technische universität dortmund

fakultät statistik

Masterarbeit

Vergleich von verschiedenen robusten und nichtrobusten Boosting-Verfahren

Autor: Alina Stammen
Betreuung: Prof. Dr. Christine Müller
Abgabedatum: 22.08.2022

Fakultät Statistik
TU Dortmund

Inhaltsverzeichnis

1 Einleitung

Das Erfassen und Analysieren von Daten ist aus der heutigen Gesellschaft nicht mehr wegzudenken. In vielen Bereichen des öffentlichen Lebens werden Daten gesammelt und verarbeitet, um unter anderem Prozesse besser zu verstehen und so zu optimieren.

Ein aktueller Bereich in dem die Datenanalyse und Vorhersage eine große Rolle spielt ist die Instandhaltung von Brücken im Straßenverkehr. Da der Bauwerksbestand immer älter wird und auch die Verkehrsbelastung stark zugenommen hat beziehungsweise auch, nach dem Bundesministerium für Verkehr (2013), noch weiter zunehmen wird, wächst die Bedeutung der Prüfung des Zustandes von Brücken.
Anfang 2020 stürzte beispielsweise die „Ponte di Albiano Magra“ zwischen Ligurien und der Toskana in Italien ein. Dabei waren zuvor sogar Meldungen über größere Risse in der Brücke bekannt gewesen, wie Tagesschau (2020) berichtet. Solche Risse im Beton der Brücke entstehen durch die stetige Belastung des immer weiter steigenden Verkehrsaufkommens. Dieses Risswachstum kann gemessen und mit Hilfe von statistischen Verfahren analysiert und prognostiziert werden. Aus den Ergebnissen wird dann geschlossen, ob oder wann die Brücke restauriert beziehungsweise abgerissen und neu aufgebaut werden muss. Ein neues Verfahren, welches unter anderem solche Prozesse analysieren kann, wird in dieser Arbeit konstruiert. Dieses neue Verfahren und zwei weitere verwandte Konzepte, die im Rahmen dieser Arbeit thematisiert werden, gehören in den Bereich des Maschinellen Lernens zugeordnet und werden als Boosting-Verfahren bezeichnet. Ursprünglich wurden Boosting-Verfahren zur Klassifikation von Daten konstruiert, mittlerweile bearbeiten sie auch Regressionsprobleme erfolgreich. Das am weitesten verbreitete Boosting-Verfahren im Bereich der Regression ist das klassische Gradienten-Boosting und ist nach Groll (2021) eine der mächtigsten Lernideen der letzten Jahre. Daher soll es im Rahmen dieser Arbeit diskutiert und mit Hilfe einer Simulationsstudie mit weiteren Verfahren verglichen

werden. Denn wie das „No-Free-Lunch Theorem“ besagt, gibt es keinen Algorithmus, der über alle Probleme hinweg besser funktioniert als alle anderen. Wird beispielsweise ein kontaminierter Datensatz betrachtet, sind also atypische Beobachtungen im Datensatz enthalten, werden die Ergebnisse des klassischen Gradienten-Boosting schnell verfälscht. Solche Ausreißer sind keine Seltenheit. Beim Modellieren des Risswachstums in Betonträgern von Brücken beispielsweise, kann es passieren, dass durch unbekannte Einflüsse, wie technische Fehler der Messgeräte, Beobachtungen verzerrt werden. Robuste Verfahren wirken genau diesen Problemen entgegen. Ein solches robustes Boosting-Verfahren für Regression ist das RRBoost-Verfahren. Es stellt eines der drei Verfahren dar, welche im Rahmen dieser Arbeit verglichen werden. RRBoost weist eine konsistent gute Performance auf, das heißt Vorhersagen für unabhängige neue Beobachtungen liegen nah an den wahren Werten der Zielvariable. Dieses Verhalten wird sowohl auf unkontaminierten als auch auf kontaminierten Daten erreicht. Jedoch werden diese guten Ergebnisse durch eine hohe Komplexität des Verfahrens erzielt. Wünschenswert ist es solche guten Ergebnisse mit möglichst simplen Verfahren zu erreichen, da diese leichter zugänglich sind. Einem solchen Verfahren wird im Rahmen dieser Arbeit der Startschuss gegeben, dem sogenannten K-Vorzeichen-Boosting. Die Grundlage bildet die K-Vorzeichen-Tiefe, ein sehr simples Kriterium, um die Anpassung eines Modells an Daten zu messen und zu bewerten. Die Tiefe basiert auf den Vorzeichen der Abstände zwischen den Vorhersagen des jeweiligen Modells und den zugehörigen wahren Werten der Zielvariablen. Dadurch, dass lediglich die Vorzeichen der Residuen betrachtet werden, ist das Kriterium sehr robust gegenüber Ausreißern in den Daten. In der Anwendung zeigt das hier vorgestellte K-Vorzeichen-Boosting in der Vorhersagegenauigkeit noch nicht ganz die gewünschten Ergebnisse. Aber gemessen mit dem RMSE auf einem unabhängigen Testdatensatz ist die erreichte Vorhersagegenauigkeit im Gegensatz zum Gradienten-Boosting konsistent für kontaminierte sowie unkontaminierte Daten. Die Vorhersagegenauigkeit ist jedoch im Vergleich zum RRBoost-Verfahren beziehungsweise dem Gradienten-Boosting auf unkontaminierten Daten noch nicht sehr hoch. Was man jedoch schon positiv hervorheben kann ist, dass das K-Vorzeichen-Boosting zumindest auf stark kontaminierten Daten eine bessere Performance zeigt als das Gradienten-Boosting.

Die Methodik der Grundlagen dieser drei Verfahren wird im dritten Kapitel der statistischen Methoden vorgestellt. Sie folgt auf die Erläuterung der Problemstellung inklusive der Beschreibung des Datenmaterials und der Vorstellung der Ziele

der Arbeit im zweiten Kapitel. Im vierten Kapitel wird dann die Implementierung des K-Vorzeichen-Boosting thematisiert. Es wird darauf eingegangen welche Probleme sich bei der Implementierung gestellt und welche Lösungsansätze sich daraus entwickelt haben. Im fünften Kapitel wird dann ein Vergleich der Vorhersagegenauigkeit zwischen den drei vorgestellten Verfahren hergestellt. Die letzten beiden Kapitel fassen die wichtigsten Ergebnisse dieser Arbeit noch einmal zusammen und geben einen Ausblick für mögliche Verbesserungen der Implementierung des K-Vorzeichen-Boosting.

2 Problemstellung

2.1 Beschreibung des Datenmaterials

Da das K-Vorzeichen-Boosting im Rahmen dieser Arbeit erst konstruiert wird, werden Daten benötigt, um dieses zu evaluieren und erste mögliche Schwachstellen festzustellen und zu verbessern. Für diese Aufgaben, den anschließenden Vergleich der drei Boosting-Verfahren und um deren Performance zu testen, werden Simulationsstudien angesetzt. Im Folgenden wird beschrieben wie die Daten in diesen Simulationen konstruiert wurden. Das Vorgehen ist an die Simulationsstudie aus Ju und Salibián-Barrera (2021) angelehnt.

Es werden Datensätze der Form $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}, i = 1, \ldots, N$, konstruiert, mit $N \in \mathbb{N}$ Beobachtungen. Für $i = 1, \ldots, N$ sind $\mathbf{x}_i \in \mathbb{R}^p, p \in \mathbb{N}$, die Werte der erklärenden Variablen, und $y_i \in \mathbb{R}$ die Werte der Zielvariablen. Die $\mathbf{x}_1, \ldots, \mathbf{x}_N$ sind dabei unabhängige Realisierungen einer Zufallsvariable X, mit $X \sim U(0, 1)$, das heißt die Werte der Zielvariablen werden jeweils aus einer Gleichverteilung auf dem Intervall $(0, 1)$ gezogen. Die Werte der Zielvariablen folgen einem Modell

$$y_i = f(\mathbf{x}_i) + C\varepsilon_i,\ i = 1, \ldots, N, \tag{2.1}$$

mit einer Konstante $C > 0$. Für die Funktion $f : \mathbb{R}^p \to \mathbb{R}$ in Gleichung (2.1) werden fünf verschiedene Funktionen betrachtet mit entsprechend hinreichend groß gewähltem p:

$$\begin{aligned} f_1(\mathbf{x}) &= 10\mathbf{x}_1, & f_4(\mathbf{x}) &= 5 + 3\mathbf{x}_1 + 7\mathbf{x}_2 \\ f_2(\mathbf{x}) &= 3\mathbf{x}_1 + 7\mathbf{x}_2, & f_5(\mathbf{x}) &= 6 + 2\mathbf{x}_1 - 7\mathbf{x}_3 + 4\mathbf{x}_6 - \mathbf{x}_7 \\ f_3(\mathbf{x}) &= 3 + 10\mathbf{x}_1. \end{aligned}$$

Die Fehler $\varepsilon_i, i = 1, \ldots, N$ sind Realisierungen einer Zufallsvariable $\mathcal{E}$. Für die Verteilung der Zufallsvariable $\mathcal{E}$ stehen vier verschiedenen Optionen zur Auswahl:

1. $D_1 : \mathcal{E} \sim \mathcal{N}(0, 1)$ (univariate Standardnormalverteilung)
2. $D_2 : \mathcal{E} \sim \mathcal{T}_1$ (Studentsche t-Verteilung mit einem Freiheitsgrad)
3. $D_3 : \mathcal{E} \sim (1 - \alpha)\mathcal{N}(0, 1) + 0.5\alpha\mathcal{N}(\beta_1, 0.1^2) + 0.5\alpha\mathcal{N}(-\beta_1, 0.1^2)$
4. $D_4 : \mathcal{E} \sim (1 - \alpha)\mathcal{N}(0, 1) + 0.5\alpha\mathcal{N}(\beta_2, 0.1^2) + 0.5\alpha\mathcal{N}(-\beta_2, 0.1^2)$.

Die Fehlerverteilungen D_3 und D_4 sind trimodale Normalverteilungen, für die zwei Parameter gewählt werden müssen. Der Parameter $\alpha > 0$ bestimmt den Anteil der Kontamination in den Daten. Um ein angemessenes Verhältnis zwischen Ausreißern und Nicht-Ausreißern zu schaffen, wird $\alpha = 0.2$ gewählt. Die beiden trimodalen Verteilungen unterscheiden sich in der Lage der äußeren Gipfel, die durch β_1 beziehungsweise β_2 bestimmt sind. In D_3 wird $\beta_1 = 10$ gesetzt, um ein moderates Ausmaß an Ausreißern zu schaffen. In D_4 gilt $\beta_2 = 100$, um die Verfahren auch bei sehr großen Abweichungen vergleichen zu können.
Der Einstellungsparameter C kontrolliert das Signal-Rausch-Verhältnis (engl. signal-to-noise ratio, kurz SNR) gegeben durch

$$\text{SNR} = \frac{\text{Var}(f(X))}{\text{Var}(C\mathcal{E})}.$$

Ist die Variabilität des Rauschens, also der Fehler $\mathcal{E}$, im Vergleich zur Variabilität des Signals $f(X)$ zu groß, ist das Signal nicht mehr deutlich genug. Wenn jedoch andersherum die Variabilität des Rauschens im Vergleich zu gering ausfällt werden die Daten zu „perfekt“ simuliert und die Situation wird unrealistisch. Eine Simulationsstudie soll eine möglichst reale Situation erschaffen um Modelle hinsichtlich ihrer Fähigkeiten zu beurteilen und zu vergleichen. Im Einklang mit Ju und Salibián-Barrera (2021) wird SNR = 6 gesetzt.
Als Maß für die Güte eines Modells wird die Wurzel aus der mittleren quadratischen Abweichung (kurz: RMSE, englisch: root mean squared error) genutzt. Der RMSE ist definiert durch

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$$

mit $\hat{y}_i$ als Vorhersage des Modells für die Beobachtung mit wahrem Wert y_i für $i = 1, \ldots, N$. Es gilt, desto kleiner der RMSE ausfällt, desto kleiner sind die quadratischen Abstände zwischen den Vorhersagen des Modells und den jeweiligen wahren Werten der Zielvariablen. Dies spricht für eine bessere Anpassung des Modells an die vorgegebenen Daten $\mathcal{D}$.
Um die drei Boosting-Verfahren anzuwenden, zu evaluieren und mit den jeweils anderen beiden Verfahren zu vergleichen, werden bis zu drei Teildatensätze verwendet, ein Trainings-, Validierungs- und Testdatensatz. Beim RRBoost-Verfahren werden manche der Parameter im Prozess der Konstruktion validiert, beim K-Vorzeichen-Boosting und Gradienten-Boosting wird das im Rahmen dieser Arbeit nicht genutzt. Für das K-Vorzeichen-Boosting wird dies aber im Ausblick noch thematisiert. Der Trainings- beziehungsweise Validierungsdatensatz wird als ein gesamter Datensatz erzeugt und erst im Anschluss in zwei Teildatensätze aufgesplittet. Beim K-Vorzeichen-Boosting und Gradienten-Boosting wird dann keine Aufteilung vorgenommen. Der Trainingsdatensatz wird mit der gewünschten Kombination an Einstellungen, das heißt der Wahl der Zielfunktion, der Fehlerverteilung etc. konstruiert und zum Trainieren des Modells verwendet. Erst wenn das finale Modell steht, kommt der Testdatensatz zum Einsatz. Er wird zur unabhängigen Evaluation der Verfahren genutzt und daher getrennt von den anderen beiden Teildatensätzen erzeugt. Bei den Testdaten wird immer eine unkontaminierte Verteilung für die Fehler gewählt. Hier fällt die Wahl der Verteilung also immer auf die Standardnormalverteilung D_1. Da der Testdatensatz die Konstruktion des Modells nicht beeinflusst, haben die Testdaten auch keinen Einfluss auf die Vorhersagen. Bei simulierten Daten einen kontaminierten Testdatensatz zur Performancemessung zu verwenden ist in sofern nicht sinnvoll, als dass dies lediglich eine verfälschte Messung der Performance des Modells mit sich bringen würde. Denn es ist nicht die Performance des Modells bei atypischen Beobachtungen von Interesse. Diese entstehen lediglich ungewollt und gehören nicht zur Regel. Da Ausreißer in den Testdaten oft große Abweichungen zu den zugehörigen wahren Werten erzeugen, würden diese die Performancemessung, wie zum Beispiel den RMSE, unnötig negativ beeinflussen. Die Anzahl der Beobachtungen in jedem der Teildatensätze wird anteilig an N gewählt.

2.2 Zielsetzung

Der Fokus dieser Arbeit liegt auf der Konstruktion eines robusten Boosting-Verfahrens, dem K-Vorzeichen-Boosting. Bei dieser Konstruktion ist das Ziel vor allem ein Verfahren zu schaffen, dessen Vorhersagegenauigkeit möglichst präzise ist, um mit vergleichbaren Verfahren mithalten zu können. Um diesen Aspekt beurteilen zu können, wird ein Vergleich zu anderen Boosting-Verfahren hergestellt. In dem Vergleich werden neben dem K-Vorzeichen-Boosting ein robustes und nichtrobustes Boosting-Verfahren betrachtet, das nichtrobuste Gradienten-Boosting und das robuste Boosting-Verfahren für Regression, RRBoost. Als Maß für die Modellwahl im K-Vorzeichen-Boosting wird die K-Vorzeichen-Tiefe verwendet. Diese hat den Vorteil gegenüber anderen Maßzahlen, dass sie sehr simpel ist, sodass diese Eigenschaft auch für das darauf basierende Boosting-Verfahren wünschenswert ist.

3 Statistische Methoden

Im Folgenden werden die, in dieser Arbeit verwendeten, statistischen Methoden vorgestellt. Dazu gehören das klassische Boosting (Abschnitt 3.1), das RRBoost-Vefahren (Abschnitt 3.2) und die im K-Vorzeichen-Boosting verwendeten Konzepte (Abschnitt 3.3). Das K-Vorzeichen-Boosting an sich wird im Detail im vierten Kapitel thematisiert und erläutert. Das Ziel aller dieser Boosting-Verfahren ist es den Zusammenhang zwischen Einfluss- und einer Zielvariablen zu analysieren. Dabei gilt es eine Zielfunktion $F : \mathbb{R}^p \to \mathbb{R}$, die diesen Zusammenhang beschreibt, zu approximieren.
Bei der Beschreibung der Verfahren wird folgende Notation verwendet:
$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, mit $N \in \mathbb{N}$ Beobachtungen, bezeichnet den zu Grunde liegenden Datensatz. Für die p Einflussvariablen gilt $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathbb{R}^p, p \in \mathbb{N}$, und für die Zielvariable $y_i \in \mathbb{R}, i = 1, \ldots, N$. Die Notation $\mathbf{x}^{(j)} = (x_{1j}, \ldots, x_{Nj})^T \in \mathbb{R}^N$ gibt alle N Beobachtungen an, die der j-ten Einflussvariable, $j = 1, \ldots, p$, zugeordnet sind. Mit $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(p)}) \in \mathbb{R}^{N \times p}$ wird die komplette Datenmatrix über alle Beobachtungen aller Einflussvariablen notiert. Wird eine Funktion auf die Datenmatrix $\mathbf{x}$ angewendet, also beispielsweise die Notation $F(\mathbf{x})$ genutzt, impliziert dies eine Anwendung der Funktion auf die Zeilen der Matrix. Das Resultat ist ein Vektor $(F(\mathbf{x}_1), \ldots, F(\mathbf{x}_N))^T$. Das gilt analog für die Funktionen $h_{\boldsymbol{\theta}} : \mathbb{R}^p \to \mathbb{R}$, die im Laufe des Kapitels noch näher erläutert wird. Weiter suggerieren Variablen, die fett gedruckt sind einen Vektor der Dimension größer als 1, während Großbuchstaben Zufallsvariablen notieren.
Alle Implementierungen und Berechnungen dieser Arbeit wurden mit Hilfe der Software R entwickelt (R Core Team (2020)).

3.1 Klassisches Boosting

Das klassische *Boosting* ist eine der mächtigsten Lernideen der letzten Jahre, wie Groll (2021) anmerken. Das Verfahren wurde ursprünglich für das Klassifizieren von Daten konzipiert, jedoch kann es auch für die Vorhersage einer stetigen Zielvariable erweitert werden. Dieser Fall wird im Folgenden thematisiert.
Die grundsätzliche Idee des Boosting ist es „simple“ Modelle, auch *Basislerner* genannt, so zu kombinieren, dass daraus ein starkes Modell resultiert. Wie in Abbildung 3.1 zu sehen, ist das Verfahren sequentiell aufgebaut, das heißt jede neue Iteration basiert auf allen vorangegangenen Schritten.

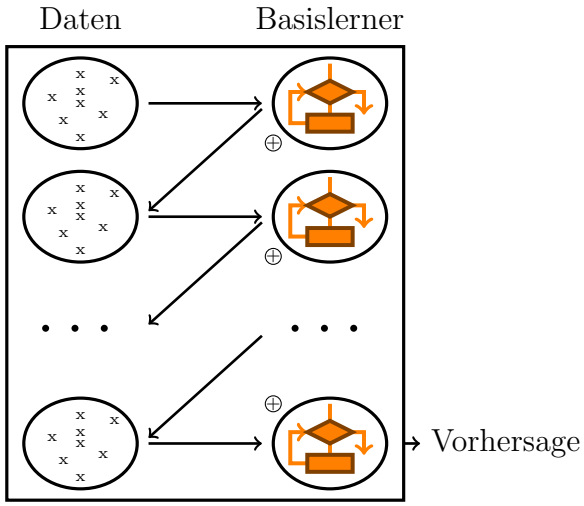

Abbildung 3.1: Schematische Darstellung des Boosting-Verfahrens (Groll (2021)).

In der linken Spalte der Grafik sind für jede Iteration die zur Anwendung kommenden Daten abgebildet. Eine Iteration oder auch Boosting-Runde des Verfahrens ist in dem Schaubild als eine Zeile zu verstehen. In jeder Runde werden die Informationen des kompletten Datensatzes verwendet, sodass alle Punktwolken jeder Zeile gleich aussehen. Es folgt die Anpassung eines Basislerners auf die Daten. Dabei wird jedoch nicht der originale Datensatz verwendet, sondern eine leicht modifizierte Version. Am Anfang des Verfahrens steht eine „naive“ Vorhersage für alle Daten. Danach wird die Diskrepanz zwischen dem gemessenen Wert der Zielvariablen und dieser Vorhersage gemessen. Diese Diskrepanz sind die Residuen oder auch die Fehler des Modells. Die Basislerner werden also auf die Residuen angepasst, um diese möglichst zu reduzieren. Diese Anpassung kann jeweils als Korrektur für die Anfangsvorhersage gesehen werden. Die Vorhersagen der Fehler in jedem neuen Schritt werden dann additiv mit

den vorangegangenen Iterationen des Verfahrens verknüpft. Wie genau diese Initialisierung zustande kommt und wie die Residuen definiert werden wird im Folgenden noch näher erläutert.
Es gibt viele verschiedene Ansätze diese grundlegende Idee des Boostings umzusetzen. Einer dieser Ansätze ist das *Gradienten-Boosting*, welches unter anderem im R-Paket *mboost* (Hothorn u. a. 2021) implementiert ist. Bei diesem Ansatz steht eine *Verlustfunktion* $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ im Vordergrund. Sie misst den Verlust, der bei einem gemessenen Wert y der Zielvariablen und einer zugehörigen Vorhersage $\hat{F}(\mathbf{x})$ erreicht wird.
Friedman (2001) beschreiben die grundlegende Idee des Gradienten-Boosting durch das Finden einer Zielfunktion $\hat{F}$, die den erwarteten Wert der Verlustfunktion über die gemeinsame Verteilung $\mathbb{P}_{YX}$ aller $(Y, \mathbf{X})$-Werte minimiert. Es gilt

$$\hat{F} = \underset{F^*}{\operatorname{argmin}}\, E_{Y,\mathbf{X}} L(Y, F^*(\mathbf{X})). \tag{3.1}$$

Da $\mathbb{P}_{YX}$ unbekannt ist, kann Gleichung (3.1) nicht direkt benutzt werden um optimale Vorhersagen zu schaffen. Daher wird das obige Problem approximiert durch die Minimierung des *empirischen Risikos* $\mathcal{R}_{\text{emp}}(F^*)$, sodass das folgende Optimierungsproblem im Fokus des Gradienten-Boosting steht

$$\hat{F} = \underset{F^*}{\operatorname{argmin}}\, \mathcal{R}_{\text{emp}}(F^*) = \underset{F^*}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, F^*(\mathbf{x}_i)). \tag{3.2}$$

Dabei wird die Struktur der Zielfunktion F als additiv angenommen und durch

$$\hat{F}(\mathbf{x}) = \sum_{m=1}^{M} \beta_m h_{\boldsymbol{\theta}_m}(\mathbf{x})$$

mit $\beta_m \in \mathbb{R}$ und $M \in \mathbb{N}$ als Anzahl der Boosting-Runden, dargestellt. Die Funktionen $h_{\boldsymbol{\theta}_m} : \mathbb{R}^p \to \mathbb{R}$ sind sogenannte Basislerner mit Parameter $\boldsymbol{\theta}_m$ und werden alle aus der gleichen Klasse von Funktionen $\mathcal{H} = \{h_{\boldsymbol{\theta}} \,|\, \boldsymbol{\theta} \in \Theta\}$ gewählt. Diese Wahl wird im weiteren Verlauf noch näher thematisiert.

Zunächst soll die Frage geklärt werden, wie das Optimierungsproblem in 3.2 gelöst werden kann? Beim Gradienten-Boosting wird eine differenzierbare, meist konvexe

Verlustfunktion verwendet, sodass der Gradient $g : \mathbb{R} \to \mathbb{R}$ mit

$$g(z) = \frac{\partial L(y, z)}{\partial z}$$

in jeder Iteration $m = 1, \ldots, M$ leicht berechenbar ist.

Beispiele von Verlustfunktionen

Für die Wahl der Verlustfunktion für das Boosting im Bereich der Regressionsprobleme gibt es je nach Anwendung verschiedene Optionen. Am weitesten verbreitet ist die *L2-Verlustfunktion*. Sie ist definiert durch

$$L_2\left(y, z\right) = \left(y - z\right)^2,$$

mit $z, y \in \mathbb{R}$. In der Praxis wird die Funktion dazu genutzt, um den Verlust zwischen den wahren Werten der Zielvariablen $y = y_i$ und den Vorhersagen des Modells $z = \hat{F}(\mathbf{x}_i)$, für alle Beobachtungen $i = 1, \ldots, N$, zu messen. Dank ihrer Form bringt die L2-Verlustfunktion günstige Eigenschaften mit. Durch die Konvexität und Differenzierbarkeit wird der Gradient leicht berechenbar. Er ist gegeben durch

$$g(z) = \frac{\partial L_2(y, z)}{\partial z} = \frac{\partial (y - z)^2}{\partial z} = 2(y - z).$$

Dadurch, dass der Verlust der bei einer Vorhersage $z = \hat{F}(\mathbf{x}_i)$ gegenüber einem wahren Wert y_i entsteht quadratisch gemessen wird, sind vor allem Ausreißer problematisch, da großen Abständen viel Gewicht gegeben wird. Dadurch können solche atypischen Beobachtungen leicht zur einer Verfälschung der Ergebnisse führen.
Die *L1-Verlustfunktion* ist etwas robuster gegenüber Ausreißern bei der Zielvariablen als die L2-Verlustfunktion, dafür ist sie aber nicht differenzierbar in 0. Dadurch wird die Optimierung des empirischen Risikos komplizierter. Die L1-Verlustfunktion ist gegeben durch

$$L_1\left(y, z\right) = |y - z|.$$

Große Werte zwischen einem wahren Wert der Zielvariable und einer Vorhersage werden also nicht so stark gewichtet wie bei der L2-Verlustfunktion, das Quadrieren fällt weg.
Eine Kombination der L1- und L2-Verlustfunktion ist die *Huber-Verlustfunktion*.

Sie ist stückweise definiert, behält jedoch an den Übergangspunkten ihre Differenzierbarkeit. Um den Nullpunkt herum, verhält sie sich wie die L2-Verlustfunktion, ab einem Punkt $c > 0$ beziehungsweise $-c < 0$ jedoch wird sie linear weitergeführt. Damit kombiniert sie die Eigenschaft der Konvexität und Differenzierbarkeit der L2-Verlustfunktion mit der Robustheit der L1-Verlustfunktion. Die Huber-Verlustfunktion ist gegeben durch

$$L_{\text{huber},c}(y,z) = \begin{cases} \frac{1}{2}(y-z)^2 & \text{, wenn } |y-z| \leq c \\ c \cdot (|y-z| - \frac{1}{2}c) & \text{, wenn } |y-z| > c. \end{cases}.$$

Das Optimieren des empirischen Risikos ist jedoch problematischer als bei der L2-Verlustfunktion. Denn wird in Gleichung (3.2) F^* als konstant vorausgesetzt, das heißt die Basislerner sind konstante Funktionen, dann gibt es bezüglich der Huber-Verlustfunktion schon keine Lösung dieser Gleichung mit geschlossener Form. Das Analogon bei der L2-Verlustfunktion, also diejenige Konstante, die das empirische Risiko minimiert, ist der Mittelwert.
Eine weitere Wahl für die Verlustfunktion ist die *Tukey-Verlustfunktion*. Sie wird vor allem noch im nächsten Kapitel über das RRBoost-Verfahren interessant werden. Die Tukey-Verlustfunktion ist eine noch robustere Variante der L2-Verlustfunktion und stückweise definiert durch

$$L_{\text{tukey},c}(y,z) = \begin{cases} 1 - (1 - (\frac{y-z}{c})^2)^3 & \text{, wenn } |y-z| \leq c \\ 1 & \text{, wenn } |y-z| > c. \end{cases}.$$

mit Konstante $c \geq 0$. Ein Verlustgraph mit allen vier vorgestellten Verlustfunktionen im Vergleich ist in Abbildung 3.2 gegeben.

Beim Gradienten-Boosting gibt der negative Gradient der $m-ten$ Boosting-Runde $-g_m(z_i)$ der Verlustfunktion jeweils die Abstiegsrichtung der Funktion am Punkt $z_i = \hat{F}(\mathbf{x}_i)$ an, also die Richtung in die der Verlust reduziert wird. Der nächste Basislerner wird dann so gewählt, dass er die Abstiegsrichtung $-g_m(z_i)$ möglichst gut approximiert, der quadratische Abstand also minimal ist. Dieser Schritt ist in Algorithmus 1 in Zeile 10 dargestellt.

