the regressor matrix are multivariate normal without correlation. We use $n_{train} = 160$, $n_{cmb} = n_{train}$, $n_{val} = 40$ and $n_{test} = 50$, $B = 1$, $B^{sing} = 100$, $\alpha = 0.2$, $m_{iter} = 100$, $\kappa = 0.1$, $M = 10$ and $M^{final} = 2$ for the final SingBoost coefficients. For the Stability Selection, we use a grid $\pi_{grid} = \{0.5, 0.55, ..., 1\}$. We consider 1000 different data sets and $V = 25$ partitions into training and validation data as described above.

Scenario 2: Same as scenario 1, but we have SNR $= 1$ and we use $M^{final} = 5$ and a grid $q_{grid} = \{1, 2, ..., 20\}$.

Scenario 3: Same as scenario 1, but with $M^{final} = 5$.

Scenario 4: Same as scenario 3, but with $s^0 = 10$.

Scenario 5: Same as scenario 4, but with $p = 1000$.

Scenario 6: Same as scenario 5, but with $M^{final} = 2$.

Scenario 7: Same as scenario 1, but with $s^0 = 10$ and $M^{final} = 5$.

Scenario 8: Same as scenario 7, but with $p = 1000$.

Scenario 9: Same as scenario 8, but with $M^{final} = 2$.

The results can be found in table 16.1.

Scenario	CMB-2S>GLM	CMB-3S>CMB-2S	CMB-3S>GLM
Scenario 1	57%	53%	65%
Scenario 2	64%	50%	75%
Scenario 3	58%	55%	66%
Scenario 4	49%	73%	78%
Scenario 5	49%	72%	79%
Scenario 6	51%	71%	77%
Scenario 7	45%	75%	77%
Scenario 8	50%	73%	80%
Scenario 9	52%	71%	78%

Table 16.1: First column: Relative number of data sets on which CMB-2S performed better than L_2-Boosting; analogous interpretation of the second and third column

Remark 16.1.2. *It may sound unfair at the first glance to compare models whose coefficients have been computed by linear regression resp. by SingBoost since the latter performs model selection. But in our setting, both computations are based on a subset of the selected variables which has been optimized* ***separately****. Even if for, say, $\pi_{thr} = 0.6$, the SingBoost model again deleted some variables when computing the coefficients which leads to a sparser model, the same sparsity may be achieved for a higher threshold π_{thr}. Since the grid π_{grid} is chosen sufficiently fine, the respective sparsity may also be available for the stable L_2-based model based on which the linear regression coefficients are computed.*

To guard against possible weaknesses, we let our grid run from 0.5 to 1 in suitably small steps (step size $\Delta = 0.05$). The change of measure performed by CMB indeed states that a *column measure component of 1 means that the corresponding variable has been detected by all SingBoost algorithms before, so running SingBoost on a set corresponding to such a high threshold with a usually lower value for M^{final} than for M should not delete (a considerable amount of) variables in the last step since a SingBoost algorithm with a higher frequency of singular iterations is expected to select even more variables than with a lower frequency of singular iterations.*

Apart from this, although Meinshausen and Bühlmann [2010] and Hofner et al. [2015] did not make recommendations how to compute the coefficients on the model that the Stability Selection has chosen, it would be unlikely that they would perform another Lasso or Boosting step after the Stability Selection. This again motivates the computation of our final least squares coefficients in the CMB-2S experiments since it resembles the original Stability Selection most and is expected to perform not worse thanks to the grid search for the parameters.

Finally, we remind that the grid search is based on aggregated selection frequencies computed by different SingBoost models, so we can expect that performing model selection by choosing a higher threshold in the grid search would lead to a more sophisticated model than in the case that SingBoost itself selects variables in the final step.

Note the considerable difference between the results from scenarios 1-3 and scenarios 4-9. In the first three scenarios, CMB-2S already beats L_2-Boosting in the majority of cases and CMB-3S is better than L_2-Boosting in even more cases, but on approximately the half of the data sets, CMB-3S cannot beat CMB-2S in terms of hard ranking performance. On the other hand, CMB-2S is not better than L_2-Boosting in the latter six scenarios but CMB-3S outperforms CMB-2S and L_2-Boosting on a clear majority of data sets.

We try to explain these results. In the first three scenarios, there are already 20 out of 500 variables which are relevant. Maybe even if there was a singular part $J_{\tilde{L}}^{L_2}$, the gain in performance when selecting variables from it would be superimposed by the other variables, even when considering stable sets. In scenarios 4-9, the relative importance of a singular part in terms of ranking performance may be higher compared to the importance of all 10 relevant variables. Therefore, our interpretation of the results is that in scenarios 1-3, we do not significantly benefit from accounting for singular parts but that this is indeed the case for scenarios 4-9. It is interesting to see that even a stable model does not improve the hard ranking performance here if it ignores potential singular parts.

The latter aspect is indeed a very strong result which justifies our assumption 10.1.3 that the true column measures differ on the stable sets and which once more demonstrated that issues as shown in example 10.2.1 are not just a consequence of the analyst's input arguments for model selection algorithms but that there is indeed more theoretical background.

To be honest, we once more take a look at scenarios 1 and 6, but where we generate data sets with a rather high signal-to-noise ratio.

Scenario 10: Same as scenario 1, but with SNR = 2.5.

Scenario 11: Same as scenario 6, but with SNR = 2.5.

Scenario	CMB-2S>GLM	CMB-3S>CMB-2S	CMB-3S>GLM
Scenario 10	92%	26%	94%
Scenario 11	63%	63%	92%

Table 16.2: First column: Relative number of data sets on which CMB-2S performed better than L_2-Boosting; analogous interpretation of the second and third column

The results in 16.2 show that in cases of strong signals, a Stability Selection already leads to a better hard ranking performance than simple L_2-Boosting models do in the majority of cases. This is no contradiction to the results in the next section which we discuss there.

An interesting case appears in scenario 10 where CMB-3S performs worse than CMB-2S on the most data sets. We are not sure how this can be explained since the argumentation why CMB-3S performs better only one the half of the data sets as in scenarios 1-3 does not suffice. We think that in the case of a strong signal and a large number of true non-zero coefficients, the Boosting models are rather greedy and select many (relevant) variables. The change of measure that is performed when invoking singular iterations may cause some aggregated selection frequencies becoming too low (when some SingBoost models do not contain the respective variables) to let the corresponding variables enter the stable model which essentially leads to false negatives induced by CMB-3S. That could explain that CMB-3S performs worse than CMB-2S in the most cases. Note that CMB-3S indeed beats L_2-Boosting in nearly all cases. If one takes a look at the hard ranking losses themselves, we see that the aggregated hard ranking loss of the CMB-3S models over all data sets is lower than the loss of the L_2-Boosting models but higher than the aggregated loss of the CMB-2S models.

The results of scenario 11 are different to that of scenario 6 in the sense that CMB-2S beats L_2-Boosting on a clear majority of data sets and that CMB-3S performs better than

L_2-Boosting on nearly all data sets.

However, our experiments confirm that respecting singular parts or at least performing a suitable change of measure is indeed recommended since **in each scenario, CMB-3S beats L_2-Boosting in the majority of cases, even if a pure Stability Selection does not improve the out-of-sample performance**.

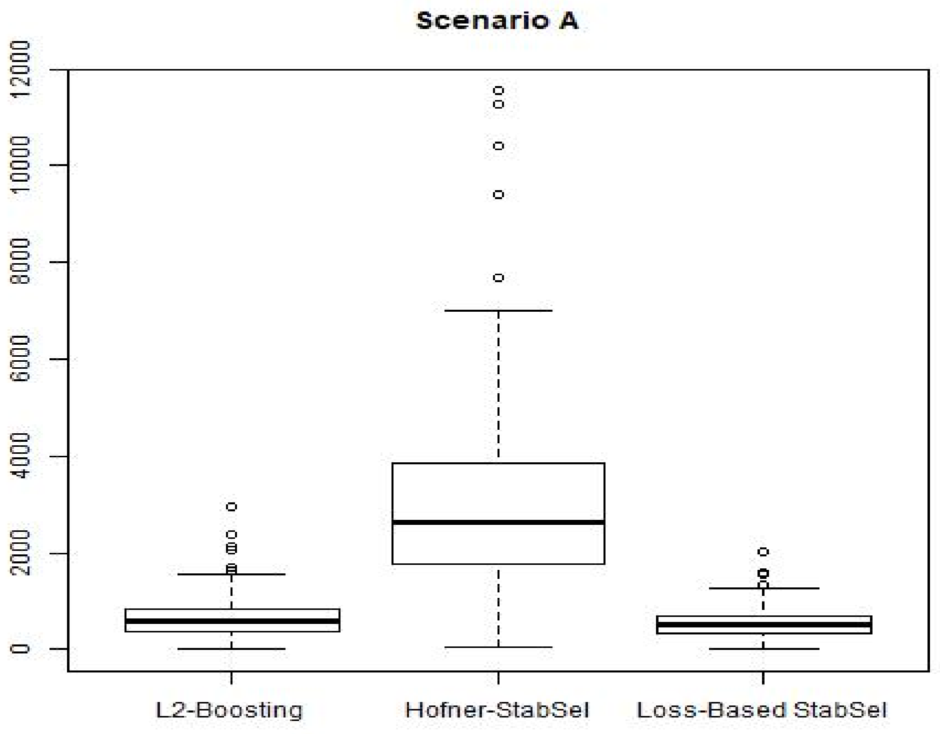
16.2 The power of our loss-based Stability Selection

In this section, we show that our Stability Selection is indeed capable to handle high-dimensional noisy data. In contrast, the Stability Selection proposed by Hofner (Hofner et al. [2015]) fails in such cases, especially if the signal-to-noise ratio is very low. The reason is that in such cases, there are seldom clearly dominating variables in terms of selection frequencies, or in other words, the Boosting algorithm gets irritated by the noise variables, resulting in no variable at all passing the threshold in the Stability Selection.

One can argue that one just has to modify either q, the PFER or the threshold when applying `stabsel`, but as we show in scenario A, even that does not result in good models.

Scenario A: We have $p = 100$, $n = 250$, $s^0 = 5$, SNR $= 0.5$ and the non-zero coefficients are standard normally distributed. The rows of the regressor matrix are multivariate normal without correlation. We use $n_{train} = 150$, $n_{val} = 50$ and $n_{test} = 50$, $B^{sing} = 20$, $\alpha = 0.5$, $B = 25$, $m_{iter} = 100$, $n_{cmb} = 120$, $n_{sing} = 100$ and $\kappa = 0.1$. For the Stability Selection, we use a grid $q_{grid} = \{1, 2, ..., 10\}$. We consider 200 different data sets and $V = 10$ different partitions into training, validation and test part for each data set.

For a fair comparison, we use two grids, i.e., the grid $\{1, ..., 10\}$ for Hofner's q and the grid $\{1, ..., 5\}$ for the PFER. For each combination, we apply `stabsel` and compute the cross-validated test loss based on the $V = 10$ partitions. At the end, we only report the **minimal test loss**.



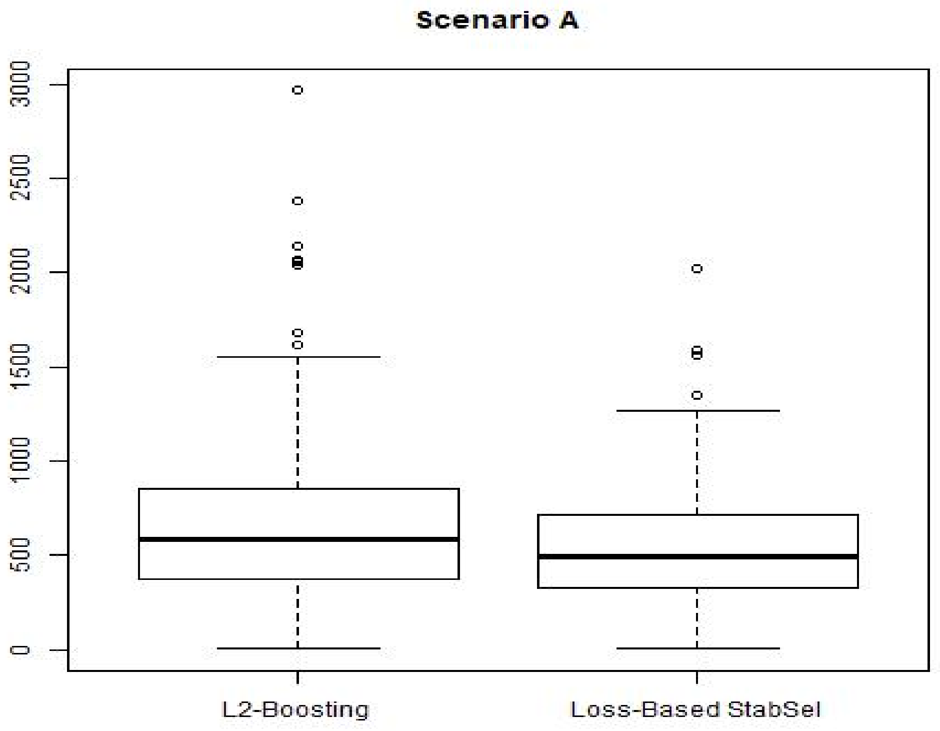


Figure 16.1: Cross-validated test losses of L_2-Boosting, Hofner's Stability Selection combined with linear regression and our Stability Selection combined with linear regression

We see in the first plot in figure 16.1 that even on such a small data set, Hofner's Stability Selection already fails due to the low signal-to-noise ratio. Our Stability Selection does not improve the L_2-Boosting models significantly as seen in the second plot in figure 16.1, but since we just have $p = 100$, however, we do not expect a great benefit from the Stability Selection.

Therefore, we propose some other scenarios where we only compute the cross-validated test losses for the model arising from our Stability Selection and for standard Boosting (or L_2-Boosting in the case of the hard ranking loss), not regarding Hofner's Stability Selection anymore since the data that we use are at least equally noisy and at least as large as in Scenario A. We do not concern about singular iterations here, so in fact we apply the algorithm **CMB-2S**.

Scenario B: Same as scenario A with 1000 other data sets and without Hofner's Stability Selection.

Scenario C: Same as scenario B, but with the Huber loss function instead of the squared loss.

Scenario D: Same as scenario B, but with the hard ranking loss function instead of the squared loss. Note that since we do not have a standard Boosting procedure for the hard ranking loss, we compare the cross-validated test loss of our stable model with the cross-validated test loss of standard L_2-Boosting.

Scenario E: Same as scenario A, but with $p = 500$ and $s^0 = 10$ and without Hofner's Stability Selection.

Scenario F: Same as scenario E, but with the Huber loss function instead of the squared loss.

Scenario G: Same as scenario E, but with $p = 1000$ and 500 different data sets.

Scenario H: Same as scenario G, but with the Huber loss function instead of the squared loss.

Scenario I: Same as scenario H, but with the hard ranking loss function instead of the squared loss.

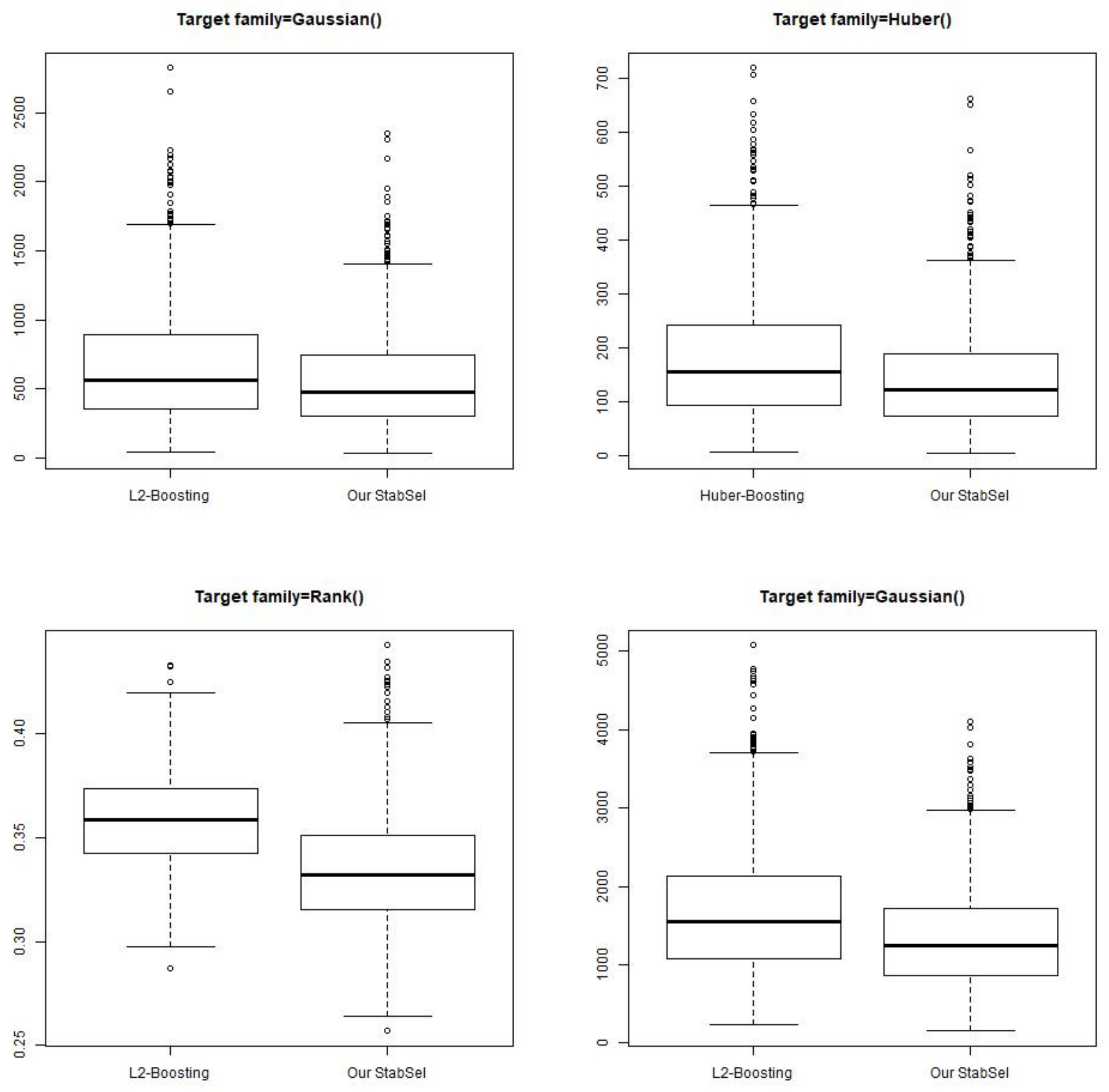


Figure 16.2: Upper left: Scenario B; Upper right: Scenario C; Bottom left: Scenario D; Bottom right: Scenario E

We see in figures 16.2 and 16.3 that the models resulting from our loss-based Stability Selection always outperform the standard Boosting models in terms of the cross-validated test loss w.r.t. $\tilde{L}$. As expected, the improvement w.r.t. the squared and the Huber loss is even better the higher the number p of predictors is.

Despite we also significantly beat L_2-Boosting w.r.t. the hard ranking loss, the difference is not as significant as for the other losses in the sense that the highest losses exceed the highest losses produced by L_2-Boosting. This highlights again the anomality of the hard ranking loss since although the L_2-Boosting model may select several noise variables, the hard ranking performance does not necessarily suffer. However, if one desires to have a well-interpretable model, a stable and therefore sparser and more reliable predictor set is the better choice.

