Kostensensitive Entscheidungsbäume für beobachtungsabhängige Kosten

Bachelorarbeit

Verfasser: Lorenz Haller

Betreuer: Prof. Dr. Bernd Bischl

Datum: 22. August 2016

Abstract

Diese Bachelorarbeit beschäftigt sich mit beobachtungsabhängig kostensensitiver Klassifikation. Dabei geht sie zuerst auf die Grundlagen von Kosten und Kostensensitivität ein, bevor sie sich mit verschiedenen bestehenden Verfahren auseinandersetzt.

In der Folge wird ein neues, auf Entscheidungsbäumen basierendes, Verfahren vorgestellt. Dafür werden sowohl die theoretischen Hintergründe erläutert als auch die Implementierung in R beschrieben. Im Anschluss wird die Methode an verschiedenen realen Anwendungsbeispielen getestet und mit den bestehenden Techniken verglichen.

Inhaltsverzeichnis

1 Motivation

Viele konventionelle Klassifikationsverfahren verfolgen in erster Linie das Ziel, möglichst wenige Beobachtungen falsch zu klassifizieren. Für einige Anwendungen ist dies aber nicht das sinnvollste Vorgehen, da jede Missklassifikation mit bestimmten Kosten verbunden ist. Die klassische Herangehensweise ist dabei von der Annahme vollends gleicher Missklassifikationskosten geprägt, was in der Realität oft aber nicht der Fall ist. Insbesondere tritt häufig der Fall auf, dass eine der Klassen nur sehr selten vorkommt, die Fehlklassikationskosten für Beobachtungen aus dieser Klasse aber sehr hoch liegen. Die Fragestellung kann beispielsweise sein, ob ein bestimmter zusätzlicher medizinischer Test zur Untersuchung einer seltenen Krankheit durchgeführt werden soll. Trotz ihrer Rarität kann ein Nichtentdecken dieser Krankheit nämlich mit äußert schwerwiegenden Folgen bis hin zum Tod des Patienten verbunden sein. Für solche Klassifikationsprobleme sind deshalb zahlreiche Methoden, die mit klassenabhängigen Kosten umgehen können, entwickelt worden. Entscheidend ist hierbei also, aus welcher Klasse die Beobachtung stammt und zu welcher Klasse sie letztendlich zugeordnet wird. Doch reicht es immer aus, wenn die Kosten einer Fehlklassifikation allein von der wahren Klasse abhängen?

Ein Beispiel, für das dieser Ansatz nicht ausreichend ist, ist die Anwendung im Kreditscoring. Genauer geht es darum, dass eine Bank herauszufinden versucht, ob ein Kunde kreditwürdig ist, das heißt, ob ihm ein bestimmter gewünschter Kredit gewährt wird oder nicht. Kann der Kunde den Kredit nicht regulär zurückzahlen, bedeutet das für die Bank das Ausbleiben eingeplanten Geldes, das möglicherweise erst mit großer Verzögerung oder im schlimmsten Fall gar nicht zurückgezahlt wird. Die Kosten hängen hierbei aber nicht nur von der „Klasse“ selbst ab, sondern vielmehr von der Höhe des beantragten Kredits und weiteren Merkmalen des Kunden, wie seinen Vermögenswerten - das heißt also die Kosten variieren von Person zu Person bzw. von Beobachtung zu Beobachtung. Aus diesem Grunde sind hierfür Klassifikationsmethoden notwendig, die auch solche beobachtungsspezifischen Kosten berücksichtigen können. Diese Verfahren werden als beobachtungsabhängig kostensensitiv bezeichnet und werden der Hauptaspekt dieser Arbeit sein.

Im Folgenden sollen zunächst die theoretischen Grundlagen von Kosten und Kostensensitivität erläutert werden, bevor dann verschiedene beobachtungsabhängig kostensensitive Ansätze diskutiert werden. Daraufhin wird ein neuer Ansatz von beobachtungsabhängig kostensensitiven Entscheidungsbäumen vorgestellt sowie seine Funktionsweise

für verschiedene Beispiele getestet und mit anderen Verfahren verglichen.

2 Kostensensitivität

Konventionelle Klassifikationsmethoden zielen hauptsächlich darauf ab, möglichst viele Beobachtungen richtig bzw. möglichst wenige Beobachtungen falsch zu klassifizieren [1, S. 6609]. Eine Fehlklassifikation ist dabei die Zuordnung einer Beobachtung zu einer anderen als der wahren Klasse. Die klassischen Vorgehen basieren dabei auf der Annahme, dass alle Fehlklassifikationen mit denselben Kosten verbunden sind. Die fälschliche Zuordnung einer Beobachtung aus Klasse A in Klasse B fällt hier also genauso schwer ins Gewicht, wie eine fälschliche Zuordnung einer Beobachtung aus Klasse B in Klasse A. Für viele reale Probleme trifft diese Annahme jedoch nicht zu. So zum Beispiel bei der Thematik des Kreditkartenbetrugs: eine betrügerische Transaktion nicht zu entdecken, kann einen großen wirtschaftlichen Verlust von tausenden von Euro bedeuten, je nach Höhe der Transaktion. Eine eigentlich zulässige Transaktion nicht zuzulassen, verursacht im Gegensatz dazu deutlich geringere Kosten [1, S. 6609].

Die hier genannten Kosten müssen nicht wie im eben genannten Beispiel stets von monetärer Natur sein, es kann sich dabei auch um einen Verlust an Zeit oder, wie bereits beschrieben, die Nichterkennung einer tödlichen Erkrankung handeln [2, S. 973]. Häufig ist es auch so, dass eine Klasse nur zu einem sehr geringen Prozentsatz auftritt, zum Beispiel im Fall einer seltenen Krankheit. Die Kosten einer Missklassifikation einer Beobachtung aus dieser Klasse liegen dagegen aber sehr hoch, zum Beispiel weil der Patient nicht weiter untersucht wird, die Erkrankung unerkannt bleibt und er in der Folge stirbt [3, S. 435].

Alles in allem sind klassische Verfahren mit gleichen Missklassifikationskosten aus den eben angeführten Gründen für viele Anwendungen nicht ausreichend. Deshalb sind Methoden, die mit unterschiedlichen Missklassifikationskosten arbeiten können, immer gefragter. Ansätze, die diese unterschiedlichen Missklassifikationskosten berücksichtigen, sind als kostensensitive Klassifikationsmethoden bekannt [1, S. 6609]. Die Klasse einer Beobachtung sollte hierbei dann möglichst so vorhergesagt werden, dass sie diejenige mit den geringsten zu erwartenden Kosten ist [2, S. 973]. Nun soll zuerst auf verschiedene Typen von Kosten eingegangen werden, bevor im Anschluss kostensensitive Methoden erörtert werden.

2.1 Kosten

Zur Veranschaulichung soll sich die Erklärung zu Beginn auf Kosten im binären Klassifikationsfall beschränken, das heißt für den Fall, dass die Zielvariable nur zwei Klassen $y_i \in 0{,}1$ besitzt. Der Zwei-Klassen-Fall kann anhand einer Kostenmatrix einfach dargestellt werden [2, S. 973].

	tatsächlich negativ y_i=0	tatsächlich positiv y_i=1
vorhergesagt negativ y_i*=0	C(0,0) = c_{TN}	C(0,1) = c_{FN}
vorhergesagt positiv y_i*=1	C(1,0) = c_{FP}	C(1,1) = c_{TP}

In den Zeilen stehen die jeweils vorhergesagten Klassen, in den Spalten die wahren Klassen. In dieser Notation steht c_{FP} für eine fälschlich positive Klassifikation und c_{FN} für eine fälschlich negative Klassifikation. c_{TN} steht dagegen für eine richtige negative Klassifikation, c_{TP} für eine richtige positive Klassifikation. Die in der Literatur gängigen Bezeichnungen als 'positiv' und 'negativ' stehen hier ohne Wertung allein für die beiden Klassen, vergleichbar mit einem positiven oder negativen Testergebnis auf eine Erkrankung.
Da laut den Plausibilitätsbedingungen von Elkan [2, S. 973] die Kosten einer richtigen Prädiktion immer unter den Kosten einer falschen Prädiktion liegen sollten, muss außerdem folgendes gelten: $c_{FP} > c_{TN}$ sowie $c_{FN} > c_{TP}$.

Ein Klassifikationsproblem wird als **kosteninsensitiv** bezeichnet, wenn die Kosten beider Fehlklassifikationen gleich sind, also $c_{FN} = c_{FP}$ gilt. Als **klassenabhängig kostensensitiv** wird ein Problem bezeichnet, falls die Kosten der Fehlklassifikationen zwar verschieden, jedoch konstant sind. Das bedeutet, dass die Kosten einer Missklassifikation einzig von der wahren Klasse abhängen. Für alle Beobachtungen, die dieser wahren Klasse angehören, sind die Kosten jedoch wieder gleich. Die generellste und flexibelste Art der Kosten sind die **beobachtungsabhängigen** Kosten. Die Kosten einer Missklassifikation sind nicht mehr von der wahren Klasse abhängig, sondern von jeder Beobachtung selbst bzw. von ihren Kovariablen [1, S. 6611]. Im zu Anfang angeführten Beispiel des Kreditkartenbetrugs variieren die Kosten einer Fehlklassifikation von Beobachtung zu Beobachtung, je nach Merkmalen einer Person und Höhe des Kredits.

2.2 Kostenbasierte Klassifikation

Unter Berücksichtigung von Kosten soll die Klasse einer Beobachtung möglichst so vorhergesagt werden, dass sie diejenige mit den geringsten zu erwartenden Kosten ist. Die optimale Prädiktion einer Beobachtung x ist demnach die Klasse j, die folgendes minimiert:

$$L(x, j) = \sum_{k=1}^{K} P(k|x)C(j, k)$$

Für jede Klasse j ist $L(x, j)$ eine Summe über alle möglichen „wahren“ Klassen k von x. $P(k|x)$ beschreibt die Wahrscheinlichkeit, dass k die wahre Klasse von x ist, während $C(j, k)$ die Kosten davon enthält, dass j als Klasse vorhergesagt wird und k die wahre Klasse ist.
Das bedeutet unter anderem, dass es in diesem Kontext nun besser sein kann, eine bestimmte Klasse vorherzusagen, obwohl eine andere Klasse wahrscheinlicher ist. So kann es beispielsweise in der Realität vernünftiger sein, eine große Kreditkartentransaktion zu blockieren, obwohl die Wahrscheinlichkeit dafür, dass die Transaktion rechtmäßig ist, größer ist als die, dass sie es nicht ist. [2, S. 973].

Ein grundlegender Aspekt, in dem sich die Ansätze zur beobachtungsunabhängigen kostensensitiven Klassifikation unterscheiden, ist der Zeitpunkt, zu dem die Kosten dem Algorithmus zugeführt werden. Dieser kann sowohl vor, während oder nach dem Trainieren des Algorithmus liegen [1, S. 6611]. Jede dieser drei Möglichkeiten soll in der Folge kurz anhand populärer Vertreter vorgestellt werden.

Ein häufig gewählter Ansatz für einen kostensensitiven Klassifizierer, der die Kosten vor dem Training aufnimmt, basiert darauf, die Trainingsbeobachtungen entsprechend ihrer Kosten neuzugewichten. Entsprechende Ansätze hierzu sind das kostenproportionale Rejection-Sampling von Zadrozny et al. (2003) oder das kostenproportionale Oversampling von Elkan (2001). Das Rejection-Sampling besteht darin, zufällig Teilmengen aus Beobachtungen des Trainingssets zu bilden, die dann mit einer Wahrscheinlichkeit entsprechend ihrer Missklassifikationskosten in den Trainingsprozess eingehen. Dem gegenüber besteht das Oversampling darin, ein neues Datenset zu kreieren, in dem jede Beobachtung ihrer Kosten entsprechend oft vorkommt.

