Karlsruhe Institute of Technology

Bachelorarbeit

Automatische Auswahl von maschinellen Lernverfahren für kausale Inferenz

von

Atanas Dimitrov

Pervasive Computing Systems / TECO
Institut für Telematik
Fakultät für Informatik

Abgabadatum: 02.09.2019

Verantwortlicher Betreuer: Prof. Dr. Michael Beigl
Betreuerin: Ployplearn Ravivanpong

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und weiterhin die Richtlinien des KIT zur Sicherung guter wissenschaftlicher Praxis beachtet habe.

Karlsruhe, 02.09.2019

Abstract

Diese Bachelorarbeit beschäftigt sich mit dem Vergleich von Methoden für kausale Inferenz und mit der automatischen Auswahl von dem besten von denen abhängig von dem vorhandenen Datensatz. Dazu benutzen und erweitern wir Synth-Validation - ein Verfahren, mit dem von den echten Daten synthetische Daten mit einem gewünschtem durchschnittlichen Behandlungseffekt erstellt und dann ausgewertet werden. Dabei haben die Datensätze, auf denen wir unsere Experimente durchführen, unterschiedliche Natur - echte Rohdaten, synthetisch generierte Daten aus Rohdaten und zufällig generierte Daten. Die Verfahren für kausale Inferenz, von denen Synth-Validation auswählt, benutzen ausschließlich Algorithmen aus dem maschinellem Lernen. Es wird die Fähigkeit von Synth-Validation gemessen, den Verfahren zu wählen, der die beste Schätzung von dem durchschnittlichen Behandlungseffekt hat. Das wird unter unterschiedlichen Konstellationen unterstellt - nach der Art der Daten, nach der Anzahl der Elementen in der Stichprobe usw.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

Um die Welt besser zu verstehen, haben die Menschen immer die Antwort auf die folgenden Frage gesucht: Was passiert, wenn eine bestimmte Tat oder Handlung durchgeführt wird? Manche Handlungen kann man als „einfach“ qualifizieren und bei denen ist diese Antwort leicht zu erreichen. Wenn man einen Apfel nach oben wirft, fällt er wieder nach unten. Es kostet (fast) nichts, diese Tatsache zu prüfen. Fast jeder kann dieses Ergebnis vorhersagen, weil man irgendwann mal ein fallendes Objekt beobachtet hat, unabhängig davon, ob man weiß, warum die Objekte fallen. Handlungen können aber deutlich komplexer sein wie z.B. die Einnahme einer neuen Steuerpolitik oder die medizinische Behandlung mit einem Medikamenten. Diese Handlungen sind schwerer durchzuführen, sind von mehreren Faktoren oder Naturgesetzen betroffen, sind „abhängiger“. Auf Basis früherer Erfahrung im Bereich können Experten Rahmen davon setzen, was passieren wird. Sicher kann man nur dann sein, wenn die Handlung durchführt ist und man beobachtet und bemisst, was passiert ist. Die letzte Schlussfolgerung gilt sowohl für die „einfacheren“, als auch für die „komplexeren“ Handlungen. Also um eine sichere Kenntnis zu erschaffen, brauchen wir Erfahrung - in dem Fall mit dem Apfel entweder werfen wir den alleine oder beobachten wir jemanden, der das macht oder der das irgendwann mal gemacht hat. Bei den „komplexeren“ Handlungen ist meistens die Antwort auf der Frage „Was?“ nicht ausreichend - man braucht „Wie viel?“. Das führt uns langsam zu dem Sachverhalt der kausalen Inferenz.

1.1 Kausale Inferenz

Wenn wir über die Kausalität in ihrem wissenschaftlichen Sinn sprechen, muss es klar sein, dass „$\boldsymbol{A}$ verursacht $\boldsymbol{B}$“ nicht bedeutet, dass $\boldsymbol{A}$ der Hauptgrund ist, warum $\boldsymbol{B}$ passiert ist. Es bedeutet, dass $\boldsymbol{A}$ den Unterschied oder die Differenz gemacht hat, die zu $\boldsymbol{B}$ geführt haben. Es existieren möglicherweise noch weitere Gründe z.B. $\boldsymbol{C}$ und $\boldsymbol{D}$, ohne deren Existenz und ohne deren verursachten Differenz $\boldsymbol{B}$ nicht möglich wäre oder nicht den Wert halten würde, den es hält. $\boldsymbol{A}$ soll man nicht als eine Zuschreibung von $\boldsymbol{B}$ anschauen, sondern als einen Beitrag[Mac]. Dabei sollen wir unter **kausale Inferenz** ähnlich wie bei der statistischen Inferenz den Prozess und die Methodiken zum Finden von Kenntnissen über die Kausalität von irgendeiner Handlung, insbesondere das Finden von der Differenz, die sie verursacht.

Die kausale Inferenz hat aber ein fundamentales Problem. Wenn wir den Effekt von einer Handlung wissen wollen, führen wir diese Handlung aus und bemessen den Wert von irgendeiner Variable, die uns interessiert. Der echte Effekt ist aber die Differenz zwischen dieser Variable, bedingt, dass wir die Handlung durchgeführt haben, und der gleichen Variable, bedingt, dass wir die Handlung nicht durchgeführt haben[Hol86]. Natürlich ist nur eine der beiden Optionen direkt beobachtbar. Formal kann man es so darstellen:

$$\boldsymbol{TE} = (\boldsymbol{Y}|\boldsymbol{W} = \mathbf{1}) - (\boldsymbol{Y}|\boldsymbol{W} = \mathbf{0}) \tag{1.1}$$

Hier bezeichnet $\boldsymbol{TE}$ das Behandlungseffekt, $\boldsymbol{Y}$ ist die beobachtete Variable und $\boldsymbol{W}$ ist eine binäre Variable, die bezeichnet, ob die Handlung durchgeführt wurde oder nicht.

Obwohl wir den Behandlungseffekt von einer Handlung individuell nicht beobachten können, können wir schätzen, wie hoch durchschnittlich dieser Effekt ist, wenn wir die Handlung auf mehrere Objekten durchführen beziehungsweise auch auf mehrere nicht durchführen und die Differenz zwischen diesen beiden Ergebnissen als **durchschnittlichen Behandlungseffekt** oder auf Englisch **average treatment effekt(ATE)** bezeichnen. Meistens können wir die Behandlung nicht auf allen (oder eigentlich auf der Hälfte von allen) Objekten durchführen, sondern müssen wir eine Stichprobe ziehen. Formal stellen wir das so dar:

$$\boldsymbol{\tau} = \boldsymbol{E_Y}[\boldsymbol{Y}|\boldsymbol{W} = \mathbf{1}] - \boldsymbol{E_Y}[\boldsymbol{Y}|\boldsymbol{W} = \mathbf{0}] \tag{1.2}$$

In dieser Modelldarstellung sind $\boldsymbol{W}$ und $\boldsymbol{Y}$ Zufallsvariablen. $\boldsymbol{W}$ ist binär, wo $\mathbf{1}$ für eine Behandlung und $\mathbf{0}$ für keine Behandlung steht. $\boldsymbol{Y}$ ist die Zufallsvariable von einem gewünschten Wert nach der potenziellen Behandlung. $\boldsymbol{E_Y}$ steht für den Erwartungswert von $\mathbf{Y}$. Schon in dieser Darstellung treffen wir alle Probleme, die man sonst in der statistischen Inferenz trifft - damit unsere Schätzung über die Stichprobe näher zu der echten ATE der Gesamtpopulation liegt, müssen bestimmte Regeln eingehalten werden.

Gleichung 1.2 ist immer noch kein gutes Modell der Wirklichkeit, weil es oft der Fall ist, dass die Objekte, die wir behandeln, sich durch einige Merkmale unterscheiden. Diese andere Differenzen können Einfluss auf $\boldsymbol{Y}$ sowohl bedingt $\boldsymbol{W}$, als auch unbedingt $\boldsymbol{W}$ haben und diese Tatsache soll in der Gleichung einbezogen werden:

$$\boldsymbol{\tau} = \boldsymbol{E_{Y,X}}[\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{W} = \mathbf{1}] - \boldsymbol{E_{Y,X}}[\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{W} = \mathbf{0}] \tag{1.3}$$

$\boldsymbol{X}$ ist eine Zufallsvariable für die so genannten **Kovariaten**. $\boldsymbol{X}$ ist mehrdimensional und stellt den Zustand dar, in dem sich ein Objekt oder allgemeiner gesagt eine Welt befindet. Die Kovariaten sollen nicht, aber können einen Einfluss auf das Ergebnis $\boldsymbol{Y}$ haben. Diejenige mit Einfluss nennen wir **Confounders** oder **confounding Variablen**. Auf Deutsch kann man den Begriff am nächsten mit dem Wort Verwirrungsvariablen übersetzen. Die Existenz von Confounders erschwert unsere Aufgabe, den Effekt zu finden, den die Behandlung verursacht.[VS13].

In der Tat können Confounders auch indirekten Einfluss auf die Behandlung haben. Beispiel: Sei $\boldsymbol{X}$ das Alter, $\boldsymbol{W}$ - ob man raucht oder nicht und $\boldsymbol{Y}$ - ein Gesundheitsmaß. Es ist klar, dass das Alter an sich einen Effekt auf die Gesundheit hat. Die empirischen Daten zeigen aber auch,

dass das Rauchen unter älteren Menschen verbreiteter ist als unter jüngeren, d.h das Alter bewirkt sowohl das Rauchen, als auch die Gesundheit, die an sich vom Rauchen bewirkt wird. Diese Abhängigkeiten werden in *Abbildung 1* veranschaulicht.

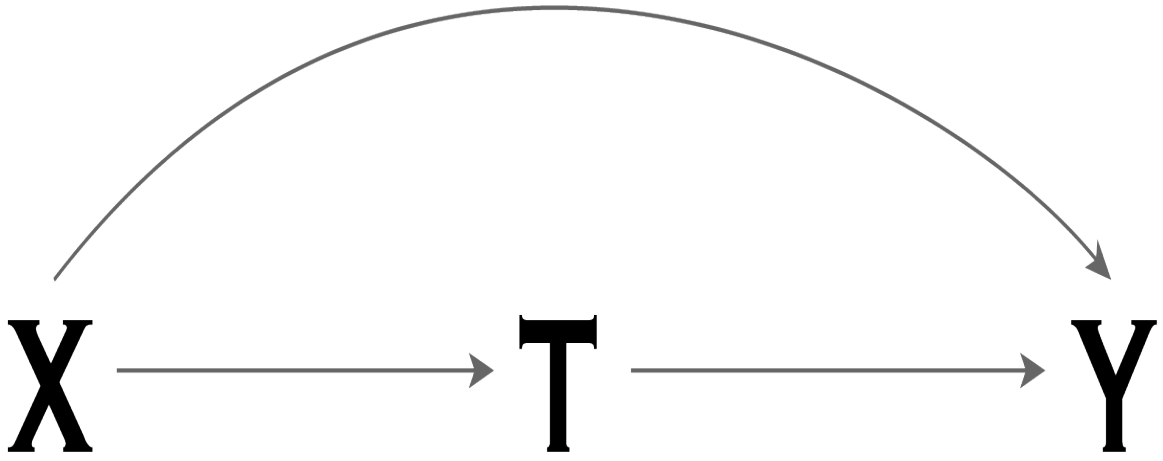

Abbildung 1: Einfluss von Confounder auf Behandlung und Ergebnis[Reb]. Hier ist $\boldsymbol{X}$ der Confounder, $\boldsymbol{T}$ - die Behandlung und $\boldsymbol{Y}$ - das Ergebnis. Es wird gezeigt, dass das Ergebniss nicht nur von der Behandlung abhängt, sondern auch vom Confounder. Der Confouder hat auch eine Auswirkung auf die Behandlung.

Wir schließen diesen Abschnitt mit dem Gedanken von dem Philosophen David Lewis über die Kausalität, der sie folgendermaßen beschreibt: „etwas, was einen Unterschied verursacht und der Unterschied, den dieses etwas verursacht, soll der Unterschied davon sein, was ohne dieses etwas passieren würde“ [Lew74].

1.2 Motivation

Es ist wichtig zu wissen, was genau eine Handlung verursacht oder welche Differenz sie ausmacht. Mit diesen Kenntnissen können wir besser die passenden Handlungen und deren genauen Maße zum Erreichen unseres Ziels einschätzen und dann auswählen. Diese Behauptungen sind allgemeingültig und wir betrachten konkret drei Beispiele von unterschiedlichen Bereichen, wo das Kennen von dem durchschnittlichen Behandlungseffekt von großer Bedeutung ist.

Beispiel 1 In der Pharmazie spielt die Kausalität eine besonders große Rolle. Wenn man ein Medikament entwickelt, soll man ganz genau wissen, welchen durchschnittlichen Effekt dieses Medikament auf unterschiedene variable körperliche Werte hat - sowohl für wohltuende, als auch für schädliche Effekte. Hier ist das Nutzen, das die kausale Inferenz bringt, die Gesundheit der Menschen.

Beispiel 2 Weitere Bedeutung hat die kausale Inferenz im Marketing und in der Bewertung von Werbungen. Eine Werbung kann eine Kette von unterschiedlichen Zielen haben, aber am Ende steht die Erhöhung von tatsächlichen Einkäufen von dem geworbenen Produkt oder Dienstleistung. Natürlich ist die Werbung nicht der einzige Treiber des Umsatzes. Also muss man nur den Effekt der Werbung einschätzen. Dann stellt man abhängig von diesem durchschnittlichen Effekt und dem Preis der Werbung fest, ob eine Werbung Nutzen bringt bzw. welche Werbung am nützlichsten ist.

Beispiel 3 Die kausale Inferenz ist bedeutend für die Auswahl vom eigenen Verhalten sein. Zum Beispiel kann man den Einfluss vom regelmäßigen Rauchen von Rauschgift auf die mentalen Fähigkeiten untersuchen. Wenn man diesen durchschnittlichen Effekt kennt, sollte es wahrscheinlicher sein, dass man auf diese schlechte Sucht verzichtet oder am besten gar nicht damit anfängt. Diese deutlichen wissenschaftlichen Daten können durch den Gesetzgeber als Rechtfertigung für einen Verbot von diesem Stoff benutzt werden.

1.3 Finden von ATE

Wir gehen davon aus, dass unser Problem schon klar definiert ist: Welchen durchschnittlichen Effekt(ATE) verursacht eine Behandlung? In diesem Abschnitt erläutern wir, wie man genau mit diesem Problem in den unterschiedlichen Situationen umgeht.

Grundsätzlich muss man *Gleichung 1.3* lösen, die wir zur Bequemlichkeit hier noch mal hinbringen.

$$\boldsymbol{\tau = E_{Y,X}[Y|X, W = 1] - E_{Y,X}[Y|X, W = 0]} \tag{1.4}$$

Dazu gibt es zwei Vorgehensweisen, die sich durch die Art von Datensammlung unterscheiden - Durchführung von einem Experimenten und Durchführung von einer Beobachtungsstudie. Die Daten aus den unterschiedlichen Studien wertet man dann anders aus.

1.3.1 Experimentdurchführung

Es gilt, dass der durch Daten von Experimenten geschätzte Behandlungseffekt weniger Bias hat und deswegen näher zu dem echten durchschnittlichen Effekt der Population liegt. Der Grund dafür ist, dass die Objekten für die beiden Gruppen zufällig ausgewählt sind und somit wie die Population verteilt sind[Rub74]. Man soll immer diese Art von Datensammlung für kausale Inferenz wählen, wenn sie möglich ist. Von den im *Abschnitt 1.2* erwähnten Beispielen ist eine Experimentdurchführung bei Beispielen 1 und 2 möglich.

Das Experiment läuft folgendermaßen durch: zwei Gruppen von Objekten werden gebildet. Die erste Gruppe heißt Behandlungsgruppe und die zweite - Kontrollgruppe. Die Objekten von der ersten Gruppe werden mit irgendetwas z.B. einem Medikamenten behandelt, diese in der zweiten jedoch nicht. Alle anderen Konditionen, auf die die Objekten in der Kontrollgruppe unterlegt sind, sollen sich nicht unterscheiden. Natürlich bleiben aber die einzelnen Objekte unterschiedlich, was zu unterschiedlichen Ergebnissen bei den individuell gemessenen Werten führt. Uns interessiert aber der durchschnittliche Wert von jeder Gruppe. Deswegen muss sichergestellt werden, dass die Objekte in jeder Gruppe gleichmäßig verteilt sind. Es gibt unterschiedliche Strategien das zu erreichen. Eine der besten, wenn sie richtig durchgeführt wird, ist die zufällige Auswahl von Objekten aus der Population für jede Gruppe. Am Ende wird für jedes Objekt der Wert von Bedeutung gemessen. Die Mittelwerte von diesen Werten werden für jede Gruppe berechnet und die Differenz dazwischen ist der erwartete durchschnittliche Behandlungseffekt.

Man kann auch Daten von den beiden Gruppen benutzen, um signifikante Aussagen über den durchschnittlichen Behandlungseffekt mit einer bestimmten Konfidenz zu treffen. Dazu führt man meistens einen statistischen $\boldsymbol{t}$-test durch. Man erstellt erstmal eine Nullhypothese und prüft, ob diese abzulehnen ist. Dafür berechnet man erstmal die Teststatistik $\boldsymbol{t}$, die hier dargestellt ist:

$$\boldsymbol{t} = \sqrt{\frac{\boldsymbol{n_0 n_1}}{\boldsymbol{n_0 + n_1}}} \frac{\overline{\boldsymbol{Y_1}} - \overline{\boldsymbol{Y_0}} - \boldsymbol{\omega}}{\boldsymbol{s}} \tag{1.5}$$

$$\boldsymbol{s} = \sqrt{\frac{(\boldsymbol{n_0} - \boldsymbol{1})\boldsymbol{s_0^2} + (\boldsymbol{n_1} - \boldsymbol{1})\boldsymbol{s_1^2}}{\boldsymbol{n_0 + n_1 - 2}}} \tag{1.6}$$

$\boldsymbol{n_0}$ und $\boldsymbol{n_1}$ sind die Anzahl von Objekten in den beiden Gruppen, $\overline{\boldsymbol{Y_1}}$ und $\overline{\boldsymbol{Y_0}}$ - die Mittelwerte von den Ergebnissen, $\boldsymbol{s_0^2}$ und $\boldsymbol{s_1^2}$ - die Varianzen, $\boldsymbol{s}$ - die gewichtete Abweichung, $\boldsymbol{\omega}$ - der Wert von dem durchschnittlichen Behandlungseffekt, den man für die Nullhypothese genommen hat. Dann abhängig davon, was man testen will und mit welcher Konfidenz die Behauptung gültig sein soll, vergleicht man den Wert von $\boldsymbol{t}$ mit dem Wert von einem Quantil von der studentischen $\boldsymbol{t}$-Verteilung. Am Ende hat man eine Behauptung, die mit irgendeiner Signifikanz wahr ist.[Wikb]

1.3.2 Beobachtungsstudiedurchführung

Wir wiederholen noch mal, dass man lieber ein Experiment anstatt einer Beobachtungsstudie durchführen muss, um die Daten zu sammeln, wenn es möglich ist. Und natürlich ist es wegen unterschiedlichen Gründen nicht immer so.

Grundsätzlich kann ein Experiment viel mehr als eine Beobachtungsstudie kosten, weil man die

Objekte (die Leute) in experimentellen Bedingungen haben möchte. Man hat (höhere) Kosten für:

- Vergütung der Leute
- die Behandlung
- Mitarbeiter, die das Experiment durchführen
- usw.

Bei manchen Studien kann der Prozess der Behandlung sogar länger dauern, was die obengenannten Kosten multipliziert.

Es gibt Behandlungen, wo es klar ist, dass sie einen negativen Effekt haben; man weiß doch nicht genau, wie hoch er ist. In diesem Fall ist es unethisch, Leute auf diese negative Behandlung in Rahmen eines Experiments zu unterlegen. Vielleicht ist die Behandlung auch durchs Gesetz verboten. Das ist die Situation im Beispiel 3 vom *Abschnitt 1.2* . Man darf nicht die Gesundheit der Leute opfern, um etwas zu erfahren. In diesem Fall führt man eine Beobachtungsstudie unter Menschen durch, die auf die Behandlung unterlegt sind und zur Kontrolle auch unter solchen, die nicht unterlegt sind.

Und hier kommt ein großes Problem. Die Objekte, die behandelt sind, können anders verteilt sein als die ganze Population. Beispielsweise gibt es unter den Behandelten viel mehr Männer als Frauen. Die Behandlung auf Männer könnte aber einen anderen Effekt haben als diesen auf Frauen. Also wenn man seine Stichprobe durch zufälliges Ziehen bildet, bekommt man einen geschätzten durchschnittlichen Effekt, der von der höheren Anwesenheit von Männern gestreut ist und nicht dem durchschnittlichen Effekt der Population entspricht. Man kann sich die *Abbildung 1* noch mal anschauen und feststellen, dass die Behandlung $\boldsymbol{T}$ von den Confounders $\boldsymbol{X}$ abhängt. In einem Experiment ist das ausgeschlossen, da es bedingt ist, dass am Anfang niemand behandelt war.

