Ensemble-Techniken und ordinale Klassifikation

Klaus Hechenbichler

München 2005

Ensemble-Techniken und ordinale Klassifikation

Klaus Hechenbichler

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Klaus Hechenbichler
aus München

München, den 17. Juni 2005

Erstgutachter: Prof. Dr. Gerhard Tutz
Zweitgutachter: Prof. Dr. Helmut Küchenhoff
Externer Gutachter: Prof. Dr. Klaus-Dieter Wernecke
Tag der mündlichen Prüfung: 8. November 2005

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Zusammenfassung

Aufgrund der Entwicklung von aggregierten Methoden wurde in den letzten Jahren eine Reihe neuer Verfahren im Forschungsfeld der Klassifikation und Prädiktion eingeführt. Die beiden wichtigsten Entwicklungen, nämlich Bagging und Boosting, wurden sowohl für den Fall von binären Zielvariablen als auch für den Mehrklassenfall ausführlich diskutiert und analysiert. Während diese angeführten Methoden die Klassenvariable jedoch als nominale Größe ohne Ordnungsstruktur betrachten, kann die Zielvariable in vielen Anwendungen als eine geordnete kategoriale Größe angesehen werden.

In dieser Arbeit sollen deshalb Varianten für Bagging und Boosting entwickelt und vorgestellt werden, die in der Lage sind, die durch die ordinale Struktur in den Daten gegebene Information zu nutzen. Ferner wird aufgezeigt, wie die Qualität der Vorhersagen durch die Verwendung dieser auf die Problemstellung abgestimmten Aggregationsverfahren verbessert wird. Die dazu nötigen empirischen Vergleiche zwischen diversen Klassifikationstechniken werden nicht nur anhand von Fehlklassifikationsraten durchgeführt; stattdessen sollen auch Kriterien, die die Ordinalität in Vorhersage und Zielgröße zu berücksichtigen vermögen, herangezogen werden. Hier spielen in erster Linie verschiedene Abstandsmaße eine Rolle.

Aber auch auf der Basis anderer Techniken und Ansätze kann man versuchen, dem Problem der ordinalen Klassenstruktur zu begegnen: Nächste-Nachbarn-Verfahren, die eine der intuitivsten und einfachsten Methoden zur Klassifikation darstellen, werden in dieser Arbeit durch einige Modifikationen an die besonderen Strukturen von ordinalen Zielgrößen angepaßt. Abschließend können die Resultate dieses zweiten Ansatzes, der ohne großen Rechenaufwand auskommt, mit denjenigen der modernen und rechenintensiven Aggregationstechniken verglichen werden.

Kapitel 1

Einleitung

1.1 Klassifikation und ordinale Zielgrößen

Eine gängige Problemstellung in multivariater Statistik, bei der diverse Verfahren für nominal- und kardinalskalierte Zielgrößen existieren, bilden Modelle zur Vorhersage einer Zielvariable Y anhand einer Menge von Kovariaten $x_1, \ldots, x_p$. Während für die Einflußgrößen (zumindest nach Umkodierung) jedes Skalenniveau zugelassen werden kann, befindet man sich je nach Skalenniveau der abhängigen Variable in einem anderen Feld der statistischen Modellierung: Eine metrische Zielgröße läßt sich mit Hilfe von Regressionsverfahren prognostizieren, deren bekanntestes Beispiel sicherlich die lineare Regression ist, während man sich bei nominalen Zielgrößen im Bereich der Klassifikationsverfahren bewegt. Im zweiten Fall spricht man anstatt von Prognose auch häufig von Klassifikation, denn über die unabhängigen Variablen soll eine Zuordnung der Untersuchungsobjekte in eine von mehreren gleichwertigen nominalen Klassen erfolgen. Über die Jahre wurde eine Vielzahl von Klassifikationsverfahren entwickelt. Zu den wichtigsten Techniken können neben der klassischen linearen Diskriminanzanalyse von Fisher oder dem Logit-Modell auf jeden Fall auch Verfahren gezählt werden, die von ganz anderen Ansätzen ausgehen, wie zum Beispiel Nächste-Nachbarn-Methoden (Fix & Hodges (1951)) oder Klassifikationsbäume (Breiman, Friedman, Olshen & Stone (1984)), die beide im Rahmen dieser Arbeit eine bedeutende Rolle als Basis für Weiterentwicklungen spielen werden. Statistische Klassifikationskonzepte sind beispielsweise gut und umfangreich in den Büchern von McLachlan (1992), Ripley (1996) oder Hastie, Tibshirani & Friedman (2001) zusammengefaßt.

Wie bereits erwähnt werden sogenannte Nächste-Nachbarn-Techniken zu den Standardverfahren der Klassifikation gezählt. In der einfachsten Variante wird dabei ein Datenpunkt der Klasse des bezüglich der Kovariaten ähnlichsten Punktes der Lernstichprobe zugeteilt. Eine Erweiterung, die k-Nächste-Nachbarn-Technik, weist den Datenpunkt dagegen derjenigen Klasse zu, die die Mehrheit der k ähnlichsten Punkte aufweist. Nächste-Nachbarn-Methoden sind von ihrer Grundidee her sehr einfach, aber erstaunlich effektiv und erfolg-

reich in vielen praktischen Aufgabenstellungen, speziell aber bei komplexen Problemen wie der Analyse von Genexpressionsdaten in der statistischen Genetik.

Zwei fortgeschrittene Techniken, die im Feld der gewöhnlichen nominalen Klassifikation in den letzten Jahren neben den Support Vector Machines (Bishop & Tipping (2000)) in der Literatur vielleicht die größte Rolle spielen, sind Bagging (Breiman (1996)) und Boosting (Freund (1995), siehe auch Freund & Schapire (1996)), die beide wiederum unter dem gemeinsamen Oberbegriff *Aggregierte Klassifikationsverfahren* oder *Ensemble-Techniken* zusammengefaßt werden können. Es wurde vielfach nachgewiesen, daß derartige Techniken zu spektakulären Verbesserungen gegenüber den etablierten Standardverfahren führen können. Besonders die Entwicklung von Boosting auf Basis von Klassifikationsbäumen sorgte für intensive Forschungsarbeit im Feld der aggregierten Verfahren. Hier wird als Basis eine gängige Klassifikationstechnik herangezogen und wiederholt auf eine zufällig oder systematisch variierende Datengrundlage zur Modellbildung angewendet. Die einzelnen Resultate werden dann in einem abschließenden Abstimmungsschritt (*Voting*) zu einem Gesamtergebnis kombiniert. Während beim Bagging die einzelnen Unterstichproben lediglich auf dem gewöhnlichen, ungewichteten Bootstrap-Verfahren basieren, werden beim Boosting Gewichte verwendet, die vom Resultat bezüglich der jeweils letzten Unterstichprobe abhängen. Beide Methoden, vor allem aber das im Gegensatz zu Bagging bereits in viele Richtungen weiterentwickelte Boosting, liefern in den meisten Fällen erstaunliche Verbesserungen der Prognosegüte. Sie wurden in der Literatur bereits ausführlich untersucht und auf eine Vielzahl von Problemen und Aufgabenstellungen der Klassifikation von nominalen Zielgrößen angewendet.

In der statistischen Praxis ist man allerdings immer wieder mit Daten konfrontiert, die ordinales Skalenniveau aufweisen, das heißt, eine Ordnung zwischen den verschiedenen Klassen oder Ausprägungen der Variable ist gegeben, aber Abstände oder gar Verhältnisse zwischen diesen Werten sind nicht definiert. Damit ist dieses Skalenniveau zwischen den beiden Extremen von nominalen und kardinalskalierten Daten angesiedelt. Typischerweise liegen ordinale Daten genau dann vor, wenn eine (zumindest theoretisch existierende) metrische Größe nicht präzise gemessen werden soll oder kann, sprich wenn die Mittel und Wege, also ein geeignetes Meßinstrument, fehlen, um das Merkmal präzise zu erheben. Man muß sich dann damit behelfen, daß man auf Kosten der Genauigkeit nur wenige geordnete Ausprägungen vorgibt.

Als Beispiel kann man sich vorstellen, daß anstatt des exakt bestimmten Alters von Versuchsteilnehmern an einem klinischen Experiment lediglich interessiert, in welches Lebensjahrzehnt die jeweilige Person eingruppiert werden kann. Das Niveau der tatsächlich existierenden und leicht zu erhebenden Größe “Alter“ wird also durch Gruppierung in einige wenige Altersintervalle mit dem damit verbundenen Informationsverlust von einer Kardinal- auf eine Ordinalskala reduziert. Hier verzichtet man demnach freiwillig auf die präzisere Information. Ein anderes Beispiel sind dagegen Ratingskalen, wie sie häufig in sozialwissenschaftlichen Fragebögen, aber auch im Feld der Marktforschung vorkommen. Hier könnte der Grad der Zustimmung zu einer Aussage anhand von fünf Antwortvorgaben

gemessen werden, die von “stimme voll und ganz zu“ bis zu “stimme überhaupt nicht zu“ reichen. Auch hier kann man sich eine zugrundeliegende latente (und metrische) Größe “Zustimmung“ vorstellen, die allerdings im Gegensatz zum oben genannten Alter nicht praktisch gemessen werden kann. Man beschränkt sich demnach gezwungenermaßen auf das ordinale Niveau. Ein letztes Beispiel stellt die Anzahl der Jahre dar, die ein Objekt in einer Überlebenszeitstudie verbracht hat, wobei die letzte Kategorie für ein Überleben bis jenseits des Studienendes steht.