Ein Vertreter aus der Gruppe der Verfahren, die die Kosten erst nach dem Training berücksichtigen, ist die „Bayes minimum risk“–Methode (*BMR*), die auf den erwarteten Klassifikationskosten basiert. Sie versucht, im Nachhinein einen Ausgleich zwischen der Wahrscheinlichkeit einer Klasse bzw. Entscheidung und den zugehörigen Kosten zu schaffen.
Beide grob beschriebenen Ansätze, die die Kosten entweder vor dem Training oder danach in den Algorithmus einfließen lassen, sind jedoch insgesamt immer noch herkömmliche Ansätze in der Hinsicht, dass sie erst durch das Modifizieren des Trainingsdatensatzes oder das Verändern der resultierenden Wahrscheinlichkeiten zu kostensensitiven Ansätzen konvertiert werden.

Die dritte Methode nämlich, das heißt diejenige, die Kosten direkt während des Trainings in den Algorithmus einfließen lässt, besteht darin, die Methoden selbst entsprechend zu modifizieren.
Verschiedene Ansätze haben dies bereits für die sogenannten Entscheidungsbäume, ein Verfahren aus dem Bereich des Maschinellen Lernens, versucht (vgl.[1, S. 6610]). Diese Ansätze versuchen im Allgemeinen die Kosten dadurch einzuführen, dass sie das sogenannte Unreinheitsmaß, das misst, wie gut eine Splitregel eine Menge an Beobachtungen in zwei Gruppen unterteilt, verändern und in Bezug auf die Kosten gewichten. Während sich frühere Ansätze diesbezüglich nur auf klassenabhängige Kosten fokussiert haben, haben Bahnsen et al. 2015 ([1]) einen neuen Ansatz erarbeitet, der auch beobachtungsabhängige Kosten berücksichtigt. Diese neue Methode basiert auf einem neuen kostensensitiven Unreinheitsmaß und wird im weiteren Verlauf dieser Arbeit noch von Bedeutung sein.

Neben dem Zeitpunkt, zu dem Methoden die Kosten in den Algorithmus einfließen lassen, gibt es noch einen zweiten Aspekt, der die Vielzahl an Verfahren in zwei Gruppen unterteilt. Es handelt sich dabei darum, ob ein Verfahren lediglich für eine binäre Zielvariable, also nur für den Zwei-Klassen-Fall, anwendbar ist oder ob es auch für den Mehrklassen-Fall ausgeführt werden kann. Binäre kostensensitive Methoden sind zum Beispiel die Ansätze von Domingos (1999), Elkan (2001) oder eben genannter von Bahnsen et al. (2015). Zu den Mehrklassen-Methoden zählt dagegen der Ansatz von Lin (2014). Bei der Entwicklung des neuen Ansatzes wurde entsprechend darauf geachtet, dass dieser auch für mehrklassige Zielgrößen anwendbar sein wird.

3 Costing

Ein erstes Beispiel für einen Ansatz zu binärer kostensensitiver Klassifikation ist der des *Cost-Proportionate Example Weighting* bzw. *Costing* von Zadrozny et al. aus dem Jahr 2003([3]). Er fällt in die Kategorie von Verfahren, bei denen die Kosten bereits vor dem Training des Algorithmus einbezogen werden (vgl. [1, S. 6610]) und basiert auf einer kostenproportionalen Gewichtung der Trainingsdaten. Die Funktionsweise dieses Ansatzes ist es, einfache Klassifikationsalgorithmen in kostensensitive zu verwandeln. Das Ziel dabei ist, die erwarteten Kosten $E_{x,y,c\sim D}[c\ I(h(x) \neq y)]$ eines Klassifiziers h für die Trainingsdaten (x, y, c) zu minimieren.

3.1 Motivation

Motiviert wird das ganze durch ein Theorem, das aussagt, dass das Verändern einer ursprünglichen Verteilung D zu einer zweiten Verteilung $\hat{D}$ durch Multiplizieren mit einem Faktor, der proportional zu den relativen Kosten des jeweiligen Beispiels ist, für jeden fehlerminimierenden Klassifizierer die erwarteten Kosten der ursprünglichen Verteilung D minimieren kann (vgl. [3, S. 436]).
Die Beziehung zwischen D und $\hat{D}$ kann wie folgt ausgedrückt werden:

$$\hat{D}(x, y, c) \equiv \frac{c}{E_{x,y,c\sim D}[c]} D(x, y, c)$$

Wenn also Beobachtungen aus der Verteilung $\hat{D}$ nach obiger Definition gegeben sind, dann sind Klassifizierer, die die Fehlerrate für $\hat{D}$ optimieren, optimale kostenminimierende Klassifizierer für Beobachtungen aus D.

Das Theorem besagt, dass für alle Verteilungen D ein $N = E_{x,y,c\sim D}[c]$ existiert, sodass für alle Klassifizierer h gilt:

$$E_{x,y,c\sim \hat{D}}[I(h(x) \neq y)] = \frac{1}{N} E_{x,y,c\sim D}[c\ I(h(x) \neq y)] \tag{1}$$

Die Vorgehensweise, mit kostensensitiven Problemen umzugehen, ist dann unkompliziert: die Verteilung der Trainingsdaten muss entsprechend bestimmter Gewichte c neugewichtet werden, sodass die Trainingsbeobachtungen quasi effektiv aus $\hat{D}$ gezogen werden. Diese Gewichte sind die Kostendifferenzen zwischen einer richtigen und einer

falschen Klassifikation, das heißt entweder die Differenz $c_{FP} - c_{TN}$ oder die Differenz $c_{FN} - c_{TP}$ (vgl. [3, S. 436]).

3.2 Transparent Box: Direkte Verwendung der Gewichte

Um die oben genannten Gewichte c direkt verwenden zu können, werden genauere Informationen über den jeweiligen Lernalgorithmus benötigt. Der Klassifikationsalgorithmus muss die Gewichte so verwenden, dass er für die Daten aus $\hat{D}$ möglichst effektiv lernt.

Miteinbezogen wird dafür die Tatsache, dass viele Lernalgorithmen grob aus zwei Komponenten bestehen: ein Teil berechnet den erwarteten Wert einer Funktion oder Fragestellung f, der andere Teil formt diese Fragestellung und benutzt ihren Output, um daraus einen Klassifizierer zu konstruieren. Falls ein Lernalgorithmus so aufgeteilt werden kann, gibt es eine simple Möglichkeit, um Gewichte direkt zu verwenden. Statt den Erwartungswert mit $\frac{1}{|S|}\sum_{(x,y)\in S} f(x,y)$ zu simulieren, benutzt man dafür $\frac{1}{\sum_{(x,y,c)\in S} c}\sum_{(x,y,c)\in S} cf(x,y)$ (vgl. [3, S. 437]). Die Beobachtungen gehen nun also nicht mehr alle mit gleicher Gewichtung, sondern jeweils unterschiedlich entsprechend ihrer Kostengewichte c in die Berechnung des Erwartungswerts des Klassifizierers ein. Für viele Verfahren, wie Naive Bayes, Boosting oder den C4.5-Algorithmus, ist es dadurch möglich, beobachtungsabhängige Gewichte einfließen zu lassen (vgl. [3, S. 437]).

3.3 Black Box: Gewichtung über Sampling-Verfahren

Doch nicht immer kann direkt auf den Lernalgorithmus zugegriffen werden. Gibt es keinen direkten Zugang auf den Algorithmus, kann über Sampling die Verteilung der Daten umgewandelt werden, um so einen kostensensitiven Lernalgorithmus zu erhalten. Da einfaches Sampling hier weniger gut funktioniert, wird eine andere Methode vorgestellt, die auf der sogenannten *rejection-sampling*-Technik basiert.

Die Rejection-Sampling-Technik ermöglicht es, Beobachtungen unabhängig aus einer Verteilung $\hat{D}$ zu ziehen, unter der Bedingung, dass unabhängig gezogene Beobachtungen aus D gegeben sind. Zwischen D und $\hat{D}$ gilt die anfangs formulierte Beziehung. Die Beobachtungen aus $\hat{D}$ erhält man dabei, indem man zuerst Beobachtungen aus D zieht und dann die Stichprobe mit einer Wahrscheinlichkeit proportional zu $\frac{\hat{D}}{D}$ aufnimmt. Dabei gilt, dass $\frac{\hat{D}}{D} \propto c$.

Eine Beobachtung wird dann mit der Wahrscheinlichkeit c/Z aufgenommen, wobei Z eine Konstante ist, sodass $max_{(x,y,c)\in S} c \leq Z^3$. Aufgrund dieser Verwendung der Gewichte c wird das Verfahren auch *kostenproportionales Rejection-Sampling* genannt.

Als Resultat erhält man ein Datenset S', das in der Regel kleiner ist als das ursprüngliche Datenset S. Außerdem gilt, dass die Beobachtungen in S' entsprechend $\hat{D}$ unabhängig verteilt sind. Indem über das kostenproportionale Rejection-Sampling ein Datensatz S' erzeugt wird, auf den dann ein Lernalgorithmus A(S') angewandt wird, wird garantiert, dass ein näherungsweise kostenminimierender Klassifizierer geschaffen wird, solange der Lernalgorithmus A wiederum in etwa eine Minimierung des einfachen Klassifikationsfehlers erreicht (vgl. [3, S. 438].
Da für dieselbe originale Trainingsstichprobe S in verschiedenen Durchläufen verschiedene Trainingsstichproben S' entstehen, ist es sinnvoll, diese noch mitteln, auch da es sonst leicht zu Overfitting kommen kann. Dafür werden viele Trainingsstichproben $S'_1, ..., S'_m$ erzeugt und für alle der Klassifizierer darauf trainiert. Der endgültige Klassifizierer ist dann ein über alle einzelnen Klassifizierer gemittelter. Dieses Vorgehen ist von Zadrozny et al. als *Costing* bezeichnet worden(vgl. [3, S. 439].

Zur Veranschaulichung sollen die einzelnen Schritte noch einmal zusammengefasst dargestellt werden:

Costing (Lernalgorithmus A, Stichprobe S, Durchlauf t)

1. Für i von 1 bis t
    a. S' = die aus S mit der Wahrscheinlichkeit c/Z entstehende Rejection-Stichprobe
    b. Es gelte: $h_i \equiv A(S')$
2. Klassifzierer $h(x) = sign(\sum_{i=1}^{t} h_i(x))$

3.4 Fazit

Zusammenfassend ist Costing also eine Technik, die beobachtungsabhängig kostensensitive Klassifikation mithilfe eines kosteninsensitiven Klassifizieres ausführen kann (vgl. [3, S. 441]. Das Costing-Verfahren ist eine black-box-Methode und basiert auf einer Neugewichtung der Trainingsdaten entsprechend ihrer Kosten durch die Rejection-Sampling-Technik. Dabei werden aus einer Ursprungsstichprobe S zufällig mehrere

Teilstichproben $S'_1, ..., S'_m$ gezogen, wobei Beobachtungen jeweils mit Wahrscheinlichkeiten c/Z darin aufgenommen werden, die proportional zu ihren Kosten sind. Durch die Mittelung über viele einzelne Klassifizierer wird darüber hinaus die Gefahr des Overfittings reduziert. Ein Nachteil der Methode ist, dass sie nur für eine binäre Zielgröße anwendbar ist (vgl. [4, S. 1]).