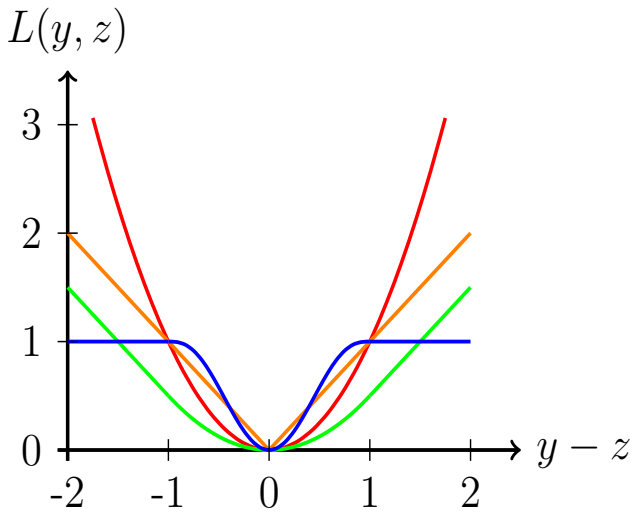

Abbildung 3.2: Graphische Darstellung der L2-Verlustfunktion L_2, L1-Verlustfunktion L_1, Huber-Verlustfunktion $L_{huber,1}$ und der Tukey-Verlustfunktion $L_{tukey,1}$

Algorithmus 1 Gradienten-Boosting

Eingabe:

Daten $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$

Verlustfunktion L

Anzahl Iterationen M

Klasse der Basislerner $\mathcal{H}$ mit $h_{\boldsymbol{\theta}} \in \mathcal{H}$

Initialisierung:

$\hat{F}_0(\mathbf{x}) = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^N L(y_i, \theta)$ (optimale Konstante)

for $m = 1 \to M$ **do**:

  for $i = 1 \to N$ **do**:

    $r_i = -\left[\dfrac{\partial L(y_i, z_i)}{\partial z_i}\right]_{z_i = \hat{F}_{m-1}(\mathbf{x}_i)}$

  end for

  Anpassen eines Basislerners für Regression auf die Pseudo-Residuen r_i:

  $\hat{\boldsymbol{\theta}}_m = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^N (r_i - h_{\hat{\boldsymbol{\theta}}}(\mathbf{x}_i))^2$

  Liniensuche: $\hat{\beta}_m = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^N L\left(y_i, \hat{F}_{m-1}(\mathbf{x}_i) + \beta\, h_{\hat{\boldsymbol{\theta}}_m}(\mathbf{x}_i)\right)$

  Update: $\hat{F}_m(\mathbf{x}) = \hat{F}_{m-1}(\mathbf{x}) + \hat{\beta}_m\, h_{\hat{\boldsymbol{\theta}}_m}(\mathbf{x})$

end for

Ausgabe: $\hat{F}(\mathbf{x}) = \hat{F}_M(\mathbf{x})$

Als nächstes wird die Schrittweite $\hat{\beta}_m$ bestimmt, mit welcher in die Richtung des Abstiegs „gelaufen“ werden soll. Dazu wird der Parameter $\hat{\beta}_m$ gesucht, der

$$\sum_{i=1}^{N} L\left(y_i, \hat{F}_{m-1}\left(\mathbf{x}_i\right) + \beta\, h_{\hat{\boldsymbol{\theta}}_m}\left(\mathbf{x}_i\right)\right)$$

minimiert, also den Verlust der neuen Vorhersage am meisten reduziert. Anschließend wird die Vorhersage des $(m-1)$-ten Schrittes additiv durch das Produkt der Schrittweite mit der Approximation des negativen Gradienten erweitert. Die Aktualisierung der Vorhersage ist also gegeben durch

$$\hat{F}_m\left(\mathbf{x}\right) = \hat{F}_{m-1}\left(\mathbf{x}\right) + \hat{\beta}_m\, h_{\hat{\boldsymbol{\theta}}_m}\left(\mathbf{x}\right).$$

Wird keine Abbruchregel eingeführt, erfolgt eine M-malige Durchführung dieses Vorgehens und die M-te Vorhersage ist die finale Vorhersage des gesamten Modells. Die Basislerner sind alle aus der gleichen Klasse von Funktionen $\mathcal{H}$. Wie können $\mathcal{H}$ und die zugehörigen Parameter $\boldsymbol{\theta}$ aussehen?

Beispiele für Basislerner

Wie oben schon beschrieben hängen die Basislerner $h_{\boldsymbol{\theta}}\left(\mathbf{x}\right)$ von 2 Funktionsargumenten ab, den Einflussvariablen $\mathbf{x}$ und dem Parametervektor $\boldsymbol{\theta}$, der die Funktion auch charakterisiert. Die einzelnen Basislerner in den verschiedenen Iterationen unterscheiden sich dabei nur durch diesen Parametervektor und sind darüber hinaus von gleicher Struktur.

Die simpelste Wahl für die Klasse an Funktionen $\mathcal{H}$, aus denen die Basislerner gewählt werden, ist $\mathcal{H} = \{\mathbf{1}, x^{(1)}, \ldots, x^{(p)} | i = 1, \ldots, N\}$. Die Bezeichnung $\mathbf{1}$ notiert die konstante Einsfunktion und die $x^{(j)}$ die Koordinatenprojektionen für die einzelnen Variablen $j = 1, \ldots, p$. Der Basislerner $h_\theta \in \mathcal{H}$ ist durch die Projektion auf die θ-te Variable gegeben. Hier gilt also $\theta \in \{0, \ldots, p\}$. Die 0-te Variable beschreibt den Intercept des Modells, es gilt $h_0(\mathbf{x}_i) = 1$. Für $\theta \in \{1, \ldots, p\}$ können die Basislerner $h_\theta : \mathbb{R}^p \to \mathbb{R}$ beschrieben werden durch

$$h_\theta(\mathbf{x}_i) = x_{i\theta}, \quad i = 1, \ldots, N.$$

Die am weitesten verbreitete Wahl für die Basislerner im Kontext des klassischen Gradienten-Boosting ist die der *Entscheidungsbäume*, beziehungsweise für Regressionsprobleme, der Regressionsbäume. Die Grundidee ist es, die Wertebereiche der

Einflussvariablen in $S \in \mathbb{N}$ Teilregionen aufzuteilen und für jede dieser Teilregionen eine konstante Vorhersage zu kalkulieren. Regressionsbäume sind daher sehr simpel aufgebaut und auch leicht interpretierbar, was sie in der Anwendung sehr attraktiv macht. Ein solcher Baum $h_{\boldsymbol{\theta}}$ kann durch

$$h_{\boldsymbol{\theta}}(\mathbf{x}_i) = \sum_{s=1}^{S} \gamma_s \mathbb{1}(\mathbf{x}_i \in R_s), \quad i = 1, \ldots, N,$$

beschrieben werden. Die Parameter γ_s sind die Vorhersagen für die jeweiligen Teilbereiche $R_s \subset \mathbb{R}^p, s = 1, \ldots, S$. Wird mit $w \in \mathbb{N}$ die Tiefe des Baumes notiert, gilt $2^w = S$. In Abbildung 3.3 ist ein solcher Regressionsbaum für ein Minimalbeispiel dargestellt.

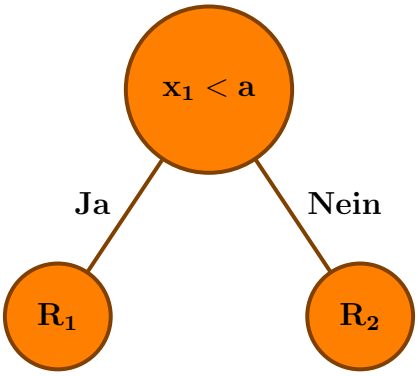

Abbildung 3.3: Darstellung eines Regressionsbaumes. Aufgrund seiner Tiefe kann der Baum auch als Regressionsstumpf bezeichnet werden. Sei $(x_1, y_1) \in \mathcal{D}$, mit $p = 1$, dann sind die Entscheidungsregeln gegeben durch $x_1 \in R_1 \Leftrightarrow x_1 < a$ und $x_1, y_1 \in R_2 \Leftrightarrow x_1 \geq a$, mit $a \in \mathbb{R}$.

Die Zuordnung der Datenpunkte in die Teilregionen passiert basierend auf den jeweiligen Ausprägungen der Einflussvariablen. In Abbildung 3.3 ist ein Regressionsstumpf zu sehen, es gilt daher $w = 1$. Das heißt, es gibt nur einen Spaltungspunkt und daher wird auch nur eine Einflussvariable zum Separieren der Werte verwendet. Es ist jedoch möglich Regressionsbäume beliebig zu erweitern. Welche Variable mit welchem Spaltungswert in welcher Tiefe des Baumes sitzt, wird mit Hilfe von Risikominimierung einer gewählten Verlustfunktion entschieden. Welche Splitpunkte und -variablen gewählt werden, charakterisiert den jeweiligen Entscheidungsbaum und ist in dem Parameter $\boldsymbol{\theta}$ der Basisfunktion $h_{\boldsymbol{\theta}}$ festgehalten. Alle Datenpunkte $\{\mathbf{x}_i | \mathbf{x}_i \in R_s, i = 1, \ldots, N\}$, die einer Teilregion R_s zugeordnet werden, erhalten dann die gleiche Vorhersage γ_s. Diese wird ebenfalls durch Risikominimierung mit Hilfe ei-

ner gewählten Verlustfunktion ermittelt. Mehr Details können in Hastie, Tibshirani und Friedman (2009) nachgelesen werden.

Doch um den ersten Basislerner konzipieren zu können braucht es eine erste Vorhersage. Dazu wird die Konstante, die das empirische Risiko minimiert, verwendet. Diese Konstante ist gegeben durch den Wert, der die Verlustfunktion über alle Beobachtungen minimiert, das heißt

$$\hat{F}_0(\mathbf{x}) = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, \theta).$$

Es spielt also eine zentrale Rolle, welche Verlustfunktion Anwendung findet. Wie wird diese also gewählt? Die Wahl der Verlustfunktion und auch der Basislerner liegt beim Anwender und richtet sich meist nach dem Kontext des Gebrauchs beziehungsweise der gegebenen Problemstellung. Beim klassischen Gradienten-Boosting wird häufig die schon erwähnte L2-Verlustfunktion benutzt. Diese Funktion findet viel Zuspruch, da sie sehr benutzerfreundlich ist. Durch die Konvexität und Differenzierbarkeit der Funktion, ist der Gradient leicht berechenbar, was den Optimierungsschritt in jeder Boosting-Iteration deutlich erleichtert. Auch die Wahl des Initialwerts spielt keine so große Rolle. Eine weitere Eigenschaft der L2-Verlustfunktion ist die hohe Verlustanzeige bei großen Residuen. Durch das quadratische Verhalten wird die Diskrepanz zwischen wahrem Wert und Vorhersage mit der Potenz 2 sehr stark gewichtet. Diese Eigenschaft kann beim Vorhandensein von Ausreißern zu einer drastischen Verfälschung der Vorhersagen führen. In einer solchen Lage bieten robuste Verfahren für Regression einen Ausweg. Mehr Robustheit kann beispielsweise durch eine robustere Wahl der Verlustfunktion erreicht werden, wie in Friedman (2001) vorgestellt. Ju und Salibián-Barrera (2021) sehen dies jedoch kritisch hinsichtlich der Eigenschaften der unterschiedlichen Verlustfunktionen. Wird beispielsweise die L1-Verlustfunktion verwendet, schwindet die Performance des Modells. Die Anwendung von Funktionen wie der Huber- oder Tukey-Verlustfunktion macht die Spezifizierung eines Skalierungsparameters für die Residuen nötig. Dafür wird in jeder Boosting-Runde ein robuster Skalierungsschätzer, basierend auf den Residuen der Anpassung des Modells in der Vorrunde, erzeugt. Problematisch ist dabei, dass der Schätzer schon in den ersten Runden zu groß ausfallen kann. Der Fokus wird dann zu stark auf die Ausreißer gelegt, was wiederum in einer Verfälschung der Ergebnisse resultiert. Um das zu vermeiden, wurden weitere robuste Ansätze entwickelt, welche in

dieser Arbeit thematisiert werden. Dazu gehören das K-Vorzeichen-Boosting und das RRBoost-Verfahren. Letzteres soll im folgenden Abschnitt erläutert werden.

3.2 RRBoost

Alle Informationen des folgenden Kapitels wurden aus Ju und Salibián-Barrera (2021) entnommen. Das *RRBoost*-Verfahren ist im R-Paket *RRBoost* (Ju und Salibian-Barrera 2020) implementiert.
RRBoost adressiert die Probleme des klassischen Boosting hinsichtlich möglicher Ausreißer in den Daten. Solche atypischen Beobachtungen entstehen in der Praxis häufig durch Messfehler, können aber auch durch Beobachtungseinheiten entstehen, die zur Population hinzu genommen wurden, jedoch nicht dazu gehören. Meist fallen solche atypischen Beobachtungen dadurch auf, dass sie klar von den anderen Datenwerten separierbar sind. Wie stark Ausreißer Ergebnisse und Vorhersagen beeinflussen können, wird schon bei der Lageschätzfunktion des arithmetischen Mittels klar. Das arithmetische Mittel kann zum Beispiel als Initialwert des im vorherigen Kapitel erläuterten Gradienten-Boosting mit L2-Verlustfunktion verwendet werden. Dort reicht schon nur eine Beobachtung aus, um die Schätzung stark zu verfälschen. Um durch diese atypischen Datenpunkte bei der Anpassung des Modells nicht fehlgeleitet zu werden, haben Ju und Salibián-Barrera (2021) für die Aufgabe der Regression robuste, nicht-parametrische Regressionsschätzer entwickelt, die auf dem Gradienten-Boosting-Verfahren basieren. Im Gegensatz zu den am Ende des letzten Abschnittes schon thematisierten robusten Verfahren für Regression, wie in Friedman (2001), muss bei dem RRBoost-Verfahren kein zusätzlicher Skalierungsschätzer für die Residuen kalkuliert werden. Sondern es wird direkt eine robuste Skalierung der Residuen minimiert. Genauer wird der Fokus auf die Minimierung eines M-Schätzers für die Skalierung der Residuen gelegt.

Das RRBoost-Verfahren kann in zwei aufeinander aufbauende Schritte unterteilt werden:

Schritt 1: SBoost
Der erste Schritt wird als *SBoost* bezeichnet. Das Vorgehen des SBoost-Verfahrens ist dem des Gradienten-Boosting sehr ähnlich. Beim Gradienten-Boosting gilt es Gleichung (3.2) zu lösen. Im SBoost-Schritt wird das empirische Risiko durch einen

M-Skalierungsschätzer $\hat{\sigma}(F)$ der Residuen $r_i = y_i - F(\mathbf{x}_i), i = 1, \ldots, N$ ersetzt. Das heißt es gilt

$$\hat{F} = \underset{F^*}{\operatorname{argmin}}\, \hat{\sigma}(F^*(\mathbf{x})). \tag{3.3}$$

mit $\hat{\sigma} : \mathbb{R}^N \to \mathbb{R}$ zu lösen. Der Schätzer $\hat{\sigma}(F(\mathbf{x})) = \hat{\sigma}(F(\mathbf{x}_1), \ldots, F(\mathbf{x}_N))$ ist gegeben durch die Lösung von

$$\frac{1}{N} \sum_{i=1}^{N} L_0 \left(\frac{y_i - F(\mathbf{x}_i)}{\hat{\sigma}(F(\mathbf{x}))} \right) = \kappa, \tag{3.4}$$

mit Konstante $\kappa > 0$ und einer um 0 symmetrischen Funktion $L_0 : \mathbb{R} \to \mathbb{R}_{\geq 0}$. Nach Ju und Salibián-Barrera (2021) kontrollieren κ und L_0 die Robustheit und Effizienz des resultierenden Schätzers. Für L_0 wird die Tukey-Verlustfunktion verwendet, die nach oben und unten beschränkt ist. Damit führt $\kappa = \frac{1}{2}$ zu einem Skalierungsschätzer mit maximalem Bruchpunkt und wird daher als Einstellung für die Implementierung festgelegt. Um Gleichung (3.3) zu lösen, wird in jeder Iteration $m = 1, \ldots, M_1$ der Gradient

$$g_m(\mathbf{z}) := \left[\frac{\partial \hat{\sigma}(\mathbf{z})}{\partial \mathbf{z}} \right] = \begin{bmatrix} \frac{\partial \hat{\sigma}(\mathbf{z})}{\partial z_1} \\ \vdots \\ \frac{\partial \hat{\sigma}(\mathbf{z})}{\partial z_N} \end{bmatrix} =: \begin{bmatrix} g_{m,1}(\mathbf{z}) \\ \vdots \\ g_{m,N}(\mathbf{z}) \end{bmatrix}$$

mit $\mathbf{z} = (z_1, \ldots, z_N)^T$ berechnet. Dazu werden beide Seiten der Gleichung (3.4) nach $z_j, j = 1, \ldots, N$, abgeleitet. Mit l_0 als Ableitung von L_0 ergibt sich

$$\begin{aligned}
\frac{\partial \kappa}{\partial z_j} &= \frac{\partial}{\partial z_j} \left(\frac{1}{N} \sum_{i=1}^{N} L_0 \left(\frac{y_i - z_i}{\hat{\sigma}(\mathbf{z})} \right) \right) \\
\Leftrightarrow\ 0 &= \sum_{i=1}^{N} \frac{\partial}{\partial z_j} \left(L_0 \left(\frac{y_i - z_i}{\hat{\sigma}(\mathbf{z})} \right) \right) \\
\Leftrightarrow\ 0 &= \sum_{i \neq j} l_0 \left(\frac{y_i - z_i}{\hat{\sigma}(\mathbf{z})} \right) \cdot \frac{\partial \left(\frac{y_i - z_i}{\hat{\sigma}(\mathbf{z})} \right)}{\partial z_j} \\
&\quad + l_0 \left(\frac{y_j - z_j}{\hat{\sigma}(\mathbf{z})} \right) \cdot \frac{\partial \left(\frac{y_j - z_j}{\hat{\sigma}(\mathbf{z})} \right)}{\partial z_j}
\end{aligned}$$

$$
\begin{aligned}
\Leftrightarrow\ 0 &= \sum_{i \neq j} l_0\left(\frac{y_i - z_i}{\hat{\sigma}(\mathbf{z})}\right) \cdot \left(\frac{-(y_i - z_i) g_{m,j}(\mathbf{z})}{\hat{\sigma}(\mathbf{z})^2}\right) \\
&\quad + l_0\left(\frac{y_j - z_j}{\hat{\sigma}(\mathbf{z})}\right) \cdot \left(\frac{-\hat{\sigma}(\mathbf{z}) - (y_j - z_j) g_{m,j}(\mathbf{z})}{\hat{\sigma}(\mathbf{z})^2}\right) \\
\Leftrightarrow\ 0 &= \sum_{i=1}^{N} l_0\left(\frac{y_i - z_i}{\hat{\sigma}(\mathbf{z})}\right) \cdot \left(\frac{(y_i - z_i) g_{m,j}(\mathbf{z})}{\hat{\sigma}(\mathbf{z})^2}\right) \\
&\quad + l_0\left(\frac{y_j - z_j}{\hat{\sigma}(\mathbf{z})}\right) \cdot \left(\frac{1}{\hat{\sigma}(\mathbf{z})}\right).
\end{aligned}
$$

Mit Auflösen der Gleichung nach $g_{m,j}$ folgt

$$
\begin{aligned}
g_{m,j}(\mathbf{z}) &= \frac{-l_0\left(\frac{y_j - z_j}{\hat{\sigma}(\mathbf{z})}\right) \frac{1}{\hat{\sigma}(\mathbf{z})}}{\sum_{i=1}^{N} l_0\left(\frac{y_i - z_i}{\hat{\sigma}(\mathbf{z})}\right) \cdot \left(\frac{y_i - z_i}{\hat{\sigma}^2(\mathbf{z})}\right)} \\
\Leftrightarrow\ g_{m,j}(\mathbf{z}) &= \frac{-l_0\left(\frac{y_j - z_j}{\hat{\sigma}(\mathbf{z})}\right)}{\sum_{i=1}^{N} l_0\left(\frac{y_i - z_i}{\hat{\sigma}(\mathbf{z})}\right) \cdot \left(\frac{y_i - z_i}{\hat{\sigma}(\mathbf{z})}\right)}.
\end{aligned}
$$

Wird C_m definiert durch die Inverse des Nenners ausgewertet an der Stelle $z = \hat{F}_{m-1}(\mathbf{x}) = (\hat{F}_{m-1}(\mathbf{x}_1), \ldots, \hat{F}_{m-1}(\mathbf{x}_N))^T$, also

$$
C_m := \left[\sum_{i=1}^{N} l_0\left(\frac{y_i - \hat{F}_{m-1}(\mathbf{x}_i)}{\hat{\sigma}(\hat{F}_{m-1}(\mathbf{x}))}\right) \cdot \left(\frac{y_i - \hat{F}_{m-1}(\mathbf{x}_i)}{\hat{\sigma}(\hat{F}_{m-1}(\mathbf{x}))}\right)\right]^{-1},
$$

dann resultiert daraus Zeile 8 in Algorithmus 2. Das heißt, der Gradient wird jeweils an der Vorhersage der vorangegangenen Iteration ausgewertet. Anschließend folgt eine Approximation des negativen Gradienten $-g_m$ durch einen Basislerner $h_{\boldsymbol{\theta}_m} : \mathbb{R}^p \to \mathbb{R}$, indem der quadratische Abstand minimiert wird. Dies ist auch in Zeile 12 in Algorithmus 2 aufgezeigt. Die Form der Zielfunktion F wird auch hier als additiv in Form von

$$
\hat{F}(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m h_{\boldsymbol{\theta}_m}(\mathbf{x}), \tag{3.5}
$$

angenommen. Die Parameter $\boldsymbol{\theta}_m$ der Basislerner $h_{\boldsymbol{\theta}_m}$ werden durch deren Wahl $h_{\boldsymbol{\theta}_m} \in \mathcal{H}, m = 1, \ldots, M_1$, festgelegt, das heißt je nach Wahl der Klasse von Funktionen $\mathcal{H}$, werden die Parameter bestimmt. Die Vorfaktoren $\alpha_m \in \mathbb{R}, m = 1, \ldots, M_1$ werden

Algorithmus 2 SBoost

Eingabe:

1: Daten $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$

2: Anzahl Interationen M_1

3: Klasse der Basislerner $\mathcal{H}$ mit $h_{\boldsymbol{\theta}} \in \mathcal{H}$

4: Verlustfunktion L_0

5: **Initialisierung:**

$\hat{F}_0(\mathbf{x})$ LAD-Baum

6: **for** $m = 1 \to M_1$ **do**:

7: $\hat{\sigma}(\hat{F}_{m-1}(\mathbf{x})) \in \{\sigma : \frac{1}{N}\sum_{i=1}^N L_0\left(\frac{y_i - \hat{F}_{m-1}(\mathbf{x}_i)}{\sigma}\right) = \kappa\}$

8: $C_m = \left[\sum_{i=1}^N l_0\left(\frac{y_i - \hat{F}_{m-1}(\mathbf{x}_i)}{\hat{\sigma}(\hat{F}_{m-1}(\mathbf{x}))}\right)\left(\frac{y_i - \hat{F}_{m-1}(\mathbf{x}_i)}{\hat{\sigma}(\hat{F}_{m-1}(\mathbf{x}))}\right)\right]^{-1}$

9: **for** $j = 1 \to N$ **do**:

10: $g_{m,j} = -C_m \cdot l_0\left(\frac{y_i - \hat{F}_{m-1}(\mathbf{x}_j)}{\hat{\sigma}(\hat{F}_{m-1}(\mathbf{x}))}\right)$

11: **end for**

12: $\hat{\boldsymbol{\theta}}_m = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^N (g_{m,i} + h_{\boldsymbol{\theta}}(\mathbf{x}_i))^2$

13: $\alpha_m = \underset{\alpha}{\operatorname{argmin}}\, \hat{\sigma}(\hat{F}_{m-1}(\mathbf{x}) + \alpha \cdot h_{\hat{\boldsymbol{\theta}}_m}(\mathbf{x}))$

14: $\hat{F}_m(\mathbf{x}) = \hat{F}_{m-1}(\mathbf{x}) + \alpha_m \cdot h_{\hat{\boldsymbol{\theta}}_m}(\mathbf{x})$

15: **end for**

16: **Ausgabe:** $\hat{F}_{M_1}(\mathbf{x})$, $\hat{\sigma}(\hat{F}_{M_1}(\mathbf{x}))$

durch

$$\alpha_m = \underset{\alpha}{\operatorname{argmin}}\, \hat{\sigma}(\hat{F}_{m-1}(\mathbf{x}) + \alpha \cdot h_{\hat{\boldsymbol{\theta}}_m}(\mathbf{x}))$$

kalkuliert, wie auch in Zeile 13 in Algorithmus 2 aufgeführt. Anschließend wird die alte Vorhersage aktualisiert.

Doch wie wird im SBoost-Schritt der Initialwert bestimmt? Dadurch, dass die Verlustfunktion in Algorithmus 2 nicht konvex ist, ist eine robuste Wahl für den Anfangswert von großer Bedeutung um eine gute Vorhersagegüte des Modells zu erreichen. Wie Ju und Salibián-Barrera (2021) anmerken, ist dazu die optimale Konstante bezüglich einer Verlustfunktion nicht die beste Wahl. Stattdessen wird zur Initialisierung des SBoost-Schrittes ein LAD-Baum (LAD, englisch: least absolute deviation, deutsch: kleinste absolute Abweichung) verwendet. Dies ist ein Regressionsbaum, wie er schon im vorherigen Abschnitt beschrieben wurde. Zur Konstruktion des Baumes werden absolute Abstände verwendet. Die Aufteilung passiert über die Minimierung des mittleren absoluten Wertes der Residuen. Vor allem aufgrund der erwarteten Robustheit gegenüber Ausreißern und der Handhabbarkeit von Daten mit verschie-

denen Arten von Ausprägungen sehen Ju und Salibián-Barrera (2021) diese Wahl als passende Alternative für die Initialisierung des SBoost-Verfahrens an.
Dieser SBoost-Schritt kann auch als eigenständiges Verfahren für die Aufgabe der Regression verwendet werden. Beinhalten die Daten Ausreißer zeigt sich die Robustheit des SBoost-Verfahrens, da die Prädiktion durch atypischen Beobachtungen nicht verfälscht wird. Sind jedoch keine abweichenden Datenpunkte im Datensatz enthalten ist die Effizienz des Verfahrens nicht hoch, wie numerische Experimente von Ju und Salibián-Barrera (2021) zeigen. Aus diesem Grund wurde das Verfahren mit einem weiteren Schritt verknüpft, welcher das RRBoost-Verfahren auch bei nicht kontaminierten Daten effizienter macht. Die Vorhersagegüte des klassischen Gradienten-Boosting mit L2-Verlustfunktion wird übertroffen oder zumindest nicht unterschritten.

Schritt 2: RRoost

Der zweite Schritt von RRBoost steigert die Effizienz des Verfahrens. Ausgehend vom bereits berechneten M-Schätzer $\hat{\sigma}(\hat{F}_{M_1}(\mathbf{x}))$ und der finalen Vorhersage $\hat{F}_{M_1}(\mathbf{x})$ aus dem SBoost-Schritt wird im zweiten Schritt des RRBoost-Verfahrens analog zum Gradienten-Boosting eine Verlustfunktion minimiert. Die Zielfunktion F wird durch die Lösung des folgenden Optimierungsproblems approximiert

$$\hat{F} = \underset{F^*}{\operatorname{argmin}} \sum_{i=1}^{N} L_1 \left(\frac{y_i - F^*(\mathbf{x}_i)}{\hat{\sigma}(\hat{F}_{M_1}(\mathbf{x}))} \right), \tag{3.6}$$

wobei $L_1 : \mathbb{R} \to \mathbb{R}_{\geq 0}$ eine um 0 symmetrische Funktion ist und M_1 die Anzahl an Boosting-Runden des SBoost-Schrittes angibt. Diese Verlustfunktion wird so gewählt, dass sie der L2-Verlustfunktion „ähnlicher“ ist, als die, die für den M-Skalierungsschätzer im SBoost-Schritt verwendet wurde. So kann Beobachtungen mehr Gewicht gegeben werden, die eventuell wichtig für die Schätzung und Vorhersage der Daten sind, jedoch durch die Wahl der Verlustfunktion nicht so stark in die Schätzung beziehungsweise Anpassung des Modells mit einbezogen wurden. Ein Beispiel für eine solche Wahl ist die Tukey-Verlustfunktion, wie sie in Abschnitt 3.1 schon erläutert wurde. Im zweiten Schritt wird die Konstante c größer gewählt als im ersten Schritt.

Gelöst wird Gleichung (3.6) dann mit Hilfe der Ableitung der Verlustfunktion

$$\begin{aligned} g_i &= \frac{\partial}{\partial z_i} L_1 \left(\frac{y_i - z_i}{\hat{\sigma}(\hat{F}_{M_1}(\mathbf{x}))} \right) \\ &= -\frac{1}{\hat{\sigma}(\hat{F}_{M_1}(\mathbf{x}))} l_1 \left(\frac{y_i - z_i}{\hat{\sigma}(\hat{F}_{M_1}(\mathbf{x}))} \right) \end{aligned} \tag{3.7}$$

für $i = 1, \ldots, N$. In Algorithmus 3 wird $g_i, i = 1, \ldots, N$, in jeder Iteration $M_1 + m, m = 1, \ldots, M_2$ berechnet und basiert jeweils auf der Vorhersage der vorherigen Iteration. Die z_i aus Gleichung (3.7) werden in der Anwendung durch $\hat{F}_{M_1+m-1}(\mathbf{x}_i)$ ersetzt, $i = 1, \ldots, N$. Die Ableitung L_1' sei mit l_1 bezeichnet.