Während bei beiden anderen, extremen Skalenniveaus gängige statistische Verfahren für beinahe jede Aufgabenstellung exisiteren, stellt der Umgang mit ordinalen Daten oft ein Problem dar, für das wenig eigenständige Methodik vorliegt. Parametrische ordinale Modelle, die diese Art von Skalenniveau berücksichtigen, und deren Nutzen für die Praxis wurden beispielsweise in Anderson & Philips (1981) und Rudolfer, Watson & Lesaffre (1995) diskutiert. Diesem Mangel an gebräuchlichen Methoden für ordinale Daten wird manchmal damit begegnet, daß man die entsprechenden Variablen einfach uminterpretiert: Liegen sehr viele verschiedene mögliche ordinale Ausprägungen vor, so sieht man die Werte als quasi-stetig an und rechnet mit ihnen, als würde es sich um kardinalskalierte Daten handeln. Bei nur wenigen ordinalen Werten verzichtet man dagegen oft auf die Information, die sich aus der Ordnung der Werte ergibt, und betrachtet sie lediglich als ungeordnete Ausprägungen einer nominalen Größe.

Alternativ kann man aber auch versuchen, unter Zuhilfenahme der oben genannten Verfahren für kardinal- oder nominalskalierte Größen Ansätze zu finden, die die Besonderheit der ordinalen Struktur erfassen und berücksichtigen. Dies wäre insofern wünschenswert, als in der Ordinalität Information verborgen ist, deren Nutzung möglicherweise zur Verbesserung der Präzision von Vorhersagen verwendet werden kann. Bekannte Verfahren sind hier sequenzielle oder kumulative Ansätze, die die ordinale Größe in mehrere binäre (also nominale) Variablen zerlegen, um auf gewöhnliche Klassifikationsverfahren zurückgreifen zu können. Danach werden diese Resultate erst wieder zu einem Gesamtergebnis unter Berücksichtigung der ordinalen Struktur zusammengesetzt. Genaueres zu derartigen Verfahren folgt im Hauptteil dieser Arbeit. Aber auch der umgekehrte Weg, das Arbeiten mit Regressionstechniken für kardinalskalierte Größen, deren stetige Prognose im Nachhinein wieder in ordinale Klassen gruppiert wird, kommt im Rahmen dieser Arbeit zur Anwendung.

Die Problematik der Berücksichtigung von ordinalen Strukturen in der Klassifikation ist also die Hauptmotivation dieser Forschungsarbeit. Von dieser Voraussetzung ausgehend stellt sich die Frage, welche grundlegenden Verfahren auf den Fall von ordinalen Zielgrößen übertragen werden sollen. Es erschien naheliegend, die Anwendungsmöglichkeiten von modernen und erfolgreichen Verfahren wie Bagging oder Boosting für ordinale Zielgrößen zu untersuchen. Diese Kombination von Methoden aus einerseits der Theorie der aggregierten Klassifikationsverfahren und andererseits aus dem Bereich der Nutzung von ordinaler Struktur in der Zielgröße stellt einen vielversprechenden Ansatz zum Umgang mit ordinalen Klassifikationsproblemen dar. Außerdem wird in dieser Arbeit versucht, alternative

Ansätze für ordinale Aufgabenstellungen auf Basis von Nächste-Nachbarn-Verfahren zu entwickeln, die einerseits unkompliziert und intuitiv sind, aber andererseits ebenfalls als sehr effektiv gelten können.

1.2 Aufbau der Arbeit

Im Anschluß an diese kurze Einführung in die Problemstellung wird in Kapitel 2 ein Überblick über den Themenkomplex der Klassifikation sowie eine ausführliche Darstellung der Methoden und Techniken für die Aggregation von Klassifikationsverfahren gegeben. Da, ebenso wie im überwiegenden Anteil der Literatur, Klassifikations- und Regressionsbäume (CART) als Basis-Verfahren herangezogen werden, folgt zunächst eine kurze Beschreibung dieser Methode. Anschließend wird ausführlich auf Bagging, die einfachste Variante eines Abstimmungsverfahrens, sowie auf die wichtigsten Erweiterungen dieser Methode, eingegangen. Den Schwerpunkt dieses Kapitels bilden jedoch Boosting-Verfahren, die ursprünglich aus dem Bereich des maschinellen Lernens in der Informatik stammen und in zahlreichen Varianten für die Statistik weiterentwickelt wurden. Zum Abschluß dieses zweiten Kapitels soll noch ein ausführlicher Überblick über Bagging und Boosting in der Literatur gegeben werden. Hier ist die Entwicklung von immer neuen Ansätzen und Kombinationen von Methoden noch längst nicht abgeschlossen.

Das dritte Kapitel beschäftigt sich mit Ansätzen zur Nutzung von ordinaler Struktur in der abhängigen Variable. Den beiden grundlegenden konzeptionellen Vorgehensweisen, nämlich dem kumulativen sowie dem sequenziellen Ansatz, ist dabei jeweils ein eigener Abschnitt gewidmet. Regressionsbasierte Ansätze sowie der Umgang mit ordinalen Datenstrukturen in der Literatur folgen in je einem weiteren Abschnitt.

Im vierten Kapitel werden schließlich die neuentwickelten Techniken und Algorithmen vorgestellt, die dazu dienen, ordinale Klassifikation mit Unterstützung von aggregierten Klassifikationsverfahren zu betreiben. Alle Varianten werden hier ausführlich vorgestellt und voneinander abgeleitet. Ebenso finden sich schematische Beschreibungen der algorithmischen Umsetzung dieser diversen Ideen.

Kapitel 5 ist dann dem Thema der Nächste-Nachbarn-Verfahren gewidmet. Während zunächst die herkömmlichen Ansätze wie k-Nächste-Nachbarn vorgestellt werden, folgt anschließend die Einführung von Erweiterungen in Form von gewichteten Ansätzen. Da all diese Techniken lediglich zur gewöhnlichen nominalen Klassifikation verwendet werden können, wird zuletzt noch aufgezeigt, wie eine einfache Erweiterung auf ordinale Problemstellungen erreicht werden kann.

Im sechsten Kapitel folgen zunächst empirische Untersuchungen mit Hilfe von Nächste-Nachbarn-Techniken sowie der gängigen Bagging- und Boostingvarianten für gewöhnliche nominale Problemstellungen. Unter Anderem wird dabei auch auf Datensätze aus dem aktuellen Themenbereich der statistischen Genetik eingegangen. Als Schwerpunkt werden

aber hier die neu entwickelten ordinalen Verfahren evaluiert. Dazu wird zunächst noch einmal ein Überblick über die Vielzahl der zu vergleichenden Verfahren gegeben und Überlegungen zu relevanten Gütemaßen zur Beurteilung der Klassifikationsgüte für ordinale Zielgrößen angestellt. Anschließend erfolgt eine kurze Beschreibung aller verwendeten Datensätze sowie eine Übersicht über Ergebnisse, die mit Hilfe der verschiedenen Verfahren erzielt werden.

Zuletzt wird in Abschnitt 7 noch eine Zusammenfassung der Aufgabenstellungen und Resultate sowie ein Ausblick auf mögliche Weiterentwicklungen und offene Fragen, die im Laufe dieser Arbeit aufgeworfen wurden, dargestellt.

Kapitel 2

Aggregierte Klassifikationsverfahren

2.1 Diskriminanzanalyse und Klassifikation

Im Aufgabenfeld der multivariaten Klassifikation nimmt man an, daß jede Untersuchungseinheit aus einer von k Klassen stammt. Diese Klassenzugehörigkeit wird als Ausprägung einer Zufallsvariable Y angesehen. Auf Basis eines Merkmals- oder Beobachtungsvektors x, der aus p Komponenten besteht und sich auf eine spezielle Untersuchungseinheit bezieht, soll eine Zuordnung in die Klassenzugehörigkeit dieses Objektes erfolgen. Dazu ist es nötig, den Zusammenhang zwischen Y und x zu untersuchen, für gewöhnlich anhand der bedingten Verteilung von Y gegeben x. Ein Prädiktor oder Klassifikator, der dieses Ziel erreichen soll, muß dabei aus der Erfahrung mit anderen gleichartigen Untersuchungseinheiten anhand einer Stichprobenziehung gewonnen werden.

Der Merkmalsvektor x kann ebenfalls als Ausprägung eines Zufallsvektors X oder aber als fest vorgegebene Designvariable aufgefaßt werden. Letztere Betrachtungsweise bietet sich vor allem bei geplanten Laborstudien an, bei denen die Ausprägungen von X gezielt gesteuert werden können. Neben diesen beiden Alternativen der Betrachtung der gemeinsamen Verteilung von (Y, X) oder der bedingten Verteilung $Y|x$ sind aber auch Situationen vorstellbar, in denen eine Stichprobe auf Basis der umgekehrten bedingten Verteilung $X|Y = r$ gezogen wird. Will man beispielsweise in einer medizinischen Studie gezielt Kranke mit Gesunden vergleichen, so bietet es sich an, die Probanden so auszuwählen, daß bezüglich des Y-Merkmals "Erkrankungsstatus" in etwa gleich viele Untersuchungseinheiten aus beiden Populationen vorliegen. X wurde dann demzufolge gegeben $Y = r$ realisiert.

Ein ausführlicher Überblick über die verschiedenen Aspekte von Klassifikation, an dem sich auch die folgenden Ausführungen orientieren, wird in Tutz (2000) gegeben.