4 Kostensensitiver One-Versus-One

Der im Folgenden zweite vorgestellte Ansatz stammt von Lin ([4]) und kann im Gegensatz zur Costing-Methode von Zadrozny et al. beobachtungsabhängig kostensensitive Klassifikation auch für den Mehrklassen-Fall durchführen. Dies erreicht er, indem er kostensensitive Mehrklassenklassifikationsprobleme auf einfache binäre Klassifikationstasks reduziert. Der Algorithmus basiert dabei auf der Kostentransformationstechnik, die bestimmte Kosten in andere umwandelt. Gepaart wird die Kostentransformationstechnik mit einem anderen Algorithmus, dem One-versus-one (OVO) Algorithmus, der Mehrklassenprobleme in binäre Probleme zerlegt. Der entstehende Algorithmus wird dann als kostensensitiver One-versus-one (CSOVO) Algorithmus bezeichnet und kann kostensensitive Mehrklassenklassifikation mit jedem beliebigen binären Klassifizierer durchführen.

4.1 Grundlegendes

Eine kostensensitive Beobachtung wird in der Folge ein Tupel (x, y, c) sein, wobei $c[k]$ die Kosten darstellt, wenn für die Beobachtung Kategorie k vorhergesagt wird. Ein Trainingsdatensatz S_c aus kostensensitiven Beobachtungen sieht demnach wie folgt aus: $S_c = \{(x_n, y_n, c_n)\}_{n=1}^{N} \sim D_c^N$.
Die erwarteten Kosten eines beliebigen Klassifiziers h sind so definiert:

$$E(h, D) \equiv \underset{(x,y,c)\sim D}{\mathcal{E}} c[h(x)]$$

Das Ziel ist es, für den Trainingsdatensatz S_c einen Klassifzierer $\hat{h}$ zu finden, der die erwarteten Kosten $E(\hat{h}, D_c)$ minimiert.
Der „Klassifikationskostenvektor“ ist folgendermaßen definiert: $c_c^{(k)} \equiv [\![l \neq k]\!]$. Der Klassifikationskostenvektor ist demnach gleich 1, wenn die vorhergesagte Klasse mit der wahren Klasse **nicht** übereinstimmt, und gleich 0, wenn vorhergesagte und wahre Klasse übereinstimmen. Ein solcher Vektor kann auch als „Basisvektor“ gesehen werden (vgl. [4, S. 3]).

4.2 Kostentransformation

Die Kostentransformation ist eine Methode, die verschiedene Kostenvektoren miteinander verknüpft. Speziell geht es darum, einen bestimmten Kostenvektor c mit der Menge der Klassifikationskostenvektoren $C_C = \{c_c^{(l)}\}_{l=1}^{K}$ zu verbinden, denn das er-

möglicht es, kostensensitive Klassifikation auf reguläre Klassifikation zu reduzieren. Zu Beginn werden dafür verschiedene Definitionen zu Kostenvektoren benötigt.

Definition (*ähnliche Kostenvektoren* (vgl. [4, S. 4])) Ein Kostenvektor $\tilde{c}$ ist mittels Δ zu einem Kostenvektor c ähnlich, genau dann, wenn $\tilde{c}[\cdot] = c[\cdot] + \Delta$, mit einem konstanten Δ.

Definition (*zerlegbare Kostenvektoren* (vgl. [4, S. 4])) Ein Kostenvektor c ist zerlegbar in eine Menge von Basiskostenvektoren $C_b = \{c_b^{(t)}\}_{t=1}^{T}$, genau dann, wenn nichtnegative Koeffizienten q[t] existieren, sodass $c[\cdot] = \sum_{t=1}^{T} q[t] \cdot c_b^{(t)}[\cdot]$.

Diese zweite Definition ist deshalb sinnvoll, da aus ihr folgendes folgt: Wenn c in Basisvektoren C_c zerlegbar ist, dann gilt für jeden beliebigen Klassifizierer h:

$$c[h(x)] = \sum_{l=1}^{K} q[l] \cdot c_c^{(l)}[h(x)] = \sum_{l=1}^{K} q[l] \cdot [\![l \neq h(x)]\!].$$

Das bedeutet, dass wenn ein Klassenlabel l so generiert wird, dass es proportional zu $q[l]$ ist, und eine kostensensitive Beobachtung (x, y, c) zu einer normalen Beobachtung $(x, l, w = 1)$, mit der jeweils neuen „wahren“ Klasse l, umgelabelt wird, die Kosten, die irgendein Klassifizierer h für seine Prädiktion auf x mit sich bringt, proportional zum erwarteten Klassifikationsfehler sind. Demzufolge würde ein Klassifizierer, der für den umgelabelten, das heißt den einfachen Klassifikationstask, gute Ergebnisse erzielt, auch für den originalen kostensensitiven Klassifikationstask ähnlich gute Ergebnisse erzielen. Somit ist es durch diese Definition nicht nur möglich, einen Kostenvektor in eine equivalente Darstellung umzuwandeln, sondern vielmehr eine kostensensitive Beobachtung zu einer gewöhnlichen umzulabeln (vgl. [4, S. 4]).

Da jedoch nicht alle Kostenvektoren c zu C_c zerlegt werden können, bedarf es hierfür einer weiteren Spezifizierung.

Theorem (*Zerlegung über einen ähnlichen Vektor* (vgl. [4, S. 4f.])) Für einen beliebigen Kostenvektor c ist $\tilde{c}$ über Δ ähnlich zu c. $\tilde{c}$ ist genau dann in C_c zerlegbar, wenn gilt:

$$\Delta \geq (K-1)c_{max} - \sum_{k=1}^{K} c[k], \quad wobei \quad c_{max} = \max_{1 \leq l \leq K} c[l]$$

Aus dem Theorem folgt, dass zwar nicht alle, jedoch unendlich viele Kostenvektoren $\tilde{c}$ verwendet werden können. Nun gilt es herauszufinden, welcher davon vorzuziehen ist. Für ein gegebenes $\tilde{q}$ ist die neue Wahrscheinlichkeitsverteilung $\tilde{p}[l] = \frac{\tilde{q}[l]}{\sum_{k=1}^{K} \tilde{q}[k]}$.
Eine Möglichkeit, um die Varianz entsprechend der Umlabelung zu reduzieren, besteht darin zu fordern, dass die diskrete Wahrscheinlichkeitsverteilung $\tilde{p}[\cdot]$ die minimale Entropie ist. Folgendes Optimierungsproblem gilt es hier demnach zu lösen (vgl. [4, S. 5]).

$$\min_{\tilde{p},\tilde{q},\Delta} \sum_{l=1}^{K} \tilde{p}[l] log \frac{1}{\tilde{p}[l]},$$
$$wobei \;\; c[\cdot] = \sum_{l=1}^{K} \tilde{q}[l] \cdot c_c^{(l)}[\cdot] - \Delta.$$

Die Lösung dazu ist folgende:

Theorem (*Zerlegung über minimale Entropie*) Solange nicht alle c[l] gleich sind, ist die optimale Lösung für die obere Gleichung folgende:

$$\tilde{q}[l] = c_{max} - c[l] \tag{2}$$

$$\Delta = (K-1)c_{max} - \sum_{k=1}^{K} c[k], \quad wobei \quad c_{max} = \max_{1 \le l \le K} c[l] \tag{3}$$

Damit kann dann die resultierende neue Verteilung der Trainingsbeobachtungen $D_r(x, l, w)$ aus $D_c(x, y, c)$ definiert werden:

$$D_r(x, l, w) = [\![w = 1]\!] \cdot \Lambda_1^{-1} \cdot \int_{y,c} \tilde{q}[l] \cdot D_c(x, y, c)$$
$$wobei \;\; \Lambda_1 = \int_{x,y,c} \sum_{k=1}^{K} \tilde{q}[l] \cdot D_c(x, y, c) \;\; die \; Normalisierungskonstante \; ist$$

Die erwarteten Kosten des Klassifizierers h für kostensensitive Beobachtungen aus der ursprünglichen Verteilung D_c lassen sich in Abhängigkeit der neuen Verteilung der

Beobachtungen D_r dann wie folgt darstellen:

$$E(h, D_c) = \Lambda_1 \cdot E(h, D_r) - \Lambda_2$$

Λ_2 erhält man dabei, indem man über alle Δ-Terme (3), die sich für die verschiedenen $c \sim D_c$ ergeben, integriert.

Zum Abschluss kann die Kostentransformationstechnik zur Übersicht noch in Form eines Algorithmus dargestellt werden (vgl. [4, S. 6]):

1. Erhalte N·K Trainingsbeobachtungen $S_w = \{(x_{nl}, y_{nl}, w_{nl})\}$:
   (a) Transformiere alle (x_n, y_n, c_n) über (2) zu $(x_n, \tilde{q}_n)$.
   (b) Für jedes l: $(x_{nl}, y_{nl}, w_{nl}) = (x_n, l, \tilde{q}_n[l])$
   (c) Füge (x_{nl}, y_{nl}, w_{nl}) zu S_w hinzu
2. Verwende einen gewichteten Klassifikationsalgorithmus A_w auf S_w
   um einen Klassifizierer $\hat{h}_w$ zu erhalten.
3. $\hat{h} \equiv \hat{h}_w$

4.3 Kostensensitiver One-Versus-One

4.3.1 One-Versus-One Algorithmus

Der Kostensensitive One-Versus-One Algorithmus ist eine Kombination aus der Kostentransformationstechnik und dem One-Versus-One-Algorithmus. Eine **gewichtete** Version des OVO soll nun kurz vorgestellt werden.

Der One-Versus-One-Algorithmus zerlegt Mehrklassen-Klassifikationstasks in $\frac{K(K-1)}{2}$ binäre Klassifikationstasks (vgl. [4, S. 7]). Für einen Trainingsdatensatz $S_w = \{(x_n, y_n, w_n)\}_{n=1}^{N}$, wobei die w_n jeweils Gewichte sind, soll die Funktionsweise des OVO-Algorithmus nun veranschaulicht werden:

Algorithmus 1: One-Versus-One Algorithmus

1. Für alle i, j, sodass $1 \leq i < j \leq K$

   (a) Nimm die Originalstichprobe $S_w = \{(x_n, y_n, w_n)\}_{n=1}^{N}$ und konstruiere daraus eine binäre Trainingsstichprobe $S_b^{i,j} = \{(x_n, y_n, w_n) : y_n = i \; oder \; j\}$.

   (b) Verwende einen gewichteten binären Klassifikationsalgorithmus $\mathrm{A_b}$ auf $S_b^{(i,j)}$, um einen binären Klassifizierer $\hat{h}_b^{(i,j)}$ zu erhalten.

2. $\hat{h}(x) = \underset{1 \leq l \leq K}{argmax} \sum_{i<j} [\![\hat{g}_b^{(i,j)}(x) = l]\!]$.

Alle binären Tasks vergleichen Beobachtungen aus jeweils nur zwei Kategorien. Jedes $\hat{g}_b^{(i,j)}$ versucht vorherzusagen, ob x Kategorie i oder Kategorie j bevorzugt. $\hat{g}$ sagt dann über ein Mehrheitsvotum aus den einzelnen $\hat{g}_b^{(i,j)}$ voraus.

4.3.2 Kostensensitiver One-Versus-One-Algorithmus

Indem man den OVO-Algorithmus mit der Kostentransformationstechnik kombiniert, erhält man folgende, vorläufige Version des kostensensitiven One-Versus-One-Algorithmus (vgl. [4, S. 8]):

Algorithmus 2: Zwischenalgorithmus

1. Für alle i, j, sodass $1 \leq i < j \leq K$

   (a) Transformiere jede kostensensitive Beobachtung (x_n, y_n, c_n) über (2) zu $(x_n, \tilde{q}_n)$

   (b) Verwende alle $(x_n, \tilde{q}_n)$, um eine binäre Klassifikations-Trainingsstichprobe zu konstruieren.

$$S_b^{(i,j)} = \{(x_n, i, \tilde{q}_n[i]\} \cup \{(x_n, j, \tilde{q}_n[j]\}$$

   (c) Verwende einen gewichteten binären Klassifikationsalgorithmus $\mathrm{A_b}$ auf $S_b^{(i,j)}$, um einen binären Klassifizierer $\hat{h}_b^{(i,j)}$ zu erhalten.

