Erweiterung inferenzstatistischer Fähigkeiten modellbasierter Gradient-Boosting Algorithmen

–

Extending the inferential capabilities of model-based gradient boosting algorithms

Der Medizinischen Fakultät
der Friedrich-Alexander-Universität
Erlangen-Nürnberg
zur Erlangung des Doktorgrades
Dr. rer. biol. hum.

vorgelegt von
M.Sc. Tobias Hepp
aus Bamberg

Als Dissertation genehmigt
von der Medizinischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg
Tag der mündlichen Prüfung: 23.07.2019

Vorsitzender des Promotionsorgans: Prof. Dr. Dr. h. c. Jürgen Schüttler

Gutachter/in: Prof. Dr. Olaf Gefeller
Prof. Dr. Andreas Mayr
Prof. Dr. Mark Stemmler
PD Dr. Dr. Werner Adler

Inhaltsverzeichnis

Zusammenfassung

Hintergrund und Ziele

Die rasante technologische Entwicklung der vergangenen Jahrzehnte ermöglichte nicht nur die praktische Anwendung zuvor lediglich theoretischer Konzepte der statistischen Datenanalyse sondern führte auch zu einer Vielzahl neuer, zunehmend rechenintensiven Analysestrategien aus dem Umfeld des maschinellen Lernens. Weiterentwicklungen der erfolgreichen Boosting-Algorithmen offenbarten deren Nähe zu bekannten statistischen Konzepten und machten diese für die Schätzung regularisierter Regressionsparameter additiver Modelle nutzbar. Die vorliegende Arbeit richtet den Fokus auf die daraus resultierenden Modelleigenschaften sowie deren Verbesserung und Erweiterung bezüglich inferenzstatistischer Validität und Interpretierbarkeit.

Methoden

Alle vorgestellten Ansätze beziehen sich auf unterschiedliche Formen modellbasierter Boosting-Algorithmen. Diese starten bei der Initialisierung mit einem leeren Nullmodell, welches in den nachfolgenden Iterationen schrittweise durch wiederholte Anwendung von Regressionsfunktionen sequentiell erweitert wird um schließlich ein additives Modell zu bilden. Die vorliegende Arbeit untersucht die daraus resultierenden Schätzer und Modelleigenschaften zunächst im Vergleich mit anderen Regularisierungsmethoden wie L_1-Penalisierung. Darüber hinaus werden alternative Strategien zur Verbesserung der Variablenselektion des Gesamtmodells vorgeschlagen sowie alternative Teststrategien zur Überprüfung einzelner Effekte entwickelt. Dabei wird vornehmlich auf Varianten der Variablenpermutation und Bootstrapping-Methoden zurückgegriffen.

Ergebnisse

Die Regularisierung linearer Effektschätzer mittels modellbasierten Gradientenboostings verhält sich im Falle diagonaler Dominanz der inversen Kovarianzmatrix der Prädiktorvariablen mit sinkender Schrittlänge ν asymptotisch zur L_1-Penalisierung. Unterschiede zwischen den Verfahren lassen sich auf die sequentielle Aggregation des

Boosting-Modells zurückführen, wodurch zwar einerseits die Regularisierungspfade stabilisiert werden, andererseits aber die Modelle tendenziell mehr Variablen aufnehmen. Um eine Vielzahl falsch positiver Selektionen zu vermeiden, kann über die Erweiterung der Daten um permutierte Varianten der Prädiktorvariablen der Fokus von der Prognosegüte auf die Variablenselektion gelenkt werden. Residuenpermutation und parametrischer Bootstrap ermöglichen die Berechnung von p-Werten, die in niedrigdimensionalen Szenarien die gleiche Power erreichen wie Wald-Tests für Maximum-Likelihood-Schätzer.

Praktische Schlussfolgerungen

Die Ergebnisse dieser Arbeit bieten eine Entscheidungshilfe bei der Wahl zwischen Boosting und L_1-Penalisierung als Regularisierungsmethode für statistische Modelle. Zudem wird die Anwendbarkeit modellbasierter Gradient-Boosting-Algorithmen in Situationen verbessert, in denen die weiterführende Interpretation der selektierten Variablen von zentralem Interesse ist. Zum Einen lässt sich die Genauigkeit der Variablenselektion durch alternatives Tuning mittels permutierter Variablen erhöhen. Darüber hinaus erlaubt die Verwendung des parametrischen Bootstraps erstmals die Berechnung von p-Werten für einzelne Effektschätzer modellbasierter Gradient-Boosting Algorithmen in hochdimensionalen Szenarien mit korrelierten Prädiktorvariablen.

Abstract

Background and aims

The rapid development of computer technology in recent decades has not only enabled the practical application of previously merely theoretical ideas of statistical data analysis but has also led to a multitude of new and increasingly computationally intensive analysis strategies emerging from the field of machine learning. Further developments of the successful boosting algorithms revealed their relationship to known statistical concepts and made them usable for the estimation of regularized regression parameters of additive models. This thesis focuses on the resulting model properties as well as their improvement and extension with regard to inferential statistical validity and interpretability.

Methods

All presented approaches address various forms of model-based boosting algorithms. The algorithm is initialized with an empty model, which is sequentially updated in the following iterations by repeated application of small regression functions to build a final additive model. This thesis examines the resulting estimators and model properties in comparison with other regularization methods such as L_1-penalization. In addition, alternative strategies for improving the variable selection properties of the overall model are proposed and strategies for testing individual effects are developed. For this purpose, variants of variable permutation and bootstrapping methods are developed.

Results

Regularization of linear effect estimators by means of model-based gradient boostings exhibits asymptotic behaviour to L_1-penalization with decreasing learning rate ν if and only if the inverse covariance matrix of the predictor variables is diagonally dominant. Differences between the methods can be traced back to the sequential aggregation of the boosting model, which stabilizes the regularization paths but makes the models relatively larger. Therefore, in order to avoid a large number of false positive selections, the focus can be shifted from the prediction to variable

section accuracy by extending the dataset with permutations of the predictor variables. Residual permutation and parametric bootstrap allow the computation of p-values with test power on par with Wald-tests for maximum likelihood estimators in low-dimensional scenarios.

Practical conclusions

The results of this work provide a guideline for the choice between boosting and L_1-penalty as regularization method for statistical models. In addition, the applicability of model-based gradient-boosting algorithms is improved in situations where more detailled interpretation of the selected variables is of central interest. The reliability of true informative value of selected variables is increased by using alternative tuning via permuted variables. Moreover, making use of the parametric bootstrap allows for the first time the calculation of p-values for single effect estimators of gradient boosting algorithms in high dimensional scenarios with correlated predictor variables.

Einleitung

Eine Brücke zwischen zwei Kulturen

I keep saying the sexy job in the next ten years will be statisticians. People think I'm joking, but who would've guessed that computer engineers would've been the sexy job of the 1990s? The ability to take data – to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it – that's going to be a hugely important skill in the next decades, not only at the professional level but even at the educational level for elementary school kids, for high school kids, for college kids. Because now we really do have essentially free and ubiquitous data. So the complimentary scarce factor is the ability to understand that data and extract value from it.

Ich sage immer wieder, dass in den nächsten zehn Jahren einer der attraktivsten Berufe der des Statistikers sein wird. Die Leute glauben ich mache Witze, aber wer hätte geahnt, dass dies in den 90er Jahren auf Computeringenieure zutreffen würde? Die Befähigung Daten zu erfassen – diese zu verstehen, sie zu verarbeiten, zu verwerten, zu visualisieren und zu kommunizieren – all das wird in den nächsten Jahrzehnten eine enorm wichtige Fähigkeit sein, nicht nur im Beruf sondern auch in der Ausbildung für Grundschüler, Oberschüler, Studenten. Denn heute haben wir im Grunde genommen frei verfügbare und allgegenwärtige Daten. Der ergänzende Schlüsselpunkt ist also die Fähigkeit, diese Daten zu verstehen und daraus Werte zu gewinnen.

– Hal Varian [1]

Seit Beginn diesen Jahres ist dieser Kommentar von Googles Chefökonom Hal Varian nun genau 10 Jahre alt. Vor allem der erste Satz gelangte schnell zu großer Aufmerksamkeit und verhalf dem Zitat somit zu einiger Bekanntheit. Im Rückblick lässt sich

nun wohl tatsächlich feststellen, dass diese Prophezeiung weitgehend eingetroffen ist. Allerdings mit einer großen Einschränkung: die gemeinsame Erwähnung der Wörter Statistik und "sexy" führt heute allgemein immer noch eher zum Schmunzeln als zu euphorischer Zustimmung. Die von Varian beschriebenen Aufgabenfelder der Datenanalyse finden sich heute vielmehr unter den Profilen von *Data Scientists* zusammen mit Kompetenzen im Bereich *Big Data* und *Machine Learning* – von traditionellen statistischen Verfahren hingegen ist eher selten die Rede.

In der Tat gab es in den letzten Jahren eine Vielzahl von Diskussionen und kontroverse Standpunkte zu möglichen Gemeinsamkeiten und Trennlinien zwischen der Statistik und dem Feld der "Datenwissenschaften". Häufig scheint jedoch neben der immer noch unklaren Definition dessen, was einen Data Scientist genau ausmacht, auch eine Reihe von Voreingenommenheiten bezüglich der jeweils anderen Berufsbezeichnung die Diskussion zu verkomplizieren. Dabei sind es nachvollziehbarerweise häufig Verfechter des neuen Begriffes, die sich um eine stärkere Abgrenzung zur "traditionellen" Statistik bemühen indem unter anderem auf größere Aufgaben- und Kompetenzbereiche im Vergleich zur angeblich stur auf die Theorie fokussierten Statistik verwiesen wird. Exemplarisch für die andere Seite hingegen steht die Position Nate Silvers, eines amerikanischen Statistikers und Journalisten, der vor allem durch erfolgreiche Baseballanalysen und Wahlprognosen auf sich aufmerksam gemacht hat. In einer Fragerunde des *Joint Statistical Meeting 2013* brachte er – möglicherweise in direkter Anlehnung an Hal Varians Kommentar – folgende Meinung zum Ausdruck:

> *I think data scientist is a sexed up term for a statistician. Statistics is a branch of science. Data scientist is slightly redundant in some way and people shouldn't berate the term statistician.*
>
> *Ich denke, dass "Data Scientist" ein aufpolierter Begriff für Statistiker ist. Statistik ist ein Zweig der Wissenschaft. Datenwissenschaft ist also in gewisser Weise redundant und man sollte die Bezeichnung Statistiker nicht geringschätzen.*
>
> – Nate Silver [2]

Wenig überraschend kommt solch eine Aussage vor dem Publikum einer Statistikkonferenz denkbar gut an. Um jedoch zu verstehen, warum dieser Standpunkt teilweise nicht nur Anklang findet, sondern sich zuweilen durchaus auch eine gewisse Skepsis gegenüber diesen Entwicklungen bemerken lässt, ist es notwendig sich stärker auf den Bereich zu konzentrieren, welcher wohl unstrittigerweise den Kern beider Berufsgruppen ausmacht: die *Datenanalyse*.

Tatsächlich verlaufen die eigentlichen Konfliktlinien der Diskussion zwischen dem, was Leo Breiman bereits 2001 als die "zwei Kulturen der statistischen Modellierung" beschreibt [3]. Im Fokus der klassischen statistischen Modelle stand lange der Versuch, unter Zuhilfenahme mathematischer Verteilungsannahmen und den dazugehörigen Parametern sowie bestimmten Annahmen über die Struktur des Problems die Mechanismen möglicher Zusammenhänge anhand der erhobenen Daten zu beschreiben und zu interpretieren. Der Erfolg der Datenwissenschaften hingegen scheint eng verwoben mit den eingangs erwähnten Methoden des *maschinellen Lernens* (ML). Dieser Sammelbegriff beinhaltet unterschiedliche Algorithmen, die anhand eines vorhandenen Datensatzes bestimmte Zielvorgaben so effizient wie möglich lösen und dabei nur vergleichsweise wenige Annahmen benötigen. Im Gegensatz zu herkömmlichen statistischen Modellen besteht das Hauptinteresse jedoch nicht in der Interpretation der Datenstruktur. Das zentrale Gütekriterium stellt vielmehr die Vorhersagegenauigkeit dar, d.h. die Fähigkeit eines auf vorhandenen Daten "trainierten" Algorithmus neue Beobachtungen korrekt zu klassifizieren oder, im Falle kontinuierlicher Zielgrößen, möglichst genau zu treffen. Obwohl die einzelnen Arbeitsschritte innerhalb der Iterationen eines ML-Algorithmus oft zu nicht unerheblichen Teilen auf statistischen Konzepten wie beispielsweise Korrelationen beruhen, bleibt die Struktur oder der Mechanismus in den Daten bewusst unspezifiziert, um dem Modell die größtmögliche Flexibilität z.B. im Hinblick auf Nicht-Linearitäten oder Interaktionen zwischen verschiedenen Einflussgrößen zu erhalten. Zwar lassen sich somit unter Umständen deutlich bessere Vorhersagen erreichen, das Ergebnis ist aber meist eine sogenannte *Black Box*, d.h. die konkrete Interpretation der Rolle einzelner Variablen kann nicht mehr so einfach vollzogen werden.