Die Bayes-Zuordnung Als Grundkonzept zur Klassifikation kann die sogenannte *Bayes-Zuordnung* angesehen werden, die von der Betrachtung der zuletzt geschilderten Verteilung

des Merkmalsvektors X gegeben $Y = r$ ausgeht. Die Wahrscheinlichkeiten

$$P(X_i = x_i|Y = r) \qquad \text{mit } i = 1, \ldots, p$$

können allgemeiner als Dichten

$$f(x_i|Y = r) \qquad \text{mit } i = 1, \ldots, p$$

dargestellt werden, die im stetigen Fall für die gewöhnliche Dichte und im diskreten Fall für die diskrete Dichte (also unmittelbar die Wahrscheinlichkeiten) stehen. Diese Dichten der bedingten Ereignisse werden ebenso wie die absoluten Wahrscheinlichkeiten $P(Y = r)$ für die verschiedenen Klassenzugehörigkeiten als gegeben angenommen. Die letztgenannten Größen werden oft auch als *a priori-Wahrscheinlichkeiten* bezeichnet. Sie geben das Vorwissen über Y an, also wie wahrscheinlich das Vorliegen der jeweiligen Klassenzugehörigkeit ohne Kenntnis der Ausprägungen von X ist.

Liegen nun aber Ausprägungen x des Kovariatenvektors X vor, so soll deren Kenntnis zur Prognose von Y genutzt werden. Aufgabe im Klassifikationsproblem ist es also, eine möglichst optimale Zuordnungsregel

$$\begin{array}{rcl} C(.) : \mathcal{X} & \longrightarrow & \{1, \ldots, k\} \\ x & \longmapsto & C(x) \end{array}$$

zu finden.

Eine vernünftige Zuordnungsregel besteht darin, als Prognose diejenige Klasse r zu wählen, deren *a posteriori-Wahrscheinlichkeit* gegeben x am größten ist:

$$C^*(x) = r \Longleftrightarrow \mathrm{P}(Y = r|x) = \max_j \mathrm{P}(Y = j|x)$$

Diese Regel wird Bayes-Zuordnung genannt, da der Zusammenhang zwischen den beiden bedingten Wahrscheinlichkeiten und der gegebenen absoluten Wahrscheinlichkeit auf dem Satz von Bayes

$$\mathrm{P}(Y = r|x) = \frac{f(x|Y = r)\mathrm{P}(Y = r)}{\sum_{j=1}^{k} f(x|Y = j)\mathrm{P}(Y = j)}$$

basiert.

Aus der Dichte $f(x|Y = r)$ der Merkmale, gegeben die Klassenzugehörigkeit, entsteht dann die sogenannte Mischverteilung der Population

$$f(x) = \sum_{j=1}^{k} f(x|Y = j)\mathrm{P}(Y = j) \quad .$$

Diese Dichte beschreibt die Verteilung des Merkmalsvektors in einer Population, in der die Verteilung der Klassenzugehörigkeiten durch $\mathrm{P}(Y = r)$ gegeben ist. Die Formel bezeichnet man als den Satz von der totalen Wahrscheinlichkeit.

Minimierung der Gesamtfehlerrate Eine solche Prognose auf Basis einer beliebigen Zuordnungsregel muß selbstverständlich nicht immer korrekt sein. Deshalb unterscheidet man eine Reihe von verschiedenen Fehlermaßen:

- Wahrscheinlichkeit für eine Fehlklassifikation, gegeben ein fester Merkmalsvektor x

$$\epsilon(x) = \mathrm{P}(C(x) \neq Y|x) = 1 - \mathrm{P}(C(x) = Y|x)$$

- Verwechslungswahrscheinlichkeit oder individuelle Fehlerrate, also die Wahrscheinlichkeit, ein Objekt aus Klasse r in Klasse s zuzuordnen

$$\epsilon_{rs} = \mathrm{P}(C(x) = s|Y = r) = \int_{x:C(x)=s} f(x|Y = r)dx$$

- Wahrscheinlichkeit für eine Fehlklassifikation, gegeben eine Klassenzugehörigkeit r

$$\epsilon_r = \mathrm{P}(C(x) \neq r|Y = r) = \sum_{s \neq r} \epsilon_{rs}$$

Von besonderem Interesse ist jedoch die Gesamtfehlerrate oder globale Fehlklassifikationswahrscheinlichkeit

$$\epsilon = \mathrm{P}(C(x) \neq Y) = 1 - \mathrm{P}(C(x) = Y) \quad .$$

Sie hängt eng mit den oben genannten Fehlerraten zusammen:

$$\begin{aligned} \epsilon &= \sum_{r=1}^{k} \mathrm{P}(C(x) \neq r|Y = r)\mathrm{P}(Y = r) = \sum_{r=1}^{k} \epsilon_r \mathrm{P}(Y = r) = \sum_{r=1}^{k} \sum_{s \neq r} \epsilon_{rs} \mathrm{P}(Y = r) \\ \epsilon &= \int \mathrm{P}(C(x) \neq Y|x) f(x)dx = \int \epsilon(x) f(x)dx \end{aligned}$$

Anhand der zweiten Gleichung erkennt man, daß eine Minimierung der Gesamtfehlerrate gleichbedeutend mit der Minimierung der bedingten Fehlerraten gegeben x für alle x ist. Diese Fehlerrate wird aber genau dann minimal, wenn die Zuordnung $C(x)$ bei gegebenem x die Wahrscheinlichkeit $\mathrm{P}(C(x) = j|x)$ maximiert. Dies entspricht exakt der Vorgehensweise bei der Bayes-Zordnung C^*. Damit kann man als fundamentale Optimalitätseigenschaft festhalten, daß die Bayes-Zuordnung die Gesamtfehlerrate ϵ minimiert. Die daraus resultierende optimale Gesamtfehlerrate kann auch angegeben werden:

$$\epsilon_{opt} = \int \min_r \left(1 - \mathrm{P}(Y = r|x)\right) f(x)dx$$

Diskriminanzfunktionen Alternativ lassen sich Zuordnungsregeln auch über sogenannte *Diskriminanzfunktionen* beschreiben. Dabei werden jedem Beobachtungsvektor x Scores $d_r(x)$ für $r = 1, \ldots, k$ zugewiesen, die als Maß für die Tendenz der Beobachtung x zur Klassenzugehörigkeit in r angesehen werden können. Ein geeignetes solches Maß ist selbstverständlich die Klassifikationswahrscheinlichkeit $\mathrm{P}(Y = r|x)$ selbst, so daß die Bayes-Regel auch alternativ als

$$C(x) = r \Longleftrightarrow d_r(x) = \max_j d_j(x)$$

dargestellt werden kann. In dieser Darstellung einer Zuordnungsregel können jetzt aber selbstverständlich auch andere Diskriminanzfunktionen als $\mathrm{P}(Y = r|x)$ verwendet werden, was dann zu anderen Klassifikationstechniken führt.

Betrachten wir dazu den Satz von Bayes in der folgenden Form

$$\mathrm{P}(Y = r|x) = \frac{f(x|Y = r)\mathrm{P}(Y = r)}{\sum_{j=1}^{k} f(x|Y = j)\mathrm{P}(Y = j)} = \frac{f(x|Y = r)\mathrm{P}(Y = r)}{f(x)} \quad .$$

Sind jetzt s und r zwei verschiedene Klassen, so ergibt sich eine Reihe von äquivalenten Ungleichungen:

$$\begin{aligned}
& \mathrm{P}(Y = r|x) \leq \mathrm{P}(Y = s|x) \\
\Longleftrightarrow \quad & \frac{f(x|Y = r)\mathrm{P}(Y = r)}{f(x)} \leq \frac{f(x|Y = s)\mathrm{P}(Y = s)}{f(x)} \\
\Longleftrightarrow \quad & f(x|Y = r)\mathrm{P}(Y = r) \leq f(x|Y = s)\mathrm{P}(Y = s) \\
\Longleftrightarrow \quad & \log(\mathrm{P}(Y = r)) + \log(f(x|Y = r)) \leq \log(\mathrm{P}(Y = s)) + \log(f(x|Y = s))
\end{aligned}$$

Demzufolge existieren also mehrere äquivalente Darstellungen der Bayes-Regel, die auf verschiedenen Diskriminanzfunktionen $d_r(x)$ basieren:

- $d_r(x) = \mathrm{P}(Y = r|x)$
- $d_r(x) = \frac{f(x|Y=r)\mathrm{P}(Y=r)}{f(x)}$
- $d_r(x) = f(x|Y = r)\mathrm{P}(Y = r)$
- $d_r(x) = \log(\mathrm{P}(Y = r)) + \log(f(x|Y = r))$

Diese alternativen Darstellungsweisen ein und derselben Zuordnungsregel werden sich vor allem dann als nützlich erweisen, wenn man sich dem Klassifikationsproblem von der praktischen Seite her nähert, wenn also bisher rein theoretisch betrachtete Größen empirisch geschätzt werden müssen.

Die Maximum-Likelihood-Zuordnung Wie man anhand der verschiedenen Diskriminanzfunktionen leicht nachvollziehen kann, treten bei der Bayes-Zuordnung Schwierigkeiten

auf, wenn die a priori-Wahrscheinlichkeiten für die verschiedenen Klassen stark unterschiedlich sind. Weist nämlich eine Klasse eine extrem hohe priori-Wahrscheinlichkeit auf, so wird die globale Fehlklassifikationsrate minimal, wenn die Klassifikation unabhängig von x stets in diese größte Klasse erfolgt. Die Bayes-Regel ist also nicht mehr in der Lage, die Beobachtungen x zur Differenzierung zwischen den Klassenzugehörigkeiten zu nutzen.