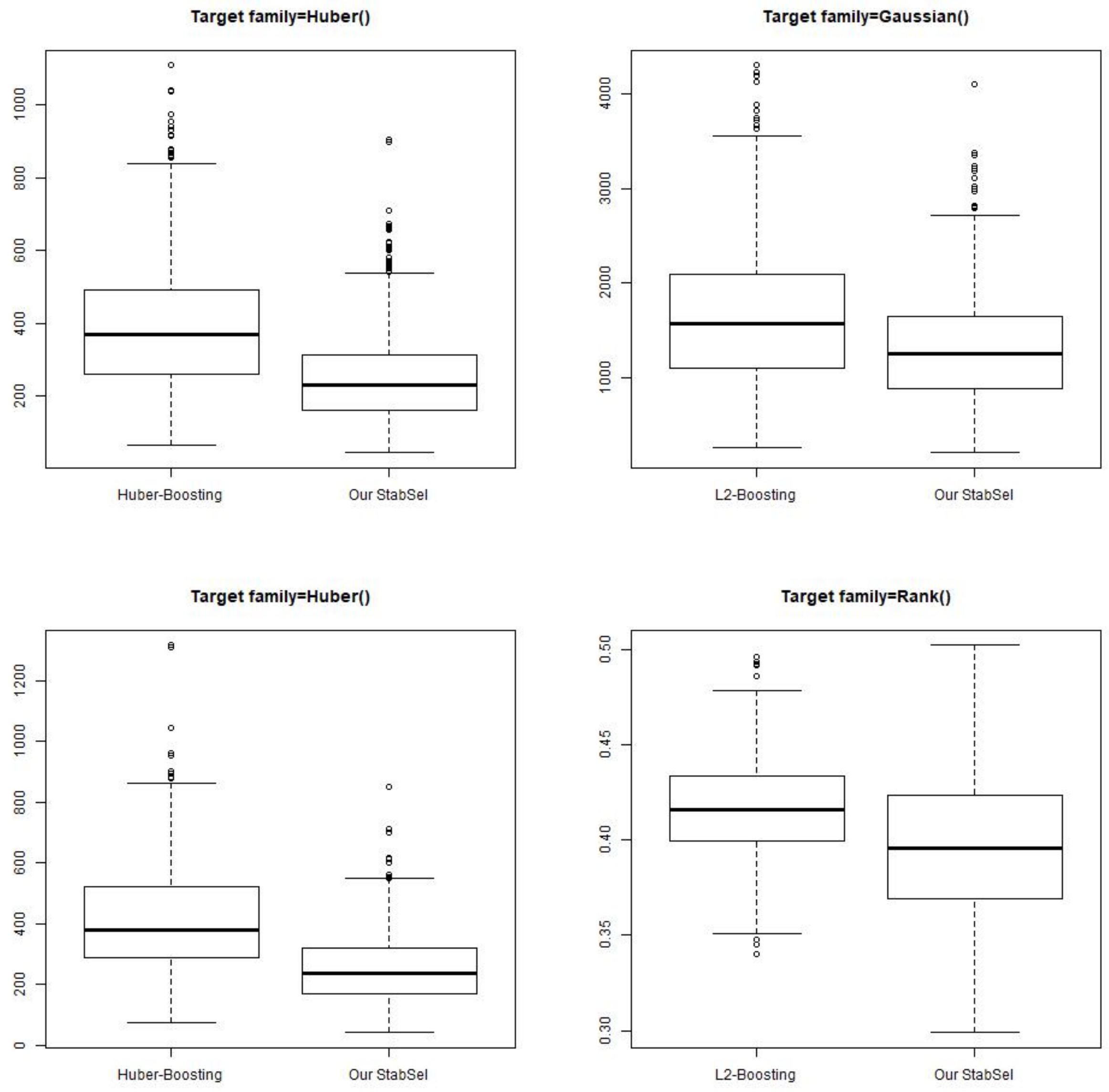


Figure 16.3: Upper left: Scenario F; Upper right: Scenario G; Bottom left: Scenario H; Bottom right: Scenario I

Note that these results are no contradiction to the results of the previous section since we compare inherently different strategies. In the previous section, we first computed an L_2-based column measure and a column measure which respects singular parts and computed the reduced predictor set using our Stability Selection. In contrast, the experiments in this section did not invoke singular parts but a pure investigation of the performance of our Stability Selection. As we have seen, both approaches are promising, so it is expected that their combination. i.e., our complete CMB-3S algorithm, would outperform standard approaches even more significantly.

To be honest, we also provide some simulation results where we used data with a much stronger signal.

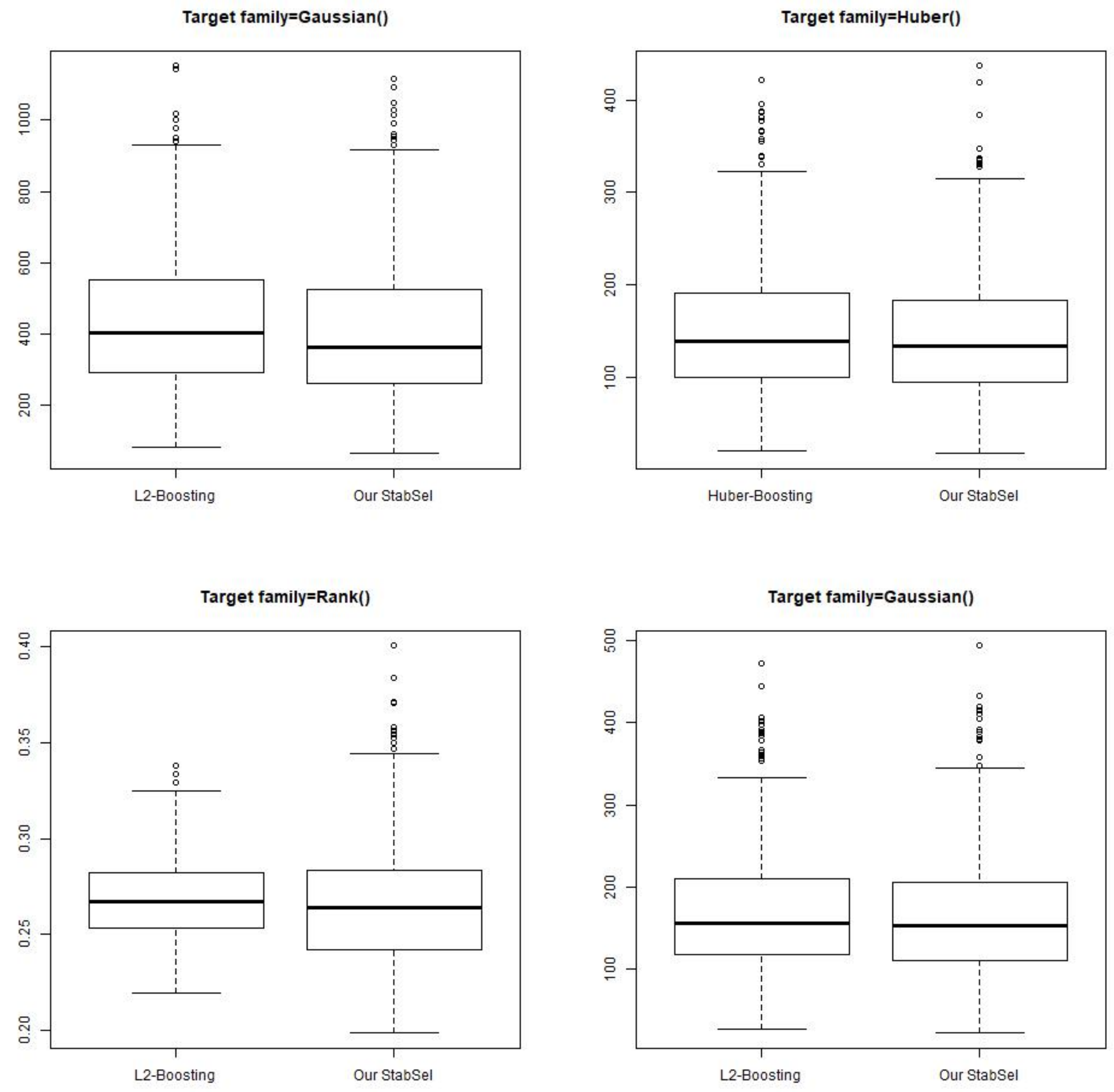


Figure 16.4: Upper left: Scenario J; Upper right: Scenario K; Bottom left: Scenario L; Bottom right: Scenario M

Scenario J: Same as scenario G, but with SNR = 2.

Scenario K: Same as scenario H, but with SNR = 2.

Scenario L: Same as scenario I, but with SNR = 2.

Scenario M: Same as scenario J, but with SNR = 5.

Scenario N: Same as scenario K, but with SNR = 5.

Scenario O: Same as scenario L, but with SNR = 5.

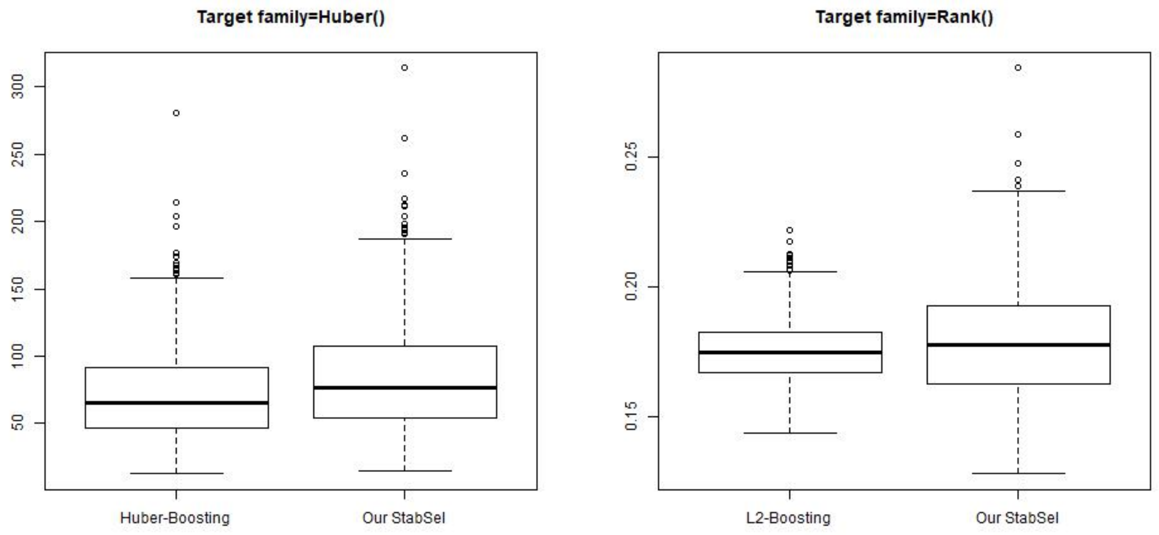


Figure 16.5: Left: Scenario N; Right: Scenario O

As we see in figures 16.4 and 16.5, the improvement of the performance when using our Stability Selection is no longer significant for SNR = 2.5. However, the result from scenario L does not contradict the result from scenario 11 since a mean of the test losses of our stable models which is not significantly smaller than the respective mean for the L_2-Boosting models does not imply that we did not beat L_2-Boosting on the majority of data sets.

Remark 16.2.1 (**Future simulation studies**). *In particular, one of the most interesting simulation studies would be the performance of a fully-tuned CMB-3S algorithm for hard and localized continuous ranking problems where no corresponding Boosting algorithm already exists. One would need at least two competitors for each setting: L_2-Boosting with Stabil-*

ity Selection which sparsely approximates the conditional expectation $\mathbb{E}[Y|X]$ and therefore delivers the approximation of an optimal scoring rule ([Clémençon and Achab, 2017, Prop. 1]), but with a model that is not $\tilde{L}-$adapted, and clearly the CRank algorithm of Clémençon and Achab [2017] which indeed fits no linear model but a step function. Although a sparse linear model beats tree-based algorithms in terms of interpretability, we are curious how the algorithms would be ranked in terms of their out-of-sample ranking performance, especially on very high-dimensional noisy data.

However, a universal comparison of CMB-3S with competitors is out of scope for this work since CMB-3S can be adapted to probably any loss function $\tilde{L}$ and rather forms a whole family of algorithms. Remind again that, to the best of our knowledge, it is not even known if for a given data set, one should either apply L_2-Boosting or the Lasso (Meinshausen et al. [2007], Bühlmann and Hothorn [2007]).

Remark 16.2.2. *Let us remind again that the computational time of the algorithms invoking singular iterations is rather expensive due to a simple* R*−implementation. Once this cost has been massively reduced by calling C and FORTRAN code, the additional complexity caused by the singular iterations is negligible, compare lemma 10.3.1 and remark 10.3.15.*

The Stability Selection causes essentially nothing but a scaling of the complexity of the underlying Boosting algorithm by the number B of subsamples for the usual Stability Selection and by the product BB^{sing} for CMB-3S.

We did not yet point out how we would proceed in the case of extremely high p such that even L_2-Boosting gets infeasible. Such dimensions can indeed occur in practice especially in the presence of categorical variables and interactions, see table 1.2. In this case, we would need a suitable pre-processing step.

Remark 16.2.3 (Handling ultra-high dimensions). *If p is extremely large, we recommend to use the (Iterative) Sure Independence Screening ((I)SIS) procedure from Fan and Lv [2008] to perform a screening step such that all but $d < n$ variables are eliminated first which is implemented in* R *in the package* ***SIS*** *(Saldana and Feng [2018]).*

SIS is based on the correlations of each predictor column with the response and only selects the d variables with the highest absolute correlation with the response. ISIS is an iterative procedure to address issues like correlation of noise variables with predictors or multicollinearity. After selecting a set of variables like in SIS, the corresponding responses w.r.t. a linear regression model are computed and SIS is again performed w.r.t. the set of remaining predic-

tors and so forth, where finally the union of all selected sets (with cardinality d) is taken.

After having performed screening using SIS or ISIS, a variable selection method like the Lasso is then performed on the reduced predictor set according to Fan and Lv [2008]. In contrast, we would proceed by applying CMB-3S on the reduced data.

There is no contradiction to the variable selection respecting singular parts that we want to perform since (I)SIS selects a predictor set which asymptotically contains the true predictor set with probability 1. This is exactly the property which we need since we try to select a stable subset of the true predictor set which is $\tilde{L}-$*adapted and therefore potentially different from an* L_2-*adapted stable predictor set, but both should be subsets of the true predictor set.*

We remind again that SingBoost uses simple least squares models as baselearners, so a very low correlation of a predictor column with the response vector indeed indicates that the corresponding predictor is not relevant at all. See also our discussion in sections 10.4 and 10.5.

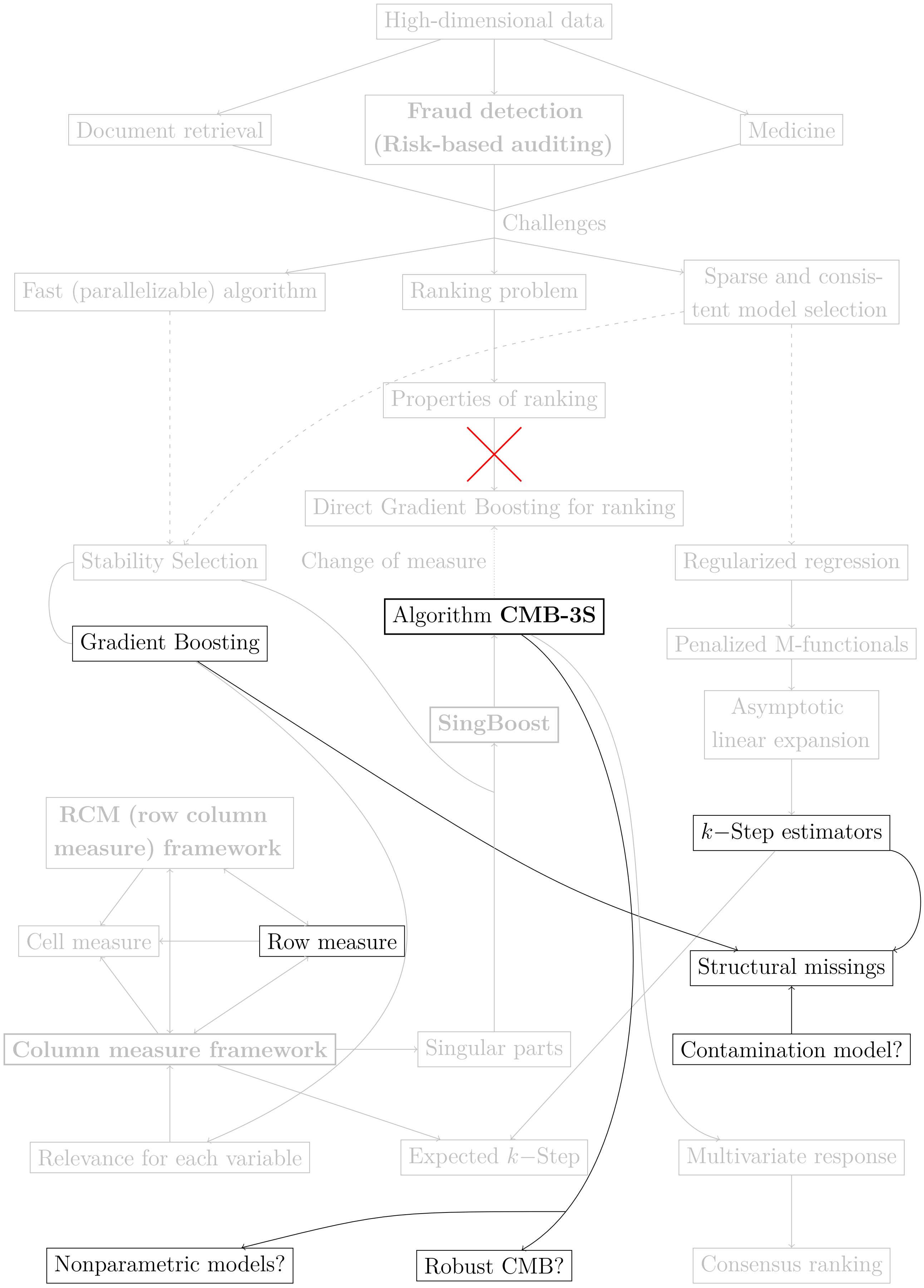
16.2.1 Availability of the R code

The R code for the algorithms developed in this thesis are available in the Cloud-Storage of the Carl von Ossietzky Universität Oldenburg at

https://cloudstorage.elearning.uni-oldenburg.de/s/DJt5FLxKHC8WYRB.

Part VII

Outlook


High-dimensional data
Document retrieval
Fraud detection (Risk-based auditing)
Medicine
Challenges
Fast (parallelizable) algorithm
Ranking problem
Sparse and consistent model selection
Properties of ranking
Direct Gradient Boosting for ranking
Stability Selection
Change of measure
Regularized regression
Algorithm CMB-3S
Gradient Boosting
Penalized M-functionals
SingBoost
Asymptotic linear expansion
RCM (row column measure) framework
$k-$Step estimators
Cell measure
Row measure
Structural missings
Column measure framework
Singular parts
Contamination model?
Relevance for each variable
Expected $k-$Step
Multivariate response
Nonparametric models?
Robust CMB?
Consensus ranking

Chapter 17

Learning with missing data

Missings are a frequently faced problem in many real-world data sets (see e.g. Little and Rubin [2014]) ranging from astronomic (Wagstaff [2004]) to click-through data (Haffari et al. [2008]), but they are especially relevant for biological or medical data (see e.g. Stekhoven and Bühlmann [2011], Städler et al. [2014]).

In the following sections, we carefully distinguish between two main types of missings: structural and non-structural missings. Since there already exist a lot of work on non-structural missings, we propose two possible strategies to work with data containing structural missings. Although our proposals are not yet theoretically founded, they could be superior to any complete case algorithm for certain missingness patterns, aside from the fact that imputation algorithms are meaningless in the presence of structural missings.

The last section of this chapter proposes a new concept that tries to connect missings with contamination balls borrowed from robust statistics.

17.1 Types of missings

Usually, one distinguishes between three types of missing values (see e.g. Friedman et al. [2001]). We just repeat these (unfavorably named) definitions informally.

A missing is of **type MCAR** ("missing completely at random") if the missing scheme, i.e., the distribution of the missingness, is independent from the observed as well as from the unobserved values. If the distribution of the missingness is still independent from the unobserved, but dependent from the observed data, then one calls the missings of **type MAR**

(”missing at random”). In contrast, a missing scheme where the missingness also depends on the unobserved data is referred to as of **type MNAR** (”missing not at random”).

Example 17.1.1. *There are many standard examples for these situations in literature. If a person which is old refuses to answer a question because of his or her age (but the age is reported), then the missing is of type MAR since it only depends on the (observed) variable age. On the other hand, if a person does not want to report her or his income because the income is very low or very high, then we face a missing of type MNAR since the reason for non-reporting, i.e., the variable income, remained unobserved. A MCAR missingness pattern can be interpreted in the sense that this information somehow got lost, as if one had randomly distributed inkblots on the questionnaires.*

Apart from the three types of missingness already listed, there is another one that seems not having gained too much attention yet. Consider a medical questionnaire where there is a question about the pregnancy duration. If the answering person is male or a child, there clearly has to be a missing. In contrast to MAR (note that the missing here depends on the age resp. the sex of the person), there is no randomness since if we knew that we have a male person or a child, we could have predicted with (nearly?) certainty that this missing occurred which is not true for a missingness scheme of type MAR. Note also that if the answering person is female and an adult, the missingness of the pregnancy duration variable can be of each type (MCAR, MAR, MNAR) without additional information.

Those non-random missings are, to the best of our knowledge, seldomly treated in literature and are referred to under different names. For example, those missingness schemes are called skip patterns (Arslanturk et al. [2012]) or conditional missing patterns (Cohen and Huang [2000]). Other variants are cited in Manewitsch [2013]. We think that the names STMIS (”strukturelle missings”, Manewitsch [2013]) and ”structurally absent” values (Chechik et al. [2008]) are the most appropriate to reflect their nature, thus we will refer to such missings as **structural missings** and to the other types of missings as **non-structural missings**. Note that, as already pointed out in Manewitsch [2013], structural missings are informative.