Weil die Daten aus Beobachtungsstudien viel Bias haben, würde eine bloße Berechnung von den Mittelwerten von den beiden Gruppen nicht ausreichend gut sein. Deswegen hat man unterschiedliche Methoden entwickelt, die wir ab hier **Methoden für kausale Inferenz** nennen, die grundsätzlich das Ziel haben, die Behandlugs- und die Kontrollgruppe zu normalisieren, damit die vergleichbar sind und damit das Ergebnis näher am Ergebnis der Population liegt. In dieser Bachelorarbeit haben wir über einige dieser Methoden im *Abschnitt 2* erzählt. Es ist natürlich zu erwarten, dass diese Methoden unterschiedliche Ergebnisse liefern. Dabei gibt es keine allgemein beste - in unterschiedlichen Situationen funktioniert die eine besser als die anderen. Deswegen wurde ein Verfahren entwickelt, das die Methode für einen Datensatz automatisch wählt und auf dieses Verfahren basieren wir diese Bachelorarbeit.

1.4 Ziele und Methodik

Nachdem wir in das Thema „Kausale Inferenz“ eingestiegen sind, werden wir im folgenden Abschnitt die Arbeitsmethodik dieser Bachelorarbeit erläutern. Dabei nennen wir auch die Ziele und Aufgaben, die wir uns setzen.

Wir basieren unsere Arbeit auf das Synth-Validation Verfahren[SJT+17]. Synth-Validation ist ein Verfahren, das für einen bestimmten Datensatz, zu dem wir den durchschnittlichen Behandlungseffekt schätzen wollen, versucht, von einer Menge von Methoden für kausale Inferenz, die beste auszuwählen und setzt diese ein. Im *Abschnitt 3* erzählen wir ausführlicher darüber. Wir arbeiten grundsätzlich an den Aufgaben, die im Abschnitt **Future work** im Artikel über Synth-Validation[SJT+17] genannt sind.

Das erste Ziel, das wir uns setzen, ist Synth-Validation auf R zu implementieren. Dazu verfügen wir über einen großen Teil des Codes. Der ist auf Julia geschrieben und wir haben den persönlich von einem der Autoren von Synth-Validation bekommen. Außer dem Übersetzen von Julia auf R müssen einige Teile von Synth-Validation, die im Artikel beschrieben sind, aber im vorgegebenen Julia Code jedoch fehlen, neu geschrieben werden. Darunter sind das Lesen von echten Daten, die Wahl von synthetischen Effekten, die finale Auswahl von einer Methode für kausale Inferenz von den synthetischen Daten und der ganzen Benchmarkprozess von Synth-Validation. Wir erzählen mehr darüber im *Abschnitt 4* .

Wir wählen R als Implementierungssprache, weil sie sehr verbreitet und ein Standard für statistikbasierte Ausarbeitungen ist. Es gibt eine große Gemeinschaft (auch unter Wissenschaftlern), die sie benutzt und deswegen sind auch viele Bibliotheken verfügbar. Dabei bietet die Skriptsprache eine einfache Syntax, die erlaubt schnell einzusteigen und eine Basis für kürzere Implementierungszeit ist. Wir kennen natürlich auch die potentiellen Nachteile, die die Sprache hat und die dazu geführt haben, dass man am Anfang entschieden hat, Synth-Validation auf Julia anstatt auf R zu implementieren. Diese besprechen wir im *Abschnitt 5* .

Eine weitere Aufgabe ist zu testen, wie Synth-Validation funktioniert, wenn sie auf echten Daten eingesetzt ist. Das Finden von solchen Daten ist keine triviale Aufgabe, weil der echte Behandlungseffekt mit einer großer Sicherheit bekannt sein muss, damit wir Synth-Validation benchmarken können. Im Artikel von Synth-Validation wird beschrieben, dass sie nur mit zufällig generierten Daten getestet wurde. Bei diesen Daten kann man natürlich immer einen beliebigen Effekt haben, aber bei den echten ist das Finden von diesem Effekt eigentlich das, was wir von Anfang an erzielen wollen.

Die Daten, über die wir verfügen, kommen aus Wettbewerben für kausale Inferenz, deswegen haben sie einen bekannten durchschnittlichen Behandlungseffekt. Dabei unterscheiden wir zwischen Daten, die roh oder gar nicht verarbeitet wurden und solche, die durch echten Daten generiert wurden. Die Datensätze haben auch eine unterschiedliche Anzahl von Beobachtungen und eine unterschiedliche Anzahl von Kovariaten. Neben diesen echten Daten testen wir auch

mit solchen, die zufällig generiert sind und vergleichen unsere Ergebnisse mit diesen von den Authoren von Synth-Validation, soweit es möglich ist. Mehr über die Ergebnisse, ihre Auswertung und den Sachverhalt der Daten erzählen wir im *Abschnitt 5* .

Eine weitere Aufgabe ist Synth-Validation mit neuen Methoden zu testen. Seitdem Synth-Validation entwickelt und der Artikel darüber geschrieben wurde, sind einige neuen Methoden für kausale Inferenz entstanden, die bessere Ergebnisse im Allgemein aufweisen. Diese Methoden benutzen Algorithmen aus dem maschinellen Lernen. Es wird nützlich sein, wenn man weiß, ob Synth-Validation immer noch erfolgreich unter denen wählt oder ob eine von denen sich besser als Synth-Validation vorstellt.

2 Methoden für kausale Inferenz

Man kann den durchschnittlichen Behandlungseffekt einer Handlung am einfachsten schätzen, indem man von dem Mittelwert der Ergebnisvariable der Behandlungsgruppe den Mittelwert der Ergebnisvariable der Kontrollgruppe abzieht. Diese einfache Schätzung ist sehr gut, wenn wir davon ausgehen, dass die beiden Gruppen homogen zueinander sind. Also wenn in der ersten Gruppe es keine Untergruppe von Objekten mit spezifischen Kovariaten gibt, die in der zweiten nicht vorgestellt sind. In diesem Fall sind die Daten im Gefallen von dieser Untergruppe gestreut und dieses Bias wird auf die Schätzung übertragen. Um das auszuschließen, wählt man zufällig Objekte aus, die nicht behandelt sind und führt ein Experiment mit diesen Objekten durch, in dem man die Hälfte behandelt und erst dann die Ergebnisvariable bemisst. Es gibt aber zu teure, unmögliche oder unethische Behandlungen, was dazu führt, dass die Daten darüber nur aus Beobachtungsstudien kommen. Diese Daten sind oft gestreut, was nämlich zu einer gestreuten Schätzung von der allereinfachste Methode führt[CM82a].

Um dieses Bias zu bekämpfen, hat man unterschiedliche Methoden und Techniken entwickelt. Im ersten *Unterabschnitt 2.1* werden wir über einige allgemeine, konzeptuelle Techniken erzählen, die zum Schätzen von kausalen Effekten aus Beobachtungsstudien erarbeitet wurden. Es gibt kaum ein konkretes Verfahren, an dem irgendeine davon nicht teilnimmt. Im zweiten *Unterabschnitt 2.2* stellen wir die Methoden vor, die wir in unserer Implementierung von Synth-Validation integriert haben, damit ihre Fähigkeit getestet wird, für einen bestimmten Datensatz die beste Methode zu finden.

2.1 Allgemeine Techniken

In diesem Abschnitt erzählen wir über einige allgemeine Techniken, die für die bessere Modellierung unseres Problems beigetragen haben. Man benutzt sie in vielen konkreten Verfahren zur Schätzung von kausalen Effekten unter der Konstellation, dass die Daten aus Beobachtungsstudien kommen und die allereinfachste Schätzung ungenau ist. Diese Techniken wurden originell für lineare Schätzer erarbeitet und aus diesem Grund werden wir bei deren Darstellung hier einen linearen Schätzer benutzen.

2.1.1 Anpassung für Kovariaten

Wir schauen uns erstmal die einfachste Methode zum Schätzen von dem durchschnittlichen Behandlungseffekt genauer an.

$$\widetilde{\tau} = \overline{Y_1} - \overline{Y_0} \tag{2.1}$$

Die Schätzung bekommen wir, indem wir die Differenz zwischen dem Mittelwert der Ergebnisse der Behandlungsgruppe und dem Mittelwert der Ergebnisse der Kontrollgruppe finden. Man stellt nicht so schwierig fest, dass die Kovariaten $\boldsymbol{X}$ an der Rechnung gar nicht teilnehmen. Wenn die Annahme zutrifft, dass sie homogen in den beiden Gruppen sind, sollte das nicht problematisch sein. Wenn aber nicht, wird die Schätzung gestreut sein.

In diesem Abschnitt erzählen wir über einen Ansatz, der darauf basiert, wie $\boldsymbol{Y}$ gebildet ist und wovon es abhängt. Nämlich wissen wir, dass $\boldsymbol{Y}$ sowohl von den Kovariaten $\boldsymbol{X}$, als auch von der Behandlung $\boldsymbol{W}$ abhängen kann. Wenn wir annehmen, das diese Abhängigkeit linear ist, kann $\boldsymbol{Y}$ durch das folgende lineare Model dargestellt werden:

$$\boldsymbol{Y_i} = \boldsymbol{\alpha} + \beta \boldsymbol{W_i} + \boldsymbol{\gamma} \boldsymbol{X_i} + \boldsymbol{\epsilon_i} \tag{2.2}$$

In dieser Gleichung sind $\boldsymbol{Y_i}$, $\boldsymbol{W_i}$ und $\boldsymbol{X_i}$ wie üblich das Ergebnis, die binäre Behandlungsvariable und die Kovariaten. $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ und $\boldsymbol{\gamma}$ sind die Koeffizienten, die den Beitrag von $\boldsymbol{W_i}$ und $\boldsymbol{X_i}$ in $\boldsymbol{Y_i}$ zeigen. $\boldsymbol{\gamma}$ ist natürlich ein Vektor, da $\boldsymbol{X_i}$ ein Vektor ist. $\boldsymbol{\epsilon_i}$ ist ein Störterm. Wir können diese Parameter durch lineare Regression schätzen. Unser durchschnittlicher Behandlungseffekt ist in dem Fall der geschätzte Wert von $\boldsymbol{\beta}$[CM82a].

Damit wir aber kein gestreutes Ergebnis bekommen, müssen wir einige Bedingungen einhalten:

- Nur solche Kovariaten ins Modell nehmen, die $\boldsymbol{Y}$ tatsächlich beeinflussen
- Keine Kovariaten nehmen, die von der Behandlung beeinflusst sind
- Die Anzahl der Objekten in der Stichprobe soll viel größer als die Anzahl gewählter Kovariaten sein.

Wir müssen nur solche Kovariaten ins Modell nehmen, die einen Einfluss auf das Ergebnis haben. Ansonsten wird die lineare Regression die Koeffizienten mit Bias anpassen, unter denen sich auch die Schätzung vom unseren durchschnittlichen Behandlungseffekten $\boldsymbol{\beta}$ befindet. Und das wollen wir natürlich nicht. Ein Problem ist aber, dass man nicht immer weiß, welche Variablen Einfluss haben. Eine hohe Korrelation zwischen dem Ergebnis und einem Kovariaten kann ein positives Zeichen dafür sein. Als Faustregel kann angenommen werden, dass diejenigen mit einer absoluten Korrelation kleiner als 0.2 nicht ins Modell genommen werden sollen. Es gilt, dass man die Kovariaten für das Modell bestimmen muss, bevor das Modell mit Daten angepasst wird. Ansonsten könnte man zu dem Ergebnis kommen, zu dem man kommen möchte, ohne dass es in der Grundgesamtheit repräsentativ ist[CM82a][Win].

Soweit man einen Kovariaten ins Modell nimmt, der von der Behandlung beeinflusst wurde, bekommt man Bias in der Schätzung von dem durchschnittlichen Effekt. Dieses Problem kann bei Experimenten nicht vorkommen, da man die Kovariaten immer vor der Behandlung bemisst. Bei den Beobachtungsstudien muss aufgepasst werden, dass ein Kovariat nicht mit der

Behandlung korreliert ist.[Lin]

2.1.2 Propensity Score Matching

Wenn wir Daten aus einer Beobachtungsstudie zum Finden von kausalen Effekten benutzen, ist es möglich, dass die Objekte, die behandelt sind, sich im Durchschnitt wesentlich oder zumindest in einigem Maße von den Objekten, die nicht behandelt sind, unterscheiden. Das erklärt sich mit den Confoundingvariablen, die einen Effekt auf den Zustand der Behandlung haben. Wenn wir versuchen den Behandlungseffekt von diesen qualitativ unterschiedenen Gruppen zu schätzen, bekommen wir ein Ergebnis mit Bias. Aus diesem Grund sind die beiden Gruppen nicht direkt vergleichbar.

Bevor man den Effekt schätzt, ist es sinnvoll, die beiden Gruppen irgendwie in Gleichgewicht zu bringen. Dieses Gleichgewicht werden wir durch zwei Eigenschaften definieren, die unsere beiden Gruppen erfüllen sollen. Diese Eigenschaften sind bei den Gruppen in Experimenten immer erfüllt und sind ausreichend für eine nicht gestreute Schätzung des durchschnittlichen Behandlungseffekts.

Die erste Eigenschaft ist folgende: bedingt von den Kovariaten $\boldsymbol{X}$ sind die geschätzte Werte $\boldsymbol{Y_0}$ und $\boldsymbol{Y_1}$ unabhängig von $\boldsymbol{W}$. Also gegeben $\boldsymbol{X}$ tragen $\boldsymbol{Y_0}$ und $\boldsymbol{Y_1}$ die gleichen Werte sowohl für $\boldsymbol{W = 1}$, als auch für $\boldsymbol{W = 0}$ [RR83]. Formal wird diese Eigenschaft so dargestellt:

$$\boldsymbol{Y_0, Y_1 \perp\!\!\!\perp W | X} \tag{2.3}$$

Die zweite Eigenschaft besagt, dass jedes Objekt von der Grundgesamtheit die Chance haben muss, sowohl behandelt, als auch unbehandelt zu sein. Anders gesagt ist die Wahrscheinlichkeit für ein Objekt mit bestimmten Kovariaten, behandelt zu sein, streng größer 0 und streng kleiner 1[RR83]. Formal haben wir:

$$\boldsymbol{0 < pr(W = 1|X) < 1} \tag{2.4}$$

Wir brauchen also irgendeine Funktion, die unseren Kovariaten normalisiert und nach deren Einsatz die oberen Bedingungen erfüllt sind. Deswegen führen wir die Balancing Score oder die Gleichgewichtsmaßfunktion $\boldsymbol{b(X)}$ ein, die die Kovariaten als Argument nimmt und die folgende Eigenschaft hat: die bedingte Verteilung von $\boldsymbol{X}$ gegeben $\boldsymbol{b(X)}$ ist gleich für $\boldsymbol{W = 1}$ und $\boldsymbol{W = 0}$ oder anders gesagt ist sie unabhängig von $\boldsymbol{W}$ [CM82b]. Formal wird das so dargestellt:

$$\boldsymbol{X \perp\!\!\!\perp W | b(X)} \tag{2.5}$$

Die einfachste und die feinste mögliche Balancing Score ist $\boldsymbol{b(X) = X}$. Eine Balancing Score $\boldsymbol{b(X)}$ ist feiner als eine andere Balancing Score $\boldsymbol{e(X)}$, wenn es eine Funktion $\boldsymbol{f}$ gibt, die nicht die Identität ist und für die $\boldsymbol{e(X) = f(b(X))}$ gilt. Je feiner eine Balancing Score ist, desto besser, soweit sie natürlich die Eigenschaften von einer Balancing Score trägt. Wenn wir aber viele Kovariaten haben, ist das Finden von der feinsten Balancing Score keine leichte Aufgabe. Dazu ist die Schätzung von der grobsten Balancing Score relativ einfach. Die nennen wir Propencity Score oder die Neigungsmaßfunktion $\boldsymbol{p(X)}$. Die Propensity Score ist die Wahrscheinlichkeit für einen Objekt mit bestimmten Kovariaten $\boldsymbol{X}$, dass es behandelt wird. Formal:

$$\boldsymbol{p(X) = pr(W = 1|X)} \tag{2.6}$$

Dementsprechend kann es mehr unterschiedliche Objekte geben, die die gleiche Wahrscheinlichkeit zur Behandlung haben. Rosenbaum und Rubin haben bewiesen, dass Gruppen, die von Objekten mit einander gleichen Propensity Scores erstellt sind, die obengenannten Eigenschaften erfüllen. Aus diesem Grund folgt, dass der geschätzte durchschnittliche Behandlungseffekt von diesen Gruppen kein Bias hat[RR83].

Die Propensity Score können wir durch logistische Regression schätzen. Wir haben das Modell:

$$\frac{\boldsymbol{pr(W = 1|X)}}{\boldsymbol{1 - pr(W = 1|X)}} = \boldsymbol{e^{\alpha+\beta X}} \tag{2.7}$$

Wir vereinfachen den Ausdruck, so dass wir die Propensity Score Funktion sauber haben:

$$\boldsymbol{p(X) = pr(W = 1|X) = \frac{1}{1 + e^{-(\alpha+\beta X)}}} \tag{2.8}$$

Hier sind $\boldsymbol{X}$ die Kovariaten, $\boldsymbol{\alpha}$ und $\boldsymbol{\beta}$ sind die Koeffizienten, die wir durch die logistische Regression schätzen, indem $\boldsymbol{\beta}$ ein Vektor ist, da $\boldsymbol{X}$ auch Vektor ist. Hier ist es wichtig, dass wir für die Schätzung nur Confounder benutzen, also Kovariaten, für die wir vermuten, dass sie den Zustand der Behandlung beeinflussen. Ansonsten riskieren wir, dass unsere Propensity Scores von unwesentlichen Kovariaten gestreut sind.

Nachdem wir die Propensity Score Funktion geschätzt haben, können wir die Propensity Scores von allen beobachteten Objekten berechnen. Dann bilden wir unsere Behandlungsgruppe, indem wir eine Stichprobe von behandelten Objekten ziehen. Jetzt müssen wir die zweite Stichprobe bilden, indem wir jedes Objekt aus der ersten mit einem oder mehr Kontrollobjekten nach Propensity Score verbinden. Für diese Verbindung können wir unterschiedliche Strategien benutzen. Man kann nach exakt gleichen Propensity Scores verbinden, was schwierig ist, da wir eine sehr große Stichprobe und/oder Kovariaten mit wenigen optionalen Werten haben sollen, damit wir genug gleiche Propensity Scores haben. Eine andere Option wäre, unterschiedliche Intervalle für die Propensity Score zu bilden und für jedes behandeltes Objekt ein Kontrollob-

jekt zu wählen, dessen Propensity Score im gleichen Intervall liegt wie die Propensity Score von dem behandelten Objekt. Man kann auch die Methode der nächsten Nachbarn nutzen, in dem man für jedes behandeltes Objekt ein Kontrollobjekt wählt, deren Propensity Score am nächsten zu der Propensity Score des behandelten Objekts liegt. Oder von allen Kontrollobjekten, deren Propensity Score in einem vordefinierten Abstand von der Propensity Score von jedem behandelten Objekt liegt, einen wählen[Stu10].

Ein Nachteil von den oben vorgestellten direkten Verbindungstechniken ist die Tatsache, dass einige beobachtete Kontrollobjekte von der Analyse ausgeschlossen werden. Wenn man das nicht will, kann man alle Objekte nach Propensity Score gewichten. Darüber erzählen wir im *Abschnitt 2.1.3* .

Nachdem die beiden Gruppen gebildet sind, bleibt nur die Schätzung von dem durchschnittlichen Behandlungseffekt, die wie üblich stattfindet: die Differenz zwischen den Mittelwerten von der Behandlung - und Kontrollgruppe.

2.1.3 Inverse Probability Weighting

In diesem Abschnitt werden wir eine Technik zur Schätzung von dem durchschnittlichen Behandlungseffekt vorstellen, die man teilweise als Erweiterung des oberen Abschnitts ansehen kann, weil sie die Propensity Scores benutzt.

Inverse probability of treatment weighting (IPTW) oder auf Deutsch Gewichtung mit der inversen Wahrscheinlichkeit der Behandlung ist ein Verfahren, mit dem man ein Gewicht für jedes Objekt von der Beobachtungsstudie schätzt und dieses Gewicht zur Schätzung des durchschnittlichen Behandlungseffekts nutzt. Wie der Name der Methode sagt, nehmen wir in der Gewichtsmaße die Wahrscheinlichkeit von einer Behandlung für dieses Objekt oder die Propensity Score, die wir im oberen *Abschnitt 2.1.2* erläutert haben. Das Gewicht eines Objekts $\boldsymbol{i}$ berechnen wir folgendermaßen:

$$\boldsymbol{w_i = \frac{T_i}{p(X_i)} + \frac{1 - T_i}{1 - p(X_i)}} \tag{2.9}$$

Hier ist $\boldsymbol{T_i}$ die Behandlung, die wir in dieser Bachelorarbeit sonst mit $\boldsymbol{W_i}$ darstellen. $\boldsymbol{p(X_i)}$ ist die Propensity Score vom Objekt $\boldsymbol{i}$. Wir beobachten, dass ein Gewicht $\boldsymbol{w_i}$ Werte von $\mathbf{1}$ bis $\boldsymbol{\infty}$ nimmt, da die Propensity Score ein Wahrscheinlichkeitsmaß ist, die zwischen 0 und 1 liegt. Der Einsatz der Gewichte kann man in diesem Fall als Erstellung von einer Pseudopopulation ansehen, in der jedes Objekt von unseren Beobachtungen $\boldsymbol{w_i}$-mal daran teilnimmt. In dieser Pseudopopulation sind die Objekten, für die eine Behandlung wahrscheinlicher ist, deutlich weniger als diese, für die sie unwahrscheinlicher ist. Deswegen gibt es in dieser neuen Population keine Confounder mehr, also keine Kovariaten, die Einfluss auf die Behandlung haben, und wir

können den durchschnittlichen Behandlungseffekt ungestreut schätzen[RHB00].