In diesem Fall, oder falls überhaupt keine a priori-Wahrscheinlichkeiten vorliegen, kann auf das alternative Verfahren der *Maximum-Likelihood*-Zuordnung zurückgegriffen werden. Der Ansatz entspricht demjenigen der Bayes-Zuordnung mit dem Unterschied, daß pauschal feste und identische a priori-Wahrscheinlichkeiten $\mathrm{P}(Y = r) = 1/r$ für $r = 1, \ldots, k$, also eine a priori-Gleichverteilung, angenommen werden. Da ein solcher konstanter Faktor in den verschiedenen Diskriminanzfunktionen ignoriert werden kann, ergibt sich als Zuordnungsregel

$$C_{ML}(x) = r \Longleftrightarrow f(x|Y = r) = \max_j f(x|Y = j) \quad .$$

Diese Regel besagt also, daß zu gegebenem x jeweils die Klasse r prognostiziert wird, die am meisten für die Beobachtung x spricht. Obwohl dadurch die Fehlerraten nicht optimiert werden, ist eine Betrachtung dieser Zuordnung wie oben geschildert sinnvoll, wenn nichts über die a priori-Verteilung bekannt oder das Vorwissen mit großer Unsicherheit behaftet ist, ebenso aber auch, wenn das Problem einer extrem hohen a priori-Wahrscheinlichkeit das Arbeiten mit der Bayes-Regel unmöglich macht.

Die kostenoptimale Bayes-Zuordnung Dieses Konzept geht davon aus, daß jede Fehlklassifikation Kosten verursacht, die aber für die verschiedenen Fehlermöglichkeiten unterschiedlich ausfallen dürfen. Dazu legt man die Kosten für die Zuordnung eines Objekts aus Klasse i in die Klasse j fest als

$$c(i, j) = c_{ij} \quad ,$$

wobei gelten soll, daß $c_{ij} \geq 0$ ist, mit $c_{ij} = 0$ genau dann, wenn $i = j$ gilt.

Anstatt von Fehlerraten spricht man nun von *Risiken* und unterscheidet zwischen

- bedingtem Risiko gegeben x, also die zu erwartenden Kosten für gegebenes x

$$r(x) = \sum_{i=1}^{k} c_{i,C(x)} \mathrm{P}(Y = i|x)$$

- individuellem Risiko

$$r_{ij} = c_{ij} \mathrm{P}(C(x) = j|Y = i) = c_{ij} \int_{x:C(x)=j} f(x|Y = i)dx$$

- Risiko gegeben Klasse i

$$r_i = \sum_{j=1}^{k} r_{ij}$$

Ebenso wie bei den Fehlerraten ist wiederum ein Gesamtmaß, hier das gesamte Bayes-Risiko, von besonderem Interesse. Dabei handelt es sich um die zu erwartenden Kosten, die sich als

$$R = \mathrm{E}(c_{Y,C(x)}) = \sum_{i=1}^{k} r_i \mathrm{P}(Y = i) = \int r(x) f(x) dx$$

ergeben. Da einer Minimierung des gesamten Bayes-Risikos wiederum die Minimierung der bedingten Risiken für alle x entspricht, erhält man leicht die (optimale) Bayes-Zuordnung mit Berücksichtigung der Kosten:

$$C^*(x) = r \Longleftrightarrow \sum_{i=1}^{k} \mathrm{P}(Y = i|x) c_{ir} = \min_j \sum_{i=1}^{k} \mathrm{P}(Y = i|x) c_{ij}$$

Eine alternative Darstellung erhält man über die Diskriminanzfunktion

$$d_r(x) = -\sum_{i=1}^{k} \mathrm{P}(Y = i|x) c_{ir} \quad .$$

Die Bayes-Zuordnung lautet dann

$$C^*(x) = r \Longleftrightarrow d_r(x) = \max_j d_j(x) \quad .$$

Interessanterweise enthält die kostenoptimale Bayes-Zuordnung die beiden anderen beschriebenen Zuordnungsregeln, nämlich Bayes und ML, als Spezialfälle, die man durch geeignete Wahl der Kostenfunktion erhält:

- Werden die Kosten für jede Art von Fehlklassifikation als konstant angenommen, also $c_{ij} = c$ für $i \neq j$, so resultiert

$$\begin{aligned} C^*(x) = r \quad &\Longleftrightarrow \quad \sum_{i \neq r} \mathrm{P}(Y = i|x) c = \min_j \sum_{i \neq j} \mathrm{P}(Y = i|x) c \\ &\Longleftrightarrow \quad 1 - \mathrm{P}(Y = r|x) = \min_j (1 - \mathrm{P}(Y = j|x)) \\ &\Longleftrightarrow \quad \mathrm{P}(Y = r|x) = \max_j \mathrm{P}(Y = j|x) \quad , \end{aligned}$$

  was genau der gewöhnlichen Bayes-Zuordnung ohne Kosten entspricht.

- Wählt man umgekehrt proportionale Kosten $c_{ij} = \frac{c}{\mathrm{P}(Y=i)}$ für $i \neq j$, so sind die Kosten umso höher, je geringer die a priori-Wahrscheinlichkeit der Klasse ist, aus der das

Objekt stammt. Es ergibt sich

$$\begin{aligned} & C^*(x) = r \\ \iff & \sum_{i \neq r} \mathrm{P}(Y = i|x) \frac{c}{\mathrm{P}(Y = i)} = \min_j \sum_{i \neq j} \mathrm{P}(Y = j|x) \frac{c}{\mathrm{P}(Y = j)} \\ \iff & \sum_{i \neq r} \frac{c f(x|Y = i)}{f(x)} = \min_j \sum_{i \neq j} \frac{c f(x|Y = j)}{f(x)} \\ \iff & f(x|Y = r) = \max_j f(x|Y = j) \quad , \end{aligned}$$

also die ML-Zuordnungsregel.

Geschätzte Zuordnungsregeln In der Praxis muß man davon ausgehen, daß Diskriminanzfunktionen nicht bekannt sind, sondern aus Daten geschätzt werden müssen. Demzufolge ist man darauf angewiesen, mit darauf basierenden, also ebenfalls geschätzten Zuordnungsregeln

$$\hat{C}(x) = r \iff \hat{d}_r(x) = \max_j \hat{d}_j(x)$$

zu arbeiten. Um diese Schätzungen zu erhalten, ist man auf empirische Daten angewiesen. Diese erhält man über die Ziehung einer sogenannten *Lernstichprobe*

$$L = \{(y_i, x_i), i = 1, \ldots, n_L\}$$

von beobachteten Daten, wobei $y_i \in \{1, \ldots, k\}$ die Klassenzugehörigkeit beschreibt und der Vektor $x_i' = (x_{i1}, \ldots, x_{ip})$ die zugehörigen Werte der Kovariaten enthält. Eine geschätzte Zuordnungsregel $\hat{C}(., L)$, die auf einem solchen Lerndatensatz L basiert, partitioniert dann den Raum $\mathcal{X}$ der Kovariaten in der folgenden Form:

$$\begin{aligned} \hat{C}(., L) : \mathcal{X} & \longrightarrow \{1, \ldots, k\} \\ x & \longmapsto \hat{C}(x, L) \end{aligned}$$

Dabei beschreibt $\hat{C}(x, L)$ die vorhergesagte Klasse für eine konkrete Beobachtung x.

Entscheidend für die Schätzungen ist hier die Art der Stichprobenziehung. Die folgenden drei Ziehungsdesigns sind möglich:

- Ziehung der Gesamtstichprobe (y_i, x_i), wobei die einzelnen Beobachtungen als unabhängige Wiederholungen angesehen werden können.
- Ziehung einer nach x geschichteten Stichprobe $y_i^{(x)}$, wobei die unabhängigen y_i für feste x beobachtet werden.
- Ziehung einer nach y geschichteten Stichprobe $x_i^{(r)}$, wobei die unabhängigen Merkmalsvektoren x_i für feste Klassen r gezogen werden.

Die meisten gängigen Verfahren schätzen $\mathrm{P}(Y = r|x)$, woraus man unmittelbar die geschätzten Diskriminanzfunktionen $\hat{d}_r(x) = \hat{\mathrm{P}}(Y = r|x)$ oder $\hat{d}_r(x) = -\sum_i \hat{\mathrm{P}}(Y = i|x)c_{ir}$ erhält. Verfahren, die dieses Ziel zu erreichen suchen, basieren auf einer Gesamt- oder auf einer nach x geschichteten Stichprobe, und werden in einem folgenden Abschnitt über gängige Klassifikationsverfahren vorgestellt. Andere Techniken setzen hingegen auf der Variante der Schätzung von $\hat{d}_r(x) = \hat{f}(x|Y = r)\mathrm{P}(Y = r)$ auf und basieren damit auf den Merkmalsverteilungen pro Klasse. Hierzu sind eine Gesamt- oder eine nach Y geschichtete Stichprobe vonnöten.

Prognosefehler Sei nun mit (Y, x) eine neue Beobachtung gegeben. Für den binären Response Y wird zunächst angenommen, daß er unbekannt ist. Schätzt man die Wahrscheinlichkeit $\pi = \mathrm{P}(Y = 1|x)$, so stellt diese Schätzung $\hat{\pi}$ eine erste Prognose für das Auftreten von $Y = 1$ dar. Eine direkte Prognose $\hat{Y} = 0$ oder $\hat{Y} = 1$ ergibt sich daraus durch Anwendung einer der oben genannten Zuordnungsregeln. Die detailliertere Information über die Genauigkeit der Vorhersage ist aber bereits in $\hat{\pi}$ enthalten.