17.2 Handling non-structural missings

There are tons of literature concerning methods to deal with non-structural missings. In fact, a crucial part is often to decide of which type the missingness pattern is. A complete case analysis which just deletes rows containing missings is only justifiable if the missingness scheme is of type MCAR or if it is of type MAR with extra conditions (see Dimou [2011], Boyko [2013]).

A common strategy is imputation which means to replace the missings by plausible values. As already stated in Friedman et al. [2001], many imputation methods require a missingness pattern of type MCAR. There is a vast variety of imputation methods ranging from multiple imputation by chaining equations (MICE, see Azur et al. [2011]) to Boosting- (Wang and Feng [2009]), SVM- (Pelckmans et al. [2005]) and RandomForest-type (Stekhoven and Bühlmann [2011]) methods.

Other techniques include the famous EM-algorithm (Dempster et al. [1977]), see also Loh and Wainwright [2012] for theoretical properties in high-dimensional settings, or tree-based algorithms. The latter are capable to deal with missings without deleting them nor imputing them. In fact, for CART, missings in columns corresponding to a categorical variable just count as an extra category where in the case of metric variables, surrogate splits are performed (see Breiman [2017], Feelders [1999]).

Though, since there exist a lot of strategies to deal with missing values, we do not propose an own one for our ranking problems.

Assumption 17.2.1 (**Missingness assumption**). *We always assume that there are no non-structural missings in our data set, either since there never have been any or they have been imputed or our data set is the complete case data set.*

17.3 Structural missings

17.3.1 Related work

To the best of our knowledge, there is only very few work on structural missings, including the references already cited. Since standard imputation is obviously not meaningful (while Manewitsch [2013] found out that the imputation with zeroes can work in linear regression settings, though he does not provide any proof), the naïve way to deal with structural missings is still the complete case analysis, i.e., one has to delete all rows containing at least one missing. Clearly, such a procedure can result in deleting every row, especially in microarray data with a huge number of predictors but only very few observations.

Manewitsch [2013] also tried some kind of graphical modelling, but did not formulate it in a general setting, disregarding that his thesis is rather non-mathematical. There is one completely different work (Wagstaff [2004]) where missings are handled without imputing nor ignoring them in a clustering setting. The motivation was that imputed values are always handled as equal informative as the observed ones. Therefore, Wagstaff [2004] translated the missings into constraints for the distance minimization.

If there are only very few variables that lead to structural missings, simple stratification, for example by dividing a medical data set into two data sets containing only the men resp. the women can be a strategy. In the already given example, the column containing the pregnancy duration would just be deleted in the subset containing the men.

We propose two different strategies that are motivated by the fact that these structural missings should not enter model fitting, but without skipping too much valid data.

Remark 17.3.1. *We do not know how one would distinguish between structural and non-structural missings in practice. It seems to be the only possibility to detect structural missings by checking if there are columns that are likely to contain a structural missing, i.e., a missing due to the reported values of the other columns. But this would lead to a possible overestimation of the number of structural missings, depending on the design of the data acquisition process. If p.e. a female does not answer the question about pregnancy duration, it can still be a non-structural missing if she also had not reported her age, so if she is an adult but the non-reporting of the pregnancy duration had other reasons.*

In case of doubt, we think that it would be more reasonable to treat possibly structural missings always as structural missings to avoid imputing meaningless information.

17.3.2 Estimation using an asymptotic linear expansion

Let $\mathcal{D}^{CC}$ be the subset of the data set $\mathcal{D}$ consisting of the rows which do not contain any missings (probably after pre-processing like imputation where it was meaningful). Let n_{CC} be the number of rows of $\mathcal{D}^{CC}$. Now, assume that an empirical risk minimization problem which can be implicitly represented as a statistical functional by a corresponding Z-equation has to be solved. Let $\hat{\theta}_{n_{CC}}$ be an $n_{CC}^{-1/2}-$consistent estimator on the data $\mathcal{D}^{CC}$.

In other words, this is nothing but a coefficient fitted on the complete cases. Based on this estimator, we can compute the influence curve of the statistical functional for the estimator $\hat{\theta}_{n_{CC}}$, say, $\psi_{\hat{\theta}_{n_{CC}}}$. Consider the following question:

Is our estimator $\hat{\theta}_{n_{CC}}$ already "good enough" to use it as starting estimator for a $k-$Step?

As already pointed out in section 3.5, we need $n^{-1/2}-$consistency of our initial estimator to reasonably update it to a $k-$Step estimator. But if $\hat{\theta}_{n_{CC}}$ was already $n^{-1/2}-$consistent, we could use it as starting estimator for the whole data set $\mathcal{D}$. Thus, our One-Step estimator on $\mathcal{D}$ is just

$$S_n^1 = \hat{\theta}_{n_{CC}} + \frac{1}{n}\sum_i \psi_{\hat{\theta}_{n_{CC}}}(X_i, Y_i). \tag{17.3.1}$$

Since the influence curve measures the infinitesimal influence of an observation on the estimator, it is very intuitive that for any influence function ψ_θ, we just define

$$\psi_\theta(X_i, Y_i) := 0 \tag{17.3.2}$$

if Y_i is missing or if X_i contains at least one structural missing where the respective component of the coefficient θ is non-zero since a structural missing cannot have any influence on the model by its inherent nature. It can be regarded as an approach to **combine cell measures (see definition 14.1.1) with a influence curves**.

Remark 17.3.2. *Though it still depends on the data providing a reasonable number of complete cases, this version of One-Step estimators (the extension to k−Steps is straightforward) also incorporates information which has been gathered in the non-complete part of the data.*

Remark 17.3.3. *In fact, it has been revealed in an oral discussion with P. Ruckdeschel that in the case of k−Step estimators, it essentially suffices that the starting estimator is $n^{-\gamma}-$consistent with a $\gamma \in]0, 0.5]$. More precisely, one can show by incorporating a higher-order Taylor expansion that for any $0 < \gamma < 0.5$ there exists k such that this starting estimator is valid for a k−step construction.*

Theoretical results concerning the required quality of the initial estimator and reasonable ranges for the proportion of n_{CC} compared to n could be a subject of future research.

17.3.3 MissBoost?

Surprisingly, to the best of our knowledge, the implementation of L_2-Boosting seems not to be capable to deal with structural missings.

There is an input argument which indicates how missings are treated. Its default is `na.omit` which results in deleting all rows that contain missings. Clearly, if every row has at least one missing value, no model can be computed and the algorithm fails. Another possibility is the argument `na.pass` which means that the algorithm is not affected by missings. However, this can indeed result in only an intercept term being fitted if there are too many missings.

This issue as well as high-dimensional data with only a few observations that make a stratification invalid due to too little evidence on each stratum is the starting point for our idea of how sophisticated models may be fitted on data sets with structural missings.

Let us translate the zero imputation strategy in the context of L_2-Boosting which we already mentioned in the beginning of this section for linear regression. We replace all missings with zeroes and in each iteration of the Boosting algorithm, we compute a simple least squares model on each column. Effectively, we created a bias in the simple least squares models in all cases where zeroes have been imputed. This leads us to the questions:

If Boosting fits baselearners separately on each column, why should we treat structural missings at all?
More precisely, cannot we just ignore the missings separately in each column when fitting the component-wise base models?

We implement exactly this strategy and call the resulting algorithm **MissBoost** (see algorithm 21).

Clearly, just comparing the raw residual sums of squares would not be fair since the models are built based on different numbers of observations. Naïvely, one could think about scaling the loss corresponding to the simple model based on column j with $(\tilde{n}_j)^{-1}$ to make the losses comparable. However, this technique would also not provide a fair comparison.

More formally, we should compare the MSE, i.e., the expected squared loss. By the well-known bias-variance-decomposition of the MSE, we essentially need to adjust the component-wise losses such that the MSE's are comparable. Since deleting rows of the data set clearly also deletes the corresponding response components, the variance of the response vector depends on $\tilde{n}_j$ for the base model computed on column j, $j = 1, ..., p$, which would motivate a scaling with $\tilde{n}_j^2$. The other ingredient is the bias which in our algorithm can be seen as an **omitted variable bias**. This is true since we only compute the baselearner on one single column, ignoring valid and necessary information of the other columns. So, it is evident that one has to account for it.

```
Initialization: Data $(X, Y)$, step size $\kappa \in ]0, 1]$, number $m_{iter}$ of iterations and
  parameter vector $\hat{\beta}^{(0)} = 0_{p+1}$;
Generate a matrix $Z$ which has zeroes where $X$ has missing values;
Compute $\tilde{n}_j := \#\{i \mid X_{ij} \neq NA\})$ for all $j$;
Compute the offset $\bar{Y}$ and the residuals $r^{(0)} := Y - \bar{Y}$;
for $k = 1, ..., m_{iter}$ do
    for $j = 1, ..., p$ do
        Fit the current residual $r^{(k-1)}$ corresponding to the non-missing entries of
          column $j$ by a simple least squares regression model using the predictor
          variable $j$ and only the valid information;
        Compute an $\tilde{n}_j-$adjusted residual sum of squares;
    end
    Take the variable $\hat{j}_k$ whose model $\hat{\beta}_{\hat{j}_k} \in \mathbb{R}^{p+1}$ provides the smallest adjusted
      residual sum of squares;
    Update the model via $\hat{\beta}^{(k)} = \hat{\beta}^{(k)} + \kappa\hat{\beta}_{\hat{j}_k}$;
    Compute the current residuals $r^{(k)} = Y - Z\hat{\beta}^{(k)}$
end
```

Algorithm 21: MissBoost

Remark 17.3.4. *It is evident that we need a suitable row measure that accounts for missingness in the respective rows. Since the missingness structure is arbitrary, we cannot expect to get theoretical results that let us select a fixed initial row measure which is already appropriate to any given data set. Therefore, the required extension of the MissBoost algorithm is a* ***perfect candidate for our RCM framework****.*

Remark 17.3.5. *Although each simple least squares model is biased itself, L_2-Boosting after imputing the missings with zeroes indeed provides reliable models in simulations. By refitting models on the current residual, Boosting seems to gradually correct this bias.*

However, along with the qualitative weaknesses of illegally imputing structural missings and treating those imputations as on par with the real observations (by not accounting for the number of missings per column), simulations may do not reflect the missingness scheme adequately because when generating simulated data where the regressor matrix contains missings, one essentially replaces the missings with zeroes in the data generation step to compute the *response, replacing the imputed zeroes by NA's again afterwards. So, it is likely to anticipate that the imputation strategy works on such a data set.*

Remark 17.3.6. *For categorical variables, MissBoost deletes the NA's when fitting the base-learner based on the respective column. Since a simple least squares model which is based only on a categorical variable just computes the class-specific means, the coefficients are the same if we delete all missing cases as well as if we treat them as an extra category. But in fact, leaving standard multiple regression, we even get a difference in the coefficients if our model matrix includes interaction terms between a categorical and a metric variable. Thus, treating the NA's as extra category affects the computed coefficients in such a case!*

Remark 17.3.7. *It is straightforward to combine MissBoost with Column Measure Boosting. This could open an opportunity to perform* ***model selection by optimizing nearly arbitrary complicated loss functions on data sets with almost arbitrary missing structure****.*

Remark 17.3.8. *Note that when applying the GSE estimator of Danilov et al. [2012] (see also remark 14.2.1), no imputation is done anywhere, but nevertheless, Danilov et al. [2012] assume a MCAR missing scheme. Additionally, it is not designed for high-dimensional data as we already mentioned in remark 14.2.1.*

The question that we pose for future work therefore is the following:

Can we find a way based on empirical row and column measures to adjust the losses in MissBoost in order to adaptively get a fair comparison of the base-learners?

17.4 Are NA's just contamination?

We already mentioned that contamination balls (definition 3.3.1) are used to describe the outlier mechanism of contaminated data sets. Now, we address the question whether one could even identify missing values with such a contamination model.

Contamination and missings are artificially connected since one option after detecting an outlier in a data set is to replace it by "NA" (see for example Agostinelli et al. [2015], Leung et al. [2016] and Rousseeuw and Van Den Bossche [2018]). A strategy to proceed can be to impute these artificial missing values in order to get an observation that is more likely to stem from the true underlying distribution.

Assume for the moment that we face missings of type MCAR. Since, as the definition says, the missingness scheme is completely random, we can identify this with a cell-wise convex contamination model, i.e., we have contamination balls

$$U_r^{cell,NA} = \{Q \mid Q = \mathcal{L}(UX + (1-U)(\tilde{X})^{NA})\} \tag{17.4.1}$$

where in this special case of cell-wise contamination with U as in definition 3.3.3, $(\tilde{X})^{NA}$ is a matrix whose entries follow the Dirac distribution on the element NA which has to be included in the image space of the distribution of X to make the definition valid.

In other words, randomly replacing existing values by NA's with a given probability r is the same as putting cell-wise convex contamination on the data with contamination radius r. More precisely, we can think of contamination since there exist data which got lost, so the NA's are "wrong" values in the data set, i.e., the contaminated values.

Remark 17.4.1. *Note that Loh et al. [2018] already worked with this idea and embedded it in their theoretical results. They describe that missing values are treated as outliers, motivating to fill them by samples from any distribution which has to be the same for each row.*

However, transferring this concept to structural missings, the argumentation has to be reversed. The NA's that arise from structural absence are the "right" values. If they are replaced by any other object, of course including the zero, one can treat this disallowed imputation as contamination as well. In this framework, one maybe can still allow to replace NA's by zeroes to use standard robust learning algorithms if these imputations are appropriately downweighted. Note that Hubert et al. [2019] who provided a robust PCA algorithm that can simultaneously handle missings and cell-wise as well as case-wise outliers relied on an MCAR missingness scheme to allow imputations which is clearly not given in the presence of structural missings.

Remark 17.4.2. *More precisely, our proposal it to replace the structural missings indeed artificially by real numbers, but to account for this forbidden imputation. It is very important to emphasize that convex contamination balls do not incorporate any distance between the ideal and the contaminating distribution. In fact, the contamination radius r in standard convex contamination balls (cf. example 3.3.1) represents how many observations are expected to be contaminated where in the case of cell-wise outliers (cf. definition 3.3.3), it quantifies the expected fraction of contaminated cells. This is not true for other types of contamination balls* *(see [Rieder, 1994, Sec. 4.2]) where indeed a distance between distributions is represented.*

It the case of structural missings, it seems to be unevitable to incorporate two radii: One for the expected fraction of cells containing structural missings in the cell-wise convex contamination setting, the other one for representing some distance of the value "NA" to some imputed value. It would be interesting to know if the distance of the value "NA" to any real value is equal or if there are real values that are "closer " to NA than other ones. In this case, imputed values for structural missings that are less close to them need to be downweighted more severely in the spirit of robust statistics.

This leads to the following questions for future research:

Can existing robust estimation methods for cell-wise contamination be adapted for the case of structural missings?
How can a distance between missings and real values be defined?
Are the weights given to former NA-cells all equal resp. are the distances of an NA to any real value equal?

Chapter 18

Miscellanea

Our proposed algorithms are based on the assumption that the class of parametric models is essentially complete, i.e., that the reduction to the class of parametric models is not too restrictive. Another weakness is their non-robustness against wrong model assumptions.

The first section shortly addresses to techniques for non-parametric models. The second section just illustrates starting points to develop robust variants of SingBoost.

18.1 Beyond parametric models?

During parts III and IV of this thesis, we mostly considered a true linear parametric model, i.e.,

$$Y = X\beta + \epsilon.$$

However, in real-world applications, we more likely face nonlinear effects (see p.e. Friedman et al. [2001]), thus our model should be

$$Y = f(X) + \epsilon$$

where f is some nonparametric function (read $f(X)$ as the vector with components $f(X_i)$). More generally, a function f with the structure

$$f(X_i) := \sum_j f_j(X_{ij})$$

yields an additive model which provides far more flexibility than the standard linear model.

Note that SingBoost and therefore CMB-3S can also compute nonlinear models by using basis functions as already discussed in section 10.3. So far, these models are still parametric, i.e., we pick an element of the parametric function class

$$\mathcal{F}_\theta = \{f_\theta : \mathcal{X} \to \mathcal{Y} \mid \theta \in \Theta\}.$$

One standard approach to construct nonparametric models when optimizing the squared loss for regression is to use smoothing splines (Reinsch [1967]). In fact, a smoothing spline minimizes the residual sum of squares penalized by the bending of the function, i.e.,

$$\hat{f} = \underset{f}{\operatorname{argmin}} \left(\frac{1}{n} \sum_i f(X_i) + \lambda \int (f''(t))^2 dt \right). \tag{18.1.1}$$

Clearly, the penalty term should enforce smoothness of the resulting function. In the context of Boosting, we get nonparametric models when considering these smoothing splines as baselearners instead of simple linear models.

Sparse modelling with smoothing splines has already been done by Bühlmann and Yu in Bühlmann and Yu [2003] with L_2-Boosting, but with component-wise smoothing splines as baselearners instead of component-wise linear baselearners as we used in this thesis up to now. They were even able to prove minimax optimality of L_2-Boosting with smoothing splines in [Bühlmann and Yu, 2003, Thm. 3] and showed experimentally that this algorithm outperforms former state-of-the-art techniques like MARS.

Since L_2-Boosting with smoothing splines computes a smoothing spline solution separately for each column and chooses the base model which best approximates the current residual in each iteration, SingBoost and therefore Column Measure Boosting could be straightforwardly adapted to nonlinear models. In a former implementation, the function `mboost` of the package `mboost` provided the input argument `bss()` which corresponded to smoothing splines. This option is no longer available because the computation time can be reduced when using **P-Splines** instead of smoothing splines.

P-Splines have been introduced in Eilers and Marx [1996]. In the right-hand side of equation (18.1.1), replace $f(X_i)$ with $\sum_j a_j B_j(X_i)$ for coefficients a_j and B-Splines B_j (for B-Splines, see De Boor [1978]). The penalty term is similarly adapted. Then, Eilers and Marx [1996] point out that the computation of the penalty term is rather time-consuming and that the second derivative could even be replaced by the first or by higher-order derivatives. For any degree k, they propose a discrete approximation of the penalty term by replacing the integral of the $k-$th derivative by the difference of order k of contiguous B-Splines, i.e., our penalty term is

$$\lambda \sum_{j=k+1}^{n} (\Delta^k a_j)^2, \quad \Delta a_j = a_j - a_{j-1}, \Delta^2 a_j = \Delta\Delta a_j, ...$$

Schmid and Hothorn [2008] showed how P-Splines can be used as component-wise baselearners for L_2-Boosting and empirically proved the good performance of this algorithm.

Though, the pseudocode for SingBoost (algorithm 6) stays essentially the same as before (without the `LS` option). While we used the function `lm` to compute our linear baselearners, we now have to use the function `bbs` that is contained in the package `mboost`. For a given response vector `y` and the regressor `x`, we would write

```
bbs(x,...)$dpp(rep(1,n))$fit(y)$model
```

to get the coefficients where we can input various arguments to `bbs` like the position of the knots of the B-Splines, the degree of the splines or the order of the differences in the penalty term. If we can get the model matrix for the P-Spline (it is computed when running `bbs`, of course, though not returned to the best of our knowledge), then we can compute the fitted values and **choose the variable whose baselearner improves the aggregated model most, evaluated in $\tilde{L}$**.

Remark 18.1.1. *Note that the number of coefficients depends on the number of knots and the degree of the regression spline. The number of coefficients is p times the sum of the number of knots and the degree plus one, so we have to store all these coefficients which is realized as a list in the function `mboost`. Aside from these rather large storage costs, fitting a P-Spline is clearly much more expensive than fitting a simple least squares model. Therefore, it is not recommended to use this kind of SingBoost if there are other possibilities.*

Recall again that for ranking problems it is not necessary to predict the responses themselves. For example, consider the regression model

$$Y_i = \beta_0 + \beta X_i^2 + \epsilon_i$$

for regressor dimension $p = 1$ and error terms ϵ_i as usual. If $\mathcal{X} \subset \mathbb{R}_{\geq 0}$, then working with a linear model can also predict the correct ranking of the responses despite the values themselves would may be gravely mis-specified (cf. also figure 5.2).