Algorithmus 3 RRBoost

Eingabe:

1: Daten $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$
2: Anzahl Interationen 1. Phase M_1, 2. Phase M_2
3: Klasse der Basislerner $\mathcal{H}$ mit $h_{\boldsymbol{\theta}} \in \mathcal{H}$
4: Initialisierung $\hat{F}_0(\mathbf{x})$ LAD-Baum
5: Verlustfunktion L_1
6: **Phase 1:**
Führe SBoost für M_1 Iterationen aus und erhalte: $\hat{F}_{M_1}(\mathbf{x})$ und $\hat{\sigma}(\hat{F}_{M_1}(\mathbf{x}))$
7: **Phase 2:**
8: **for** $m = 1 \to M_2$ **do**:
9: **for** $i = 1 \to N$ **do**:
10: $g_{m,i} = -\frac{1}{\hat{\sigma}(\hat{F}_{M_1}(\mathbf{x}))} l_1 \left(\frac{y_i - \hat{F}_{M_1+m-1}(\mathbf{x}_i)}{\hat{\sigma}(\hat{F}_{M_1}(\mathbf{x}))} \right)$
11: **end for**
12: $\hat{\boldsymbol{\theta}}_m = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^N (g_{m,i} + h_{\boldsymbol{\theta}}(\mathbf{x}_i))^2$
13: $\alpha_m = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^N L_1 \left(\frac{y_i - \hat{F}_{M_1+m-1}(\mathbf{x}_i) - \alpha h_{\hat{\boldsymbol{\theta}}_m}(\mathbf{x}_i)}{\hat{\sigma}(\hat{F}_{M_1}(\mathbf{x}))} \right)$
14: $\hat{F}_{M_1+m}(\mathbf{x}) = \hat{F}_{M_1+m-1}(\mathbf{x}) + \alpha_m \cdot h_{\hat{\boldsymbol{\theta}}_m}(\mathbf{x})$
15: **end for**
16: **Ausgabe:** $\hat{F}_{M_1+M_2}(\mathbf{x})$

Dieser Gradient g_i wird wie in Zeile 12 mit einem Basislerner approximiert und analog zu Zeile 13 die Schrittweite $\alpha_m, m = 1, \ldots, M_1$, berechnet, die den minimalen Wert der neuen Prädiktion für die Verlustfunktion erzeugt. Schlussendlich wird die

alte Vorhersage durch den Summanden $\alpha_m \cdot h_{\boldsymbol{\theta}_m}(\mathbf{x})$ aktualisiert. Die finale Prädiktion ist dann gegeben durch $\hat{F}_{M_1+M_2}(\mathbf{x})$.
Durch die iterative Struktur des Verfahrens kann es leicht zu einer Überanpassung des Modells an die Daten kommen. Um dies zu vermeiden, haben Ju und Salibián-Barrera (2021) einen frühzeitigen Stoppmechanismus eingeführt.

Early Stopping
Da das RRBoost-Verfahren komplex ist und sich daher schnell zu stark an die Trainingsdaten anpassen kann, wird eine Art Regularisierung eingeführt. Diese Regularisierung passiert mittels einer sogenannten „Early Stopping“ Regel. Dazu wird die Performance der Zielvariablen in jeder Boosting-Runde auf dem Validierungsdatensatz gemessen. Abgebrochen wird dann bei derjenigen Iteration, welche den kleinsten Wert der jeweiligen Verlustfunktion erreicht. Die beiden Schritte des RRBoost-Verfahrens werden dabei einzeln betrachtet. Das heißt, die Gesamtanzahl an Boosting-Runden setzt sich zum Schluss aus der Summe der Early Stopping Iterationen zusammen. Für weitere Informationen siehe Ju und Salibián-Barrera (2021).

3.3 K-Vorzeichen-Tiefe und verwandte Konzepte

Das *K-Vorzeichen-Boosting* ist ein ebenfalls robustes Boosting-Verfahren für Regression und kombiniert die klassische Boosting-Idee mit dem simplen Maß der *K-Vorzeichen-Tiefe*. Dieses Verfahren wird in Kapitel 4 konzipiert, sodass in diesem Abschnitt zunächst die dazu nötigen statistischen Methoden vorgestellt werden. Ideen zur Weiterentwicklung sind im Ausblick dieser Arbeit zu finden. Die Implementierung basiert hauptsächlich auf dem R-Paket *GSignTest* (Horn (2021a)). In diesem Paket sind auch alle in diesem Abschnitt vorgestellten Funktionen enthalten.

Bei Verwendung des K-Vorzeichen-Boosting wird der Einfachheit halber ein linearer Zusammenhang zwischen den Einfluss- und der Zielvariablen angenommen, sodass die Zielfunktion F durch

$$\hat{F}(\mathbf{x}) = \theta_0 \mathbb{1} + \theta_1 \mathbf{x}^{(1)} + \ldots + \theta_p \mathbf{x}^{(p)} = (\mathbb{1}, \mathbf{x})\, \boldsymbol{\theta} \tag{3.8}$$

dargestellt werden kann. In Gleichung (3.8) wird $\mathbb{1} = (\overbrace{1, \ldots, 1}^{N-mal})^T$ notiert und mit der Schreibweise $(\mathbb{1}, \mathbf{x})$ ist das Anfügen des Vektors $\mathbb{1}$ an die Datenmatrix $\mathbf{x}$ gemeint.

Der Parametervektor $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_p)^T \in \Theta \subset \mathbb{R}^{p+1}$ stellt den Modellparametervektor dar. Wird ein Modell ohne Intercept angenommen, setzt man $\theta_0 = 0$.
Die Idee des K-Vorzeichen-Boosting ist es auf iterative Weise ein Modell zu konstruieren, ähnlich zum klassischen Boosting. Es gilt das Verhalten der Daten, beschrieben durch eine Zielfunktion F, möglichst gut zu approximieren.

K-Vorzeichen-Tiefe

Im Fokus des Verfahrens steht die K-Vorzeichen-Tiefe, kurz auch K-Tiefe, die die Anpassung des Modells an die Daten quantifiziert. Die Definition der K-Tiefe ist unter anderem in Leckey, Malcherczyk und Müller (2020) zu finden und wird hier mit $d_K(r_1(\boldsymbol{\theta}), \ldots, r_N(\boldsymbol{\theta}))$ bezeichnet. Die Funktion benötigt einen Parameter $K \in \mathbb{N}$, der die Größe der Tupel angibt, die betrachtet werden. Diese Tupel bestehen aus den Residuen $r_1(\boldsymbol{\theta}), \ldots, r_N(\boldsymbol{\theta})$, mit $r_i(\boldsymbol{\theta}) \in \mathbb{R}, i = 1, \ldots, N$. Die $r_i(\boldsymbol{\theta})$ sind Realisierungen der Zufallsvariablen $R_i(\boldsymbol{\theta}), i = 1, \ldots, N$. Die Abhängigkeit der Residuen von $\boldsymbol{\theta}$ beschreibt deren Zugehörigkeit zu einem Modell mit Parametervektor $\boldsymbol{\theta}$, wie in Gleichung (3.8). Die Residuen können also dargestellt werden durch

$$r_i(\boldsymbol{\theta}) = y_i - (1, \mathbf{x}_i) \cdot \boldsymbol{\theta}, \qquad i = 1, \ldots, N.$$

Die K-Tiefe gibt den relativen Anteil der geordneten K-elementigen Tupel der Residuen $r_i(\boldsymbol{\theta})$ an, die alternierende Vorzeichen aufweisen und ist definiert durch

$$d_K(r_1(\boldsymbol{\theta}), \ldots, r_N(\boldsymbol{\theta})) := \frac{1}{\binom{N}{K}} \sum_{1 \le n_1 < n_2 < \ldots < n_K \le N} \left(\prod_{k=1}^{K} \mathbb{1}\{(-1)^k r_{n_k}(\boldsymbol{\theta}) > 0\} + \prod_{k=1}^{K} \mathbb{1}\{(-1)^k r_{n_k}(\boldsymbol{\theta}) < 0\} \right). \tag{3.9}$$

Damit steigt der Wert der K-Tiefe in Gleichung (3.9) mit jedem K-Tupel mit alternierenden Vorzeichen an. Daher impliziert eine große Tiefe einen besseren Fit. Da sie nur auf den Vorzeichen der Residuen beruht, ist die K-Tiefe ein sehr robustes Kriterium um eine Modellanpassung zu messen. Die K-Vorzeichen-Tiefe betrachtet dabei alle K-Tupel der Residuen, auch solche, die nicht direkt aufeinander folgen. Dies inkludiert also alle K-elementigen Tupel der Residuen, die mit Berücksichtigung der Reihenfolge kombiniert werden können.
Diese Funktion ist unter dem Namen *calcDepth* im R-Paket *GSignTest* (Horn (2021a)) implementiert. Dabei wird angenommen, dass $\mathbb{P}(R_i(\boldsymbol{\theta}) = 0) = 0, i = 1, \ldots, N$. Daher

ist die Funktion so implementiert, dass wenn ein $r_i(\boldsymbol{\theta}), i = 1, \ldots, N$ existiert, dieses Residuum zunächst komplett aus dem Residuenvektor herausgenommen wird. Dann erst beginnt die Kalkulation der K-Vorzeichen-Tiefe mit der erfüllten Bedingung.

Vereinfachte K-Vorzeichen-Tiefe

Wird die K-Vorzeichen-Tiefe leicht modifiziert, resultiert daraus die *vereinfachte K-Vorzeichen-Tiefe*. Der Unterschied der beiden Kriterien liegt bei der Auswahl der K-Tupel, die für die Kalkulation der Tiefen genutzt werden. Die vereinfachte K-Vorzeichen-Tiefe betrachtet nur solche Tupel, dessen Elemente unmittelbar aufeinanderfolgen. Sie ist definiert durch

$$d_K^S(r_1(\boldsymbol{\theta}), \ldots, r_N(\boldsymbol{\theta})) := \frac{1}{N-K+1} \sum_{n=1}^{N-K+1} \left(\prod_{k=1}^{K} \mathbb{1}\{(-1)^k r_{n+k-1}(\boldsymbol{\theta}) > 0\} + \prod_{k=1}^{K} \mathbb{1}\{(-1)^k r_{n+k-1}(\boldsymbol{\theta}) < 0\} \right). \tag{3.10}$$

Die vereinfachte K-Tiefe in Gleichung (3.10) gibt den relativen Anteil der aufeinanderfolgenden K-elementigen Tupel der Residuen an, die alternierende Vorzeichen aufweisen. Auch hier bedeutet ein größerer Wert der Tiefe einen besseren Fit des zugehörigen Modells an die Daten.

Nächste-Nachbarn-Ordnungsverfahren

Da es um die Eigenschaft des Alternierens geht, sind sowohl die volle als auch vereinfachte K-Vorzeichen-Tiefe stark abhängig von der Ordnung der Residuen. In Horn (2021b) werden verschiedene Ordnungsverfahren für den Fall des einfachen sowie multiplen linearen Modells vorgestellt und verglichen. Vor allem hervorzuheben ist das Ordnungsverfahren des nächste-Nachbarn Ansatz (kurz: NN, englisch: nearest neighbor). Dies ist ein abstandbasiertes Verfahren, das heißt, die Ordnung der Residuen wird über die Abstände der Datenpunkte generiert. Gemessen wird der Abstand mit der euklidischen Norm $\|\mathbf{d}\|_2 = \|(d_1, \ldots, d_p)^T\|_2 = (\sum_{i=1}^{p} d_i^2)^{\frac{1}{2}}$. Die Datenpunkte werden nach ihrem kleinsten Abstand zur nächsten Beobachtung geordnet. Jeder Datenpunkt $\mathbf{x}_i, i = 1, \ldots, N$ wird einmal als Startpunkt genutzt. Es wird also N-mal ein approximativ kürzester Weg durch die Daten gesucht. Die finale Ordnung der Residuen ist dann der approximativ kürzeste Weg über alle N kürzesten Wege durch die Daten. Jeder Datenpunkt darf dabei nur einmal passiert werden, sodass Start- und Endpunkt des Weges nicht gleich sind. Dieses Problem ist auch bekannt als das

Hamiltonkreisproblem (kurz: SHP, englisch: shortest hamiltonian path problem). Da die exakte Lösung des Problems sehr rechenaufwendig ist und daher eine sehr lange Laufzeit des Algorithmus mit sich bringt wird im Rahmen dieser Arbeit ausschließlich mit der approximativen Lösung gerechnet. Diese approximative Lösung wird mit Hilfe des NN-Ansatzes ermittelt. Weitere Informationen dazu sind Horn (2021b) zu finden.

(Vereinfachter) K-Vorzeichen-Tiefe-Test

Basierend auf der K-Vorzeichen-Tiefe kann der *K-Vorzeichen-Tiefe-Test* konzipiert werden. Er betrachtet für $\theta_0 \in \mathbb{R}$ die Nullhypothese

$$H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0.$$

Die Teststatistik ist mit $K \geq 2$ gegeben durch

$$T_K(\boldsymbol{\theta}) := T_K(R_1(\boldsymbol{\theta}), \ldots, R_N(\boldsymbol{\theta})) = N \left(d_K(R_1(\boldsymbol{\theta}), \ldots, R_N(\boldsymbol{\theta})) - \left(\frac{1}{2}\right)^{K-1} \right). \tag{3.11}$$

Dabei wird angenommen, dass die $r_i(\boldsymbol{\theta}), i = 1, \ldots, N$ Realisationen der Zufallsvariablen $R_1(\boldsymbol{\theta}), \ldots, R_N(\boldsymbol{\theta})$ sind. Die Form der Teststatistik wird durch die Existenz einer asymptotischen Verteilung dieser begründet. Erfüllen die $R_i(\boldsymbol{\theta}), i = 1, \ldots, N$ die Bedingungen

$$R_1(\boldsymbol{\theta}), \ldots, R_N(\boldsymbol{\theta}) \text{ sind unabhängig und} \tag{3.12}$$

$$\mathbb{P}(R_i(\boldsymbol{\theta}) > 0) = \mathbb{P}(R_i(\boldsymbol{\theta}) < 0) = \frac{1}{2} \text{ mit } i = 1, \ldots, N, \tag{3.13}$$

dann gilt für $N \to \infty$ und $K \geq 3$

$$T_K(\boldsymbol{\theta}) \xrightarrow{d} \Psi_K(W). \tag{3.14}$$

Mit $W = (W_t)_{t \in [0,1]}$ wird die standardisierte Brownsche Bewegung notiert. Die Funk-

tion Ψ_K ist dabei definiert durch

$$\begin{aligned}
\Psi_3(W) &= \frac{3}{4}\left(1 - \int_0^1 (W_1 - 2W_t)^2 \mathrm{d}t\right), \\
\Psi_K(W) &= -\frac{K!}{4(K-4)!}\int_{-0.5}^{1}\int_{t\vee 0}^{t+0.5}\left(\frac{1}{2}+t-s\right)^{K-4}((W_{s\wedge 1} - W_{t\vee 0})^2 - \dots \\
&\qquad \dots((s\wedge 1) - (t\vee 0)))\,\mathrm{d}s\mathrm{d}t \\
&\quad -\frac{K!}{2(K-4)!}\int_{0.5}^{1}\int_{0}^{t-0.5}\left(\frac{1}{2}+s-t\right)^{K-4} W_s(W_1 - W_t)\,\mathrm{d}s\mathrm{d}t, \quad K \geq 4.
\end{aligned}$$

Der Beweis von Gleichung (3.14) kann in Malcherczyk, Leckey und Müller (2021) nachgeschlagen werden. Wird mit q_α das α-Quantil der asymptotischen Verteilung der Teststatistik $T_K(\boldsymbol{\theta})$ in Gleichung (3.11) unter $\boldsymbol{\theta}$ bezeichnet, dann ist der K-Vorzeichen-Tiefe-Test gegeben durch

$$\text{verwerfe } H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0 \text{ wenn } T_K(\boldsymbol{\theta}_0) < q_\alpha.$$

Das heißt, die Nullhypothese H_0 wird abgelehnt, wenn die K-Tiefe zu klein ausfällt. Das spricht dafür, dass das zum Parameter $\boldsymbol{\theta}$ zugehörige Modell die Daten nicht genügend gut widerspiegelt. Für $K = 2$ resultiert der klassische Vorzeichen-Test.
Analog wird auch der K-Vorzeichen-Tiefe-Test mit der vereinfachten K-Vorzeichen-Tiefe konstruiert. In Kustosz, Müller und Wendler (2016) wird die Teststatistik definiert durch

$$T_K^S(\boldsymbol{\theta}) := \sqrt{N-K+1}\frac{d_K^S(R_1(\boldsymbol{\theta}),\dots,R_N(\boldsymbol{\theta})) - \left(\frac{1}{2}\right)^K}{\sqrt{\left(\frac{1}{2}\right)^K \cdot [3 - \left(\frac{1}{2}\right)^{K-1}\cdot K - 3\cdot\left(\frac{1}{2}\right)^K]}}.$$

Unter den Bedingungen (3.12) und (3.13) ist die asymptotische Verteilung der Teststatistik $T_K^S(\boldsymbol{\theta})$ für $N \to \infty$ gegeben durch die Standardnormalverteilung, es gilt also

$$T_K^S(\boldsymbol{\theta}) \xrightarrow{d} \mathcal{N}(0,1).$$

für $K \geq 3$. Wird das α-Quantil der Standardnormalverteilung notiert durch q_α^S dann wird die Nullypothese $H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0$ verworfen, wenn, $T_K^S(\boldsymbol{\theta}_0) < q_\alpha^S$, also nicht genug aufeinanderfolgende K-Tupel in den Daten vorhanden sind, die alterniere Vorzeichen aufweisen.

4 Konstruktion und Implementierung des K-Vorzeichen-Boosting

Im folgenden Kapitel wird die Konstruktion und Implementierung des K-Vorzeichen-Boosting erläutert. Anwendung finden dabei die in Abschnitt 3.3 vorgestellten statistischen Methoden. Dabei soll vor allem auch thematisiert werden, welche Ansätze im Verlauf der Konstruktion verfolgt wurden, welche Probleme aufgetreten sind und welche Lösungsansätze in Folge dessen in Betracht gezogen und verfolgt wurden.
Wie schon in Abschnitt 3.3 angedeutet wird ausschließlich von einem linearen Zusammenhang zwischen den Einfluss- und der Zielvariablen ausgegangen. Die Zielfunktion ist folglich durch

$$F(\mathbf{x}) = \theta_0 \mathbb{1} + \theta_1 \mathbf{x}^{(1)} + \ldots + \theta_p \mathbf{x}^{(p)} = (\mathbb{1}, \mathbf{x})\, \boldsymbol{\theta} \tag{4.1}$$

gegeben. Charakterisiert wird die zugehörige Modellgleichung durch den Parametervektor $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_p)^T \in \Theta \subset \mathbb{R}^{p+1}$. Mit dieser Annahme sind die zugehörigen Residuen gegeben durch

$$r_i(\boldsymbol{\theta}) = y_i - (1, \mathbf{x}_i) \cdot \boldsymbol{\theta}, \qquad i = 1, \ldots, N. \tag{4.2}$$

Die $r_i(\boldsymbol{\theta})$ sind Realisierungen der Zufallsvariablen $R_i(\boldsymbol{\theta}) = Y_i - (1, \mathbf{x}_i) \cdot \boldsymbol{\theta}$ für $i = 1, \ldots, N$. Für die hier betrachteten linearen Regressionsmodelle der Form $Y_i = (1, \mathbf{x}_i) \cdot \boldsymbol{\theta} + \mathcal{E}_i$ gilt, dass die Zufallsvariablen $R_1(\boldsymbol{\theta}), \ldots, R_N(\boldsymbol{\theta})$ unabhängig sind und die Bedingung

$$\mathbb{P}_{\boldsymbol{\theta}}(R_i(\boldsymbol{\theta}) > 0) = \frac{1}{2} = \mathbb{P}_{\boldsymbol{\theta}}(R_i(\boldsymbol{\theta}) < 0), \qquad i = 1, \ldots, N \tag{4.3}$$

erfüllt ist. Denn im Rahmen dieser Arbeit werden ausschließlich stetige, um 0 symmetrische Fehlerverteilungen betrachtet.

4.1 Motivation

Der K-Vorzeichen-Tiefe-Test kann als Verallgemeinerung des Vorzeichen-Tests gesehen werden, denn für $K = 2$ ist der Vorzeichen-Test ein Sonderfall des K-Vorzeichen-Tests. Der Vorzeichen-Test kann die Nullhypothese $H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0$ testen. Es wird also überprüft ob ein Modell mit Parameter $\boldsymbol{\theta}_0$ hinreichend gut an die Daten angepasst ist. Das Kriterium zum Ablehnen der Nullhypothese ist durch die Anzahl der positiven Vorzeichen der Residuen $r_i(\boldsymbol{\theta}), i = 1, \ldots, N$, gegeben. Die Nullhypothese wird verworfen, wenn diese Anzahl zu groß oder zu klein ausfällt. Der Vorzeichen-Test weist jedoch für manche Hypothesen eine sehr schlechte Güte auf, wie es zum Beispiel in Abbildung 4.1 der Fall ist.

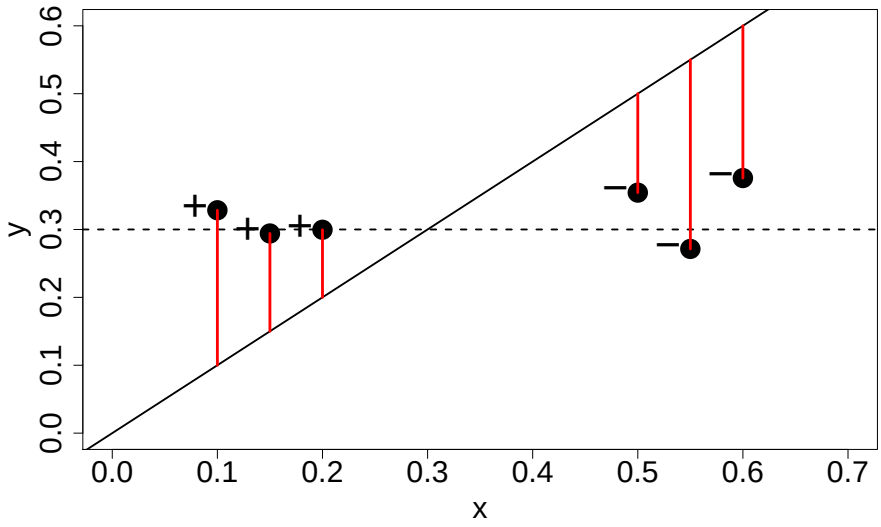

Abbildung 4.1: $N = 6$ Beobachtungen, die aus dem Modell $Y_i = 0.3 + 0.05\,\mathcal{E}, i = 1, \ldots, N$ gezogen wurden. Dabei gilt $\mathcal{E} \sim \mathcal{N}(0, 1)$. Die gestrichelte Linie ist die konstante Gerade mit dem wahren Parameter $\boldsymbol{\theta} = (0.3, 0)^T$ und die durchgezogene Linie zeigt einen Fit mit $\boldsymbol{\theta}_0 = (0, 1)^T$. Für die ersten drei Datenpunkte resultiert ein positives Vorzeichen für die Residuen, für die letzten drei ein Negatives.

Es sind gleich viele positive wie negative Vorzeichen der Residuen vorhanden. Hier würde der Test die Nullhypothese nicht ablehnen, obwohl $\boldsymbol{\theta}_0$ weit entfernt ist vom wahren Wert für $\boldsymbol{\theta}$. Die K-Vorzeichen-Tiefe adressiert dieses Problem. Es werden nicht nur die Vorzeichen der Residuen gezählt, also nicht nur 2-Tupel betrachtet, sondern K-Tupel für $K \geq 3$. Eine große Tiefe setzt nun mehr voraus, dass die an die Daten angepasste Gerade innerhalb der Datenpunkte liegt. In dem Beispiel aus

Abbildung 4.1 erreicht die (vereinfachte) K-Tiefe für $K \geq 3$ ihren kleinstmöglichen Wert $d_K(r_1(\boldsymbol{\theta}), \ldots, r_N(\boldsymbol{\theta})) = d_K^S(r_1(\boldsymbol{\theta}), \ldots, r_N(\boldsymbol{\theta})) = 0$, da keine (aufeinanderfolgenden) K-Tupel mit alternierenden Vorzeichen vorhanden sind. Der K-Vorzeichen-Tiefe-Test würde die Nullhypothese also ablehnen.

4.2 Ordnen der Residuen

Bevor ein erster Ansatz für die Implementierung des K-Vorzeichen Boosting vorgestellt werden kann, ist zunächst die Frage zu klären, welche Ordnungsstrategie für die Residuen zur Berechnung der K-Vorzeichen-Tiefe verwendet wird. Wie in Abschnitt 3.3 schon erläutert, spielt die Anordnung der Residuen für das Ergebnis der K-Vorzeichen-Tiefe und damit auch für den K-Vorzeichen-Tiefe-Test eine große Rolle. Da die Residuen von den Werten der Einfluss- und Zielvariablen abhängen, basiert das Ordnen der Residuen auf der Sortierung der Einflussvariablen.

Im Fall der einfachen linearen Regression wird der Zusammenhang zwischen einer Einfluss- und einer Zielvariablen betrachtet. Dabei werden die Residuen nach den Werten der einzigen Einflussvaribalen der Größe nach geordnet, wie es in Horn (2021b) beschrieben wird. Doch tritt mehr als eine Einflussvariable auf, ist die Reihenfolge der Residuen auf diese Weise nicht mehr eindeutig bestimmbar. Horn (2021b) adressiert dieses Problem und stellt verschiedene Varianten für das Ordnen der Residuen im Mehrdimensionalen gegenüber.

Für das Ordnen der Residuen bei der Berechnung der K-Vorzeichen-Tiefe im Fall der multiplen linearen Regression ist einer der intuitivsten Ansätze die Datenpunkte nach den Werten einer der Einflussvariablen zu sortieren, sodass die Werte aller anderen Einflussvariablen als Kriterium für das Ordnen außen vor gelassen werden. Diese Vorgehensweise hat vor allem den Vorteil, dass sie sehr simpel zu verstehen und anzuwenden ist und daher eine sehr geringe Laufzeit aufweist. Dieser Ansatz kann vor allem für Zeitreihen sinnvoll sein, denn dort steht die Zeitvariable im Fokus. Ein großer Nachteil, wenn lediglich die Werte einer Variablen betrachtet werden ist jedoch, dass die Informationen der Werte der $p - 1$ anderen Variablen für das Ordnen verloren gehen und die Wahl der Ordnungsvariable sehr großes Gewicht hat. Eine zweite, komplexere Ordnungsstrategie ist das nächste-Nachbarn Verfahren (kurz: NN, englisch: nearest neighbor) wie es schon in Abschnitt 3.3 erläutert wurde. Es

basiert ausschließlich auf den paarweisen Abständen der Datenpunkte und bezieht alle Einflussvariablen mit ein. Die Idee dieser Ordnungsstrategie ist es, dass Datenpunkte, die nah beieinander liegen, diese Eigenschaft auch beim Ordnen beibehalten sollten. Dadurch, dass das Sortieren ausschließlich die Abstände der Datenpunkte mit einbezieht, spielen die Absolutwerte der Einflussvariablen keine Rolle für die Ordnung. Diese Eigenschaft bringt zwei Vorteile mit sich. Es können nicht nur metrische Einflussvariablen betrachtet werden, sondern auch nominale und ordinale Variablen, da auch für diese Arten von Ausprägungen Abstandsmetriken existieren. Ein weiterer Vorteil des NN-Ordnungsverfahrens ist, wie es Horn (2021b) hervorhebt, dass die Werte der Einflussvariablen in additiver und multiplikativer Weise transformiert werden können ohne dabei die Ordnung der Residuen zu verändern. Eine additive Transformation von zwei Werten ändert den Abstand dieser Werte nicht und auch durch eine skalare Multiplikation bleiben die Proportionen der Abstände erhalten. Ein Nachteil dieser Vorgehensweise gegenüber der Sortierung nach einer Variable ist jedoch weiterhin die vergleichsweise langsame Laufzeit. Das NN-Ordnungsverfahren ist eine approximative Lösung des SHP-Problems.
Eine Gegenüberstellung der drei Verfahren ist mittels eines Beispiels in Abbildung 4.2 gegeben.