Beide Arten von Prognosen, also $\hat{\pi}$ und $\hat{Y}$, können mit dem tatsächlichen, wahren Wert Y verglichen werden:

$$\begin{aligned} \mathrm{E}(\hat{\pi} - Y) &= \pi(\hat{\pi} - 1) + (1 - \pi)\hat{\pi} = \hat{\pi} - \pi \quad , \\ \mathrm{E}(\hat{Y} - Y) &= \pi(\hat{Y} - 1) + (1 - \pi)\hat{Y} = \hat{Y} - \pi \quad . \end{aligned}$$

Neben der hier betrachteten Differenz ist aber auch der Absolutbetrag des Abstandes von Interesse, da Abweichungen nach oben und nach unten als gleichartige Fehler angesehen werden sollen. Man erhält für diese zu erwartende absolute Abweichung

$$\mathrm{E}(|Y - \hat{Y}|) = \pi(1 - \hat{Y}) + (1 - \pi)\hat{Y} = \begin{cases} 1 - \pi & \text{für } \hat{Y} = 0 \\ \pi & \text{für } \hat{Y} = 1 \end{cases} .$$

Da $\pi = \mathrm{P}(Y = 1|x)$ und $1 - \pi = \mathrm{P}(Y = 0|x)$ gilt, kann dieser Prognosefehler auch als

$$\mathrm{E}(|Y - \hat{Y}|) = 1 - \mathrm{P}(\hat{C}(x)|x) = \epsilon(x)$$

dargestellt werden und entspricht so der Fehlklassifikationswahrscheinlichkeit, gegeben x. Eine andere Bezeichnung für diesen Fehler lautet *tatsächliche Irrtumsrate*.

Sei jetzt allgemeiner (Y, x) mit unbekanntem $Y \in \{1, \ldots, k\}$ eine neue Beobachtung. Die Zielgröße wird also nicht mehr auf den binären Fall beschränkt. $Y' = (Y_1, \ldots, Y_k)$ wird in diesem Fall vektoriell kodiert, wobei $Y_r = 1$ gilt, falls $Y = r$, und umgekehrt $Y_r = 0$, falls $Y \neq r$ ist. Auf dieselbe Art und Weise wird auch der zugehörige Prognosevektor $\hat{Y}' = (\hat{Y}_1, \ldots, \hat{Y}_k)$ kodiert.

Dann gilt für die zu erwartende absolute Abweichung

$$\begin{aligned}
\mathrm{E}\left(\sum_{r=1}^{k}|Y_r-\hat{Y}_r|\right) &= \sum_{r=1}^{k}\mathrm{E}(|Y_r-\hat{Y}_r|) \\
&= \sum_{r=1}^{k}\left(\pi_r|1-\hat{Y}_r|+(1-\pi_r)|\hat{Y}_r|\right) \\
&= (1-\pi_{\hat{Y}})+\sum_{r\neq\hat{Y}}\pi_r \\
&= 2(1-\pi_{\hat{Y}}) \\
&= 2(1-\mathrm{P}(\hat{C}(x)|x)) \quad .
\end{aligned}$$

Damit ergibt sich bis auf den Faktor 2 das identische Ergebnis wie im binären Fall. Dieser Unterschied ist darauf zurückzuführen, daß über alle Kategorien summiert wird und nicht wie im dichotomen Fall nur über die erste Klasse.

Insgesamt läßt sich daher festhalten, daß sich die tatsächliche Fehlerrate (oder äquivalent die Fehlklassifikationswahrscheinlichkeit) bis auf einen konstanten Faktor als erwartete absolute Abweichung ergibt:

$$\epsilon(x) = 1-\mathrm{P}(\hat{C}(x)|x)$$

Als Gütemaß für einen Klassifikator $\hat{C}$ bei zufälligem (Y,x) kann

$$\epsilon = \int \epsilon(x)f(x)dx = \mathrm{E}_x(\epsilon(x))$$

verwendet werden. Da die tatsächliche Fehlerrate für eine fest gegebene Zuordnungsregel gilt, muß zur Beurteilung des gesamten Verfahrens, also der Schätzung der Zuordnungsregel und der resultierenden Treffsicherheit, bedacht werden, daß die Zuordnungsregel auf einer Stichprobe beruht. Deshalb muß $\epsilon(x)$ sinnvollerweise durch den Erwartungswert $\mathrm{E}_S(\epsilon(x))$ über die Stichprobenverteilung ersetzt werden.

Zur endgültigen Bestimmung der tatsächlichen Fehlerrate muß schließlich noch die unbekannte a posteriori-Wahrscheinlichkeit geschätzt werden. Ausgehend von einer Gesamtstichprobe (Y_i,x_i) mit $i=1,\ldots,n$ bestimmt man die Zuordnungsschätzungen $\hat{y}'(x_i) = (\hat{y}_1(x_i),\ldots,\hat{y}_k(x_i))$ mit

$$\hat{y}_r(x_i) = \begin{cases} 1 & \text{wenn } \hat{C}(x_i)=r \\ 0 & \text{sonst} \quad . \end{cases}$$

Nun ist es naheliegend, die daraus resultierenden geschätzten Wahrscheinlichkeiten einfach in die Formel für den Fehler einzusetzen. Man spricht dann auch von einer *plug-in*-Schätzung der Fehlerrate über

$$\epsilon_{PI} = \frac{1}{n}\sum_{i=1}^{n}(1-\hat{\mathrm{P}}(\hat{C}(x_i)|x_i)) \quad .$$

Eine Alternative ist die sogenannte *Resubstitutionsfehlerrate*

$$\epsilon_R = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\sum_{r=1}^{k}|y_r(x_i) - \hat{y}_r(x_i)| \quad ,$$

bei der lediglich gezählt wird, wieviele Beobachtungen aus der Lernstichprobe falsch klassifiziert werden. Problem ist hierbei, daß die Daten der Stichprobe zweimal herangezogen werden, sowohl zur Schätzung der Zuordnungsregel, als auch zur Beurteilung ihrer Güte. Deshalb zeigt sie zwar auf, wie gut die konkrete Stichprobe durch diese Regel trennbar ist, kann aber keine Hinweise darauf liefern, wie gut Prognosen für zukünftige, neue Stichproben sind. Allgemein kann von einer systematischen Unterschätzung dieses Generalisierungsfehlers ausgegangen werden.

Um diese Verzerrung zu vermeiden, greift man zu Methoden, die in der Lage sind, fiktive neue Beobachtungen aus der Lernstichprobe zu extrahieren. Ein klassischer Ansatz ist hier die rechenintensive *Kreuzvalidierung*. Dabei wird jeweils eine Beobachtung aus der Lernstichprobe gestrichen und die Zuordnungsregel anhand der verbliebenen Beobachtungen bestimmt. Anschließend kann für diese entfernte Beobachtung die Prognose bestimmt und festgestellt werden, ob korrekt klassifiziert wurde. Zusammengefaßt ergibt sich so

$$\epsilon_{CV} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\sum_{r=1}^{k}|y_r(x_i) - \hat{y}_r^{-i}(x_i)|$$

als Schätzung der Fehlerrate, wobei $\hat{y}_r^{-i}(x_i)$ die Prognose darstellt, die aus der Lernstichprobe ohne die i-te Beobachtung geschätzt wurde.

Zum Abschluß dieser allgemeinen Betrachtungen sei noch auf die entsprechenden Formeln für Resubstitutions- und Kreuzvalidierungsfehlerrate hingewiesen, die aus einer bezüglich der Klassen von Y geschichteten Stichprobe resultieren:

$$\begin{aligned}
\epsilon_R &= \sum_{s=1}^{k}\mathrm{P}(Y=s)\left(\frac{1}{n_s}\frac{1}{2}\sum_{r=1}^{k}|y_r(x_i^{(s)}) - \hat{y}_r(x_i^{(s)})|\right) \\
\epsilon_{CV} &= \sum_{s=1}^{k}\mathrm{P}(Y=s)\left(\frac{1}{n_s}\frac{1}{2}\sum_{r=1}^{k}|y_r(x_i^{(s)}) - \hat{y}_r^{-i}(x_i^{(s)})|\right)
\end{aligned}$$

Für die hier benötigten a priori-Wahrscheinlichkeiten müssen in der Praxis geeignete Schätzungen herangezogen werden.

2.2 Wahl der Basis-Methode

Im Folgenden wird ein kurzer Überblick über die gängigsten Verfahren gegeben, die das Ziel der Vorhersage von Klassenzugehörigkeiten auf teilweise sehr unterschiedlichem Weg zu erreichen suchen.