Es gibt eine alternative Weise, die Gewichte zu berechnen. Da bleibt das Gewicht der behandelten Objekte konstant 1 und das Gewicht der Kontrollobjekten wird durch seine Propensity Score bestimmt. Formal gilt für ein Objekt $\boldsymbol{i}$:

$$\boldsymbol{w_i = T_i + (1 - T_i)\frac{p(X_i)}{1 - p(X_i)}} \tag{2.10}$$

Die Signatur der Variablen ist wie bei *Gleichung 2.9* . Jetzt haben wir eine Pseudopopulation, in der jedes behandelte Objekt einmal vorkommt. Je wahrscheinlicher es ist, dass für ein Kontrollobjekt die Behandlung **1** wäre, desto öfter kommt dieses Kontrollobjekt vor[HIR03].

Nachdem wir die Gewichte geschätzt haben, können wir unseren durchschnittlichen Behandlungseffekt schätzen. Im Unterschied zu dem üblichen Fall, wo wir die Mittelwerte der Ergebnisvariable $\boldsymbol{Y}$ für die beiden Gruppen ausrechnen, nehmen wir jetzt die gewichteten Mittelwerte mit dem individuellen Gewicht von jedem Objekt und ziehen diese ab.

Der Vorteil von einer Gewichtung ist, dass diese Methode alle Objekte für die Schätzung von dem durchschnittlichen Behandlungseffekt nimmt, was bei den üblichen Verbindungsstrategien, die wir kurz am Ende des *Abschnitt 2.1.2* erwähnt haben, nicht der Fall ist. Der Nachteil ist, dass ein Gewicht bei einer zu hohen oder zu niedrigen Propensity Score sehr groß werden kann, was Varianz in die finalen Schätzung bringt. Wenn die Propensity Scores richtig geschätzt sind, sollte dieser Nachteil keine Rolle spielen. Wenn sie aber gestreut sind, wird dieses Bias noch stärker in der finalen Schätzung übertragen. Deswegen kann man einen Maximumwert einführen, der genommen wird, wenn ein Gewicht ihn überschreitet [Stu10].

2.2 Methoden in unserer Implementierung

In diesem Abschnitt werden wir die Methoden für kausale Inferenz vorstellen, die wir in unserer Implementierung von Synth-Validation integriert haben. Sie sind nach der Entwicklung von Synth-Validation entstanden und zeigen bessere Ergebnisse im Allgemeinen als die älteren Methoden, deswegen haben wir sie genommen. Da die drei getesteten Methoden heterogene Behandlungseffekte schätzen, erklären wir zuerst, wie sich der heterogene Behandlungseffekt von dem durchschnittlichen Behandlungseffekt unterscheidet. Dann deuten wir an, wie wir aus den heterogen Schätzungen den durchschnittlichen Effekt berechnen. Im Abschnitt 2.2.1 *stellen wir die Modellbedingungen der R-learner Methode vor, die sich für die beiden Schätzer (das Lasso und das Gradient Boosting) nicht unterscheiden. Dann stellen wir individuell die beiden Schätzer dar. Im* Abschnitt 2.2.2 *erzählen wir über eine andere Schätzmethode - die kausalen Wälder, die Random Forests für die Schätzung benutzt.*

Die Methoden, die wir in unserer Implementierung integriert haben, schätzen den heterogenen Behandlungseffekt. Wir wiederholen, dass der durchschnittliche Effekt die Differenz zwischen der durchschnittliche Ergebnisvariable aller behandelten Objekte und der durchschnittliche Ergebnisvariable aller Kontrollobjekten ist, wenn wir annehmen, dass es keine Counfonder gibt oder ihr Effekt entfernt ist. Wir bringen noch mal *Gleichung 1.3* zur Bequemlichkeit:

$$\boldsymbol{\tau = E_{Y,X}[Y|X, W = 1] - E_{Y,X}[Y|X, W = 0]} \tag{2.11}$$

Der heterogene Behandlungseffekt ist nicht der durchschnittliche, sondern der individuelle Effekt, den ein Objekt oder eine Ergebnisvariable dieses Objekts nach einer Behandlung bekommt. Der heterogene Effekt ist die Differenz zwischen die potentiellen Werten dieser Variable mit und ohne Behandlung für ein spezifisches Objekt. Die Schätzung ist nicht mehr ein einziger Wert $\boldsymbol{\tau}$, sondern eine Funktion $\boldsymbol{\tau(x)}$, die von den Werten der Kovariaten $\boldsymbol{x}$ eines Objektes abhängt. Formal stellen wir die Funktion folgendermaßen dar:

$$\boldsymbol{\tau(x) = E_X\Big[E_Y[Y|W = 1] - E_Y[Y|W = 0]\Big|X = x\Big]} \tag{2.12}$$

Da wir für unsere Implementierung den durchschnittlichen Effekt brauchen, wenden wir die geschätzte Funktion $\boldsymbol{\tau(x)}$ für alle unserer Objekte an und finden dann den Mittelwert von diesen individuellen Behandlungseffekten. Formal:

$$\boldsymbol{\tau = \frac{1}{N}\sum_{i=1}^{N}\tau(x_i)} \tag{2.13}$$

2.2.1 Quasi-Oracle Schätzung

In diesem Abschnitt werden wir über die Modellrahmen von der Quasi-Oracle Schätzung von heterogenen Behandlungseffekten[NW17] erzählen, die von Xinkun Nie und Stefan Wager entwickelt wurde. Intern hat die Methode entweder das Lasso oder Gradient Boosting als Schätzer und wir haben die beiden Varianten der Methode in unserer Implementierung von Synth-Validation integriert. Über die beiden Schätzer erzählen wir in den Unterabschnitten.

Die Methode der Quasi-Oracle Schätzung von heterogenen Behandlungseffekten oder der R-learner, der von Nie und Wager entwickelt wurde, definiert das allgemeine Problem zum Schätzen vom heterogenen Behandlungseffekt als Optimierungsproblem, das man mit einer Menge von unterschiedlichen Verfahren lösen kann. Davor muss man zwei Funktionen schätzen, die im Problem eingesetzt sind. Wenn man zum Lösen des Optimierungsproblems das Lasso benutzt, verhält sich die Methode wie ein Quasi-Orakel Schätzer - sogar im Fall von einer ungenauen Schätzung von den beiden Funktionen im ersten Schritt ist der Schätzwert

der Methode so gut wie dieser von einem Quasi-Orakel. Der Quasi-Orakel kennt die beiden Funktionen und unterscheidet sich von dem normalen Orakel dadurch, dass er nur den echten Behandlungseffekt nicht kennt. Wir werden jetzt das von der Methode benutzte Modell formal ausführlich anschauen. Alle mit $*$ hochgestellten Signaturen in diesem Abschnitt sind unbekannt, können aber geschätzt werden. Zuerst stellen wir die potenziellen Ergebnisse $\boldsymbol{Y_i(1)}$ und $\boldsymbol{Y_i(0)}$ vor, die die Variable $\boldsymbol{Y}$ nimmt, wenn man das Objekt $\boldsymbol{i}$ behandelt oder nicht behandelt. Weiter benutzt das Modell die Propensity Score $\boldsymbol{e^*(x)}$, über die wir im *Abschnitt 2.1.2* erzählt haben:

$$\boldsymbol{e^*(x) = P\Big[W = 1 | X = x\Big]} \tag{2.14}$$

Danach stellen wir die bedingte Oberfläche vom Ergebnis $\boldsymbol{\mu^*_{(w)}(x)}$ vor, die wir für die Behandlung $\boldsymbol{w = \{0, 1\}}$ haben:

$$\boldsymbol{\mu^*_{(w)}(x) = E\Big[Y(w) | X = x\Big]} \tag{2.15}$$

Noch haben wir den von der Behandlung bedingten Restwert $\boldsymbol{\epsilon_i(w)}$ von dem potenziellen Ergebnis $\boldsymbol{Y_i(w)}$:

$$\boldsymbol{\epsilon_i(w) = Y_i(w) - \Big(\mu^*_{(0)}(X_i) + w\tau^*(X_i)\Big)} \tag{2.16}$$

Weiter stellen wir den bedingten Erwartungswert des Ergebnisses $\boldsymbol{m^*(x)}$ dar:

$$\boldsymbol{m^*(x) = E\Big[Y | X = x\Big] = \mu^*_{(0)}(X_i) + e^*(x)\tau^*(X_i)} \tag{2.17}$$

Man kann jetzt die Funktion des heterogenen Behandlungseffekts $\boldsymbol{\tau^*(x)}$ anders darstellen, indem man die folgende Transformation benutzt:

$$\boldsymbol{Y_i - m^*(X_i) = (W_i - e^*(X_i))\tau^*(X_i) + \epsilon_i(W_i)} \tag{2.18}$$

Diese Darstellung nennt man die Transformation von Robinson[Rob88] und um ihn zu beehren, haben die Autoren von dieser Methode für kausale Inferenz sie nach ihm R-learner benannt. Der heterogene Behandlungseffekt nimmt dann die Form:

$$\boldsymbol{\tau^*(.) = argmin_r \Bigg\{ E\Bigg[\Bigg((Y_i - m^*(X_i)) - (W_i - e^*(X_i))\tau^*(X_i) \Bigg)^2 \Bigg] \Bigg\}} \tag{2.19}$$

Man bemerkt, dass der Term von $\epsilon_i(W_i)$ im oberen Ausdruck 2.19 im Vergleich zum Ausdruck 2.18 ausgefallen ist. Das ist ganz gezielt passiert, weil wir beobachten, dass die Confounder genauso dann keinen Einfluss auf den finalen Wert von Y_i haben, wenn gilt:

$$E\Big[\epsilon_i(W_i)|X_i, W_i\Big] = 0 \tag{2.20}$$

Wir sind jetzt bereit, unser finales Problem in zwei Schritte zu zerlegen:

1. Schätzung von $m^*(x)$ und $e^*(x)$
2. Lösung von dem folgenden Optimierungsproblem:

$$\begin{aligned} &\tau^*(.) = argmin_r\Big\{L_n(r(.)) + \Lambda_n(r(.))\Big\}, \\ &L_n(r(.)) = \frac{1}{n}\sum_{i=1}^{n}\Big(\big(Y_i - m(X_i)\big) - \big(W_i - e(X_i)\big)\tau^*(X_i)\Big)^2 \end{aligned} \tag{2.21}$$

$L_n(r(.))$ ist die Verlustfunktion, für die wir $r(.)$ finden wollen, für die die Funktion minimal ist. $\Lambda_n(r(.))$ ist ein Regulierungsterm, der durch Cross-Validation bestimmt werden kann. Bei einer solchen Definition des Problems hat man die Möglichkeit und die Flexibilität, es mit unterschiedlichen Verfahren zu lösen - lineare Modelle, Boosting, neuronale Netze usw. Die Autoren vom R-learner benutzen das Lasso oder Gradient Boosting für den ersten Schritt und das Lasso für den zweiten. In den folgenden zwei Unterabschnitten werden wir über diese zwei Schätzmethoden erzählen.

2.2.1.1 Lasso

Das Lasso oder LASSO steht für Least Absolute Shrinkage and Selection Operator oder auf Deutsch Selektionsoperator mit absoluter Schwindung. Es ist eine Erweiterung von der üblichen linearen Regression und hat viele Varianten. Wir stellen hier die allereinfachste Form vor, die aber schon die wichtigsten Charakteristiken der Methode beinhaltet.

Die lineare Regression oder genauer gesagt die Methode der kleinsten Quadrate ist im Grunde genommen für eine Datenmenge von y_i und x_i eine Lösung des folgenden Optimierungsproblems:

$$\arg\min_{\alpha,\beta}\left\{\frac{1}{N}\sum_{i=1}^{n}(y_i - \alpha - x_i^T\beta)^2\right\} \tag{2.22}$$

Dabei sind die Ergebnisse α und β, die die Methode angepasst hat, die Koeffizienten von einem linearen Model. Durch Anwendung dieser Koeffizienten kann man weitere y_i für gegebene

$\boldsymbol{x_i}$ vorhersagen. Der Fehler dieser Vorhersagen bemisst man mit der Größe des Bias und der Varianz. Und soweit die lineare Regression mit wenig Bias schätzt, hat sie oft viel Varianz. Das liegt daran, dass manche von den Inputvariablen in $\boldsymbol{x}$ in der Tat keinen Einfluss auf das Ergebnis $\boldsymbol{y}$ haben. Diese falsche Modellierung führt dazu, dass die $\boldsymbol{\alpha}$ und $\boldsymbol{\beta}$ nicht im allgemein, sondern nur für den Trainingdatensatz stimmen.

Um dieses Problem zu vermindern, kann man die Betas begrenzen oder regulieren, so dass sie nicht beliebig viel wachsen und damit Werte nehmen, die das Optimierungsproblem eigentlich besser lösen, aber in der Tat ungültig sind. Man bestimmt einen Parameter $\boldsymbol{t}$, den die absolute Summe von $\boldsymbol{\beta}$ nicht überschreiten soll[Tib96]. Wir fügen diese neue Bedingung in das Optimierungsproblem ein:

$$\mathbf{arg\,min}_{\boldsymbol{\alpha},\boldsymbol{\beta}} \left\{ \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{y_i} - \boldsymbol{\alpha} - \boldsymbol{x_i^T}\boldsymbol{\beta})^{\mathbf{2}} \right\} \text{ beschränkt von } \sum_{\boldsymbol{j=1}}^{\boldsymbol{p}} |\boldsymbol{\beta_j}| \leq \boldsymbol{t} \tag{2.23}$$

Wir geben das Problem auch in seiner Lagrangeform:

$$\mathbf{arg\,min}_{\boldsymbol{\alpha},\boldsymbol{\beta}} \left\{ \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{y_i} - \boldsymbol{\alpha} - \boldsymbol{x_i^T}\boldsymbol{\beta})^{\mathbf{2}} + \boldsymbol{\lambda} \sum_{\boldsymbol{j=1}}^{\boldsymbol{p}} |\boldsymbol{\beta_j}| \right\} \tag{2.24}$$

Durch diese Regulierung erzielen wir folgendes: unwichtige Faktoren bekommen kleinere Werte in ihre Betas. Manche Betas werden zu $\mathbf{0}$ und sind somit als unwichtig von dem Modell ausgeschlossen. Das bedeutet, dass wir das Lasso benutzen können, wenn wir unsicher sind, welche $\boldsymbol{x}$ den Wert von $\boldsymbol{y}$ bestimmen[Tib96]. Wir müssen nur den passenden Wert von $\boldsymbol{\lambda}$ finden. Je höher $\boldsymbol{\lambda}$ ist, desto mehr Faktoren reduzieren wir. Je kleiner ($\mathbf{0}$ als Minimum), desto ähnlich ist das Lasso an der linearen Regression. Zu niedrige $\boldsymbol{\lambda}$ bedeutet potenziell hohe Varianz, zu hohe - ein gestreutes Ergebnis. Für ihre Bestimmung können wir Cross-Validation benutzen[Tib96].

2.2.1.2 Gradient Boosting

Gradient Boosting ist eine Technik aus dem maschinellen Lernen, die als Ziel hat, für gegebene Daten $\boldsymbol{y_i}$ und $\boldsymbol{x_i}$ eine Funktion $\boldsymbol{y = F(x)}$ zu finden, die ihre Beziehung am besten beschreibt. Dazu benutzt sie viele „schwache“ Lerner, die individuell nicht so gut sind, aber zusammen eine insgesamt gute Schätzung produzieren. Jeder neue Lerner trägt iterativ bei. Wie üblich im maschinellen Lernen haben wir auch hier eine Verlustfunktion $\boldsymbol{L(y, \hat{y})}$, in der $\boldsymbol{y}$ für die echten Werte von den Daten und $\boldsymbol{\hat{y}}$ für die von dem Modell geschätzten Werte steht. Wenn die Summe der Verlustfunktion eingesetzt mit jedem $\boldsymbol{y_i}$ minimal ist, haben wir die beste Anpassung erreicht. Beim Gradient Boosting passiert diese Anpassung iterativ: wir fangen mit einem schwachen Schätzer $\boldsymbol{F_0(x)}$ an, der oft der Mittelwert $\boldsymbol{\overline{y}}$ ist[Wika]. Jeden folgenden Schätzer $\boldsymbol{F_{m+1}}$ stellen wir dann so dar:

$$\boldsymbol{F_{m+1}} = \boldsymbol{F_m(x)} + \boldsymbol{h(x)} \tag{2.25}$$

Hier ist $\boldsymbol{h(x)}$ die Funktion, die den vorherigen Schätzer am besten ergänzt, so dass die neue Schätzung am wenigsten von den echten $\boldsymbol{y}$ abweicht. Diese Abweichung bemessen wir durch die Verlustfunktion. Unser finaler Schätzer $\boldsymbol{\hat{F}(x)}$ hat dann die Form:

$$\boldsymbol{\hat{F}(x)} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{M}} \boldsymbol{\gamma_i h_i(x)} + \boldsymbol{const} \tag{2.26}$$

Hier ist $\boldsymbol{h_i(x)}$ die Ergänzung von jedem weiteren schwachen Schätzer und $\boldsymbol{\gamma_i}$ ist sein Gewicht. $\boldsymbol{const}$ ist der Schätzwert von dem allerersten Schätzer. In jedem Schritt schätzen wir dann die $\boldsymbol{h_m(x)}$ mit einem schwachen Schätzer $\boldsymbol{m}$, der als Eingabe alle $\boldsymbol{x_i}$ und die so genannten Pseudorestwerte $\boldsymbol{r_{im}}$ nimmt. Danach wird $\boldsymbol{\gamma_m}$ gefunden, für die die Verlustfunktion $\boldsymbol{L(y_i, F_{m-1}(x_i) + \gamma_m h_m(x_i))}$ mit der geschätzten $\boldsymbol{h_m(x)}$ minimal ist. Die Pseudorestwerte $\boldsymbol{r_{im}}$ sind der negierte Gradient von der Verlustfunktion $\boldsymbol{L(y_i, F_{m-1}(x_i))}$ mit den letzten Schätzwerten. Eine oft benutzte Verlustfunktion ist $\boldsymbol{\frac{1}{2}(y - F(x))^2}$, da ihre Pseudorestwerte den üblichen Restwerte entsprechen[Wika]. Wir geben den Algorithmus von Gradient Boosting an:

1. Konstantenwert für den ersten Schätzer setzen:
   $\boldsymbol{F_0(x) = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma)}$

2. Für $\boldsymbol{m = 1}$ bis $\boldsymbol{M}$:

   a) Pseudorestwerte ausrechnen:
   $\boldsymbol{r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}}$ für $\boldsymbol{i = 1, \ldots, n}$

   b) Modell $\boldsymbol{m}$ zur Schätzung von $\boldsymbol{h_m(x)}$ durch $\boldsymbol{r_{im}}$ anpassen

   c) $\boldsymbol{\gamma_m}$ durch das Lösen vom folgenden Optimierungsproblem finden:
   $\boldsymbol{\gamma_m = argmin_\gamma \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))}$

   d) Modell aktualisieren:
   $\boldsymbol{F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)}$

3. $\boldsymbol{F_M(x)}$ zurückgeben

2.2.2 Kausale Wälder

Die kausalen Wälder oder Causal Forests sind eine Methode zum Schätzen von dem heterogenen Behandlungseffekt $\boldsymbol{\tau(x)}$ von einer Behandlung. Dabei sind sie von den Random

Forests inspiriert und von der Struktur her unterscheiden sich von ihnen nicht. Das „Kausale“ bei den kausalen Wäldern ist die Tatsache, dass wir zwei Arten von Kovariaten in den Blättern haben - von behandelten Objekten und von Kontrollobjekten. Das verlangt von uns, dass wir eine andere finale Schätzung von den Blattelementen nutzen, den Baum etwa anders aufspalten und noch einige Bedingungen einhalten[WA18].