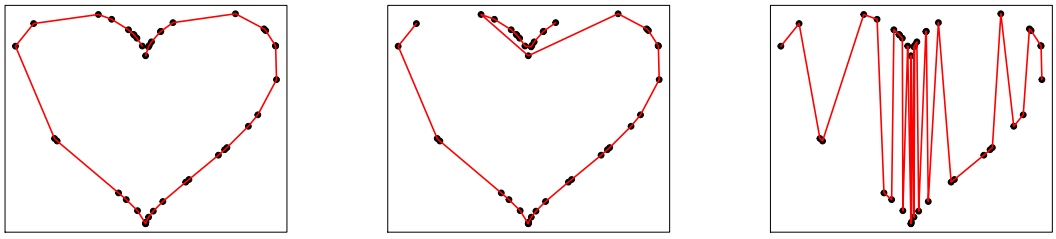

Abbildung 4.2: Verschiedene Ordnungsverfahren für einen Datensatz mit zwei Einflussvariablen, deren Ausprägungen eine Herzstruktur aufweisen. Gegenüberstellung des exakten SHP-Ordnungsverfahrens (erstes Bild von links), des NN-Ordnungsverfahrens (zweites Bild) und des Ordnungsverfahrens nach einer Variablen (drittes Bild).

Der Datensatz weist als natürliche Ordnung eine Herzstruktur auf. Das erste Bild zeigt die Anwendung des exakten SHP-Ordnungsverfahren und gibt die natürliche Struktur genau wieder. In allen drei Bildern sind die Pfade nicht geschlossen, da bei der Sortierung der Residuen Start- und Endpunkt nicht gleich sind. Das approxima-

tive NN-Verfahren im mittleren Bild, gibt grob die Herzstruktur wieder, jedoch ist das Verfahren an der oberen Spitze des Herzens inakkurat. Für die deutlich geringere Laufzeit sind solche Abweichungen jedoch vertretbar. Im letzten der drei Bilder sind die Ergebnisse des Ordnungsverfahren nach einer der beiden Variablen aufgezeigt. Die natürliche Ordnung wird komplett ignoriert und es ist keine Struktur zu erkennen.

Aufgrund der vielfältigen Anwendungsmöglichkeiten und der trotzdem bestehenden Einfachheit des Verfahren, wird in dieser Arbeit ausschließlich das NN-Verfahren zum Ordnen der Residuen für die Kalkulation der (vereinfachten) K-Vorzeichen-Tiefe verwendet.

4.3 Ein erster Ansatz

Die grundlegende Idee des K-Vorzeichen-Boostings ist es den Parametervektor $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_p)^T$ aus Gleichung (4.1) so zu schätzen, dass die zugehörigen Residuen des Modells eine maximale K-Vorzeichen-Tiefe erreichen. Für den Anfang werden zunächst nur Modelle ohne Intercept betrachtet, das heißt es wird $\theta_0 = 0$ gesetzt. Ein erster Ansatz für die Implementierung des Vorgehens ist in Algorithmus 4 aufgeschrieben. Gegeben sind ein Datensatz $\mathcal{D}$, ein Wert K für die Länge der Tupel

Algorithmus 4 K-Vorzeichen-Boosting: Ansatz 1

Eingabe:

Daten $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, Tiefe K, Anzahl Iterationen M

Initialisierung:

$\boldsymbol{\theta}_0 = (\theta_1, \ldots, \theta_p)^T = \mathbf{0}, p$ Anzahl Einflussvariablen,
$(r_1^{[0]}, \ldots, r_N^{[0]})^T = \mathbf{r}^{[0]} = \mathbf{y} = (y_1, \ldots, y_N)^T$

for $m = 1$ to M **do**:

  for $j = 1$ to p **do**:

    $\hat{\theta}_j \in \underset{\theta \in \Theta_j \subset \mathbb{R}}{\operatorname{argmax}}\, d_K(r_1^{[m-1]} - \theta x_{1j}, \ldots, r_N^{[m-1]} - \theta x_{Nj})$

  end for

  Bestimme $\hat{j} \in \underset{j \in \{1,\ldots,p\}}{\operatorname{argmax}}\, d_K(r_1^{[m-1]} - \hat{\theta}_j x_{1j}, \ldots, r_N^{[m-1]} - \hat{\theta}_j x_{Nj})$

  Update: $\mathbf{r}^{[m]} = \mathbf{r}^{[m-1]} - \hat{\theta}_{\hat{j}} \boldsymbol{x}_{\hat{j}}$ und $\theta_{\hat{j}} = \theta_{\hat{j}} + \hat{\theta}_{\hat{j}}$

  if K-Vorzeichen-Tiefe-Test lehnt die Hypothese $H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0$ nicht ab **then**

    break

  end if

end for

Ausgabe: $\boldsymbol{\theta_0}$

in der K-Vorzeichen-Tiefe und eine Anzahl M der Boosting-Runden. Das Vorgehen des Algorithmus soll der allgemeinen Boosting-Struktur treu bleiben, das heißt, die Werte der Vorhersagen beziehungsweise Parameter werden sequentiell verbessert und an die jeweiligen wahren Werte herangeführt. Um eine sequentielle Verbesserung erreichen zu können, muss ein Startpunkt gesetzt werden. Der Parametervektor $\boldsymbol{\theta}$ zunächst auf 0 gesetzt, da dies eine neutrale Wahl ist. Die Residuen $r_1^{[0]}, \ldots, r_N^{[0]}$ sind zu Beginn durch die Werte der Zievariablen gegeben. Jede Boosting-Iteration $m = 1, \ldots, M$ kann dann in vier Sinnschritte aufgeteilt werden. Anfangs wird für jede der $j = 1, \ldots, p$ Variablen derjenige Schätzer $\hat{\theta}_j \in \Theta_j \subset \mathbb{R}$ gesucht, sodass die neuen Residuen $r_1^{[m-1]} - \theta_j x_{1j}, \ldots, r_N^{[m-1]} - \theta_j x_{Nj}$ die K-Vorzeichen-Tiefe maximieren. Es gilt dasjenige $\hat{\theta}_j$ zu finden, welches für die neu formierten Residuen die meisten K-Tupel mit alternierenden Vorzeichen erzeugt. Denn diese Eigenschaft impliziert eine gute Anpassung des Modells an die vorgegebenen Daten. Für diese Suche wird die Funktion *optim* in R benutzt. Hinter *optim* steckt ein Optimierungsalgorithmus, der für allgemeine Optimierungsprobleme verwendet werden kann. Im Rahmen dieser Arbeit verwendet man dazu das simulierte Abkühlungsverfahren (kurz: SA, englisch: simulated annealing). Weitere Informationen zum Verfahren können in Bélisle (1992) nachgelesen werden. Die genauen Einstellungen sind im zugehörigen R-Code zu finden. Mit Hilfe eines vorgegebenen Startwerts wird die K-Vorzeichen-Tiefe maximiert. Die Funktion hat den Vorteil, dass nur ein Startwert angegeben werden muss und der optimale Parameter $\hat{\theta}_j$ für jede Variable dann von *optim* gesucht und ausgegeben wird. Für ein hoffentlich besseres Ergebnis kann *optim* noch mit einer Gittersuche kombiniert werden. Das heißt, es wird für jede Variable ein Intervall für den zugehörigen Parameter angegeben und für jeden Wert aus diesem Intervall die *optim*-Funktion angewendet. Ist der Startwert nahe am Parameter dran, der die maximale Tiefe erzeugt, erhöht dies die Chance diesen Wert auch zu finden, denn die K-Vorzeichen-Tiefe ist nicht konvex.

Ist für jede Variable der nach dem vorangegangen Kriterium „beste“ Parameter gefunden, wird die Variable mit dem zugehörigen Parameter ermittelt, die in Kombination die größte Tiefe erzeugen. Dies ist in Zeile 7 in Algorithmus 4 aufgeschrieben. Da die K-Vorzeichen-Tiefe keine injektive Funktion ist, kann es sein, dass mehr als ein $\hat{\theta}_j \in \Theta_j$ gefunden wird, welches die Tiefe maximiert. In diesem Fall, wird der Parameter gewählt, welcher im Parameterbereich Θ_j als erster in der Reihenfolge steht. War also die Suche nach dem die Tiefe maximierenden Parameter erfolgreich, werden die Residuen dementsprechend geupdated. Das Produkt aus geschätztem Parameter

und zugehöriger Variable wird von den bisher entstandenen Residuen subtrahiert, wie es in Zeile 8 aufgezeigt ist. Diese Form des Updates kommt durch die additive Form der Zielfunktion zustande. Der ausgewählte Parameter $\hat{\theta}_{\hat{j}}$ wird, aufgrund der Additivität der Zielfunktion, zu dem jeweiligen bereits existierenden Vorfaktor der Variablen mit Index $\hat{j}$ hinzuaddiert. Im letzten Schritt jeder Boosting-Runde wird noch ein Abbruchkriterium überprüft, welches das Verfahren zu einem frühzeitigen Abbruch führt, wenn die Anpassung des Modells an die Daten schon ausreichend gut ist. Als Kriterium wird der K-Vorzeichen-Tiefe-Test verwendet, wie er in Abschnitt 3.3 beschrieben wird. Die Nullhypothese $H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0$ wird überprüft, wobei $\boldsymbol{\theta}_0$ den bis dahin konstruierten Parameterschätzer darstellt. Denn sind in den bis dahin konstruierten Residuen schon genug K-Tupel mit alternierenden Vorzeichen enthalten, lehnt der Test die Nullhypothese nicht ab und es wird angenommen, dass die Approximation der Zielfunktion die Daten schon ausreichend gut beschreibt.

Für eine erste Evaluation dieser Implementierung des K-Vorzeichen-Boosting werden Daten, wie es in Abschnitt 2.1 beschrieben ist, simuliert. Betrachtet werden, wegen der Einschränkung auf lineare Modelle ohne Intercept, nur die Funktionen

$$f_1(\mathbf{x}) = 10\mathbf{x}_1 \text{ und } f_2(\mathbf{x}) = 3\mathbf{x}_1 + 7\mathbf{x}_2.$$

Kombiniert werden diese Zielfunktionen jeweils mit allen vier in Abschnitt 2.1 vorgestellten Fehlerverteilungen. Damit können die drei Verfahren sowohl für unkontaminierte Daten verglichen werden, als auch für Daten, die wenig bis stark durch Ausreißer belastet sind. Insgesamt werden dabei $p = 10$ Einflussvariablen betrachtet, wobei nicht alle einen echten Effekt auf die Zielvariable haben. Die Einflussvariablen sind alle gleichverteilt auf $(0, 1)$ gewählt, sodass deren Ausprägungen im gleichen Wertebereich liegen. Für den Trainingsdatensatz werden zunächst, angelehnt an die Simulationsstudie aus Ju und Salibián-Barrera (2021), $N = 300$ Beobachtungen verwendet. Für den Testdatensatz werden $0.3 \cdot N = 90$ extra Datenpunkte simuliert, die mit $\mathcal{D}_1$ konstruiert sind und daher keine Ausreißer enthalten. Für das RRBoost-Verfahren sind 30% der Datenpunkte aus dem Trainingsdatensatz, also 90 Beobachtungen, Validierungsdaten. Der Validierungsdatensatz wird vor allem für das Early Stopping Kriterium verwendet, aber auch um die besten Parameter des Initialisierungs-LAD-Baum zu finden. Dabei wird die Mindestgröße des Baums und die Mindestanzahl an Beobachtungen pro Blatt bestmöglich festgelegt. Bestmöglich heißt dabei, dass die kleinste mittlere absolute Abweichung zwischen den erzeug-

ten Vorhersagen des jeweiligen resultierenden Modells bei Anwendung des jeweiligen Baumes und den wahren Werten der Zielvariablen am kleinsten ist. Die Beobachtungen werden für das K-Vorzeichen-Boosting vor der Anwendung nach dem NN-Verfahren geordnet. Für das Gradienten-Boosting und das RRBoost-Verfahren ist ein vorheriges Ordnen der Datenpunkte irrelevant und daher unnötig zeitaufwendig. Das Signifikanzniveau für das Abbruchkriterium des K-Vorzeichen-Boosting, also den K-Vorzeichen-Tiefe-Test, wurde auf 0.05 gesetzt. Auch über den Test hinaus wird eine Tiefe von 3 gewählt. Aufgrund der sehr langen Laufzeit der *optim*-Funktion wird für das K-Vorzeichen-Boosting zur Bestimmung des optimalen Parameters in Zeile 5 in Algorithmus 4 nur ein Startwert verwendet. Der Startpunkt wird aus der Studentschen t-Verteilung mit einem Freiheitsgrad gezogen. Alle weiteren Einstellungen der Verfahren können im zugehörigen R-Code nachgelesen werden. Aufgrund der langen Rechenzeit werden bei dem K-Vorzeichen-Boosting lediglich zehn Boosting-Runden eingestellt, bei den anderen beiden Verfahren jeweils 100. Weiter wird 10 mal simuliert, das heißt es werden 10 unabhängige Datensätze erzeugt. Nach dem Anpassen der Verfahren auf den Trainingsdatensatz wird der RMSE für jedes Verfahren und jede Simulation auf dem Testdatensatz berechnet und anschließend der Mittelwert über die 10 Datensätze kalkuliert. Die Ergebnisse dieser Simulationsstudie für die zwei verschiedenen Zielfunktionen f_1 und f_2 sind in Tabelle 4.1 und Tabelle 4.2 dargestellt. Schaut man sich die Werte in Tabelle 4.1 an, wo bei der Simulation der Daten

	RRBoost	**GBoost**	**KVBoost**
D_1	1.3687	1.2215	15.2200
D_2	1.4298	10.6957	16.6938
D_3	1.4139	2.3675	16.0743
D_4	1.4766	19.7534	16.2471

Tabelle 4.1: Mittelwerte des RMSE der drei Verfahren bei Verwendung der Zielfunktion f_1 mit 300 Trainings- und 90 Testbeobachtungen über 10 unabhängige Simulationen.

lediglich die erste Einflussvariable eine echten Effekt auf die Zielvariable hat, dann zeigt sich ein klares Bild. Das RRBoost-Verfahren erzielt durchweg für alle vier Fehlerverteilungen einen vergleichsweise kleinen mittleren RMSE. Alle vier Werte liegen zwischen 1.3687 für die Standardnormalverteilung der Fehler und 1.4766 für $\mathcal{D}_4$. Das RRBoost-Verfahren performt also sowohl für Daten ohne als auch mit Ausreißern

gut, so wie es Ju und Salibián-Barrera (2021) auch in ihrer Simulationsstudie zeigen. Die Werte des klassichen Gradienten-Boosting mit L2-Verlustfunktion, hier kurz mit „GBoost“ bezeichnet, weichen teilweise sehr stark von diesen Werten des RRBoost-Verfahrens ab. Denn bei den Fehlerverteilungen D_2 und D_4, die sehr stark durch Ausreißer belastet sind, sind die mittleren RMSE sehr groß. Mit Werten von 10.6957 und 19.7534 ist der mittlere RMSE deutlich größer als beim RRBoost-Verfahren. Bei der Fehlerverteilung D_3 mit kleineren Ausreißern ist der mittlere RMSE dagegen nur leicht größer mit einem Wert von 2.3675 gegenüber 1.4139 bei RRBoost. Im Fall der unkontaminierten Daten $\mathcal{D}_1$ schneidet das Gradienten-Boosting dagegen leicht besser ab als das RRBoost-Verfahren. Die Werte für das K-Vorzeichen-Boosting, kurz mit „KVBoost“ bezeichnet, zeigen ein anderes Bild. Denn die Werte der mittleren RMSE sind bei allen Einstellungen der Fehlerverteilung deutlich größer als beim RRBoost-Verfahren. Was jedoch positiv bewertet werden kann ist, dass der mittlere RMSE beim K-Vorzeichen-Boosting kleiner ist als beim Gradienten-Boosting bei der stark kontaminierten Verteilung $\mathcal{D}_4$. Dies ist schonmal ein Indiz dafür, dass das K-Vorzeichen-Boosting bei stark kontaminierten Daten besser beziehungsweise verlässlicher funktioniert als das Gradienten-Boosting. Der kleinste Wert wird beim KVBoost für die Fehlerverteilung D_1 erreicht, mit einem Wert von 15.2200 und am schlechtesten ist die Performance auf den Daten mit $\mathcal{D}_2$. Dort ergibt sich über die 10 Simulationen ein mittlerer RMSE von 16.6938. Diese Konsistenz der Werte ist nicht sehr verwunderlich, da die Resiuden Gleichung (4.3) erfüllen. Da nur die Vorzeichen der Datenpunkte betrachtet werden, sollte die Art der Fehlerverteilung hier keine allzu große Rolle spielen. Die Werte für die zweite Zielfunktion f_2 in Tabelle 4.2 zeigen ähnliches Verhalten. Der einzige Unterschied in der zweiten Tabelle ist, dass für die

	RRBoost	**GBoost**	**KVBoost**
D_1	1.2312	0.9969	15.7309
D_2	1.2294	14.9794	15.9090
D_3	1.4025	1.8164	16.1475
D_4	1.2737	14.3135	15.1359

Tabelle 4.2: Mittelwerte des RMSE der drei Verfahren bei Verwendung der Zielfunktion f_2 mit 300 Trainings- und 90 Testbeobachtungen über 10 unabhängige Iterationen.

Fehlerverteilung D_4 hier keine bessere Performance des K-Vorzeichen-Boosting vorzuweisen ist. Das K-Vorzeichen-Boosting und Gradienten-Boosting zeigen hier ähnlich

große Werte des mittleren RMSE über die 10 Testsimultionen. Die Werte zwischen den beiden Tabellen sind nicht vergleichbar, denn der RMSE ist ein absolutes und kein relatives Maß.

4.4 Untersuchen der K-Vorzeichen-Tiefe

Da die Performance dieser Implementierung des K-Vorzeichen-Boosting noch nicht zufriedenstellend ist, soll im Folgenden die Vorgehensweise des Verfahrens näher analysiert werden. Dazu wird erneut ein Datensatz mit der sehr simplen Zielfunktion f_1 in Kombination mit der Studentschen t-Verteilung mit einem Freiheitsgrad D_2 als Fehlerverteilung betrachtet. Um das Verhalten der K-Tiefe besser untersuchen zu können, wird zunächst nur die erste Iteration des Algorithmus 4 isoliert durchlaufen. Bei einer so simplen Zielfunktion, die eine Variable deutlich hervorhebt, sollte diese Einflussvariable schon in der ersten Iteration die größte K-Vorzeichen-Tiefe hervorbringen. Um das Verhalten der Tiefen der einzelnen Variablen besser visualisieren und damit analysieren zu können, wird eine Änderung vorgenommen im Vergleich zu der obigen Simulationsstudie. In Zeile 5 des Algorithmus 4 wird nicht die *optim*-Funktion mit einem randomisiert gewählten Startwert genutzt. Sondern es wird für jede der $p = 10$ Variablen die Tiefe über die Residuen $r_i^{[0]} - \theta x_{ij} = y_i - \theta x_{ij}, i = 1, \ldots, N$ der ersten Iteration mit $\theta \in [-20, 20] \cap \mathbb{Z}$ kalkuliert. Das Ergebnis ist in Abbildung 4.3 dargestellt.

Sofort ins Auge fällt, dass alle Kurven der K-Vorzeichen-Tiefe der verschiedenen Variablen ihr Maximum ungefähr bei dem gleichen Wert von $\theta = 10$ erreichen. Zu erwarten gewesen wäre, dass alle bis auf die erste Variable ihren Höhepunkt bei einem Parameterwert von 0 erreichen, da dies den wahren Werten der Parameter, also $\theta_j = 0$, für die Variablen im Modell entsprechen würde. Dass dieses Verhalten nicht erreicht wird, lässt sich durch die Art des Erzeugens der Variablen erklären. Dadurch, dass alle Realisierungen der Einflussvariablen aus $X \sim U(0, 1)$ gezogen werden, liegen die Ausprägungen aller Variablen in der gleichen Größenordnung. Die ähnliche Lokalisierung der Höhepunkte entsteht also in Folge dessen, dass in der ersten Iteration jede Variable zunächst einzeln für die K-Tiefe betrachtet wird. Damit ist gemeint, dass die Variablen, die keinen Einfluss haben, isoliert bewertet werden und somit noch kein Teil der Variabilität in den Daten durch die erste Variable erklärt wird. Werden die Wertebereiche der Einflussvariablen verschoben, verschieben sich auch die Maxima der Kurven. Die Abbildung A.1 im Anhang zeigt dieses Verhalten an

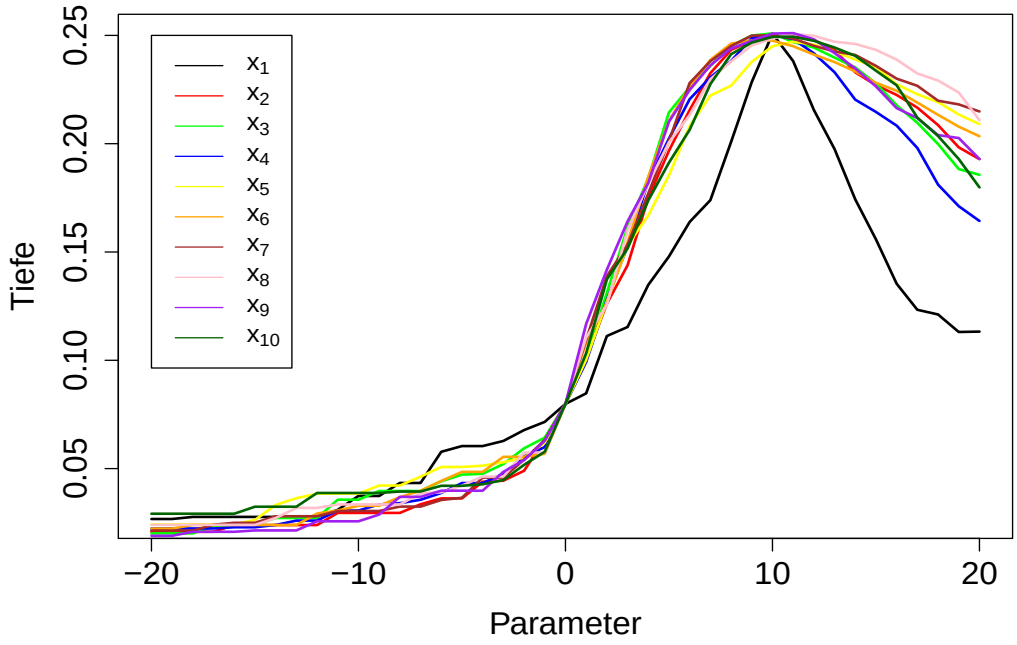

Abbildung 4.3: Linear interpolierte Kurven der K-Vorzeichen-Tiefe $d_K(y_1 - \theta x_{1j}, \ldots, y_N - \theta x_{Nj})$ der $j = 1, \ldots, 10$ Variablen für $\theta \in [-20, 20] \cap \mathbb{Z}$.

einem Beispiel.

Betrachtet man die Maxima der Kurven der K-Vorzeichen-Tiefe aus Abbildung 4.3 für die ausgewählten $\hat{\theta}_j \in [-20, 20] \cap \mathbb{Z}, j = 1, \ldots, 10$, dann bleibt das eindeutige Hervorheben der ersten Variable aus. Die Werte der Maxima sind in Tabelle 4.3 aufgelistet. Schaut man sich die Zahlen in der zweiten Spalte genauer an, fällt sogar auf, dass die erste Variable nicht die maximale K-Tiefe, über alle Variablen hinweg, erzeugt. Das Maximum der K-Tiefe der ersten Variablen, mit einem Wert von 0.2507, wird wie erwartet bei einem Parameterwert von 10 erreicht, allerdings ist die maximale K-Tiefe der neunten Variable, mit einem Wert von 0.2510, bei einem Parameterwert von 11 noch leicht größer. Und auch die Maxima der K-Vorzeichen-Tiefen aller anderen Variablen sind nur minimal schlechter. Dies wirft die Frage auf, inwieweit es nach dieser Beobachtung überhaupt sinnvoll ist die maximale K-Vorzeichen-Tiefe als Kriterium für die Anpassung des Modells zu verwenden, wenn in einem so simplen Fall schon keine Eindeutigkeit herrscht.
Zunächst fällt an dem Verlauf der Kurven in Abbildung 4.3 noch etwas auf. Wie

	K-Vorzeichen-Tiefe	$\theta_{j,\max}$
x_1	0.2507	10
x_2	0.2497	10
x_3	0.2509	10
x_4	0.2501	10
x_5	0.2470	11
x_6	0.2476	9
x_7	0.2500	10
x_8	0.2506	11
x_9	**0.2510**	11
x_{10}	0.2493	11

Tabelle 4.3: In der zweiten Spalte sind die Werte der Maxima der K-Vorzeichen-Tiefe für die Kurven in Abbildung 4.3 für die jeweilige Variable in erste Spalte aufgelistet. Die dritte Spalte gibt für jede Variable den Wert $\theta_{j,\max}$ des Parameters an, der die maximale K-Tiefe erzeugt.

zuvor schon festgestellt, haben alle Kurven einen ähnlichen Verlauf, nur die Kurve der ersten Variable weist eine Besonderheit auf. Denn um das Maximum herum ist die Kurve der ersten Variable viel steiler als die der restlichen Variablen. Das heißt, vor dem Maximum steigt die Kurve schneller an und danach fällt sie auch schneller wieder ab. Die Erklärung für diese Beobachtung ist sehr naheliegend. Denn wird jede Einflussvariable gesondert der Zielvariablen gegenübergestellt, dann streuen die Datenpunkte mit der ersten Variable viel weniger als die der anderen Einflussvariablen. Für das obige Beispiel sind Scatterplots in Abbildung 4.4 und Abbildung 4.5 für die Kombinationen mit der erste und neunten Variablen dargestellt. Die neunte Variable wird hier mit x_{max} bezeichnet, da sie diejenige Variable ist, die die maximale K-Tiefe der Variablen erzeugt. Die drei Geraden in den beiden Abbildungen stellen eine Anpassung des Modells mit bestimmtem Parameter dar. Die rote Gerade in Abbildung 4.4 ist beispielsweise durch die Gleichung $g(x_1) = \hat{\theta}_1 \cdot x_1$ gegeben. Die rote Gerade ist die Modellgerade, welche für die jeweilige Variable die maximale K-Tiefe erzeugt. Die grüne und blaue Gerade sind die Geraden, welche durch die zugehörigen zweitnächsten Nachbarparameter entstehen. Ist also beispielsweise $\hat{\theta_j} = 10$, dann gilt $\hat{\theta}_j + 2 = 12$ und $\hat{\theta}_j - 2 = 8$, da der Parameterbereich Θ_j nur ganze Zahlen enthält. Für eine bessere Sichtbarkeit des hier beleuchteten Sachverhalts, ist der Wertebereich der Einflussvariablen hier auf $(-5, 15)$ eingeschränkt. Dadurch werden einzelne Ausreißer aus dem Bild heraus geschnitten, da diese für die Erklärung vernachlässigbar sind.
In Abbildung 4.5 ist gut zu erkennen, dass die Datenpunkte sehr viel weiter um die

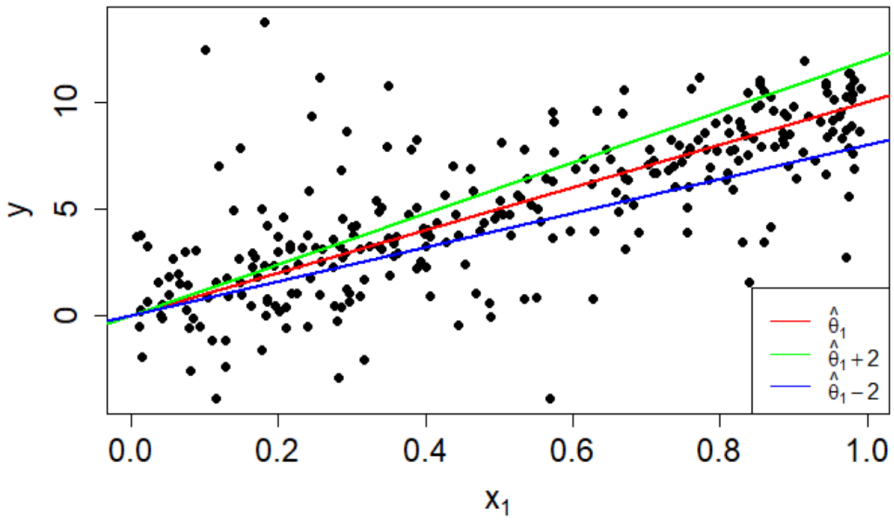

Abbildung 4.4: Scatterplot der Werte der ersten Variablen x_1 gegenüber den Ausprägungen der Zielvariablen y inklusive drei Modellgeraden.