2.2.1 Überblick über gängige Klassifikationsverfahren

Klassische Diskriminanzanalyse

Die klassische Diskriminanzanalyse geht von multivariat normalverteilten Einflußgrößen innerhalb der Klassen aus. Für die Dichten soll also gelten

$$f(x|Y = r) \sim \mathrm{N}(\mu_r, \Sigma_r) \quad .$$

Setzt man diese Verteilungsannahme in die Bayes-Regel ein und vereinfacht die entstehenden Terme, so erhält man als Diskriminanzfunktionen

$$d_r(x) = -\frac{1}{2}(x - \mu_r)'\Sigma_r^{-1}(x - \mu_r) - \frac{1}{2}\log(\det\Sigma_r) + \log\left(\mathrm{P}(Y = r)\right)$$

mit $r = 1, \ldots, k$. Je nachdem, welche Annahmen nun über die Kovarianzmatrizen Σ_r getroffen werden, ergeben sich unterschiedliche Zuordnungsregeln:

- Für unabhängig standardisierte Merkmale mit $\Sigma_r = \sigma^2 I$ ergibt sich die Diskriminanzfunktion
$$d_r(x) = -\frac{(x - \mu_r)'(x - \mu_r)}{2\sigma^2} + \log\left(\mathrm{P}(Y = r)\right) \quad ,$$
die sich unter der zusätzlichen Annahme von identischen a priori-Wahrscheinlichkeiten vereinfacht zu
$$d_r(x) = -(x - \mu_r)'(x - \mu_r) \quad .$$
Die Entscheidungsregel beschreibt dann eine Minimum-Distanz-Klassifikation: Ordne ein neues Objekt derjenigen Klasse zu, deren Klassenmittel μ_r die geringste euklidische Distanz zu x aufweist.

- Bei Merkmalen mit klassenweise identischen Kovarianzmatrizen $\Sigma_r = \Sigma$ erhält man
$$d_r(x) = -\frac{1}{2}(x - \mu_r)'\Sigma^{-1}(x - \mu_r) + \log\left(\mathrm{P}(Y = r)\right)$$
als Diskriminanzfunktion. Auch dieser Ausdruck läßt sich unter der zusätzlichen Annahme von identischen a priori-Wahrscheinlichkeiten vereinfachen zu
$$d_r(x) = -(x - \mu_r)'\Sigma^{-1}(x - \mu_r)$$
Dies ist eine verallgemeinerte Minimum-Distanz-Regel, die auf Mahalanobis-Abständen beruht, und zugleich die Grundlage der meisten Algorithmen für lineare Diskriminanzanalyse in diversen Programmpaketen.

- Der komplizierteste Fall liegt dann vor, wenn von klassenweise verschiedenen Kovarianzmatrizen ausgegangen werden muß. Hier ist keine Vereinfachung der Diskriminanzfunktion möglich. Man spricht hier auch von quadratischer Diskriminanzanalyse, da im Gegensatz zu den beiden oberen Fällen die Diskriminanzfunktionen nicht mehr als lineare Funktionen in x dargestellt werden können.

Interessanterweise ist diese Methode äquivalent zum älteren Ansatz von Fisher, der ohne Verteilungstyp auskommt und auf Linearkombinationen der unabhängigen Variablen beruht. Dies zeigt bereits, daß die klassische Analyse relativ robust gegenüber Verletzung der Verteilungs- und Kovarianzannahmen sein muß, weshalb man sich meist auf die einfachere Variante von linearen Diskriminanzfunktionen stützen kann.

Loglineare und Logit-Modelle

Liegen ausschließlich kategoriale Merkmale als Kovariaten vor, so ist die Normalverteilungsannahme der klassischen Diskriminanzanalyse definitiv nicht angebracht. Weist das Merkmal x_i genau m_i verschiedene Ausprägungen auf ($i = 1, \ldots, p$), so ist stattdessen eine Multinomialverteilung mit

$$m = \prod_{i=1}^{p} m_i$$

möglichen Ausprägungen die geeignete Verteilung zur Modellierung von x. Ein solches volles multinomiales Modell ist zwar theoretisch genauso zur Bestimmung einer Bayes-Zuordnung geeignet, weist aber für die Praxis zu viele zu schätzende Parameter auf, nämlich insgesamt $km-1$ Wahrscheinlichkeiten $\mathrm{P}(x|Y = r)$. Für eine zuverlässige Schätzung wären hier immense Stichprobenumfänge vonnöten. Um diese Schwierigkeit zu vermeiden, kann ein Modell unabhängiger Merkmale herangezogen werden, die Kovariaten $x_1, \ldots, x_p$ werden also als bedingt unabhängig, gegeben $Y = r$, angenommen. Dieser Ansatz weist deutlich weniger zu schätzende Parameter auf, muß allerdings in der Praxis meist als viel zu einschränkend angesehen werden.

Als geeigneter Mittelweg erscheint es sinnvoll, allgemeinere Modellfamilien zu betrachten. Es soll sich um parametrische Klassifikationsmodelle von verschiedenster Komplexität handeln, die die beiden Extrema, nämlich volles und Unabhängigkeitsmodell, als Spezialfälle mit einschließen. Eine Vereinfachung des vollen Modells entsteht dabei durch gezieltes Weglassen von Interaktionen höherer Ordnung, bis man schließlich beim Unabhängigkeitsmodell landet, das keine Interaktionen mehr aufweist. Oft beschränkt man sich bei der Modellwahl aufgrund der Vielfalt von möglichen Modellen auf eine solche hierarchische Vorgehensweise. Zwei gängige konkrete Ansätze für derartige Modellfamilien sind:

- loglineare Modelle, bei denen logarithmierte erwartete Häufigkeiten $m(x|Y = r)$ der Merkmalskombinationen über

  $$\log\left(m(x|Y = r)\right) = \nu^{(r)} + \nu_1^{(r)} x_1 + \cdots + \nu_p^{(r)} x_p + \nu_{12}^{(r)} x_1 x_2 + \cdots + \nu_{12\ldots p}^{(r)} x_1 x_2 \ldots x_p$$

  modelliert werden. Daraus können Schätzungen der für die Diskriminanzfunktionen nötigen bedingten Klassenwahrscheinlichkeiten über den Zusammenhang

  $$\mathrm{P}(x|Y = r) = \frac{m(x|Y = r)}{n_r}$$

bestimmt werden, wobei n_r für den Stichprobenumfang in der r-ten Klasse steht. Alle verwendeten Merkmale x_i dürfen entweder von Natur aus nur zwei Kategorien aufweisen oder müssen dummy-kodiert in das Modell eingehen.

- das Logit-Modell, bei dem ein linearer Ansatz für den Zusammenhang zwischen den kategorialen Merkmalen x und dem logarithmierten Verhältnis der Wahrscheinlichkeiten zweier Klassen

$$\log\frac{\mathrm{P}(Y=i|x)}{\mathrm{P}(Y=j|x)} = \log\frac{n_j}{n_i}\frac{\mathrm{P}(Y=i)}{\mathrm{P}(Y=j)} + \lambda + \lambda_1 x_1 + \cdots + \lambda_p x_p + \lambda_{12} x_1 x_2 + \cdots + \lambda_{12\ldots p} x_1 x_2 \ldots x_p$$

gewählt wird. Die gesuchten Wahrscheinlichkeiten $\mathrm{P}(Y=i|x)$ sind also bereits unmittelbar in der Modellgleichung enthalten. Da im Gegensatz zum loglinearen Modell beide Klassen (in Form des Quotienten der Klassenwahrscheinlichkeiten) in einer Modellgleichung auftreten, kann hier eine noch geringere Zahl von zu schätzenden Parametern erzielt werden.

Zwischen den Parameterschätzungen dieser beiden Modellklassen (und damit zwischen den Modellklassen insgesamt) besteht ein enger Zusammenhang. Das Logit-Modell besitzt jedoch den Vorteil, daß es im Rahmen der logistischen Modelle erweiterbar auf beliebiges (auch gemischtes) Skalenniveau der Merkmale ist.

Logistische und Lokalisationsmodelle

Da in der praktischen Anwendung fast immer gemischt kategorial und stetige Variablen gemeinsam auftreten, und man sich weder auf die Robustheit der klassischen Diskriminanzanalyse verlassen noch den Informationsverlust einer Kategorisierung von stetigen Größen in Kauf nehmen will, stellen die folgenden beiden Modelle für gemischte Variablen eine wichtige Erweiterung dar.

Das Lokalisationsmodell betrachtet dazu die bedingte Verteilung der stetigen Größen $x = (x_1, \ldots, x_p)'$, gegeben die dichotomen $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_q)'$ Merkmale sowie die Klassenvariable Y, wobei für jede solche bedingende Merkmalskombination eine Normalverteilung der stetigen Größen angenommen wird. Dabei soll nur der Mittelwert $\mu(\tilde{x}, Y)$ von der jeweiligen Merkmalskombination abhängen, die Kovarianzmatrix Σ dagegen nicht:

$$(x|\tilde{x}, Y) \sim \mathrm{N}_p\left(\mu(\tilde{x}, Y), \Sigma\right)$$

Bezeichnet dann $\mathrm{P}(\tilde{x}|Y=r)$ die Auftretenswahrscheinlichkeit der Merkmalskombinationen von x in der Klasse r, so läßt sich die gemeinsame Verteilung $f((\tilde{x}, x)|Y=r)$ darstellen als

$$f((\tilde{x}, x)|Y=r) = f(x|(\tilde{x}, Y=r))P(\tilde{x}|Y=r) \quad .$$

So entsteht für jede solche Kombination der Merkmale $\tilde{x}$ eine eigene Diskriminanzfunktion, insgesamt also 2^q pro Klasse r. Auf diese Art und Weise kann der Ansatz als natürliche Erweiterung der klassischen Diskriminanzanalyse mit Normalverteilungsannahme und

gleichen Kovarianzmatrizen angesehen werden, wobei dort noch keine bedingenden Merkmalskombinationen $\tilde{x}$, also auch nur eine Diskriminanzfunktion pro Klasse r, zum Tragen kommen.