Random Forest ist eine Technik aus dem maschinellem Lernen, die zur Klassifikation von Objekten oder zur Regression von Funktionen benutzt wird. Der Wald besteht aus $\boldsymbol{N}$ Bäumen. Jeder Baum ist ein einzelner Schätzer und der Schätzwert des Waldes ist der durchschnittliche Schätzwert von allen Bäumen. So vermindert man die Varianz, die sonst eine Schätzung eines einzigen Baums hat. Ein Baum besteht aus Knoten und Blättern. Bei einer Modellvorhersage läuft man für ein $\boldsymbol{x}$ den Baum durch und sucht, in welchem Blatt dieses $\boldsymbol{x}$ liegen würde. Dann wird der Durchschnitt von allen Elementen in diesem Blatt als Schätzwert für $\boldsymbol{y}$ herausgegeben. Man muss aber zuerst durch den gegebenen Datensatz $(\boldsymbol{x_i}, \boldsymbol{y_i})$ den Baum anpassen. Dazu wird der Raum von $\boldsymbol{x}$ abhängig von den Daten in Unterräume aufgespaltet und die Knoten des Bäumes merken diese Aufspaltungregel. Wenn ein Unterraum eine minimale Anzahl $\boldsymbol{k}$ von Trainingdaten in sich hat oder eine weitere Aufspaltung keine Verbesserung bei der Schätzung verspricht, werden die Elemente in diesem Unterraum in ein Blatt gesteckt. Die Daten für jeden Baum werden durch zufälliges Stichprobeziehen ohne Zurücklegen von den gemeinsamen Datensatz erstellt [Bre01].

Die Daten, die die Kausalen Wälder anpassen müssen, haben außer den üblichen $(\boldsymbol{X_i}, \boldsymbol{Y_i})$ auch die Variable der Behandlung $\boldsymbol{W_i}$. Damit die Schätzung von dem heterogenen Behandlungseffekt nicht gestreut ist, muss jedes Blatt eine minimale Anzahl $\boldsymbol{k}$ von Objekten in sich haben, die sowohl behandelt als auch unbehandelt sind. Dann hat die finale Blattschätzung eine andere Form als bei dem normalen Regressionbaum - wir suchen nach der Differenz zwischen den Mittelwerten von den behandelten und unbehandelten Objekten[WA18]. Wir stellen formal die beiden Varianten dar:

$$\hat{\boldsymbol{\mu}}(\boldsymbol{x}) = \frac{\mathbf{1}}{|\{\boldsymbol{i} : \boldsymbol{X_i} \in \boldsymbol{L}(\boldsymbol{x})\}|} \sum_{\{\boldsymbol{i}:\boldsymbol{X_i}\in\boldsymbol{L}(\boldsymbol{x})\}} (\boldsymbol{Y_i}) \tag{2.27}$$

$$\begin{aligned} \hat{\boldsymbol{\tau}}(\boldsymbol{x}) = \quad & \frac{\mathbf{1}}{|\{\boldsymbol{i} : \boldsymbol{W_i} = \mathbf{1}, \boldsymbol{X_i} \in \boldsymbol{L}(\boldsymbol{x})\}|} \sum_{\{\boldsymbol{i}:\boldsymbol{W_i}=\mathbf{1},\boldsymbol{X_i}\in\boldsymbol{L}\}} (\boldsymbol{Y_i}) \quad - \\ & \frac{\mathbf{1}}{|\{\boldsymbol{i} : \boldsymbol{W_i} = \mathbf{0}, \boldsymbol{X_i} \in \boldsymbol{L}(\boldsymbol{x})\}|} \sum_{\{\boldsymbol{i}:\boldsymbol{W_i}=\mathbf{0},\boldsymbol{X_i}\in\boldsymbol{L}\}} (\boldsymbol{Y_i}) \end{aligned} \tag{2.28}$$

Gleichung 2.27 ist die finale Blattschätzung von dem Regressionsbaum, *Gleichung 2.28* - von dem kausalen Baum. $\boldsymbol{L}(\boldsymbol{x})$ ist die Menge von allen $\boldsymbol{X_i}$ aus den Trainingdaten, die in diesem Blatt liegen. Damit die Blattschätzung nicht gestreut ist, muss jeder Baum in unserem Wald

„ehrlich“ sein. Wagner und Athey[WA18] haben die Bedingung der Ehrlichkeit so erklärt: für jedes Objekt $\boldsymbol{i}$ aus den Trainingsdaten wird entweder sein $\boldsymbol{Y_i}$ für die Blattschätzung, oder sein $\boldsymbol{X_i}$ für die Aufspaltung bei dem Erstellen des Baums benutzt, aber nicht beides. Um das zu erzielen, teilt man die Daten für jeden Baum auf 2 Untergruppen $\boldsymbol{A}$ und $\boldsymbol{B}$. Dann darf man für die Aufspaltung des Baums alle Daten aus $\boldsymbol{A}$ und die $\boldsymbol{X_i}$ und $\boldsymbol{W_i}$ aus $\boldsymbol{B}$ benutzen. In den Blättern werden aber nur die Objekten aus $\boldsymbol{B}$ gesteckt, sodass nur seine $\boldsymbol{Y_i}$ für die Blattschätzung benutzt werden. Noch eine sehr wichtige Bedingung ist, dass jedes Blatt mindestens $\boldsymbol{k}$ behandelte und $\boldsymbol{k}$ unbehandelte Objekten enthalten muss. Je kleiner $\boldsymbol{k}$ ist, desto weniger gestreut wird die Schätzung sein. Desto länger wird aber die Anpassung des Baums dauern[WA18].

3 Synth-Validation

In diesem Abschnitt erzählen wir über die Theorie hinter Synth-Validation - eine Technik, die man zur Auswahl von einer Methode für kausale Inferenz für einen bestimmten Datensatz benutzt und die von Alejandro Schuler, Ken Jung, Robert Tibshirani, Trevor Hastie und Nigam Shah entwickelt wurde[SJT+17]. Wir erläutern zuerst die Motivation für die Erstellung von Synth-Validation und gehen grob durch die ganze Methode durch. Dann schauen wir uns die einzelnen Schritte des Verfahrens genauer an und erzählen über die da benutzten Algorithmen und Heuristiken in den Unterabschnitten.

In den vorigen Abschnitten haben wir unser Problem - das Finden von dem durchschnittlichen Behandlungseffekt - definiert und über seine Lösung - unterschiedliche Methoden für kausale Inferenz - erzählt. Wir gehen also davon aus, dass man seine Daten hat und die unterschiedlichen Methoden für kausale Inferenz kennt. Man weiß, dass die Methoden unterschiedliche Schätzwerte liefern werden. Einer von diesen Schätzwerten ist am nächsten zu dem echten Behandlungseffekt, aber man weiß natürlich nicht welcher. Noch ist es bekannt, dass es keine allgemeingültige Methode gibt, die für alle Daten der beste Schätzer ist - für die unterschiedlichen Datensätze liefern unterschiedliche Methoden das beste Ergebnis. Das gleiche Problem treffen wir bei den üblichen Schätzverfahren aus dem maschinellen Lernen. Da kann man aber relativ sicher die Erfolgsrate von jeder Methode ermessen, indem man einen Teil der Daten bei dem Lernen des Modells verhindert und danach mit diesen Daten die Erfolgsrate des Modells testet. Dieser Trick ist in dem kausalen Fall unmöglich, weil wir den Schätzwert der Methoden (den durchschnittlichen Behandlungseffekt) nicht direkt beobachten können und dementsprechend können wir ihn nicht aus den Testdaten nehmen[SJT+17].

Um unterschiedliche Verfahren für kausale Inferenz zu vergleichen, sollte man generative Verteilungen von Hand erstellen. Diese Verteilungen sollen dann Daten generieren, die ähnlich den echten Daten sind. Der Behandlungseffekt für die Daten wird von Anfang an vorbestimmt und es werden solche Verteilungen mit solchen Effekten gewählt, die zu den Expertenvermutungen über den Echten entsprechen. Auf den von diesen Verteilungen generierten Daten werden die Methoden ausgeführt und es wird diejenige ausgewählt, die den kleinsten durchschnittlichen Schätzfehler auf allen Datensätze herausgibt. Diese Vergleichstechnik hat einen großen Nachteil - um gute generative Verteilungen zu erstellen, muss man sowohl sehr gute Statistik-, als auch sehr gute Domainkenntnisse haben. Ansonsten können die generierten Daten nicht den Echten entsprechen[SJT+17].

Das motivierte die Erstellung von Synth-Validation. Im Unterschied zu der oberen Vorgehensweise erstellt Synth-Validation die generative Verteilungen automatisch, indem sie die echten Daten benutzt. Das vermindert den notwendigen kognitiven Aufwand und führt zu einer besseren Auswahl. Man generiert dann synthetische Daten durch die Verteilungen, führt alle vorhandenen Methoden für kausale Inferenz auf diesen Daten aus und misst den durchschnittlichen

Fehler von jeder Methode. Diejenige mit dem kleinsten Fehler wird ausgewählt. *Abbildung 2* veranschaulicht, wie Synth-Validation funktioniert. [SJT+17].

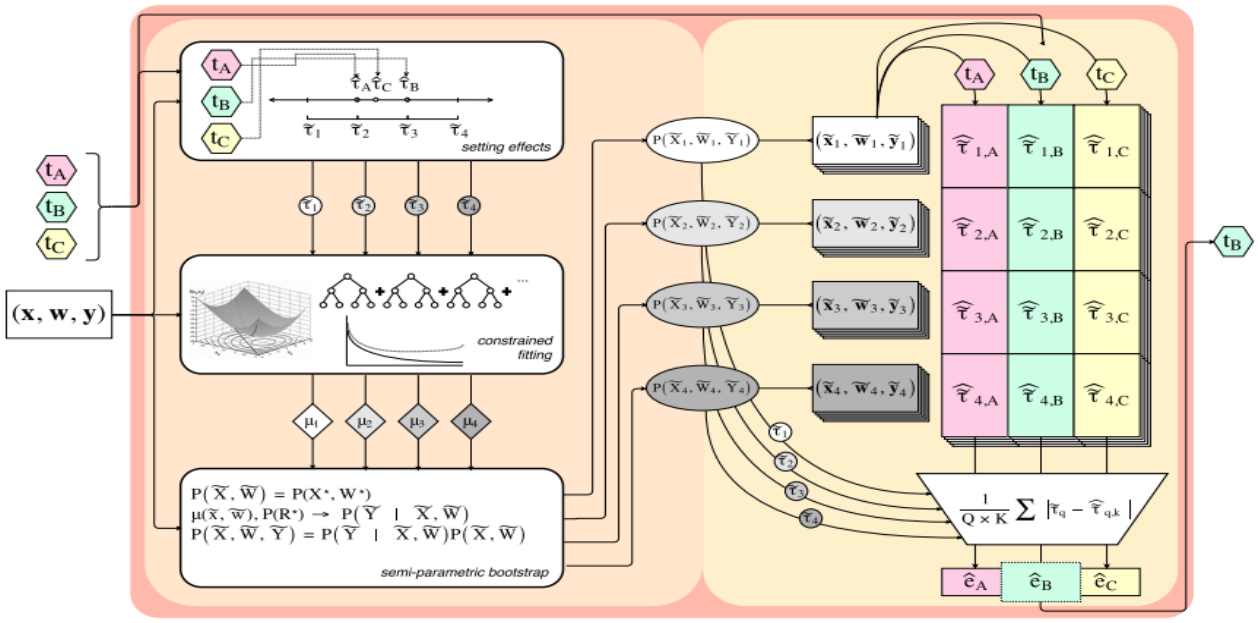

Abbildung 2: Synth-Validation[SJT+17]. Die Daten $(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y})$ und die Methoden für kausale Inferenz $\boldsymbol{t_A}$, $\boldsymbol{t_B}$ und $\boldsymbol{t_C}$ kommen als Eingabe. Man wählt zuerst die synthetischen Effekte $\boldsymbol{\tau_i}$, indem man die Methoden auf die Daten einsetzt. Dann schätzt man für jeden vorgegebenen synthetischen Effekt die bedingten Erwartungswerte $\boldsymbol{\mu_{0i}(x)}$ und $\boldsymbol{\mu_{1i}(x)}$. Man zieht eine Stichprobe von $(\boldsymbol{x}, \boldsymbol{w})$ und wendet diese auf den bedingten Erwartungswerten. Somit kriegt man die synthetischen Datensätze mit einem vorgegebenen synthetischen Effekt $\boldsymbol{\tau_i}$. Dann werden alle Methoden auf die Daten eingesetzt und da man den Behandlungseffekt von den Daten kennt, wird der durchschnittliche Fehler von jeder Methode ausgerechnet. Am Ende wird diejenige mit dem kleinsten Fehler ausgewählt [SJT+17].

In den folgenden *Abschnitt 3.1* und *Abschnitt 3.2* erzählen wir ausführlich wie die synthetischen Daten generiert werden und wie dann mit diesen neuen Daten der beste Methode für kausale Inferenz ausgewählt wird.

3.1 Generierung von synthetischen Daten

Die generative Verteilung, die wir schätzen möchten, ist nicht die echte Verteilung $\boldsymbol{P(X, W, Y)}$. Wenn man diese schätzen könnte, würden die Auswahl von den Methoden und auch die Methoden an sich umsonst, denn man könnte in diesem Fall den echten Behandlungseffekt direkt berechnen. Stattdessen wählt man zuerst synthetische Behandlungseffekten $\tilde{\boldsymbol{r}}_i$. Wie genau das passiert, erzählen wir im *Abschnitt 3.1.1* . Für unsere generativen Verteilungen

$\boldsymbol{P(\widetilde{X}, \widetilde{W}, \widetilde{Y})}$ muss gelten, dass die davon generierten Daten maximal ähnlich zu den echten Daten sind und der durchschnittliche Behandlungseffekt von diesen Daten unseren vorgegebenen $\tilde{\boldsymbol{r}}_{\boldsymbol{i}}$ entspricht [SJT+17]. Formal:

$$\tilde{\boldsymbol{\tau}} = \boldsymbol{E}_{\widetilde{Y},\widetilde{X}}[\widetilde{\boldsymbol{Y}}|\widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{W}} = \boldsymbol{1}] - \boldsymbol{E}_{\widetilde{Y},\widetilde{X}}[\widetilde{\boldsymbol{Y}}|\widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{W}} = \boldsymbol{0}] \tag{3.1}$$

Für die Schätzung von $\boldsymbol{P(\widetilde{X}, \widetilde{W}, \widetilde{Y})}$ benutzten wir die Beobachtung, dass $\boldsymbol{P(\widetilde{X}, \widetilde{W}, \widetilde{Y}) = P(\widetilde{Y}|\widetilde{X}, \widetilde{W})P(\widetilde{X}, \widetilde{W})}$ und erstellen die beiden Verteilungen individuell. Für $\boldsymbol{P(\widetilde{X}, \widetilde{W})}$ nehmen wir die empirische Verteilung von $\boldsymbol{(X, W)}$. Den Behandlungseffekt von *Gleichung 3.1* kann man auch durch die von der Behandlung bedingten Erwartungswerten von $\widetilde{\boldsymbol{Y}}$ darstellen[SJT+17]:

$$\boldsymbol{\tau} = \frac{\boldsymbol{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i}} [\boldsymbol{\mu_1(x_i) - \mu_0(x_i)}] \tag{3.2}$$

$\boldsymbol{\mu_1(\tilde{x})}$ und $\boldsymbol{\mu_0(\tilde{x})}$ sind die von uns angesprochenen bedingten Erwartungswerten, indem für sie gilt:

$$\begin{aligned} \boldsymbol{\mu_0(\tilde{x})} &= \boldsymbol{E}[\widetilde{\boldsymbol{Y}}|\widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{W}} = \boldsymbol{0}] \\ \boldsymbol{\mu_1(\tilde{x})} &= \boldsymbol{E}[\widetilde{\boldsymbol{Y}}|\widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{W}} = \boldsymbol{1}] \end{aligned} \tag{3.3}$$

Durch die bedingten Erwartungswerte können wir ein Modell für die Generierung von $\tilde{\boldsymbol{y}}_{\boldsymbol{i}}$ erstellen:

$$\tilde{\boldsymbol{y}}_{\boldsymbol{i}} = \boldsymbol{I_0(\tilde{w}_i)\mu_0(\tilde{x}) + I_1(\tilde{w}_i)\mu_1(\tilde{x}) + \epsilon_i} \tag{3.4}$$

Hier ist $\boldsymbol{I_w(w_i)}$ die Indikatorfunktion für die Behandlung $\boldsymbol{w_i}$, indem $\boldsymbol{I_w(w_i) = 1}$ für $\boldsymbol{w_i = w}$ und $\boldsymbol{I_w(w_i) = 0}$ für $\boldsymbol{w_i \neq w}$. $\boldsymbol{\epsilon_i}$ ist der Restwert von $\boldsymbol{y_i}$, mit dem unsere Schätzungen von $\boldsymbol{\mu_0(\tilde{x})}$ und $\boldsymbol{\mu_1(\tilde{x})}$ von den echten $\boldsymbol{y_i}$ abweichen. Wir müssen also zuerst die bedingten Erwartungswerte schätzen und mit deren Hilfe die Restwerten $\boldsymbol{r_i = y_i - I_0(w_i)\mu_1(x) - I_1(w_i)\mu_1(x)}$ ausrechnen. Wir benutzen dann die empirische Verteilung der Restwerte $\boldsymbol{P(R^*)}$ als Rauschmodell für die Schätzung von den synthetischen $\tilde{\boldsymbol{y}}_{\boldsymbol{i}}$[SJT+17].

Wir betonen, dass $\boldsymbol{\mu_0(\tilde{x})}$ und $\boldsymbol{\mu_1(\tilde{x})}$ nicht die echten bedingten Erwartungswerte von $\boldsymbol{Y}$ sind, sondern die bedingten Erwartungswerte, für die $\widetilde{\boldsymbol{X}}$ und $\widetilde{\boldsymbol{Y}}$ aus unseren Daten kommen und für die der Behandlungseffekt $\boldsymbol{\tau}$ aus *Gleichung 3.2* unserem vorgegebenen synthetischen Behandlungseffekt $\tilde{\boldsymbol{\tau}}$ gleich ist. Wie $\boldsymbol{\mu_0(\tilde{x})}$ und $\boldsymbol{\mu_1(\tilde{x})}$ geschätzt werden, erzählen wir im *Abschnitt 3.1.2* .

Zusammenfassend geben wir die Reihenfolge von Aktionen für die Generierung von den synthetischen Daten:

1. Auswahl von synthetischen Behandlungseffekten $\tilde{\tau_1}, \tilde{\tau_2} \ldots$ (3.1.1)
2. Schätzung von den bedingten Erwartungswerten $\mu_0(\tilde{x})$ und $\mu_1(\tilde{x})$ mit den echten Daten (x_i, w_i, y_i) und den ausgewählten synthetischen Behandlungseffekten $\tilde{\tau_1}, \tilde{\tau_2} \ldots$ aus Schritt 1 (3.1.2)
3. Stichprobeziehen aus den echten Daten (x_i, w_i) für die Generierten $(\tilde{x_i}, \tilde{w_i})$
4. Einsetzen von $(\tilde{x_i}, \tilde{w_i})$ aus Schritt 3 in $\mu_0(\tilde{x})$ und $\mu_1(\tilde{x})$ aus Schritt 2 für die Schätzung von den Ergebnissen ohne Restwerte
5. Ausrechnung von den Restwerten r_i durch y_i und die Ergebnisse aus Schritt 4; Stichprobeziehen aus den ausgerechneten Restwerten r_i für $\tilde{r_i}$
6. Generierung von $\tilde{y_i}$ durch Addieren von den Ergebnissen aus Schritt 4 mit den gezogenen Restwerten $\tilde{r_i}$ aus Schritt 5

3.1.1 Auswahl von synthetischen Effekten

Damit man die vorhandenen Methoden für kausale Inferenz auf Daten mit bekannten Behandlungseffekten testen kann, muss man zuerst diese Effekte bestimmen. Sie müssen so nah wie möglich an dem echten Behandlungseffekt liegen, damit die Auswahl am Ende sinnvoll ist. Die besten Schätzungen für ihn kriegt man durch den Einsatz von den Methoden. Es ist unklar, welche die beste ist, deswegen benutzt man eine Heuristik[SJT+17].