2. $\hat{h}(x) = \underset{1 \leq l \leq K}{argmax} \sum_{i<j} [\![\hat{h}_b^{(i,j)}(x) = l]\!]$.

Die binären Trainingdatensätze $S_b^{(i,j)}$ lassen sich durch einen Trick noch vereinfachen

(vgl. [4, S. 8]):

$$S_b^{(i,j)} = \{(x_n, \underset{l=i \text{ oder } j}{argmax} \ \tilde{q}_n[l], |\tilde{q}_n[i] - \tilde{q}_n[j]|)\} = \{(x_n, \underset{l=i \text{ oder } j}{argmin} \ c_n[l], |c_n[i] - c_n[j]|)\} \tag{4}$$

Algorithmus 3: Kostensensitiver One-Versus-One-Algorithmus (vgl. [4, S. 9])

1. Für alle i, j, sodass $1 \leq i < j \leq K$
   (a) Nimm die originalen Beobachtungen $S_c = \{(x_n, y_n, c_n)\}_{(n=1)}^{N}$ konstruiere $S_b^{(i,j)}$ über (4)
   (b) Verwende einen gewichteten binären Klassifikationsalgorithmus $\mathrm{A_b}$ auf $S_b^{(i,j)}$, um einen binären Klassifizierer $\hat{h}_b^{(i,j)}$ zu erhalten.
2. $\hat{h}(x) = \underset{1 \leq l \leq K}{argmax} \sum_{i<j} [\![\hat{h}_b^{(i,j)}(x) = l]\!]$.

Der Algorithmus konstruiert aus einem Datensatz viele verschiedene Teilstichproben, die jeweils nur Beobachtungen zweier Klassen enthalten. Darauf werden anschließend binäre gewichtete Klassifikationsalgorithmen angewandt, um binäre Klassifizierer $\hat{h}_b^{(i,j)}$ zu erhalten. Diese entscheiden sich für neue Beobachtungen dann also jeweils für eine der beiden Klassen. Der fertige Klassifizierer kombiniert alle einzelnen binären Klassifizierer, berücksichtigt demnach also alle Kombinationen von Klassen, und entscheidet sich dann nach einer Art Mehrheitsvotum für eine der Klassen (vgl. [5, S. 16]).

4.4 Ähnlichkeit zum Weighted-All-Pairs-Algorithmus

Ein dem Algorithmus von Lin sehr ähnlicher Algorithmus ist der Weighted-All-Pairs-Algorithmus von Beygelzimer et al. aus dem Jahr 2005 ([6]). Der Unterschied des Weighted-All-Pairs-Algorithmus zum CSOVO besteht in einer unterschiedlichen Definition von $S_b^{(i,j)}$, also den binären Teilsets. Während beim CSOVO die $S_b^{(i,j)}$ über (4) gebildet werden, geschieht dies beim Weighted-All-Pairs-Algorithmus so (vgl. [4, S. 10]):

$$S_b^{(i,j)} = \{(x_n, \ \underset{l=i \text{ oder } j}{argmin} \ c_n[l], \ |v_n[i] - v_n[j])\}$$

$$wobei \ \ v_n[i] = \int_{c_{min}}^{c_n[i]} \frac{1}{|\{k : c_n[k] \leq t\}|} dt$$

4.5 Schlussfolgerung

Lin hat in seinem Paper eine Kostentransformationstechnik vorgestellt, die jeden beliebigen Kostenvektor c in einen ähnlichen Kostenvektor umwandeln kann, der wiederum mit der minimalen Entropie in die Klassifikationskosten C_c zerlegbar ist. Er hat einen Algorithmus konzipiert, der jeden einfachen Klassifikationsalgorithmus zu einem kostensensitiven machen kann. Zusammen mit dem One-Versus-One Algorithmus hat er den neuen Kostensensitiven One-Versus-One-Algorithmus (CSOVO) geschaffen, der kostensensitive Mehrklassenklassifikation durch Reduzierung auf viele binäre Klassifikationsprobleme ausführen kann. Der CSOVO-Algorithmus besitzt darüber hinaus eine große Ähnlichkeit zum Weighted-All-Pairs-Algorithmus (WAP) von Beygelzimer, was im weiteren Verlauf dieser Arbeit noch eine Rolle spielen wird.

5 Entscheidungsbäume

Da der in diesem Paper vorgestellte neue Ansatz auf Entscheidungsbäumen basiert, soll zunächst allgemein auf diese eingegangen werden. Das Ziel der Entscheidungsbäume ist, wie für viele andere statistische Methoden auch, die Vorhersage einer Zielvariable Y durch verschiedene Kovariablen X_1,...,X_n (vgl. [7, S. 5]). Es gibt zwei Typen von Entscheidungsbäumen, abhängig vom Skalenniveau der Zielvariable. Für eine stetige Zielvariable sind es die Regressionsbäume, für eine kategoriale Zielvariable die Klassifikationsbäume. Die beiden bekanntesten Algorithmen für Entscheidungsbäume sind der C4.5-Algorithmus von Quinlan ([8]) und der CART-Algorithmus von Breiman ([9]) (vgl. [7, S. 5]), wobei hier in der Folge nur der CART-Algorithmus eine Rolle spielen wird. CART steht abgekürzt für 'Classification and Regression Trees', was auch wiederum die Unterteilung in Klassifikations- und Regressionsbäume widerspiegelt.
Die Funktionsweise der Entscheidungsbäume nach Breiman ist folgende: Eine Stichprobe wird immer weiter in jeweils zwei Untermengen - die sogenannten Knoten - aufgeteilt. Begonnen wird im Wurzelknoten, der die vollständige Trainingsstichprobe enthält. Jeder Knoten teilt sich dann immer weiter in zwei Kindsknoten auf, bis hin zu den Terminalknoten, die sich nicht mehr aufteilen. Die Aufteilung geschieht anhand eines bestimmten Splitkriteriums über die Kovariablen [7, S. 7ff.]. Solche Kriterien, die auch als Unreinheitsmaße bezeichnet werden, sind zum Beispiel die Residuenquadratsumme für Regressionsbäume oder die Entropie oder Gini für Klassifikationsbäume. Ein Split wird dann immer mit der Kovariable und an dem Splitpunkt durchgeführt, durch den die größte Verbesserung des Splitkriteriums erzielt werden kann, also z.B. die größte Verringerung der Residuenquadratsumme. Der Zugewinn eines Splitkriteriums für eine bestimmte Splitregel lässt sich dann über die Unreinheit des Vaterknotens abzüglich der Unreinheit in den beiden Kindsknoten berechnen.
Die beiden Kindsknoten S^l und S^r sind dabei folgendermaßen definiert [1, S. 6612]:

$$S^l = \{X_i^a | X_i^a \in S \wedge x_i^j \leq l^j\} \; und \; S^r = \{X_i^a | X_i^a \in S > x_i^j \leq l^j\}$$

Zudem benötigt man die Anzahl an 'positiven' Beobachtungen in jedem Knoten (= S_1), sowie die prozentualen Anteile an 'Positiven' (=π_1).

$$S_1 = \{X_i^a | X_i^a \in S \wedge y_i = 1\} \; sowie \; \pi_1 = |S_1|/|S|$$

Der Zugewinn berechnet sich dann wie folgt:

$$Zugewinn(X^j, l^j) = I(\pi_1) - \frac{|S^l|}{|S|} I(\pi_1^l) - - \frac{|S^r|}{|S|} I(\pi_1^r)$$

Dieser Zugewinn wird für alle möglichen Splitregeln für alle Kovariablen berechnet und der Split mit dem größten Zugewinn ausgewählt und der Vaterknoten S dementsprechend in zwei Kindsknoten S^l und S^r aufgeteilt.

$$(best_x, best_l) = \underset{X^j, l^j}{arg\ max}\ [Zugewinn(X^j, l^j)]$$

Dies geschieht so lange, bis ein bestimmtes Stoppkriterium erreicht ist, z.B. eine festgelegte Minimalanzahl an Beobachtungen in einem Knoten (*Node*) nicht mehr gegeben ist. Ist das der Fall, so befindet sich jede Beobachtung in einem Terminalknoten. Neue Beobachtungen durchlaufen dann den Baum, bis sie in einem Terminalknoten angelangt sind, der dann eine Vorhersage für sie trifft.
Die Vorteile der Entscheidungsbäume liegen in einer leichten Verständlichkeit und guten grafischen Interpretierbarkeit. Zusätzlich werden Interaktionen zwischen Kovariablen gut erkannt. Überdies hinaus sind sie auch für eine Vielzahl an Kovariablen ohne Probleme anwendbar.

Auch wenn bei der Erläuterung der Entscheidungsbäume bisher immer sowohl auf Klassifikations- als auch auf Regressionsbäume eingegangen wurde, werden für den weiteren Verlauf dieser Arbeit nur die Klassifikationsbäume eine Rolle spielen.

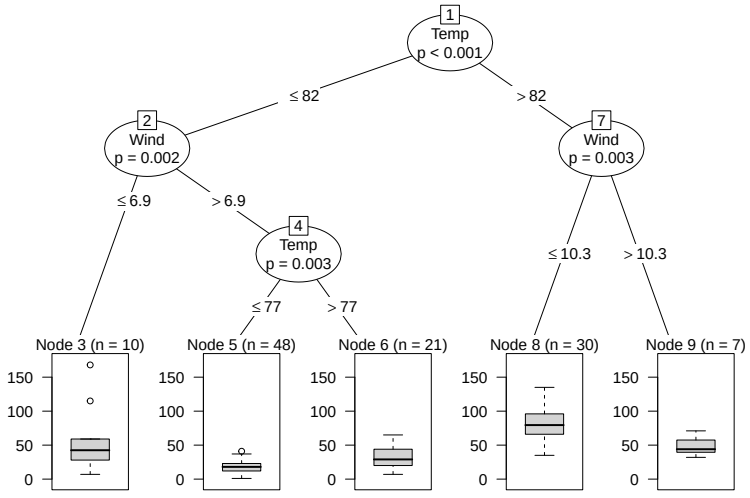

Abbildung 1 : Beispiel eines Entscheidungsbaums

6 Neuer Ansatz

6.1 Theoretische Überlegungen

Im Folgenden soll ein neuer Ansatz für kostensensitive Mehrklassen-Klassifikation für beobachtungsabhängige Kosten, basierend auf Entscheidungsbäumen, vorgestellt werden. Gegeben seien dafür eine Prädiktormatrix $X_{(n,p)}$, die die Information über die Kovariablen $\{x_1, ..., x_p\}$ enthält, und eine Kostenmatrix $Y_{(n,K)}$, die die Kosten C, die jede Handlung $k \in \{1, ..., K\}$ für jede Beobachtung $i \in \{1, ..., n\}$ mit sich bringt, enthält.

$$Y = \begin{array}{c} \\ 1 \\ 2 \\ ... \\ n \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & ... & K \end{array} \\ \begin{pmatrix} C_{1,1} & C_{1,2} & ... & C_{1,K} \\ C_{2,1} & C_{2,2} & ... & C_{2,K} \\ ... & ... & ... & ... \\ C_{n,1} & C_{n,2} & ... & C_{n,K} \end{pmatrix} \end{array} \qquad X = \begin{array}{c} \\ 1 \\ 2 \\ ... \\ n \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & ... & p \end{array} \\ \begin{pmatrix} x_{1,1} & x_{1,2} & ... & x_{1,p} \\ x_{2,1} & x_{2,2} & ... & x_{2,p} \\ ... & ... & ... & ... \\ x_{n,1} & x_{n,2} & ... & x_{n,p} \end{pmatrix} \end{array}$$