Diese knappe Gegenüberstellung mag somit zwar einerseits mögliche Vorbehalte gegenüber maschinellen Lernverfahren erklären, zeigt allerings auch, dass der eigentliche Unterschied in erster Linie in der Zielsetzung der Analyse zu bestehen scheint. Während es aus Sicht mancher Statistiker daher durchaus so wirken kann, als ob es sich bei bestimmten Verfahren des ML lediglich um Formen der explorativen Datenanalyse oder – noch schlimmer – sogenanntes "data mining" mit all den damit verbundenen Problemen handelt, können hartgesottene ML-Fürsprecher wohl nur schwer nachvollziehen, weshalb man sich durch die in ihren Augen sehr restriktiven und unrealistischen Modellannahmen von vornherein so stark einschränken sollte und die Aufgabe komplizierter macht als sie eigentlich ist. Tom Fawcett und Drew Hardin veranschaulichen dieses Dilemma mit einer treffenden Metapher:

[ML and statistics] are like two pairs of old men sitting in a park playing two different board games. Both games use the same type of board and the same set of pieces, but each plays by different rules and has a different goal because the games are fundamentally different. Each pair looks at the other's board with bemusement and thinks they're not very good at the game.

[ML und Statistik] sind wie zwei Paare alter Männer, die in einem Park sitzen und zwei unterschiedliche Brettspiele spielen. Beide verwenden das gleiche Spielbrett und den gleichen Figurensatz, aber jedes spielt nach unterschiedlichen Regeln und hat ein anderes Ziel, da die Spiele grundsätzlich unterschiedlich sind. Jedes Paar schaut mit Belustigung auf das Brett der anderen und denkt, dass diese nicht sehr gut in dem Spiel sind.

– Tom Fawcett & Drew Hardin [4]

Das interessante an dieser Metapher ist, dass sich das Bild des Brettspiels konstruktiv weiterdenken lässt. So unterschiedlich die Regeln sein mögen, den übergeordneten Rahmen gibt das Spielbrett vor – und das ist in beiden Fällen identisch. Unabhängig davon, was das Ziel der Analyse ist, sind sowohl maschinelle Lernverfahren als auch herkömmliche statistische Methoden an die Qualität und den Informationsgehalt der Datengrundlage gebunden. Dies berechtigt die Frage, ob es daher nicht deutlich sinnvoller wäre, das jeweils andere Spielgeschehen mit Interesse und Neugier statt Belustigung oder gar Ablehnung zu verfolgen. Auch wenn die Siegbedingungen der Spiele unterschiedlich sind, wäre es nicht denkbar, dass ich aus den Spielzügen der anderen etwas lernen, sie vielleicht sogar indirekt übertragen kann oder diese zumindest die Perspektive auf mein eigenes Spiel erweitern? Tatsächlich gibt es solche Brückenschläge zwischen den Perspektiven, was wohl auf kaum ein weiteres Verfahren so deutlich zutrifft wie auf *modellbasierte Gradient-Boosting Algorithmen.*

Das "hypothesis-boosting" Problem

Boosting hat seinen Ursprung im maschinellen Lernen und der bereits Ende der 80er Jahre des letzten Jahrhunderts gestellten Frage, ob die Existenz eines "schwachen Lerners" für ein bestimmtes Problem auch die eines "starken Lerners" für dieses Problem impliziert [5, 6]. Der Begriff "Lerner" bezeichnet hierbei eine Entscheidungsregel, die genutzt wird um die Klassen bzw. Kategorien einer Zielvariable Y zu bestimmen. Während ein schwacher Lerner bei der Unterscheidung zwischen zwei

Klassen unter Umständen nur ein wenig besser abschneidet als ein Münzwurf, kann ein starker Lerner idealerweise nahezu alle Kategorien korrekt zuordnen. Bereits kurze Zeit später gelang es Robert Shapire diese kurz als "hypothesis-boosting" Problem bezeichnete Frage positiv zu beantworten [7], was den später auf den entsprechenden Erkenntnissen aufbauenden Verfahren zu der Bezeichnung "Boosting-Algorithmen" verhalf.

Die Grundidee besteht dabei darin einen unter Umständen recht schwachen *base-learner* $h(\cdot)$ wiederholt an die vorhandenen "Trainingsdaten" anzupassen und ihn immer wieder mit seinen bisherigen Fehlern zu konfrontieren. Dieses Konzept lässt sich in Anlehnung an Bühlmann und Hothorn allgemein so formulieren [8]:

$$\begin{aligned}
(y_1, \boldsymbol{x}_1), \ldots, (y_n, \boldsymbol{x}_n) &\xrightarrow{base-learner} \hat{h}^{[1]}(\cdot) \\
\text{Umgewichtete Daten 1} &\xrightarrow{base-learner} \hat{h}^{[2]}(\cdot) \\
\text{Umgewichtete Daten 2} &\xrightarrow{base-learner} \hat{h}^{[3]}(\cdot) \\
&\ldots \\
\text{Umgewichtete Daten } M-1 &\xrightarrow{base-learner} \hat{h}^{[M]}(\cdot)
\end{aligned}$$

Die Zielvariable y_i stellt Realisationen der eindimensionalen qualitativen Zufallsvariable Y mit zwei Klassen $\{-1, 1\}$ dar, während der Vektor der Prädiktorvariablen $\boldsymbol{x}_i$ aus der p-dimensionalen Zufallsvariable X realisiert wird, wobei alle $i = 1, \ldots, n$ Beobachtungen unabhängig und identisch verteilt sind. Die erste Anwendung des base-learners an die Originaldaten resultiert in die geschätzten Klassen $\hat{h}^{[1]}(\cdot)$, welche nur zu einem bestimmten Teil mit den tatsächlichen Ausprägungen in $\boldsymbol{y}$ übereinstimmen. In jeder darauffolgenden Iteration eines entsprechenden Algorithmus werden die Daten nun so umgewichtet, dass diejenigen Beobachtungen, die zuvor falsch klassifiziert wurden, anschließend stärker gewichtet werden und der base-learner somit eine bessere Strategie für diese Fälle finden muss. Wiederholt man das Verfahren M-mal, können die Ergebnisse als gewichtete Summe aggregiert werden:

$$\hat{f}(\cdot) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m \hat{h}^{[m]}(\cdot)\right)$$

Somit entscheiden am Ende alle Iterationen gemeinsam über die final vergebene Klasse. Die Gewichte α_m werden dafür aus der Fehlklassifikationsrate des base-learners in der entsprechenden Iteration m gebildet, so dass erfolgreichere Itera-

tionen am Ende ein höheres Gewicht bei der Bestimmung der finalen Klassifikation bekommen.

Einer der ersten und bekanntesten Boosting-Algorithmen, der auf dem oben beschriebenen, aber allgemein gehaltenen Prinzip aufbaut, ist *AdaBoost*, kurz für "adaptive boosting" [9]. Adaptiv bedeutet an dieser Stelle, dass sowohl die Gewichte für die Umgewichtung der Daten als auch α_m direkt anhand der Erfolgsquote des base-learners in der aktuellen Iteration bestimmt werden und sich so sequentiell aufeinander aufbauend weiterentwickeln. In einer Vielzahl von Klassifikationsproblemen hat sich dieses Vorgehen bereits unter Verwendung einfacher base-learner dann auch als erstaunlich effektive Methode erwiesen, um Verzerrungen und Abweichungen zu reduzieren und die Fehlklassifikationsraten zu verbessern [10].

Modellbasiertes Gradient-Boosting

Der Erfolg von AdaBoost verhalf der Idee des Boosting schnell zu steigender Bekannt- und Beliebtheit auch über die Grenzen des maschinellen Lernens hinaus. Versuche, diesen Erfolg zu erklären, führten zu der wegweisenden Erkenntnis von Friedmann, Hastie und Tibshirani, die belegen konnten, dass die Stärke des AdaBoost-Algorithmus auf den statistischen Grundprinzipien der *Additivität* und *Maximum-Likelihood* beruht [11]. Über dieses Verständnis von AdaBoost als einer Art additiver logistischer Regression wurde somit der Grundstein für eine statistische Perspektive auf Boosting gelegt. Aufbauend auf früheren Erkenntnissen von Breimann und Mason [12, 13, 14] verschob sich diesbezüglich der Fokus von der Umgewichtung der Daten auf die Berechnung des negativen Gradienten einer Verlustfunktion $\rho(\cdot)$ [15]:

$$-\frac{\partial \rho(\boldsymbol{y}, f)}{\partial f}\Big|_{f=\hat{f}^{[m-1]}(\cdot)} \xrightarrow{\text{base-learner}} \hat{h}^{[m]}(\cdot)$$

Darüber hinaus diskutiert Friedmann die Notwendigkeit einer *learning rate* bzw. Schrittlänge ν, durch die bei der abschließenden Summierung aller base-learner eine Überanpassung an die Stichprobe, sogenanntes *Overfitting*, vermieden werden soll. In jeder Iteration wird demnach nur ein kleiner Teil des base-learners zur bisherigen Summe hinzugefügt:

$$\hat{f}^{[m]}(\cdot) = \hat{f}^{[m-1]}(\cdot) + \nu \cdot \hat{h}^{[m]}(\cdot)$$

Eine abschließende Gewichtung der base-learner findet bei diesem Vorgehen nicht mehr statt. Unter Verwendung der quadratischen Verlustfunktion lässt sich zudem die Analogie zur ursprünglichen Boosting-Idee veranschaulichen, da der negative

Gradient

$$\boldsymbol{u} = -\frac{\partial \left(\boldsymbol{y} - f(\cdot)\right)^2}{\partial f(\cdot)} = 2\left(\boldsymbol{y} - f(\cdot)\right) = 2\boldsymbol{\epsilon}$$

und somit bis auf den Faktor 2 den Residuen $\boldsymbol{\epsilon}$ entspricht, also dem Teil einer Zielvariable $\boldsymbol{y} \in \mathbb{R}$ der noch nicht durch $f(\cdot)$ erklärt wird. In jeder weiteren Iteration des Algorithmus wird nun ein weiterer Teil der Residuen erklärt, so dass sich der Fokus immer stärker in Richtung der schwer zu erklärenden Bereiche bewegt.

Obwohl Friedmann bereits den Vergleich zu Regressionsverfahren mit schrittweiser Variablenselektion zieht, beziehen sich viele der vorgeschlagenen Algorithmen noch auf baumbasierte base-leaner, d.h. Entscheidungs- oder Regressionsbäume. Im Zuge der Erweiterung des Boosting-Konzeptes auf *smoothing splines* schlugen Bühlmann und Yu schließlich "komponentenweises" Boosting vor [16]. Anstelle eines einzelnen base-learners $h(\cdot)$ werden dabei innerhalb jeder Iteration mehrere base-learner parallel an den Gradienten angepasst, wobei typischerweise jeder der p Prädiktorvariablen $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p$ ein base-leaner $h_j(\boldsymbol{x}_j)$ zugeordnet wird. In jeder Iteration m wird dann nur noch der base-learner für das Update der gewichteten Summe verwendet, der den Gradienten im Sinne von

$$\underset{1 \leq j \leq p}{\arg\min} \sum_{i=1}^{n} \left(u_i - \hat{h}_j^{[m]}(x_{ij})\right)^2$$

am Besten erklärt. Setzt man all diese Schritte zusammen, führt dies zu der in Algorithmus 1 beschriebenen Vorgehensweise. Die für diesen Ansatz verwendete Bezeichnung des "modellbasierten" Gradient-Boosting geht zurück auf Hothorn und Bühlmann [17] und bezieht sich auf den Umstand, dass sich durch diese Modifikation der ursprünglichen Boosting-Idee eine Vielzahl unterschiedlicher statistischer Modelle schätzen lässt. Als Verlustfunktion kann hierbei beispielsweise auf die bereits aus dem Kontext der verallgemeinterten linearen Regression bekannten Log-Likelihood-Funktionen zurückgegriffen werden. Wählt man anschließend für die verschiedenen einzelnen base-learner $h_j(\boldsymbol{x}_j)$ selbst Regressionsfunktionen, werden diese durch den Algorithmus schließlich zu dem additiven Prädiktor bzw. Modell der Form

$$f(\boldsymbol{x}_i) = h_1(x_{i1}) + h_2(x_{i2}) + \cdots + h_p(x_{ip})$$

zusammengeführt. Bei unterschiedlicher Lauflänge des Algorithmus bis zur finalen Iteration m_{stop} bedeutet dies allerdings, dass nicht zwangsläufig alle base-learner am Ende in den Prädiktor eingehen. Durch diese *Variablenselektion* bietet modellbasiertes Gradient-Boosting somit eine einfache Möglichkeit zur datengetriebenen

Algorithmus 1 Modellbasiertes Gradient-Boosting

Setze $m := 0$ und initialisiere $\hat{f}^{[0]}(\boldsymbol{x}_i) = \arg\max_c \sum_{i=1}^{n} \rho\left(y_i, f(\boldsymbol{x}_i) = c\right)$.