Maybe one could even extend this idea. Suppose that the true underlying function is highly nonlinear. Then in order to reasonably solve the regression task, one needs to set up a nonlinear model. But in fact, for the ranking problem, it suffices to know the **true sign of each partial derivatives**. Therefore, there is no necessity to build a nonlinear model when concerning the ranking problem. Of course, a single linear model would be not appropriate but piece-wise linear models that capture the main slope structure. Concerning the example above, when the regressors take values in $\mathbb{R}_{\geq 0}$ as well as in $\mathbb{R}_{<0}$, one could use one linear model on the negative and one on the positive halfspace instead of a quadratic polynomial. Solving the ranking problem in this manner would **not only replace optimizing a pairwise loss function by optimizing a standard loss function but also reduce the degree of nonlinearity**. The latter may be interpreted as approximating a non-linear true relation of X and Y by a linear function in the same sense as in [Bühlmann and Van De Geer, 2011, Ch. 6.2.3].

Future researchers may find answers to the following problems:

If the true model is nonlinear, under which conditions a piece-wise linear model is still appropriate when considering a ranking problem?
What consequences would this approach have on model selection, i.e., can the true relevant variables still be selected when working with a simplified structure?
More precisely, are there analoga to singular parts, here in the sense of singular parts of the column measure w.r.t. a ranking loss function using a nonlinear resp. a linear model?

18.2 Robustifying SingBoost?

It is evident that standard L_2-Boosting (here w.l.o.g. again with component-wise linear baselearners) is highly non-robust due to the non-robustness of each baselearner.

This problem has already been addressed in Lutz et al. [2008] where numerous robustifications of L_2-Boosting have been developed that we briefly recapitulate.

Their first approach is to replace the non-robust squared loss by the Huber loss, resulting in fitting huberized residuals. Since this procedure clearly only guards against $y-$outliers (cf. section 3.4), they proposed downweighting leverage points, where their proposed weight w_{ij} for X_{ij} depends on an outlyingness quantification w.r.t. the corresponding column $X_{\cdot,j}$. These weights are optionally also used to correct for the residuals by severely downweighting only bad leverage points. However, this method would be inappropriate when concerning singular steps where a ranking loss has to be evaluated, since it is unclear how weights could be reasonably combined with such a loss function as it has been done in Lutz et al. [2008] for the residual sum of squares (cf. remark 12.6.3). The other quality measure in Lutz et al. [2008] which is the Q_n-estimator (Rousseeuw and Croux [1993]) on the respective column multiplied with the absolute value of the fitted coefficient also contradicts the intention of the singular steps.

Another proposal of Lutz et al. [2008] was to replace the simple least squares fits by robust fits, but they already mentioned the high computational costs. Additionally, they used the property of the simple least squares estimator that the slope is just the correlation of the regressor and the response, weighted with the quotient of the standard deviations. Thus, they replaced those ingredients by robust substitutes and recommend to choose the variable with the highest robust correlation with the current residual.

We think that the latter approaches better reflect the structure of SingBoost. According to [Lutz et al., 2008, Prop. 1], a Boosting procedure inherits the breakdown point of the baselearner. Though SingBoost is not a standard Boosting procedure, it is evident that even if the iterations w.r.t. the squared loss were robustified, the whole procedure gets worthless if the singular steps are sensitive to outliers. For example, let w.l.o.g. the first component of the original response be contaminated. If the current model is robust, we can expect the first component of the current residual to be rather large. Thus, the contaminated component ”passes” the robust iterations and lets the models in the singular steps become unreliable. It is easy to see that $x-$outliers affect the Boosting procedure even worse.

Summing up, robustifying SingBoost and therefore Column Measure Boosting requires robust baselearners, but additionally we need a variable selection criterion that still depends on the loss function $\tilde{L}$. Maybe a direction of future research can use the following idea.

Replace L_2-Boosting by one of the variants proposed by Lutz et al. [2008] which are both robust against $y-$ and $x-$outliers in the first $(M-1)$ iterations. Use the robust current estimate $\hat{\beta}^{(M-1)}$ as initial estimator in the $M-$th iteration. If it would be reasonable to assume that this estimator is already well enough, one could construct a One-Step estimator based on each column separately. To get the required influence curve, we mimic the idea of Öllerer et al. [2015] where they computed the influence curve of the Lasso solved by coordinate descent. Since this algorithm is based on simple Lassos on the columns of the regressor matrix, they write this simple Lasso estimator in the usual form, replacing the original response by the partial residuals implying that the influence function always depends on the previous coefficients.

Similarly, we may compute each component of our required influence function by

$$(\mathrm{IC}((x,y),\hat{\beta}^{(M-1)},F_{\hat{\beta}^{(M-1)}}))_j = \frac{x\left(y-\left[\sum_k x_k\hat{\beta}_k^{(M-1)}\right]-x_j\hat{\beta}_j^{(M-1)}\right)}{\mathbb{E}_{F_{\hat{\beta}^{(M-1)}}}[x^2]} \tag{18.2.1}$$

where we denote by $F_{\hat{\beta}^{(M-1)}}$ the regression model corresponding to $\hat{\beta}^{(M-1)}$ in the sense of (4.4.2). Roughly spoken, instead of measuring the influence of a whole new observation on the estimator, we progress as in Boosting or in the coordinate descent algorithm for the Lasso (Friedman et al. [2007], Öllerer et al. [2015]) and treat the influences of each column of a new regressor separately. Since the full residuals are fitted in each Boosting iteration, it is more appropriate in our case to include full residuals rather than partial residuals.

Remark 18.2.1. *Since the influence curve (18.2.1) depends on all previously fitted coefficients through the residuals (and also on the step size κ), it can be regarded as an influence curve for L_2-Boosting.*

```
Initialization: Data $(X,Y)$, step size $\kappa \in ]0,1]$, number $m_{iter}$ of iterations, number
  $M \leq m_{iter}$ (each $M-$th iteration is a singular iteration) and target loss $\tilde{L}$ (as part of
  a family object singfamily);
Set
    $$\text{runs} = \left\lfloor \frac{m_{iter}}{M} \right\rfloor$$
  Define $\hat{f}^{(0)} := 0$, thus $r^{(0)} = Y$;
for $k = 1, ..., \text{runs}$ do
    Perform $(M-1)$ steps of a robust version of $L_2-$Boosting starting with the
      residuals w.r.t. the model $\hat{f}^{((k-1)M)}$;
    Get the model $\hat{f}^{((k-1)M+M-1)}$;
    Compute the component-wise influence function in (18.2.1) w.r.t. the joint
      distribution corresponding to $\hat{f}^{((k-1)M+M-1)}$;
    Compute an optimally-robust version $\tilde{\psi}$;
    Compare the One-Steps in (18.2.2) w.r.t. $\tilde{L}$;
    Get the weak model $\hat{g}^{(kM)}$ and update the model via
        $$\hat{f}^{(kM)} = \hat{f}^{((k-1)M+M-1)} + \kappa \hat{g}^{(kM)}$$
end
```

Algorithm 22: RobSingBoost

Of course, this standard influence curve is unbounded. As we already mentioned in section 3.5, there exist techniques to find optimally-robust influence curves. This is also possible for influence curves in regression, see [Rieder, 1994, Sec. 7] and [Kohl, 2005, Sec. 7]. Hence, let $\tilde{\psi}$ be an optimally-robust influence curve (depending on the chosen optimization problem and the contamination model). Then we propose to proceed computing the One-Steps

$$\hat{\beta}_j^1 := \hat{\beta}_j^{(M-1)} + \frac{1}{n} \sum_i \tilde{\psi}_j(X_i, r_i^{(M-1)}). \tag{18.2.2}$$

In the last step, we investigate the performance w.r.t. loss $\tilde{L}$ and update the component of the coefficient that improves it most. The residuals serve as the responses for the subsequent robust L_2-Boosting iteration and so forth.

Note that we do not have robustified Column Measure Boosting yet. If we assume that there are contaminated observations in the training data, we also would get contaminated test data since the training observations for one SingBoost algorithm may be the validation observations of another one. Without referring to all existing techniques of outlier detection again (see for instance Filzmoser et al. [2008]), we just assume here that we have separated a test set before drawing subsamples to get training data and that this test set is already clean. Then we get a robust version of Column Measure Boosting (say **Robust Column Measure Boosting; RCMB**). Clearly, handing over the respective data to each core, a parallelized version of RCMB is designed straightforwardly.

Questions for future work include:

Is the definition of the influence curve in (18.2.1) allowed?
Is it meaningful to compute One-Steps as in (18.2.2)?
Is the proposed Robust Column Measure Boosting feasible concerning computational cost?
How efficient are the robustifications in the ideal model?

Chapter 19

Summarizing conclusion

We list our main theoretical and conceptual contributions in the following two sections. The third section is a more detailed overview of the power of our implementations.

19.1 Theoretical contributions of this work

The first major theoretical contribution of this work was the extension of the theory of asymptotic linearity of M-estimators to the case of regularized M-estimators for both differentiable (theorem 4.5.2) and non-differentiable (theorem 4.5.3) regularization terms. By specifying the theoretical requirements, we identified the Lasso, the elastic net and the Adaptive Lasso as regularized M-estimators that can be asymptotically linearly expanded. We additionally pointed out that our results can even handle the case of data-driven penalty parameters which manifests itself in a sequence of penalty parameters which depend on the sample size and which converge to zero as the sample size grows to infinity.

We provided minor results where we computed influence curves for the ranking problems via ordered discrete choice models (section 6.2) and where we proved a Riesz representation result for functionals corresponding to asymptotically linear estimators (theorem 9.3.1). The technically very simple lemma 6.1.1 connected the evaluation of the hard ranking loss with the evaluation of Kendall's Tau in the absence of ties in a bijective way which indeed has a great impact for practically working with ranking losses of this kind, significantly reducing the computational time for large n.

Furthermore, the Expected One-Step estimator extended the connection of the theory of asymptotically linear estimators with variable selection through partial influence curves (section 3.5) to a simultaneous treatment of influence curves that are related to the rows with empirical column measures which correspond to the columns of a data matrix (section 9.8).

The major theoretical statement of this work is that SingBoost keeps the estimation consistency and the prediction consistency of L_2-Boosting when assuming a Corr-min condition in the sense that each singular iteration selects a variable whose corresponding column is at least sufficiently enough correlated with the current residual compared to the correlation of the most correlated column with the current residual (theorems 10.5.1+10.5.2). These theorems guarantee that SingBoost asymptotically computes the true coefficients, even if the variables selected in the singular steps are not the same as L_2-Boosting had selected which is exactly the theoretical property that we need to justify our strategy. Moreover, in an asymptotic sense, the number of predictors is allowed to grow nearly exponentially fast w.r.t. the number of observations.

In section 13.2.2, we already were able to show that a MultiSingBoost, i.e., a SingBoost procedure for multivariate responses, would also keep the estimation consistency of multivariate L_2-Boosting under an additional Corr-min condition without yet having constructed a worked-out algorithm.

Finally, to reasonably compare different ranking models, the elicitation property of the corresponding statistical ranking functionals is required. We showed that both for univariate and uncorrelated multivariate responses, the hard ranking functionals are indeed elicitable (lemmas 7.1.1 and 13.3.2). In the case of ties in the response vector which causes multiple true rankings, we proved that the corresponding ranking functional for univariate responses is strongly (exhaustively) elicitable (lemma 7.2.1). When having multiple responses which are uncorrelated, we showed that the corresponding multivariate hard ranking functional satisfies the property of strong (exhaustive) $k-$elicitability (lemma 13.3.3).

19.2 Conceptual contributions of this work

The first main conceptual contribution was to show that Gradient Boosting for the continuous ranking problem cannot be achieved by simply replacing the hard ranking loss with a (piece-wise) convex and (piece-wise) differentiable pair-wise surrogate loss (see chapter 8). This demonstrated a further difference between the continuous and the bipartite ranking problem since the latter can indeed be solved by optimizing a suitable pair-wise surrogate loss.

The introduction of the row measure and the column measure in definitions 9.5.1 and 9.6.1 formed the root for a new view on variable selection that initially enabled us to treat traditional learning procedures without model selection and variable selection procedures as special cases of our column measure framework. We provided a combination of the One-Step estimators with the column measure, namely the Expected One-Step (section 9.8) which unified the column-based variable importances with the row-based influence curves. Together with our theoretical results on asymptotic linear expansions of regularized M-estimators, even estimators like the Lasso or the Adaptive Lasso can be identified as aggregation procedures that can be divided into row-wise and column-wise aggregation.

The column measure framework revealed a great issue when performing model selection w.r.t. different loss functions which we mathematically identified with singular parts between column measures (definition 10.1.1) that we postulated to exist. That warns us to be very cautious when trying to perform model selection for some loss function $\tilde{L}$ by using an algorithm that is tailored to another loss function L.

Identifying each iteration of component-wise Boosting as a rejection step itself, we proposed the algorithm SingBoost (algorithm 6) that includes singular steps where a linear baselearner is evaluated in the target loss $\tilde{L}$, so that we get the chance to select variables from potential singular parts. We provided arguments that raw SingBoost models are either not sufficient for stable models including potential singular parts or that the column measure w.r.t. $\tilde{L}$ will not be sufficiently adapted. Therefore, we proposed Column Measure Boosting (algorithm 7) that leads to a stabilization of such a potential singular part for further analysis. We also provided ideas to overcome computationally intensive evaluations in section 11.5.

We learned that a sparse learning technique, performed on different subsamples of the data, indeed also supplies an empirical row measure, see equation (12.2.2). We finally proposed a loss-based Stability Selection which we combined with the CMB algorithm resulting in the CMB-3S algorithm (algorithm 12). As a simple extension to immunize against a given partition of the data, we provided the CV.CMB-3S algorithm (algorithm 16 and figure 12.3) which can be used to compute an ultra-stable sparse column measure (equation (12.4.1)). Note that when applying one of these algorithms to a ranking problem, we essentially replace a pair-wise loss function by a univariate loss function.

One of our major conceptual contributions was the simultaneous treatment of row and column measures which led to RCM-pairs which assign at least importances to cells of the regressor matrix. In section 12.5, we identified sparsity as a property of empirical column measures associated with the column measure framework whereas stability of empirical column measures is essentially related to the row measure framework. The row column measure framework therefore unifies sparsity, stability and even robustness which eventually is connected to the interplay of row and column measures. We indeed embedded inherently different algorithms like Lasso, BlockForest, SLTS and CMB-3S into this framework.

We also provided a first approach to achieve a parametric consensus ranking model (section 13.1) and a first idea how to extend SingBoost to multivariate responses where we highlighted that the extension of our row column measure framework to the case of multivariate responses indeed produces new singular parts (section 13.2.2). To describe the sparse estimation of covariance or precision matrices, we introduced the cell measure (section 14.1).

One of our most striking conceptual contributions was to embed outlier detection algorithms like the DDC prodecure and robust algorithms like the Fast-MCD and SLTS into our RCM framework. Even more astoundingly, we were able to rewrite these algorithms in Boosting form (sections 14.2 and 14.3). We provided a first idea for a Generalized L_2-Boosting algorithm (algorithm 20) and suggested a Stability Selection for the rows in combination with Stability Selection for the columns in order to simultaneously account for sparsity, stability and robustness (section 14.4). We assume that if one succeeded in generalizing the RCM framework for univariate responses to multivariate responses, we had tools to describe and to handle any possible peculiarity or heterogeneity in the data.

Without yet having worked-out theory or algorithms, we provided first ideas how to use Boosting or asymptotically linear estimators to handle structural missings (chapter 17) and how SingBoost may be robustified (section 18.2). Furthermore, we discussed how CMB-3S could be extended to nonparametric baselearners and if it would be necessary to model nonlinearities when concerning ranking problems (section 18.1).

19.3 Algorithmic contributions of this work

We implemented the SingBoost algorithm (algorithm 6) as the `R`$-$function `singboost`, the CMB algorithm (algorithm 7) as `cmb`, the CMB-3S algorithm (see algorithms 12, 15) as `CMB3S` and finally the CV.CMB-3S procedure (see algorithm 16 and figure 12.3) as `CV.CMB3S`.

Our `CV.CMB3S` function uses three `mboost`-type families instead of one which is the case for standard Boosting algorithms and for SingBoost.

The `singfam` family is, as its name suggests, representing the loss function $\tilde{L}^{sing}$ that appears in the singular steps of SingBoost. The `evalfam` family manifests itself in the Column Measure Boosting as well as in the Stability Selection. Concretely, the evaluation loss function $\tilde{L}^{eval}$ is used to compute the out-of-sample performance of the SingBoost models in order to determine the best $\lceil \alpha B^{sing} \rceil$ SingBoost models to get appropriately aggregated empirical row and column measures. Furthermore, the sparse and stable model selection needs the computation of the out-of-sample performance on a validation set $\mathcal{D}^{valid}$ to choose either the best threshold from the grid π_{grid} or the best number of variables from the grid q_{grid}. This is also done by using $\tilde{L}^{eval}$. Finally, $\tilde{L}^{target}$ which is represented by the `targetfam` object delivers us the cross-validated test loss of our final model.

This implementation gives us the opportunity of a very flexible usage of our `CV.CMB3S` implementation. Inserting different family objects at each step or special combinations of hyperparameter settings lead to the opportunity to easily compare CMB-3S with classical approaches.

Indeed, we can perform an analog of the classical Stability Selection. Setting $n_{cmb} = n_{train}$ means that no CMB-type subsampling is necessary to get an aggregated column measure, but instead, we outsource the subsampling procedure to the Column Measure Boosting where we draw subsamples of size $n_{sing} = 0.5n_{train}$ as suggested in Meinshausen and Bühlmann [2010]. The number B^{sing} is exactly the number of subsamples that have to be generated for CMB. Setting `sing=F` and `singfam=Gaussian()` means that we indeed perform L_2-Boosting instead of SingBoost so that M does not influence the result. `m_iter` and `kap` are the usual hyperparameters of L_2-Boosting. Setting $\alpha = 1$ implies that the selection frequencies of all fitted models are used for aggregation. Therefore, it is indifferent how to choose `evalfam` here. Performing a Stability Selection in the sense of Bühlmann and Meinshausen would be realized if one just proposed an one-elemental grid, so we essentially choose a fixed π_{thr} instead of optimizing it via a grid search, so `evalfam` has again no influence. The reason why we just "nearly" perform standard Stability Selection with L_2-Boosting is that Hofner et al. [2015] recommend to let the Boosting procedure only run as long as q variables have been selected which is not the case for the given settings in our algorithm.

Much more relevant is the application of our loss-based Stability Selection for any existing Boosting procedure (for regression) that is possible by just using the argument `evalfam` which has to be the same `family` object as `singfam`, and defining some grid with more than one element to perform the suggested grid search. By changing the argument `singfam=Gaussian()` to any other yet existing `mboost` family object (until now, just for regression) representing some loss L, we aggregate the respective $L-$Boosting models instead of L_2-Boosting models. This may sound contradictory, but together with the arguments `sing=F` and `LS=F` (both defaults), we run `glmboost` with the respective `singfam` object in `cmb` and therefore get an empirical `glmboost` column measure which enters our Stability Selection.

As (partially) demonstrated in part VI, our algorithm can easily cope with categorical variables, interaction terms or basis functions since we just need to insert a model matrix and not some formula argument, so generating the model matrix with the command `model.matrix` already suffices as pre-processing step to apply our algorithms.

Apart from these functionalities, we provided the function `path.singboost` that enables us to draw coefficient paths for SingBoost models by applying the function `singboost.plot`.

More detailed, illustrative examples of our algorithms can be found in part VI.

Let us highlight again that our loss-based Stability Selection can be applied to models based on high-dimensional and noisy data where Hofner's Stability Selection is no longer meaningful and that the resulting models significantly outperform the standard Boosting models (see chapter 16) and that it can easily be applied to very high-dimensional data (see section 15.4).

Admittedly, the computational time is very high once we invoke singular iterations. This is a direct consequence of the pure $\mathsf{R}-$implementation and no weakness of the underlying SingBoost algorithm since its theoretical complexity essentially just differs from the complexity of L_2-Boosting by an additional factor $\ln(n)$ in cases where the loss function needs $\mathcal{O}(n\ln(n))$ operations to be evaluated based on two vectors of length n, and equals the complexity of L_2-Boosting in cases of loss functions that can be evaluated in $\mathcal{O}(n)$ steps, see again lemma 10.3.1 and remark 10.3.15. Invoking C and FORTRAN code in a future implementation can make our algorithms competetive with Boosting and Stability Selection from `mboost` and `stabs`.

Appendix

A.1 Coercivity

An important property of real-valued functions that is used to show the existence of minimizers is coercivity or coerciveness. We note that the definition of coercivity is not unique in literature. We use the following definition of Werner [2006].

Definition A.1.1. *Let X be a normed space. A function $f : X \to \mathbb{R}$ is* ***coercive*** *if*

$$\lim_{||x|| \to \infty} (f(x)) = \infty.$$

In fact, Rockafellar and Wets [1998] define coercivity by

$$\liminf_{||x|| \to \infty} \left(\frac{f(x)}{||x||} \right) = \infty$$

with an additional boundedness condition where a related property (with the limit inferior replaced by the limit and allowing f to map into $\bar{\mathbb{R}}$) is referred to as **supercoercivity** in [Bauschke and Combettes, 2011, Def. 11.11].