Geraden streuen als in Abbildung 4.4. Diese Beobachtung ist auch erwartbar, denn die neunte Variable hat keinen echten Einfluss auf die Zielvariable, die erste hingegen schon. Die Einflussvariable und die Zielvariable sind unkorreliert und zeigen daher keinen Zusammenhang. Bei der ersten Variablen ergibt sich ein anderes Bild. Die Datenpunkte in Abbildung 4.4 liegen vergleichsweise nah an den Geraden und bilden einen schmaleren Schlauch um die rote Gerade und nicht in x_1-Richtung. Es stellt sich die Frage, warum die K-Vorzeichen-Tiefe der neunten Variablen trotzdem größer ist als die Tiefe der ersten Variablen? Dies liegt daran, dass die K-Tiefe nur die Vorzeichen der Residuen betrachtet, nicht aber deren absolute Größe. Im Scatterplot der neunten Variablen liegen links viele Datenpunkte über und rechts viele unterhalb der Geraden. Da die volle K-Tiefe mit $K = 3$ alle Tupel mit Länge 3 in die Bewertung mit einbezieht, tragen auch solche Tupel zu einer Erhöhung der Tiefe bei, dessen Einträge weit auseinanderliegen. Es ist also mehr oder weniger Zufall, welche Variable die maximale K-Tiefe erreicht. Der Grund für den ausgeprägten Anstieg um das Maximum der K-Vorzeichen-Tiefe bei der wahren Variable wird durch das Betrachten der grünen und blauen Linie in den beiden Abbildungen klarer. Ein Wegbewegen vom wahrem Parameter $\hat{\theta}_j$ impliziert ein Verändern der Steigung der Geraden. In Abbildung 4.4 liegen zwischen der roten und der grünen beziehungsweise der roten und der blauen Geraden viele Datenpunkte. In Abbildung 4.5 ist das nicht so. Dadurch, dass bei x_1 schon nur durch ein leichtes Verändern der Steigung der Geraden viele Residuen ihr Vorzeichen wechseln, verändert sich die K-Tiefe dadurch

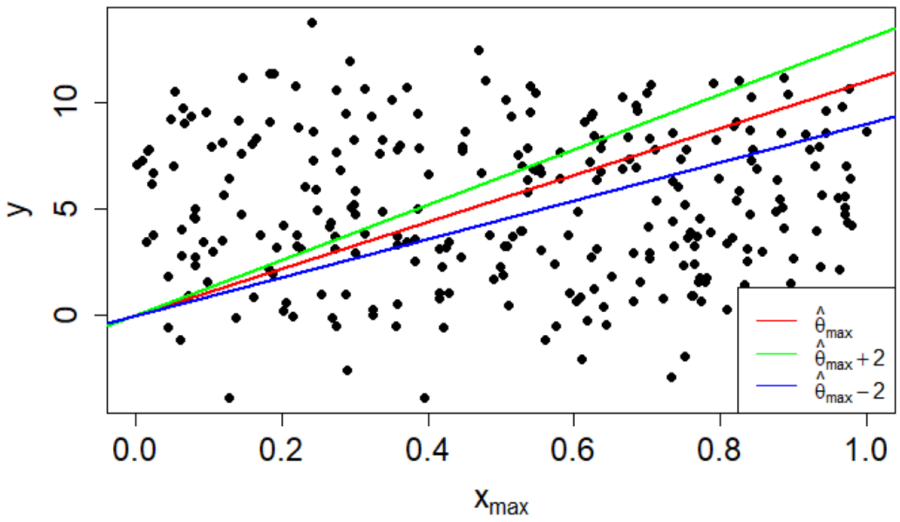

Abbildung 4.5: Scatterplot der Werte der neunten Variablen $x_9 = x_{max}$ gegenüber den Ausprägungen der Zielvariablen y inklusive drei Modellgeraden.

stärker, als wenn dieses Verhalten nur wenige der Residuen zeigen. Dadurch steigt die Kurve der K-Vorzeichen-Tiefe schneller an und fällt aber nach dem Erreichen des Maximums auch schneller wieder ab.

Nach dieser Erkenntnis stellt sich nun die Frage, wie ein solcher stärksten An- und Abstieg um das Maximum gemessen beziehungsweise gefiltert werden kann? Ein erste Idee wäre es, den größten L2-Abstand zu messen. Damit ist gemeint, dass ein Wert der K-Tiefe vorgegeben wird und dann der Abstand der Parameter, die diese K-Vorzeichen-Tiefe erzeugen, gemessen wird. Bei diesem Ansatz ist es jedoch erstmal nicht naheliegend wie dieser Wert der K-Tiefe gewählt werden soll.
Ein weiterer Ansatz für die Messung des steilsten Anstiegs um das Maximum der K-Tiefe wäre es, die Standardabweichung als Maß für die Streuung der Werte um den maximalen Wert der K-Vorzeichen-Tiefe zu verwenden. Dies ist jedoch nur schwer umzusetzen, da der Schätzer $\hat{\theta}_j$, der für jede Variable mit dem Index $j = 1, \ldots, p$, die maximale K-Tiefe erzeugt, nicht bekannt ist. Dadurch wird es schwierig ein geeignetes Intervall um das Maximum zu legen, indem die Standardabweichung berechnet werden soll. Wie in Abbildung 4.3 zu erkennen ist, liegt die Kurve der K-Vorzeichen-Tiefe der wahren Variablen ab einem Wert des Parameters $\hat{\theta}_j$ über den Kurven der anderen Variablen. Dies macht eine sinnvolle Wahl des Intervalls wichtig.

4.5 Ein zweiter Ansatz

Parameterwahl

Wie kann man also sinnvoll eine Vorauswahl für die Parameter treffen, für die das Verhalten der K-Vorzeichen-Tiefe analysiert wird? Beziehungsweise eine andere Formulierung der Frage wäre: Wie soll der Parameterbereich Θ_j aus Algorithmus 4 gewählt werden, sodass der wahre Parameterwert der Variablen θ_j möglichst darin liegt? Zu dieser Fragestellung soll im Folgenden eine mögliche Lösung vorgestellt werden.

Ein Ansatz für die Lösung dieses Problems ist es, die Datenpunkte an sich dafür zu verwenden. Damit ist gemeint, dass jeweils die Variable mit Index j, für die der Parameterbereich gesucht wird, mit der Zielvariablen kombiniert wird. Dazu werden jeweils alle Datenpunkte der Form $(x_{ij}, y_i), i = 1, \ldots, N$ betrachtet. Die Idee ist es, für alle Zweier-Kombinationen von Datenpunkten eine Verbindungsgerade zu konstruieren und deren Steigung zu betrachten. Für die Laufzeit des Verfahrens macht es einen Unterschied, ob ein lineares Modell ohne oder mit Intercept betrachtet wird. Nimmt man keinen Intercept in das Modell auf, ist einer dieser zwei Datenpunkte immer der Punkt $(0, 0)$. Damit müssen nur N Kombinationen betrachtet werden. Der Parameterbereich der j-ten Variablen ist dann gegeben durch $\Theta_j = \{y_i/x_{ij} \,|i = 1, \ldots, N\}$. Wird ein Intercept in das Modell mit aufgenommen, werden alle Zweiertupel von Datenpunkten ohne Betrachten der Reihenfolge kombiniert. Damit steigt die Anzahl an zu evaluierenden Kombinationen auf $\binom{N}{2}$. Man erhält dann jeweils einen Wert für den Intercept θ_0 und einen für die Steigung θ_1 der jeweiligen Geraden. Für ein Datenpaar $\{(x_1, y_1), (x_2, y_2)\}$ können diese beiden Werte kalkuliert werden durch

$$\theta_0 = y_1 - \frac{y_2 - y_1}{x_2 - x_1} x_1, \qquad \theta_1 = \frac{y_2 - y_1}{x_2 - x_1}.$$

Zur Veranschaulichung der beiden Fälle ist in Abbildung 4.6 und Abbildung 4.7 ein Beispiel angegeben. Die erste Grafik setzt ein Modell ohne Intercept voraus, sodass alle drei Geraden durch den Ursprung gehen. Bei der zweiten Grafik wurde auch ein Intercept mit in das Modell aufgenommen.

Dieser Ansatz ist insofern sinnvoll um θ_j für $j = 1, \ldots, p$ zu bestimmen, als dass die Informationen der Datenpunkte verwendet werden. Damit ist gemeint, dass es sehr naheliegend ist, dass die beste Anpassung des Modells an die Daten eine Gerade ist, die inmitten der Datenpunkte liegt. Legt man nun die Gerade durch die Punkte,

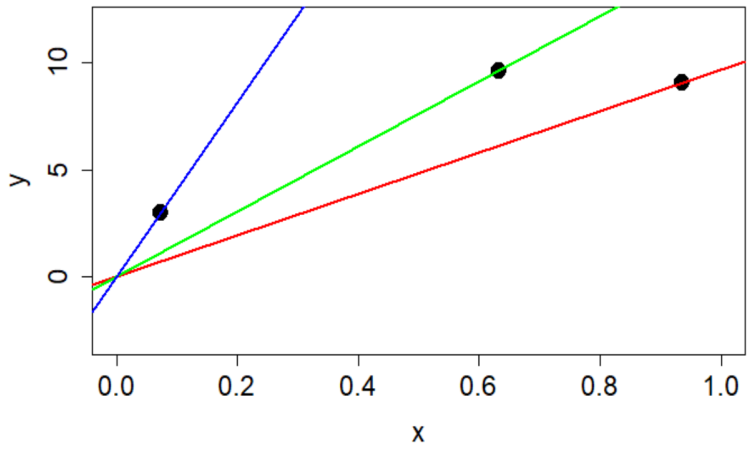

Abbildung 4.6: Beispiel für eine Wahl für den Wertebereich von θ. Hier gilt $\Theta = \{9.68, 15.24, 40.82\}$.

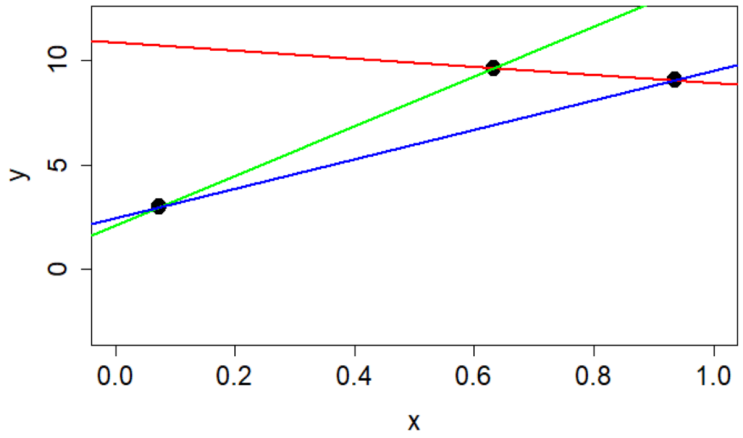

Abbildung 4.7: Beispiel für eine Wahl für den Wertebereich von θ. Hier gilt $\Theta = \{(10.83, -1.89), (2.11, 11.90), (2.46, 7.05)\}$.

wird diese Bedingung automatisch erfüllt. Wichtig zu beachten ist jedoch, dass die Bedingung

$$\mathbb{P}(R_i(\boldsymbol{\theta}) = 0) = 0, \qquad i = 1, \ldots, N$$

durch die Konstruktion der Geraden nicht mehr erfüllt ist. Dies ist jedoch nicht weiter tragisch, denn ein Residuum, welches den Wert 0 besitzt, spricht für eine gute Anpassung des zugehörigen Modells. Die zwei Residuen, für die $r_i(\boldsymbol{\theta}) = 0$ gilt, werden in allen Bewertungen nicht miteinbezogen.

Vereinfachte K-Vorzeichen-Tiefe

Eine weitere Stellschraube an der noch gedreht werden kann, ist die Wahl der K-Vorzeichen-Tiefe als Kriterium für die Güte der Anpassung der Modelle. Denn es gibt eine Modifikation der vollen K-Vorzeichen-Tiefe, die vereinfachte K-Tiefe wie sie schon in Abschnitt 3.3 vorgestellt wurde, die vielversprechend ist die beste Modellanpassung eindeutiger zu identifizieren.

Der Unterschied zwischen der vollen im Vergleich zu der vereinfachten K-Tiefe liegt in der Art und damit auch der Anzahl der K-Tupel, die für die Beurteilung der Anpassung des Modells an die Daten betrachtet wird. Welchen Unterschied dies macht wird im Folgenden herausgestellt. Dazu wird noch einmal der oben schon verwendete Datensatz mit Zielfunktion f_1 betrachtet. Es werden die exakt gleichen Datenpunkte genutzt, wie zuvor. Wünschenswert ist auch hier ein eindeutiges Hervorheben der ersten Variablen durch eine vergleichsweise große vereinfachte K-Tiefe. Wird das Analogon zu Abbildung 4.3 für die vereinfachte K-Vorzeichen-Tiefe konstruiert, zeigt sich ein ganz anderes Bild. Dieses ist in Abbildung 4.8 dargestellt. Die Kurven der

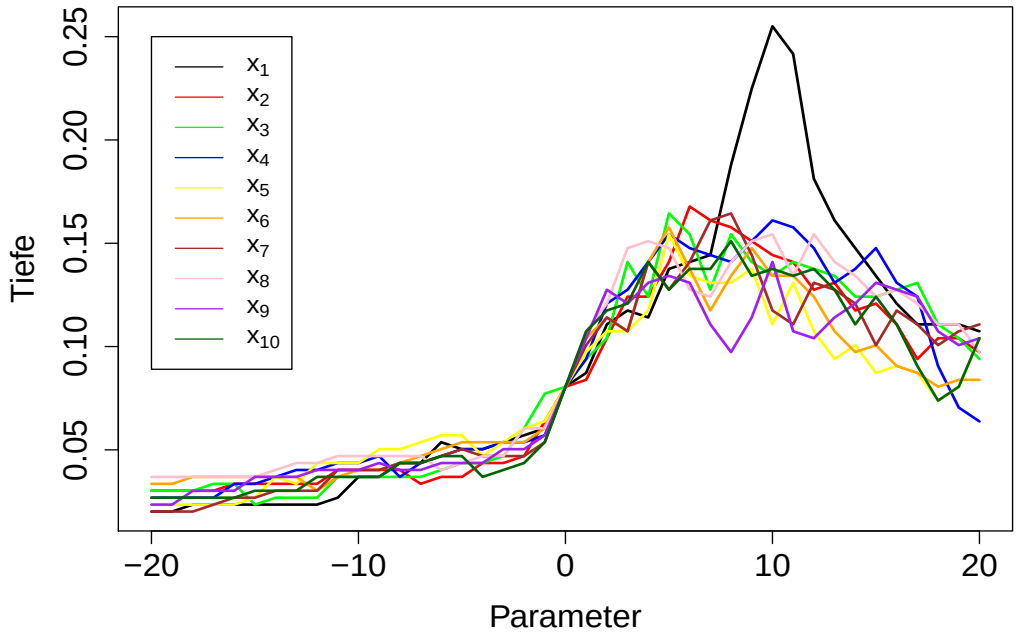

Abbildung 4.8: Linear interpolierte Kurven der vereinfachten K-Vorzeichen-Tiefe $d_K^S(y_1 - \theta x_{1j}, \ldots, y_N - \theta x_{Nj})$ der $j = 1, \ldots, 10$ Variablen für $\theta \in [-20, 20] \cap \mathbb{Z}$.

vereinfachten Tiefen der Variablen $x_2, \ldots, x_{10}$ zeigen einen deutlicheren Unterschied

zu der Kurve der ersten Variable. Alle zehn Kurven haben einen ähnlichen Verlauf. Das heißt, ab einem Parameterwert von $\theta = 0$ steigen alle Kurven an, und fallen dann spätestens ab einem Wert von $\theta = 10$ wieder ab. Ein klarer Unterschied ist in der Ausprägung des Hochpunkts der Kurven zu erkennen. Denn der An- und Abstieg nach dem Maximum der Kurve der Variablen x_1 ist wie auch in Abbildung 4.3 deutlich steiler. Hier hebt sich das Maximum klar von denen der anderen Variablen ab. In Tabelle 4.4 sind in der zweiten Spalte die maximalen Werte der vereinfachten K-Vorzeichen-Tiefe der jeweiligen Variablen in der ersten Spalte dargestellt. Das Er-

	Vereinfachte K-Vorzeichen-Tiefe	$\theta_{j,\max}$
x_1	**0.2550**	10
x_2	0.1677	6
x_3	0.1644	5
x_4	0.1610	10
x_5	0.1543	5
x_6	0.1577	5
x_7	0.1644	8
x_8	0.1543	10
x_9	0.1409	10
x_{10}	0.1510	8

Tabelle 4.4: In der zweiten Spalte sind die Werte der Maxima der vereinfachten K-Vorzeichen-Tiefe für die Kurven in Abbildung 4.8 für die jeweilige Variable in Spalte 1 aufgelistet. Die dritte Spalte gibt für jede Variable den Wert $\theta_{j,\max}$ des Parameters aus den gewählten Werten an, der die maximale vereinfachte K-Tiefe erzeugt.

gebnis ist klar. Der Wert der vereinfachten K-Vorzeichen-Tiefe der ersten Variablen ist fast doppelt so groß wie die Werte der Maxima der anderen Einflussvariablen. Die maximale vereinfachte K-Tiefe der ersten Variablen ist mit 0.2550 angegeben und die zweit größte vereinfachte K-Tiefe erreicht die zweite Variable mit einem Wert von nur 0.1677. Dies ist ein viel eindeutigeres Ergebnis als noch bei der vollen K-Vorzeichen-Tiefe. Das lässt sich gut an den Scatterplots in Abbildung 4.4 und Abbildung 4.5 erklären. Wie weiter oben schon angemerkt wurde, sammeln sich in Abbildung 4.4 die Datenpunkte näher um die Geraden, während die Punkte in Abbildung 4.5 einen breiten Schlauch in x_{max}-Richtung bilden. Das heißt, die Ausprägungen der Zielvariablen sind nicht korreliert mit den Werten der Variable $x_{\max}$. Da die volle K-Vorzeichen-Tiefe alle K-Tupel betrachtet, wird eine Anpassung eines Modells, welche zunächst viele positive und am Ende viele negative Vorzeichen erzeugt, mit einer hohen vollen Tiefe assoziiert. Denn dadurch entstehen, auch über weitere Entfernungen der Vor-

zeichen, K-Tupel, die alternierende Vorzeichen aufweisen und damit als positiv in die Bewertung mit einfließen. Dass eine solche Bewertung nicht immer sinnvoll ist, wird im Beispiel aus Abbildung 4.5 deutlich. Da bei der vereinfachten K-Vorzeichen-Tiefe nur direkt aufeinander folgende K-Tupel gewertet werden, fällt die Bewertung der vereinfachten Tiefe in einem solchen Fall schlechter aus.

Einfaches lineares Modell ohne Intercept

Diese beiden gerade vorgestellten Ansätze sollen in einem Boosting-Verfahren kombiniert und evaluiert werden. Zunächst wird dazu ein einfaches lineares Modell ohne Intercept beziehungsweise ein multiples lineares Modell mit nur einer Variablen betrachtet, die einen echten Effekt auf die Zielvariable hat. Das heißt, aus p Einflussvariablen soll die Variable herausgesucht werden, welche einen echten Einfluss auf die Zielvariable hat. Alle anderen Variablen erhalten einen wahren Parameterwert von 0. Die Zielfunktion wird daher wie in Gleichung (4.1) angenommen, wobei $\theta_0 = \theta_1 = \ldots = \theta_{j-1} = \theta_{j+1} = \ldots = \theta_p = 0$ und $\theta_j \neq 0$ gilt. Gesucht wird $j \in \{1, \ldots, p\}$ mit dem zugehörigen Wert θ_j. In Algorithmus 5 ist ein Pseudocode für die Umsetzung aufgeführt. Für diesen Ansatz gibt es sowohl die Möglichkeit die

Algorithmus 5 K-Vorzeichen-Boosting (Einfaches lineares Modell ohne Intercept)

Eingabe:

Daten $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, volle K-Tiefe V (Boolesch), Länge der Tupel K
for $j = 1 \rightarrow p$ **do**:
    $\Theta_j = \{y_i / x_{ij} \,|i = 1, \ldots, N\}$
    if $V ==$ TRUE **then**
        $\hat{\theta}_j \in \underset{\theta \in \Theta_j}{\operatorname{argmax}}\, d_K^S(y_1 - \theta x_{1j}, \ldots, y_N - \theta x_{Nj})$
    else
        $\hat{\theta}_j \in \underset{\theta \in \Theta_j}{\operatorname{argmax}}\, d_K(y_1 - \theta x_{1j}, \ldots, y_N - \theta x_{Nj})$
    end if
end for
if $V ==$ TRUE **then**
    Bestimme $\hat{j} \in \underset{j \in \{1,\ldots,p\}}{\operatorname{argmax}}\, d_K^S(y_1 - \hat{\theta}_j x_{1j}, \ldots, y_N - \hat{\theta}_j x_{Nj})$
else
    Bestimme $\hat{j} \in \underset{j \in \{1,\ldots,p\}}{\operatorname{argmax}}\, d_K(y_1 - \hat{\theta}_j x_{1j}, \ldots, y_N - \hat{\theta}_j x_{Nj})$
end if
Ausgabe: $\hat{\theta}_{\hat{j}}$, $d_K^S(y_1 - \hat{\theta}_{\hat{j}} x_{1\hat{j}}, \ldots, y_N - \hat{\theta}_{\hat{j}} x_{N\hat{j}})$
(volle K-Tiefe: $d_K(y_1 - \hat{\theta}_{\hat{j}} x_{1\hat{j}}, \ldots, y_N - \hat{\theta}_{\hat{j}} x_{N\hat{j}})$)

vereinfachte als auch die volle K-Tiefe als Kriterium für die Auswahl des „besten“

Parameters zu wählen. So kann mittels einer Simulationsstudie evaluiert werden, welches Kriterium zu besseren Ergebnissen führt. Da kein Intercept in das Modell mit aufgenommen wird, besteht der Parameterbereich Θ_j für jede Variable mit Index $j = 1, \ldots, p$ aus den Steigungen der Geraden, die für $i = 1, \ldots, N$ durch $\theta_j = \frac{y_i}{x_{ij}}$ gegeben sind. Danach wird für jede Variable der Parameter $\theta_j \in \Theta_j$ gesucht, der die (vereinfachte) K-Tiefe der Residuen maximiert und anschließend die Variable angegeben, dessen Kombination aus Parameter und Variable für die entsprechenden Residuen die maximale (vereinfachte) K-Tiefe erzeugt. Dabei werden einige Einstellungen variiert, um verschiedene Szenarien nachzubauen. Die Zielfunktion ist wieder durch $f_1(\mathbf{x}) = 10\mathbf{x}_1$ gegeben. Die Werte der Zielvariablen werden simuliert durch

$$y_i = f_1(\mathbf{x}_i) + C\varepsilon_i, \qquad i = 1, \ldots, N.$$

mit $p = 10$ Einflussvariablen, die alle gleichverteilt auf (0,1) sind. Es wird eine Tupellänge von $K = 3$ gewählt. Weitere Einstellungen, welche für die Simulation variiert werden sind

- die Verteilung der Fehler $\mathcal{E} \in \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4\}$
- die Anzahl an Beobachtungen $N \in \{30, 300\}$
- ob die vereinfachte K-Tiefe verwendet wird $V \in \{\text{TRUE}, \text{FALSE}\}$.

Aufgrund der langen Laufzeit wird auch mit 30 Beobachtungen simuliert, um alle Fehlerverteilungen einmal evaluieren zu können. Für $N = 300$ werden nur die Fehlerverteilungen $\mathcal{D}_1$ und $\mathcal{D}_2$ verglichen, um das Verhalten des Verfahrens sowohl auf kontaminierten als auch unkontaminierten Datensätzen zu beobachten. Mit jeder Kombination an Einstellungen werden jeweils 100 unabhängige Datensimulationen durchgeführt und danach alle Werte der (vereinfachten) K-Vorzeichen-Tiefe für jede Variable über alle $S = 100$ Simulationen gemittelt. Von den jeweils zugehörigen Parameterwerten wird der Mittelwert bestimmt. Die Ergebnisse dieser Simulationen sind für die vereinfachte K-Vorzeichen-Tiefe in Tabelle 4.5 dargestellt. Was auffällt ist, dass die erste Variable in allen hier betrachteten Fällen die maximale mittlere K-Vorzeichen-Tiefe erzeugt. Hinsichtlich der Tatsache, dass die erste Variable die einzige mit echtem Einfluss auf die Zielvariable ist, ist dieses Ergebnis wünschenswert. Weiter ist zu erkennen, dass auch der wahre Parameter θ, der die Größe des Einflusses bestimmt, bei allen Einstellungskombinationen gut approximiert wird. Die

# Beob.	Fehlervert.	größte Tiefe	max. Tiefe-Index	2. größte Tiefe
30	$\mathcal{D}_1$	0.3678 (9.9652)	1	0.2675 (12.4383)
30	$\mathcal{D}_2$	0.3682 (9.9954)	1	0.3050 (13.9622)
30	$\mathcal{D}_3$	0.3782 (10.3974)	1	0.3035 (11.7743)
30	$\mathcal{D}_4$	0.3828 (11.3274)	1	0.3089 (10.5040)
300	$\mathcal{D}_1$	0.2781 (10.0200)	1	0.1228 (9.7883)
300	$\mathcal{D}_2$	0.2763 (9.9896)	1	0.1772 (8.2575)

Tabelle 4.5: Kennzahlen einer Simulationsstudie über 100 unabhängige Simulationen mit Verwendung der vereinfachten K-Vorzeichen-Tiefe. Die verschiedenen Einstellungen werden in den ersten beiden Spalten angegeben. In der ersten steht die jeweilige Anzahl der verwendeten Beobachtungen und in der zweiten die Wahl der Fehlerverteilung. Die letzten drei Spalten geben die Ergebnisse der Simulationsstudie an. Es wird die jeweilige maximale mittlere Tiefe über die $p = 10$ Einflussvariablen in der dritten Spalte angegeben. Der Mittelwert der zur Variable zugehörigen Parameter steht in Klammern dahinter. Die zweitgrößte mittlere Tiefe wird in der letzten Spalte der Tabelle festgehalten mit zugehörigem Mittelwert der Parameter.

größte Abweichung im Mittelwert besteht bei der Fehlerverteilung $\mathcal{D}_4$ mit einer Abweichung des Mittels zum wahren Wert von ungefähr 1.3. Die Parametermittelwerte bei $N = 300$ Beobachtungen liegen im Vergleich zu denen bei 30 Beobachtungen noch näher an den wahren Werten. Das kann damit begründet werden, dass die Steigungen der Datenpunkte zur Parameterwahl verwendet werden. Stehen mehr Datenpunkte zur Verfügung, ist es wahrscheinlicher eine bessere Approximation an den wahren Wert der Parameter zu erhalten. Schaut man zum Vergleich die Mittelwerte der Parameter für die Variablen an, die die zweitgrößte Tiefe erzeugen, sind die Approximationen weiter vom wahren Wert entfernt. Dort liegt die größte Abweichung bei einem Wert von ungefähr 4. Vergleicht man die Werte der vereinfachten K-Vorzeichen-Tiefe zwischen der dritten und letzten Spalte in Tabelle 4.5, dann bestätigt sich die Beobachtung, die sich nach Betrachten von Abbildung 4.8 schon aufgetan hat. Die Abstände der Mittelwerte der Tiefen sind hier deutlich, denn die Werte liegen immer mindestens 0.07 auseinander. Bei $N = 300$ Beobachtungen liegt der kleinste Abstand bei 1. Wie schon bei der Untersuchung des ersten Ansatzes für das K-Vorzeichen-Boosting festgestellt wurde, ist diese Eigenschaft bei der vollen Tiefe nicht so ausgeprägt. Die Ergebnisse der Simulationsstudie wie sie in Tabelle 4.5 für die vereinfachte K-Vorzeichen-Tiefe festgehalten sind, sind in Tabelle 4.6 für die

volle Tiefe aufgeführt. Was zunächst auch bei den Simulationen mit der vollen Tiefe

# Beob.	Fehlervert.	größte Tiefe	max. Tiefe-Index	2. größte Tiefe
30	$\mathcal{D}_1$	0.2645 (9.9476)	1	0.2595 (11.1229)
30	$\mathcal{D}_2$	0.2654 (9.8712)	1	0.2607 (10.8161)
30	$\mathcal{D}_3$	0.2666 (10.0673)	1	0.2599 (10.4409)
30	$\mathcal{D}_4$	0.2663 (10.0870)	1	0.2607 (11.0756)
300	$\mathcal{D}_1$	0.2513 (10.0051)	1	0.2493 (9.9534)
300	$\mathcal{D}_2$	0.2510 (10.0000)	1	0.2498 (9.9620)

Tabelle 4.6: Kennzahlen einer Simulationsstudie über 100 unabhängige Simulationen mit Verwendung der vollen K-Vorzeichen-Tiefe. Die verschiedenen Einstellungen werden in den ersten beiden Spalten angegeben. In der ersten steht die jeweilige Anzahl der verwendeten Beobachtungen und in der zweiten die Wahl der Fehlerverteilung. Die letzten drei Spalten geben die Ergebnisse der Simulationsstudie an. Es werden die maximale mittlere Tiefe über die $p = 10$ Einflussvariablen in der dritten Spalte angegeben. Der Mittelwert der zur Variable zugehörigen Parameter steht in Klammern dahinter. Die zweitgrößte mittlere Tiefe wird in der letzten Spalte der Tabelle festgehalten mit zugehörigem Mittelwert der Parameter.

auffällt, ist dass auch hier im Mittel die erste Variable die größte volle K-Vorzeichen-Tiefe erzeugt. Das war in der Untersuchung des ersten Ansatzes noch nicht so. Dort wurde jedoch auch nur eine Simulation angeschaut. Auch schon auffällig war, dass die Werte der Tiefen alle sehr nah beieinander liegen. Diese Beobachtung bestätigt sich auch in dieser Simulationsstudie. Denn der Abstand zur zweitgrößten mittleren Tiefe ist jeweils uneindeutiger als noch in Tabelle 4.5. Hier liegen die Werte maximal 0.007 auseinander, sodass sich die Abstände ungefähr um einen Faktor 10 unterscheiden. Bei $N = 30$ und $\mathcal{D}_2$ liegt der Abstand der Tiefen sogar bei einem Wert kleiner als 0.005. Das Verhalten der Mittelwerte der Parameter zeigt ein ähnliches Verhalten wie im Fall der vereinfachten K-Vorzeichen-Tiefe. Die Mittelwerte der Parameter liegen durchweg nah am wahren Wert $\theta_1 = 10$, für $N = 300$ näher als für $N = 30$. Die maximale Abweichung des Mittelwerts der Parameter vom wahren Wert wird bei der Studentschen t-Verteilung mit einem Abstand von 0.13 erreicht. Dieser Wert fällt im Vergleich zu dem maximalen Abstand von 1.3 bei der vereinfachten K-Vorzeichen-Tiefe klein aus.
Was man zu Tabelle 4.6 noch sagen kann ist, dass obwohl die zweitgrößten mittleren vollen Tiefe-Werte nur wenig schlechter sind als die größten mittleren vollen Tiefen,

die Mittelwerte der Parameter deutlich weiter vom wahren Wert entfernt sind. Denn hier liegt der maximale Abstand schon bei ca. 1.1.