1. Solange $m < m_{\text{stop}}$, setze $m := m + 1$ und berechne den negativen Gradienten der Verlustfunktion:
$$u_i = -\frac{\partial \rho(y_i, f)}{\partial f}\Big|_{f=\hat{f}^{[m-1]}(\boldsymbol{x}_i)}$$

2. Passe jeden base-learner $h_j(\boldsymbol{x}_j)$ separat an den Vektor des negativen Gradienten $\boldsymbol{u}$ an.

3. Finde $\hat{h}_{j^*}^{[m]}(\boldsymbol{x}_{j^*})$, d.h. den base-learner mit der besten Anpassung:
$$j^* = \arg\min_{1 \leq j \leq p} \sum_{i=1}^{n} \left(u_i - \hat{h}_j^{[m]}(x_{ij})\right)^2$$

4. Verwende einen kleinen Anteil $0 \leq \nu \leq 1$ dieser Komponente um den Prädiktor zu erweitern:
$$\hat{f}^{[m]}(\boldsymbol{x}_i) = \hat{f}^{[m-1]}(\boldsymbol{x}_i) + \nu \cdot \hat{h}_{j^*}^{[m]}(x_{ij^*})$$

Unterscheidung zwischen informativen und nicht-informativen Einflussgrößen. Ein weiterer Vorteil ist die Anwendbarkeit der Methode in sogenannten *hochdimensionalen* Szenarien, in denen die Anzahl der Prädiktorvariablen p größer ist als die der Beobachtungen n.

Diese Vorteile führten in den folgenden Jahren zu einer Vielzahl an Weiterentwicklungen im Hinblick auf unterschiedliche Verlustfunktionen und base-learner (siehe Mayr et al. für eine umfangreiche Zusammenfassung [18, 19]). Eine der wohl spannendsten Erweiterungen stellt dabei Boosting für Verteilungsregression in Form sogenannter *Generalized Additive Models for Location, Scale and Shape* (GAMLSS) dar [20, 21]. Aufbauend auf klassischen generalisierten additiven Modellen [22] verfolgt diese Modellklasse das Ziel, zusätzlich zum Erwartungswert der Zielvariablen weitere Verteilungsparameter wie beispielsweise die Varianz bei Normalverteilung (vgl. Abbildung 1) oder dem Dispersionsparameter bei Negativ-Binominalverteilung durch die Einflussgrößen zu modellieren. Somit lassen sich unter Umständen problematische Phänomene wie Heteroskedastie oder Überdispersion nicht nur direkt in der Verteilungsannahme adressieren, sondern auch deren konkrete Zusammenhänge in den Daten beschreiben und quantifizieren.

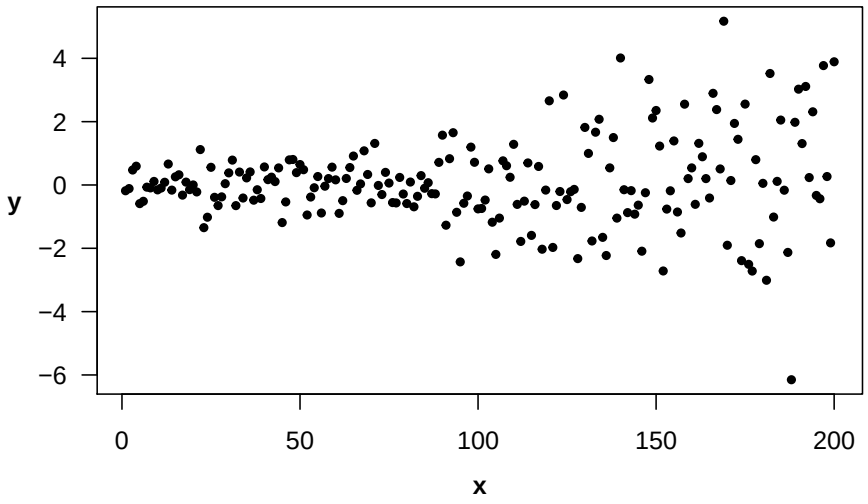

Abbildung 1: Einfaches Simulationsbeispiel einer heteroskedastisch normalverteilten Zielgröße. Die Einflussgröße x hat nur einen positiven Effekt auf die Varianz, jedoch nicht auf den Erwartungswert von y.

Ziele der Arbeit

Durch die Verwendung von Gradient-Boosting Algorithmen beim Schätzen statistischer Modelle ergeben sich einige Unterschiede im Vergleich zu Modellen, die mit Verfahren wie der Kleinst-Quadrate-Methode oder Maximum-Likelihood-Verfahren geschätzt werden. Einer der zentralen Vorteile ist die bereits beschriebene Möglichkeit, mittels Boosting auch klassische statistische Modelle an hochdimensionale Daten mit $p > n$ anzupassen, d.h. Datensätzen bei denen die Anzahl der Prädiktorvariablen größer ist als die Anzahl der Beobachtungen. Dies ist deshalb möglich, da innerhalb jeder Iteration die base-learner einzeln und unabhängig voneinander an den negativen Gradienten angepasst werden. Da diese jedoch nur schrittweise und nacheinander in das Gesamtmodell eingehen, führt das zu dem als *Shrinkage* bezeichneten Phänomen: Je weniger Iterationen der Algorithmus durchläuft, desto stärker sind die Schätzer in der Regel Richtung Null verzerrt. Interessanterweise führt diese Eigenschaft dazu, dass modellbasierte Gradient-Boosting Algorithmen als implizit regularisierte Regressionsmodelle interpretiert werden können.

Dadurch drängt sich der Vergleich mit alternativen Regularisierungsverfahren auf (vgl. Abbildung 2). Der prominenteste Vertreter ist hierbei die explizite Penalisierung der L_1-Norm der Koeffizienten durch den sogenannten *least absolute shrinkage and selection operator*, kurz *Lasso* [23]. Tatsächlich wurde bereits eine verblüffende Ähnlichkeit zwischen den Regularisierungspfaden des Lasso und der *incremental*

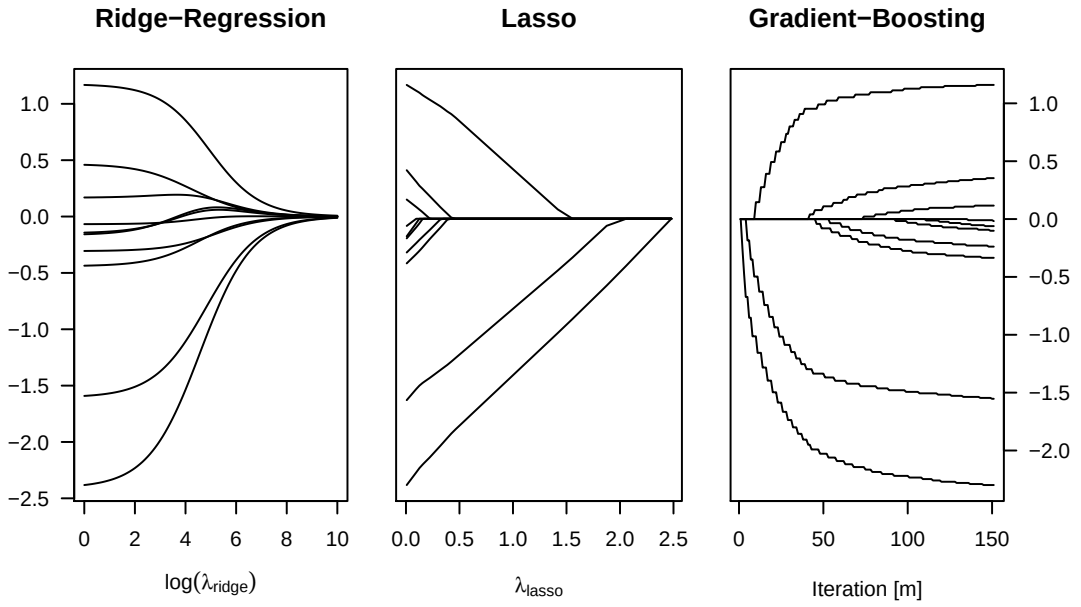

Abbildung 2: Beispiele unterschiedlicher Regularisierungsmethoden in einfachem Simulationsszenario mit $p < n$. Je stärker die Regularisierung, d.h. je größer λ bzw. je kleiner m, desto stärker werden die Koeffizienten Richtung Null geschrumpft.

forward stagewise regression festgestellt, welche mehr oder weniger identisch mit Gradient-Boosting mit quadratischer Verlustfunktion ist. In den folgenden Jahren hat sich somit die etwas vage Auffassung verbreitet, dass es sich bei Gradient-Boosting Algorithmen um so etwas Ähnliches wie L_1-Regularisierung handeln würde [24]. Die erste Arbeit dieser kumulativen Dissertation setzt sich deshalb gezielt mit diesem Vergleich auseinander, wobei neben den Ähnlichkeiten vor allem auch die Unterschiede und deren Konsequenzen unter verschiedenen Tuning-Ansätzen beleuchtet werden.

Tuning bezeichnet im Zusammenhang mit regularisierten Regressionsverfahren, dass der Hyperparameter, der die Stärke der Regularisierung bzw. Penalisierung steuert, auf einen bestimmten sinnvollen Wert festgelegt werden soll, damit das resultierende Modell die gewünschten Eigenschaften besitzt. Der wohl am weitesten verbreitete Ansatz hierfür ist die sogenannte *Kreuzvalidierung*, welche bei einer Vielzahl maschineller Lernverfahren effektiv eingesetzt werden kann. Vor allem bei hochdimensionalen Datensätzen wie beispielsweise Genomstudien muss davon ausgegangen werden, dass viele der als Prädiktorvariablen zur Verfügung stehenden Kandidaten keinen wahren Effekt auf die Zielvariable haben. Will man das finale Modell inhaltlich sinnvoll interpretieren, sollte ein guter Tuning-Ansatz einen modellbasierten Gradient-Boosting Algorithmen daher rechtzeitig stoppen und dadurch idealerwei-

se gerade so regularisieren, dass nicht zu viele dieser nicht-informativen Variablen in das Modell aufgenommen werden. Bisherige Simulationsstudien zeigen jedoch, dass die Anwendung der Kreuzvalidierung bei regularisierten Regressionsmodellen eher zu relativ großen Modellen führt, d.h. die Modelle enthalten häufig viele falsch positive Variablen [25, 26]. Die Ursache hierfür ist, dass Kreuzvalidierung – ganz im Sinne der bereits in der Einleitung beschriebenen Tradition des maschinellen Lernens – in erster Linie versucht die Prognosegüte der Algorithmen zu optimieren. Das bedeutet offenbar, dass die Modelle ein paar stark "geschrumpfte" falsch positive Variablen verkraften können, wenn dafür im Gegenzug die wahrhaftig informativen Variablen weniger stark regularisiert werden. Dies wirkt sich bei Gradient-Boosting Algorithmen nochmal drastischer aus als zum Beispiel beim Lasso, da im Gegensatz zu Letzterem einmal selektierte Variablen nicht mehr aus dem Modell gestrichen werden können. Der zweite Artikel der Dissertation setzt sich mit diesem Problem auseinander und stellt einen alternativen Tuningansatz vor, der den Fokus explizit auf die Optimierung der Variablenselektion statt der Prognosegüte richtet. Dieser wird anschließend mittels Simulationsstudien sowohl anhand des Vergleichs mit Kreuzvalidierung als auch *stability selection* [27, 28], einem deutlich rechenintensiveren Ansatz zur verbesserten Variablenselektion, evaluiert und abschließend auf Daten dreier unterschiedlicher Genexpressionsstudien angewandt [29, 30, 31].