Logistische Modelle nähern sich den gemischten Variablen dagegen nicht von der Seite der stetigen, sondern vielmehr aus Richtung der dichotomen Ansätze und können als Erweiterung der Logit-Modelle angesehen werden. Das allgemeine lineare logistische Modell für k Klassen modelliert die a posteriori-Wahrscheinlichkeiten über

$$\begin{aligned} \mathrm{P}(Y = r|x) &= \frac{\exp(\beta_{0r} + \beta_r' x)}{1 + \sum_{l=1}^{k-1} \exp(\beta_{0l} + \beta_l' x)} \qquad \text{für } r = 1, \ldots, k-1 \\ \mathrm{P}(Y = k|x) &= \frac{1}{1 + \sum_{l=1}^{k-1} \exp(\beta_{0l} + \beta_l' x)} \quad . \end{aligned}$$

Dieser Modellansatz entspricht genau einer linearen Modellierung des logarithmierten Verhältnisses der Klassendichten

$$\log \frac{f(x|Y = r)}{f(x|Y = k)} = \beta_{0r}^* + \beta_r' x \qquad \text{für } r = 1, \ldots, k-1 \quad .$$

Diese Art der Modellierung ist für eine ganze Reihe von Verteilungsannahmen bezüglich der Klassendichten geeignet. Dazu gehören neben Normalveteilungen mit gleichen Kovarianzmatrizen oder diskreten Verteilungen mit in loglinearer Form identischen Interaktionstermen aber auch gemeinsame Verteilungen von stetigen und diskreten Merkmalen mit derartigen Randverteilungen, worin die besondere Bedeutung dieses Ansatzes liegt. Die Zuordnung erfolgt dann mit Hilfe der Diskriminanzfunktionen

$$d_r(x) = \beta_{0r}^* + \beta_r' x \qquad \text{für } r = 1, \ldots, k-1 \quad ,$$

wobei Klasse $l \neq k$ prognostiziert wird, wenn

$$d_l(x) \geq d_r(x) \qquad \text{für alle } r \neq k$$

und zugleich

$$d_l(x) \geq 0$$

gilt. Klasse k wird dagegen prognostiziert, falls

$$d_r(x) < 0 \qquad \text{für } r = 1, \ldots, k-1$$

gilt.

Nächste-Nachbarn-Verfahren

Diese nichtparametrischen Ansätze basieren darauf, eine neue Untersuchungseinheit in diejenige Klasse zuzuordnen, die das Objekt aus der Lernstichprobe aufweist, das bezüglich

der betrachteten Merkmale dem neu zu klassifizierenden Objekt am ähnlichsten ist. Diese Ähnlichkeit wird anhand von Abstandsmaßen ermittelt, die im von den Merkmalen aufgespannten Raum zur Anwendung kommen. Als Variante dazu kann man auch die k nächsten Nachbarn betrachten und dann diejenige Klasse wählen, der die meisten dieser k Nachbarn angehören. Da das Verfahren im Rahmen dieser Arbeit weiterentwickelt wird, folgt eine detailliertere und ausführlichere Beschreibung im fünften Kapitel, das sich speziell mit Nächste-Nachbarn-Techniken und deren Erweiterungen auseinandersetzt.

Klassifikationsbäume

Hier wird Schritt für Schritt ein binärer Baum gebildet, wobei an jedem Knoten diejenige Variable mit demjenigen Trennpunkt verwendet werden, die gemeinsam zur optimalen Partitionierung führen. Dies bedeutet, daß die beiden Tochterknoten jeweils bezüglich der Klassenzugehörigkeit in sich möglichst homogen und untereinander möglichst heterogen sind. Diese sukzessiv dichotomisierende Partitionierung des Merkmalsraums wird durchgeführt, bis sich in jedem entstandenen Endknoten nur noch eine bestimmte festzulegende Höchstzahl von Einheiten der Lernstichprobe befinden. Den Endknoten wird dann jeweils die am häufigsten auftretende Klassenzugehörigkeit als Prognose zugeordnet. Diese riesigen und unübersichtlichen Bäume werden häufig rückwärts wieder um diejenigen Knoten beschnitten, die bezüglich der Fehlklassifikationsraten keine signifikante Verbesserung mit sich bringen, wobei die Komplexität des Baumes in Form eines Strafterms mit eingeht. Auf diese Weise entstehen gut interpretierbare Bäume, bei denen der Weg der Klassifikation unmittelbar von der Wurzel bis zum letztendlichen Endknoten verfolgt werden kann. Auch hierzu folgt eine detailliertere Beschreibung des Verfahrens in einem späteren Abschnitt, da die Wahl eines geeigneten Basis-Verfahrens für aggregierte Ansätze aus mehreren Gründen letztendlich auf diese Methode fiel.

Weitere Klassifikationsverfahren

Ebenso wäre es möglich, andere Klassifikationsverfahren als Basis für aggregierte Methoden heranzuziehen. Insbesondere neuronale Netze, die ähnlich gut geeignete Eigenschaften wie Klassifikationsbäume aufweisen, wären hier vorstellbar. Aufgrund der einfachen Struktur und der klaren Interpretierbarkeit eines binären Baumes wurde aber auf die Anwendung von komplexeren und unanschaulichen Verfahren wie neuronale Netze verzichtet.

2.2.2 Klassifikationsbäume

Als Basis, auf die die in den folgenden Abschnitten vorgestellten Aggregationsverfahren aufsetzen können, benötigt man ein herkömmliches Klassifikationsverfahren. In dieser Arbeit werden dazu wie bereits erwähnt Klassifikationsbäume oder CARTs (*Classification*

And Regression Trees) herangezogen. Neben den für Aggregationstechniken günstigen Eigenschaften dieses Verfahrens, wie Instabilität und damit starke Variabilität der Resultate, spielte für diese Entscheidung in erster Linie eine Rolle, daß sowohl Bagging als auch Boosting in der Literatur primär in Verbindung mit diesem Klassifikationsverfahren auftauchen und damit ein unmittelbarer Vergleich mit Ansätzen anderer Autoren möglich ist.

Abbildung 2.1 stellt einen typischen Klassifikationsbaum dar: Beginnend in der Wurzel durchläuft eine zu klassifizierende Untersuchungseinheit den Baum von oben nach unten, bis sie schließlich einen Endknoten erreicht. Dort läßt sich die prognostizierte Klasse (hier das Risiko für eine fiktive Erkrankung) unmittelbar ablesen. Diese einfache und übersichtliche Möglichkeit, den Klassifikationsvorgang graphisch darzustellen, kann als einer der Hauptgründe für die Beliebtheit dieser Methode angesehen werden. Eine solche Abbildung kann auch interessierten Personen ohne grundlegende Kenntnisse von Statistik vorgelegt und verständlich gemacht werden.

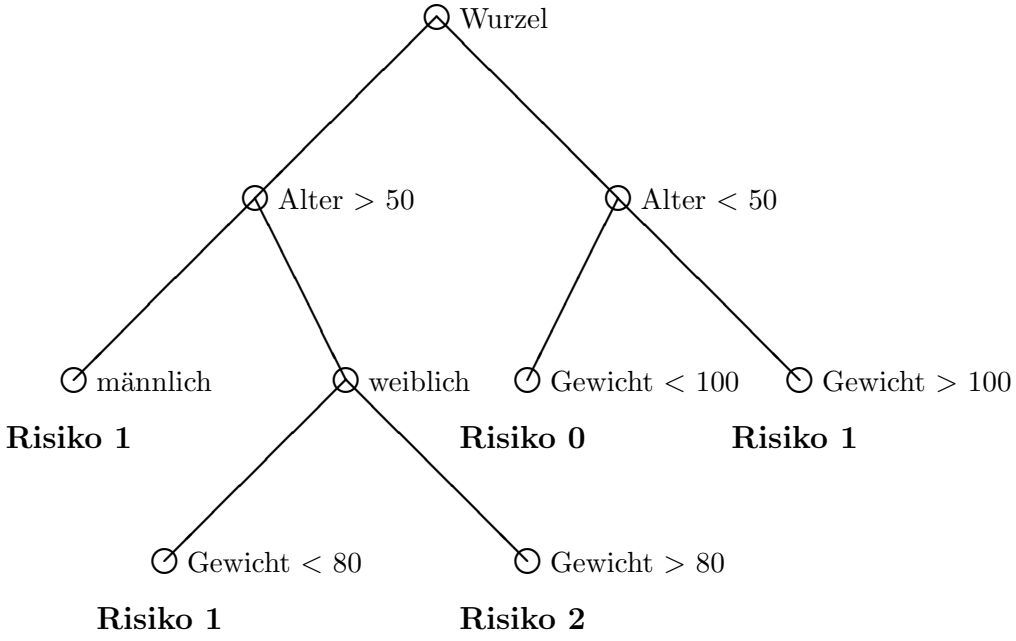

Abbildung 2.1: Beispiel für einen Klassifikationsbaum

Für Berechnungen und Algorithmen am Computer wurde zunächst das *tree*-Package des Programmpaketes *R* verwendet. Später folgte dann eine Umstellung auf das etwas schnellere und flexiblere *rpart*-Package. Beide Funktionen basieren aber auf der prinzipiell identischen Vorgehensweise.

Grundidee des Verfahrens ist es, den Merkmalsraum sukzessive dichotom zu unterteilen, mit der Zielvorgabe, daß die so entstehenden Partitionen bezüglich der Klassenzugehörigkeit möglichst homogen und untereinander möglichst heterogen sind. Diese unkomplizierte

rekursive Vorgehensweise hat den Vorteil, daß keine limitierenden Modell- oder Verteilungsannahmen benötigt werden.