Roughly spoken, a function is coercive if the function value grows to infinity when the norm of the input argument grows to infinity. This property is used to guarantee the existence of a minimum in the interior domain. More precisely, one can state that a coercive and continuous function $f : \mathbb{R}^n \to \mathbb{R}$ has a minimizer (cf. Butenko and Pardalos [2014]). This also holds if the continuity assumption is weakened to lower semi-continuity for convex f (cf. [Werner, 2006, Satz III.5.8]).

Note that the definition can be reversed in the sense that the function tends to $-\infty$ to show the existence of maximizers (cf. Levitin and Tichatschke [1998]).

The notion of coercivity can be concretized to certain arguments of f. For the following lemma, we refer to Evgrafov and Patriksson [2004] and Levitin and Tichatschke [1998].

Lemma A.1.1. *Let $f : \mathcal{X} \times \mathcal{Y} \times \Theta \to \mathbb{R}$ be continuous, where $\mathcal{X} \subset \mathbb{R}^n$, $\mathcal{Y} \subset \mathbb{R}^m$, $\Theta \subset \mathbb{R}^k$. Define $\Xi(x,y) := \operatorname{argmin}_\theta(f(x,y,\theta))$. If f is **coercive w.r.t.** θ, i.e., the sets*

$$\{\theta \in \Theta \mid f(x,y,\theta) \leq c\}$$

are bounded for all $c \in \mathbb{R}$ for every $x \in \mathcal{X}$, $y \in \mathcal{Y}$, then $\min_\theta(f(x,y,\theta)) > -\infty$ and $\Xi(x,y)$ is nonempty and compact for any x, y.

For a generalization to families of functions, we refer to the following definition of equi-coercivity of Dal Maso [2012].

Definition A.1.2. *Let X be a topological space and let $f_n : X \to \bar{\mathbb{R}}$. Then the family $\{f_n \mid n \in \mathbb{N}\}$ is said to be **equi-coercive** if for any $c \in \mathbb{R}$ there exists $K_c \subset\subset X$ such that $\{x \mid f_n(x) \leq c\} \subset K_c$ for every n.*

This definition is equivalent to the existence of a lower semi-continuous, coercive function $\Psi : X \to \mathbb{R}$ such that $f_n \geq \Psi$ for all n which is used in Le et al. [2017].

A.2 Uniform integrability

See e.g. Rieder [1994] for the following definition of uniform integrability of a sequence of random variables.

Definition A.2.1. *Let $(\Omega_n, \mathcal{A}_n, P_n)$ be probability spaces for $n \geq 0$. A sequence $X_n : (\Omega_n, \mathcal{A}_n, P_n) \to (\mathbb{R}^p, \mathcal{B}^p)$ of random variables is **uniformly integrable** if*

$$\lim_{c \to \infty}\left(\sup_n\left(\int_{\{|X_n| \geq c\}} |X_n| dP_n\right)\right) = 0.$$

When investigating the conditions that are necessary to interchange the derivative and the integral, we make use of the theorem of Lebesgue-Vitali (cf. [Bogachev, 2007a, Thm. 4.5.4]) and the Dunford-Pettis criterion (cf. [Attouch et al., 2014, Thm. 2.4.5]).

Theorem A.2.1 (**Lebesgue-Vitali**). *Let $(\Omega, \mathcal{A}, \mu)$ be a finite measure space. Let f be $\mu-$measurable and let the family $\{f_n\}$ consist of $\mu-$integrable functions. Then $f_n \to f$ in $L^1(\mu)$ with integrable limit f if and only if $\{f_n\}$ is uniformly integrable and $f_n \to f$ in measure.*

Theorem A.2.2 (**Dunford-Pettis criterion**). *Let $(\Omega, \mathcal{A}, \mu)$ be a probability space and let $\mathcal{F} \subset L^1(\mu)$ be bounded. Then $\mathcal{F}$ is uniformly integrable if and only if $\mathcal{F}$ is a relatively compact subset of $L^1(\mu)$ w.r.t. the weak topology, i.e., if the closure of $\mathcal{F}$ is compact in $L^1(\mu)$*

Note that although the last theorem is often called ”Dunford-Pettis theorem” in literature, we referred to it as the ”Dunford-Pettis criterion” to avoid confusion with the Dunford-Pettis theorem that characterizes the spaces that provide the Radon-Nikodym property (cf. e.g. Dunford and Pettis [1940] or [Diestel and Uhl, 1977, III.Lemma 9]).

A.3 Tools from measure theory

We start with the definition of signed measures (see e.g. Elstrodt [2006]).

Definition A.3.1. *Let $(\Omega, \mathcal{A})$ be a measurable space. A map $\mu : \mathcal{A} \to \mathbb{R}$ is called a **signed measure** if μ is sigma-additive and if $\mu(\emptyset) = 0$.*

Remark A.3.1. *In fact, a signed measure μ can be represented by two measures μ^-, μ^+ by setting $\mu = \mu^+ - \mu^-$ and can take values in $[-\infty, \infty[$ or $]-\infty, \infty]$.*

The following definition generalizes $\mathbb{R}-$valued measures to vector-valued measures, see e.g. Diestel and Uhl [1977] or Blasco and Gregori [2002].

Definition A.3.2. *Let $(\Omega, \mathcal{A})$ be a measurable space and let $\mathcal{X}$ be a Banach space. Then a map $\mu : \mathcal{A} \to \mathcal{X}$ is a **countably additive vector measure** if for any countable index set J it holds that*

$$\mu\left(\bigcup_{j \in J} A_j\right) = \sum_j \mu(A_j)$$

for disjoint $A_j \in \mathcal{A}$ for all j. For $A \in \mathcal{A}$ let Π be the set consisting of all partitions of A. Then the function

$$|\mu|(A) := \sup_{\pi \in \Pi} \left(\sum_{B \in \pi} ||\mu(B)|| \right)$$

is the ***variation of*** *μ. In the case $|\mu|(\Omega) < \infty$, the vector measure μ is said to be of* ***bounded variation****.*

The following definitions of [Brooks and Lewis, 1974, Def. 2.3+Def. 2.6] are needed for a special Riesz representation theorem. See also definition A.4.3 for the notation.

Definition A.3.3. *Let $\mathcal{X}$, $\mathcal{Y}$ be locally convex spaces. Let $(\Omega, \mathcal{A})$ be a measurable space and let $\rho : \mathcal{A} \to \mathcal{C}_f(\mathcal{X}, \mathcal{Y})$ be finitely additive. Then, for any $A \in \mathcal{A}$, the $(p, q)-$**semi-variation of** ρ is defined by*

$$\sup_{\pi \in \Pi,\ ||x_i||_{\mathcal{X}} \leq 1} \left(\left|\left| \sum_{B \in \pi} \rho(B) x_i \right|\right|_{\mathcal{Y}} \right)$$

where Π denotes the set of all partitions of A and $||\cdot||_{\mathcal{X}}$, $||\cdot||_{\mathcal{Y}}$ are a semi-norms on $\mathcal{X}$ resp. $\mathcal{Y}$.

Definition A.3.4. *Let $(\Omega, \mathcal{A})$ be a measurable space and let E, F be Banach spaces. Then a finitely additive function $\rho : \mathcal{A} \to \mathcal{C}_f(E, F'')$ with finite $(p, q)-$semi-variation is* ***weakly regular*** *if for any $x \in E$, $i \in F'$ the function*

$$\rho_{(x,i)}(\cdot) := \langle m(\cdot)x, i \rangle$$

is a finite Radon measure.

There are certain types of convergence of functions with respect to a measure. Egorov's theorem below connects pointwise convergence with almost uniform convergence ([Bogachev, 2007a, Thm. 2.2.1]). Thereafter, we state the well-known Radon-Nikodym theorem as it is presented in [Bauer, 1992, Satz 17.10].

Theorem A.3.1 (Egorov). *Let $(\Omega, \mathcal{A}, \mu)$ be a finite measure space with non-negative measure μ. Let $\{f_n \mid n \in \mathbb{N}\}$ be a family of $\mu-$measurable functions. If the pointwise convergence $f_n \to f$ for some $\mu-$measurable function f holds $\mu-$almost everywhere, then the sequence $(f_n)_n$ converges to f $\mu-$almost uniformly.*

Theorem A.3.2 (**Radon-Nikodym**). *Let $(\Omega, \mathcal{A})$ be a measurable space. Let μ and ρ be measures on $\mathcal{A}$ where μ is sigma-finite. Then $\rho \ll \mu$ if and only if there exists $f \in L^1(\mu)$ such that*

$$\rho(A) = \int_A f d\mu$$

for any $A \in \mathcal{A}$.

Remark A.3.2. *Note that absolute continuity of a measure ρ on $\mathcal{A}$ w.r.t. a measure μ on $\mathcal{A}$ is usually defined in the sense that ρ is dominated by μ, i.e., $\rho(A) = 0$ if $\mu(A) = 0$ for any $A \in \mathcal{A}$. By the Radon-Nikodym theorem, this is in fact equivalent to the existence of a density f of ρ w.r.t. μ.*

For the case that domination does not hold, there exists the following decomposition (see e.g. Elstrodt [2006]). The definition below of absolute continuity in more than one dimension is borrowed from Pupashenko et al. [2015].

Theorem A.3.3 (**Lebesgue decomposition**). *Let $(\Omega, \mathcal{A})$ be a measurable space and let ρ be a $\sigma-$finite signed measure and μ be a $\sigma-$finite measure on $\mathcal{A}$. Then there exists a decomposition*

$$\rho = \rho^c + \rho^s$$

where ρ^c, ρ^s are signed measures of $\mathcal{A}$ with $\rho^c \ll \mu$ and $\rho^s \perp \mu$.

Definition A.3.5. *A function $f : \mathbb{R}^p \to \mathbb{R}$ is* ***absolutely continuous in p dimensions*** *if for any x, $y \in \mathbb{R}^p$, the function*

$$G : [0, 1] \to \mathbb{R}, \quad G(s) := f(x + s(y - x))$$

is absolutely continuous in the usual sense.

Note that there exists another definition that comes from Ruckdeschel [2010b] which is more evolved since it respects Lebesgue zero sets. The following definition of continuity sets is borrowed from [Elstrodt, 2006, Def. 4.9].

Definition A.3.6. *Let Z be a topological space and let μ be a Borel measure on Z. Then $B \in \mathbb{B}(Z)$ is called $\mu-$**continuity set** if $\mu(\bar{B} \setminus B^\circ) = 0$, i.e., if the boundary of B is a $\mu-$null set.*

In an even more abstract setting, Le Cam (Le Cam [1986]) defines when two experiments which are understood as maps from a set Θ of "theories" into a so-called $L-$space ([Le Cam, 1986, p. 4]) such that each θ is mapped to some positive P_θ with unit norm ([Le Cam, 1986, Def. 1]). We borrow the following definitions from [Le Cam, 1986, Ch. 2] and [Le Cam, 1986, Ch. 6].

Definition A.3.7. *Let E, F be two experiments defined on the same set Θ where E maps each θ into some P_θ and F maps each θ into some Q_θ.*

a) *The **deficiency** of the two experiments is given by*

$$\delta(E,F) = \inf_T \left(\sup_\theta \left(\frac{1}{2} ||Q_\theta - TP_\theta|| \right) \right)$$

*where T is a **transition**, i.e., a norm-preserving, positive linear map.*

b) *The experiments E and F are **equivalent** if both $\delta(E,F) = 0$ and $\delta(F,E) = 0$.*

c) *An experiment $E = \{P_\theta \mid \theta \in \Theta\}$ with $P_\theta \in L$ for some $L-$space L is **dominated** by $\lambda \in L$ if each P_θ is dominated by λ.*

In other words, two experiments are equivalent if there exists a suitable transition that linearly transforms the probability measures P_θ of experiment E into the counterparts Q_θ for experiment F.

A.4 Topological spaces

The following definition ([Rudin, 1987, Def. 2.3] and [Brooks and Lewis, 1974, Lemma 2.3]) concerns about general topological spaces. The definition of continuity in the compact-open topology is also borrowed from Brooks and Lewis [1974].

Definition A.4.1. ***a)*** *A topological space Z is called a **Hausdorff space** if for any x, $y \in Z$ such that $x \neq y$ there exist neighborhoods U_x of x and U_y of y satisfying $U_x \cap U_y = \emptyset$.*
b) *A topological space Z is **locally compact** if every $x \in Z$ has a neighborhood whose closure is compact.*
c) *Let E be a Banach space and Z be a Hausdorff space. Then Z is called an **S-space** if $\mathcal{C}(Z,E)|_K = \mathcal{C}(K,E)$ for each $K \subset\subset Z$ or equivalently, if Z is separated by $\mathcal{C}(Z)$.*

Definition A.4.2. *Let E, F be Banach spaces and let Z be a Hausdorff space. Then $T : \mathcal{C}(Z,E) \to F$ is* ***continuous in the compact-open topology*** *if there exists a constant $M \geq 0$ and $K \subset\subset Z$ such that for any $f \in \mathcal{C}(Z,E)$ it holds that*

$$||Tf||_F \leq M||f||_K$$

for $||f||_K := \sup\{||f(t)||_E \mid t \in K\}$.

When we talk about spaces of certain continuous functions, we will always have the following notation (cf. [Rudin, 1987, Def. 3.16], see [Rudin, 1987, Thm. 3.17])).

Definition A.4.3. ***a)*** *The space $\mathcal{C}_f(Z)$ contains all functions on Z that* ***vanish at infinity****, i.e., for any $\epsilon > 0$ there exists a compact $K \subset\subset Z$ such that $|f(x)| < \epsilon$ for every $x \notin K$.*
b) *The space $\mathcal{C}_0(Z)$ contains all functions f on Z with a* ***compact support****, i.e., $\{z \in Z \mid f(z) \neq 0\} \subset K \subset\subset Z$.*

Remark A.4.1. *A typical example of a function in $\mathcal{C}_f(\mathbb{R})$ is the density of a standard normal distribution which obviously does not have compact support. In fact, $\mathcal{C}_f(Z)$ is the completion of $\mathcal{C}_0(Z)$ with respect to the supremum norm if Z is a locally compact Hausdorff space.*

A.5 Riesz representation theorems and dual spaces

Definition A.5.1. *A functional $T : X \to \mathbb{R}$ is called* ***positive*** *if $Tf \geq 0$ for any $f \geq 0$.*

The classical Riesz representaton theorem (cf. [Rudin, 1987, Thm. 6.16]) connects L_p−spaces with their dual spaces L_q for $q = \frac{p}{p-1}$. However, Riesz representation theorems that are used in measure theory are more likely to provide representing measures than representing functions. A standard example is the following one of [Rudin, 1987, Thm. 2.14], [Elstrodt, 2006, 2.5]. Compare with definitions A.4.1, A.5.1 and A.4.3 for the notation.

Theorem A.5.1. *Let Z be a locally compact Hausdorff space and let $T : C_0(Z) \to \mathbb{R}$ be a positive linear functional. Then there exists a unique Radon measure μ such that*

$$Tf = \int_Z f d\mu$$

for any $f \in C_0(Z)$.

Riesz representation theorems of this kind identify the duals of spaces of continuous functions with spaces of certain Radon measures. So, the previous theorem relates the dual space $(C_0(Z))^*$ with the space of all Radon measures. Analogously, for a locally compact Hausdorff space Z, the space $(C_f(Z))^*$ can be shown to be isometrically isomorphic to the space of all finite Radon measures (cf. [Elstrodt, 2006, 2.23+2.26]).

Since we are interested in vector-valued functionals, we need multidimensional representation theorems. In fact, we borrow the following one from [Brooks and Lewis, 1974, Thm. 2.7], see also definitions A.4.2 A.3.4, A.4.1.

Theorem A.5.2. *Let Z be an S-space and let E, F be Banach spaces. Let $T : \mathcal{C}(Z, E) \to F$ (where $\mathcal{C}(Z, E)$ is equipped with the compact-open topology) be linear and continuous. Then there exists a unique, weakly regular function $\rho : \mathbb{B}_Z \to \mathcal{C}_f(E, F'')$ such that*

$$Tf = \int_Z f d\rho$$

for any $f \in \mathcal{C}(Z, E)$ where $\mathbb{B}_Z$ denotes the sigma algebra of all Borel sets on Z.

A.6 Distribution theory

The following definitions and properties concerning weak derivatives can be found in standard textbooks of functional analysis or partial differential equations, e.g. Werner [2006].

Definition A.6.1. *Let $A \subset \mathbb{R}^n$ open and let $f \in L_2(A)$. Then $g \in L_2(A)$ is the **weak derivative of** f **of order** $|\alpha|$ if for all test functions φ on A, i.e., smooth real-valued functions with compact support in A, it holds that*

$$\langle g, \varphi \rangle_{L_2(A)} := \int_A g(x)\varphi(x)dx = \int_A f(x)D^\alpha \varphi(x)dx$$

where α denotes a multi-index, so

$$D^\alpha \varphi = \partial_{x_1}^{\alpha_1} \dots \partial_{x_n}^{\alpha_n} \varphi.$$

Clearly, if the weak distributional derivative exists, then it is unique since two weak derivatives g, h of f would lead to $\langle g - h, \varphi \rangle_{L_2(A)} = 0$ for all test functions φ on A, so by denseness of the space of all test functions on A in $L^p(A)$ for $1 \leq p < \infty$ (see e.g. [Werner, 2006, Lemma V.1.10], $g = h$ is valid.

Definition A.6.2. *Let $A \subset \mathbb{R}^n$ be open. The* ***Sobolev space*** *$W^{2,2}(A)$ is defined as*

$$W^{2,2}(A) := \left\{ f \in L_2(A) \;\middle|\; \sum_{|\alpha| \leq 2} ||D^\alpha f||^2_{L_2(A)} < \infty \right\},$$

where $D^\alpha f$ again is the weak distributional derivative.

The following theorem ([Berge, 1963, p. 116]) is referred to as ”Berge's maximum theorem” in Avella-Medina [2017]. We directly use the notation as in chapter 4.

Theorem A.6.1 (Berge's maximum theorem). *Let $R : \Theta \to \mathbb{R}$ be continuous and let $\Gamma : W^{2,2} \to \Theta$ be a mapping such that $\Gamma(J) \neq \emptyset$ for all $J \in W^{2,2}$. Then*

$$M : W^{2,2} \to \mathbb{R}, \quad M(J) := \max\{-R(\theta) \mid \theta \in \Gamma(J)\},$$

is continuous and

$$\Phi : W^{2,2} \to \Theta, \quad \Phi(J) := \{\theta \in \Gamma(J) \mid R(\theta) = M(J)\},$$

is upper semi-continuous.

A.7 Tools from asymptotic statistics

We recapitulate the well-known delta method as presented in Van der Vaart [2000]. The subsequent definitions from asymptotic statistics stem from Rieder [1994] and De la Peña and Giné [2012], respectively.

Lemma A.7.1 (Delta-Method). *Let $(X_n)_{n \geq 0}$ be a sequence of random variables taking values in $\mathcal{X} \subset \mathbb{R}^p$. Let $f : \mathcal{X} \to \mathbb{R}^k$ be a differentiable function. If for a sequence $s_n \to \infty$, it holds that*

$$s_n(X_n - \theta) \xrightarrow{w} Y,$$

then

$$s_n(f(X_n) - f(\theta)) \xrightarrow{w} f'(\theta)Y.$$

Definition A.7.1. *Let $(P_n)_n$ be a family of probability measures on measurable spaces $(\Omega_n, \mathcal{A}_n)$. Then $(P_n)_n$ is* ***tight*** *if for any $\epsilon > 0$ there exists a compact set $K \subset\subset \Omega_n$ such that*

$$\limsup_n (P_n(\Omega_n \setminus K)) < \epsilon.$$

Definition A.7.2. *Let (Θ, d) be a metric space. Then the minimal number of balls of radius ϵ that are necessary to cover Θ is called the* ***covering number*** *and is denoted by $N(\Theta, d, \epsilon)$. The logarithm of the covering number,*

$$H(\epsilon) := \ln(N(\Theta, d, \epsilon)),$$

is referred to as the ***metric entropy of*** Θ.

We recall following definition from Van der Vaart and Wellner [2013] for working with non-measurable functions.

For example, neither the empirical distribution function nor the uniform empirical process, written as mappings from $[0, 1]$ into the Skorohod space $\mathbb{D}([0, 1])$, are measurable (Van der Vaart and Wellner [2013]).