In vielen biomedizinischen Anwendungen ist es jedoch nicht nur sinnvoll, die Zahl falsch positiver Variablen insgesamt niedrig zu halten, sondern es steht die inferenzstatistische Beurteilung eines bestimmten Effektes im Zentrum der Analyse. Während dies in klassischen statistischen Modellen in der Regel über den t-Test für Regressionskoeffizienten erfolgen kann, ist die Berechnung der entsprechenden Teststatistik bei Gradient-Boosting Algorithmen nicht möglich, da keine Standardfehler zu den regularisierten Effektschätzern verfügbar sind. Eine mögliche Alternative in solchen Situationen sind *Permutationstests* [32, 33]. Diese wurden bereits auch im Zusammenhang mit Gradient-Boosting für Verteilungsregression zur Beurteilung der Messgenauigkeit medizintechnischer Geräte untersucht [34]. Tatsächlich zeigen die Ergebnisse dieser Simulationsstudien, dass die aus den Permutationstests resultierenden p-Werte im Rahmen des konkreten Anwendungsfalls in der Lage sind, die Typ-I-Fehlerrate einzuhalten, d.h. die Wahrscheinlichkeit einen aus den Daten geschätzten Effekt einer Variable als signifikant zu bezeichnen, obwohl er es in Wahrheit nicht ist, entspricht in etwa dem zuvor bestimmten Risiko (üblicherweise 5%). Dies setzt jedoch voraus, dass die entsprechende Variable unabhängig von allen anderen Prädiktorvariablen ist, was vermutlich nur in den wenigsten Anwendungsproblemen der Fall sein dürfte. Der dritte Artikel dieser Dissertation beschäftigt

sich damit, durch Abwandlungen und Alternativen zu den vorgeschlagenen Permutationstests diese praktischen Einschränkungen zu überwinden. Konkret sollen dabei Ansätze der Residuenpermutation und der parametrische Bootstrap auf ihre Anwendbarkeit überprüft werden [35, 36, 37]. Neben der zentralen Eigenschaft der Typ-I-Fehlerrate wird zusätzlich die Power der Tests untersucht und mit den Standardtest der Maximum-Likelihood-Schätzung verglichen.

Zusammengefasst sollen die in den Artikeln geleisteten Beiträge dabei helfen, zunächst ein genaueres Verständnis des Regularisierungsverhaltens modellbasierter Gradient-Boosting Algorithmen zu erlangen. Anschließend sollen die Erkenntnisse dem zentralen Anliegen helfen die inferenzstatistischen Eigenschaften der resultierenden Modelle zu verbessern bzw. Möglichkeiten wiederherzustellen, auf die aufgrund der Annäherung an die Kultur der algorithmischen Modellierung bisher noch verzichtet werden muss. Dies kann die Anwendbarkeit von Gradient-Boosting Algorithmen auf Bereiche ausdehnen, in denen diese trotz der daraus gewonnenen Vorteile wie der Variablenselektion und Verfügbarkeit in hochdimensionalen Problemen bisher noch keine brauchbare Alternative zu herkömmlichen Verfahren darstellen. Auf diese Weise könnte ein bereits mächtiges Werkzeug zwischen den Welten der statistischen Modellierung und dem maschinellen Lernen vielleicht dazu beitragen, sich statt aller vermeintlicher Differenzen stärker auf den wenig beachteten Nachsatz Nate Silvers zu konzentrieren:

Just do good work and call yourself whatever you want.	*Mach einfach gute Arbeit und nenn dich wie immer du willst.*

– Nate Silver [38]

Originalpublikationen

Hepp T, Schmid M, Gefeller O, Waldmann E, Mayr A.

Approaches to regularized regression - a comparison between gradient boosting and the lasso.

Methods of Information in Medicine, 55(5):422-430

http://dx.doi.org/10.3414/me16-01-0033

Hepp T, Schmid M, Gefeller O, Waldmann E, Mayr A.

Addendum to: Approaches to regularized regression - a comparison between gradient boosting and the lasso.

Methods of Information in Medicine, 58(1):60

http://dx.doi.org/10.1055/s-0038-1669389

Hindawi
Computational and Mathematical Methods in Medicine
Volume 2017, Article ID 1421409, 8 pages
https://doi.org/10.1155/2017/1421409

Research Article

Probing for Sparse and Fast Variable Selection with Model-Based Boosting

Janek Thomas,[1] Tobias Hepp,[2] Andreas Mayr,[2,3] and Bernd Bischl[1]

[1]*Department of Statistics, LMU München, München, Germany*
[2]*Department of Medical Informatics, Biometry and Epidemiology, FAU Erlangen-Nürnberg, Erlangen, Germany*
[3]*Department of Medical Biometry, Informatics and Epidemiology, University Hospital Bonn, Bonn, Germany*

Correspondence should be addressed to Tobias Hepp; tobias.hepp@uk-erlangen.de

Janek Thomas and Tobias Hepp contributed equally to this work.

Received 9 February 2017; Accepted 13 April 2017; Published 31 July 2017

Academic Editor: Yuhai Zhao

We present a new variable selection method based on model-based gradient boosting and randomly permuted variables. Model-based boosting is a tool to fit a statistical model while performing variable selection at the same time. A drawback of the fitting lies in the need of multiple model fits on slightly altered data (e.g., cross-validation or bootstrap) to find the optimal number of boosting iterations and prevent overfitting. In our proposed approach, we augment the data set with randomly permuted versions of the true variables, so-called shadow variables, and stop the stepwise fitting as soon as such a variable would be added to the model. This allows variable selection in a single fit of the model without requiring further parameter tuning. We show that our probing approach can compete with state-of-the-art selection methods like stability selection in a high-dimensional classification benchmark and apply it on three gene expression data sets.

1. Introduction

At the latest since the emergence of genomic and proteomic data, where the number of available variables p is possibly far higher than the sample size n, high-dimensional data analysis becomes increasingly important in biomedical research [1–4]. Since common statistical regression methods like ordinary least squares are unable to estimate model coefficients in these settings due to singularity of the covariance matrix, varying strategies have been proposed to select only truly influential, that is, informative, variables and discard those without impact on the outcome.

By enforcing sparsity in the true coefficient vector, regularized regression approaches like the *lasso* [5], *least angle regression* [6], *elastic net* [7], and *gradient boosting* algorithms [8, 9] perform variable selection directly in the model fitting process. This selection is controlled by tuning hyperparameters that define the degree of penalization. While these hyperparameters are commonly determined using resampling strategies like cross-validation, bootstrapping, and similar methods, the focus on minimizing the prediction error often results in the selection of many noninformative variables [10, 11].

One approach to address this problem is *stability selection* [12, 13], a method that combines variable selection with repeated subsampling of the data to evaluate selection frequencies of variables. While stability selection can considerably improve the performance of several variable selection methods including regularized regression models in high-dimensional settings [12, 14], its application depends on additional hyperparameters. Although recommendations for reasonable values exist [12, 14], proper specification of these parameters is not straightforward in practice as the optimal configuration would require a priori knowledge about the number of informative variables. Another potential drawback is that stability selection increases the computational demand, which can be problematic in high-dimensional settings if the computational complexity of the used selection technique scales superlinearly with the number of predictor variables.

In this paper, we propose a new method to determine the optimal number of iterations in model-based boosting for variable selection inspired by *probing*, a method frequently used in related areas of machine learning research [15–17] and the analysis of microarrays [18]. The general notion of probing involves the artificial inflation of the data with random noise variables, so-called *probes* or *shadow variables*. While this approach is in principle applicable to the lasso or least angle regression as well, it is especially attractive to use with more computationally intensive boosting algorithms, as no resampling is required at all. Using the first selection of a shadow variable as stopping criterion, the algorithm is applied only once without the need to optimize any hyperparameters in order to extract a set of informative variables from the data, thereby making its application very fast and simple in practice. Furthermore, simulation studies show that the resulting models in fact tend to be more strictly regularized compared to the ones resulting from cross-validation and contain less uninformative variables.

In Section 2, we provide detailed descriptions of the model-based gradient boosting algorithm as well as stability selection and the new probing approach. Results of a simulation study comparing the performance of probing to cross-validation and different configurations of stability selection in a binary classification setting are then presented in Section 3 before discussing the application of these methods on three data sets with measurements of gene expression levels in Section 4. Section 5 summarizes our findings and presents an outlook to extensions of the algorithm.

2. Methods

2.1. Gradient Boosting. Given a learning problem with a data set $D = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1,\ldots,n}$ sampled i.i.d. from a distribution over the joint space $\mathcal{X} \times \mathcal{Y}$, with a p-dimensional input space $\mathcal{X} = (\mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_p)$ and an output space $\mathcal{Y}$ (e.g., $\mathcal{Y} = \mathbb{R}$ for regression and $\mathcal{Y} = \{0, 1\}$ for binary classification), the aim is to estimate a function, $f(\mathbf{x})$, $\mathcal{X} \to \mathcal{Y}$, that maps elements of the input space to the output space as good as possible. Relying on the perspective on boosting as gradient descent in function space, gradient boosting algorithms try to minimize a given loss function, $\rho(y^{(i)}, f(\mathbf{x}^{(i)}))$, $\rho : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}$, that measures the discrepancy between a predicted outcome value of $f(\mathbf{x}^{(i)})$ and the true $y^{(i)}$. Minimizing this discrepancy is achieved by repeatedly fitting weak prediction functions, called *base learners*, to previous mistakes, in order to combine them to a strong ensemble [19]. Although early implementations in the context of machine learning focused specifically on the use of regression trees, the concept has been successfully extended to suit the framework of a variety of statistical modelling problems [8, 20]. In this model-based approach, the *base learners* $h(\mathbf{x})$ are typically defined by semiparametric regression functions on $\mathbf{x}$ to build an additive model. A common simplification is to assume that each base learner h_j is defined on only one component x_j of the input space

$$f(\mathbf{x}) = \beta_0 + h_1(x_1) + \cdots + h_p(x_p). \tag{1}$$

For an overview of the fitting process of model-based boosting see Algorithm 1.

Algorithm 1 (model-based gradient boosting). Starting at $m = 0$ with a constant loss minimal initial value $\widehat{f}^{[0]}(\mathbf{x}) \equiv c$, the algorithm iteratively updates the predictor with a small fraction of the base learner with the best fit on the negative gradient of the loss function:

(1) Set iteration counter $m := m + 1$.

(2) While $m \leq m_{\text{stop}}$, compute the negative gradient vector of the loss function:

$$u^{(i)} = -\left.\frac{\partial \rho(y, f)}{\partial f}\right|_{f=\widehat{f}^{[m-1]}(\mathbf{x}^{(i)}), y=y^{(i)}}. \tag{2}$$

(3) Fit every base learner $h_j^{[m]}(x_j)$ separately to the negative gradient vector $\mathbf{u}$.

(4) Find $\widehat{h}_{j^*}^{[m]}(\mathbf{x}_{j^*})$, that is, the base learner with the best fit:

$$j^* = \arg\min_{1 \leq j \leq p} \sum_{i=1}^{n} \left(u^{(i)} - \widehat{h}_j^{[m]}\left(x_j^{(i)}\right)\right)^2. \tag{3}$$

(5) Update the predictor with a small fraction $0 \leq \nu \leq 1$ of this component:

$$\widehat{f}(\mathbf{x})^{[m]} = \widehat{f}(\mathbf{x})^{[m-1]} + \nu \cdot \widehat{h}_{j^*}^{[m]}\left(x_{j^*}\right). \tag{4}$$

The resulting model can be interpreted as a generalized additive model with partial effects for each covariate contained in the additive predictor. Although the algorithm relies on two hyperparameters ν and m_{stop}, Bühlmann and Hothorn [9] claim that the *learning rate* ν is of minor importance as long as it is "sufficiently small," with $\nu = 0.1$ commonly used in practice.

The stopping criterion, m_{stop}, determines the degree of regularization and thereby heavily affects the model quality in terms of overfitting and variable selection [21]. However, as already outlined in the introduction, optimizing m_{stop} using common approaches like cross-validation results in the selection of many uninformative variables. Although still focusing on minimizing prediction error, using a 25-fold bootstrap instead of the commonly used 10-fold cross-validation tends to return sparser models without sacrificing prediction performance [22].

2.2. Stability Selection. The weak performance of cross-validation regarding variable selection partly results from the fact that it pursues the goal of minimizing the prediction error instead of selecting only informative variables. One possible solution is the *stability selection* framework [12, 13], a very versatile algorithm that can be combined with all kinds of variable selection methods like gradient boosting, lasso, or forward stepwise selection. It produces sparser solutions by controlling the number of false discoveries. Stability selection defines an upper bound for the per-family error rate (PFER),

for example, the expected number of uninformative variables $\mathbb{E}(V)$ included in the final model.

Therefore, using stability selection with model-based boosting means that Algorithm 1 is run independently on B random subsamples of the data until either a predefined number of iterations m_{stop} is reached or q different variables have been selected. Subsequently, all variables are sorted with respect to their selection frequency in the B sets. The amount of informative variables is then determined by a user-defined threshold π_{thr} that has to be exceeded. A detailed description of these steps is given in Algorithm 2.

Algorithm 2 (stability selection for model-based boosting [14]).

(1) For $b = 1, \ldots, B$,

  (a) draw a subset of size $\lfloor n/2 \rfloor$ from the data;
  (b) fit a boosting model to the subset until the number of selected variables is equal to q or the number of iterations reaches a prespecified number (m_{stop}).