Die Anzahl aller synthetischen Effekte Q und ein Parameter γ kommen als Eingabe, indem wir $Q = 5$ und $\gamma = 2$ in unserer Implementierung benutzen. Man führt zuerst jede Methode t auf den Daten aus und kriegt die Schätzungen $\hat{\tau}_t$. Dann wird die Mediane von allen Schätzungen gefunden. Den kleinsten synthetischen Effekt bestimmt man, indem von der Mediane das Produkt von der Spannweite R aller Schätzungen und dem Parameter γ abgezogen wird. Der größte synthetische Effekt ist die Summe von der Mediane und diesem Produkt. Für den Rest der Effekte nimmt man die Werte, die auf gleichen Intervallen in dieser Spannweite liegen, sodass alle Effekte insgesamt Q sind[SJT+17].

$$\begin{aligned} \tilde{\tau_1} &= med(\hat{\tau}_t) - \gamma R \\ \tilde{\tau_Q} &= med(\hat{\tau}_t) + \gamma R \end{aligned} \tag{3.5}$$

3.1.2 Schätzung von bedingten Erwartungswerten

Wir müssen die bedingten Erwartungswerte $\boldsymbol{\mu_0(\tilde{x})}$ und $\boldsymbol{\mu_1(\tilde{x})}$ schätzen. Es lohnt sich, noch einmal zu wiederholen, dass sie nicht die echten bedingten Erwartungswerte, sondern diejenige, für die unsere Daten einen vorgegebenen Behandlungseffekt $\tilde{\boldsymbol{\tau}}$ haben. Für die Schätzung soll man das folgende Problem lösen[SJT+17]:

$$\begin{aligned} \boldsymbol{\mu_0(\tilde{x}), \mu_1(\tilde{x})} = \mathbf{arg}\min_{f_0, f_1 \in F} \quad & \sum_{S_0} \boldsymbol{l(y_i, f_0(x_i))} + \sum_{S_1} \boldsymbol{l(y_i, f_1(x_i))} \\ \text{bedingt von: } & \frac{\mathbf{1}}{\boldsymbol{n}} \sum_i \Big[\boldsymbol{f_1(x_i) - f_0(x_i)} \Big] = \tilde{\boldsymbol{\tau}} \end{aligned} \tag{3.6}$$

Hier ist $\boldsymbol{l(y_i, f)}$ eine Verlustfunktion z.B. $\boldsymbol{(y-f)^2}$. Für die Lösung des Problems wird Constrained Gradient Boosting eingesetzt. Constrained Boosting ist eine Form von Gradient Boosting (2.2.1.2), die aber eine Nebenbedingung für die geschätzten Funktionen hat. Im Folgenden werden wir den Algorithmus von Constrained Boosting darstellen[SJT+17].

Man versucht, die bedingten Erwartungswerte als Summe von vielen Basisfunktionen $\boldsymbol{b_{w_j}(\tilde{x})}$ zu berechnen. Formal:

$$\boldsymbol{\mu_{w_m}(\tilde{x})} = \sum_j^m \boldsymbol{\upsilon_{w_j} b_{w_j}(\tilde{x})} \tag{3.7}$$

Man schätzt die Basisfunktionen in jedem Schritt neu und addiert sie zu dem bisherigen Schätzwert. Die ersten Schätzwerten $\boldsymbol{b_{0_1}}$ und $\boldsymbol{b_{1_1}}$ sind Konstanten und es gilt $\boldsymbol{\upsilon_{0_1} = \upsilon_{1_1} = 1}$. Wir finden sie durch die Lösung vom folgenden Problem[SJT+17]:

$$\begin{aligned} \boldsymbol{\mu_{0_1}(\tilde{x}), \mu_{1_1}(\tilde{x}) = b_{0_1}, b_{1_1}} = \mathbf{arg}\min_{c_0, c_1} \quad & \sum_{S_0} \boldsymbol{l(y_i, c_0)} + \sum_{S_1} \boldsymbol{l(y_i, c_1)} \\ \text{bedingt von: } & \tilde{\boldsymbol{\tau}} = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_i \Big[\boldsymbol{c_1 - c_0} \Big] \end{aligned} \tag{3.8}$$

In den weiteren Schritten $\boldsymbol{m > 1}$ nimmt man die Restwerte $\boldsymbol{r_{m-1_i}}$ von der letzten Schätzung $\boldsymbol{\mu_{w_m}(x_i)}$ und $\boldsymbol{y_i}$ und passt die Basisfunktionen $\boldsymbol{b_{w_m}(x)}$ zu ihnen an. Dafür wird ein Regressionsbaum benutzt. In *Gleichung 3.9* ist $\boldsymbol{I_w(w_i)}$ die Indikatorfunktion für die Behandlung $\boldsymbol{w_i}$ [SJT+17]. Formal:

$$\boldsymbol{r_{m-1_i} = y_i - I_0(w_i)\mu_{0_{m-1}}(x_i) - I_1(w_i)\mu_{1_{m-1}}(x_i)} \tag{3.9}$$

$$\begin{aligned} \boldsymbol{b}_{\mathbf{0}_m}(\boldsymbol{x}) &= \arg\min_{b \in B} \sum_{S_0} \boldsymbol{l}(\boldsymbol{r}_{m-\mathbf{1}_i}, \boldsymbol{b}(\boldsymbol{x}_i)) \\ \boldsymbol{b}_{\mathbf{1}_m}(\boldsymbol{x}) &= \arg\min_{b \in B} \sum_{S_1} \boldsymbol{l}(\boldsymbol{r}_{m-\mathbf{1}_i}, \boldsymbol{b}(\boldsymbol{x}_i)) \end{aligned} \tag{3.10}$$

Man merkt, dass die Anpassung von den Basisfunktionen ohne die Bedingung über den synthetischen Behandlungseffekt stattfindet. Aus diesem Grund darf man nicht direkt $\boldsymbol{\mu}_{w_{m-1}}(\boldsymbol{x}_i)$ mit $\boldsymbol{b}_{w_m}(\boldsymbol{x})$ addieren. Wir suchen deswegen nach den Koeffizienten $\boldsymbol{\upsilon}_{\mathbf{0}_m}$ und $\boldsymbol{\upsilon}_{\mathbf{1}_m}$, für die der Behandlungseffekt $\tilde{\boldsymbol{\tau}}$ bleibt. Die Bedingung ist im ersten Schritt erfüllt. Soweit der Beitrag zu dem Behandlungseffekt in jedem weiteren Schritt $\mathbf{0}$ ist, bleibt die Bedingung erfüllt[SJT+17]. Deswegen lösen wir das Optimierungsproblem:

$$\begin{aligned} \boldsymbol{\upsilon}_{\mathbf{0}_m}, \boldsymbol{\upsilon}_{\mathbf{1}_m} = \arg\min_{c_0, c_1} \quad & \sum_{i \in S_0} \boldsymbol{l}\Big(\boldsymbol{y}_i, \boldsymbol{\mu}_{\mathbf{0}_{m-1}}(\boldsymbol{x}_i) + \boldsymbol{c}_0 \boldsymbol{b}_{\mathbf{0}_{im}}(\boldsymbol{x}_i)\Big) + \boldsymbol{\lambda} \boldsymbol{c}_0^2 + \\ & \sum_{i \in S_1} \boldsymbol{l}\Big(\boldsymbol{y}_i, \boldsymbol{\mu}_{\mathbf{1}_{m-1}}(\boldsymbol{x}_i) + \boldsymbol{c}_1 \boldsymbol{b}_{\mathbf{1}_{im}}(\boldsymbol{x}_i)\Big) + \boldsymbol{\lambda} \boldsymbol{c}_0^2 \\ \text{bedingt von:} \quad & \sum_i \Big[\boldsymbol{c}_1 \boldsymbol{b}_{\mathbf{1}_{im}}(\boldsymbol{x}_i) - \boldsymbol{c}_0 \boldsymbol{b}_{\mathbf{0}_{im}}(\boldsymbol{x}_i)\Big] = \mathbf{0} \end{aligned} \tag{3.11}$$

$\boldsymbol{\lambda}$ dient zur Regulierung von der Schätzung, sodass sie weniger Varianz hat. Am Ende berechnet man die $\boldsymbol{\mu}_{\mathbf{0}_m}(\tilde{\boldsymbol{x}})$ und $\boldsymbol{\mu}_{\mathbf{1}_m}(\tilde{\boldsymbol{x}})$ durch *Gleichung 3.7* . Die Anzahl von Schritten $\boldsymbol{m}$ ist von Anfang an nicht bekannt. Wenn sie zu niedrig ist, wird die Schätzung viel Bias haben. Wenn zu hoch, zu viel Varianz. Deswegen wird die optimale $\boldsymbol{m}$ durch Cross-Validation bestimmt[SJT+17].

3.2 Methodenauswahl

Nachdem man die generativen Verteilungen geschätzt hat, kann man synthetische Daten mit bekannten Behandlungseffekten generieren und die Methoden darauf ausführen. Von jeder Verteilung generiert man $\boldsymbol{K}$ Stichproben, was für die $\boldsymbol{Q}$ Verteilungen insgesamt $\boldsymbol{Q} \times \boldsymbol{K}$ Datensätze $\boldsymbol{d}_{q,k}$ macht. Jede Methode $\boldsymbol{t}$ wird auf jeden Datensatz $\boldsymbol{d}_{q,k}$ ausgeführt, was insgesamt $\boldsymbol{T} \times \boldsymbol{Q} \times \boldsymbol{K}$ Schätzungen $\hat{\tilde{\boldsymbol{\tau}}}_{t,q,k}$ von dem synthetischen Behandlungseffekt erzeugt. Es wird dann der absoluten Fehler $\tilde{\boldsymbol{e}}_{t,q,k} = |\tilde{\boldsymbol{\tau}}_q - \hat{\tilde{\boldsymbol{\tau}}}_{t,q,k}|$ von jeder Schätzung ausgerechnet. Zum Schluss rechnet man den durchschnittlichen Fehler $\boldsymbol{e}_t$ für jede Methode und wählt die Methode mit dem kleinsten Fehler[SJT+17]:

$$\boldsymbol{t}^{\times} = \arg\min_{t \in T} \sum_{q,k}^{Q,K} \tilde{\boldsymbol{e}}_{t,q,k} \tag{3.12}$$

4 Implementierung

In diesem Abschnitt erzählen wir über unsere Implementierung. Wir erläutern zuerst, was wir genau programmiert haben, welche Technologien wir benutzt haben und warum und stellen einige allgemeinen Angaben bereit. Dann gehen wir konkret durch die einzelnen Bestandteile der Implementierung durch und erläutern, wozu sie dienen, welche Algorithmen und Techniken wir benutzt haben und wie sich die Theorie aus den vorigen Abschnitten in der Implementierung abbildet.

Wir haben Synth-Validation[SJT+17] implementiert - ein Auswahlverfahren von Methoden für kausale Inferenz, über den wir im vorigen *Abschnitt 3* erzählt haben. Auf den Code kann man unter `https://github.com/naskoD/bachelorThesis` frei zugreifen. Wir haben einen wesentlichen Teil des Codes von den Autoren von Synth-Validation bekommen. Er war auf Julia geschrieben und wir haben diesen Code direkt mit winzigen Logikänderungen in R übersetzt. In den Unterabschnitten werden wir für die einzelnen Bestandteile bezeichnen, ob sie übersetzt sind oder rein von uns kommen.

Die Implementierung hat sich als aufwändiger als erwartet erwiesen. Am Ende sind wir auf insgesamt 1700 Codezeilen ohne die Tests und 2250 mit den Tests gekommen. Sie sind in 14 Skripte ohne die Tests und 20 mit den Tests verteilt. Wir könnten wegen Zeitmangel und Streben nach Effizienz nicht alle Bestandteile mit Unittests überdecken. Wenn wir auch die Skripte von unseren Experimenten in die Rechnung hinzufügen, wo wir Synth-Validation mit unterschiedlichen Daten und unter unterschiedlichen Bedingungen benchmarken, kommen wir zu insgesamt 2400 Codezeilen in 26 Scripten.

Wir haben R für die Implementierung gewählt, weil sich diese Skriptsprache als Standard für statistikbezogene Projekte durchgesetzt hat. Es gibt eine große Anzahl von fertigen Bibliotheken, die zur Verfügung stehen. Außerdem hat sie eine intuitive Syntax, die eine schnelle Implementierung erlaubt. Die Sprache wird von vielen Wissenschaftlern und Praktikern benutzt und in den seltenen Fällen, in denen eine Bibliothek nicht ausreichend oder gar nicht dokumentiert ist, kann man schnell Unterstützung im Netz finden. R hat natürlich Nachteile - z.B die niedrigere Geschwindigkeit im Vergleich zu anderen Alternativen wie Python oder Julia. Mehr darüber erzählen wir im *Abschnitt 5.1* , wo wir unter anderem auch Daten über die Performanz von der Implementierung geben.

4.1 Externe Pakete

In diesem Abschnitt erzählen wir über die externen Pakete, die wir in unserer Implementierung benutzt haben. Wir werden sie in der Reihe des Auftretens im Programm vorstellen.

Das erste Paket ist **tictoc**. Es dient zur Zeitbemessung bei der Ausführung von einem Skript

oder einem Codeteil. Da ein Lauf von Synth-Validation sogar mit Standardparametern relativ lange dauert, ist es sinnvoll zu wissen, wie viel Zeit jeder einzelne Bestandteil braucht. Das gilt sowohl für die Zeit während des Laufs, als auch für die Analyse nach dem Ende. Dementsprechend ist ein Code aus dem Paket an vielen Stellen in unserer Implementierung vertreten.

Ein weiteres Paket, dessen Code man oft in unserer Implementierung trifft, ist **assertthat**. Es dient zur Validierung von Argumenten und ist in fast jeder Funktion unseres Codes zu finden.

Aus dem Paket **psych** benutzen wir einmalig die logistische Funktion.

An mehreren Stellen (z.B intern in unseren Datenstrukturen und in der Logik von Constrained Boosting) benutzen wir Dictionaries, die wir aus dem Packet **collections** nehmen. Meistens hat die Dictionary zwei Schlüssel für behandelt und unbehandelt und die Werte sind Arrays mit unterschiedlichen Daten.

Wir nutzen einmalig die Funktion *fold* aus dem Paket **mltools** bei der Cross-Validation, die die optimale Anzahl von Schritten für Constrained Boosting feststellt. *fold* teilt den Datensatz in Training- und Testdata nach bestimmten Kriterien. Ansonsten enthält das Paket Hilfefunktionen, die man in Techniken aus dem maschinellen Lernen benutzt.

Durch Funktionen aus dem Paket **NlcOptim** kann man Optimierungsproblemen lösen. Wir brauchen es bei der Bestimmung von den Koeffizienten $\boldsymbol{v_{0_m}}$ und $\boldsymbol{v_{1_m}}$ für die geschätzten Basisfunktionen in Constrained Boosting.

Das Paket **devtools** dient zur Arbeit mit Paketen z.B. Entwicklung, Installierung, usw. Wir benutzen es, um das Paket **rlearner** direkt aus Github zu installieren.

Mit **gridExtra** kann man Grids mit Daten schön stilisieren und dann generieren. Wir benutzen es einmal bei der Erstellung von Abbildungen für die Erfolgsrate von Synth-Validation.

Mit dem Paket **testthat** erstellt man Unittests. Wir haben sechs Testskripte, die es benutzen.

Vielleicht die wichtigsten externen Paketen, die wir nutzen, sind **rlearner** und **grf**. Sie beinhalten die Schätzer von den Methoden für kausale Inferenz, von denen Synth-Validation wählt. Wir haben die Theorie dahinter im *Abschnitt 2.2* erläutert. **rlearner** hat den Quasi-Orakel Schätzer mit Gradient Boosting und Lasso und **grf** hat die kausale Wälder. Wir benutzen die Funktionen aus diesen Paketen in unseren Wrapper-Funktionen, die die getesteten Methoden für kausale Inferenz darstellen.

4.2 Lesen/Schreiben von Daten

Im Script **read_write_data.r** haben wir Funktionen implementiert, die sich um das Lesen und Schreiben von Daten in Dateien kümmern. Wir haben sie selbst entwickelt und nicht aus

dem Juliacode übersetzt. Grundsätzlich lesen wir unsere Daten aus .csv Dateien und schreiben dann die Ergebnisse aus dem Benchmark wieder in .csv Dateien. Ab dem Moment, in dem wir angefangen haben, Analyseszenarien für die Evaluation auszudenken, haben wir festgestellt, dass es sehr nützlich wäre, wenn wir eine kleine Datenbank eingesetzt hätten. Die Zeit war aber für solche Änderungen knapp, deswegen sind wir bei der einfachen .csv Abspeicherung geblieben.

Wir haben unterschiedliche Datenquellen, deren Daten ein unterschiedliches Format hatten, deswegen war es die erste Aufgabe, die Daten zu normalisieren. Dann war der echte Behandlungseffekt von einigen Datensätzen nicht direkt gegeben und man sollte ihn aus den echten bedingten Erwartungswerten ausrechnen und in einer Datei abspeichern. Das waren die beiden Vorbereitungsschritten, die wir durchgeführt haben und deren Code im Lauf von Synth-Validation nicht läuft. Dann haben wir Funktionen für das Lesen von den Eingabedaten (Kovariaten, Behandlung, Ergebnisvariable und echten Behandlungseffekt). Weitere Funktionen dienen zum Lesen und Schreiben von Benchmarkdaten. Wenn es einen neuen Dateneintrag gibt, lesen wir zuerst die vorhandenen Daten, fügen ihn dazu ein und speichern alles zusammen. Wir haben eine Funktion, die den normalen Lauf des Programms sichert, indem sie die Verzeichnisstruktur der Benchmarkdaten neu erstellt, wenn das Programm zum ersten Mal läuft oder wenn man sie gelöscht hat.

4.3 Ziehen von Stichproben

Das Ziehen von Stichproben spielt eine sehr wichtige Rolle in unserer Implementierung. Aus den Daten, die wir gelesen haben, nehmen wir für die Schätzung nicht alle, sondern nur einen Teil davon (Standardwert ist 100). So simulieren wir eine echte Beobachtungsstudie, in der man nicht alle Objekte aus der Grundgesamtheit beobachtet, weil das oft unmöglich ist. Wir kennen den durchschnittlichen Behandlungseffekt von allen Beobachtungen und analysieren nur einen Teil davon. So stellen wir die ganze Datenmenge als die Grundgesamtheit dar, obwohl sie keine echte Grundgesamtheit ist, über die wir den echten Behandlungseffekt kennen. In einem Experiment im *Abschnitt 5* bemessen wir die Auswirkung von der Anzahl der analysierten Objekten auf die Schätzung von Synth-Validation.

Wir ziehen Stichproben an noch einer Stelle in der Implementierung und zwar bei der Generierung von synthetischen Daten. Nachdem wir die bedingte Erwartungswerte durch Constrained Boosting geschätzt haben und die entsprechenden Restwerte ausgerechnet haben, ziehen wir $\boldsymbol{2k}$ Stichproben mit Zurücklegen mit der gleichen Anzahl von Elementen für die $\boldsymbol{k}$ synthetischen Datensätze. Es gibt also zwei unabhängige Stichproben für jeden Datensatz - die erste für die $(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{w}}_i)$, deren Werte wir dann in den geschätzten bedingten Erwartungswerten einsetzen und die zweite für die Restwerte, die wir dann mit den vorigen Ergebnissen addieren und die fertigen synthetischen $\tilde{\boldsymbol{y}}_i$ bekommen. Dieser Codebestandteil haben wir auch nicht aus den Juliacode übersetzt, sondern selbst implementiert. Die Logik befindet sich im Script **sampling.r**.

4.4 Methoden für kausale Inferenz

In unserer Implementierung wählt Synth-Validation aus vier Methoden für kausale Inferenz - die naive Methode, Quasi-Orakel Schätzer mit Gradient Boosting, Quasi-Orakel Schätzer mit Lasso und Kausale Wälder. Wir haben sie außer der naiven Methode nicht selber implementiert, sondern aus den Paketen **rlearner** und **grf** genommen. Wie schon mal im *Abschnitt 2.2* angesprochen, schätzen die drei Methoden den heterogenen Behandlungseffekt. Deswegen haben wir in **causal_inference_methods.r** für jede Methode eine Wrapperfunktion geschrieben, in der wir den durchschnittlichen Behandlungseffekt aus allen heterogenen Schätzungen ausrechnen. Wir führen die Methoden an zwei Stellen in der Implementierung aus - einmal am Anfang auf den echten Daten, um die synthetischen Effekten zu bestimmen und einmal auf jeden generierten synthetischen Datensatz.

4.5 Synth-Validation

Alle in diesem Abschnitt beschriebenen Codebestandteile (ohne diese im Unterabschnitt Methodenauswahl *haben wir nicht selber entworfen, sondern aus dem Juliacode von den Autoren in R übersetzt.*

4.5.1 Datenstrukturen

Wir könnten für unsere Implementierung nur die einfachen Listen und Vektoren benutzen. So eine Entscheidung würde aber den Code unnötig kompliziert und unleserlich machen. Deswegen haben wir die Datenstrukturen genommen, die im Juliacode waren. Das sind nämlich die Klassen **Data**, **Counterfactuals**, **TreatmentDictionary**, **NumericTreatmentDictionary** und **Synth_Validation_Result**, die sich im Script **data_structures.r** befinden.

Die Klasse **Data** enthält eine Matrix $\boldsymbol{X}$ für die Kovariaten, einen logischen Vektor $\boldsymbol{W}$ für die Behandlung und einen Vektor $\boldsymbol{Y}$ für die Ergebnisvariable. Wir benutzen die Klasse sowohl für die echten, als auch für die synthetischen Daten.

Ein Objekt aus der Klasse **Counterfactuals** ist das Ergebnis aus einem Schritt von Constrained Boosting. Es enthält die bedingten Erwartungswerte für alle Elemente aus **Data**.

TreatmentDictionary hat für die beiden Schlüssel *behandelt* und *unbehandelt* einen Vektor mit irgendwelchen Elementen. Wir benutzen es bei der Schätzung, wo die Elemente z.B. angepasste Regressionsbäume sind.