Definition A.7.3. *Let $(\Omega, \mathcal{A}, P)$ be a probability space and let $T : \Omega \to \bar{\mathbb{R}}$ be a map. Then*

$$\mathbb{E}^*[T] := \inf\{\mathbb{E}[U] \mid U \geq T,\ U : \Omega \to \bar{\mathbb{R}} \quad \textit{measurable}, \quad \mathbb{E}[U] \quad \textit{ex.}\}$$

is called the ***outer integral or outer expectation of*** T ***w.r.t.*** P*. The* ***outer probability of a set*** $B \subset \Omega$ *is given by*

$$P^*(B) := \inf\{P(A) \mid A \supset B, A \in \mathcal{A}\}.$$

Analogously, the ***inner integral or inner expectation of*** T ***w.r.t.*** P *is given by*

$$\mathbb{E}_*[T] := \sup\{\mathbb{E}[S] \mid S \leq T,\ S : \Omega \to \bar{\mathbb{R}} \quad \textit{measurable}, \quad \mathbb{E}[S] \quad \textit{ex.}\}$$

and the ***inner probability of a set*** $B \subset \mathcal{A}$ *is given by*

$$P_*(B) := \sup\{P(C) \mid C \subset B,\ C \in \mathcal{A}\}.$$

In these cases, the random variable U that is taken for the computation of $\mathbb{E}^*[T]$ is referred to as ***minimal measurable majorant*** *whereas in the inner expectation case, the respective variable S is the* ***maximal measurable minorant****.*

We continue to list some simple properties of inner and outer expectations and probabilities (cf. [Kosorok, 2007, Lemma 6.3+6.4+6.5], [Van der Vaart and Wellner, 2013, Lemma 1.2.1+1.2.3]).

Remark A.7.1. *Let $(\Omega, \mathcal{A}, P)$ be a probability space and $T : \Omega \to \bar{\mathbb{R}}$.*
i) *There exists a minimal measurable majorant $U : \Omega \to \bar{\mathbb{R}}$ and a maximal measurable minorant $S : \Omega \to \bar{\mathbb{R}}$.*
ii) *The relations*

$$\mathbb{E}^*[T] = \mathbb{E}_*[-T] \quad \text{and} \quad P_*(B) = 1 - P^*(\Omega \setminus B)$$

hold, where $B \in \mathcal{A}$.
iii) *It holds that*

$$\mathbb{E}^*[I_B] = P^*(B) \quad \text{and} \quad \mathbb{E}_*[I_B] = P_*(B)$$

for any $B \in \mathcal{A}$.

The general L_r-differentiability of which the frequently used L_2-differentiability is just a special case for $r = 2$ is defined as follows (cf. Rieder and Ruckdeschel [2001]). The subsequent lemma was proven in chapter 3.1 of the cited reference.

Definition A.7.4. *Let $\mathcal{P} := \{P_\theta \mid \theta \in \Theta\}$ be a family of probability measures on some measurable space $(\Omega, \mathcal{A})$ and let Θ be a subset of $\mathbb{R}^p$. Then $\mathcal{P}$ is L_r-**differentiable at** θ_0 if there exists $\Lambda_{\theta_0} \in L_r^p(P_{\theta_0})$ such that*

$$\left|\left|\sqrt[r]{dP_{\theta+h}} - \sqrt[r]{dP_\theta}\left(1 + \frac{1}{r}\Lambda_\theta^T h\right)\right|\right|_{L_r}^r = \int \left|\sqrt[r]{dP_{\theta+h}} - \sqrt[r]{dP_\theta}\left(1 + \frac{1}{r}\Lambda_\theta^T h\right)\right|^r = o(||h||^r).$$

Lemma A.7.2. *If the situation of definition A.7.4 the model $\mathcal{P}$ is L_r-differentiable in θ_0 with L_r-derivative Λ_{θ_0}, then it is also L_s-differentiable in θ_0 with the same derivative for any $1 \leq s \leq r$.*

If one faces parametric arrays of the form

$$\mathcal{P}_{n,i} := \{P_{n,i,\beta} \mid \beta \in \mathbb{R}^p\} \subset \mathcal{M}_1(\mathcal{A}_{n,i})$$

on measurable spaces $(\Omega_{n,i}, \mathcal{A}_{n,i})$ for $n \in \mathbb{N}$, $i = 1, ..., i_n$, L_2-differentiability can be defined in the following way (see [Rieder, 1994, Def. 2.3.8], [Pupashenko et al., 2015, Def. 2.5]).

Definition A.7.5. *Let $\Lambda_{n,i,\beta_0} \in L_2^p(P_{n,i,\beta_0})$ satisfy $\mathbb{E}_{n,i,\beta_0}[\Lambda_{n,i,\beta_0}] = 0$ for any i, n. Let*

$$I_{n,\beta_0} := \sum_i I_{n,i,\beta_0}$$

and for $t \in \mathbb{R}^p$, define

$$t_n := (I_{n,\beta_0})^{-1/2}, \quad U_{n,i} := t_n^T \Lambda_{n,i,\beta_0}.$$

If it holds for any $\epsilon \in]0,\infty[$ and any $t \in \mathbb{R}^p$ that

$$\lim_n \left(\sum_i \int_{\{|U_{n,i}|>\epsilon\}} U_{n,i}^2 dP_{n,i,\beta_0} \right) = 0,$$

and if for every $b \in]0,\infty[$ it holds that

$$\lim_n \left(\sup_{|t|\le b} \left(\sum_i \left|\left| \sqrt{dP_{n,i,\beta_0+t_n}} - \sqrt{dP_{n,i,\beta_0}} \left(1 + \frac{1}{2} U_{n,i,\beta_0}(t)\right) \right|\right|_{L_2}^2 \right) \right) = 0,$$

*then the parametric array $\mathcal{P} := \bigotimes_{i=1}^{i_n} \mathcal{P}_{n,i}$ is L_2-**differentiable in** $\beta_0 \in \mathbb{R}^p$ **at time** n with L_2-derivative Λ_{n,i,β_0} and Fisher information I_{n,i,β_0}. If additionally for any $h_n \to 0$ the property*

$$\lim_n \left(\sup_{|t|\le b} \left(\sum_i \left|\left| \sqrt{dP_{n,i,\beta_0+h_n}} U_{n,i,\beta_0+h_n}(t) - \sqrt{dP_{n,i,\beta_0}} U_{n,i,\beta_0}(t) \right|\right|_{L_2}^2 \right) \right) = 0$$

*holds, then the parametric array $\mathcal{P}$ is **continuously** L_2-**differentiable in** $\beta_0 \in \mathbb{R}^p$ **at time** n.*

Rieder shows that L_2-differentiability of a parametric array in this sense suffices to guarantee that the log-likelihoods of local alternatives are asymptotically normal by using the theorem of Lindeberg-Feller.

A.8 Functional Gradient Boosting

The following generic functional Gradient Boosting algorithm goes back to Friedman (Friedman [2001]). We use the notation of Bühlmann and Hothorn [2007].

The function L is concerned to be a loss function with two arguments. One is the response, the other one is the predicted response, understood in a functional way as the model that is used for prediction. The loss function has to be differentiable and convex in the second argument. The main idea behind this algorithm is to iteratively proceed along the steepest gradient.

```
Initialization: Data $(X,Y)$, step size $\kappa \in ]0,1]$, number $m_{iter}$ of iterations and
    $$\hat{f}^{(0)}(\cdot) \equiv \underset{c}{\text{argmin}} \left( \frac{1}{n} \sum_i L(Y_i, c) \right)$$
as offset value;
for $k = 1, ..., m_{iter}$ do
    Compute the negative gradients and evaluate them at the current model:
        $$U_i = -\partial_f L(Y_i, f)|_{f=\hat{f}^{(k-1)}(X_i)}$$
    for all $i = 1, ..., n$;
    Treat the vector $U = (U_i)_i$ as response and fit a model
        $$(X_i, U_i)_i \overset{\text{base procedure}}{\longrightarrow} \hat{g}^{(k)}(\cdot)$$
    with a preselected real-valued base procedure;
    Update the current model via
        $$\hat{f}^{(k)}(\cdot) = \hat{f}^{(k-1)}(\cdot) + \kappa \hat{g}^{(k)}(\cdot)$$
end
```

Algorithm 23: Generic functional Gradient Boosting

As pointed out in Bühlmann and Hothorn [2007], the models $\hat{g}^{(k)}$ can be regarded as an approximation of the current negative gradient vector.

A.9 Code for table 1.1

```
sumrand←0
sumclass←0
sumreg←0
B←10000
classvec←numeric(B)
for(i in 1:B){
        set.seed(i)
        D←genData0(5,200,s=1,snr=0.5)$D
        Dclass←D
        Dclass$Y[Dclass$Y>0]←1
        Dclass$Y[Dclass$Y≤ 0]←0
        Dtr←D[1:100,]
        Dclasstr←Dclass[1:100,]
        Dte←D[101:200,]
        Dclasste←Dclass[101:200,]
        Yhat←predict(lm(Y~.,Dtr),Dte)
        reslogit←glm(Y~.,Dclasstr,family='binomial')
        Yhatlog←1*(predict(reslogit,Dte)≥0)
        indlog←sample(which(Yhatlog==1),10,replace=F)
        classvec[i]←sum(Yhatlog==1)
        sumrand←sumrand+sum(Dte$Y[sample(1:100,10,replace=F)])
        sumclass←sumclass+sum(Dte$Y[indlog])
        sumreg←sumreg+sum(Dte$Y[order(Yhat,decreasing=T)[1:10]])
}
cat("The average profit made by randomly reviewing is",sumrand/B)
cat("The average profit made by classification and randomly revieving from the
      predicted fraud class is",sumclass/B)
cat("The average profit made by regression and reviewing the instances that
    are predicted to be the most profitable is",sumreg/B)
```

Index

Bibliography

S. Agarwal. Surrogate regret bounds for bipartite ranking via strongly proper losses. *The Journal of Machine Learning Research*, 15(1):1653–1674, 2014.

S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, and D. Roth. Generalization bounds for the area under the ROC curve. *Journal of Machine Learning Research*, 6(Apr):393–425, 2005.

C. Agostinelli, A. Leung, V. J. Yohai, and R. H. Zamar. Robust estimation of multivariate location and scatter in the presence of cellwise and casewise contamination. *Test*, 24(3): 441–461, 2015.

A. Alexandridis and A. Zapranis. Wind derivatives: Modeling and pricing. *Computational Economics*, 41(3):299–326, 2013.

A. Alfons. *robustHD: Robust Methods for High-Dimensional Data*, 2016. URL `https://CRAN.R-project.org/package=robustHD`. R package version 0.5.1.

A. Alfons, C. Croux, and S. Gelper. Sparse least trimmed squares regression for analyzing high-dimensional large data sets. *The Annals of Applied Statistics*, 7(1):226–248, 2013.

J. Alm, M. B. Cronshaw, and M. McKee. Tax compliance with endogenous audit selection rules. *Kyklos*, 46(1):27–45, 1993.

F. Alqallaf, S. Van Aelst, V. J. Yohai, R. H. Zamar, et al. Propagation of outliers in multivariate data. *The Annals of Statistics*, 37(1):311–331, 2009.

C. Anagnostopoulos, D. K. Tasoulis, N. M. Adams, N. G. Pavlidis, and D. J. Hand. Online linear and quadratic discriminant analysis with adaptive forgetting for streaming classification. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(2):139–166, 2012.

P. Anand, J. Krishnakumar, and N. B. Tran. Measuring welfare: Latent variable models for happiness and capabilities in the presence of unobservable heterogeneity. *Journal of public economics*, 95(3-4):205–215, 2011.

A. Y. Aravkin, J. V. Burke, and G. Pillonetto. Sparse/robust estimation and Kalman smoothing with nonsmooth log-concave densities: Modeling, computation, and theory. *The Journal of Machine Learning Research*, 14(1):2689–2728, 2013.

S. Arlot, A. Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

S. Arslanturk, M.-R. Siadat, T. Ogunyemi, K. Demirovic, and A. Diokno. Skip pattern analysis for detection of undetermined and inconsistent data. In *Biomedical Engineering and Informatics (BMEI), 2012 5th International Conference on*, pages 1122–1126. IEEE, 2012.

K. Ataman and W. N. Street. Optimizing area under the ROC curve using ranking SVMs. In *Proceedings of International Conference on Knowledge Discovery in Data Mining*, 2005.

H. Attouch, G. Buttazzo, and G. Michaille. *Variational analysis in Sobolev and BV spaces: applications to PDEs and optimization*. SIAM, 2014.

M. Avella-Medina. Influence functions for penalized M-estimators. *Bernoulli*, 23(4B):3178–3196, 2017.

V. Averbukh and O. Smolyanov. The theory of differentiation in linear topological spaces. *Russian Mathematical Surveys*, 22(6):201–258, 1967.

M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf. Multiple Imputation by Chained Equations: What is it and how does it work? *International journal of methods in psychiatric research*, 20(1):40–49, 2011.

F. R. Bach. Bolasso: Model consistent lasso estimation through the bootstrap. *arXiv preprint arXiv:0804.1302*, 2008.

M.-F. Balcan, N. Bansal, A. Beygelzimer, D. Coppersmith, J. Langford, and G. B. Sorkin. Robust reductions from ranking to classification. *Machine learning*, 72(1-2):139–153, 2008.

O. Banerjee, L. E. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine learning Research*, 9(Mar):485–516, 2008.

M. J. Bárcena Ruiz and F. J. Tusell Palmer. Multivariate data imputation using trees. 2002.

D. Bárcenas. The Radon-Nikodym theorem for reflexive banach spaces. *Divulgaciones Matemáticas*, 11(1):55–59, 2003.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

P. L. Bartlett, S. Boucheron, and G. Lugosi. Model selection and error estimation. *Machine Learning*, 48(1):85–113, 2002.

H. Bauer. *Maß-und Integrationstheorie.* Walter de Gruyter, 1992.

H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

C. Berge. *Topological Spaces: Including a treatment of multi-valued functions, vector spaces, and convexity.* Courier Corporation, 1963.

D. M. Berridge and R. Crouchley. *Multivariate generalized linear mixed models using R.* CRC Press, 2011.

E. Beutner and H. Zähle. A modified functional delta method and its application to the estimation of risk functionals. *Journal of Multivariate Analysis*, 101(10):2452–2463, 2010.

E. Beutner, H. Zähle, et al. Functional delta-method for the bootstrap of quasi-hadamard differentiable functionals. *Electronic Journal of Statistics*, 10(1):1181–1222, 2016.

A. Beygelzimer, E. Hazan, S. Kale, and H. Luo. Online gradient boosting. In *Advances in neural information processing systems*, pages 2458–2466, 2015a.

A. Beygelzimer, S. Kale, and H. Luo. Optimal and adaptive algorithms for online boosting. In *International Conference on Machine Learning*, pages 2323–2331, 2015b.

P. J. Bickel. One-step Huber estimates in the linear model. *Journal of the American Statistical Association*, 70(350):428–434, 1975.

P. J. Bickel and E. Levina. Regularized estimation of large covariance matrices. *The Annals of Statistics*, pages 199–227, 2008.

T. Björk. *Arbitrage theory in continuous time.* Oxford university press, 2009.

O. Blasco and P. Gregori. Vector measures with variation in a Banach function space. In *Proceedings of the Sixth Conference Function Spaces*, pages 65–79, 2002.

V. I. Bogachev. *Measure theory*, volume 1. Springer Science & Business Media, 2007a.

V. I. Bogachev. *Measure theory*, volume 2. Springer Science & Business Media, 2007b.

D. Borsboom, G. J. Mellenbergh, and J. Van Heerden. The theoretical status of latent variables. *Psychological review*, 110(2):203–219, 2003.

S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: probability and statistics*, 9:323–375, 2005.

K. Bowlin. Risk-based auditing, strategic prompts, and auditor sensitivity to the strategic risk of fraud. *The Accounting Review*, 86(4):1231–1253, 2011.

J. Boyko. *Handling Data with Three Types of Missing Values.* PhD thesis, University of Connecticut, 2013.

U. Brefeld and T. Scheffer. AUC maximizing support vector learning. In *Proceedings of the ICML 2005 workshop on ROC Analysis in Machine Learning*, 2005.

P. Breheny and J. Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The Annals of Applied Statistics*, 5(1):232–253, 2011.

L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

L. Breiman. Prediction games and arcing algorithms. *Neural computation*, 11(7):1493–1517, 1999.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

L. Breiman. *Classification and regression trees.* Routledge, 2017.

M. Broadie and P. Glasserman. Estimating security price derivatives using simulation. *Management science*, 42(2):269–285, 1996.

J. K. Brooks and P. W. Lewis. Linear operators and vector measures. *Transactions of the American Mathematical Society*, 192:139–162, 1974.

P. Bühlmann. Boosting for high-dimensional linear models. *The Annals of Statistics*, pages 559–583, 2006.

P. Bühlmann and T. Hothorn. Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science*, pages 477–505, 2007.

P. Bühlmann and S. Van De Geer. *Statistics for high-dimensional data: Methods, theory and applications.* Springer Science & Business Media, 2011.

P. Bühlmann and B. Yu. Boosting with the l_2 loss: Regression and Classification. *Journal of the American Statistical Association*, 98(462):324–339, 2003.

P. Bühlmann and B. Yu. Sparse boosting. *Journal of Machine Learning Research*, 7(Jun): 1001–1024, 2006.

P. Bühlmann, L. Meier, and H. Zou. Discussion of "One-step sparse estimates in nonconcave penalized likelihood models" by H. Zou and R. Li. *The Annals of Statistics*, 36:1534–1541, 2008.

P. Bühlmann, J. Gertheiss, S. Hieke, T. Kneib, S. Ma, M. Schumacher, G. Tutz, C.-Y. Wang, Z. Wang, and A. Ziegler. Discussion of "The evolution of boosting algorithms" and "Extending statistical boosting". *Methods of information in medicine*, 53(06):436–445, 2014.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

S. Butenko and P. M. Pardalos. *Numerical Methods and Optimization: An Introduction*. CRC Press, 2014.

B. L. Cabrera, M. Odening, and M. Ritter. Pricing rainfall futures at the CME. *Journal of Banking & Finance*, 37(11):4286–4298, 2013.

T. Cai, W. Liu, and X. Luo. A constrained l_1-minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011.

T. Calders and S. Jaroszewicz. Efficient AUC optimization for classification. In *PKDD*, volume 4702, pages 42–53. Springer, 2007.

E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Found. Comput. Math.*, 9(6):717–772, Dec. 2009. ISSN 1615-3375. doi: 10.1007/s10208-009-9045-5.

E. J. Candès, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted l_1-minimization. *Journal of Fourier analysis and applications*, 14(5):877–905, 2008.

Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking SVM to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193. ACM, 2006.

A. Capatina et al. *Variational inequalities and frictional contact problems*. Springer, 2014.

L. Chang, S. Roberts, and A. Welsh. Robust lasso regression using tukey's biweight criterion. *Technometrics*, 60(1):36–47, 2018.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

G. Chechik, G. Heitz, G. Elidan, P. Abbeel, and D. Koller. Max-margin classification of data with absent features. *Journal of Machine Learning Research*, 9(Jan):1–21, 2008.

S.-T. Chen, H.-T. Lin, and C.-J. Lu. An online boosting algorithm with theoretical justifications. *arXiv preprint arXiv:1206.6422*, 2012.

S.-T. Chen, H.-T. Lin, and C.-J. Lu. Boosting with online binary learners for the multiclass bandit problem. In *International Conference on Machine Learning*, pages 342–350, 2014.

X. Chen, Z. J. Wang, and M. J. McKeown. Asymptotic analysis of the Huberized lasso estimator. In *2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP),*, pages 1898–1901. IEEE, 2010a.

X. Chen, Z. J. Wang, and M. J. McKeown. Asymptotic analysis of robust lassos in the presence of noise with large variance. *IEEE Transactions on Information Theory*, 56(10): 5131–5149, 2010b.

A. Christmann and I. Steinwart. On robustness properties of convex risk minimization methods for pattern recognition. *Journal of Machine Learning Research*, 5(Aug):1007–1034, 2004.

A. Christmann and I. Steinwart. Consistency and robustness of kernel-based regression in convex risk minimization. *Bernoulli*, pages 799–819, 2007.

A. Christmann and A. Van Messem. Bouligand derivatives and robustness of support vector machines for regression. *Journal of Machine Learning Research*, 9(May):915–936, 2008.

A. Christmann, A. Van Messem, and I. Steinwart. On consistency and robustness properties of support vector machines for heavy-tailed distributions. *Statistics and Its Interface*, 2(3): 311–327, 2009.

S. Clémençon, G. Lugosi, and N. Vayatis. Ranking and empirical minimization of U-statistics. *The Annals of Statistics*, pages 844–874, 2008.

S. Clémençon and M. Achab. Ranking data with continuous labels through oriented recursive partitions. In *Advances in Neural Information Processing Systems*, pages 4603–4611, 2017.