(2) Compute the selection frequencies per variable j:

$$\widehat{\pi}_j := \frac{1}{B} \sum_{b=1}^{B} \mathbb{I}_{\{j \in \widehat{S}_b\}}, \tag{5}$$

where $\widehat{S}_b$ denotes the set of selected variables in iteration b.

(3) Select variables with a selection frequency of at least π_{thr}, which yields a set of stable covariates:

$$\widehat{S}_{\text{stable}} := \left\{ j : \widehat{\pi}_j \geq \pi_{\text{thr}} \right\}. \tag{6}$$

Following this approach, the upper bound for the PFER can be derived as follows [12]:

$$\mathbb{E}(V) \leq \frac{q^2}{(2\pi_{\text{thr}} - 1)\, p}. \tag{7}$$

With additional assumptions on exchangeability and shape restrictions on the distribution of simultaneous selection, even tighter bounds can be derived [13]. While this method is successfully applied in a large number of different applications [23–26], several shortcomings impede the usage in practice. First off, three additional hyperparameters π_{thr}, PFER, and q are introduced. Although only two of them have to be specified by the user (the third one can be calculated by assuming equality in (7)), it is not intuitively clear which parameter should be left out and how to specify the remaining two. Even though recommendations for reasonable settings for the selection threshold [12] or the PFER [14] are proposed, the effectiveness of these settings is difficult to evaluate in practical settings. The second obstacle in the usage of stability selection is the considerable computational power required for calculation. Overall B boosting models ([13] recommends $B = 100$) have to be fitted and a reasonable m_{stop} has to be found as well, which will most likely require cross-validation. Even though this process can be parallelized quite easily, complex model classes with smooth and higher-order effects can become extremely costly to fit.

2.3. Probing. The approach of adding *probes* or *shadow variables*, for example, artificial uninformative variables to the data, is not completely new and has already been investigated in some areas of machine learning. Although they share the underlying idea to benefit from the presence of variables that are known to be independent from the outcome, the actual implementation of the concept differs (see Guyon and Elisseeff (2003) [15] for an overview). An especially useful approach, however, is to generate these additional variables as randomly shuffled versions of all observed variables. These permuted variables will be called *shadow variables* for the remainder of this paper and are denoted as $\tilde{x}_j$. Compared to adding randomly sampled variables, shadow variables have the advantage that the marginal distribution of x_j is preserved in $\tilde{x}_j$. This approach is tightly connected to the theory of permutation tests [27] and is used similarly for *all-relevant* variable selection with random forests [28].

Implementing the *probing* concept to the sequential structure of model-based gradient boosting is rather straightforward. Since boosting algorithms proceed in a greedy fashion and only update the effect which yields the largest loss reduction in each iteration, selecting a shadow variable essentially implies that the best possible improvement at this stage relies on information that is known to be unrelated to the outcome. As a consequence, variables that are selected in later iterations are most likely correlated to y only by chance as well. Therefore, all variables that have been added prior to the first shadow variable are assumed to have a true influence on the target variable and should be considered informative. A description of the full procedure is presented in Algorithm 3.

Algorithm 3 (probing for variable selection in model-based boosting).

(1) *Expand* the data set X by creating randomly shuffled images $\tilde{x}_j$ for each of the $j = 1, \ldots, p$ variables x_j such that

$$\tilde{x}_j \in S_{x_j}, \tag{8}$$

where S_{x_j} denotes the symmetric group that contains all $n!$ possible permutations of x_j.

(2) *Initialize* a boosting model on the inflated data set

$$\overline{X} = \left[x_1 \cdots x_p \; \tilde{x}_1 \cdots \tilde{x}_p \right] \tag{9}$$

and start iterations with $m = 0$.

(3) *Stop if* the first $\tilde{x}_j$ is selected; see Algorithm 1 step (3).

(4) *Return* only the variables selected from the original data set X.

The major advantage of this approach compared to variable selection via cross-validation or stability selection is that one model fit is enough to find informative variables and no expensive refitting of the model is required. Additionally, there is no need for any prespecification like the search space (m_{stop}) for cross-validation or additional hyperparameters (q, π_{thr}, PFER) for stability selection. However, it should be noted that, unlike classical cross-validation, probing aims at optimal variable selection instead of prediction performance of the algorithm. Since this usually involves stopping much earlier, the effect estimates associated with the selected variables are most likely strongly regularized and might not be optimal for predictions.

3. Simulation Study

In order to evaluate the performance of our proposed variable selection method, we conduct a benchmark simulation study where we compare the set of nonzero coefficients determined by the use of shadow variables as stopping criterion to cross-validation and different configurations of stability selection. We simulate n data points for p variables from a multivariate normal distribution $X \sim \mathcal{N}(0, \Sigma)$ with Toeplitz correlation structure $\Sigma_{ij} = \rho^{|i-j|}$ for all $1 < i, j < p$ and $\rho = 0.9$. The response variable $y^{(i)}$ is then generated by sampling Bernoulli experiments with probability

$$\pi^{(i)} = \frac{\exp\left(\eta^{(i)}\right)}{1 + \exp\left(\eta^{(i)}\right)}, \tag{10}$$

with $\eta^{(i)}$ the linear predictor for the ith observation $\eta^{(i)} = X^{(i)}\beta$ and all nonzero elements of β sampled from $\mathcal{U}(-1, 1)$. Since the total amount of nonzero coefficients determines the number of informative variables in the setting, it is denoted as p_{inf}.

Overall, we consider 12 different simulation scenarios defined by all possible combinations of $n \in \{100, 500\}$, $p \in \{100, 500, 1000\}$, and $p_{\text{inf}} \in \{5, 20\}$. Specifically, this leads to the evaluation of 2 low-dimensional settings with $p < n$, 4 settings with $p = n$, and 6 high-dimensional settings with $p > n$. Each configuration is run 100 times. Along with new realizations of X and y, we also draw new values for the nonzero coefficients in β and sample their position in the vector in each run to allow for varying correlation patterns among the informative variables. For variable selection with cross-validation, 25-fold bootstrap (the default in `mboost`) is used to determine the final number of iterations. Different configurations of stability selection were tested to investigate whether and, if so, to what extent these settings affect the selection. In order to explicitly use the upper error bounds of stability selection, we decided to specify 9 combinations with PFER $\in \{1, 2.5, 8\}$ and $\pi_{\text{thr}} \in \{0.6, 0.75, 0.9\}$ and calculate q from (7). Aside from the learning rate ν, which is set to 0.1 for all methods, no further parameters have to be specified for the probing scheme. Two performance measures are considered for the evaluation of the methods with respect to variable selection: first, the true positive rate (TPR) as the fraction of (correctly) selected variables from all true informative variables and, second, the false discovery rate (FDR) as the fraction of uninformative variables in the set of selected variables. To ensure reproducibility the R package `batchtools` [29] was used for all simulations.

The results of the simulations for all settings are illustrated in Figure 1. With TPR and FDR on the y-axis and x-axis, respectively, solutions displayed in the top left corner of the plots therefore successfully separate p_{inf} informative variables from the ones without true effect on the response. Although already using a sparse cross-validation approach, the FDR of variable selection via cross-validation is still relatively high, with more than 50% false positives in the selected sets in the majority of the simulated scenarios. Whereas this seems to be mostly disadvantageous in the cases where $p_{\text{inf}} = 5$, the trend to more greedy solutions leads to a considerably higher chance of identifying more of the truly informative variables if $p_{\text{inf}} = 20$ or with very high p, however, still at the price of picking up many noise variables on the way. Pooling the results of all configurations considered for stability selection, the results cover a large area of the performance space in Figure 1, thereby probably indicating high sensitivity on the decisions regarding the three tuning parameters.

Examining the results separately in Figure 2, the dilemma is particularly clearly illustrated for $p_{\text{inf}} = 20$ and $n = 500$. Despite being able to control the upper bounds for expected false positive selections, only a minority of the true effects are selected if the PFER is set too conservative. In addition, the high variance of the FDR observed for these configurations in some settings somewhat counteracts the goal to achieve more certainty about the selected variables one might probably pursue by setting the PFER very low. The performance of probing, on the other hand, reveals a much more stable pattern and outperforms stability selection in the difficult $p_{\text{inf}} = 20$ and $n = 100$ settings. In fact, the TPR is either higher or similar to all configurations used for stability selection, but exhibiting slightly higher FDR especially in settings with $n = 500$. Interestingly, probing seems to provide results similar to those of stability selection with PFER = 8, raising the question if the use of shadow variables allows statements about the number of expected false positives in the selected variable set.

Considering the runtime, however, we can see that probing is orders of magnitudes faster with an average runtime of less than a second compared to 12 seconds for cross-validation and almost one minute for stability selection.

4. Application on Gene Expression Data

In this section we exploit the usage of probing as a tool for variable selection on three gene expression data sets. More specifically, this includes data from using oligonucleotide arrays for colon cancer detection [30] with 40 tumor and 22 regular colon tissue samples and $p = 2000$ measured genes expression levels. In addition, we analyse data from a study aiming to predict metastasis of breast carcinoma [31], where patients were labelled good or poor ($n = 111$ and $n = 57$, resp.) depending on whether they remained event-free for a five-year period after diagnosis or not. The data set contains log-transformed expression levels of $p =$

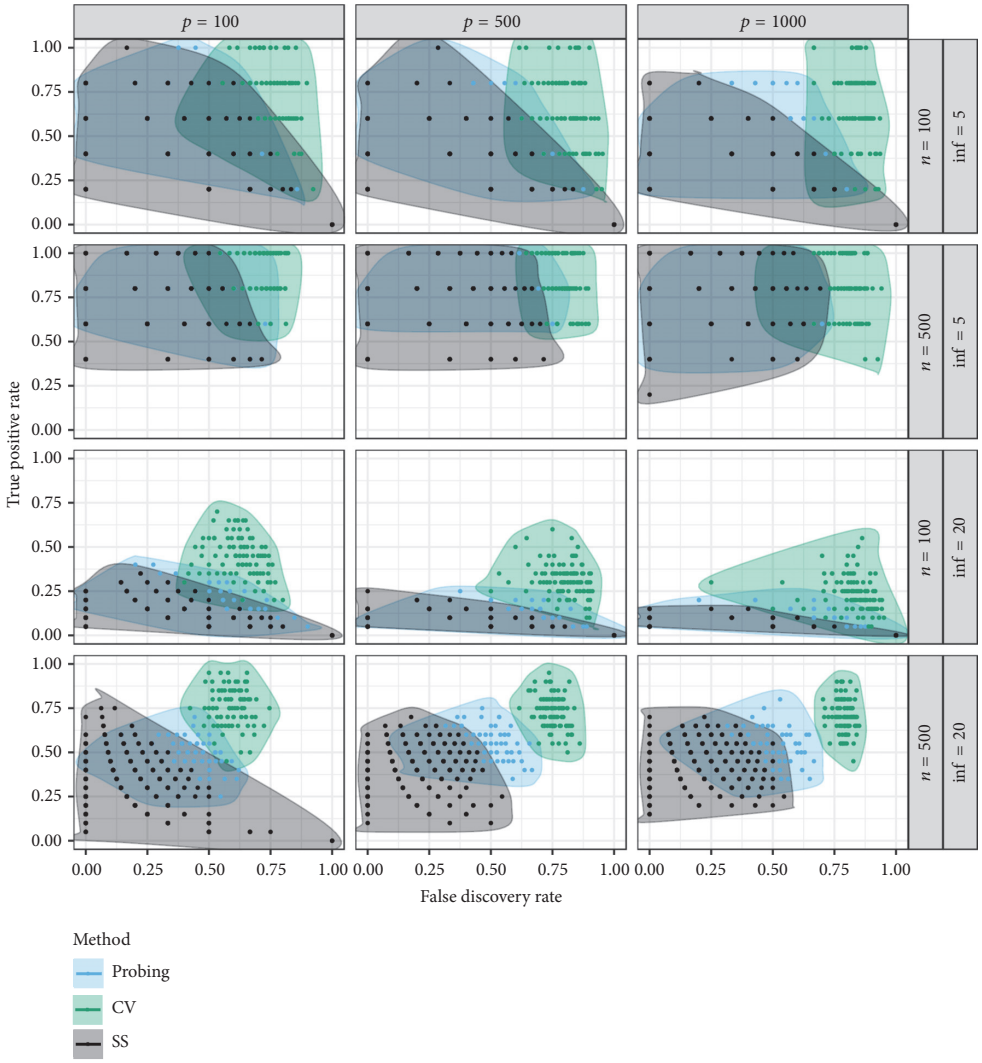

Figure 1: True positive rate (on y-axis) and false discovery rate (on x-axis) for three different, boosting-based variable selection algorithms, probing (black), stability selection (green), cross-validation (blue), and different simulation settings: $n \in \{100, 500\}$, $p \in \{100, 500, 1000\}$, and $p_{\text{inf}} \in \{5, 20\}$. All settings of stability selection are combined. Shaded areas are smooth hulls around all observed values.