Multiples lineares Modell ohne Intercept

Nachdem der wahre Parameter im einfachen linearen Modell schon gut approximiert wurde, soll das K-Vorzeichen-Boosting auch auf multiple lineare Modelle ohne Intercept erweitert werden. Die Zielfunktion wird also wie in Gleichung (4.1) angenommen, wobei $\theta_0 = 0$, und $\theta_1, \ldots, \theta_p \in \mathbb{R}$. Gesucht werden $\theta_1, \ldots, \theta_p$. Im Vergleich zum Vorgehen bei dem einfachen linearen Modell, werden in $M \geq 1$ Boosting-Runden die p Parameter einzeln an ihre wahren Werte herangeführt. Dafür ist ein Initialisierungsschritt nötig. Das heißt, zu Beginn des Verfahrens wird, wie es in Algorithmus 6 mit Hilfe von Pseudocode aufgeschrieben ist, $(\theta_1, \ldots, \theta_p)^T = \mathbf{0}$ gesetzt. Die Residuen entsprechen anfangs den Werten der Einflussvariable y. In allen M Boosting-Runden

Algorithmus 6 K-Vorzeichen-Boosting (Multiples lineares Modell ohne Intercept)

Eingabe:

Daten $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, Länge der Tupel K, Anzahl Iterationen M, Toleranz tol, volle K-Tiefe V (Boolesch)

Initialisierung:

$\boldsymbol{\theta}_0 = (\theta_1, \ldots, \theta_p)^T = \mathbf{0}, p$ Anzahl Einflussvariablen,
$(r_1^{[0]}, \ldots, r_N^{[0]})^T = \mathbf{r}^{[0]} = \mathbf{y} = (y_1, \ldots, y_N)^T$

for $m = 1 \rightarrow M$ **do**:

  for $j = 1 \rightarrow p$ **do**:

    $\Theta_j = \{r_i^{[m-1]} / x_{ij} \,|i = 1, \ldots, N\}$

    $\hat{\theta}_j \in \underset{\theta \in \Theta_j}{\operatorname{argmax}}\, d_K^{(S)}(r_1^{[m-1]} - \theta x_{1j}, \ldots, r_N^{[m-1]} - \theta x_{Nj})$

  end for

  Bestimme $\hat{j} \in \underset{j \in \{1,\ldots,p\}}{\operatorname{argmax}}\, d_K^{(S)}(r_1^{[m-1]} - \hat{\theta}_j x_{1j}, \ldots, r_N^{[m-1]} - \hat{\theta}_j x_{Nj})$

  Update: $\mathbf{r}^{[m]} = \mathbf{r}^{[m-1]} - \hat{\theta}_{\hat{j}} \boldsymbol{x}_{\hat{j}}$ und $\theta_{\hat{j}} = \theta_{\hat{j}} + \hat{\theta}_{\hat{j}}$

  if $|d_K^S(r_1^{[m]}, \ldots, r_N^{[m]}) - d_K^S(r_1^{[m-1]}, \ldots, r_N^{[m-1]})| < tol$ **then**

    break

  end if

end for

Ausgabe: $\boldsymbol{\theta_0}$

wird für jede der p Variablen zunächst der Parameterbereich Θ_j bestimmt, aus welchem die Werte für θ_j gewählt werden. Dieser besteht analog wie in Algorithmus 5 aus den Steigungen der Geraden durch die Datenpunkte. Nach Bestimmen der Kombination aus der Variable $\hat{j}$ mit zugehörigem Parameter $\hat{\theta}_{\hat{j}}$, die die maximale (vereinfachte) Tiefe erzeugen, folgt ein Update der Residuen und Parameter. Von den

Werten der Residuen aus der vorangegangen Iteration wird das Produkt aus den Ausprägungen der gewählten Variablen und dem zugehörigen geschätzten Parameterwert subtrahiert. Die bereits erklärte Variabilität in den Daten wird herausgerechnet. In den folgenden Iterationen kann dann darauf aufbauend die verbleibende Variabilität erklärt werden kann. So werden die Residuen schrittweise verkleinert und die Parameter an ihren wahren Wert herangeführt. Um eine Überanpassung und unnötig lange Laufzeit des Modells zu verhindern, gibt es ein Abbruchkriterium. Verbessert sich die (vereinfachte) K-Vorzeichen-Tiefe nicht mehr hinreichend von einer Iteration zur nächsten, das heißt, ist der Abstand zwischen der neuen und der alten (vereinfachten) Tiefe kleiner als eine gesetzte Toleranzgrenze, dann wird das Verfahren frühzeitig abgebrochen und der aktuell geschätzte Parametervektor ausgegeben.
Die Performance des K-Vorzeichen-Boostings für den Fall des multiplen linearen Modells ohne Intercept soll ebenfalls mit Hilfe einer Simulationsstudie evaluiert werden. Mit der Zielfunktion f_2 und den zwei Fehlerverteilungen $\mathcal{D}_1$ und $\mathcal{D}_2$ werden in 20 Simulationen unabhängige Datensätze konstruiert. Bei der Funktion f_2 haben lediglich die ersten beiden Variablen einen echten Einfluss auf die Zielvariable. Die wahren Werte der Parameter θ_1 und θ_2 sind 3 und 5. Diese gilt es möglichst gut zu approximieren. Dadurch, dass die Einflussvariablen wie bisher alle gleichverteilt auf (0,1) gewählt sind, kann herausgelesen werden, dass die zweite Variable mehr Einfluss hat als die erste. Es gibt zwei unterschiedliche Anzahlen an möglichen Einflussvariablen, $p = 2$ und $p = 10$. In Tabelle 4.7 sind die Ergebnisse der Studie für $p = 10$ Einflussvariablen aufgezeigt für $N = 300$ und $N = 30$ Beobachtungen und sowohl die vereinfachte als auch die volle Tiefe wurden verwendet. Lediglich die Fälle mit $N = 300$ Beobachtungen und der vereinfachten Tiefe spiegeln den größten Einfluss der ersten und zweiten Variablen durch das Mittel der geschätzten Parameter wieder. Bei den unkontaminierten Datensätzen werden die Parameter im Mittel durch $\hat{\theta}_1 \approx 1.27$ und $\hat{\theta}_2 \approx 8.70$ geschätzt. Der größere Einfluss der zweiten Variable wird also deutlich, jedoch sind die Schätzer an sich noch weit von den wahren Werten entfernt. Alle restlichen Schätzer der Parameter liegen im Mittel nahe bei 0, sodass zumindest die Gewichtung der Einflüsse bei den richtigen Variablen liegt. Ein ähnliches Verhalten zeigt sich auch bei den kontaminierten Datensätzen mit der Fehlerverteilung $\mathcal{D}_2$. Hier ist das Herausarbeiten der ersten beiden Variablen nicht ganz so gelungen, denn der Parameter der ersten Variablen wird im Mittel mit $\hat{\theta}_1 \approx 1.04$ geschätzt, die vierte Variable erhält jedoch im Mittel auch eine Gewichtung von $\hat{\theta}_4 \approx 0.72$, sodass hier keine klare Abgrenzung stattfindet. Bei den Kombinationen dieser beiden gerade ge-

	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\theta}_4$	$\hat{\theta}_5$	$\hat{\theta}_6$	$\hat{\theta}_7$	$\hat{\theta}_8$	$\hat{\theta}_9$	$\hat{\theta}_{10}$
$\mathcal{D}_1$, $N = 300$ $V = \text{TRUE}$	**1.27**	**8.70**	0.08	-0.11	0.04	0.03	0.01	0.00	-0.04	0.05
$\mathcal{D}_2$, $N = 300$ $V = \text{TRUE}$	**1.04**	**8.20**	0.26	0.72	-0.06	-0.04	-0.14	-0.11	-0.06	0.05
$\mathcal{D}_1$, $N = 30$ $V = \text{TRUE}$	**2.80**	**1.95**	-0.03	0.37	-0.86	0.54	0.00	1.81	3.19	0.95
$\mathcal{D}_2$, $N = 30$ $V = \text{TRUE}$	**0.75**	**0.70**	1.41	1.40	2.03	-0.06	0.64	0.80	0.49	2.16
$\mathcal{D}_1$, $N = 300$ $V = \text{FALSE}$	**0.55**	**1.98**	0.01	0.46	2.64	1.50	1.05	0.54	1.52	0.04
$\mathcal{D}_2$, $N = 300$ $V = \text{FALSE}$	**0.49**	**2.53**	0.95	-0.02	0.47	1.56	1.59	0.93	0.48	1.04

Tabelle 4.7: Mittelwerte der geschätzten Modellparameter über 20 unabhängige Simulationen hinweg. Verwendet wurden: Zielfunktion $f_2(\mathbf{x}) = 3\mathbf{x}_1 + 7\mathbf{x}_2$ mit insgesamt $p = 10$ Variablen, jeweils $M = 20$ Boosting-Runden, eine Toleranzgrenze von $tol = 10^{-3}$. Die Daten wurden nach dem NN-Verfahren sortiert und es wurde eine Tupellänge von $K = 3$ evaluiert.

nannten Fehlerverteilung $\mathcal{D}_1$ und $\mathcal{D}_2$ mit jeweils nur $N = 30$ Beobachtungen bei der vereinfachten Tiefe und $N = 300$ Beobachtungen bei der vollen K-Vorzeichen-Tiefe, kann dieses Verhalten nicht beobachtet werden. Die Schätzungen der Parameter wirken willkürlich und sind weit weg von den wahren Werten.
Werden nur $p = 2$ Variablen betrachtet, also lediglich die Variablen, die unter Verwendung der Zielfunktion f_2 auch einen echten Einfluss haben, zeigt sich ein ähnliches Bild, wie schon bei der Studie zuvor für $N = 300$ Beobachtungen und der vereinfachten Tiefe. Die Ergebnisse dieser Simulationsstudie sind in Tabelle 4.8 aufgezeigt. Die Mittelwerte der Schätzungen der ersten und zweiten Variablen spiegeln in allen Fällen den größeren Einfluss der zweiten Variablen wieder. Lediglich die Kombination aus $N = 30$ Beobachtungen, $\mathcal{D}_2$ und der vereinfachten Tiefe als Maß fällt in dieser Hinsicht aus der Reihe. Das Betrachten von ausschließlich zwei Variablen, die beide einen echten Einfluss auf die Zielvariable haben, verbessert also nur begrenzt die Schätzung der Parameter.

	$\mathcal{D}_1$, $N = 300$ $V = \text{TRUE}$	$\mathcal{D}_2$, $N = 300$ $V = \text{TRUE}$	$\mathcal{D}_1$, $N = 30$ $V = \text{TRUE}$	$\mathcal{D}_2$, $N = 30$ $V = \text{TRUE}$	$\mathcal{D}_1$, $N = 300$ $V = \text{FALSE}$	$\mathcal{D}_2$, $N = 300$ $V = \text{FALSE}$
$\hat{\theta}_1$	**1.61**	**1.57**	**2.80**	**5.17**	**3.58**	**1.28**
$\hat{\theta}_2$	**8.40**	**8.28**	**7.29**	**4.82**	**6.10**	**8.74**

Tabelle 4.8: Mittelwerte der geschätzten Modellparameter über 20 unabhängige Simulationen hinweg. Verwendet wurden: Zielfunktion $f_2(\mathbf{x}) = 3\mathbf{x}_1 + 7\mathbf{x}_2$ mit insgesamt $p = 2$ Variablen, jeweils $M = 20$ Boosting-Runden, eine Toleranzgrenze von $tol = 10^{-3}$. Die Daten wurden nach dem NN-Verfahren sortiert und es wurde eine Tupellänge von $K = 3$ evaluiert.

Multiples lineares Modell mit Intercept

Um das K-Vorzeichen-Boosting auch auf multiple lineare Modelle mit Intercept anwenden zu können, müssen in Algorithmus 6 ein paar Kleinigkeiten geändert werden. Dazu gehört vor allem das Erweitern des Parametervektors um einen Intercept, da $p + 1$ Parameter geschätzt werden. Die grundlegende Idee bei Hinzunahme eines Intercept ist es, in jedem Boosting-Schritt nun für jede Variable die Geraden zwischen allen $\binom{N}{2}$ Zweierkombinationen an Datenpunkten betrachtet, sodass für jede Variable zwei Parameter gesucht werden, die die maximale (vereinfachte) Tiefe erzeugen, jeweils ein Intercept und die zugehörige Steigung der Geraden. Dadurch, dass für jede Variable ein Intercept geschätzt wird, wird dieser in jeder Boosting-Runde geupdated. Der Parameterbereich enthält jetzt Zweiertupel. Das erste Element der Tupel gibt jeweils den Intercept der Verbindungsgeraden an und das zweite Element die Steigung wie es in Zeile 5 in Algorithmus 7 angegeben ist. Ist die Kombination $\hat{\theta}_{0\hat{j}}, \hat{\theta}_{\hat{j}}$ der m-ten Boosting-Runde gefunden, werden die Parameter und Residuen geupdated. Die Residuen werden um die Linearkombination $\hat{\theta}_{0\hat{j}} + \hat{\theta}_{\hat{j}}\mathbf{x}_{\hat{j}}$ reduziert, da die dadurch entstehende Variabilität schon erklärt wird. Das Update des Parameters passiert analog wie in Algorithmus 6 mit dem einzigen Unterschied, dass zusätzlich der Intercept in jeder Boosting-Runde geupdated wird. Das heißt, der neu geschätzte Intercept der jeweiligen Variablen wird auf den schon existierenden Wert für den Intercept des gesamten Modells aufaddiert.

Auch das K-Vorzeichen-Boosting für das multiple lineare Modell mit Intercept wird mit Hilfe einer Simulationsstudie evaluiert. Dazu werden die Zielfunktionen $f_4(\mathbf{x}) = 5 + 3\mathbf{x}_1 + 7\mathbf{x}_2$ und $f_5(\mathbf{x}) = 6 + 2\mathbf{x}_1 - 7\mathbf{x}_3 + 4\mathbf{x}_6 - \mathbf{x}_7$ genutzt. Das Verfahren hat eine sehr lange Laufzeit, da im Gegensatz zum K-Vorzeichen-Boosting nicht in jeder Boosting-Runde N-mal die (vereinfachte) K-Vorzeichen-Tiefe berechnet werden muss, sondern $\binom{N}{2}$-mal. Diese Gegebenheit verlangsamt die Laufzeit deutlich. Aus

Algorithmus 7 K-Vorzeichen-Boosting (Multiples lineares Modell mit Intercept)

Eingabe:

Daten $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, Länge der Tupel K, Anzahl Iterationen M, Toleranz tol, volle K-Tiefe V (Boolesch)

Initialisierung:

$\boldsymbol{\theta}_0 = (\theta_0, \ldots, \theta_p)^T = \mathbf{0}, p$ Anzahl Einflussvariablen,
$(r_1^{[0]}, \ldots, r_N^{[0]})^T = \mathbf{r}^{[0]} = \mathbf{y} = (y_1, \ldots, y_N)^T$

for $m = 1 \rightarrow M$ **do**:

  for $j = 1 \rightarrow p$ **do**:

$$\Theta_j = \{(r_{i_1}^{[m-1]} - \frac{r_{i_2}^{[m-1]} - r_{i_1}^{[m-1]}}{x_{i_2} - x_{i_1}} x_{i_1}, \frac{r_{i_2}^{[m-1]} - r_{i_1}^{[m-1]}}{x_{i_2} - x_{i_1}}) \mid i_1 = 1, \ldots, N-1, i_2 = i_1, \ldots, N\}$$

$$\hat{\theta}_{0j}, \hat{\theta}_j \in \underset{(\theta_0, \theta_j) \in \Theta_j}{\operatorname{argmax}} \; d_K^{(S)}(r_1^{[m-1]} - (\theta_0 + \theta_j x_{1j}), \ldots, r_N^{[m-1]} - (\theta_0 + \theta_j x_{Nj}))$$

  end for

  Bestimme $\hat{j} \in \underset{j \in \{1, \ldots, p\}}{\operatorname{argmax}} \; d_K^{(S)}(r_1^{[m-1]} - (\hat{\theta}_{0j} + \hat{\theta}_j x_{1j}), \ldots, r_N^{[m-1]} - (\hat{\theta}_{0j} + \hat{\theta}_j x_{Nj}))$

  Update: $\mathbf{r}^{[m]} = \mathbf{r}^{[m-1]} - (\hat{\theta}_{0\hat{j}} + \hat{\theta}_{\hat{j}} \boldsymbol{x}_{\hat{j}})$ und $\theta_{\hat{j}} = \theta_{\hat{j}} + \hat{\theta}_{\hat{j}}$, $\theta_0 = \theta_0 + \hat{\theta}_{0\hat{j}}$

  if $|d_K^S(r_1^{[m]}, \ldots, r_N^{[m]}) - d_K^S(r_1^{[m-1]}, \ldots, r_N^{[m-1]})| < tol$ **then**

    break

  end if

end for

Ausgabe: $\boldsymbol{\theta_0}$

diesem Grund werden in der Evaluation des K-Vorzeichen-Boostings für das multiple lineare Modell mit Intercept jeweils nur 100 Datenpunkte betrachtet. Die Ergebnisse einer Simulationsstudie für 20 unabhängige Datensätze mit $p = 10$ Einflussvariablen ist in Tabelle 4.9 festgehalten. Die besten Approximationen der wahren Parameter im Mittel werden auch hier mit den unkontaminierten Datensätzen erreicht. In Kombination mit der Zielfunktion f_4 erreichen die geschätzten Parameter im Mittel Werte von $(\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3) \approx (5.98, 1.40, 6.45)$. Diese Approximation gibt vor allem die Größe des Einflusses der Variablen und den Intercept richtig wieder. Die Mittel der Schätzungen der übrigen Parameter der Variablen sind auch alle vergleichsweise klein. Für die kontaminierten Datensätze sind die Schätzungen der Parameter $\hat{\theta}_0$ und $\hat{\theta}_2$ am größten, das heißt, hier werden die Variablen mit echtem Einfluss teilweise richtig gefiltert. θ_1 wird im Mittel mit nur 0.63 geschätzt und liegt damit unter den Schätzungen der Parameter für die vierte beziehungsweise neunte Variable. Ein ähnliches Bild zeigt sich auch für die Zielfunktion f_5, bei welcher mehr Einflussvariablen einen echten Einfluss auf die Zielvariable haben. Während die Mittelwerte der geschätzten Parameter über die 20 Simulationen bei den unkontaminierten Daten noch recht gute

	$\hat{\theta}_0$	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\theta}_4$	$\hat{\theta}_5$	$\hat{\theta}_6$	$\hat{\theta}_7$	$\hat{\theta}_8$	$\hat{\theta}_9$	$\hat{\theta}_{10}$
$\mathcal{D}_1$, f_4	**5.98**	**1.40**	**6.45**	-0.20	-0.09	-0.04	0.26	0.10	-0.20	-0.27	0.18
$\mathcal{D}_2$, f_4	**6.81**	**0.63**	**3.51**	-0.12	1.08	0.16	0.00	0.40	0.23	0.70	-0.29
$\mathcal{D}_1$, f_5	**7.01**	**0.15**	0.64	**-6.44**	0.22	0.02	**0.38**	**-0.08**	0.28	0.389	0.86
$\mathcal{D}_2$, f_5	**6.53**	**0.30**	0.49	**-4.08**	0.11	0.22	**0.27**	**-0.18**	0.05	-0.05	-0.55
$\mathcal{D}_3$, f_5	**7.07**	**1.93**	0.26	**-1.76**	-1.73	-0.57	**-0.11**	**-0.77**	-1.06	-0.03	-0.36
$\mathcal{D}_4$, f_5	**8.92**	**-0.32**	-2.00	**-2.55**	1.97	-0.69	**-8.22**	**0.11**	0.37	-0.15	2.58

Tabelle 4.9: Mittelwerte der geschätzten Modellparameter über 20 unabhängige Simulationen hinweg. Verwendet wurden: Zielfunktionen f_4 und f_5 mit insgesamt $p = 10$ Variablen, jeweils $M = 20$ Boosting-Runden, eine Toleranzgrenze von $tol = 10^{-3}$. Die Daten wurden nach dem NN-Verfahren sortiert und es wurde eine Tupellänge von $K = 3$ evaluiert.

Approximationen an die wahren Werte liefern, sind die Schätzungen für die weiteren Fehlerverteilungen $\mathcal{D}_2$,$\mathcal{D}_3$ und $\mathcal{D}_4$ wieder recht willkürlich und zeigen nur selten die Tendenzen in die Richtung der wahren Werte.
Anders sieht es in der Simulationsstudie aus, in welcher für f_4 lediglich $p = 2$ Einflussvariablen betrachtet werden. Die Ergebnisse der Mittelwerte der Parameter über 20 unabhängige Simulationen sind in Tabelle 4.10 notiert. Für die Fehlerverteilun-

	$\mathcal{D}_1$, $V = \text{TRUE}$	$\mathcal{D}_2$, $V = \text{TRUE}$	$\mathcal{D}_1$, $V = \text{FALSE}$	$\mathcal{D}_2$, $V = \text{FALSE}$
$\hat{\theta}_0$	**5.07**	**5.29**	**5.80**	**5.58**
$\hat{\theta}_1$	**2.60**	**2.16**	**2.80**	**2.60**
$\hat{\theta}_2$	**7.10**	**7.16**	**6.31**	**6.10**

Tabelle 4.10: Mittelwerte der geschätzten Modellparameter über 20 unabhängige Simulationen hinweg. Verwendet wurden: Zielfunktion $f_4(\mathbf{x}) = 5 + 3\mathbf{x}_1 + 7\mathbf{x}_2$ mit insgesamt $p = 2$ Variablen, jeweils $M = 20$ Boosting-Runden, eine Toleranzgrenze von $tol = 10^{-3}$. Die Daten wurden nach dem NN-Verfahren sortiert und es wurde eine Tupellänge von $K = 3$ evaluiert.

gen $\mathcal{D}_1$ und $\mathcal{D}_2$, in Kombination mit dem Maß der vereinfachten beziehungsweise vollen K-Vorzeichen-Tiefe, werden die Parameter des Modells im Mittel durchweg gut approximiert. Die Schätzungen des Intercepts $\hat{\theta}_0$ liegen im Mittel zwischen 5.07 und 5.80. Diese Werte sind im Vergleich zu den bis hierhin betrachteten Approxi-

mationen nah am wahren Wert von 5. Doch es wird nicht nur der Intercept gut approximiert. Denn auch für die zwei weiteren Modellparameter $\theta_1 = 3$ und $\theta_2 = 7$ liegen die mittleren Schätzungen mit Werten zwischen 2.16 und 2.8 beziehungsweise 6.1 und 7.16 nah an den wahren Werten der Parameter.

5 Vergleich der Verfahren in einer Simulationsstudie

Zum Abschluss soll das im Rahmen dieser Arbeit implementierte K-Vorzeichen-Boosting für multiple lineare Modelle mit Intercept, wie es in Algorithmus 7 aufgeschrieben ist, noch einmal mit dem RRBoost-Verfahren und dem Gradienten-Boosting innerhalb einer Simulationsstudie verglichen werden. Dabei wird, wie zuvor auch, in jeder Simulation ein neuer Datensatz mit den gegebenen Einstellungen konstruiert und das jeweilige Verfahren auf diesen Datensatz angewandt. Alle Verfahren werden auf einen Trainingsdatensatz angepasst und anschließend mit Hilfe des RMSE auf einem Testdatensatz evaluiert. Für das RRBoost-Verfahren werden dabei 30% der Trainingsdaten zur Validierung des Initialbaums verwendet. Bei der Einstellung mancher Parameter müssen aufgrund der sehr langen Laufzeit Abstriche gemacht werden. Das bedeutet, es werden nur 20 Simulationen gekoppelt mit jeweils 100 Trainings- und 30 Testdatenpunkten betrachtet. Und auch die Anzahl der Boosting-Runden wird klein gehalten. Beim RRBoost-Verfahren sowie dem Gradienten-Boosting werden 100 Iterationen eingestellt, beim K-Vorzeichen-Boosting werden 20 Boosting-Runden angesetzt. Alle weiteren Einstellungen beim RRBoost-Verfahren und Gradienten-Boosting halten sich an die Grundeinstellungen der Funktionen in den R-Paketen. Die Feineinstellungen beim K-Vorzeichen-Boosting bleiben wie im vorherigen Abschnitt bestehen. Für mehr Details kann im zugehörigen R-Code nachgelesen werden. In Tabelle 5.1 sind die Ergebnisse der Simulationsstudie mit der Zielfunktion f_1 dargestellt. Die Zeilen in Tabelle 5.1 geben die verwendete Fehlerverteilung für die Datenerzeugung an und die Spalten kennzeichnen welches Verfahren evaluiert wird. Die Mittelwerte des RMSE zeigen für alle Verfahren und Fehlerverteilungen ein ähnliches Verhalten, wie schon bei dem Vergleich der Verfahren mit dem ersten Ansatz des K-Vorzeichen-Boosting. Das RRBoost-Verfahren schneidet über alle Fehlerverteilungen konsistent gut ab. Die Werte liegen alle um

	RRBoost	**GBoost**	**KVBoost**
D_1	1.5783 (0.2997)	1.3215 (0.2052)	9.9841 (1.4645)
D_2	1.7009 (0.2609)	32.3651 (90.5051)	10.8743 (3.2397)
D_3	1.6351 (0.2826)	2.8325 (0.5607)	10.5347 (3.4724)
D_4	1.6547 (0.2755)	24.4738 (5.9242)	10.2786 (1.9528)

Tabelle 5.1: Mittelwerte des RMSE mit den zugehörigen Standardabweichungen in Klammern der drei Verfahren bei Verwendung der Zielfunktion $f_1(\mathbf{x}) = 10\mathbf{x}_1$ mit 100 Trainings- und 30 Testbeobachtungen über 20 unabhängige Simulationen.