S. Clémençon and S. Robbiano. Building confidence regions for the ROC surface. *Pattern Recognition Letters*, 46:67–74, 2014.

S. Clémençon and S. Robbiano. An ensemble learning technique for multipartite ranking. In *Proceedings*, pages 397–402. Presses universitaires de Louvain, 2015a.

S. Clémençon and S. Robbiano. The TreeRank Tournament algorithm for multipartite ranking. *Journal of Nonparametric Statistics*, 27(1):107–126, 2015b.

S. Clémençon and N. Vayatis. Ranking the best instances. *Journal of Machine Learning Research*, 8(Dec):2671–2699, 2007.

S. Clémençon and N. Vayatis. Tree-structured ranking rules and approximation of the optimal ROC curve. In *Proceedings of the 2008 conference on Algorithmic Learning Theory. Lect. Notes Art. Int*, volume 5254, pages 22–37, 2008.

S. Clémençon and N. Vayatis. Overlaying classifiers: a practical approach to optimal scoring. *Constructive Approximation*, 32(3):619–648, 2010.

S. Clémençon, G. Lugosi, N. Vayatis, P. Aurer, and R. Meir. Ranking and scoring using empirical risk minimization. In *Colt*, volume 3559, pages 1–15. Springer, 2005.

S. Clémençon, M. Depecker, and N. Vayatis. Bagging ranking trees. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, pages 658–663. IEEE, 2009.

S. Clémençon, M. Depecker, and N. Vayatis. Adaptive partitioning schemes for bipartite ranking. *Machine Learning*, 83(1):31–69, 2011.

S. Clémençon, M. Depecker, and N. Vayatis. Ranking forests. *Journal of Machine Learning Research*, 14(Jan):39–73, 2013a.

S. Clémençon, M. Depecker, and N. Vayatis. An empirical comparison of learning algorithms for nonparametric scoring: the TreeRank algorithm and other methods. *Pattern Analysis and Applications*, 16(4):475–496, 2013b.

S. Clémençon, S. Robbiano, and N. Vayatis. Ranking data with ordinal labels: optimality and pairwise aggregation. *Machine Learning*, 91(1):67–104, 2013c.

M. Cohen and G. Huang. Analyzing survey data by complete-case and available case methods. *Proceedings of the Survev Research Methods*, pages 289–294, 2000.

C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In *Advances in neural information processing systems*, pages 313–320, 2004.

C. Croux and V. Öllerer. Robust and sparse estimation of the inverse covariance matrix using rank correlation measures. In *Recent Advances in Robust Statistics: Theory and Applications*, pages 35–55. Springer, 2016.

G. Dal Maso. *An introduction to Γ-convergence*, volume 8. Springer Science & Business Media, 2012.

A. D'Ambrosio, S. Amodio, and G. Mazzeo. *ConsRank: Compute the Median Ranking(s) According to the Kemeny's Axiomatic Approach*, 2017. URL `https://CRAN.R-project.org/package=ConsRank`. R package version 2.0.1.

M. Danilov, V. J. Yohai, and R. H. Zamar. Robust estimation of multivariate location and scatter in the presence of missing data. *Journal of the American Statistical Association*, 107(499):1178–1186, 2012.

C. Daskalakis, R. M. Karp, E. Mossel, S. J. Riesenfeld, and E. Verbin. Sorting and selection in posets. *SIAM Journal on Computing*, 40(3):597–622, 2011.

A. Davenport and D. Lovell. Ranking pilots in aerobatic flight competitions. Technical report, Technical report, IBM Research Report RC23631 (W0506-079), TJ Watson . . . , 2005.

L. Davies. Lasso, knockoff and Gaussian covariates: a comparison. *arXiv preprint arXiv:1805.01862*, 2018.

L. Davies and L. Duembgen. A model-free approach to linear least squares regression with exact probabilities. *arXiv preprint arXiv:1807.09633*, 2018.

J.-E. Dazard, M. Choe, M. LeBlanc, and J. S. Rao. R package PRIMsrc: Bump hunting by patient rule induction method for survival, regression and classification. In *Proceedings/American Statistical Association. American Statistical Association. Meeting*, volume 2015, pages 650–664. NIH Public Access, 2015.

C. De Boor. *A practical guide to splines*, volume 27. Springer-Verlag New York, 1978.

V. De la Peña and E. Giné. *Decoupling: from dependence to independence*. Springer Science & Business Media, 2012.

J. C. De los Reyes, C.-B. Schönlieb, and T. Valkonen. The structure of optimal parameters for image restoration problems. *Journal of Mathematical Analysis and Applications*, 434 (1):464–500, 2016.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, pages 1–38, 1977.

K. Deng, S. Han, K. J. Li, and J. S. Liu. Bayesian aggregation of order-based rank data. *Journal of the American Statistical Association*, 109(507):1023–1039, 2014.

P. Descloux and S. Sardy. Model selection with lasso-zero: adding straw to the haystack to better find needles. *arXiv preprint arXiv:1805.05133*, 2018.

L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

R. Dezeure, P. Bühlmann, L. Meier, and N. Meinshausen. High-dimensional inference: Confidence intervals, p-values and R-software hdi. *Statistical Science*, 30(4):533–558, 2015.

C. Dhanjal and S. Clémençon. On recent advances in supervised ranking for metabolite profiling. *arXiv preprint arXiv:1402.1054*, 2014.

A. Dickerson and G. K. Popli. Persistent poverty and children's cognitive development: evidence from the UK millennium cohort study. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 179(2):535–558, 2016.

J. Diestel and J. Uhl. *Vector Measures*. Surveys, 1977.

I. N. Dimou. *Design and Implementation of Support Vector Machines and Information Fusion Methods for Bio-medical Decision Support Systems*. PhD thesis, Technical University of Crete, 2011.

D. L. Donoho and P. J. Huber. The notion of breakdown point. *A Festschrift for Erich L. Lehmann*, pages 157–184, 1983.

H. H. Duan. Bounding the fat shattering dimension of a composition function class built using a continuous logic connective. *arXiv preprint arXiv:1105.4618*, 2011.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

R. Dudley. Fréchet differentiability, p-variation and uniform donsker classes. *The Annals of Probability*, pages 1968–1982, 1992.

N. Dunford and B. J. Pettis. Linear operations on summable functions. *Trans. Amer. Math. Soc*, 47:323–392, 1940.

C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622. ACM, 2001.

B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

B. Efron, T. Hastie, and R. Tibshirani. Discussion: The Dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, 35(6):2358–2364, 2007.

J. Ehrlinger, H. Ishwaran, et al. Characterizing l_2Boosting. *The Annals of Statistics*, 40(2): 1074–1101, 2012.

P. H. Eilers and B. D. Marx. Flexible smoothing with B-splines and penalties. *Statistical Science*, pages 89–102, 1996.

M. Elad. *Sparse and Redundant Representations - From Theory to Applications in Signal and Image Processing*. 2010. ISBN 978-1-4419-7010-7.

S. M. A. Elrahman and A. Abraham. A review of class imbalance problem. *Journal of Network and Innovative Computing*, 1(2013):332–340, 2013.

J. Elstrodt. *Maß-und Integrationstheorie*. Springer-Verlag, 2006.

A. Evgrafov and M. Patriksson. On the existence of solutions to stochastic mathematical programs with equilibrium constraints. *Journal of Optimization Theory and Applications*, 121(1):65–76, 2004.

L. Fahrmeir and G. Tutz. *Multivariate statistical modelling based on generalized linear models*. Springer Science & Business Media, 2013.

L. Fahrmeir, T. Kneib, S. Lang, and B. Marx. *Regression*. Springer, 2007.

J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001.

J. Fan and J. Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(5):849–911, 2008.

J. Fan and W. Zhang. Statistical estimation in varying coefficient models. *Annals of Statistics*, pages 1491–1518, 1999.

J. Fan and W. Zhang. Statistical methods with varying coefficient models. *Statistics and its Interface*, 1(1):179–195, 2008.

J. Fan, Y. Liao, and H. Liu. An overview of the estimation of large covariance and precision matrices. *The Econometrics Journal*, 19(1):C1–C32, 2016.

A. Feelders. Handling missing data in trees: surrogate splits or statistical imputation? *Principles of Data Mining and Knowledge Discovery*, pages 329–334, 1999.

P. Filzmoser, R. Maronna, and M. Werner. Outlier identification in high dimensions. *Computational Statistics & Data Analysis*, 52(3):1694–1711, 2008.

P. Filzmoser, H. Fritz, and K. Kalcher. *pcaPP: Robust PCA by Projection Pursuit*, 2018. URL `https://CRAN.R-project.org/package=pcaPP`. R package version 1.9-73.

R. A. Fisher. Theory of statistical estimation. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 22, pages 700–725. Cambridge University Press, 1925.

T. Fissler, J. F. Ziegel, and T. Gneiting. Expected shortfall is jointly elicitable with value at risk-implications for backtesting. *arXiv preprint arXiv:1507.00244*, 2015.

T. Fissler, J. F. Ziegel, et al. Higher order elicitability and Osband's principle. *The Annals of Statistics*, 44(4):1680–1707, 2016.

T. Fissler, J. Hlavinová, and B. Rudloff. Elicitability and identifiability of systemic risk measures and other set-valued functionals. 07 2019.

R. Fraiman, V. J. Yohai, R. H. Zamar, et al. Optimal robust m-estimates of location. *The Annals of Statistics*, 29(1):194–223, 2001.

M. Fréchet. Sur la notion de différentielle dans l'analyse générale, 1937.

R. M. Freund, P. Grigas, R. Mazumder, et al. A new perspective on boosting in linear regression via subgradient optimization and relatives. *The Annals of Statistics*, 45(6): 2328–2364, 2017.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4(Nov):933–969, 2003.

J. Friedman, T. Hastie, R. Tibshirani, et al. Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics*, 28 (2):337–407, 2000.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

J. Friedman, T. Hastie, H. Höfling, R. Tibshirani, et al. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

R. Frongillo and I. A. Kash. On elicitation complexity and conditional elicitation. *arXiv preprint arXiv:1506.07212*, 2015.

X. Geng, T.-Y. Liu, T. Qin, and H. Li. Feature selection for ranking. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 407–414. ACM, 2007.

J. Gertheiss and G. Tutz. Penalized regression with ordinal predictors. *International Statistical Review*, 77(3):345–365, 2009.

J. Gertheiss, G. Tutz, et al. Sparse modeling of categorial explanatory variables. *The Annals of Applied Statistics*, 4(4):2150–2180, 2010.

J. Gertheiss, S. Hogger, C. Oberhauser, and G. Tutz. Selection of ordinally scaled independent variables with applications to international classification of functioning core sets. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 60(3):377–395, 2011.

R. D. Gill, J. A. Wellner, and J. Præstgaard. Non- and semi-parametric maximum likelihood estimators and the von mises method (part 1)[with discussion and reply]. *Scandinavian Journal of Statistics*, pages 97–128, 1989.

R. Gnanadesikan and J. R. Kettenring. Robust estimates, residuals, and outlier detection with multiresponse data. *Biometrics*, pages 81–124, 1972.

T. Gneiting. Making and evaluating point forecasts. *Journal of the American Statistical Association*, 106(494):746–762, 2011.

T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10):1469–1495, 2017.

P. J. Green. Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives. *Journal of the Royal Statistical Society: Series B (Methodological)*, 46(2):149–170, 1984.

C. Gulcehre, J. Sotelo, M. Moczulski, and Y. Bengio. A robust adaptive stochastic gradient method for deep learning. *arXiv preprint arXiv:1703.00788*, 2017.

M. Gupta and V. Nagadevara. Audit selection strategy for improving tax compliance–Application of data mining techniques. In *Foundations of Risk-Based Audits. Proceedings of the eleventh International Conference on e-Governance, Hyderabad, India, December*, pages 28–30, 2007.

R. Hable. Asymptotic normality of support vector machine variants and other regularized kernel methods. *Journal of Multivariate Analysis*, 106:92–117, 2012.

R. Hable and A. Christmann. On qualitative robustness of support vector machines. *Journal of Multivariate Analysis*, 102(6):993–1007, 2011.

G. Haffari, Y. Wang, S. Wang, G. Mori, and F. Jiao. Boosting with incomplete information. In *Proceedings of the 25th international conference on Machine learning*, pages 368–375. ACM, 2008.

J. Hájek. Local asymptotic minimax and admissibility in estimation. In *Proceedings of the sixth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 175–194, 1972.

F. Hampel. *Contributions to the theory of robust estimation.* PhD thesis, University of California, Berkeley, Calif, USA, 1968.

F. Hampel, E. Ronchetti, P. Rousseeuw, and W. Stahel. *Robust statistics: The approach based on influence functions*, volume 114. John Wiley & Sons, 2011.

F. R. Hampel. A general qualitative definition of robustness. *The Annals of Mathematical Statistics*, pages 1887–1896, 1971.

W. K. Härdle, B. L. Cabrera, and M. Ritter. Forecast based pricing of weather derivatives. 2012.

T. Hastie and B. Efron. *lars: Least Angle Regression, Lasso and Forward Stagewise*, 2013. URL `https://CRAN.R-project.org/package=lars`. R package version 1.2.

T. Hastie and R. Tibshirani. Varying-coefficient models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 757–796, 1993.

C. Heinrich. The mode functional is not elicitable. *Biometrika*, 101(1):245–251, 2013.

S. Heinrich. Multilevel Monte Carlo methods. In *International Conference on Large-Scale Scientific Computing*, pages 58–67. Springer, 2001.

R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. 1999.

W. Hersh, C. Buckley, T. Leone, and D. Hickam. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *SIGIR'94*, pages 192–201. Springer, 1994.

B. Hofner and T. Hothorn. *stabs: Stability Selection with Error Control*, 2017. URL `https://CRAN.R-project.org/package=stabs`. R package version 0.6-3.

B. Hofner, A. Mayr, N. Robinzonov, and M. Schmid. Model-based Boosting in R: A Hands-on Tutorial Using the R Package mboost. *Computational Statistics*, 29(1-2):3–35, 2014.

B. Hofner, L. Boccuto, and M. Göker. Controlling false discoveries in high-dimensional situations: Boosting with stability selection. *BMC Bioinformatics*, 16(1):144, 2015.

R. Hornung and M. N. Wright. Block forests: random forests for blocks of clinical and omics covariate data. 2018.

T. Hothorn. *TH.data: TH's Data Archive*, 2019. URL `https://CRAN.R-project.org/package=TH.data`. R package version 1.0-10.

T. Hothorn and P. Bühlmann. Model-based boosting in high dimensions. *Bioinformatics*, 22 (22):2828–2829, 2006.

T. Hothorn, P. Bühlmann, T. Kneib, M. Schmid, and B. Hofner. Model-based boosting 2.0. *Journal of Machine Learning Research*, 11(Aug):2109–2113, 2010.

T. Hothorn, P. Bühlmann, T. Kneib, M. Schmid, and B. Hofner. *mboost: Model-Based Boosting*, 2017. URL `https://CRAN.R-project.org/package=mboost`. R package version 2.8-1.

K.-W. Hsu, N. Pathak, J. Srivastava, G. Tschida, and E. Bjorklund. Data mining based tax audit selection: a case study of a pilot project at the Minnesota department of revenue. In *Real world data mining applications*, pages 221–245. Springer, 2015.

H. Hu, W. Sun, A. Venkatraman, M. Hebert, and J. A. Bagnell. Gradient boosting on stochastic data streams. *arXiv preprint arXiv:1703.00377*, 2017.

P. J. Huber and E. Ronchetti. *Robust Statistics.* Wiley, 2009.

M. Hubert and M. Debruyne. Minimum covariance determinant. *Wiley interdisciplinary reviews: Computational statistics*, 2(1):36–43, 2010.

M. Hubert, P. J. Rousseeuw, and S. Van Aelst. High-breakdown robust multivariate methods. *Statistical Science*, pages 92–119, 2008.

M. Hubert, P. J. Rousseeuw, and W. Van den Bossche. MacroPCA: An all-in-one PCA method allowing for missing values as well as cellwise and rowwise outliers. *Technometrics*, pages 1–18, 2019.

N. Jain and M. Marcus. Central limit theorems for C(S)-valued random variables. *Journal of Functional Analysis*, 19(3):216–231, 1975.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

J. G. Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

J. A. Khan, S. Van Aelst, and R. H. Zamar. Robust linear model selection based on least angle regression. *Journal of the American Statistical Association*, 102(480):1289–1299, 2007.

V. K. Khanna. Risk-based internal audit in Indian banks: A modified and improved approach for conduct of branch audit. *ICFAI Journal of Audit Practice*, 5(4), 2008.

A. Khedher. *Sensitivity and robustness to model risk in Lévy and jump-diffusion setting.* PhD thesis, University of Oslo, 2011.

W. R. Knight. A computer method for calculating Kendall's tau with ungrouped data. *Journal of the American Statistical Association*, 61(314):436–439, 1966.

R. Koenker and G. Bassett Jr. Regression quantiles. *Econometrica: Journal of the Econometric Society*, pages 33–50, 1978.

R. Koenker and S. Portnoy. *Quantile regression.* ABE, 1996.

M. Kohl. *Numerical contributions to the asymptotic theory of robustness.* PhD thesis, University of Bayreuth, 2005.

M. Kohl, P. Ruckdeschel, and H. Rieder. Infinitesimally robust estimation in general smoothly parametrized models. *Statistical Methods & Applications*, 19(3):333–354, 2010.

V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, pages 1–50, 2002.

A. Korba, S. Clémençon, and E. Sibony. A learning theory of ranking aggregation. In *Artificial Intelligence and Statistics*, pages 1001–1010, 2017.

M. R. Kosorok. *Introduction to empirical processes and semiparametric inference*. Springer Science & Business Media, 2007.

V. Krätschmer, A. Schied, and H. Zähle. Qualitative and infinitesimal robustness of tail-dependent statistical functionals. *Journal of Multivariate Analysis*, 103(1):35–47, 2012.

H.-P. Kriegel, P. Kröger, A. Pryakhin, and M. Schubert. Using support vector machines for classifying large sets of multi-represented objects. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 102–113. SIAM, 2004.

H. Lai, Y. Pan, C. Liu, L. Lin, and J. Wu. Sparse learning-to-rank via an efficient primal-dual algorithm. *IEEE Transactions on Computers*, 62(6):1221–1233, 2013a.

H. Lai, Y. Pan, Y. Tang, and N. Liu. Efficient gradient descent algorithm for sparse models with application in learning-to-rank. *Knowledge-Based Systems*, 49:190–198, 2013b.

N. S. Lambert, D. M. Pennock, and Y. Shoham. Eliciting properties of probability distributions. In *Proceedings of the 9th ACM Conference on Electronic Commerce*, pages 129–138. ACM, 2008.

S. Lambert-Lacroix, L. Zwald, et al. Robust regression through the Huber's criterion and adaptive lasso penalty. *Electronic Journal of Statistics*, 5:1015–1053, 2011.

T. Lan, W. Yang, Y. Wang, and G. Mori. Image retrieval with structured object queries using latent ranking SVM. In *European conference on computer vision*, pages 129–142. Springer, 2012.

L. Laporte, R. Flamary, S. Canu, S. Déjean, and J. Mothe. Nonconvex regularizations for feature selection in ranking with sparse SVM. *IEEE Transactions on Neural Networks and Learning Systems*, 25(6):1118–1130, 2014.

L. Le, R. Kumaraswamy, and M. White. Learning sparse representations in reinforcement learning with sparse coding. *arXiv preprint arXiv:1707.08316*, 2017.

L. Le Cam. *Asymptotic Methods in Statistical Decision Theory*. Springer Series in Statistics. Springer New York, 1986.

L. LeCam. On the assumptions used to prove asymptotic normality of maximum likelihood estimates. *The Annals of Mathematical Statistics*, 41(3):802–828, 1970.

E. LeDell, M. Petersen, and M. van der Laan. Computationally efficient confidence intervals for cross-validated area under the ROC curve estimates. *Electronic Journal of Statistics*, 9 (1):1583, 2015.

C. Leistner, A. Saffari, P. M. Roth, and H. Bischof. On robustness of on-line boosting-a competitive study. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1362–1369. IEEE, 2009.

A. Leung, H. Zhang, and R. Zamar. Robust regression estimation and inference in the presence of cellwise and casewise contamination. *Computational Statistics & Data Analysis*, 99:1–11, 2016.

A. Leung, V. Yohai, and R. Zamar. Multivariate location and scatter matrix estimation under cellwise and casewise contamination. *Computational Statistics & Data Analysis*, 111:59–76, 2017.

E. Levitin and R. Tichatschke. On smoothing of parametric minimax-functions and generalized max-functions via regularization. *Journal of Convex Analysis*, 5:199–220, 1998.

X. Li, T. Zhao, L. Wang, X. Yuan, and H. Liu. *flare: Family of Lasso Regression*, 2018. URL `https://CRAN.R-project.org/package=flare`. R package version 1.6.0.