NumericTreatmentDictionary ist wie **TreatmentDictionary**, aber seine Elementen dürfen nur numerisch sein. Wir benutzen sie an mehreren Stellen. Ein Objekt aus dieser Klasse ist z.B. das Ergebnis von den Schätzungen der Basisfunktionen bei Constrained Boosting.

Die letzte datenbezogene Klasse ist **Synth_Validation_Result** und sie trägt das Ergebnis von Synth-Validation. Wir haben sie selbst entwickelt. Die Klasse enthält drei Felder - den Index der ausgewählten Methode, die Schätzung dieser Methode und einen Vektor mit den Schätzungen von allen Methoden. Wir benutzen sie beim Benchmark.

4.5.2 Schätzung

Im Skript **estimators.r** haben wir den Regressionsbaum implementiert, den wir im Juliacode gefunden haben. Er schätzt die Basisfunktionen im Constrained Boosting, wobei er die Restwerte von $\boldsymbol{y_i}$ und der vorigen Schätzung nimmt. Wir setzen die minimale Anzahl von Elementen im Blatt auf fünf und die maximale Tiefe des Baums auf drei. Hier haben wir eine winzige Änderung in der Logik des Baums im Vergleich zu dieser im Juliacode - wir haben einen Sonderfall beim Aufspalten behoben, indem wir nicht aufspalten, wenn das maximale MSE null ist und alle $\boldsymbol{Y}$ gleich sind.

4.5.3 Constrained Boosting

Den Constrained Gradient Boosting Algorithmus und die Cross-Validation dazu implementieren wir im Script **constrained_boosting.r**. Grundsätzlich machen wir das, was wir im *Abschnitt 3.1.2* theoretisch beschrieben haben. Im ersten Schritt finden wir die beiden Konstanten für die bedingten Erwartungswerte und berechnen die Restwerte für jedes Element. Dann fängt eine Schleife an, wobei jeder Lauf ein weiter Schritt aus dem Constrained Boosting ist. Im jedem Lauf passen wir zuerst den Regressionbaum für die Schätzung der Basisfunktionen an. Dann erstellen wir eine Dictionaty, die als Schlüssel die einzigartigen Schätzungen des Baums hat. Die Werte für die Schlüssel sind dann die Indizes von den Elementen, die diese Schätzung haben. Wir nehmen dann diese Dictionary und benutzen sie in unserem Optimierungsproblem, indem wir den allgemeinen Verlust als die Summe von den Verlusten von allen einzigartigen Schätzungen und den Verlust von einer einzigartigen Schätzung als die Summe von den Verlusten von den Elementen mit dieser Schätzung darstellen. Für das Optimierungsproblem benutzen wir das externe Paket **NlcOptim** und finden die Koeffizienten, für die unsere Bedingung hält. Dann addieren wir die Basisfunktion mit ihren Koeffizienten zu der alten Schätzung und bekommen die neuen bedingten Erwartungswerten.

Bevor wir Constrained Boosting produktiv einsetzen, müssen wir die optimale Anzahl von Schritten bestimmen, sodass die MSE der Schätzung minimal ist. Dazu benutzen wir Cross-Validation. Wir teilen die Daten in Training- und Testdaten. Dann führen wir Constrained Boosting auf den Trainingdaten mit der Ausnahme aus, dass wir für die Bedingung über den synthetischen Effekt im Optimierungsproblem alle Daten benutzen. Dann messen wir den Schätzfehler in jedem Schritt mit den Testdaten. Wir wiederholen das vier mal und wählen als Anzahl die Nummer des Schritts mit dem kleinsten durchschnittlichen Testfehler für diese vier

Läufe.

4.5.4 Methodenauswahl

Die Methodenauswahl findet in der Methode *pick_best_method* im Skript **synth-validation.r** statt. Wir haben sie selber implementiert. Sie nimmt als Argumenten die synthetischen Datensätze und die synthetischen Effekte. Dann führt sie die Methoden für kausale Inferenz auf den Daten aus, bemisst ihren durchschnittlichen Fehler als absoluten Abstand der Schätzung von dem synthetischen Effekt und wählt die Methode mit dem kleinsten Fehler aus.

4.6 Benchmark von Synth-Validation

Durch den Benchmark prüfen wir, wie sich die Schätzungen von Synth-Validation im Vergleich zu den Schätzungen von den anderen Methoden verhalten. Das Maß, das wir nutzen, ist der absolute Fehler von dem echten durchschnittlichen Behandlungseffekt. Aus diesem Grund müssen wir ihn vorher kennen. Wir führen also mehrmals ($\boldsymbol{n > 30}$) die Synth-Validation mit gleichen Daten und gleichen Paramertern aus und vergleichen ihren durchschnittlichen Fehler mit den durchschnittlichen Fehlern der Methoden. Noch beobachten wir nebenbei, wie oft eine Methode die Orakel-Schätzung (die beste Schätzung unter allen Methoden) hat und wie oft sie von Synth-Validation ausgewählt wird. Die Orakel-Schätzung kann für einen bestimmten Datensatz unterschiedlich sein, da wir bei jedem Lauf von Synth-Validation immer eine neue Stichprobe aus diesem gleichen, viel größeren Datensatz (unsere Grundgesamtheit) ziehen. Wir beobachten noch, wie erfolgreich Synth-Validation auswählt. Das machen wir im Sinne davon, wie oft die Schätzung von Synth-Validation mit der besten Schätzung (Orakel-Schätzung), mit der zweitbesten usw übereinstimmt. Diese Analysen führen wir auf unterschiedlichen Ebenen - für einen bestimmten Datensatz, für alle Datensätze aus einer Art und für insgesamt alle Datensätze.

Im Skript **synth_validation_benchmark.r** haben wir die Struktur von dem Benchmark aufgebaut - wir rufen Funktionen aus anderen Skripten, die die Arbeit erledigen. Da berechnen wir natürlich den absoluten Fehler der Schätzungen aus einem Lauf und leiten ihn zum Speichern und zum Erstellen von Abbildungen weiter. Im Skript **benchmark_analysis.r** machen wir die Berechnungen für die im oberen Absatz beschriebenen Analysen. Durch das Skript**hypothesis_testing.r** prüfen wir einige Hypothesen über Synth-Validation.

4.7 Erstellung von Abbildungen

Wir nehmen die Daten aus den Skripten **benchmark_analysis.r** und **synth_validation_benchmark.r** und erstellen damit im Skript **plots.r** Abbildungen für unsere

Analyse. Es gibt drei Arten, die wir generieren - ein Boxplot, der den Schätzfehler von Synth-Validation und allen Methoden zeigt, ein Balkendiagramm, das die Häufigkeit von den Auswahl jeder Methode aus dem Orakel und aus Synth-Validation zeigt und eine Tabelle, die die kumulative Erfolgsrate von Synth-Validation zeigt. Dazu benutzen wir keine externen Pakete, sondern nur die Standardfunktionen in R. Die Abbildungen generieren sich bei jedem Benchmark. Man kann sie nur mit den verfügbaren Benchmarkdaten generieren, ohne Synth-Validation erneut zu benchmarken, indem man den Skript **regenerate_plots.r** ausführt. Man kann die drei Abbildungsarten im *Abschnitt 5* oder im *Anhang* anschauen.

4.8 Experimenten

In unserer Evaluation wollen wir unterschiedliche Szenarien testen - wie stellt sich Synth-Validation mit unterschiedlichen Daten, mit einer unterschiedlichen Größe der gezogenen Stichprobe und mit einer unterschiedlichen Anzahl von der maximalen Schritten bei Constrained Boosting vor. Die sechs Skripte der Experimente **experiments*.r** enthalten ausschließlich Aufrufe von der Funktion *benchmark* aus dem Skript **synth_validation_benchmark.r** mit unterschiedlichen Konfigurationsargumenten. Wir stellen die Ergebnisse aus den Experimenten im *Abschnitt 5* dar.

4.9 Unittests

Wir haben Unittests für die Funktionen aus den Skripten von Synth-Validation geschrieben. Dafür haben wir das externe Paket **testthat** benutzt. Die Unittests haben keinen besonders großen Nutzen gebracht, aber ihre Entwicklung und vor allem Wartung kosteten Zeit. Und da wir mit unserer Arbeit Effizienz anstreben und die Zeit knapp war, haben wir für den Rest der Implementierung nur manuell getestet.

4.10 Anderer Code

Wir haben das Skript **inst all_required_packages.r** geschrieben, das in jedem Experiment als Erstes gerufen wird. Er prüft, ob die notwendigen externen Pakete installiert sind und wenn nicht, installiert er sie. Im Skript **utilities.r** haben wir einige allgemein nützliche Funktionen implementiert, die wir an vielen Stellen in der Implementierung nutzen. Eine davon ist z.B. *assert_integer*.

5 Evaluation

In diesem Abschnitt stellen wir die Ergebnisse von dem Benchmark von Synth-Validation dar. Wir erläutern zuerst die Methodik, die wir für unsere Benchmarkexperimente benutzt haben. Wir erzählen über die Herkunft der Daten, deren kausalen Effekten wir schätzen. Wir sagen ein paar Worte über die Performanz der Software, die wir geschrieben haben. Dann stellen wir die Ergebnisse dar, die wir bekommen haben und evaluieren sie.

5.1 Methodik und Daten

Im *Abschnitt 1.4* haben wir die Ziele von dieser Bachelorarbeit definiert und eins davon war der Benchmark von Synth-Validation mit echten Daten. Das Finden von solchen Daten war eine große Herausforderung, denn der durchschnittliche Behandlungseffekt von ihnen müsste mit einer großen Sicherheit bekannt sein. Nur auf dieser Weise können wir die Schätzfähigkeit von Synth-Validation mit diesen von den anderen Methoden direkt vergleichen. Unser Hauptvergleichskriterium ist der durchschnittliche absolute Fehler $\boldsymbol{e_t}$ von allen Schätzungen einer Methode $\boldsymbol{t}$. Formal berechnen wir ihn folgendermaßen:

$$\boldsymbol{e_t} = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i}}^{\boldsymbol{n}} \left| \hat{\boldsymbol{\tau}}_{\boldsymbol{t_i}} - \boldsymbol{\tau} \right| \tag{5.1}$$

Hier ist $\hat{\boldsymbol{\tau}}_{\boldsymbol{t_i}}$ eine Schätzung von dem durchschnittlichen Behandlungseffekt von der Methode $\boldsymbol{t}$ und $\boldsymbol{\tau}$ ist der echte durchschnittliche Behandlungseffekt.

Wir haben drei Arten von Daten, auf den wir Synth-Validation testen. Die ersten sind Rohdaten. Sie sind durch eine echte Beobachtungsstudie gesammelt und wurden nicht verarbeitet. Die zweiten sind synthetisch generierte Daten, für deren Generierung aber echte Daten benutzt wurden. Die dritten sind die bekannten maschinell zufällig generierten Daten mit einem vorgegebenen Behandlungseffekt. Und soweit man die letzte Art mit ein paar Codezeilen zur Verfügung hat, ist das Finden von den anderen nicht trivial.

Die Daten kommen aus Wettbewerben für kausale Inferenz. Da stellt man für die Vorbereitung immer einen Testdatensatz zur Verfügung vor, für den der durchschnittliche Behandlungseffekt bekannt ist. Die Rohdaten kommen aus dem *ACIC 2018 Causal Inference Challenge* und sind unter `https://www.synapse.org/#!Synapse:syn11294478/wiki/494269` erreichbar. Es ist bekannt, aus welchem Bereich sie kommen - die amerikanische Geburtenstatistik. Die Beschreibung der Kovariate haben wir auf der Webseite der echten Datenquelle (der amerikanische National Center for Health Statistics) gefunden. In der Beschreibung von dem Wettbewerb wird aber nicht explizit vorgegeben, was hinter der Behandlung und der Ergebnisvariable der Daten steht. Also man weiß es nicht, welcher Effekt geschätzt wird. Die Situation bei den synthetisch

generierten Daten ist gleich - es ist unbekannt, welche Bedeutung die Behandlugs- und die Ergebnisvariable haben. Ansonsten kommen sie aus dem *ACIC 2019 Causal Inference Challenge* und sind unter `https://sites.google.com/view/ACIC2019DataChallenge/data-challenge` erreichbar. Die Autoren des Wettbewerbs geben die Rohdaten, die sie für die Generierung benutzt haben. Sie stammen aus mehreren Bereichen - Finanzwesen, Sozialwissenschaften, Medizin.

Die Daten waren in unterschiedlichen Formaten, deswegen haben wir sie normalisiert. Das Ergebnis ist eine Datei mit Kovariaten, Behandlung und Ergebnisvariable und eine Datei mit dem echten Behandlungseffekt für jeden Datensatz. Insgesamt haben wir sieben Rohdatensätze und drei synthetische Datensätze mit unterschiedlichen Behandlungseffekten. Auf sie, auf den Code und auf die im nächsten Abschnitt vorgestellten Ergebnisse kann man in `https://github.com/naskoD/bachelorThesis` frei zugreifen. Die zufälligen Daten mit vorgegebenem Behandlungseffekt generieren wir immer neu im Programm. Ihr Behandlungseffekt ist immer 0.

Wir haben Ideen über den Benchmark- und Evaluationprozess aus dem Synth-Validation Artikel [SJT+17] gesammelt. Da wird der absolute durchschnittliche Fehler der Methoden als Maß benutzt. Es wird noch beobachtet, wie oft eine Methode von Synth-Validation und von dem Orakel gewählt wird. Die beiden Analysen sind bei uns auch vertreten. Wir fügen noch die kumulierte Erfolgsrate von Synth-Validation hinzu - wie oft wählt sie eine von den ersten $\boldsymbol{k}$ besten Schätzungen aus. Am interessantesten ist natürlich, wie oft sie die beste (die Orakel Schätzung) auswählt. Weiter haben wir Hypothesentests für die Signifikanz von unseren Ergebnissen in der Evaluation erstellt.

Die oberen Analysen führen wir unter unterschiedlichen Szenarien durch - wir schauen uns zuerst die Ergebnisse von Synth-Validation mit allen Datensätzen zusammen an. Dann betrachten wir getrennt die unterschiedlichen Arten. Wir haben noch weitere Experimente durchgeführt, deren Ergebnisse wir hier wegen der Zeitbeschränkung nicht ausführlich evaluieren können. Die erstellten Graphiken für jeden individuellen Datensatz kann man im *Anhang* anschauen.

Wir haben in der Praxis gesehen, was wir noch vorher wussten - R ist nicht besonders performant. Um die Ergebnisse zu bekommen, haben wir Synth-Validation auf Servern laufen gelassen, die uns von der Uni zur Verfügung vorgegeben wurden. Ein einziger Lauf von Synth-Validation mit Standardparameter (100 Elementen in der Stichprobe und 100 maximale Schritte beim Constrained Boosting) dauert abhängig von dem Datensatz durchschnittlich etwa länger als eine Stunde. Für die Ergebnisse, die wir hier darstellen, waren 373 Läufe nötig, ohne die Testläufe während der Implementierung und die Läufe aus den nicht vorgestellten Experimenten einzubeziehen.

5.2 Ergebnisse

Im Folgenden werden wir die Ergebnisse von dem Benchmark von Synth-Validation vorstellen. Wir haben insgesamt elf Datensätze - sieben Rohdatensätze, drei synthetisch generierte Datensätze, die aus Rohdaten stammen und einen zufällig generierten Datensatz. Mit jedem davon haben wir Synth-Validation mindestens 30 Mal laufen gelassen. Die Ergebnisse von jeder Datenart werden hier vorgestellt und analysiert. Die Ergebnisse und Graphiken von jedem der elf individuellen Datensätze kann man sich im Anhang *anschauen.*

5.2.1 Alle Daten

Wir stellen zuerst die Ergebnisse aus allen Läufen von Synth-Validation zusammen dar. Insgesamt sind wir auf 373 Benchmarkläufe gekommen. Wir haben unsere Standardparameter benutzt - Stichprobe aus 100 gezogenen Objekten und maximal 100 Schritte beim Constrained Boosting. Auf *Abbildung 3* sieht man den durchschnittlichen absoluten Schätzfehler (MAE) der Methoden, des Orakels und der Synth-Validation:

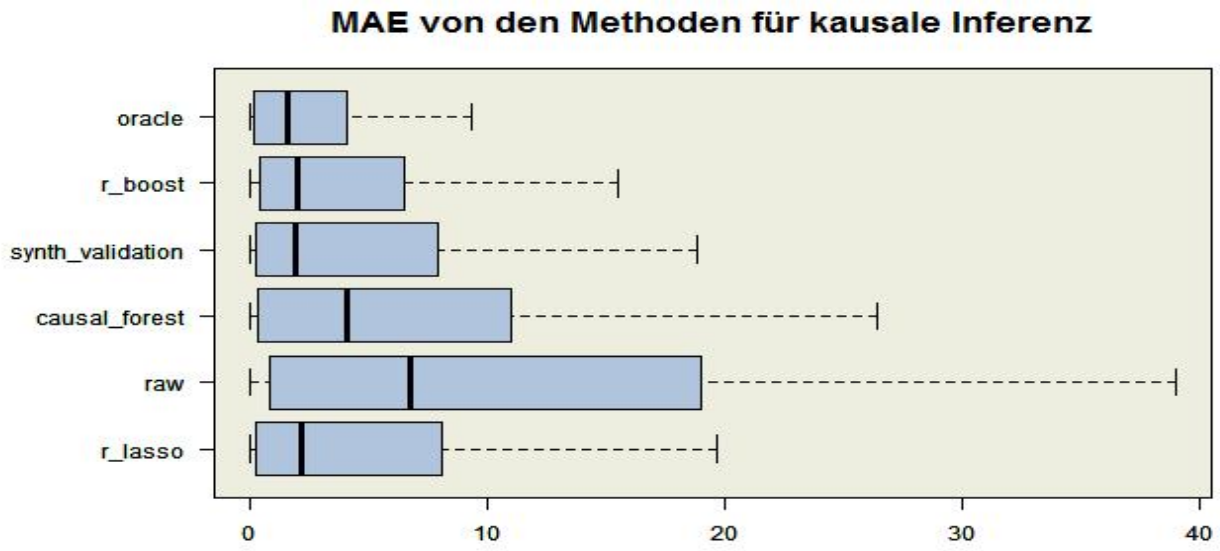

Abbildung 3: MAE von den Methoden für kausale Inferenz und Synth-Validation für alle Daten. Je kleiner der durchschnittliche Fehler der Methode ist, desto besser. Der Orakel wählt immer die beste Schätzung aus, ist aber kein realer Schätzer. Die Methoden sind aufsteigend von oben nach unter geordnet. Die Boxplots zeigen die Variaz der Schätzungen. MAEs auf die vierte Stelle gerundet: oracle - **2.4167**, r_boost - **3.7281**, synth_validation - **25.4427**, causal_forest - **26.8746**, raw - **33.0900**, r_lasso - **45.0627**

Das erste, was man auf *Abbildung 3* beobachtet, ist die große Varianz des Schätzfehlers bei

allen Methoden. Das ist ein Zeichen dafür, dass es neben den vielen guten Schätzungen auch sehr viele schlechte und sogar einige sehr schlechte gibt. Die gesamten Ergebnisse sind stark negativ von den Ergebnissen der Rohdaten bestreut. Es gibt zwei Gründe - sie nehmen den größten Anteil (243 von 373) der Läufen und sie haben den größten durchschnittlichen Fehler. Bei einer Rohdatensatz mit einer größeren Grundgesamtheit (50000 Elementen), aus dem wir eine Stichprobe von nur 100 Elementen analysieren, beobachten wir einen durchschnittlichen Fehler von über 100 bei allen Methoden außer r_boost, was auch die allgemeinen Ergebnisse stark bestreut. Es ist auch auf dem ersten Blick erstaunlich, dass r_lasso durchschnittlich schlechter als die naive Methode ist. Der Grund - auf einem der Rohdatensätze schätzt es mit einem durchschnittlichen Fehler von 347.9418. Insgesamt hat es öfter eine gute oder sogar beste Schätzung, was auf *Abbildung 4* zu sehen ist.

Im Allgemein lassen sich die Schätzungen als sehr schlecht beschreiben. Nur r_boost zeichnet sich mit einem deutlich niedrigeren durchschnittlichen Fehler im Vergleich zu allen anderen Methoden aus. Wir beobachten, dass das durchschnittliche Ergebnis von Synth-Validation vor allen anderen (außer r_boost) liegt. Ihre Varianz ist auch die zweitgrößte. Trotzdem ist das für uns nicht ausreichend, um zu behaupten, dass Synth-Validation im allgemeinen Fall durchschnittlich besser schätzt als die von ihr ausgewählten Methoden aus dem maschinellen Lernen. Der Grund dafür - unsere Signifikanztests mit der üblichen $\boldsymbol{\alpha = 0.05}$ sind gescheitert. Nur die p-Werte von dem Lasso und der naiven Methode waren nah - 0.2184 und 0.2009 und bei mehreren Läufen könnte vielleicht die Nullhypothese signifikant abgelehnt werden. Die Nullhypothese lautet in dem Fall, dass Synth-Validation nicht besser als die naive Methode und r_lasso schätzt. Wir haben wegen der ungleichen Varianz der Schätzungen einen Welch Test benutzt.