Zunächst wird $S \in \mathbb{R}^k$ definiert, das die summierten Kosten jeder Spalte, also jeder Handlung bzw. Klasse, k enthält.

$$S_k = \sum_{i=1}^{n} Y_{i,k} = \sum_{i=1}^{n} C_{i,k}$$

Ebenso werden analog zu S die mittleren Kosten jeder Handlung $m \in \mathbb{R}^k$ definiert.

$$m_k = \frac{S_k}{n}$$

Als „beste“ Klasse a wird diejenige mit den geringsten aufsummierten bzw. geringsten mittleren Kosten definiert.

$$a = arg\ \underset{k}{min}(S_k) = arg\ \underset{k}{min}(m_k)$$

Da im Kontext der Entscheidungsbäume derjenige Split, der den größten Zugewinn, d.h. die größte Verringerung der Unreinheit, mit sich bringt, der beste ist, soll ein entsprechendes Maß definiert werden. Als Unreinheit werden dabei die minimalen absoluten Kosten eines Knotens verwendet, wenn allen Beobachtungen dieses Knotens dieselbe Klasse k zugeordnet wird. Angewandt auf das zuvor für Entscheidungsbäume

definierte Zugewinns-Maß ergibt sich Folgendes:

$$Zugewinn(X^j, l^j) = min(S_k(S)) - min(S_k(S^l)) - min(S_k(S^r)) \tag{5}$$

Das Zugewinnsmaß berechnet für eine bestimmte Kovariable und eine bestimmte Splitregel also die absolute Kostenreduktion, die durch diese Aufteilung des Vaterknotens S auf die beiden Kindsknoten S^l und S^r erzielt wird, wenn allen Beobachtungen jedes Knotens jeweils nur dieselbe eine (die „beste“) Klasse zugeordnet wird.

Die „endgültigen“ Kosten des neuen Verfahrens $Cost(h(S))$ für einen Datensatz S sind die minimalen aufsummierten Kosten aller Terminalknoten S^t des Entscheidungsbaums, also der Knoten, die sich nicht mehr weiter in zwei Kindsknoten aufteilen.

$$Cost(h(S)) = \sum_{t=1}^{T} min(S_k(S^t))$$

6.2 Gütemaße

Neben dem Zugewinnsmaß, das als Entscheidung dafür dient, welche Aufteilungen der Knoten innerhalb eines Entscheidungsbaums durchgeführt werden, muss es außerdem eine Kennzahl geben, um das gesamte Verfahren an sich mit anderen Methoden vergleichen zu können. Dafür werden in der Folge zwei verschiedene Gütemaße vorgestellt. Vorab werden dafür zunächst zwei weitere Parameter definiert.

Der „single-best-Klassifizierer“ enthält die aufsummierten Kosten der „besten“ Klasse a.

$$sb = S_a = \sum_{i=1}^{n} Y_{i,a}$$

Dem single-best-Klassifizierer steht der „virtual-best-Klassifizierer“ gegenüber, der die minimalen aufsummierten Kosten enthält, die entstehen, wenn für jede Beobachtung des gesamten Datensatzes die Handlung mit den minimalen Kosten ausgeführt wird. Der virtual-best-Klassifizierer ist demnach der optimale kostensensitive Klassifizierer. Aufgrund dieser Eigenschaft wird er auch des Öfteren als „Orakel“ bezeichnet.

$$vb = \sum_{i=1}^{n} \sum_{k=1}^{K} min(Y_{i,k})$$

misclassification penalty

Die misclassification penalty, abgekürzt mcp, misst die Kostendifferenz zwischen einem Verfahren und der bestmöglichen Klassifizierung, das heißt dem virtual-best-Klassifizierer, die man erhält, wenn für jede Beobachtung immer die Klasse mit den geringsten Kosten ausgewählt wird (vgl. [10]). Das heißt also, je niedriger die mcp, desto besser.

$$mcp = Cost(h) - vb$$

Savings-Rate

Das zweite vorgestellte Gütemaß ist die sogenannte „Savings-Rate“, die die prozentuale Kostenreduktion durch ein Verfahren bzw. einen Klassifizierer h im Vergleich zum single-best-Klassifizierer darstellt. Sie stammt von Bahnsen et al. (vgl. [1, S. 6611]) und kann folgendermaßen formuliert werden:

$$Savings(h(S)) = \frac{sb - Cost(h(S))}{sb} \tag{6}$$

6.3 Implementation in R

Das neue Verfahren wird als neue Splitfunktion für die R-Funktion *'rpart'* implementiert. *Rpart* beinhaltet rekursives Partitionieren für Klassifikation und Regression auf Basis des Buchs von Leo Breiman ([9]). Dazu gehören unter anderem auch der schon beschriebene *CART*-Algorithmus für Entscheidungsbäume. Rpart ermöglicht es, eigene Splitfunktionen einzufügen. Dafür sind drei Teilfunktionen notwendig: eine Initialisierungsfunktion, eine Evaluationsfunktion und eine Splitfunktion (vgl. [11, S. 1]). Die Initialisierungsfunktion nimmt die Parameter auf und führt Checks durch. Die Evaluationsfunktion wird einmal pro Knoten ausgeführt und berechnet für jeden Knoten eine Devianz. Zudem gibt sie dem Knoten ein Label, das heißt den Namen der Klasse, die die Beobachtungen in diesem Knoten als Prädiktion erhalten würden. In diesem Fall sind das die Kosten der Klasse mit den minimalen Kosten sowie deren Label. Die Splitfunktion wird dagegen einmal für jede Kovariable ausgeführt und berechnet den besten Split für diese Kovariable. Die Devianz des besten Splits, das heißt in diesem Fall die Kosten des besten Splits, werden als 'goodness' festgehalten und dienen dazu, entscheiden zu können, mit welcher Kovariable der Split durchgeführt wird.

6.4 Evaluationsfunktion

Die Evaluationsfunktion für das neue Verfahren sieht demnach wie folgt aus:

```
asTreeEval = function(y, wt, parms) {
  cm = colSums(y)
  j = which(cm == min(cm))
  if (length(j) > 1) {
    j = sample(j, 1)
  }
  list(label = c(j,colnames(y)[j], cm), deviance = cm[j])
}
```

Wie bereits beschrieben, werden zuerst die aufsummierten Kosten der Beobachtungen des Knotens für jede Klasse berechnet. Die Klasse mit den geringsten Kosten wird als Label für den Knoten verwendet, die dazugehörigen Kosten als Devianz ausgegeben.

6.4.1 Splitfunktion für stetige Kovariablen

Nach der Evaluationsfunktion sollen nun die Splitfunktionen jeweils für stetige und kategoriale Kovariablen vorgestellt werden. Begonnen werden soll mit der für stetige Kovariablen:

```
asTreeSplit = function(y, wt, x, parms, continuous) {
  n = nrow(y)
  goodness = numeric(n-1)
  if (continuous) {
    yy0 = colSums(y[1:n,,drop=FALSE] * wt[1:n])
    for (i in 1:(n-1)) {
      yy1 = colSums(y[1:i,,drop=FALSE] * wt[1:i])
      yy2 = colSums(y[(i+1):n,,drop=FALSE] * wt[(i+1):n])
      goodness[i] = min(yy0) - min(yy1) - min(yy2)
    }
    list(goodness = goodness, direction = rep(-1, n-1))
  } else {
```

```
...
}
```

Da *rpart* hier automatisch die Beobachtungen nach dem Wert der Kovariablen sortiert, ist es ausreichend für $i \in \{1, ...n-1\}$ jeweils nur die Kosten der Beobachtungen $\{1, ..., i\}$ mit denen der Beobachtungen $\{i+1, ..., n\}$ zu vergleichen. So werden für die Beobachtungen $\{1, ..., i\}$ sowie die Beobachtungen $\{i+1, ..., n\}$ die aufsummierten Kosten aller Klassen berechnet und die minimalen Kosten verwendet. Diese sind die minimalen Kosten davon, wenn alle Beobachtungen $\{1, ..., i\}$ bzw. $\{i+1, ..., n\}$ jeweils dieselbe Klassenzuteilung erhalten. Als Maß gilt dann, wie sehr durch diesen Split die Kosten im Vergleich zum Vaterknoten verringert werden konnten, was oben definiertem Zugewinns-Maß entspricht. Demnach werden von den minimalen Kosten des Vaterknotens jeweils die minimalen Kosten der beiden Kindsknoten abgezogen. Diejenige Aufteilung der Beobachtungen in $\{1, ..., i\}$ und $\{i+1, ..., n\}$, die die größte Kostenreduktion mit sich bringt, ist der kostenoptimalste Split, der für diese Kovariable durchgeführt werden kann.

Als Beispiel zum Verständnis soll folgendes Minimalbeispiel eines Splits anhand der stetigen Kovariable *duration* dienen:

```
1) root 9000 1900.4660 2
  2) duration< 690.5 8454  277.4304 2
  3) duration>=690.5 546  546.0000 1
```

'1)' stellt den Wurzelknoten dar. Er beinhaltet 9000 Beobachtungen und seine aufsummierten minimalen Kosten betragen 1900,4600. Die Klasse mit den minimalen Kosten für den Wurzelknoten ist Klasse 2. Der Wurzelknoten wird dann anhand der Variable *duration* in zwei Kindsknoten gesplittet. Beobachtungen, deren *duration* kleiner als 690,5 ist, werden dem einen Kindsknoten zugeordnet. Dies sind 8454 Stück und die aufsummierten Kosten dieses Kindsknotens betragen 277,4304. Die kostenminimale Klasse für Beobachtungen dieses Kindsknotens ist Klasse „2“. Dem gegenüber werden Beobachtungen, deren *duration* größer oder gleich 690,5 ist, dem anderen Kindsknoten zugeordnet, dessen kostenminimale Klasse „1“ ist. Dies trifft für 546 Stück zu, die aufsummierte Kosten von 546 besitzen. Die Kostenreduktion durch diesen Split beträgt $1900,466 - 277,4304 - 546 = 1077,036$.
Dies sollte nur ein einfaches Beispiel für die Aufteilung eines Knotens darstellen. Die

Knoten werden im Normalfall nun immer weiter gesplittet, bis keine Kostenreduktion mehr möglich ist bzw. ein Stoppkriterium erreicht wurde.