2905 genes. The last example examines riboflavin production by *Bacillus subtilis* [32] with $n = 71$ observations of log-transformed riboflavin production rates and expression level for $p = 4088$ genes. All data are publicly available via `R` packages `datamicroarray` and `hdi`. Our proposed probing approach is implemented in a fork of the `mboost` [33] software for component-wise gradient boosting. It can be easily used by setting `probe=TRUE` in the `glmboost()` call.

In order to evaluate the results provided by the new approach, we analysed the data using cross-validation, stability selection [34], and the lasso [35] for comparison. Table 1 shows the total number of variables selected by each method along with the size of the intersection between the sets. Starting with the probably least surprising result, boosting with cross-validation leads to the largest set of selected variables in all examples, whereas using probing as stopping criterion instead clearly reduces these sets. Since both approaches are based on the same regularization profile until the first shadow variable enters the model, the less regularized solution of cross-validation always contains all variables selected with probing. For stability selection, we used the conservative approach with PFER = 1 and $q = 20$ as suggested by Bühlmann et al. (2014) [32]. As a consequence, the set of variables considered to be informative further

Table 1: Total number of selected variables and intersection size for four variable selection techniques (boosting with 25-fold bootstrap, probing, stability selection, and the lasso with 10-fold cross-validation) on three gene expression data sets. The last column compares algorithm runtime in seconds.

	Cross-validation	Probing	Stability selection	Lasso (`glmnet`)	Runtime (sec.)
			Colon cancer		
Cross-validation	**9**				10.52
Probing	5	**5**			1.78
Stability selection	3	3	**3**		49.4
Lasso (`glmnet`)	7	5	3	**7**	0.4
			Breast carcinoma		
Cross-validation	**32**				24
Probing	14	**14**			4.39
Stability selection	1	1	**1**		102.28
Lasso (`glmnet`)	14	14	1	**14**	1.13
			Riboflavin production		
Cross-validation	**50**				14.2
Probing	10	**10**			6.89
Stability selection	5	5	**5**		66.46
Lasso (`glmnet`)	23	7	4	**30**	0.68

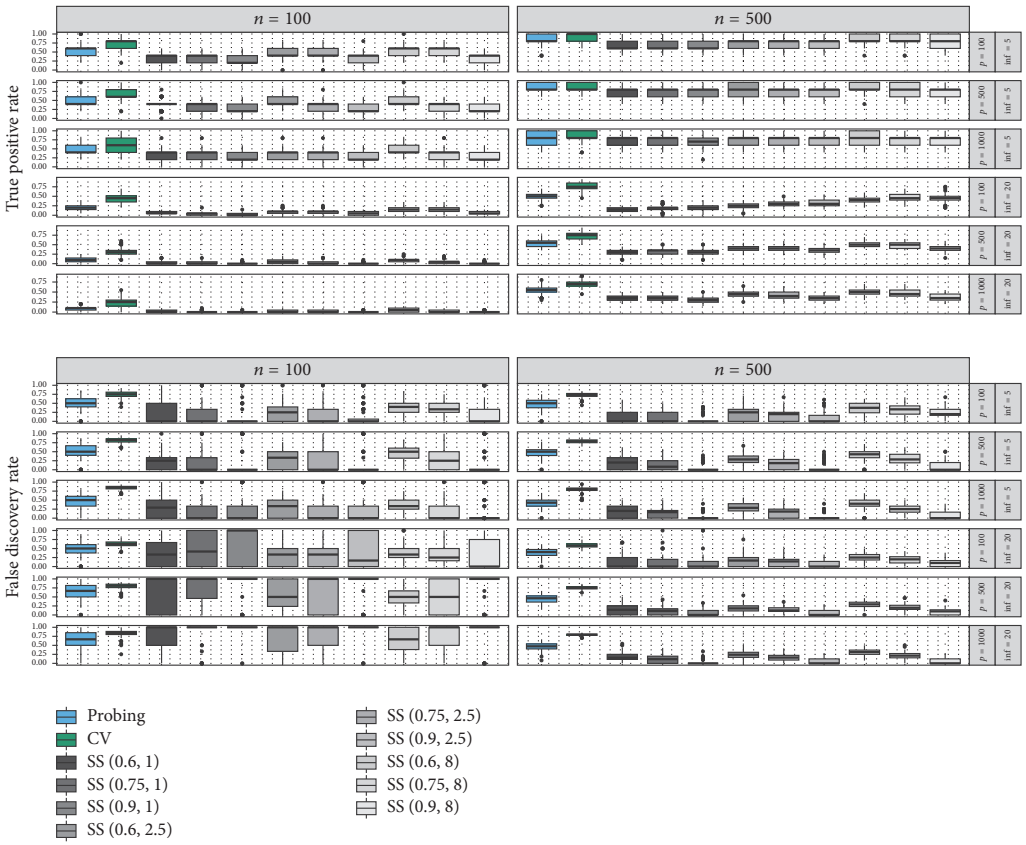

Figure 2: Boxplots of true positive rate (top) and false discovery rate (bottom) for different simulation settings and the three boosting-based, variable selection algorithms. Different Stability selection settings are denoted by SS(π_{thr}, PFER).

shrinks in all three scenarios. Again, these results clearly reflect the findings from the simulation study in Section 3, placing the probing approach between stability selection with probably overly conservative error bound and the greedy selection with cross-validation.

Since so far all approaches rely on boosting algorithms, we additionally considered variable selection with the lasso. We used the default settings of the `glmnet` package for R to calculate the lasso regularization path and determine the final model via 10-fold cross-validation [35]. Although the lasso already tends to result in sparser models under these conditions compared to model-based boosting [22], `glmnet` additionally uses a "one-standard-error rule" to regularize the solution even further. In fact, this leads to the selection of an identical set of genes as probing for the breast carcinoma example, but the final models estimated for both other examples still contain a higher number of variables. This is especially the case for the data on riboflavin production, where the lasso solution is further not simply a subset of the cross-validated boosting approach and only agrees on 23 mutually selected variables. Interestingly, even one of the 5 variables proposed by stability selection is also missing. The R code used for this analysis can be found in the Supplementary Material of this manuscript available online at https://doi.org/10.1155/2017/1421409.

5. Conclusion

We proposed a new approach to determine the optimal number of iterations for sparse and fast variable selection with model-based boosting via the addition of probes or shadow variables (*probing*). We were able to demonstrate via a simulation study and the analysis of gene expression data that our approach is both a feasible and convenient strategy for variable selection in high-dimensional settings. In contrast to common tuning procedures for model-based boosting which rely on resampling or cross-validation procedures to optimize the prediction accuracy [21], our probing approach directly addresses the variable selection properties of the algorithm. As a result, it substantially reduces the high number of false discoveries that arise with standard procedures [14] while only requiring a single model fit to obtain the set of parameters.

Aside from the very short runtime, another attractive feature of probing is that no additional tuning parameters have to be specified to run the algorithm. While this greatly increases its ease of use, there is, of course, a trade-off regarding flexibility, as the lack of tuning parameters means that there is no way to steer the results towards more or less conservative solutions. However, a corresponding tuning approach in the context of probing could be to allow a certain amount of selected probes in the model before deciding to stop the algorithm (cf. Guyon and Elisseeff, 2003 [15]). Although variables selected after the first probe can be labelled informative less convincingly, this resembles the uncertainty that comes with specifying higher values for the error bound of stability selection.

A potential drawback of our approach is that due to the stochasticity of the permutations, there is no deterministic solution and the selected set might slightly vary after rerunning the algorithm. In order to stabilize results, probing could also be used combined with resampling to determine the optimal stopping iteration for the algorithm by running the procedure on several bootstrap samples first. Of course, this requires the computation of multiple models and therefore again increases the runtime of the whole selection procedure.

Another promising extension could be a combination with stability selection. With each model stopping at the first shadow variable, only the selection threshold π_{thr} has to be specified. However, since this means a fundamental change of the original procedure, further research on this topic is necessary to better assess how this could affect the resulting error bound.

While in this work we focused on gradient boosting for binary and continuous data, there is no reason why our results should not also carry over to other regression settings or related statistical boosting algorithms as likelihood-based boosting [36]. Likelihood-based boosting follows the same principle idea but uses different updates, coinciding with gradient boosting in case of Gaussian responses [37]. Further research is also warranted on extending our approach to multidimensional boosting algorithms [25, 38], where variables have to be selected for various models simultaneously.

In addition, probing as a tuning scheme could be generally also combined with similar regularized regression approaches like the lasso [5, 22]. Our proposal for model-based boosting hence could be a starting point for a new way of tuning algorithmic models for high-dimensional data, not with the focus on prediction accuracy, but addressing directly the desired variable selection properties.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work of authors Tobias Hepp and Andreas Mayr was supported by the Interdisciplinary Center for Clinical Research (IZKF) of the Friedrich-Alexander-University Erlangen-Nürnberg (Project J49). The authors additionally acknowledge support by Deutsche Forschungsgemeinschaft and Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) within the funding programme Open Access Publishing.

References

[1] R. Romero, J. Espinoza, F. Gotsch et al., "The use of high-dimensional biology (genomics, transcriptomics, proteomics, and metabolomics) to understand the preterm parturition syndrome," *BJOG: An International Journal of Obstetrics and Gynaecology*, vol. 113, no. s3, pp. 118–135, 2006.

[2] R. Clarke, H. W. Ressom, A. Wang et al., "The properties of high-dimensional data spaces: implications for exploring gene and protein expression data," *Nature Reviews Cancer*, vol. 8, no. 1, pp. 37–49, 2008.

[3] P. Mallick and B. Kuster, "Proteomics: a pragmatic perspective," *Nature Biotechnology*, vol. 28, no. 7, pp. 695–709, 2010.

[4] M. L. Bermingham, R. Pong-Wong, A. Spiliopoulou et al., "Application of high-dimensional feature selection: evaluation for genomic prediction in man," *Scientific Reports*, vol. 5, Article ID 10312, 2015.

[5] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society B*, vol. 58, no. 1, pp. 267–288, 1996.

[6] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[7] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, vol. 67, no. 2, pp. 301–320, 2005.

[8] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[9] P. Bühlmann and T. Hothorn, "Boosting algorithms: regularization, prediction and model fitting," *Statistical Science*, vol. 22, no. 4, pp. 477–505, 2007.

[10] N. Meinshausen and P. Bühlmann, "High-dimensional graphs and variable selection with the lasso," *The Annals of Statistics*, vol. 34, no. 3, pp. 1436–1462, 2006.

[11] C. Leng, Y. Lin, and G. Wahba, "A note on the lasso and related procedures in model selection," *Statistica Sinica*, vol. 16, no. 4, pp. 1273–1284, 2006.

[12] N. Meinshausen and P. Bühlmann, "Stability selection," *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, vol. 72, no. 4, pp. 417–473, 2010.

[13] 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*, vol. 75, no. 1, pp. 55–80, 2013.

[14] B. Hofner, L. Boccuto, and M. Göker, "Controlling false discoveries in high-dimensional situations: boosting with stability selection," *BMC Bioinformatics*, vol. 16, no. 1, article 144, 2015.

[15] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[16] J. Bi, K. P. Bennett, M. Embrechts, C. M. Breneman, and M. Song, "Dimensionality reduction via sparse support vector machines," *Journal of Machine Learning Research*, vol. 3, pp. 1229–1243, 2003.

[17] Y. Wu, D. D. Boos, and L. A. Stefanski, "Controlling variable selection by the addition of pseudovariables," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 235–243, 2007.

[18] V. G. Tusher, R. Tibshirani, and G. Chu, "Significance analysis of microarrays applied to the ionizing radiation response," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 9, pp. 5116–5121, 2001.

[19] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics, Springer New York Inc., New York, NY, USA, 2001.

[20] G. Ridgeway, "The state of boosting," *Computing Science and Statistics*, vol. 31, pp. 172–181, 1999.

[21] A. Mayr, B. Hofner, and M. Schmid, "The importance of knowing when to stop: a sequential stopping rule for component-wise gradient boosting," *Methods of Information in Medicine*, vol. 51, no. 2, pp. 178–186, 2012.

[22] T. Hepp, M. Schmid, O. Gefeller, E. Waldmann, and A. Mayr, "Approaches to regularized regression—a comparison between gradient boosting and the lasso," *Methods of Information in Medicine*, vol. 55, no. 5, pp. 422–430, 2016.

[23] A.-C. Haury, F. Mordelet, P. Vera-Licona, and J.-P. Vert, "TIGRESS: trustful inference of gene regulation using stability selection," *BMC Systems Biology*, vol. 6, article 145, 2012.