Diese schlechten Ergebnisse zeigen uns, dass die Methoden für kausale Inferenz im Allgemein sehr ungenau sein können. Deswegen muss man sowohl sie, als auch den Domain von dem Datensatz sehr gut kennen. Man muss auch eine angemessen große Stichprobe für die Grundgesamtheit ziehen. Auf allgemeinen Datenarten und mit den neuen Methoden aus dem maschinellen Lernen schafft Synth-Validation nicht, die beste durchschnittliche Schätzung zu haben.

Auf *Abbildung 4* beobachten wir, wie oft jede Methode von dem Orakel und von Synth-Validation ausgewählt ist. Es bestätigt sich die These, dass keine allgemein beste Methode für alle Datensätze gibt, da jede Methode (sogar die naive in über 10% der Läufe) von dem Orakel ausgewählt wurde. Die drei Methoden mit Schätzern aus dem maschinellen Lernen sind natürlich besser als die naive vertreten, indem r_boost im Einklang mit dem durchschnittlichen Fehler aus *Abbildung 3* an der ersten Stelle steht. Für das Lasso sehen wir, dass es in rund 25% der Läufe die beste Schätzung hat, obwohl es den höchsten durchschnittlichen Fehler trägt, was seine hohe Varianz zeigt.

Keine Methode wird von Synth-Validation völlig ausgeschlossen (nie ausgewählt). Man be-

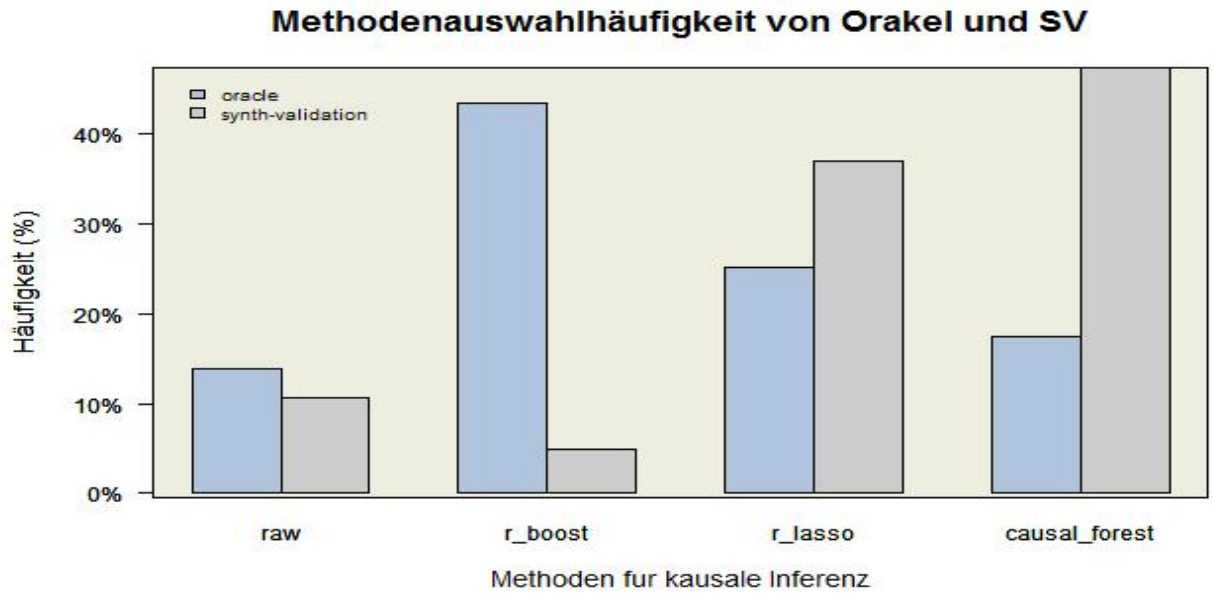

Abbildung 4: Methodenauswahlhäufigkeit von Orakel und Synth-Validation für alle Daten. Das Balkendiagramm zeigt für jede getestete Methode für kausale Inferenz, wie oft sie die tatsächliche beste Schätzung hatte und wie oft sie von Synth-Validation ausgewählt ist. Wenn die beiden Auswahlhäufigkeiten für eine Methode relativ nah aneinander liegen oder sogar gleich sind, bedeutet es nicht unbedingt, dass jede Auswahl von Synth-Validation mit dieser von dem Orakel übereinstimmt.

trachtet, dass die kausale Wälder am häufigsten ausgewählt wurden und gleich danach folgt das Lasso. Leider ist die am häufigsten vom Orakel ausgewählte Methode r_boost am seltensten von Synth-Validation ausgewählt, was schlecht für die Auswahlfähigkeiten von Synth-Validation im allgemeinen Fall spricht.

Abbildung 5 ist eine Tabelle, die uns die kumulierte Erfolgsrate vom Synth-Validation beim Auswählen zeigt. In 96 von insgesamt 373 Läufen hat Synth-Validation eine gleiche Schätzung mit dem Orakel. Diese Erfolgsrate von 25.74% ist nur mit 0.74% höher als die erwartete Erfolgsrate, wenn man die beste Methode zufällig ausgewählt hätte. Wenn wir das Ziel von Synth-Validation redefinieren - nicht unbedingt die beste Methode, sondern eine unter den zwei besten auswählen - dann stellt sich Synth-Validation etwa besser mit Erfolgsrate von 59.79% gegen 50% im Vergleich zu der zufälligen Auswahl vor. Bei den $\boldsymbol{N = 3}$ kumulierten besten Schätzungen liegt der Vorteil am höchsten. Wir haben die Signifikanz von diesen Behauptungen mit der Hilfe von der Binomialverteilung für die übliche $\boldsymbol{\alpha = 0.05}$ geprüft und nur bei der Auswahl von den ersten zwei und drei kumulierten Schätzungen ist Synth-Validation signifikant besser als der Zufall.

Wenn also unser Ziel ist, die beste Methode zur Schätzung von dem durchschnittlichen Behandlungseffekt auszuwählen, wenn wir wenige/keine Kenntnisse über die Daten und die möglichen Schätzmethoden haben, dann ist der Einsatz von Synth-Validation im allgemeinen Fall nicht

N-te beste Schätzung	# ausgewählt	# ausgewählt (kumuliert)	Kumulierte Erfolgsrate
1	96	96	25.74%
2	127	223	59.79%
3	117	340	91.15%
4	33	373	100%

Abbildung 5: Kumulierte Erfolgsrate von Synth-Validation für alle Daten. Die Tabelle zeigt, wie oft Synth-Validation die Methode mit der besten (Orakel), die zweitbesten usw. Schätzung ausgewählt hat. In der dritten Spalte sehen wir die Anzahl der kumulierten ausgewählten Schätzungen (Spalte 2 kumuliert). Die letzte Spalte enthält die kumulierte Erfolgsrate von Synth-Validation - Spalte 3 durch die Anzahl von allen Läufen (373).

besser als die zufällige Auswahl. Natürlich ist die Annahme über die wenige Kenntnisse sehr oft falsch, was die Baseline höher als die zufällige Auswahl aufhebt, was noch schlechter über die Ergebnisse von Synth-Validation im allgemeinen Fall spricht.

In den folgenden Abschnitten beobachten wir die Ergebnisse bei jeder Datenart individuell. Da sich die Art der Abbildungen wiederholt, werden wir sie nur mit Titeln beschriften.

5.2.2 Rohdaten

Wir haben sieben Rohdatensätze mit unterschiedlichen Größen und unterschiedlichen durchschnittlichen Behandlungseffekten. Drei von ihnen enthalten 1000 Elementen, andere zwei enthalten 25000 und die letzten zwei haben 50000. Die Datensätze haben eine hohe Anzahl von Kovariaten - 177. Wir haben Synth-Validation auf jedem Datensatz mindestens 30 Mal mit Standardparametern gebenchmarkt. Insgesamt sind wir auf 243 Läufe gekommen. Wir zeigen zuerst den durchschnittlichen absoluten Fehler auf *Abbildung 6*.

Wir beobachten, dass die durchschnittlichen absoluten Fehler bei den Rohdaten noch höher als im allgemeinen Fall sind. Nur r_boost hat einen etwa kleineren. Die Boxplots zeigen uns, dass die Varianz der Schätzungen insgesamt sehr hoch ist; bei einigen Methoden (causal_forest und raw) sogar extrem hoch. Die Mediane der Fehler von allen Methoden ist relativ niedrig, was bedeutet, dass die Hälfte der Schätzungen relativ gut sind. Die hohen Fehler im Durchschnitt sind vor allem auf die Datensätze mit mehreren Elementen zurückzuführen. Eine Stichprobe aus 100 Elementen kann nicht gut genug einen Datensatz mit 50000 Elementen vorstellen und das führt zu einer stark gestreuten Schätzung und zu einem hohen absoluten Fehler. Vor den Experimenten haben wir vermutetet, dass die kleine Stichprobe problematisch sein könnte. Eine höhere bei den größeren Datensätzen könnten wir uns aber nicht leisten, denn das würde die Ausführungszeit bedeutend erhöhen und wir waren unter Zeitdruck.

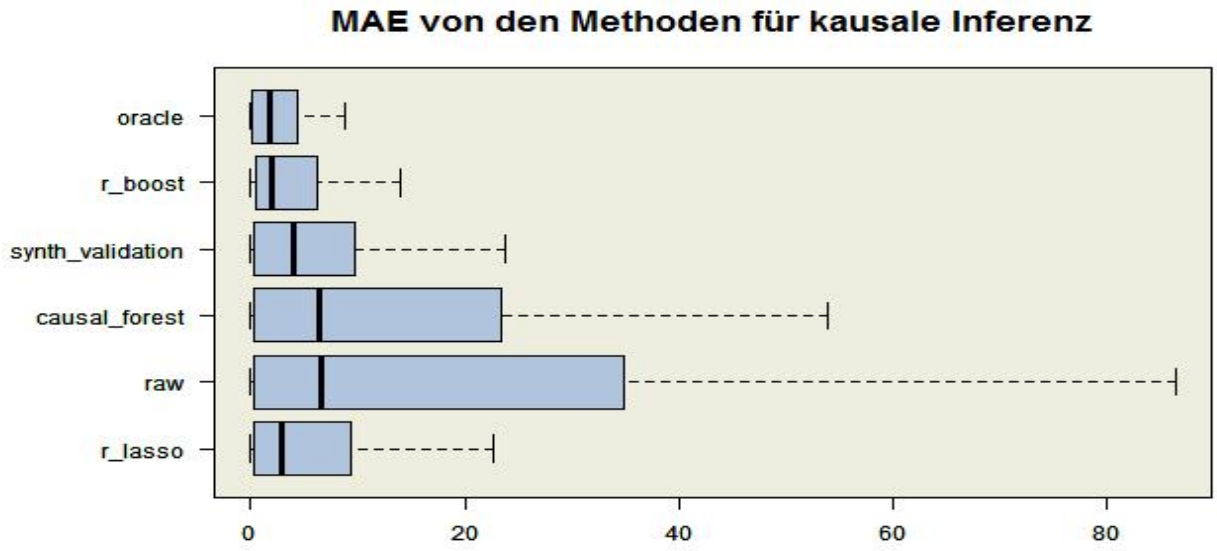

Abbildung 6: MAE von den Methoden für kausale Inferenz und Synth-Validation für Rohdaten. MAEs auf die vierte Stelle gerundet: oracle - **2.7420**, r_boost - **3.2994**, synth_validation - **37.0656**, causal_forest - **38.9371**, raw - **45.2470**, r_lasso - **66.9590**

Die Methoden sind genauso wie bei dem allgemeinen Fall geordnet. Synth-Validation hat immer noch nicht die beste Schätzung, nimmt aber den zweiten Platz nach r_boost. Man sieht, dass die Mediane von den Fehlern des Lassos sogar kleiner als diese von den durchschnittlich besseren naiven Methode und Kausalen Wäldern. Das zeigt, dass das Lasso eigentlich besser schätzt, soweit es keine extrem schlechten Schätzungen gibt, wie bei uns der Fall ist.

Obwohl Synth-Validation am zweiten Platz steht (den Orakel nehmen wir als keinen echten Schätzer), bedeutet es jedoch nicht, dass sie besser als alle darunter schätzt. Unsere Hypothesentests auf diesen Ergebnisse mit $\boldsymbol{\alpha = 0.05}$ zeigen, dass wir signifikant für keine Methode behaupten können, dass Synth-Validation durchschnittlich besser als sie schätzt. Wieder sind wir relativ nah mit dem p-Wert des Lassos von **0.2198**. Es lohnt sich zu nennen, dass wir ein Welch Test benutzen, da die Varianzen von den Methoden offensichtlich ungleich sind.

Auf *Abbildung 7* beobachten wir, wie oft der Orakel und Synth-Validation jede Methode auswählen. Es gibt keine, die vom Orakel nicht ausgewählt wurde, was noch mal bestätigt, dass es keine allgemeinbeste Methode gibt, sondern jede Methode in unterschiedlichen Konstellationen die beste sein kann. Es ist erstaunlich, dass die naive Methode öfter als die Kausale Wälder vom Orakel ausgewählt wird. Ansonsten führt r_boost deutlich, was mit seinem kleinsten durchschnittlichen Fehler übereinstimmt.

Wir betrachten, dass r_boost am seltensten von Synth-Validation ausgewählt wird, obwohl sie am häufigsten die besten Schätzungen hat. Genauso umgekehrt ist bei den Kausalen Wäl-

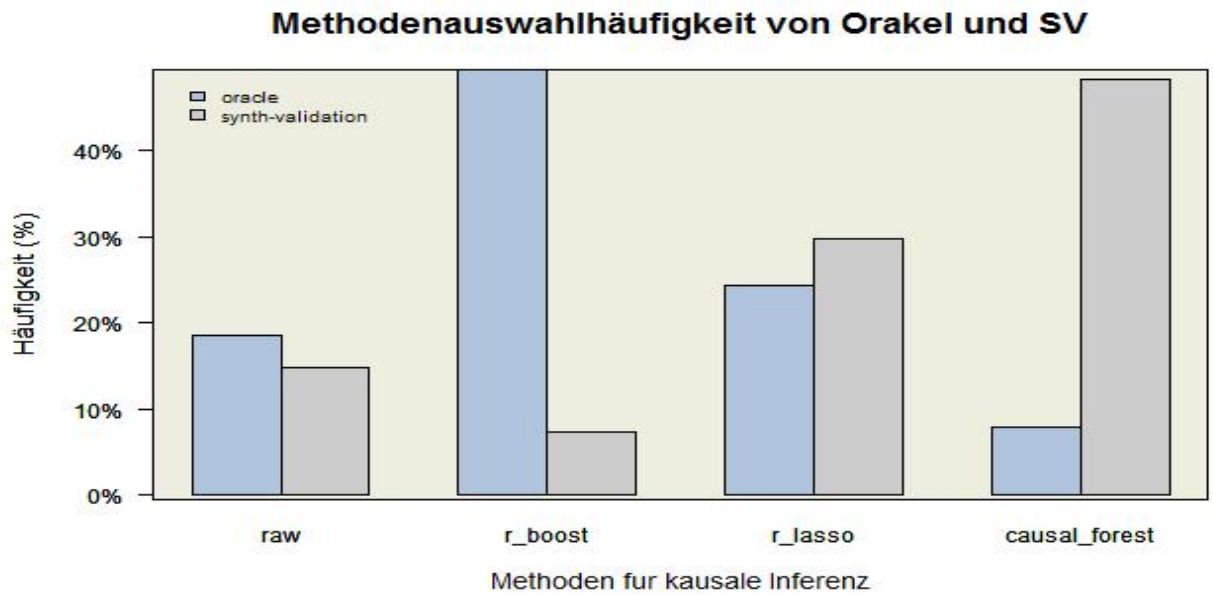

Abbildung 7: Methodenauswahlhäufigkeit von Orakel und Synth-Validation für Rohdaten

dern und das spricht schlecht über Synth-Validation. Bei raw und r_lasso haben wir näheren Anteilen, was aber nicht unbedingt bedeutet, dass die Methoden von dem Orakel und Synth-Validation im gleichen Lauf ausgewählt wurden.

Am Ende des Abschnitts beobachten wir auf *Abbildung 8*, wie oft Synth-Validation eine unter den $\boldsymbol{N}$ besten Schätzungen auswählt. Für Rohdaten ist ihre Erfolgsrate darüber, die Orakelschätzung auszuwählen, noch niedriger als im allgemeinen Fall und liegt bei 16.46%. Das ist deutlich schlechter als die erwarteten 25%, die man bei einer zufälligen Auswahl erreicht. Für $\boldsymbol{N = 2}$ und $\boldsymbol{N = 3}$ liegen die Erfolgsraten etwa besser als die zufälligen mit entsprechend 3.5% und 12.65%. Die durchgeführten Signifikanztests mithilfe der Binominalverteilung bestätigen nur die bessere Auswahl von Synth-Validation von einer der ersten drei besten Schätzungen im Vergleich zur zufälligen Auswahl. Dabei benutzen wir $\boldsymbol{\alpha = 0.05}$.

N-te beste Schätzung	# ausgewählt	# ausgewählt (kumuliert)	Kumulierte Erfolgsrate
1	40	40	16.46%
2	90	130	53.5%
3	83	213	87.65%
4	30	243	100%

Abbildung 8: Kumulierte Erfolgsrate von Synth-Validation für Rohdaten

5.2.3 Synthetisch generierte Daten aus Rohdaten

Wir verfügen über drei synthetisch generierte Datensätze. Zwei davon haben eine Größe von 2000 Elementen und enthalten 185 Kovariaten. Der dritte hat 500 Elementen und 22 Kovariaten. Auf jeden Datensatz haben wir Synth-Validation mindestens 30 Mal ausgeführt und insgesamt gibt es 99 Läufe. Wir fangen mit der Analyse von den durchschnittlichen absoluten Fehlern der Methoden auf *Abbildung 9* an.

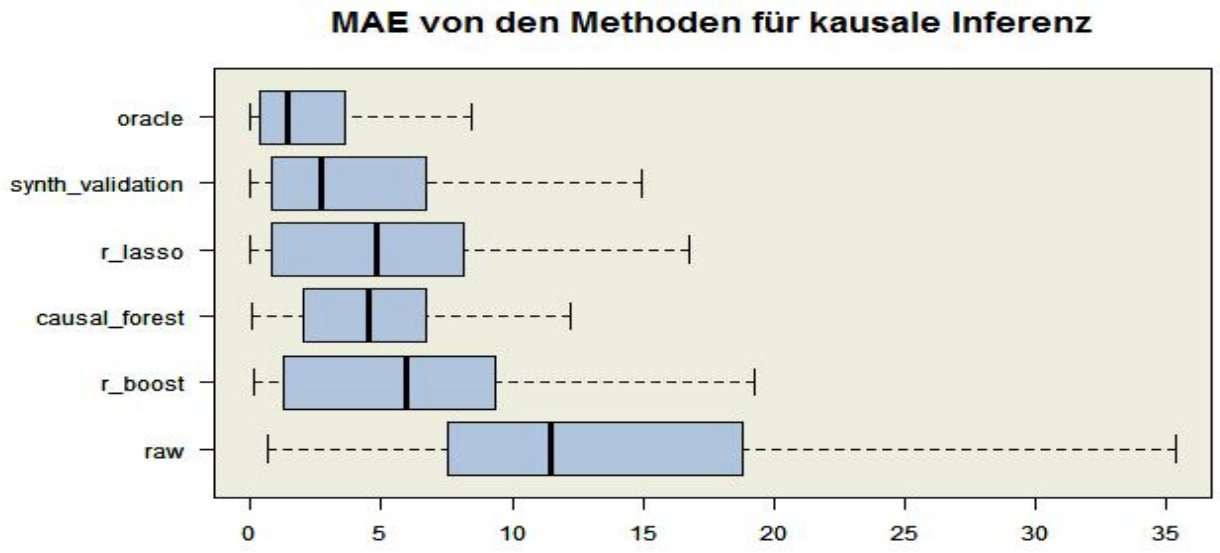

Abbildung 9: MAE von den Methoden für kausale Inferenz und Synth-Validation für synthetisch generiete Daten. MAEs auf die vierte Stelle gerundet: oracle - **2.3671**, synth_validation - **4.8606**, r_lasso - **5.4032**, causal_forest - **5.6650**, r_boost - **5.9377**, raw - **13.3760**

Man stellt zuerst fest, dass die durchschnittlichen absoluten Fehler deutlich niedriger als bei den Rohdaten liegen. Wir haben auch eine andere Ordnung von den Schätzungen. Synth-Validation hat den niedrigsten Fehler und steht deswegen auf Platz 1 von den realen Schätzer. Das Lasso ist an der zweiten Stelle, obwohl es bei den Rohdaten an der letzten war. Diese Beobachtungen bestätigen nochmal, dass die unterschiedlichen Methoden bei unterschiedlichen Datensätzen unterschiedlich gute Schätzungen geben.