1.5. Und auch die in Klammern jeweils dahinter stehende empirische Standardabweichung gegeben durch $SD = \sqrt{\frac{1}{S-1}\sum_{s=1}^{S}(z_s - \bar{z})^2}$ ist klein, sodass die Werte des RMSE z_s nicht weit um den Mittelwert $\bar{z}$ streuen. Dies ist auch in Abbildung 5.1 gut einzusehen. Erwartbar war auch die gute Performance des Gradienten-Boosting auf den unkontaminierten Daten. Dort ist der mittlere RMSE kleiner als beim RRBoost mit einem Wert von circa 1.32 und einer Standardabweichung von circa 0.2. Für die trimodale Fehlerverteilung $\mathcal{D}_3$, dessen äußere Gipfel der Wahrscheinlichkeitsdichte bei -10 beziehungsweise 10 liegen, fällt der mittlere RMSE über die 20 Simulationen schon größer aus, mit einem Wert von circa 2.83. Die Ausreißer, die bei der Anwendung der Fehlerverteilung $\mathcal{D}_3$ entstehen, sind jedoch mit großer Wahrscheinlichkeit noch vergleichsweise gering, sodass die Werte des RMSE hier im Schnitt noch nicht allzu weit ausschlagen. Denn auch die Standardabweichung ist mit circa 0.5 noch klein. Anders ist es bei $\mathcal{D}_2$ und $\mathcal{D}_4$. Da diese Verteilungen sehr wahrscheinlich große Ausreißer produzieren, fallen die Mittelwerte der RMSE hier sehr groß aus. Es werden mittlere RMSE-Werte von circa 32.36 beziehungsweise 24.47 erreicht, was im Vergleich zu circa 1.7 sehr hoch ausfällt. Durch die Wahl der L2-Verlustfunktion wird beim Gradienten-Boosting Ausreißern schnell zu viel Gewicht beim Lernen zugeteilt, sodass sich das Modell zu stark an die atypischen Beobachtungen anpasst. Dadurch werden auch die neuen Vorhersagen verfälscht und es entstehen große Abweichungen der Vorhersagen zu den wahren Werten der Zielvariablen. Und auch die Standardabweichungen sind sehr groß, da die RMSE für die verschiedenen Simulationen sehr unterschiedlich ausfallen. Vor allem bei $\mathcal{D}_2$ ist die SD mit einem Wert von über 90 sehr groß, sodass die Daten sehr stark um den Mittelwert streuen. Dies lässt sich durch die hohe Wahrscheinlichkeit für große Ausreißer erklären. Denn sind solche Ausreißer enthalten, steigt der RMSE stark an. Die Abweichungen sind hier so stark, dass in

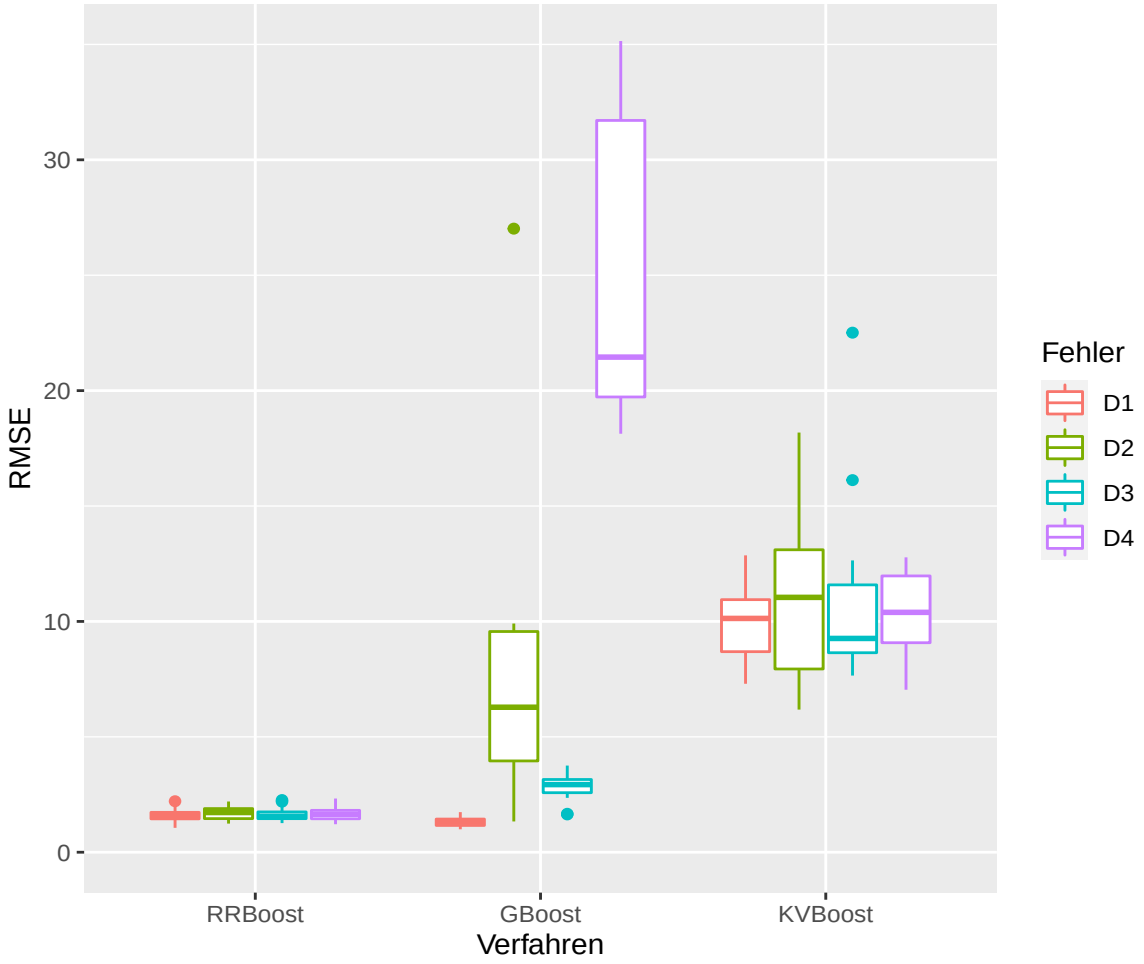

Abbildung 5.1: Boxplot der Werte des RMSE zwischen 0 und 35 der drei Verfahren über die 20 Simulationen mit Zielfunktion f_1.

Abbildung 5.1 der Wertebereich des RMSE für eine bessere Ansicht der übrigen Daten eingeschränkt wurde. Der vollständige Boxplot ist in Abbildung 5.2 zu finden und zeigt deutlich, wie stark die Ausreißer ausfallen. Mit RMSE-Werten über 400 sind die Ergebnisse des Gradienten-Boosting sehr inkonsistent bei stark durch Ausreißer belasteten Daten. Das K-Vorzeichen-Boosting zeigt im Gegensatz zum Gradienten-Boosting Konsistenz in den Mittelwerten des RMSE. Zu Beobachten ist jedoch, dass die Abweichungen zwischen Vorhersage und wahrem Wert im Mittel deutlich größer sind als zum Beispiel beim RRBoost-Verfahren. Denn die Werte liegen zwischen 9.98 und 10.88. Mit Werten zwischen 1.46 bis 3.5 streuen die RMSE über die Simulationen nicht stark. Auch wenn die Mittelwerte des RMSE beim K-Vorzeichen-Boosting groß ausfallen, ist das geringe Schwanken der Werte auch positiv zu deuten, da es

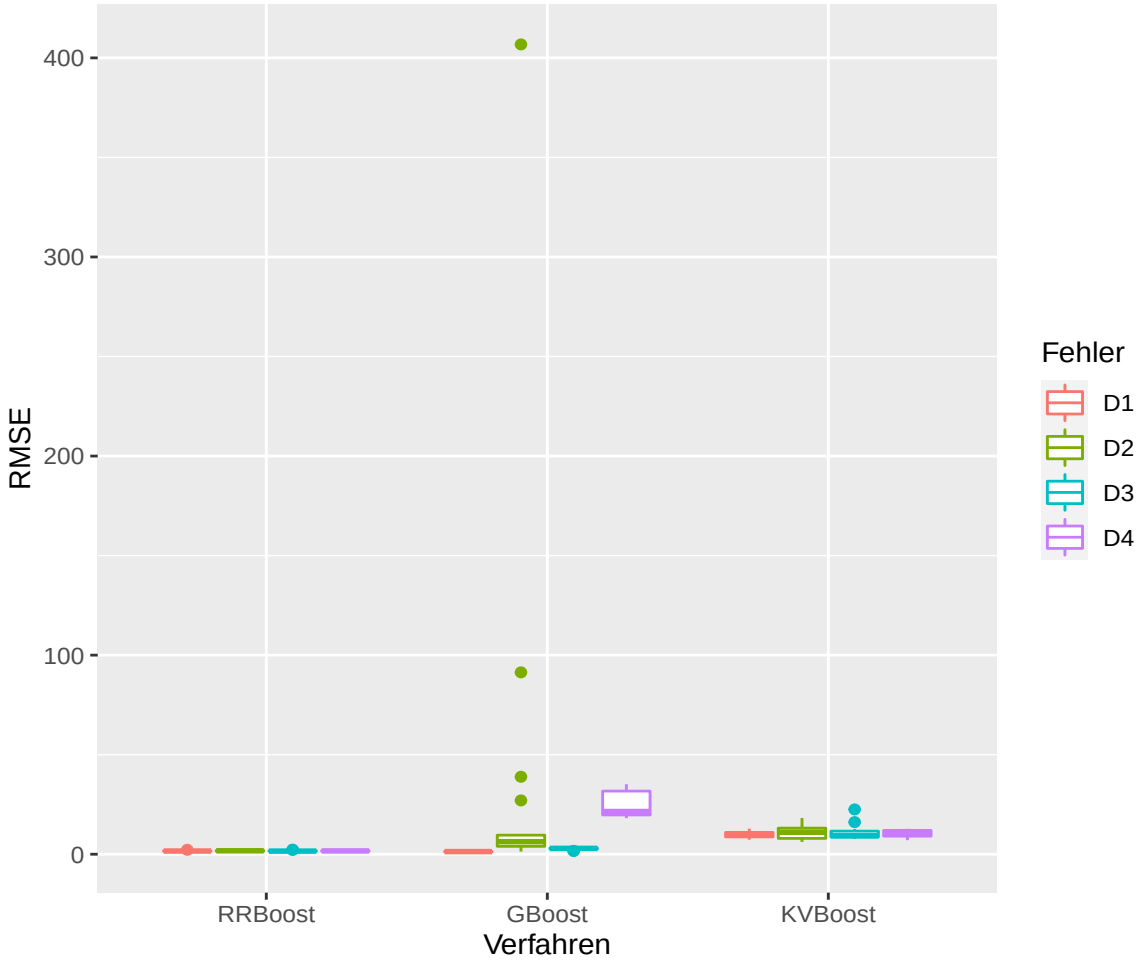

Abbildung 5.2: Boxplot der Werte des RMSE der drei Verfahren über die 20 Simulationen mit Zielfunktion f_1.

das Verfahren verlässlich macht. Dies steht im Gegensatz zum Gradienten-Boosting. Auch anzumerken ist, dass das KVBoost-Verfahren bei stark durch Ausreißer belasteten Daten ähnlich gute beziehungsweise bessere Ergebnisse im Mittelwert des RMSE liefert als GBoost. Bei der in Tabelle 5.1 und Abbildung 5.1 beziehungsweise Abbildung 5.2 betrachteten Zielfunktion f_1 ist dieses Verhalten vergleichsweise stark ausgeprägt. Schaut man sich die Ergebnisse der Mittelwerte des RMSE für die Zielfunktionen f_2, f_3, f_4 und f_5 in Tabelle 5.2, 5.3, 5.4 und 5.5 an, dann zeigt sich dieses Verhalten ebenfalls. Die zugehörigen Boxplots der Ergebnisse sind in Abbildung A.2, A.3, A.4, A.5, A.6, A.7, A.8 im Anhang zu finden.

	RRBoost	**GBoost**	**KVBoost**
D_1	1.2571 (0.1767)	1.0320 (0.1152)	5.9897 (1.3399)
D_2	1.2897 (0.2578)	8.5009 (8.2526)	6.6030 (2.8844)
D_3	1.2208 (0.1944)	2.2185 (0.4044)	6.2285 (2.2920)
D_4	1.2700 (0.2297)	19.4611 (4.9907)	9.0750 (9.3082)

Tabelle 5.2: Mittelwerte des RMSE mit den zugehörigen Standardabweichungen in Klammern der drei Verfahren bei Verwendung der Zielfunktion $f_2(\mathbf{x}) = 3\mathbf{x}_1 + 7\mathbf{x}_2$ mit 100 Trainings- und 30 Testbeobachtungen über 20 unabhängige Simulationen.

	RRBoost	**GBoost**	**KVBoost**
D_1	1.4896 (0.2481)	1.2760 (0.1952)	9.3061 (1.8555)
D_2	1.6611 (0.3044)	7.8912 (7.2263)	8.5044 (1.6313)
D_3	1.5623 (0.2826)	2.7349 (0.3716)	9.4612 (1.805)
D_4	1.5120 (0.2694)	23.9959 (4.7015)	8.9609 (1.7135)

Tabelle 5.3: Mittelwerte des RMSE mit den zugehörigen Standardabweichungen in Klammern der drei Verfahren bei Verwendung der Zielfunktion $f_3(\mathbf{x}) = 3 + 10\mathbf{x}_1$ mit 100 Trainings- und 30 Testbeobachtungen über 20 unabhängige Simulationen.

	RRBoost	**GBoost**	**KVBoost**
D_1	1.3254 (0.1163)	1.0400 (0.1796)	3.9854 (0.9253)
D_2	1.3783 (0.2597)	14.5457 (22.8518)	5.0238 (2.3326)
D_3	1.2701 (0.1909)	2.0897 (0.3338)	6.2475 (3.4301)
D_4	1.2756 (0.1603)	17.9829 (4.5291)	5.1100 (1.9438)

Tabelle 5.4: Mittelwerte des RMSE mit den zugehörigen Standardabweichungen in Klammern der drei Verfahren bei Verwendung der Zielfunktion $f_4(\mathbf{x}) = 5 + 3\mathbf{x}_1 + 7\mathbf{x}_2$ mit 100 Trainings- und 30 Testbeobachtungen über 20 unabhängige Simulationen.

	RRBoost	**GBoost**	**KVBoost**
D_1	1.5484 (0.3282)	1.2238 (0.1785)	14.7613 (2.9823)
D_2	1.7404 (0.2807)	53.8133 (98.7317)	13.1397 (4.8269)
D_3	1.5701 (0.2353)	2.5463 (0.4606)	15.7682 (4.8451)
D_4	1.5060 (0.2509)	23.0106 (4.6721)	22.8770 (18.1726)

Tabelle 5.5: Mittelwerte des RMSE mit den zugehörigen Standardabweichungen in Klammern der drei Verfahren bei Verwendung der Zielfunktion $f_5(\mathbf{x}) = 6 + 2\mathbf{x}_1 - 7\mathbf{x}_3 + 4\mathbf{x}_6 - \mathbf{x}_7$ mit 100 Trainings- und 30 Testbeobachtungen über 20 unabhängige Simulationen.

Für alle Funktionen, die hier betrachtet werden, zeigt sich das gleiche Verhalten hinsichtlich der Mittelwerte des RMSE. Lediglich für die Zielfunktion f_3 erreicht das K-Vorzeichen-Boosting für die Verteilung der Fehler D_2 einen mittleren RMSE, welcher leicht größer ist als der des Gradienten-Boosting.
Was abschließend noch bei den Ergebnissen der Simulationsstudie bei Verwendung der Zielfunktion f_5 auffällt ist, dass die Mittelwerte des RMSE bei dem K-Vorzeichen-Boosting durchweg sehr groß sind, obwohl die Werte der anderen beiden Verfahren in der gleichen Größenordnung einzusortieren sind wie bei den anderen Funktionen. Wie schon im vorherigen Kapitel festgestellt wurde, waren die Approximationen der Parameter weit von den wahren Werten entfernt. Dieses Verhalten spiegelt sich nun auch in den Vorhersagen wieder. Auch wird hier die Konsistenz in den Werten des mittleren RMSE durchbrochen. Denn schaut man sich das Ergebnis in der Tabelle bei Verwendung von $\mathcal{D}_4$ an, fällt auf, dass der RMSE hier ausschlägt. Liegen die Werte bei den übrigen Fehlerverteilungen bei Werten um die 14, wird bei Verwendung von $\mathcal{D}_4$ über die 20 Simulationen ein mittlerer RMSE von fast 23 erreicht.

6 Zusammenfassung

Die Zielsetzung dieser Arbeit ist ein möglichst simples, robustes Boosting-Verfahren basierend auf der K-Vorzeichen-Tiefe zu konstruieren. Dieses Verfahren sollte trotz seiner Einfachheit eine möglichst gute Vorhersagegenauigkeit erreichen. Um die Performance des K-Vorzeichen-Boosting zu evaluieren, wird es mit anderen state-of-the-art Boosting-Verfahren verglichen, dem robusten RRBoost-Verfahren und dem nicht-robusten Gradienten-Boosting. Der erste Ansatz des K-Vorzeichen-Boosting bestand darin, in jeder Boosting-Runde, die solche Variable mit zugehörigem Parameter zu bestimmen, die die größte volle Tiefe bei Hinzufügen zum bereits konstruierten Modell erreicht. Die Parameterwahl wurde dabei mit der R-Funktion *optim* bewältigt. Mit dem Auswahlkriterium der vollen K-Vorzeichen-Tiefe jedoch waren die Mittelwerte des RMSE auf einem unabhängigen Testdatensatz über 10 Simulationen noch vergleichsweise groß. Und bei weiterer Analyse der ersten Iteration eines Datensatzes der mit Hilfe der Zielfunktion f_1 konstruiert wurde, lagen die Maxima der vollen Tiefen der verschiedenen Variablen, nah beieinander. Die einzige Variable mit echtem Einfluss wurde bezüglich der maximalen volle K-Tiefe nicht gefiltert. Dieses Verhalten wurde erreicht, obwohl in der weiterführenden Analyse des ersten Ansatz der Parameterbereich noch eigenständig gesetzt wurde, sodass die wahren Werte der Parameter in jedem Fall in dem Parameterbereich enthalten waren. Aus diesem Grund wurde ein neues Maß für die Güte der Anpassung des Modells verwendet. Die volle K-Vorzeichen-Tiefe wurde durch die vereinfachte Tiefe ersetzt. Der Parameterbereich, wird aus den Geraden durch je zwei Datenpunkte gestellt. Diese Auswahl für den Parameterbereich ist sinnvoll, denn es liegt nahe, dass die Zielfunktion in mitten der Punktwolke liegt. Aus der Kombination dieser beiden Vorgehensweisen mit der Boosting-Idee ist schlussendlich das in Abschnitt 4.5 erläuterte K-Vorzeichen-Boosting entstanden. Wie in Tabelle 4.10 dargestellt, liegen die geschätzten Parameter eines multiplen linearen Modells mit Intercept im Mittel schon recht nah an den wahren Werten. Dies ist jedoch nur beim ausschließlichen Betrachten der Varia-

blen der Fall, die auch einen echten Einfluss auf die Zielvariable haben. Werden noch weitere Variablen ohne echten Effekt als „Störvariablen“ mit in das Modell aufgenommen, sind die Schätzungen des Einflusses der Variablen weiter von den wahren Werten entfernt, wie es in Tabelle 4.9 gezeigt wird. Im Vergleich der Verfahren auf einem unabhängigen Testdatensatz mit Hilfe des RMSE gibt es sowohl positive als auch negative Anmerkungen für die Ergebnisse des K-Vorzeichen-Boosting zu machen. Denn obwohl die Mittelwerte des RMSE beim K-Vorzeichen-Boosting durchweg deutlich größer sind als beim robusten RRBoost-Verfahren, sind die Abweichungen zwischen den wahren Werten und den Ausprägungen der Zielvariablen doch teilweise deutlich kleiner als beim Gradienten-Boosting. Da das Gradienten-Boosting kein robustes Verfahren ist, wird das Vorgehen durch Ausreißer belastet. Vor allem bei den Fehlerverteilungen $\mathcal{D}_2$ und $\mathcal{D}_4$ ist dies durchweg stark ausgeprägt. Trotz der vergleichsweise hohen Werte des RMSE im Mittel sind diese beim K-Vorzeichen-Boosting durchweg sehr konsistent und bei den stark kontaminierten Datensätzen ist der RMSE im Mittel kleiner als beim Gradienten-Boosting. Diese Ergebnisse geben einen guten Startschuss vor allem unter dem Aspekt, dass das K-Vorzeichen-Boosting sehr simpel ist und daher leicht zugänglich.

7 Ausblick

Im Rahmen dieser Arbeit wurde lediglich ein erster Anstoß für die Konstruktion eines K-Vorzeichen-Boosting-Verfahrens gegeben. Aufgrund des begrenzten Zeitraums wurden noch viele Vorgehensweisen, Konzepte und Einstellungen der Parameter nicht ausgeschöpft und evaluiert, sodass es noch zahlreiche Möglichkeiten gibt das Verfahren weiter zu entwickeln und damit auch zu verbessern.
Eine dieser Verbesserungsmöglichkeiten greift direkt im Kern des Verfahrens, der vollen beziehungsweise vereinfachten K-Vorzeichen-Tiefe. Denn im R-Paket *GSignTest* Horn (2021a) werden aufgrund der Bedingung

$$\mathbb{P}(R_i(\boldsymbol{\theta}) = 0) = 0, \qquad i = 1, \ldots, N,$$

alle Residuen, die einen Wert von 0 haben, nicht für die Berechnung der (vereinfachten) Tiefe betrachtet. Das Verfahren, welches den Parameterbereich im K-Vorzeichen-Boosting definiert, basiert darauf, dass mindestens ein oder zwei Residuen den Wert 0 haben. Dies führt dazu, dass bei jeder Berechnung Informationen verloren gehen. Darüber hinaus weist ein Residuum, dass gleich 0 ist, auf eine gute Anpassung des Modells hin, denn im besten Fall sind alle Residuen gleich 0. Eine Möglichkeit die Anpassung des Modells an die Daten besser messen zu können wäre es also, Residuen mit dem Wert 0 in die vereinfachte beziehungsweise volle Tiefe mit einzubeziehen. K-Tupel, die Nullen enthalten könnten trotzdem positiv in die Bewertung mit einfließen, wenn sie über die Nullen hinaus alternierende Vorzeichen aufweisen. Enthält ein K-Tupel nur Nullen beziehungsweise ausschließliche einen Eintrag, der nicht 0 ist, dann spricht dies für eine gute Anpassung des Modells. Die Länge K der Tupel ist noch ein weitere Stellschraube, an welcher gedreht werden kann. Im Rahmen dieser Arbeit wurde ausschließlich eine Tupellänge von drei gesetzt. Das Validieren dieses Parameters ist auch eine weitere Option um das Verfahren zu optimieren. Wie im RRBoost-Verfahren auch, könnte ein Validierungsdatensatz aus dem

Trainingsdatensatz entnommen werden, und damit für eine bestimmte Auswahl an Werten für K geschaut werden, welcher dieser Werte für ein gewähltes Kriterium die beste Performance auf dem Validierungsdatensatz liefert. Ein weiterer Parameter, welcher validiert werden könnte, wäre die Anzahl der Boosting-Runden. Dies würde gegebenenfalls eine Überanpassung des Modells verhindern. Das Vorgehen des Validierens beansprucht jedoch viel Rechenzeit. Eine weitere Möglichkeit um mit dem K-Vorzeichen-Boosting eine höhere Vorhersagegenauigkeit zu erreichen, ist es einen Schrumpfungsparameter einzuführen. Diese Option wird von vielen Boosting-Verfahren genutzt, um den Einfluss jedes Basislerners zu reduzieren und dadurch noch kleinschrittiger und akkurater die wahren Werte der Zielvariablen zu approximieren. Der Schrumpfungsparameter ist eine Konstante $\gamma \in (0, 1)$, welche in jeder Aktualisierung die Schrittweite, beziehungsweise beim K-Vorzeichen-Boosting die Approximation an die Parameter des Modells, „schrumpft“. Durch das kleinschrittigere Heranführen an die wahren Parameter, bleibt mehr Spielraum mögliche Fehler zu korrigieren, jedoch wird dadurch auch die Laufzeit des Verfahrens verlängert. Bei der Analyse des ersten Ansatzes für das K-Vorzeichen-Boosting wurde auch beobachtet, dass für die Variable, die den größten Einfluss auf die Zielvariable hat, ein stärkerer An- und Abfall um das Maximum der Tiefe stattfindet. Dieses Verhalten war zumindest bei dem einfachen linearen Modell ohne Intercept sehr ausgeprägt, sodass es sich möglicherweise auszahlt diesen Gedanken weiter zu verfolgen. Problematisch ist dabei vor allem die Parameterwahl, da der Abfall sehr davon abhängt wie weit man sich von der maximalen Tiefe entfernt. Ein weiterer Ansatz, den man zur Verbesserung der Vorhersagegüte des Verfahrens noch verfolgen kann, ist ein anderes Abbruchkriterium zu verwenden. Das Abbruchkriterium ist hier durch ein nicht-signifikantes Verbessern der vereinfachten Tiefe gegeben. Dies könnte durch das Nicht-Ablehnen der Nullhypothese des vereinfachten K-Vorzeichen-Tiefe-Tests ersetzt werden. Denn das Nicht-Ablehnen der Nullhypothese würde bereits eine gute Anpassung des Modells bedeuten. Da hier nur lineare Modelle betrachtet werden, ist man sehr eingeschränkt in der Anwendung. Denn einen linearen Zusammenhang zwischen den Einflussvariablen und der Zielvariablen vorauszusetzen bringt nicht viel Flexibilität mit sich. Daher könnte man die Basislerner, die beim K-Vorzeichen-Boosting lediglich durch die Projektionen auf die einzelnen Einflussvariablen gegeben sind, durch flexiblere Funktionen ersetzen. Die in Abschnitt 3.1 vorgestellten Regressionsbäume oder auch Splines könnten hier ihre Anwendung finden.

A Abbildungen

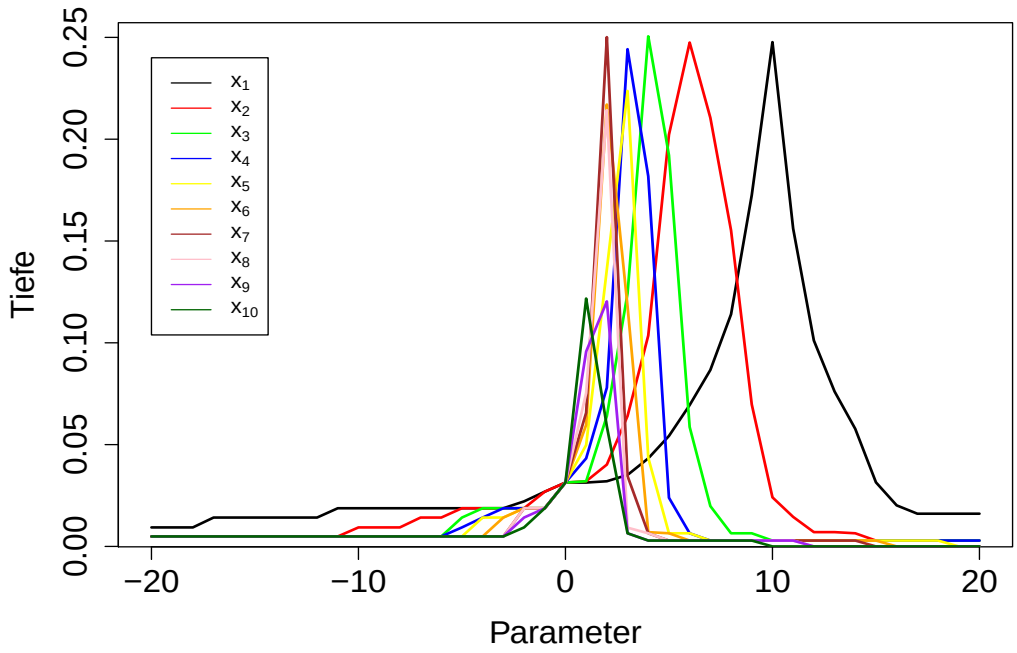

Abbildung A.1: Linear interpolierte Kurven der K-Vorzeichen-Tiefe $d_K(y_1 - \theta_j x_{1j}, \ldots, y_N - \theta_j x_{Nj})$ der $j = 1, \ldots, 10$ Einflussvariablen für $\theta \in [-20, 20] \cap \mathbb{Z}$. Die Variablen sind jedoch nicht alle gleichverteilt auf $(0, 1)$, sondern die j-te Variable ist gleichverteilt auf $(j, j + 1)$ für $j = 1, \ldots, 10$. Die Maxima mancher Variablen fallen klein aus, da die Parameter, die die maximale K-Vorzeichen-Tiefe erzeugen, nicht im betrachteten Intervall enthalten sind. Da hier die Größenordnungen der Ausprägungen der Variablen verschieden sind, erzeugen unterschiedliche Parameter die maximale Tiefe.