H.-T. Lin. *From ordinal ranking to binary classification.* PhD thesis, California Institute of Technology, 2008.

R. J. Little and D. B. Rubin. *Statistical analysis with missing data*, volume 333. John Wiley & Sons, 2014.

P.-L. Loh. Statistical consistency and asymptotic normality for high-dimensional robust M-estimators. *The Annals of Statistics*, 45(2):866–896, 2017.

P.-L. Loh and M. J. Wainwright. High-dimensional regression with noisy and missing data: Provable guarantees with nonconvexity. *The Annals of Statistics*, 40(3):1637–1664, 2012.

P.-L. Loh and M. J. Wainwright. Regularized M-estimators with nonconvexity: Statistical and algorithmic theory for local optima. *Journal of Machine Learning Research*, 16:559–616, 2015.

P.-L. Loh, M. J. Wainwright, et al. Support recovery without incoherence: A case for nonconvex regularization. *The Annals of Statistics*, 45(6):2455–2482, 2017.

P.-L. Loh, X. L. Tan, et al. High-dimensional robust precision matrix estimation: Cellwise corruption under ϵ-contamination. *Electronic Journal of Statistics*, 12(1):1429–1467, 2018.

R. D. Luce. Individual choice behavior. 1959.

R. W. Lutz and P. Bühlmann. Conjugate direction boosting. *Journal of Computational and Graphical Statistics*, 15(2):287–311, 2006a.

R. W. Lutz and P. Bühlmann. Boosting for high-multivariate responses in high-dimensional linear regression. *Statistica Sinica*, 16(2):471, 2006b.

R. W. Lutz, M. Kalisch, and P. Bühlmann. Robustified L_2boosting. *Computational Statistics & Data Analysis*, 52(7):3331–3341, 2008.

C. L. Mallows. Non-null ranking models. I. *Biometrika*, 44(1/2):114–130, 1957.

V. Manewitsch. *Statistische Methoden zur Analyse von Daten mit strukturell fehlenden Werten: Mit Anwendungen aus der Marktforschung.* PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2013.

R. Maronna, R. Martin, and V. Yohai. Robust statistics: Theory and methods. *Annals of Statistics*, 30:17–23, 2006.

R. A. Maronna. Robust ridge regression for high-dimensional data. *Technometrics*, 53(1): 44–53, 2011.

V. Maume-Deschamps, D. Rullière, and K. Said. Multivariate extensions of expectiles risk measures. *Dependence Modeling*, 5(1):20–44, 2017.

A. Mayr, N. Fenske, B. Hofner, T. Kneib, and M. Schmid. Generalized additive models for location, scale and shape for high dimensional data—a flexible approach based on boosting. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61(3):403–427, 2012a.

A. Mayr, B. Hofner, and M. Schmid. The importance of knowing when to stop. *Methods of Information in Medicine*, 51(02):178–186, 2012b.

R. Mazumder, J. H. Friedman, and T. Hastie. SparseNet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495):1125–1138, 2011.

N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

N. Meinshausen, G. Rocha, and B. Yu. Discussion: A tale of three cousins: Lasso, L2Boosting and Dantzig. *The Annals of Statistics*, 35(6):2373–2384, 2007.

R. P. Monti, P. Hellyer, D. Sharp, R. Leech, C. Anagnostopoulos, and G. Montana. Estimating time-varying brain connectivity networks from functional MRI time series. *NeuroImage*, 103:427–443, 2014.

R. P. Monti, R. Lorenz, C. Anagnostopoulos, R. Leech, and G. Montana. Measuring the functional connectome "on-the-fly": towards a new control signal for fMRI-based brain-computer interfaces. *arXiv preprint arXiv:1502.02309*, 2015.

R. P. Monti, C. Anagnostopoulos, and G. Montana. A framework for adaptive regularization in streaming lasso models. *arXiv preprint arXiv:1610.09127*, 2016.

M. Moraru and F. Dumitru. The risks in the audit activity. *Annals of the University of Petrosani. Economics*, 11:187–194, 2011.

A. More. Survey of resampling techniques for improving classification performance in unbalanced datasets. *arXiv preprint arXiv:1608.06048*, 2016.

V. Öllerer and C. Croux. Robust high-dimensional precision matrix estimation. In *Modern Nonparametric, Robust and Multivariate Methods*, pages 325–350. Springer, 2015.

V. Öllerer, C. Croux, and A. Alfons. The influence function of penalized regression estimators. *Statistics*, 49(4):741–765, 2015.

N. C. Oza. Online bagging and boosting. In *2005 IEEE international conference on systems, man and cybernetics*, volume 3, pages 2340–2345. IEEE, 2005.

L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Boberg, and T. Salakoski. Learning to rank with pairwise regularized least-squares. In *SIGIR 2007 workshop on learning to rank for information retrieval*, volume 80, pages 27–33, 2007.

F. Pan, T. Converse, D. Ahn, F. Salvetti, and G. Donato. Feature selection for ranking using boosted trees. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 2025–2028. ACM, 2009.

M. Y. Park and T. Hastie. L_1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):659–677, 2007.

T. Patel, D. Telesca, R. Rallo, S. George, T. Xia, and A. E. Nel. Hierarchical rank aggregation with applications to nanotoxicology. *Journal of agricultural, biological, and environmental statistics*, 18(2):159–177, 2013.

K. Pelckmans, J. De Brabanter, J. A. Suykens, and B. De Moor. Handling missing values in support vector machine classifiers. *Neural Networks*, 18(5):684–692, 2005.

J. Peypouquet. *Convex optimization in normed spaces: theory, methods and examples.* Springer, 2015.

K. S. Pickett. *Audit planning: a risk-based approach.* John Wiley & Sons, 2006.

G. Pison, S. Van Aelst, and G. Willems. Small sample corrections for LTS and MCD. In *Developments in Robust Statistics*, pages 330–343. Springer, 2003.

R. L. Plackett. The analysis of permutations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 24(2):193–202, 1975.

D. Pupashenko. *Robustness for regression models with asymmetric error distribution.* PhD thesis, Technische Universität Kaiserslautern, 2015.

D. Pupashenko, P. Ruckdeschel, and M. Kohl. L2 differentiability of generalized linear models. *Statistics & Probability Letters*, 97(C):155–164, 2015.

A. Rakhlin, K. Sridharan, and A. Tewari. Online learning: Beyond regret. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 559–594, 2011.

A. Rakotomamonjy. Optimizing area under Roc curve with SVMs. In *ROCAI*, pages 71–80, 2004.

A. Rakotomamonjy. Sparse support vector infinite push. *arXiv preprint arXiv:1206.6432*, 2012.

J. Reeds. *On the definition of von Mises functionals.* PhD thesis, Harvard University, 1976.

C. H. Reinsch. Smoothing by spline functions. *Numerische Mathematik*, 10(3):177–183, 1967.

H. Reiter. Spaces with compact subtopologies. *The Rocky Mountain Journal of Mathematics*, 2(2):239–247, 1972.

M. T. Ribeiro and I. Cano. Keeping things that matter: An exploration on delayed feedback online learning, 2014.

H. Rieder. *Robust asymptotic statistics*, volume 1. Springer Science & Business Media, 1994.

H. Rieder and P. Ruckdeschel. Short proofs on L_r-differentiability. *Statistics & Risk Modeling*, 19(4):419–426, 2001.

H. Rieder, M. Kohl, and P. Ruckdeschel. The cost of not knowing the radius. *Statistical Methods & Applications*, 17(1):13–40, 2008.

S. Robbiano. *Méthodes d'apprentissage statistique pour le ranking théorie, algorithmes et applications.* PhD thesis, Télécom ParisTech, 2013.

S. Robinson. An implicit-function theorem for B-differentiable functions. 1988.

R. Rockafellar and R. J.-B. Wets. *Variational Analysis.* Springer Verlag, Heidelberg, Berlin, New York, 1998.

R. T. Rockafellar. Second-order convex analysis. *Journal of Nonlinear and Convex Analysis*, 1(1-16):84, 1999.

D. M. Rocke and D. L. Woodruff. Identification of outliers in multivariate data. *Journal of the American Statistical Association*, 91(435):1047–1061, 1996.

W. Römisch. Delta method, infinite dimensional. *Encyclopedia of statistical sciences*, 3, 2004.

S. Rosset and J. Zhu. Piecewise linear regularized solution paths. *The Annals of Statistics*, pages 1012–1030, 2007.

S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5(Aug):941–973, 2004.

P. J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871–880, 1984.

P. J. Rousseeuw. Multivariate estimation with high breakdown point. *Mathematical Statistics and Applications*, 8(283-297):37, 1985.

P. J. Rousseeuw and C. Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424):1273–1283, 1993.

P. J. Rousseeuw and M. Hubert. Robust statistics for outlier detection. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):73–79, 2011.

P. J. Rousseeuw and W. Van Den Bossche. Detecting deviating data cells. *Technometrics*, 60(2):135–145, 2018.

P. J. Rousseeuw and K. Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

P. J. Rousseeuw and K. Van Driessen. An algorithm for positive-breakdown regression based on concentration steps. In *Data Analysis*, pages 335–346. Springer, 2000.

P. J. Rousseeuw and K. Van Driessen. Computing LTS regression for large data sets. *Data mining and knowledge discovery*, 12(1):29–45, 2006.

P. J. Rousseeuw, S. Van Aelst, K. Van Driessen, and J. A. Gulló. Robust multivariate regression. *Technometrics*, 46(3):293–305, 2004.

P. Ruckdeschel. *Ansätze zur Robustifizierung des Kalman-Filters*. PhD thesis, University of Bayreuth, 2001.

P. Ruckdeschel. Uniform integrability on neighborhoods. Technical report, Fraunhofer ITWM, 2010a.

P. Ruckdeschel. Fisher information in group-type models. *arXiv preprint arXiv:1005.1027*, 2010b.

C. Rudin. The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list. *Journal of Machine Learning Research*, 10(Oct):2233–2271, 2009.

C. Rudin and R. E. Schapire. Margin-based ranking and an equivalence between AdaBoost and RankBoost. *Journal of Machine Learning Research*, 10(Oct):2193–2232, 2009.

W. Rudin. *Real and complex analysis*. Tata McGraw-Hill Education, 1987.

D. F. Saldana and Y. Feng. SIS: An R package for sure independence screening in ultrahigh-dimensional statistical models. *Journal of Statistical Software*, 83(2):1–25, 2018. doi: 10.18637/jss.v083.i02.

M. Salibián-Barrera and S. Van Aelst. Robust model selection using fast and robust bootstrap. *Computational Statistics & Data Analysis*, 52(12):5121–5135, 2008.

M. Salibián-Barrera, R. H. Zamar, et al. Bootrapping robust estimates of regression. *The Annals of Statistics*, 30(2):556–582, 2002.

M. Salibián-Barrera, S. Van Aelst, and G. Willems. Fast and robust bootstrap. *Statistical Methods and Applications*, 17(1):41–71, 2008.

J. Schelldorfer, L. Meier, and P. Bühlmann. GLMMLasso: an algorithm for high-dimensional generalized linear mixed models using l_1-penalization. *Journal of Computational and Graphical Statistics*, 23(2):460–477, 2014.

M. Schmid and T. Hothorn. Boosting additive models using component-wise P-splines. *Computational Statistics & Data Analysis*, 53(2):298–311, 2008.

B. Schölkopf, R. Herbrich, and A. Smola. A generalized representer theorem. In *Computational learning theory*, pages 416–426. Springer, 2001.

R. D. Shah and R. J. Samworth. Variable selection with error control: Another look at stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(1):55–80, 2013.

A. Shapiro. On concepts of directional differentiability. *Journal of optimization theory and applications*, 66(3):477–487, 1990.

A. D. Shieh and Y. S. Hung. Detecting outlier samples in microarray data. *Statistical applications in genetics and molecular biology*, 8(1):1–24, 2009.

N. Z. Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

E. Smucler and V. J. Yohai. Robust and sparse estimators for linear regression models. *Computational Statistics & Data Analysis*, 111:116–130, 2017.

M. Sommerfeld and A. Munk. Inference for empirical Wasserstein distances on finite spaces. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(1):219–238, 2018.

N. Städler and P. Bühlmann. Missing values: sparse inverse covariance estimation and an extension to sparse regression. *Statistics and Computing*, 22(1):219–235, 2012.

N. Städler, D. J. Stekhoven, and P. Bühlmann. Pattern alternating maximization algorithm for missing data in high-dimensional problems. *The Journal of Machine Learning Research*, 15(1):1903–1928, 2014.

D. J. Stekhoven and P. Bühlmann. MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2011.

Z. Sun, T. Qin, Q. Tao, and J. Wang. Robust sparse rank learning for non-smooth ranking measures. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 259–266. ACM, 2009.

M. Taddy. One-step estimator paths for concave regularization. *Journal of Computational and Graphical Statistics*, 26(3):525–536, 2017.

V. N. Temlyakov. Weak greedy algorithms. *Advances in Computational Mathematics*, 12 (2-3):213–227, 2000.

J. Thomas, A. Mayr, B. Bischl, M. Schmid, A. Smith, and B. Hofner. Gradient boosting for distributional regression: faster tuning and improved variable selection via noncyclical updates. *Statistics and Computing*, 28(3):673–687, 2018.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

R. J. Tibshirani. A general framework for fast stagewise algorithms. *The Journal of Machine Learning Research*, 16(1):2543–2588, 2015.

E. Tsivtsivadze and T. Heskes. Semi-supervised ranking pursuit. *arXiv preprint arXiv:1307.0846*, 2013.

G. Tutz. *Regression for categorical data*, volume 34. Cambridge University Press, 2011.

G. Tutz and A. Groll. Generalized linear mixed models based on boosting. In *Statistical Modelling and Regression Structures*, pages 197–215. Springer, 2010.

S. Van Aelst and G. Willems. Multivariate regression S-estimators for robust estimation and inference. *Statistica Sinica*, pages 981–1001, 2005.

S. Van de Geer. *Estimation and testing under sparsity.* Springer, 2016.

A. Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge University Press, 2000.

A. Van der Vaart and J. Wellner. *Weak convergence and empirical processes: With applications to statistics.* Springer Science & Business Media, 2013.

V. Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

V. Vapnik. *The nature of statistical learning theory.* Springer science & Business media, 2013.

E. D. Vito, L. Rosasco, A. Caponnetto, M. Piana, and A. Verri. Some properties of regularized kernel methods. *Journal of Machine Learning Research*, 5(Oct):1363–1390, 2004.

M. Vogt. On the differences between L_2-boosting and the lasso. *arXiv preprint arXiv:1812.05421*, 2018.

R. Von Mises. On the asymptotic distribution of differentiable statistical functions. *The Annals of Mathematical Statistics*, 18(3):309–348, 1947.

K. Wagstaff. Clustering with missing values: No imputation required. In *Classification, Clustering, and Data Mining Applications*, pages 649–658. Springer, 2004.

C. Wang and Z. Feng. Boosting with missing predictors. *Biostatistics*, 11(2):195–212, 2009.

H. Wang, G. Li, and G. Jiang. Robust regression shrinkage and consistent variable selection through the LAD-lasso. *Journal of Business & Economic Statistics*, 25(3):347–355, 2007.

L. Wang, J. Zhu, and H. Zou. The doubly regularized support vector machine. *Statistica Sinica*, 16(2):589, 2006.

L. Wang, Y. Kim, and R. Li. Calibrating non-convex penalized regression in ultra-high dimension. *Annals of Statistics*, 41(5):2505, 2013.

S. Wang, B. Nan, S. Rosset, and J. Zhu. Random lasso. *The Annals of Applied Statistics*, 5 (1):468, 2011.

F. Wei and H. Zhu. Group coordinate descent algorithms for nonconvex penalized regression. *Computational Statistics & Data Analysis*, 56(2):316–326, 2012.

J. A. Wellner. Empirical processes in action: A review. *International Statistical Review/Revue Internationale de Statistique*, pages 247–269, 1992.

D. Werner. *Funktionalanalysis.* Springer, 2006.

A. Wille, P. Zimmermann, E. Vranová, A. Fürholz, O. Laule, S. Bleuler, L. Hennig, A. Prelić, P. von Rohr, L. Thiele, et al. Sparse graphical gaussian modeling of the isoprenoid gene network in arabidopsis thaliana. *Genome Biology*, 5(11):R92, 2004.

D. M. Witten and R. Tibshirani. A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105(490):713–726, 2010.

H. Yang and D. Carlin. ROC surface: a generalization of ROC curve analysis. *Journal of biopharmaceutical statistics*, 10(2):183–196, 2000.

Y. Yang. Can the strengths of AIC and BIC be shared? A conflict between model indentification and regression estimation. *Biometrika*, 92(4):937–950, 2005.

C. Yi. *hqreg: Regularization Paths for Lasso or Elastic-Net Penalized Huber Loss Regression and Quantile Regression*, 2017. URL `https://CRAN.R-project.org/package=hqreg`. R package version 1.4.

C. Yi and J. Huang. Semismooth newton coordinate descent algorithm for elastic-net penalized Huber loss regression and quantile regression. *Journal of Computational and Graphical Statistics*, 26(3):547–557, 2017.

V. J. Yohai et al. High breakdown-point and high efficiency robust estimates for regression. *The Annals of Statistics*, 15(2):642–656, 1987.

L. Yu, S. Wang, and K. K. Lai. An online learning algorithm with adaptive forgetting factors for feedforward neural networks in financial time series forecasting. *Nonlinear dynamics and systems theory*, 7(1):51–66, 2007.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

L. Zajíček. Hadamard differentiability via Gâteaux differentiability. *Proceedings of the American Mathematical Society*, 143(1):279–288, 2015.

A. Zellner. An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias. *Journal of the American Statistical Association*, 57(298):348–368, 1962.

C.-H. Zhang et al. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, 2010.

X. Zhang, Y. Wu, L. Wang, and R. Li. Variable selection for support vector machines in moderately high dimensions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(1):53–76, 2016.

P. Zhao and B. Yu. Boosted lasso. Technical report, California University Berkeley Departement of Statistics, 2004.

P. Zhao and B. Yu. Stagewise lasso. *Journal of Machine Learning Research*, 8(Dec):2701–2726, 2007.

J. Zhu, S. Rosset, R. Tibshirani, and T. J. Hastie. 1-norm support vector machines. In *Advances in neural information processing systems*, pages 49–56, 2004.

J. F. Ziegel. Coherence and elicitability. *Mathematical Finance*, 26(4):901–918, 2016.

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429, 2006.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

H. Zou and R. Li. One-step sparse estimates in nonconcave penalized likelihood models. *Annals of Statistics*, 36(4):1509, 2008.

Wissenschaftlicher Werdegang

seit 10/2016	Carl von Ossietzky Universität Oldenburg Wissenschaftlicher Mitarbeiter am Institut für Mathematik
10/2016 bis 9/2019	Promotion bei Prof. Dr. Peter Ruckdeschel
10/2014 bis 6/2016	Master Studium an der Carl von Ossietzky Universität Oldenburg Thema der Masterarbeit: Theorie, Vergleich und Anwendung des PCP- und Fast-PCP-Algorithmus in der Bildverarbeitung Mit Auszeichung bestanden
10/2011 bis 8/2014	Bachelor-Studium an der Carl von Ossietzky Universität Oldenburg Thema der Bachelorarbeit: Der NESTA-Algorithmus für Compressed Sensing
8/2004-7/2011	Besuch des Lothar-Meyer-Gymnasiums in Varel

Folgende Konferenzbeiträge sind aus dieser Arbeit hervorgegangen:

Werner, T.: The ranking problem in statistical learning (oral presentation), Joint PhD Seminar in Statistics and Stochastics, Bremen

Werner, T.: Compact differentiability of regularized M-functionals (oral presentation), Statistische Woche 2017, Rostock

Werner, T.: Asymptotic linear expansion of regularized M-estimators (oral presentation), 13th German Probability and Statistics Days, Freiburg

Werner, T.: Asymptotic linearity, robustness and sparsity: From regularized regression to ranking (oral presentation), Joint PhD Seminar in Statistics, Actuarial and Financial Mathematics, Oldenburg

Werner, T.; Ruckdeschel, P.: L_2-Boosting for complicated loss functions by means of the column measure (poster presentation), DAGStat 2019, Munich

Folgende Arbeiten wurden eingereicht:

Werner, T.: Asymptotic linear expansion of regularized M-estimators, *arXiv: 1909.00579*, 2019 (eingereicht)

Werner, T.: A review on ranking problems in statistical learning, *arXiv: 1909.02998*, 2019 (eingereicht)

Werner, T.; Ruckdeschel, P.: The column measure and Gradient-Free Gradient Boosting, *arXiv: 1909.10960*, 2019 (eingereicht)

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichungen, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.