6.4.2 Splitfunktion für kategoriale Kovariablen

Die Splitfunktion für kategoriale Kovariablen sieht des Weiteren so aus:

```
else {
    # Categorical X variable
    ux = sort(unique(x))
    l = unique(x)
    k = length(l)

    # if the number of levels k is uneven:
    if(k %% 2 == 1) {
      z = (2^(k-1)) - 1
    } else {
      z = (2^(k-1)) - 1 + (gamma(k + 1) / (gamma(k/2 + 1)
      * gamma(k/2 + 1))/2)
    }

    zmat = matrix(0,nrow=z,ncol=1)
    zmat = as.list(zmat)
    o = 0

    # create combinations of the different levels
    for(i in 1:as.integer(k/2)){
      c = combn(x = l, m = i, simplify = FALSE)
      for(j in 1:length(c)) {
        o = o+1
        zmat[o] = c[j]
      }
    }
    goodness = numeric(length(zmat))
    yy0 = colSums(y[,,drop=FALSE])
```

```
for(i in 1:length(zmat)) {
  a = which(x %in% zmat[[i]])
  yy1 = colSums(y[a,,drop=FALSE])
  yy2 = colSums(y[-a,,drop=FALSE])
  goodness[i] = min(yy0) - min(yy1) - min(yy2)
}

jmax = which(goodness == max(goodness))
jjmax = jmax[1]
lz = length(zmat[[jjmax]])

goodness_x = numeric(k-1)
goodness_x[lz] = jjmax
goodness_x
rest = l[!(l %in% zmat[[j]])]
direction = c(zmat[[j]],rest)
list(goodness=goodness_x, direction = direction)
```

Zuerst werden die Anzahl an Kategorien k sowie deren Namen festgehalten. In der Folge wird die maximale Anzahl an Kombinationen der einzelnen Kategorien berechnet. Diese berechnet sich so: $z = 2^{k-1} - 1$ (vgl. [11, S. 6]). Für alle möglichen Kombinationen werden dann entsprechend die aufsummierten Kosten aller Klassen berechnet. Als Güte dieses Splits, das heißt dieser Aufteilung der Kategorien, werden die Kosten des Vaterknotens abzüglich der minimalen Kosten der beiden Kindsknoten verwendet. Diese minimalen Kosten sind erneut die minimalen Kosten dafür, wenn für alle Beobachtungen in dem jeweiligen Kindsknoten dieselbe Klasse ausgewählt wird.
Der beste Split für diese Kovariable ist am Ende dann wiederum der, durch dessen Aufteilung der Kategorien auf die beiden Knoten die größtmögliche Kostenreduktion im Vergleich zu den Kosten des Vaterknotens erfolgt.

Zum besseren Verständnis soll die Funktionsweise dieses Splits anhand eines kleinen Beispiels erläutert werden: Gegegen seien eine Zielvariable Y mit den drei Klassen $\{A, B, C\}$ sowie eine Kovariable mit den vier Kategorien $\{rot, blau, gelb, grün\}$.
Die aufsummierten Kosten des Vaterknotens zeigt folgende Tabelle. Demnach wäre für

den Vaterknoten, in dem alle vier Kategorien noch gemeinsam auftreten, Klasse A die beste Wahl.

Tabelle 1 : Aufsummierte Kosten des Vaterknotens

A	B	C
104	105	109

Nun sollen für alle Kombinationen der vier Kategorien die entsprechenden Kosten dargestellt werden:

Tabelle 2 : Aufsummierte Kosten des Kindsknoten, je nach Aufteilung der Kategorien; in grün die jeweils geringsten Kosten

	A	B	C
{rot}	18	43	37
{blau, gelb, grün}	86	62	72
{blau}	35	15	29
{rot, gelb, grün}	69	90	80
{gelb}	17	36	15
{rot, blau, grün}	87	69	94
{grün}	34	11	28
{rot, blau, gelb}	70	94	81
{rot, blau}	53	58	66
{gelb, grün}	51	47	43
{rot, gelb}	35	79	52
{blau, grün}	69	26	57
{rot, grün}	52	54	65
{blau, gelb}	52	51	44

Nun soll noch die Güte jedes Splits dargestellt werden. Diese ergibt sich jeweils aus der Differenz der minimalen Kosten des Vaterknotens abzüglich der minimalen Kosten der Kindsknoten (oben in hellblau dargestellt).

Tabelle 3 : Güte aller möglichen Splits

	Güte
{rot},{blau, gelb, grün}	24
{blau},{rot, gelb, grün}	20
{gelb},{rot, blau, grün}	20
{grün},{rot, blau, gelb}	23
{rot, blau},{gelb, grün}	8
{rot, gelb},{blau, grün}	43
{rot, grün},{blau, gelb}	8

Der Split mit der größten Güte ist die Aufteilung der Kategorien in $\{rot, gelb\}, \{blau, grün\}$. Dieser Split wäre in diesem Fall also beste für diese Kovariable.

7 Ansatz von Bahnsen et al.

Der Ansatz von Bahnsen et al. ist ähnlich zum neuen Ansatz dieses Papers. Allerdings ist er nur für binäre Zielvariablen $y_i \in \{0, 1\}$ anwendbar. Der Ansatz definiert ein Unreinheitsmaß, das die minimalen Kosten berücksichtigt, die entstehen, wenn alle Beobachtungen in einem Knoten entweder Klasse 0 oder Klasse 1 zugeordnet werden (vgl. [1, S. 6613]).

$$I_c(S) = min\{Cost(f_0(S)), Cost(f_1(S))\}$$

Das Ziel dieses Maßes ist es, die geringsten erwarteten Kosten einer Splitregel abzuschätzen. Nach dieser Logik wird diejenige Prädiktion als Vorhersage für neue Beobachtungen verwendet, die zu den geringsten Kosten führt.

$$f(S) = \begin{cases} 0 \;\; wenn \;\; Cost(f_0(S)) \leq Cost(f_1(S)) \\ 1 \;\; sonst \end{cases}$$

Der kostenbasierte Zugewinn des Splitkriteriums unter Verwendung des neuen Unreinheitsmaßes von Bahnsen et al. ist dann wie folgt definiert (vgl. [1, S. 6612]):

$$\begin{aligned} Zugewinn_B(X^j, l^j) = & \; min\{Cost(f_0(S)), Cost(f_1(S))\} \\ & - min\{Cost(f_0(S^l)), Cost(f_1(S^l))\} \\ & - min\{Cost(f_0(S^r)), Cost(f_1(S^r))\} \end{aligned}$$

Diese einfache Darstellung ist möglich, da die minimalen Kosten als Unreinheitsmaß ein absolutes Maß darstellen.
Anzumerken ist zudem, dass für eine binäre Zielgröße diese Definition eines Zugewinns-Maß mit der aus (5) übereinstimmt. Das neue Zugewinns-Maß aus (5) ist jedoch im Gegensatz zu der Version von Bahnsen et al. auch für eine mehrklassige Zielvariable anwendbar.

Neben dem neuen Unreinheitsmaß haben Bahnsen et al. für das Verfahren zudem eine neue, kostensensitive Pruning-Methode entwickelt. Pruning ist eine Strategie, um die Gefahr des Overfittings für einen Entscheidungsbaum zu reduzieren. Dabei werden Zweige des Baums, die nicht oder nur geringfügig dazu beitragen, die Prognosegenauig-

keit zu erhöhen bzw. im kostensensitiven Fall die Kosten zu reduzieren, „abgeschnitten“ (vgl. [7, S. 10]). Da Pruning für das neue Verfahren jedoch keine Rolle spielt, wird es an diesem Punkt nicht weiter erläutert.

8 Anwendungsbeispiele

Im Folgenden soll nun das neue Verfahren anhand zweier Anwendungsbeispiele getestet und mit anderen Verfahren verglichen werden.

Als Maß für die Güte der Verfahren dient dabei jeweils die *misclassification penalty (mcp)*.

8.1 Direktmarketing

Als erstes Beispiel soll ein Datensatz aus dem Direktmarketing dienen. Beim Direktmarketing will man herausfinden, welche Kunden eher dazu geneigt sind, von einer bestimmten Marketing-Kampagne angesprochen zu werden. Der Datensatz ist derselbe, den auch Bahnsen et al. (vgl. [1, S. 6614]) in ihrem Paper als Beispiel verwenden und stammt von Moro et al. [12]. Er wird verwendet, um die Ergebnisse des neuen Verfahrens mit denen des Verfahrens von Bahnsen et al. vergleichen zu können. Der Datensatz enthält circa 45000 Bankkunden, denen telefonisch eine Langzeiteinlage mit attraktiven Zinssätzen angeboten wurde. Neben einigen Kovariablen wie dem Alter, dem Job oder dem Ehestatus steht auch die Information darüber zur Verfügung, ob der Kunde das Angebot angenommen hat. Die Problemstellung dabei ist eine beobachtungsabhängig kostensensitive, da die Kosten für falsch positive und falsch negative Klassifikationen variieren. Eine fälschlich positive Klassifikation ist mit den Kosten verbunden, die bei der Kontaktierung des Kunden entstehen, während eine fälschlich negative Klassifikation einen Einbuße an Einkommen bedeutet, da der Kunde bei einer Kontaktierung eine Langzeiteinlage eröffnet hätte (vgl. [1, S. 6614]).

Die Kostenmatrix sieht nach Bahnsen folgendermaßen aus:

	tatsächlich negativ y_i=0	tatsächlich positiv y_i=1
vorhergesagt negativ y_i*=0	$c_{TN} = 0$	$c_{FN} = Int_i$
vorhergesagt positiv y_i*=1	$c_{FP} = C_a$	$c_{TP} = C_a$

Für die administrativen Kosten C_a wird angenommen: $C_a = 1$; für Int_i wird angenommen: $Int_i = (balance * 20\%) * 2.463333\%$ (vgl. [13]).

Der Datensatz enthält insgesamt sieben numerische und neun kategoriale Kovariablen.

Der neue Ansatz dieses Papers soll auf dem Direktmarketing-Datensatz mit drei weiteren kostensensitiven Klassifizierern verglichen werden. Der erste davon ist der **Weighted All Pairs Wrapper**, der wie beschrieben eine große Ähnlichkeit zum Kostensensitiven One-Versus-One Verfahren von Lin besitzt. Die anderen beiden sind jeweils einfache Klassifikations- bzw. Regressionswrapper.
Der **kostensensitive naive Klassifikationswrapper** verwendet die Kosten als Klassenlabels, wobei für jede Beobachtung immer die Klasse mit den geringsten Kosten als Label ausgewählt wird. Dann wird ein reguläres Klassifikationsverfahren angewandt, das auf die Klassenlabels predicten soll. Ähnlich verhält es sich für den **kostensensitiven naiven Regressionswrapper**. Bei diesem wird zunächst für die Kosten jeder Klasse ein individuelles Regressionsmodell gefittet. Bei der Prädiktion werden dann zunächst die Kosten aller Klassen geschätzt und die Klasse mit den geringsten vorhergesagten Kosten als Prädiktion verwendet (vgl. [14]).

Die untenstehende Tabelle 4 zeigt die Ergebnisse einer Benchmarkstudie mit zehnfacher Kreuzvalidierung.
Benchmark-Experimente sind eine empirische Methode, um Lernalgorithmen auf einem oder mehreren Datensätzen zu vergleichen (vgl. [15, S. 3]). Die Kreuzvalidierung ist eine Methode, um die Güte eines Modells oder Verfahrens zu bewerten. Die einfache, auch K-fache Kreuzvalidierung genannt, teilt einen Datensatz in K möglichst gleich große Teildatensätze auf. Ein Teildatensatz wird immer als Testdatensatz verwendet, die anderen K-1 sind die Trainingsdatensätze, auf denen das Modell gefittet wird. Mit dem Modell werden die Werte für die Beobachtungen auf dem Testdatensatz vorhergesagt und ein Fehler ermittelt. Dieses Vorgehen wird so oft wiederholt, bis jeder Teildatensatz einmal der Testdatensatz gewesen ist. Die Fehler aller K Testdatensätze werden im Anschluss zu einem Durchschnittsfehler gemittelt. Die gängigsten Varianten sind die fünf- und die zehnfache Kreuzvalidierung (vgl. [16, S. 497f.]). Es gibt zusätzlich auch noch die Möglichkeit, eine stratifizierte Kreuzvalidierung durchzuführen, aber da in der Folge nur die einfache Kreuzvalidierung verwendet wird, wird nicht weiter auf die stratifizierte Version eingegangen.