[24] S. Ryali, T. Chen, K. Supekar, and V. Menon, "Estimation of functional connectivity in fMRI data using stability selection-based sparse partial correlation with elastic net penalty," *NeuroImage*, vol. 59, no. 4, pp. 3852–3861, 2012.

[25] J. Thomas, A. Mayr, B. Bischl, M. Schmid, A. Smith, and B. Hofner, Stability selection for component-wise gradient boosting in multiple dimensions, 2016.

[26] A. Mayr, B. Hofner, and M. Schmid, "Boosting the discriminatory power of sparse survival models via optimization of the concordance index and stability selection," *BMC Bioinformatics*, vol. 17, no. 1, article 288, 2016.

[27] H. Strasser and C. Weber, "The asymptotic theory of permutation statistics," *Mathematical Methods of Statistics*, vol. 8, no. 2, pp. 220–250, 1999.

[28] M. B. Kursa, A. Jankowski, and W. . Rudnicki, "Boruta—a system for feature selection," *Fundamenta Informaticae*, vol. 101, no. 4, pp. 271–285, 2010.

[29] M. Lang, B. Bischl, and D. Surmann, "batchtools: Tools for R to work on batch systems," *The Journal of Open Source Software*, vol. 2, no. 10, 2017.

[30] U. Alon, N. Barka, D. A. Notterman et al., "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 96, no. 12, pp. 6745–6750, 1999.

[31] E. Gravier, G. Pierron, A. Vincent-Salomon et al., "A prognostic DNA signature for T1T2 node-negative breast cancer patients," *Genes Chromosomes and Cancer*, vol. 49, no. 12, pp. 1125–1134, 2010.

[32] P. Bühlmann, M. Kalisch, and L. Meier, "High-dimensional statistics with a view toward applications in biology," *Annual Review of Statistics and Its Application*, vol. 1, no. 1, pp. 255–278, 2014.

[33] T. Hothorn, P. Buehlmann, T. Kneib, M. Schmid, and B. Hofner, mboost: Model-Based Boosting. R package version R package version 2.7-0, 2016.

[34] B. Hofner and T. Hothorn, stabs: Stability Selection with Error Control. R package version R package version 0.5-1, 2015.

[35] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010.

[36] G. Tutz and H. Binder, "Generalized additive modeling with implicit variable selection by likelihood-based boosting," *Biometrics*, vol. 62, no. 4, pp. 961–971, 2006.

[37] A. Mayr, H. Binder, O. Gefeller, and M. Schmid, "The evolution of boosting algorithms: From machine learning to statistical modelling," *Methods of Information in Medicine*, vol. 53, no. 6, pp. 419–427, 2014.

[38] 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*, vol. 61, no. 3, pp. 403–427, 2012.

Hindawi
Computational and Mathematical Methods in Medicine
Volume 2018, Article ID 2430438, 1 page
https://doi.org/10.1155/2018/2430438

Corrigendum

Corrigendum to "Probing for Sparse and Fast Variable Selection with Model-Based Boosting"

Janek Thomas[1], **Tobias Hepp**[2], **Andreas Mayr**[2,3] **and Bernd Bischl**[1]

[1]*Department of Statistics, LMU München, München, Germany*
[2]*Department of Medical Informatics, Biometry and Epidemiology, FAU Erlangen-Nürnberg, Erlangen, Germany*
[3]*Department of Medical Biometry, Informatics and Epidemiology, University Hospital Bonn, Bonn, Germany*

Correspondence should be addressed to Tobias Hepp; tobias.hepp@uk-erlangen.de

Received 10 May 2018; Accepted 16 May 2018; Published 5 July 2018

In the article titled "Probing for Sparse and Fast Variable Selection with Model-Based Boosting" [1], an Acknowledgment should be added as follows:

Acknowledgments

"Tobias Hepp performed the present work in partial fulfilment of the requirements for obtaining the degree 'Dr. rer. biol. hum' at the University of Erlangen-Nuremberg."

References

[1] J. Thomas, T. Hepp, A. Mayr, and B. Bischl, "Probing for sparse and fast variable selection with model-based boosting," *Computational and Mathematical Methods in Medicine*, Art. ID 1421409, 8 pages, 2017.

README.R

```
###### Contents ####

# For questions regarding this document or any of the code please contact.
# Janek Thomas - janek.thomas@stat.uni-muenchen.de and
# Tobias Hepp - tobias.hepp@fau.de

## Code ##
# defs.R - Definitions for the simulation study
# simulation.R - Actual code that runs the simulation study
# reduce.R - Reduce results of simulation stufy to create simulation.RData
# application.R - Code to rerun the gene-expression application

## data ##
# registry - Registry containing all information + Results of the simulation,
#            is created by simulation.R
# simulation.RData - Reduces registry, created by running reduce.R


## Required packages ##
install.packages(c("batchtools", "devtools", "stabs", "mvtnorm", "glmnet", "stringi", "hdi"))
install_github("ja-thomas/mboost") #mboost fork containing the probing code
```

defs.R

```
library(batchtools)
library(mboost)
library(stabs)
library(mvtnorm)

OVERWRITE = FALSE

# settings cluster
WALLTIME = 2L * 60L
MEMORY = 1024L
NTASKS = 1L

#settings data generation

P = c(100, 500, 1000)
N = c(100, 500)
PINF = c(5, 20)
TOEPBASE = 0.9

# setting stabsel
PI = c(0.6, 0.75, 0.9)
PFER = c(1, 2.5, 8)


REPLICATIONS = 100L
```

simulation.R

```
source("defs.R")

if (file.exists("registry")) {
  if (OVERWRITE) {
    unlink("registry_large", recursive = TRUE)
    reg = makeExperimentRegistry(seed = 123L,
      packages = c("mboost", "stabs", "mvtnorm"))
  } else {
    reg = loadRegistry("registry")
  }
} else {
  reg = makeExperimentRegistry("registry_large", seed = 123L,
    packages = c("mboost", "stabs", "mvtnorm"))
}

reg$default.resources = list(walltime = WALLTIME, memory = MEMORY, ntasks = NTASKS)
reg$max.concurrent.jobs = 1000L

#### create problems ####

for (p in P) {
  i = 1
  for (n in N) {
    j = 1
    for (pinf in PINF) {
      k = 1
      addProblem(name = paste0("P", p, "N", n, "Inf", pinf), seed = as.numeric(paste0(i,j,k)),
        data = list(p = p, n = n, pinf = pinf, toep = TOEPBASE),
        fun = function(data, job){
          p = data$p
          n = data$n
          pinf = data$pinf
          toep = data$toep

          sigma = toeplitz(sapply(seq_len(p) - 1, function(x) toep^x))
          data = data.frame(rmvnorm(n = n + 5000, sigma = sigma))

          betas = runif(n = pinf, min = -1, max = 1)
          xinf = sample(colnames(data), size = pinf)
          eta = as.matrix(data[, xinf]) %*% betas
          y = rbinom(n, 1, round(exp(eta) / (1 + exp(eta))))
          data$y = y

          train = data[1:n,]
          test = data[(n+1):nrow(data), ]

          # irrepresentable condition(
          x = as.matrix(train)
          C = (t(x) %*% x)/n
          ind = which(colnames(train) %in% xinf)
          nICviolation = sum((abs(C[-ind, ind] %*% solve(C[ind, ind]) %*% sign(betas))) >= 1)

          list(train = train, test = test, xinf = xinf, beta = beta, nICviolation = nICviolation)
        })
```

```
      k = k + 1
    }
    j = j + 1
  }
  i = i + 1
}

#### create algorithms ####

probing = function(job, data, instance) {

  train = instance$train
  test = instance$test
  xinf = instance$xinf

  time = Sys.time()
  mod = glmboost(y ~ ., data = train, control = boost_control(mstop = 500L), probe = TRUE)
  runtime = Sys.time() - time
  units(runtime) = "secs"

  # tpr & fpr
  sel = unique(variable.names(mod)[selected(mod)])
  tpr = length(intersect(xinf, sel))/length(xinf)
  fpr =  length(setdiff(sel, xinf)) / (length(variable.names(mod)) - length(xinf) - 1)

  #Prediction error
  preds = round(predict(mod, newdata = test, type = "response"))
  mmce = mean(preds[, 1] != test$y)

  mod2 = glm(as.formula(paste("y ~", paste(sel, collapse = "+"))), data = train, family = binomial)
  preds = round(predict(mod2, newdata = test, type = "response"))
  mmce_mle = mean(preds != test$y)

  list(tpr = tpr, fpr = fpr, runtime = runtime, mmce = mmce, mmce_mle = mmce_mle, mstop = mstop(mod),
       nICviolation = instance$nICviolation)
}

crossval = function(job, data, instance) {

  train = instance$train
  test = instance$test
  xinf = instance$xinf

  time = Sys.time()
  mod = glmboost(y ~ ., data = train, control = boost_control(mstop = 5 * nrow(train)))
  opt = cvrisk(mod, papply = lapply)
  mstop(mod) = mstop(opt)
  runtime = Sys.time() - time
  units(runtime) = "secs"

  # tpr & fpr
  sel = unique(variable.names(mod)[selected(mod)])
  tpr = length(intersect(xinf, sel))/length(xinf)
  fpr =  length(setdiff(sel, xinf)) / (length(variable.names(mod)) - length(xinf) - 1)

  #Prediction error
```

```
  preds = round(predict(mod, newdata = test, type = "response"))
  mmce = mean(preds[, 1] != test$y)

  mod2 = glm(as.formula(paste("y ~", paste(sel, collapse = "+"))), data = train, family = binomial)
  preds = round(predict(mod2, newdata = test, type = "response"))
  mmce_mle = mean(preds != test$y)

  list(tpr = tpr, fpr = fpr, runtime = runtime, mmce = mmce, mmce_mle = mmce_mle, mstop = mstop(mod),
       nICviolation = instance$nICviolation)
}

stabilitySelection = function(job, data, instance, pi, pfer) {

  train = instance$train
  test = instance$test
  xinf = instance$xinf

  #run stabsel
  time = Sys.time()
  mod = glmboost(y ~., data = train, control = boost_control(mstop = 5 * nrow(train)))
  res = stabsel(mod, cutoff = pi, PFER = pfer, papply = lapply)
  runtime = Sys.time() - time
  units(runtime) = "secs"

  # tpr & fpr
  sel = names(selected(res))

  if (length(sel) == 0) {
    return(list(tpr = 0, fpr = 0, runtime = runtime, mmce = 0.5, mmce_mle = 0.5, mstop = 0,
                nICviolation = instance$nICviolation))
  }

  tpr = length(intersect(xinf, sel))/length(xinf)
  fpr = length(setdiff(sel, xinf)) / (length(variable.names(mod)) - length(xinf) - 1)


  #Prediction Error
  #mle = glm(as.formula(paste("y ~", names(selected(res)))), data = train, family = binomial())
  mstop(mod) = max(1, which.min(selected(mod) %in% selected(res)) - 1)
  preds = round(predict(mod, newdata = test, type = "response"))
  mmce = mean(preds[, 1] != test$y)

  mod2 = glm(as.formula(paste("y ~", paste(sel, collapse = "+"))), data = train, family = binomial)
  preds = round(predict(mod2, newdata = test, type = "response"))
  mmce_mle = mean(preds != test$y)

  list(tpr = tpr, fpr = fpr, runtime = runtime, mmce = mmce, mmce_mle = mmce_mle, mstop = mstop(mod),
       nICviolation = instance$nICviolation)
}

addAlgorithm(reg = reg, name = "probing", fun = probing)
addAlgorithm(reg = reg, name = "crossval", fun = crossval)
addAlgorithm(reg = reg, name = "stabsel", fun = stabilitySelection)

addExperiments(reg = reg, algo.designs = list(
  probing = data.frame(),
```

```
  stabsel = expand.grid(pi = PI, pfer = PFER),
  crossval = data.frame()),
  repls = REPLICATIONS)

summarizeExperiments()
submitJobs()
```