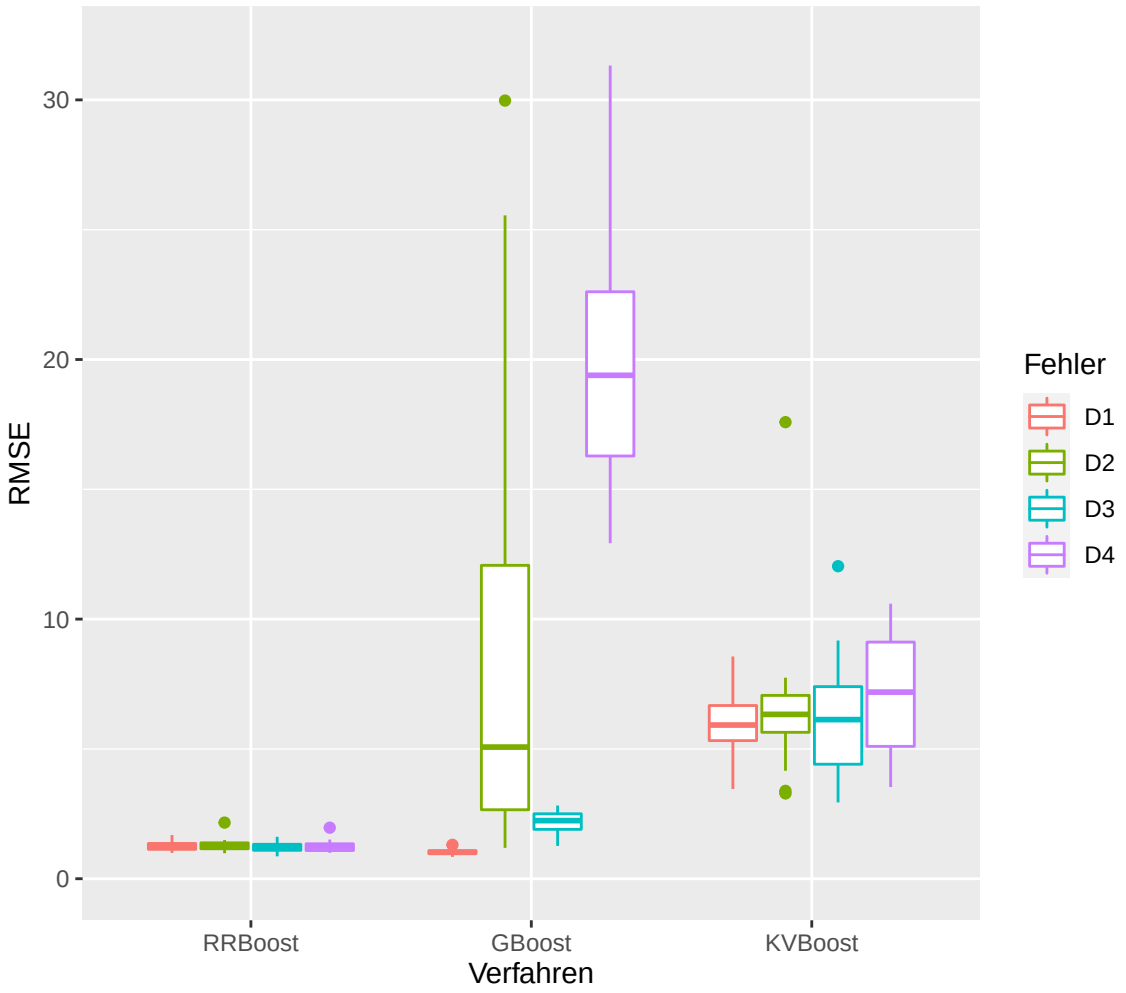

Abbildung A.2: Boxplot der Werte des RMSE zwischen 0 und 32 der drei Verfahren über die 20 Simulationen mit Zielfunktion f_2.

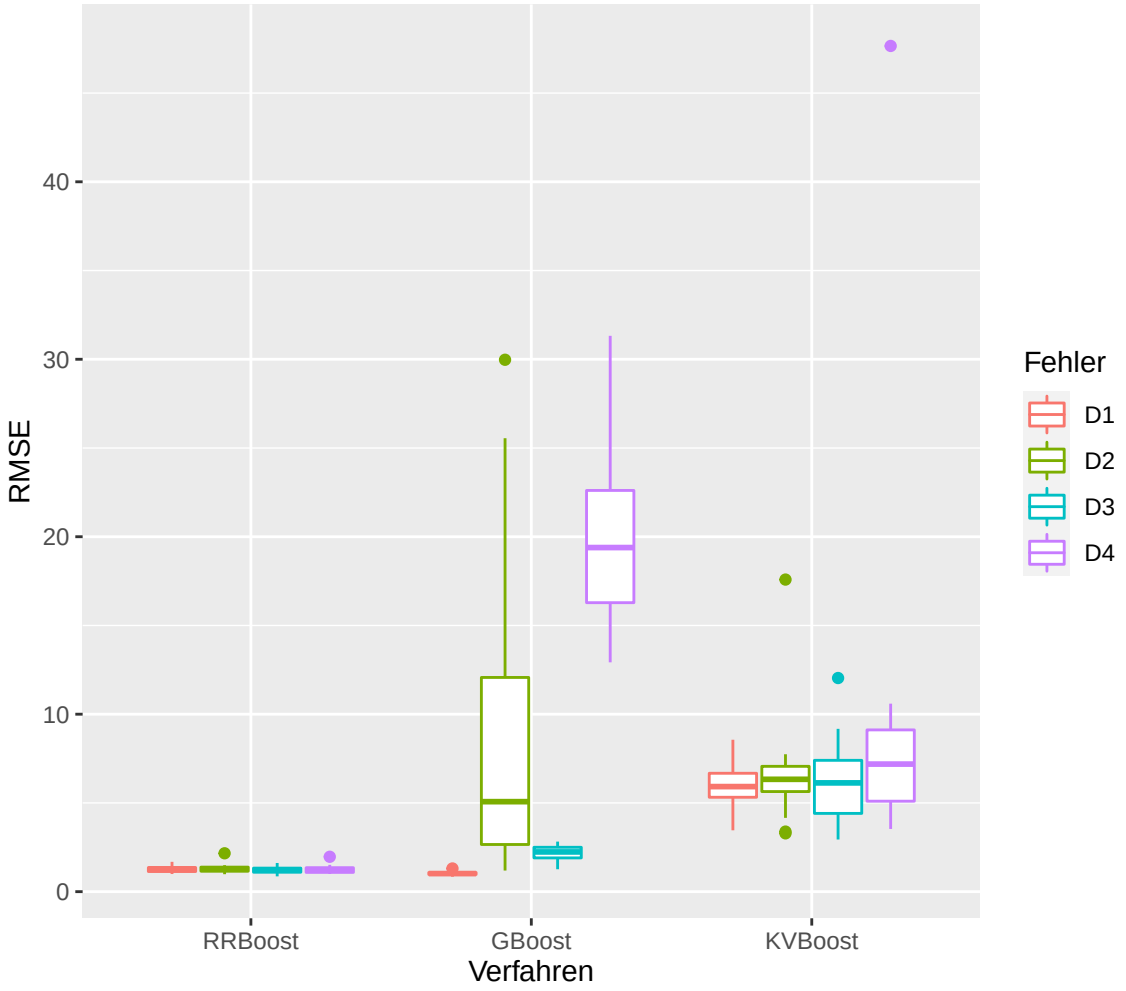

Abbildung A.3: Boxplot der Werte des RMSE der drei Verfahren über die 20 Simulationen mit Zielfunktion f_2.

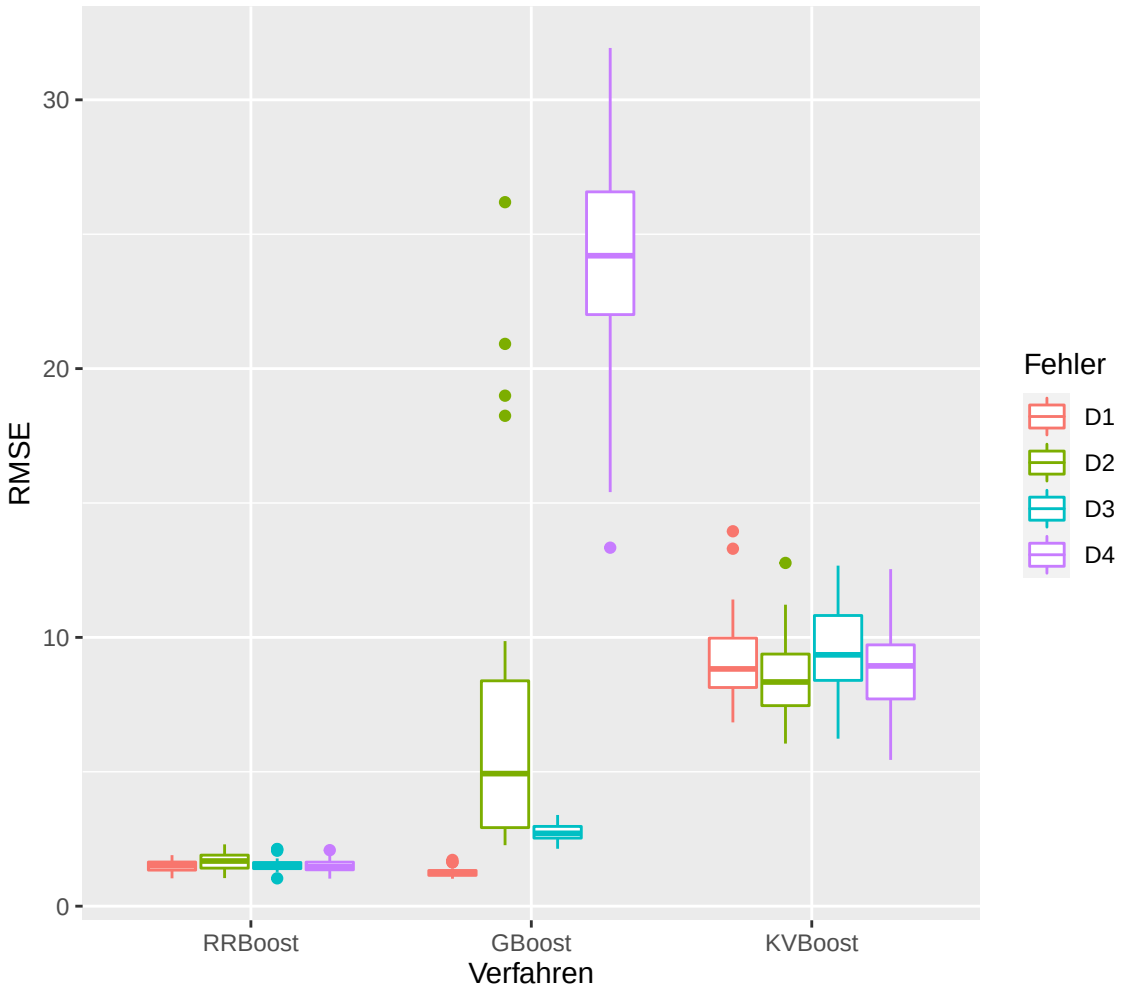

Abbildung A.4: Boxplot der Werte des RMSE der drei Verfahren über die 20 Simulationen mit Zielfunktion f_3.

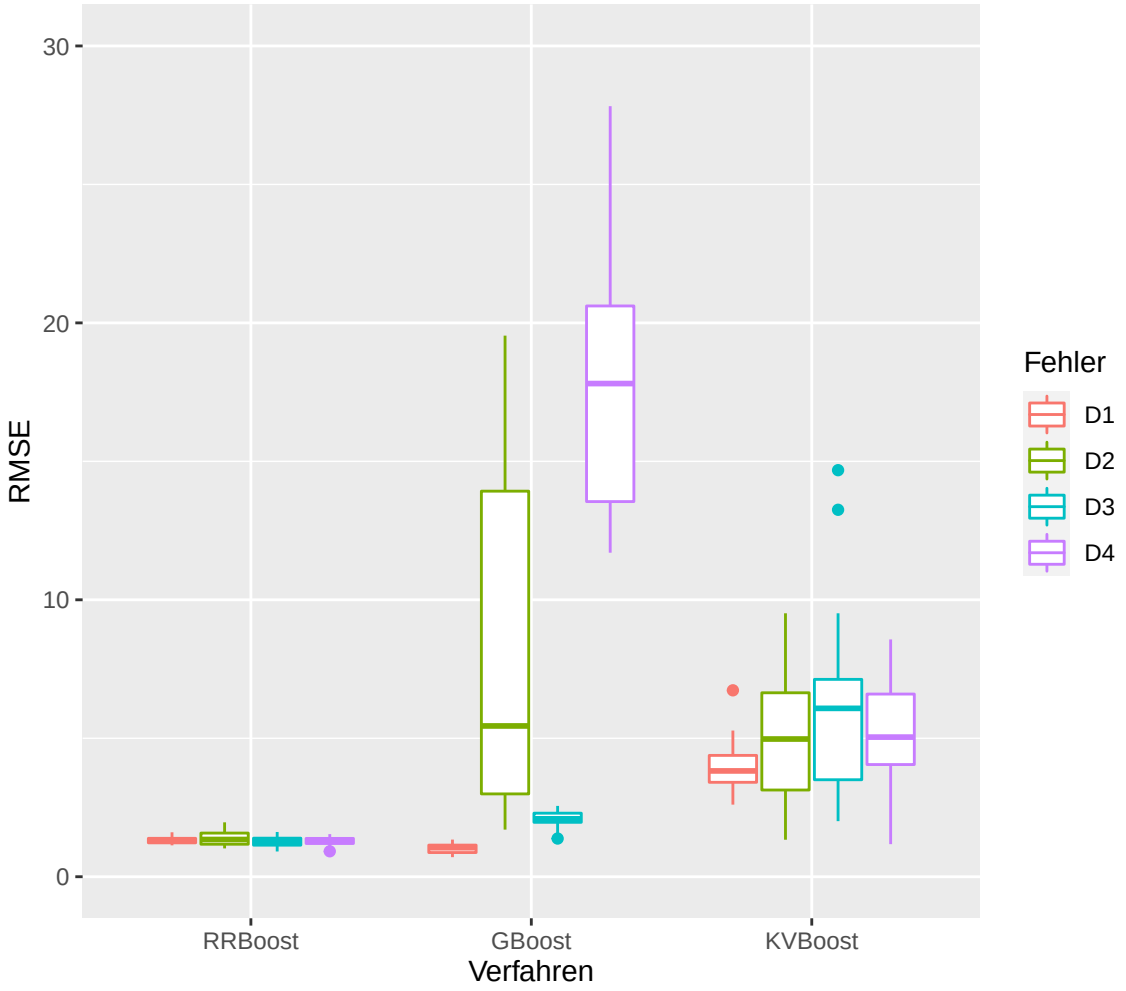

Abbildung A.5: Boxplot der Werte des RMSE zwischen 0 und 30 der drei Verfahren über die 20 Simulationen mit Zielfunktion f_4.

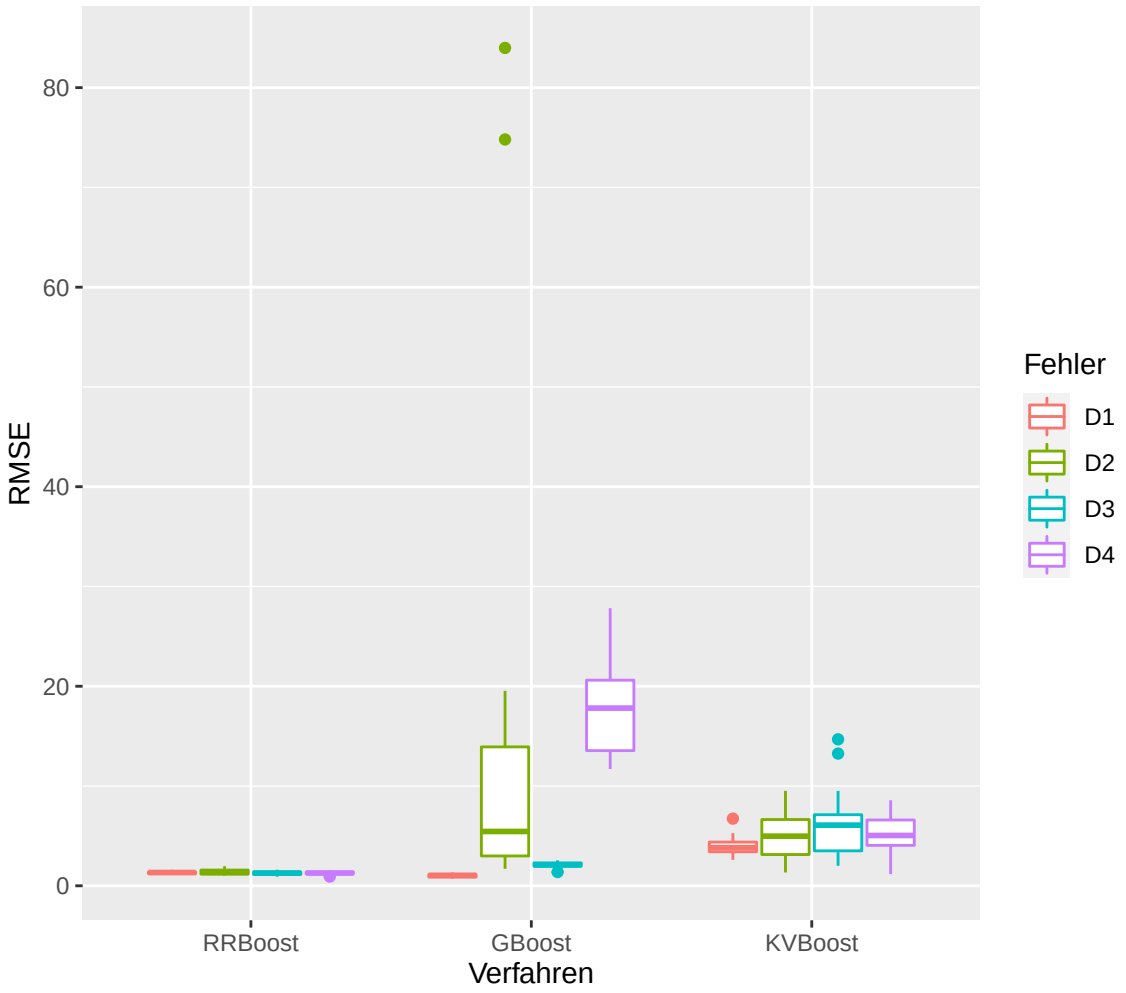

Abbildung A.6: Boxplot der Werte des RMSE der drei Verfahren über die 20 Simulationen mit Zielfunktion f_4.

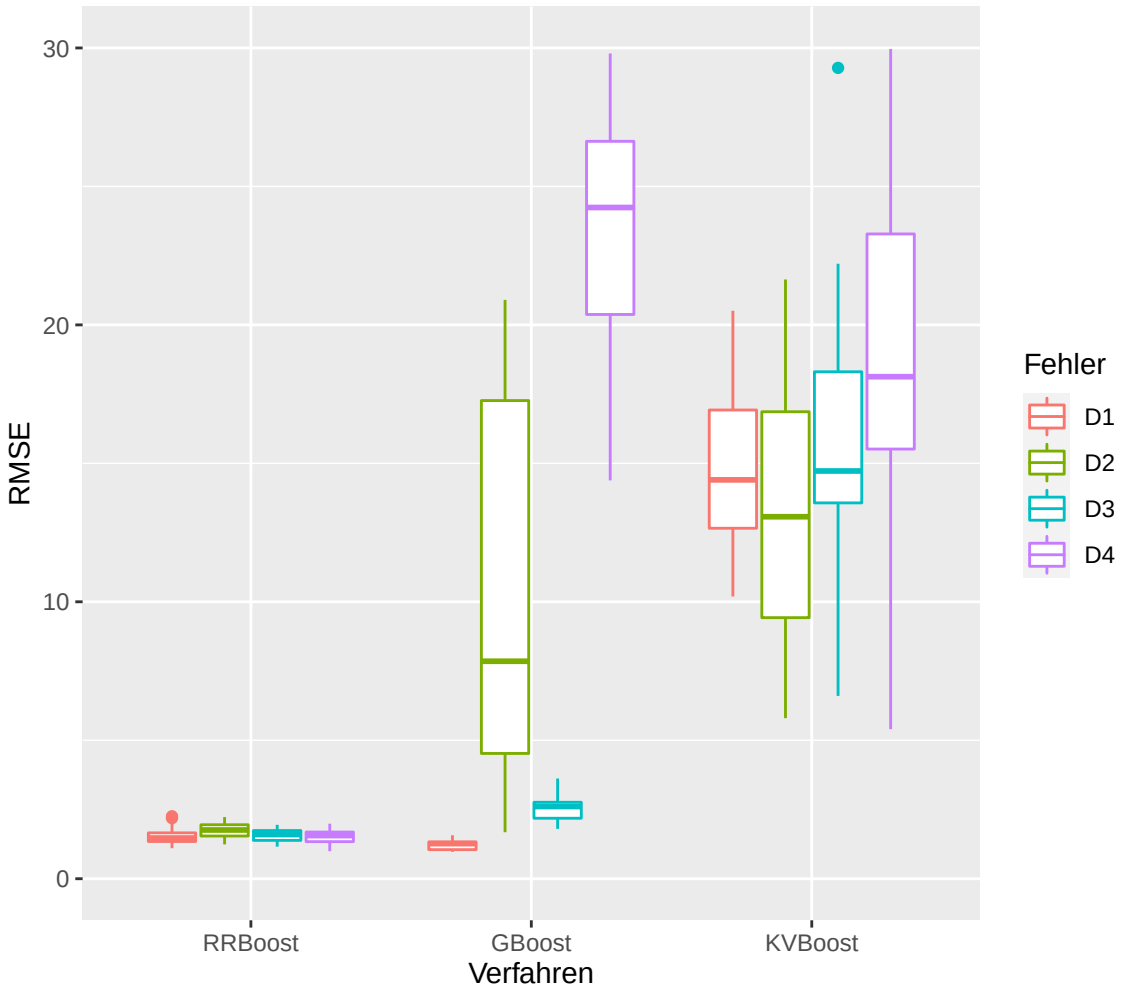

Abbildung A.7: Boxplot der Werte des RMSE zwischen 0 und 30 der drei Verfahren über die 20 Simulationen mit Zielfunktion f_5.

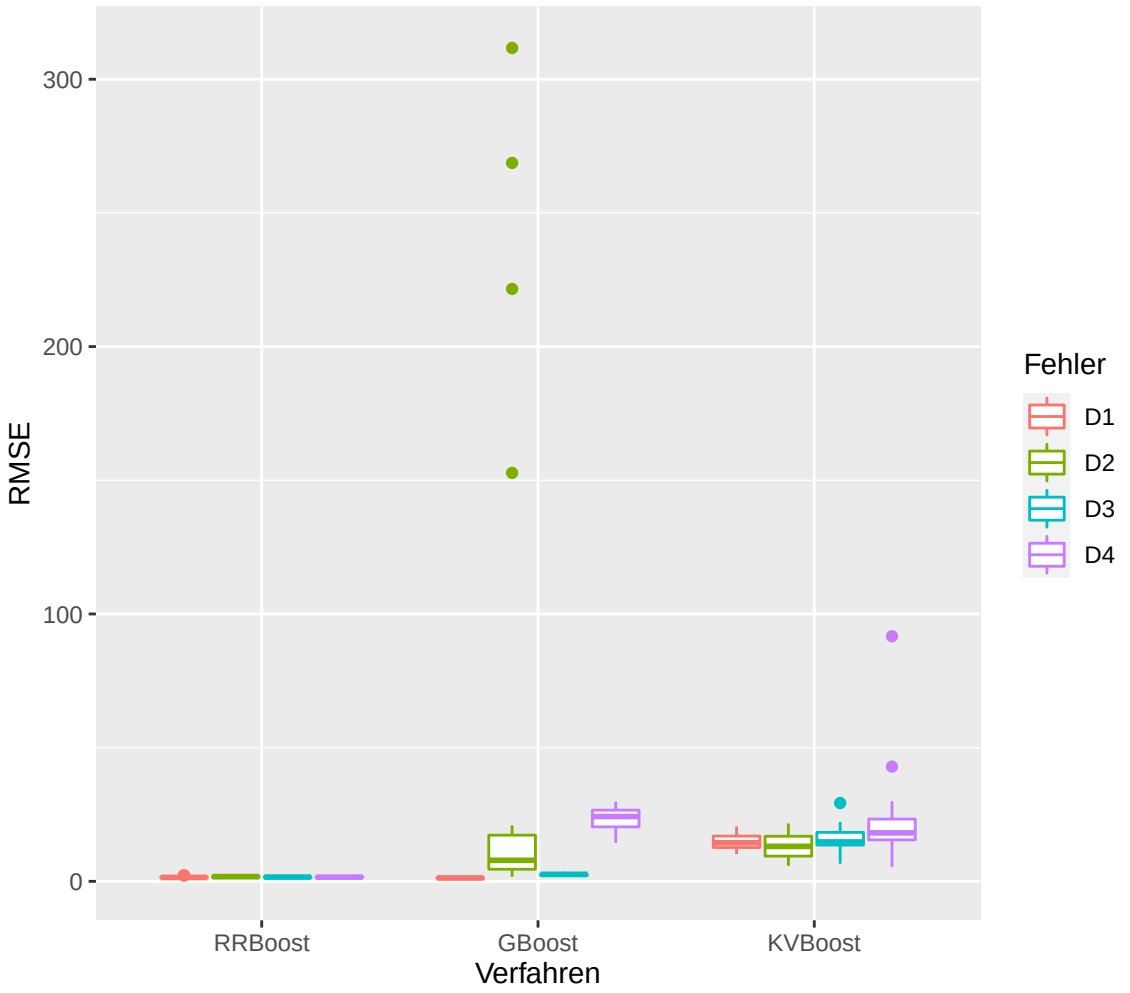

Abbildung A.8: Boxplot der Werte des RMSE der drei Verfahren über die 20 Simulationen mit Zielfunktion f_5.

Literaturverzeichnis

Bélisle, Claude J. P. 1992. "Convergence Theorems for a Class of Simulated Annealing Algorithms on Rd". *Journal of Applied Probability* 29 (4): 885–895. ISSN: 00219002, besucht am 23. Juli 2022. http://www.jstor.org/stable/3214721.

Bundesministerium für Verkehr, Bau und Stadtentwicklung. 2013. *Bauwerksprüfung nach DIN 1076 Bedeutung, Organisation, Kosten.*

Friedman, Jerome H. 2001. "Greedy function approximation: A gradient boosting machine." *The Annals of Statistics* 29 (5): 1189–1232. https://doi.org/10.1214/aos/1013203451. https://doi.org/10.1214/aos/1013203451.

Groll, Andreas. 2021. *Advanced Statistical Learning.* Vorlesungsskript.

Hastie, Trevor, Robert Tibshirani und Jerome Friedman. 2009. *The elements of statistical learning: data mining, inference and prediction.* 2. Aufl. Springer. http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

Horn, Melanie. 2021a. *GSignTest: Robust Tests for Regression-Parameters via Sign Depth.* R package version 1.0.8. https://github.com/melaniehorn/GSignTest.

———. 2021b. "Sign Depth for Parameter Tests in Multiple Regression". Dissertation, TU Dortmund University. https://eldorado.tu-dortmund.de/bitstream/2003/40483/1/Dissertation_MelanieHorn.pdf.

Hothorn, Torsten, Peter Buehlmann, Thomas Kneib, Matthias Schmid und Benjamin Hofner. 2021. *mboost: Model-Based Boosting.* R package version 2.9-5. https://CRAN.R-project.org/package=mboost.

Ju, Xiaomeng, und Matias Salibian-Barrera. 2020. *RRBoost: A Robust Boosting Algorithm.* R package version 0.1.

Ju, Xiaomeng, und Matías Salibián-Barrera. 2021. "Robust boosting for regression problems". *Computational Statistics & Data Analysis* 153:107065. ISSN: 0167-9473. https://doi.org/https://doi.org/10.1016/j.csda.2020.107065. https://www.sciencedirect.com/science/article/pii/S0167947320301560.

Kustosz, Christoph P., Christine H. Müller und Martin Wendler. 2016. "Simplified simplicial depth for regression and autoregressive growth processes". *Journal of Statistical Planning and Inference* 173:125–146. ISSN: 0378-3758. https://doi.org/https://doi.org/10.1016/j.jspi.2016.01.005. https://www.sciencedirect.com/science/article/pii/S0378375816000069.

Leckey, Kevin, Dennis Malcherczyk und Christine Müller. 2020. "Powerful generalized sign tests based on sign depth". SFB Discussionpaper. April. http://dx.doi.org/10.17877/DE290R-21017.

Malcherczyk, Dennis, Kevin Leckey und Christine H. Müller. 2021. "K-sign depth: From asymptotics to efficient implementation". *Journal of Statistical Planning and Inference* 215:344–355. ISSN: 0378-3758. https://doi.org/https://doi.org/10.1016/j.jspi.2021.04.006. https://www.sciencedirect.com/science/article/pii/S0378375821000458.

R Core Team. 2020. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Tagesschau. 2020. "Brücke in Italien eingestürzt". Besucht am 3. Juli 2022. https://web.archive.org/web/20200408181829/https://www.tagesschau.de/ausland/italien-brueckeneinsturz-101.html.

Eidesstattliche Versicherung

(Affidavit)

Stammen, Alina

Name, Vorname
(surname, first name)

222203

Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

■ Masterarbeit
(Master's thesis)

Titel
(Title)

Vergleich von verschiedenen robusten und nichtrobusten Boosting-Verfahren

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Dortmund, 22.08.2022

Ort, Datum
(place, date)

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Dortmund, 22.08.2022

Ort, Datum
(place, date)

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**