Das Verfahren von Bahnsen et al. erzielte Savings (6) von 69% (vgl. [1, S. 6615]) , das heißt die ursprünglichen Kosten des Wurzelknotens konnten um 69% verringert werden. Ein Vergleich des neuen Verfahrens mit dem Ergebnis von Bahnsen war nur

Tabelle 4 : Benchmark-Ergebnisse: die *misclassification penalty* der verschiedenen Verfahren für den Direktmarketing-Datensatz

	NKW	NRW	NEU	WAP
mcp	0.6606	0.4009	0.2763	0.2243

NKW: Naiver Klassifikations-Wrapper; NRW: Naiver Regressions-Wrapper; NEU: der neue Ansatz dieses Papers; WAP: Weighted-All-Pairs-Wrapper

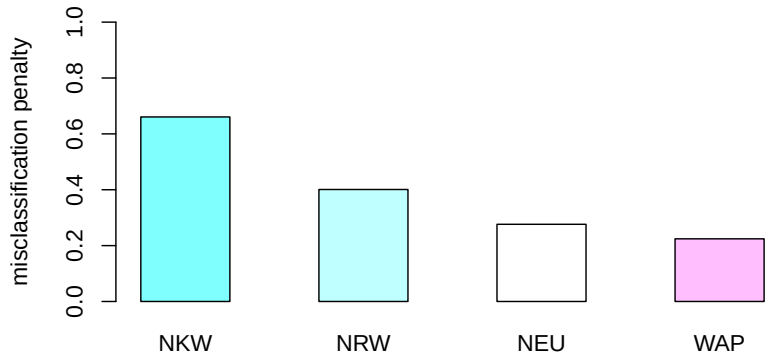

Abbildung 2 : Die *misclassification penalty* für den Direktmarketing-Datensatz NKW: Naiver Klassifikations-Wrapper; NRW: Naiver Regressions-Wrapper; NEU: der neue Ansatz dieses Papers; WAP: Weighted-All-Pairs-Wrapper

eingeschränkt möglich. Dies hängt damit zusammen, dass das Paper von Bahnsen et al. nur das Ergebnis sowie Informationen über die Anzahl an Beobachtungen im Trainings- (15346 Beobachtungen) und im Testdatensatz (11231 Beobachtungen) lieferte(vgl. [1, S. 6613]). Dadurch war es nicht möglich, für das neue Verfahren die exakt selbe Aufteilung der Beobachtungen auf Trainings- und Testdaten zu erreichen. Jedoch wurden mit den gegebenen Zahlen zwanzig mal Trainings- und Testdatensätze zufällig gezogen, um so einen groben Vergleichswert zu erhalten.

Das neue Verfahren erzielte dabei eine Savingrate von 61,7%, das heißt die Anfangskosten konnten um 61,7% reduziert werden. Die Savingrate liegt dabei also einige Prozentpunkte unter der von Bahnsen. Zwar können die beiden Werte wie beschrieben nicht exakt miteinander verglichen werden, als grobe Einschätzung, wie gut das neue Verfahren für eine binäre Zielgröße funktioniert, kann dieses Ergebnis aber durchaus dienen.

8.2 ASlib

Das zweite Beispiel, auf das das neue Verfahren getestet und verglichen werden soll, ist die „Benchmark Library for Algorithm Selection **ASlib**“. Die Aufgabe der Algorithmusselektion besteht darin, eine Menge von Algorithmen auf ihre unterschiedliche Leistungsfähigkeit für eine Reihe von Beobachtungen zu untersuchen (vgl. [17, S. 1]). Aslib repräsentiert Algorithmusselektions-Szenarien sowie eine Reihe verschiedener Datensätze aus der Literatur. Da im Normalfall kein Algorithmus für alle Beobachtungen eines Datensatzes die beste Lösung liefert, gilt es verschiedene, sich ergänzende Algorithmen auszuwählen und eine Strategie zu finden, wann welcher Algorithmus zum Einsatz kommt. Denn immer alle Algorithmen auf eine Beobachtung anzuwenden, um zu sehen, welcher Algorithmus dafür der beste ist, ist mit großem Zeitaufwand verbunden und dadurch häufig nicht machbar.
ASlib versucht nicht nur, aus einer Reihe von Algorithmen den Algorithmus zu finden, der für ein Set an Beobachtungen die beste Performanz liefert, sondern vielmehr, für jede einzelne Beobachtung den besten Algorithmus auszuwählen (vgl.[17, S. 4]). Es handelt sich um eine „per-instance algorithm selection“([17, S. 4]), also eine Algorithmusselektion für jede einzelne Beobachtung.

Ein ASlib-Scenario enthält folgende Informationen (vgl. [17, S. 9]):

- Die Kovariablen für alle Beobachtungen
- Die Performance der Algorithmen anhand eines Gütemaßes, in der Regel die Laufzeit
  - die Laufzeiten der Algorithmen werden im Folgenden als Kosten verwendet
  - es gibt eine Maximallaufzeit: erreicht eine Algorithmus diese sogenannte 'Cutoff-Time', wird seine Ausführung abgebrochen und ein fester Cutoff-Wert als Laufzeit angegeben
- Informationen, wie die Daten am besten in Trainings- und Testdatensätze aufgeteilt werden, um eine möglichst unverzerrte Schätzung zu erhalten

Für die sechs ASlib-Szenarien wird der neue Klassifizierer zusätzlich zu den drei Klassifizierern aus dem Direktmarketing-Beispiel noch mit bereits bestehenden Ergebnissen aus ASlib verglichen. Diese entstammen einem einfachen Klassifikationsbaums mit

Tabelle 5 : Überblick über die Szenarien

	# Algor.	Cutoff-Time	# Beob.	# Feat.
QBF-2011	5	3600	1368	46
PREMARSHALLING-ASTAR	4	3600 (5120)	527	22
QBF-2014	14	900	1254	46
MAXSAT12-PMS	6	2100	876	37
CSP-2010	2	5000(4096)	2024	86
ASP-POTASSCO	11	600	1294	138

'rpart', bei dem die Kosten im Nachhinein entsprechend eingetragen wurden. Vergleichbar werden diese Ergebnisse mit den Ergebnissen der eigenen Berechnungen durch die Verwendung der von ASlib mitgelieferten Information zur Aufteilung der Daten in Trainings- und Teststichprobe für die Kreuzvalidierung.

Die untenstehende Tabelle 6 zeigt wiederum die Ergebnisse einer Benchmarkstudie mit zehnfacher Kreuzvalidierung.

Tabelle 6 : Benchmark-Ergebnisse: die *misclassification penalty* der verschiedenen Verfahren für verschiedene Szenarien

	NKW	NRW	NEU	WAP	RPC
QBF-2011	301	245	172	147	227
PREMARSHALLING-ASTAR	490	586	579	548	537
QBF-2014	123	82	53	43	102
MAXSAT12-PMS	91	81	50	37	64
CSP-2010	41	39	10	10	11
ASP-POTASSCO	33	34	17	16	28

NKW: Naiver Klassifikations-Wrapper; NRW: Naiver Regressions-Wrapper; NEU: der neue Ansatz dieses Papers; WAP: Weighted-All-Pairs-Wrapper; RPC: Klassifikationsbaum mit 'rpart'

Es zeigt sich, dass der Weighted-All-Pairs-Wrapper mit Ausnahme des *PREMARSHALLING-ASTAR*-Szenarios die geringste mcp liefert. Der neue Ansatz zeigt konstant eine etwas höhere mcp als der WAP, erzielt aber insgesamt bessere Ergebnisse als der einfache rpart-Klassifikationsbaum und deutlich bessere Ergebnisse als die beiden naiven Wrappermethoden.

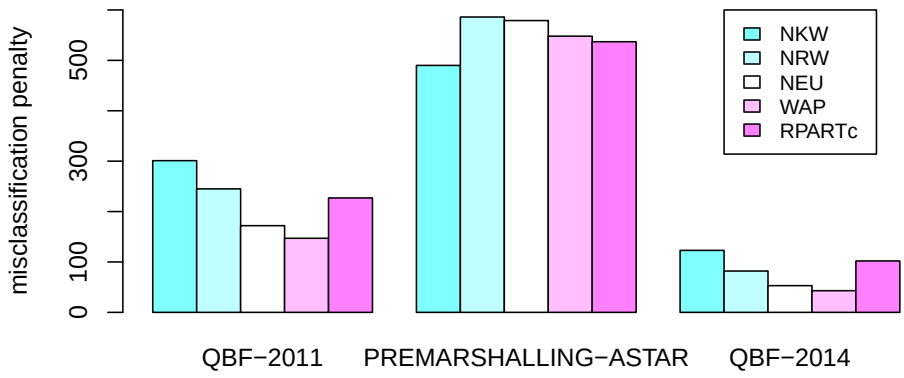

Abbildung 3 : Die *misclassification penalty* für die ersten drei Szenarien
NKW: Naiver Klassifikations-Wrapper; NRW: Naiver Regressions-Wrapper; NEU: der neue Ansatz dieses Papers; WAP: Weighted-All-Pairs-Wrapper; RPC: Klassifikationsbaum mit 'rpart'

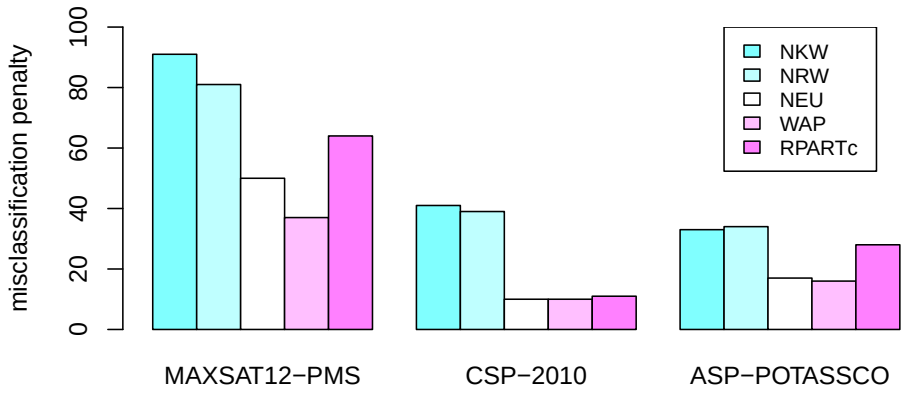

Abbildung 4 : Die *misclassification penalty* für die zweiten drei Szenarien
NKW: Naiver Klassifikations-Wrapper; NRW: Naiver Regressions-Wrapper; NEU: der neue Ansatz dieses Papers; WAP: Weighted-All-Pairs-Wrapper; RPC: Klassifikationsbaum mit 'rpart'

9 Zusammenfassung

Bei vielen Anwendungen, für die Klassifikationsmethoden verwendet werden, treten beobachtungsabhängige Kosten auf. Beobachtungsabhängige Kosten sind Missklassifikationskosten, die von Beobachtung zu Beobachtung verschieden sind. In dieser Arbeit wurden die Grundlagen von Kosten und Kostensensitivität erläutert sowie beobachtungsabhängig kostensensitive Probleme von klassenabhängig kostensensitiven und kosteninsensitiven Problemen abgegrenzt. Im Anschluss wurden zwei Verfahren für beobachtungsabhängig kostensensitive Klassifikation vorgestellt: das *Costing*-Verfahren von Zadrozny et al. ([3]) sowie der *Kostensensitive One-Versus-One-Algorithmus* von Lin ([4]).

Anschließend wurde ein neues, auf Entscheidungsbäumen basierendes, Verfahren vorgestellt. Zunächst wurden dafür als theoretische Grundlage die Entscheidungsbäume, im speziellen der CART-Algorithmus von Breiman ([9]), vorgestellt. Die neue Methode stellt eine neue kostensensitive Splitfunktion für Entscheidungsbäume dar, die durch eine entsprechende Aufteilung eines Knotens die Kosten minimiert.