reduce.R

```
library(batchtools)
source("defs.R")

reg = loadRegistry("registry/", work.dir = ".")

full_res = list()

for(e in getProblemIds()) {

  p = as.numeric(strsplit(e, split = "P|N")[[1]][2])
  n = as.numeric(strsplit(e, split = "N|I")[[1]][2])
  inf = as.numeric(strsplit(e, split = "Inf")[[1]][2])

  #probing
  res = reduceResultsList(findExperiments(prob.name = e, algo.name = "probing"))
  probing = data.frame(do.call(rbind, res), p = p, n = n, inf = inf, method = "probing")

  #cv
  res = reduceResultsList(findExperiments(prob.name = e, algo.name = "crossval"))
  cv = data.frame(do.call(rbind, res), p = p, n = n, inf = inf, method = "crossval")

  #stabsel
  pars = expand.grid(PI, PFER)
  res = do.call(rbind, mapply(function(p, pf) {
    res = reduceResultsList(findExperiments(prob.name = e, algo.name = "stabsel",
                                            algo.pars = pi == p && pfer == pf))
    data.frame(do.call(rbind, res), method = paste0("stabsel_pi", p, "_pfer", pf))},
    p = pars[,1], pf = pars[,2], SIMPLIFY = FALSE))
  stabs = data.frame(res, p = p, n = n, inf = inf)

  full_res[[e]] = rbind(probing, cv, stabs)
}

data = do.call(rbind, full_res)
# something strange is happening here
data$tpr = unlist(data$tpr)
data$fpr = unlist(data$fpr)
data$runtime = unlist(data$runtime)
data$mmce = unlist(data$mmce)
data$mmce_mle = unlist(data$mmce_mle)
data$mstop = unlist(data$mstop)

library(ggplot2)
pdf("results.pdf")
ggplot(data = data) + geom_boxplot(aes(y = tpr, x = method, fill = method), position = "dodge") +
```

```
  facet_grid(p + inf ~ n) + ggtitle("tpr")
ggplot(data = data) + geom_boxplot(aes(y = fpr, x = method, fill = method), position = "dodge") +
  facet_grid(p + inf ~ n) + ggtitle("fpr")
ggplot(data = data) + geom_boxplot(aes(y = runtime, x = method, fill = method), position = "dodge") +
  facet_grid(p + inf ~ n) + ggtitle("runtime")
ggplot(data = data) + geom_boxplot(aes(y = mmce, x = method, fill = method), position = "dodge") +
  facet_grid(p + inf ~ n) + ggtitle("mmce")
ggplot(data = data) + geom_boxplot(aes(y = mmce_mle, x = method, fill = method), position = "dodge") +
  facet_grid(p + inf ~ n) + ggtitle("mmce_mle")
ggplot(data = data) + geom_boxplot(aes(y = mstop, x = method, fill = method), position = "dodge") +
  facet_grid(p + inf ~ n) + ggtitle("mstop")
dev.off()


library(dplyr)
library(tidyr)
library(aplpack)

data %>%
  filter(method %in% c("probing", "crossval", "stabsel_pi0.75_pfer2.5")) %>%
  group_by(p, n, inf, method) %>%
  select(tpr, fpr) %>%
  summarise(firstfpr = quantile(fpr, probs = 0.25), thirdfpr = quantile(fpr, probs = 0.75),
    firsttpr = quantile(tpr, probs = 0.25), thirdtpr = quantile(tpr, probs = 0.75),
    medtpr = quantile(tpr, probs = 0.5), medfpr = quantile(fpr, probs = 0.55)) %>%
  ggplot() + geom_segment(aes(y = firsttpr, x = medfpr, yend = thirdtpr, xend = medfpr,
                              color = method), size = 1) +
  geom_segment(aes(y = medtpr, x = firstfpr, yend = medtpr, xend = thirdfpr,
                   color = method), size = 1) +
  facet_grid(p + inf ~ n, scales = "free") + xlab("fpr") + ylab("tpr")

data %>%
  filter(method %in% c("probing", "crossval", "stabsel_pi0.75_pfer2.5")) %>%
  filter(p == 100, n == 100, inf == 5) %>%
  ggplot(aes(tpr, fpr, colour = method, fill = method)) +
  geom_bag() +
  geom_point() +
 # facet_grid(p + inf ~ n, scales = "free") + xlab("fpr") + ylab("tpr") +
  theme_minimal()
```

application.R

```
library(glmnet)
library(mboost)
library(stringi)

data("riboflavin", package = "hdi")

#glmnet
x = riboflavin[,-1]
colnames(x) = stri_replace_all(colnames(x), replacement = "", fixed = "_")
y = riboflavin[,1]
data = data.frame(cbind(y, x, matrix(sample(x), nrow = 71)))
```

```
fit = glmnet(x = x, y = y)
set.seed(42)
fit.cv = cv.glmnet(x = x, y = y)
b = as.matrix(coef(fit.cv))
b = rownames(b)[b != 0]

b = b[b != "(Intercept)"]

#probing
set.seed(42)
mod = glmboost(y ~ ., data = data.frame(data), probe = TRUE, control = boost_control(mstop = 1000))
b2 = names(coef(mod))
b2 = b2[b2 != "(Intercept)"]

#CV
set.seed(123)
mod = glmboost(y ~ ., data = data.frame(data), control = boost_control(mstop = 500))
opt = cvrisk(mod, papply = lapply)
mstop(mod) = mstop(opt)
b3 = names(coef(mod))
b3 = substring(b3[-1], 2)

#stabsel
set.seed(13)
stabs = stabsel(mod, PFER = 1, q = 20, papply = lapply)
b4 = names(selected(stabs))
b4 = substring(b4, 2)

selections = list(
  glmnet = b,
  probing = b2,
  cv = b3,
  stabsel = b4
)

selectionSimilarities = function(l) {
  m = length(l)
  res = matrix(NA, ncol = m, nrow = m)
  colnames(res) = rownames(res) = names(l)

  for(i in names(l)) {
    for(j in names(l)) {
      res[i,j] = sum(l[[i]] %in% l[[j]])
    }
  }
  res
}

selectionSimilarities(selections)
```

Hepp T, Schmid M, Mayr A.

Significance tests for boosted location and scale models with linear base-learners.

The International Journal of Biostatistics [Epub ahead of print]
https://dx.doi.org/10.1515/ijb-2018-0110

Literaturverzeichnis

[1] McKinsey Insights. Hal Varian on how the web challenges managers; 2009. Aufgerufen: 27.05.2019. https://www.mckinsey.com/industries/high-tech/our-insights/hal-varian-on-how-the-web-challenges-managers.

[2] Statistics Views. Nate Silver: what I need from statisticians; 2013. Aufgerufen: 27.05.2019. https://www.statisticsviews.com/details/feature/5133141/Nate-Silver-What-I-need-from-statisticians.html.

[3] Breiman L. Statistical modeling: the two cultures (with comments and a rejoinder by the author). Stat Sci. 2001;16(3):199–231.

[4] Silicon Valley Data Science. Machine learning vs. statistics; 2017. Aufgerufen: 27.05.2019. https://www.svds.com/machine-learning-vs-statistics/.

[5] Kearns M. Thoughts on hypothesis boosting; 1988. Unpublished.

[6] Kearns MJ, Valiant LG. Cryptographic limitations on learning boolean formulae and finite automata. In: Machine learning: from theory to applications. Springer; 1993. p. 29–49.

[7] Schapire RE. The strength of weak learnability. Mach Learn. 1990;5(2):197–227.

[8] Bühlmann P, Hothorn T, et al. Boosting algorithms: Regularization, prediction and model fitting. Stat Sci. 2007;22(4):477–505.

[9] Freund Y, Schapire RE, et al. Experiments with a new boosting algorithm. In: ICML. vol. 96. Citeseer; 1996. p. 148–156.

[10] Ridgeway G. The state of boosting. Comp Sci Stat. 1999;31:172–181.

[11] Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Ann Stat. 2000;28(2):337–407.

[12] Breiman L. Arcing classifiers (with discussion and a rejoinder by the author). Ann Stat. 1998;26(3):801–849.

[13] Breiman L. Prediction games and arcing algorithms. Neural Comput. 1999;11(7):1493–1517.

[14] Mason L, Baxter J, Bartlett PL, Frean MR. Boosting algorithms as gradient descent. In: Advances in neural information processing systems; 2000. p. 512–518.

[15] Friedman JH. Greedy function approximation: a gradient boosting machine. Ann Stat. 2001;29(5):1189–1232.

[16] Bühlmann P, Yu B. Boosting with the L2 loss: regression and classification. J Am Stat Assoc;.

[17] Hothorn T, Bühlmann P. Model-based boosting in high dimensions. Bioinformatics. 2006;22(22):2828–2829.

[18] Mayr A, Binder H, Gefeller O, Schmid M. The evolution of boosting algorithms. Methods Inf Med. 2014;53(06):419–427.

[19] Mayr A, Hofner B, Waldmann E, Hepp T, Meyer S, Gefeller O. An update on statistical boosting in biomedicine. Comput Math Methods in Med. 2017;2017.

[20] Rigby RA, Stasinopoulos DM. Generalized additive models for location, scale and shape. J R Stat Soc Ser C Appl Stat. 2005;54(3):507–554.

[21] Mayr A, Fenske N, Hofner B, Kneib T, Schmid M. Generalized additive models for location, scale and shape for high dimensional data—a flexible approach based on boosting. J R Stat Soc Ser C Appl Stat. 2012;61(3):403–427.

[22] Hastie T, Tibshirani R. Generalized additive models. Stat Sci. 1986;1(3):297–310.

[23] Tibshirani R. Regression shrinkage and selection via the lasso. J R Stat Soc Series B Stat Methodol. 1996;58:267–288.

[24] Bühlmann P, Gertheiss J, Hieke S, Kneib T, Ma S, Schumacher M, et al. Discussion of "The evolution of boosting algorithms" and "Extending statistical boosting". Methods Inf Med. 2014;53(06):436–445.

[25] Meinshausen N, Bühlmann P. High-dimensional graphs and variable selection with the lasso. Ann Stat. 2006;p. 1436–1462.

[26] Leng C, Lin Y, Wahba G. A note on the lasso and related procedures in model selection. Stat Sin. 2006;16(4):1273–1284.

[27] Meinshausen N, Bühlmann P. Stability selection. J R Stat Soc Series B Stat Methodol. 2010;72(4):417–473.

[28] Hofner B, Boccuto L, Göker M. Controlling false discoveries in high-dimensional situations: boosting with stability selection. BMC Bioinformatics. 2015;16(1):144.

[29] Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, et al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc Natl Acad Sci U S A. 1999;96(12):6745–6750.

[30] Gravier E, Pierron G, Vincent-Salomon A, Gruel N, Raynal V, Savignoni A, et al. A prognostic DNA signature for T1T2 node-negative breast cancer patients. Genes Chromosomes Cancer. 2010;49(12):1125–1134.

[31] Bühlmann P, Kalisch M, Meier L. High-dimensional statistics with a view toward applications in biology. Ann Rev Stat Appl. 2014;1:255–278.

[32] Draper NR, Stoneman DM. Testing for the inclusion of variables in linear regression by a randomisation technique. Technometrics. 1966;8(4):695–699.

[33] Kennedy PE, Cade BS. Randomization tests for multiple regression. Commun Stat Simul Comput. 1996;25(4):923–936.

[34] Mayr A, Schmid M, Pfahlberg A, Uter W, Gefeller O. A permutation test to analyse systematic bias and random measurement errors of medical devices via boosting location and scale models. Stat Methods Med Res. 2017;26(3):1443–1460.

[35] Potter DM. A permutation test for inference in logistic regression with small- and moderate-sized data sets. Stat Med. 2005;24(5):693–708.

[36] Werft W, Benner A. glmperm: A permutation of regressor residuals test for inference in generalized linear models. R J. 2010;2(1):39–43.

[37] Faraway JJ. Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models. vol. 124. CRC press; 2016.

[38] Simply Statistics. Data scientist is just a sexed up word for statistician; 2013. Aufgerufen: 27.05.2019. https://simplystatistics.org/2013/08/08/data-scientist-is-just-a-sexed-up-word-for-statistician/.

Verzeichnis der Vorveröffentlichungen

Hepp T, Schmid M, Mayr A. Significance tests for boosted location and scale models with linear base-learners. Int J Biostat. https://dx.doi.org/10.1515/ijb-2018-0110 [Epub ahead of print]

Saake M, Schmidle A, Kopp M, Hanspach J, Hepp T, Laun F, Nagel A, Dörfler A, Uder M, Bäuerle T. MRI brain signal intensity and relaxation times in individuals with prior exposure to gadobutrol. Radiology, 290(3):659-668

Mayr A, Hofner B, Waldmann E, Hepp T, Meyer S, Gefeller O. An update on statistical boosting in biomedicine. Comput Math Methods Med, vol. 2017, Article ID 6083072

Thomas J, Hepp T, Mayr A, Bischl B. Probing for sparse and fast variable selection with model-based boosting. Comput Math Methods Med, vol. 2017, Article ID 1421409

Faschingbauer F, Heimrich J, Raabe E, Kehl S, Schneider M, Schmid M, Beckmann M, Hepp T, Luebke A, Mayr A, Schild RL. Longitudinal assessment of examiner experience on the accuracy of sonographic fetal weight estimation at term. J Ultrasound Med, 36(1):163-174.

Hepp T, Schmid M, Gefeller O, Waldmann E, Mayr A. Approaches to regularized regression - a comparison between gradient boosting and the lasso. Methods Inf Med, 55(5):422-430