Der erste Platz von Synth-Validation ist ein gutes Zeichen dafür, dass sie besser als die anderen Methoden den durchschnittlichen Behandlungseffekt schätzen könnte, indem sie für synthetisch generierten Daten öfter die beste Methode auswählt. Das prüfen wir durch einen Welch Test für jede Methode. Unsere Nullhypothese lautet, dass Synth-Validation einen größeren absoluten Fehler als die getestete Methode hat und wir schaffen, sie mit $\boldsymbol{\alpha = 0.05}$ nur für die naive Methode abzulehnen. Die p-Werten von allen anderen Methoden liegen aber sehr nah an $\boldsymbol{\alpha}$ - r_boost mit **0.0760**, r_lasso mit **0.2513** und causal_forest mit **0.1535**. Bei mehreren Läufen

könnte es sein, dass sie unter die Ablegungsgrenze fallen. Bis dann können wir nur signifikant behaupten, dass Synth-Validation besser als die naive Methode schätzt.

Weiter beobachten wir auf *Abbildung 10*, welche Methoden am häufigsten von Synth-Validation und dem Orakel ausgewählt werden. Wie bei den Ergebnissen von den Rohdaten haben auch hier alle Methoden mindestens einmal die beste Schätzung (werden von dem Orakel ausgewählt). Wie erwartet ist der Anteil der naiven Methode mit weniger als 10% am niedrigsten, weil sie beim Confounding ein gestreuter Schätzer ist. Die anderen Anteile sind stufenweise zwischen den erwartungsgemäß besseren Methoden verteilt.

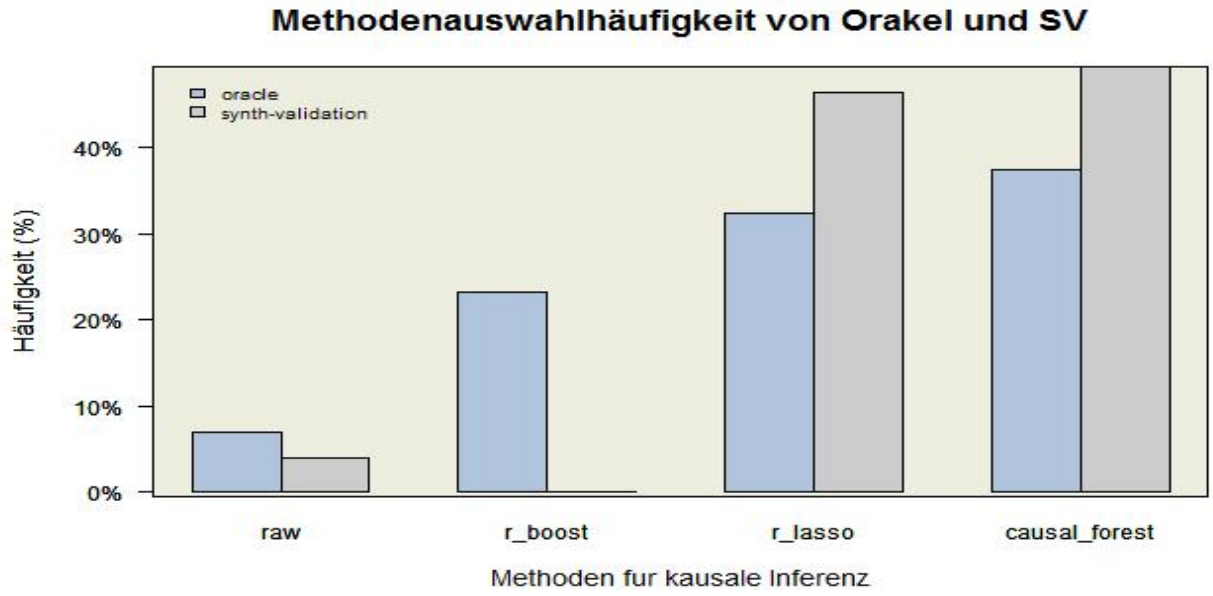

Abbildung 10: Methodenauswahlhäufigkeit von Orakel und Synth-Validation für synthetisch generiete Daten

Man sieht, dass Synth-Validation r_boost kaum ausgewählt hat, was eigentlich sehr erstaunlich ist, da sie die internen synthetischen Daten mit Constrained Boosting erstellt. Damit es keine Verwirrung entsteht, erläutern wir, dass wir hier nicht die gebenchmarkten synthetisch generierten Datensätze meinen, sondern die synthetischen Daten, die Synth-Validation intern generiert. Wir haben erwartet, dass ein Schätzer aus der gleicher Natur wie das Erstellungsverfahren (Constrained Boosting) bessere Schätzungen auf die synthetischen Daten hätte, sein Schätzfehler würde kleiner sein und Synth-Validation würde dazu neigen, ihn häufiger auszuwählen. Wir beobachten stattdessen eine Neigung zu r_lasso und causal_forest, die zusammengerechnet in über 90% der Fälle ausgewählt wurden.

Auf *Abbildung 10* kann man sich die Tabelle anschauen, die die kumulierte Erfolgsrate von Synth-Validation zeigt. In 50.51% der Fälle hat Synth-Validation die beste Schätzung ausgewählt. Im Vergleich zum Ergebnis von den Rohdaten ist die geschätzte Erfolgsrate hier deutlich höher. Das gilt auch für $\boldsymbol{N = 2}$ und $\boldsymbol{N = 3}$, also wenn das Ziel ist, eine von den ersten $\boldsymbol{N}$

besten Methoden auszuwählen. Nach den Signifikanztests, die wir durchgeführt haben, können wir behaupten, dass Synth-Validation besser bei der Auswahl von der besten Schätzung als eine einfache zufällige Auswahl ist. Wir haben die p-Werten von der Binominalverteilung mit dem vorgegebenen $\boldsymbol{\alpha = 0.05}$ verglichen und die Nullhypothese mit großer Signifikanz abgelehnt, denn die p-Werte sind gleich 0 für alle $\boldsymbol{N}$.

N-te beste Schätzung	# ausgewählt	# ausgewählt (kumuliert)	Kumulierte Erfolgsrate
1	50	50	50.51%
2	29	79	79.8%
3	17	96	96.97%
4	3	99	100%

Abbildung 11: Kumulierte Erfolgsrate von Synth-Validation für synthetisch generiete Daten

5.2.4 Zufällig generierte Daten

Zuletzt werden wir die Ergebnisse von den zufällig generierten Daten darstellen. Das war das erste Experiment, das wir durchgeführt haben. Synth-Validation wurde von ihren Autoren auf solche Daten gebenchmarkt. Deswegen war es interessant zu wissen, ob sich irgendetwas geändert hat, wenn Synth-Validation aus den neuen Methoden für kausale Inferenz auswählt, die Algorithmen aus dem maschinellen Lernen zum Schätzen benutzen bzw. ob Synth-Validation immer noch die beste durchschnittliche Schätzung hat. Wir haben 31 Läufe mit zufällig generierten Daten ausgeführt. Die Daten enthalten zwei Kovariaten und haben einen vorgegeben Behandlungseffekt von 0. Der erste Kovariat ist im Intervall $\boldsymbol{(-\pi, \pi)}$ von der Uniformverteilung generiert, der zweite ist konstant 0. Die Ergebnisvariable ist der Sinus von dem ersten Kovariat mit einem normalverteilten Rausch. In der Analyse der Ergebnisse schauen wir uns zuerst die durchschnittlichen Fehler der Methoden auf *Abbildung 12* an.

Auf dem ersten Blick hat der Boxplot über die durchschnittlichen Fehler die gleiche Struktur wie der Boxplot im Artikel von Synth-Validation[SJT+17]. Auf dem zweiten sieht man aber die Unterschiede: Synth-Validation hat nicht mehr die beste durchschnittliche Schätzung, sondern wird von r_boost und causal_forests überholt. Es gibt einige mögliche Gründe, die diesen großen Unterschied erklären:

Unterschiede bei den Implementierungen Die Ergebnisse sind nicht mit dem gleichen Code erzeugt. Obwohl wir den Kernteil von Synth-Validation zur Verfügung hatten, ist es möglich, dass die unverfügbare Logik einige Unterschiede mit unserer selbst implementierten hat. Auch kann es sein, dass es ein Bug in unserer Implementierung gibt.

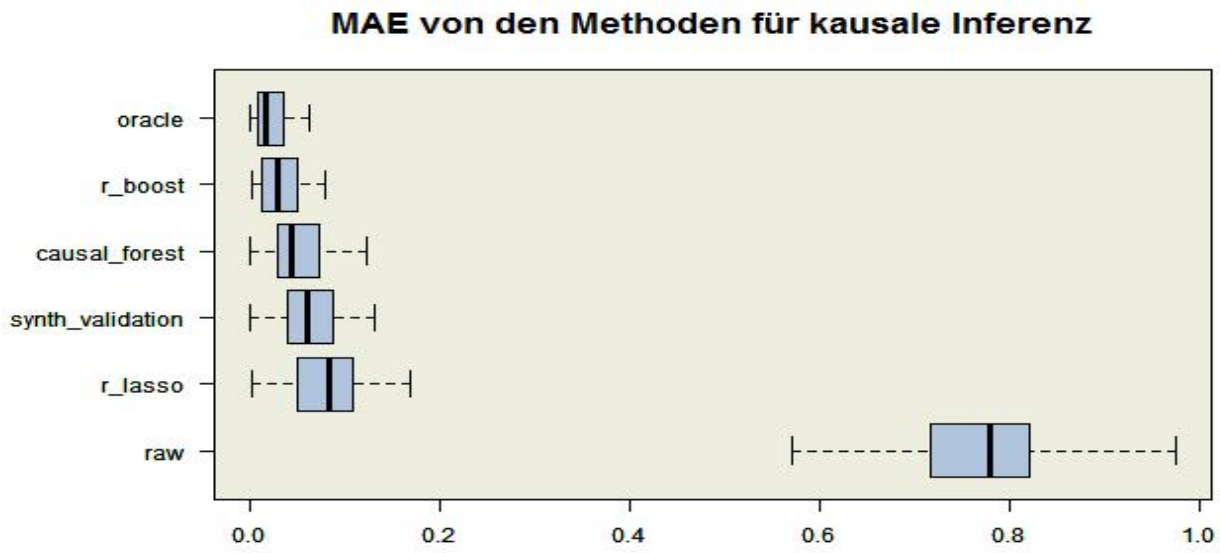

Abbildung 12: MAE von den Methoden für kausale Inferenz und Synth-Validation für zufällig generiete Daten. MAEs auf die vierte Stelle gerundet: oracle - **0.0249**, r_boost - **0.0329**, causal_forest - **0.0533**, synth_validation - **0.0636**, r_lasso - **0.0783**, raw - **0.7528**

Andere Parameter Nach der Angaben der Autoren von Synth-Validation haben sie Experimente mit mehreren unterschiedlichen Parametern durchgeführt und den Durchschnitt für die Ergebnisse genommen[SJT+17]. Wir haben aber in allen 30 Läufen nur Standardparameter benutzt. Eine andere Anzahl der maximalen Schritte beim Constrained Boosting oder breiter ausgewählte synthetische Effekte können einen Einfluss haben.

Synth-Validation wählt die neuen Methoden schlechter aus Vielleicht haben die neuen Methoden einen niedrigeren durchschnittlichen Schätzfehler. Wir sind unsicher, denn die Autoren zeigen nur den Boxplot, geben aber nicht die Zahlenwerte der durchschnittlichen Fehler der Methoden. Wenn also die neuen Methoden tatsächlich besser schätzen, kann es sein, dass die ausgewählten synthetischen Effekte wegen der benutzten Heuristik viel gestreuter als früher sind. Alle erstellten synthetischen Daten werden dann für die Auswahl der Methode genommen, obwohl jetzt einige sehr viel Bias in der Schätzung der Fehler bringen, die im Endeffekt die ausgewählte Methode feststellen. Einfacher gesagt hat die abweichende schlechte Schätzung der naiven Methode deutlich mehr Einfluss auf die Auswahl, wenn alle anderen Schätzungen viel besser sind und sehr nah aneinander liegen.

Im Vergleich zu den anderen Datensätzen haben wir hier deutlich niedrigere Schätzfehler. Wir erklären es mit der kleineren Anzahl der Kovariaten und die Existenz von nur einem, der keine Bedeutung für die Schätzung hat. Mit diesen Ergebnissen und den Signifikanztests, die wir durchgeführt haben, können wir nur behaupten, dass Synth-Validation besser als die naive Methode schätzt. Der p-Wert des Lassos von **0.08458** liegt nah an unserem Signifikanzniveau $\boldsymbol{\alpha = 0.05}$. Wie üblich haben wir einen Welch Test eingesetzt.

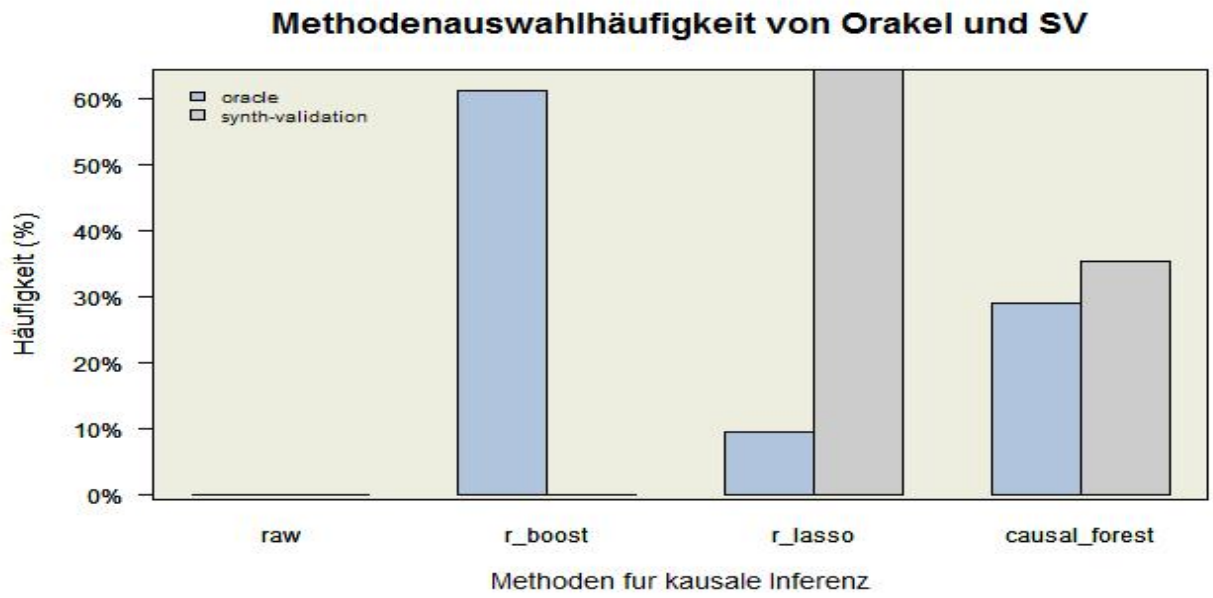

Abbildung 13: Methodenauswahlhäufigkeit von Orakel und Synth-Validation für zufällig generiete Daten

Auf *Abbildung 13* beobachten wir die Methodenauswahlhäufigkeit von dem Orakel und Synth-Validation. Die naive Methode wird kaum ausgewählt - weder vom Orakel, noch von Synth-Validation und das überrascht uns nicht. Schon von dieser Abbildung kann man erkennen, dass die Erfolgsrate von Synth-Validation niedrig sein wird - r_boost hat in 60% der Fälle die beste Schätzung, wird aber kaum von Synth-Validation ausgewählt. Dagegen wurde das Lasso in über 60% der Fälle von Synth-Validation als die beste Methode ausgewählt, obwohl es nur in weniger als 10% tatsächlich war. Bei den kausalen Wäldern sind die Balken relativ auf der gleichen Höhe. Wir haben unsere Theorie für dieses falschen Auswählen schon in einem der oberen Absätze gegeben, dass bei solchen niedrigen Differenzen zwischen den guten Schätzungen die richtige Auswahl viel schwieriger ist, insbesondere wenn es eine schlechte Schätzung gibt, die die synthetischen Effekten streut.

N-te beste Schätzung	# ausgewählt	# ausgewählt (kumuliert)	Kumulierte Erfolgsrate
1	6	6	19.35%
2	8	14	45.16%
3	17	31	100%
4	0	31	100%

Abbildung 14: Kumulierte Erfolgsrate von Synth-Validation für zufällig generiete Daten

Zuletzt betrachten wir die kumulierte Erfolgsrate auf *Abbildung 14*. Etwas fällt schnell auf: in

17 von 31 Läufen (54.84%) hat Synth-Validation die dritte beste Schätzung ausgewählt. Wenn man sich *Abbildung 13* kurz anschaut, stellt man fest, dass diese Schätzung meistens aus dem Lasso kommt. Trotzdem glauben wir, dass das Problem nicht an der Natur des Lassos liegt, sondern an der Nähe von den drei besten Schätzungen und der Existenz von einer vierten schlechteren, die von der Dreiergruppe abweicht. Das strahlt sich auf die synthetischen Effekte und dann auf die Auswahl aus.

Für diesen Datensatz lehnen wir die Nullhypothese, dass Synth-Validation nicht besser als die zufällige Auswahl ist, nur für $\boldsymbol{N = 3}$ ab, also wenn das Ziel ist, eine von den besten drei Methoden auszuwählen. Dabei haben wir durch die Binominalverteilung einen p-Wert von 0 gekriegt, was die Behauptung hochsignifikant macht.

6 Schlussfolgerung

6.1 Diskussion

6.2 Zusammenfassung

Anhang

Literatur

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. (Seite 20).

[CM82a] David R Cox and Peter McCullagh. A biometrics invited paper with discussion. some aspects of analysis of covariance. *Biometrics*, pages 541–561, 1982. (Seiten 9, 10).

[CM82b] David R Cox and Peter McCullagh. A biometrics invited paper with discussion. some aspects of analysis of covariance. *Biometrics*, pages 541–561, 1982. (Seite 11).

[HIR03] Keisuke Hirano, Guido W Imbens, and Geert Ridder. Efficient estimation of average treatment effects using the estimated propensity score. *Econometrica*, 71(4):1161–1189, 2003. (Seite 14).

[Hol86] Paul W Holland. Statistics and causal inference. *Journal of the American statistical Association*, 81(396):945–960, 1986. (Seite 1).

[Lew74] David Lewis. Causation. *The journal of philosophy*, 70(17):556–567, 1974. (Seite 3).

[Lin] Lindsay Dolan. 10 things to know about covariate adjustment. `https://egap.org/methods-guides/10-things-know-about-covariate-adjustment`. [Online; accessed 18-August-2019]. (Seite 11).

[Mac] Macartan Humphreys. 10 things to know about causal inference. `https://egap.org/methods-guides/10-things-you-need-know-about-causal-inference`. [Online; accessed 14-August-2019]. (Seite 1).

[NW17] Xinkun Nie and Stefan Wager. Quasi-oracle estimation of heterogeneous treatment effects. *arXiv preprint arXiv:1712.04912*, 2017. (Seite 15).

[Reb] Rebecca Barter. Confounding in causal inference: what is it, and what to do about it? `http://www.rebeccabarter.com/blog/2017-07-05-confounding/`. [Online; accessed 16-August-2019]. (Seite 3).

[RHB00] James M Robins, Miguel Angel Hernan, and Babette Brumback. Marginal structural models and causal inference in epidemiology. *Epidemiology*, 2000. (Seite 14).

[Rob88] Peter M Robinson. Root-n-consistent semiparametric regression. *Econometrica: Journal of the Econometric Society*, pages 931–954, 1988. (Seite 16).

[RR83] Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983. (Seiten 11, 12).

[Rub74] Donald B Rubin. Estimating causal effects of treatments in randomized and nonran-

domized studies. *Journal of educational Psychology*, 66(5):688, 1974. (Seite 4).

[SJT+17] Alejandro Schuler, Ken Jung, Robert Tibshirani, Trevor Hastie, and Nigam Shah. Synth-validation: Selecting the best causal inference method for a given dataset. *arXiv preprint arXiv:1711.00083*, 2017. (Seiten 7, 22, 23, 24, 25, 26, 27, 28, 36, 45, 46).

[Stu10] Elizabeth A Stuart. Matching methods for causal inference: A review and a look forward. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 25(1):1, 2010. (Seiten 13, 14).

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. (Seite 18).

[VS13] Tyler J VanderWeele and Ilya Shpitser. On the definition of a confounder. *Annals of statistics*, 41(1):196, 2013. (Seite 2).

[WA18] Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018. (Seiten 20, 21).

[Wika] Wikipedia Authoren. Gradient boosting. `https://en.wikipedia.org/wiki/Gradient_boosting`. [Online; accessed 24-August-2019]. (Seiten 18, 19).

[Wikb] Wikipedia Authoren. Zweistichproben-t-test. `https://de.wikipedia.org/wiki/Zweistichproben-t-Test`. [Online; accessed 16-August-2019]. (Seite 5).

[Win] Winston Lin. Regression adjustment in randomized experiments: Is the cure really worse than the disease? `https://web.archive.org/web/20151024055802/http://blogs.worldbank.org/impactevaluations/node/847`. [Online; accessed 18-August-2019]. (Seite 10).