Als Nächstes wurde diese neue Methode mit einem weiteren Ansatz für beobachtungsabhängig kostensensitive Klassifikationsbäume von Bahnsen et al. ([1]) verglichen. Dieser basiert ebenso wie der neue Ansatz dieser Arbeit auf einem Unreinheitsmaß, das die absoluten Kosten minimieren will. Im Gegensatz dazu ist der Ansatz von Bahnsen et al. allerdings nur für binäre Zielvariablen anwendbar.

Getestet wurde das neue Verfahren daraufhin anhand von zwei Anwendungsbeispielen: zum einen einem Datensatz aus dem Direktmarketing und zum anderen für ASlib, eine Datenbank zur Algorithmenselektion. Für den Direktmarketing-Datensatz wurde der Ansatz mit drei weiteren kostensensitiven Verfahren verglichen: jeweils einem einfachen Klassifikations- und Regressionswrapper sowie dem Weighted-All-Pairs-Wrapper, der, wie an früherer Stelle beschrieben, eine große Ähnlichkeit zum Kostensensitiven One-Versus-One-Algorithmus besitzt. Außerdem konnte er grob mit dem Verfahren von Bahnsen et al. verglichen werden, da auch diese in ihrer Arbeit den Direktmarketing-Datensatz verwendet haben. Gegenüber den beiden einfachen Verfahren schnitt der neue Ansatz deutlich besser ab, gegenüber dem Weighted-All-Pairs-Wrapper und dem Verfahren von Bahnsen et al. allerdings etwas schlechter.

Für die ASlib-Szenarien wurde das neue Verfahren ebenfalls wieder mit den beiden einfachen Ansätzen und dem Weighted-All-Pairs-Wrapper verglichen, zusätzlich nun jedoch auch noch mit den Ergebnissen eines klassischen Klassifikationsbaums mit *rpart*.

Hierfür lieferte der neue Ansatz wiederum deutlich bessere Ergebnisse als die beiden einfachen Verfahren und ebenso deutlich bessere Ergebnisse als der einfache Klassifikationsbaum. Gegenüber dem Weighted-All-Pairs-Wrapper schnitt der Ansatz jedoch auch hier konstant etwas schlechter ab.

Zum Abschluss bleibt zu sagen, dass der neue Ansatz eine sinnvolle Möglichkeit zur beobachtungsabhängig kostensensitiven Klassifikation darstellt. Durch leichte Verbesserungen könnte es auch möglich sein, dass der Ansatz gleich gute, wenn nicht sogar bessere Ergebnisse als der Weighted-All-Pairs-Wrapper erzielen kann. Ein weiterer Punkt, der in Zukunft zusätzlich beachtet werden sollte, ist eine Reduzierung der Laufzeit des Verfahrens im Rahmen einer Vereinfachung der Splitfunktion, insbesondere der für kategoriale Kovariablen. Schließlich noch könnte man sich darüber Gedanken machen, in den Ansatz eine Pruning-Funktion zu integrieren.

10 Literatur

[1] Alejandro Correa Bahnsen, Djamila Aouada, and Björn E. Ottersten. Example-dependent cost-sensitive decision trees. Expert Syst. Appl., 42(19):6609–6619, 2015.

[2] Charles Elkan. The foundations of cost-sensitive learning. In In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, pages 973–978, 2001.

[3] Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting. In Proceedings of the Third IEEE International Conference on Data Mining, ICDM '03, pages 435–, Washington, DC, USA, 2003. IEEE Computer Society.

[4] Hsuan-Tien Lin. Reduction from cost-sensitive multiclass classification to one-versus-one binary classification. In Proceedings of the Sixth Asian Conference on Machine Learning, ACML 2014, Nha Trang City, Vietnam, November 26-28, 2014., 2014.

[5] Alina Beygelzimer, John Langford, and Bianca Zadrozny. Machine Learning Techniques—Reductions Between Prediction Quality Metrics, pages 3–28. Springer US, Boston, MA, 2008.

[6] Alina Beygelzimer, Varsha Dani, Tom Hayes, John Langford, and Bianca Zadrozny. Error limiting reductions between classification tasks. In Proceedings of the 22Nd International Conference on Machine Learning, ICML '05, pages 49–56, New York, NY, USA, 2005. ACM.

[7] Carolin Strobl, James Malley, and Gerhard Tutz. An introduction to recursive partitioning: Rationale, application and characteristics of classification and regression trees, bagging and random forests, 2009.

[8] J. Ross Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[9] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

[10] Bernd Bischl. Implemented performance measures.

[11] Terry Thereneau and Mayo Clinic. User written splitting functions for RPART, 2015.

[12] S. Moro, P. Cortez, and P. Rita. A data-driven approach to predict the success of bank telemarketing. Decision Support Systems, pages 62:22–31, 2014.

[13] Alejandro Correa Bahnsen. Source code for costcla.datasets.base.

[14] Bernd Bischl. Cost sensitive classification: Example-dependent misclassification costs.

[15] Manuel J. A. Eugster. Benchmark experiments. März 2011.

[16] Daniel Wollschläger. Grundlagen der Datenanalyse mit R: eine anwendungsorientierte Einführung. Springer, 2010.

[17] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Thomas Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. Aslib: A benchmark library for algorithm selection. CoRR, abs/1506.02465, 2015.

11 Abbildungsverzeichnis

12 Liste der verwendeten Symbole

- y: die Zielvariable (bzw. auch: die wahre Klasse)
- c_{TN}: die Kosten einer richtig „negativen“ Klassifikation
- c_{FN}: die Kosten einer fälschlich „negativen“ Klassifikation
- c_{FP}: die Kosten einer fälschlich „positiven“ Klassifikation
- c_{TP}: die Kosten einer richtig „positiven“ Klassifikation
- k: die Klasse, wobei $k \in \{1, ..., K\}$
- $L(x, i)$: die erwarteten Kosten einer Beobachtung, wenn als Klasse i vorhergesagt wird
- h: ein (beliebiger) Klassifizierer
- j: die kostenoptimale vorhergesagte Klasse
- $Cost_o(S)$: die minimalen Kosten, wenn für alle Beobachtungen eines Testdatensatzes S dieselbe Klasse k vorhergesagt wird
- (x, y, c): eine kostensensitive Beobachtung bzw. ein kostensensitives Tupel
- D: die ursprüngliche Verteilung der Daten
- $\hat{D}$: die veränderte Verteilung der Daten
- $A(S)$: ein Lernalgorithmus, angewandt auf den Datensatz S
- S_c: ein kostensensitiver Trainingsdatensatz
- S_w: ein gewichteter Trainingsdatensatz
- $E(h, D)$: die erwarteten Kosten eines Klassifizierers h
- $c_c^{(k)}$: der „Klassifikationskostenvektor“
- C_c: die Menge der Klassifikationskostenvektoren
- $S_b^{(i,j)}$: ein binärer Trainingsdatensatz
- $\hat{h}_b^{(i,j)}$: ein binärer Klassifizierer
- S^l: der linke Kindsknoten

- S^r: der rechte Kindsknoten
- Y: die Kostenmatrix
- X: die Prädiktormatrix; enthält die Information über die Kovariablen
- S_k: die aufsummierten Kosten einer Klasse k
- m_k: die mittleren Kosten einer Klasse k
- a: die „beste“ Klasse
- n: die Anzahl an Beobachtungen
- sb: der „single-best“-Klassifizierer; entspricht $Cost_o(S)$; enthält die aufsummierten Kosten der „besten“ Klasse a
- vb: der „virtual-best“-Klassifizierer; berechnet die Kosten, wenn für jede Beobachtung die Klasse mit den geringsten Kosten ausgewählt wird
- $goodness$: Gütemaß: Differenz aus sb und vb
- $I_c(S)$: ein kostensensitives Unreinheitsmaß, angewandt auf S

13 Anhang

Kompletter R-Code

```
# initialization function

asTreeInit = function(y, offset, parms, wt) {
  if (length(offset)) y = y - offset
  sfun = function(yval, dev, wt, ylevel, digits ) {
    paste(" mean=", format(signif(yval, digits)),
          ", MSE=" , format(signif(dev/wt, digits)),
          sep = '')
  }
  environment(sfun) = .GlobalEnv
  list(y = y, parms = NULL, numresp = ncol(y)+1, numy = ncol(y), summary = sfun)
}


# eval function

asTreeEval = function(y, wt, parms) {
  #print("eval")
  cm = colSums(y)
  j = which(cm == min(cm))
  if (length(j) > 1) {
    j = sample(j, 1)
  }
  list(label = c(j,colnames(y)[j], cm), deviance = cm[j])
}

# split function

asTreeSplit = function(y, wt, x, parms, continuous) {
  n = nrow(y)
  goodness = numeric(n-1)
  if (continuous) {
    # colSums for every column - sums up the costs in the parent node
```

```
    yy0 = colSums(y[1:n,,drop=FALSE] * wt[1:n])
    for (i in 1:(n-1)) {
      yy1 = colSums(y[1:i,,drop=FALSE] * wt[1:i])
      yy2 = colSums(y[(i+1):n,,drop=FALSE] * wt[(i+1):n])
      goodness[i] = min(yy0) - min(yy1) - min(yy2)
    }
    list(goodness = goodness, direction = rep(-1, n-1))
  } else {

    # Categorical X variable
    ux = sort(unique(x))
    l = unique(x)
    k = length(l)

    # if the number of levels k is uneven:
    if(k %% 2 == 1) {
      z = (2^(k-1)) - 1
    } else {
      z = (2^(k-1)) - 1 + (gamma(k + 1) / (gamma(k/2 + 1) * gamma(k/2 + 1))/2)
    }

    zmat = matrix(0,nrow=z,ncol=1)
    zmat = as.list(zmat)
    o = 0

    # create combinations of the different levels
    for(i in 1:as.integer(k/2)){
      c = combn(x = l, m = i, simplify = FALSE)
      for(j in 1:length(c)) {
        o = o+1
        zmat[o] = c[j]
      }
    }
    goodness = numeric(length(zmat))
    yy0 = colSums(y[,,drop=FALSE])
```

```
    for(i in 1:length(zmat)) {
      a = which(x %in% zmat[[i]])
      yy1 = colSums(y[a,,drop=FALSE])
      yy2 = colSums(y[-a,,drop=FALSE])
      goodness[i] = min(yy0) - min(yy1) - min(yy2)
    }

    jmax = which(goodness == max(goodness))
    jjmax = jmax[1]
    lz = length(zmat[[jjmax]])

    goodness_x = numeric(k-1)
    goodness_x[lz] = jjmax
    goodness_x
    rest = l[!(l %in% zmat[[j]])]
    direction = c(zmat[[j]],rest)
    list(goodness=goodness_x, direction = direction)
    }
  }

# function to call the three part functions

csrpart = function (features, costs, ...) {
  checkmate::checkDataFrame(features, min.rows = 1,
          min.cols = 1, col.names = "unique")
  checkmate::checkMatrix(costs, any.missing = FALSE,
          min.rows = 1, min.cols = 2, col.names = "unique")

  # substract mincost per row, so that mincost=0 for best action
  cost = t(apply(costs, 1, function(x) x - min(x)))
  dataMat = cbind(features, cost)

  ulist = list(init = asTreeInit, eval = asTreeEval, split = asTreeSplit)
  form = as.formula(paste("cbind(",collapse(colnames(cost), ","), ")~."))
  rpart::rpart(form, data = dataMat, method = ulist, ...)
}
```

```
# plot-function


plot.csrpart = function(mod) {
  mod = as.party(mod)
  plot(mod)
}
```

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

München, den 22.08.2016

_____

Lorenz Haller