Bei der Entscheidung, welche Aufteilung des Merkmalsraums A_0 die optimale ist, wird gleichzeitig nach einem Merkmal und einem zugehörigen Trennpunkt gesucht. Anders ausgedrückt wird hier dasjenige Merkmal benötigt, das in dichotomisierter Form den Lerndatensatz am besten bezüglich der Klassenzugehörigkeit der Objekte trennen kann. Ist dieser Schritt erfolgt, so wird in den beiden entstandenen Partitionierungen A_1 und A_2, die in der graphischen Darstellung durch die Verzweigungsstellen oder Knoten symbolisiert sind, jeweils genauso vorgegangen: Wieder wird nach der optimalen Partitionierung anhand eines Merkmals gesucht, die in dieser Subgruppe der Lernstichprobe zur besten Trennung der Untersuchungseinheiten bezüglich ihrer Klassenzugehörigkeit führt. Aus der Partition A_1 werden so die Subgruppen A_3 und A_4 gebildet, analog aus A_2 die Untergruppen A_5 und A_6. Auf diese Art und Weise wird sukzessive weiter vorgegangen, bis für alle entstandenen Subgruppen ein Abbruchkriterium erreicht ist. Sie stellen dann die Endknoten des resultierenden Baumes dar. Diese Vorgehensweise wird oft auch naheliegenderweise als rekursive Partitionierung des Merkmalsraums bezeichnet.

Die daraus resultierende Schätzung der Klassenzugehörigkeit erfolgt unmittelbar nach der Bayes-Regel. Für eine Untermenge A des Merkmalsraums sind die a posteriori-Schätzer der Klassenwahrscheinlichkeiten über

$$\hat{\mathrm{P}}(Y = r|x \in A) = \frac{\mathrm{P}(Y = r)n_r(A)/n_r}{\sum_{i=1}^{k} \mathrm{P}(Y = i)n_i(A)/n_i}$$

gegeben. Dabei ist n_i die Anzahl von Klasse-i-Objekten in der Lernstichprobe und $n_i(A)$ die Anzahl von Beobachtungen aus der Submenge A mit Klasse i. Schätzt man nun noch die a priori-Wahrscheinlichkeiten über

$$\hat{\mathrm{P}}(Y = i) = \frac{n_i}{n} \quad ,$$

wobei n den gesamten Lernstichprobenumfang darstellt, so erhält man die einfache Form

$$\hat{\mathrm{P}}(Y = r|x \in A) = \frac{n_r(A)}{n(A)}$$

mit $n(A) = n_1(A) + \cdots + n_k(A)$. Die Schätzung für die a posteriori-Klassenwahrscheinlichkeiten ist also nichts anderes als die empirische prozentuale Verteilung der Klassenzugehörigkeit im jeweiligen Knoten.

Aus diesen Wahrscheinlichkeiten ergibt sich unmittelbar die Klassenzuordnung über die Diskriminanzfunktion

$$d_T(x) = r \qquad \text{falls } \hat{\mathrm{P}}(Y = r|x \in A_t) = \max_i \left(\hat{\mathrm{P}}(Y = i|x \in A_t)\right) \qquad \text{für } x \in A_t, t \in \tilde{T} \quad ,$$

wobei T für den gesamten Baum und $\tilde{T}$ für die Menge aller Endknoten von T steht.

Soll eine Subgruppe (bzw. ein Knoten) A des Baumes weiter partitioniert werden, so muß ein dazu verwendetes Verzweigungskriterium einige Anforderungen erfüllen:

- Die Verzweigung darf nur auf einem Merkmal x_i basieren.
- Ist dieses Merkmal x_i ordinal oder kardinal skaliert, so kommt jede Verzweigung der Form
$$A \cap \{x_i \leq c\} \quad \text{und} \quad A \cap \{x_i > c\}$$
in Frage, wobei $c \in \mathbb{R}$ den frei zu wählenden Trennpunkt darstellt.
- Ist das Merkmal $x_i \in \{1, \ldots, k_i\}$ dagegen nominal skaliert, so kommt jede Verzweigung der Form
$$A \cap S \quad \text{und} \quad A \cap \bar{S}$$
mit $S \subset \{1, \ldots, k_i\}$ in Frage, was einer Zerlegung in zwei beliebige Teilmengen des Wertebereichs von x_i entspricht.

Eine Möglichkeit, die Verzweigungen und damit das Wachstum eines Klassifikationsbaumes zu steuern, basiert auf sogenannten Unreinheitsfunktionen. Dies sind Funktionen $\phi(\pi)$ mit $\pi = (\pi_1, \ldots, \pi_k)$, $\pi_i \geq 0$ und $\sum_i \pi_i = 1$, die die folgenden Eigenschaften aufweisen müssen:

- $\phi\left(\frac{1}{k}, \ldots, \frac{1}{k}\right) = \max_\pi \phi(\pi)$
- $\phi(0, \ldots, 0, 1, 0, \ldots, 0) = \min_\pi \phi(\pi)$
- ϕ ist symmetrisch in π

Wichtige Beispiele für solche Unreinheitsfunktionen sind:

- Die Entropie $\phi(\pi) = -\sum_i \pi_i \log(\pi_i)$
- Der Gini-Index $\phi(\pi) = \sum_{i \neq j} \pi_i \pi_j$
- Die Resubstitutionsfehlerrate $\phi(\pi) = 1 - \max_r \pi_r$

Basierend auf diesen Funktionen kann nun die Unreinheit des Knotens A eines Baumes T berechnet werden als

$$R(A) = \phi(\hat{\mathrm{P}}(Y = 1 | x \in A), \ldots, \hat{\mathrm{P}}(Y = k | x \in A)) \quad ,$$

also anhand der geschätzten Zuordnungswahrscheinlichkeiten in die k Klassen, wie sie in dem jeweiligen Knoten A gelten. Daraus läßt sich eine Distanz zwischen den beiden Tochterknoten A_1 und A_2 von A definieren, auf deren Maximierung die Entscheidung für eine optimale Verzweigung basieren soll. Sie ist gegeben durch die Reduktion an Unreinheit

$$d_A(A_1, A_2) = R(A) - \frac{\hat{\mathrm{P}}(x \in A_1)}{\hat{\mathrm{P}}(x \in A)} R(A_1) - \frac{\hat{\mathrm{P}}(x \in A_2)}{\hat{\mathrm{P}}(x \in A)} R(A_2) \quad .$$

Von der Unreinheit von A werden also die Unreinheiten der Tochterknoten, gewichtet mit den zu erwartenden Anteilen von Beobachtungen je Knoten, abgezogen. Durch Maximierung dieser Distanz kann man sich nun für eine optimale Verzweigung in A entscheiden.

Während sich die Minimierung der Resubstitutionsfehlerrate pro Knoten als Verzweigungskriterium in der Praxis als eher ungeeignet erweist, da derartige Aufspaltungen oft ungünstig auf das weitere Wachstum des Baumes wirken, lautet die auf dem Gini-Index basierende Unreinheit eines Knotens

$$R(A) = 1 - \sum_{r=1}^{k} \hat{\mathrm{P}}(Y = r|x \in A)^2 \quad .$$

Ein darauf basierendes Verzweigungskriterium führt in der Praxis meist zu gut strukturiertem Baumwachstum. Wählt man dagegen die Entropie als zugrundeliegende Unreinheitsfunktion, so erhält man als Unreinheit des Knotens

$$R(A) = -\sum_{r=1}^{k} \hat{\mathrm{P}}(Y = r|x \in A) \cdot \log\Big(\hat{\mathrm{P}}(Y = r|x \in A)\Big) \quad .$$

Dieser Ausdruck entspricht bis auf einen im Knoten konstanten Faktor der Devianz

$$D(A) = -2n(A) \sum_{r=1}^{k} \hat{\mathrm{P}}(Y = r|x \in A) \cdot \log\Big(\hat{\mathrm{P}}(Y = r|x \in A)\Big) \quad .$$

Auf diesem Devianzkriterium zur Verzweigung der Knoten basieren sämtliche CART-Algorithmen der Programmpakete *S-PLUS* und *R*. Eine Devianz von Null bedeutet, daß ein reiner Knoten vorliegt, während die Unreinheit mit wachsender Devianz zunimmt. Die Trennvariable und der zugehörige Trennwert der Aufteilungen (engl. *Splits*) werden nun so gewählt, daß die Unreinheit der beiden Tochterknoten minimal ist.

Auf diese Art und Weise würde der Baum immer weiter wachsen, bis Homogenität in allen Endknoten erzielt ist. Dies kann immer erreicht werden, da im Extremfall so lange verzweigt werden kann, bis in jeden Endknoten nur noch eine Beobachtung fällt. Die letzten Verzweigungen tragen zwar zur optimalen Partitionierung bezüglich der Lerndaten bei, für eine gute Prognose stellen sie aber eher ein Hindernis dar. Deshalb werden Kriterien benötigt, die das Wachstum eines Baumes rechtzeitg stoppen.

Ein gängiger Ansatz hierzu ist es, einen Knoten dann nicht mehr weiter zu verzweigen, wenn auf ihn weniger als eine festgelegte kleine Anzahl (zum Beispiel 10) von Beobachtungen entfallen. Alternativ kann Homogenität dann als erreicht angesehen werden, sobald die Devianz im Knoten kleiner als ein vorgegebener Bruchteil (zum Beispiel 1/100) der Devianz in der Wurzel wird.

Dennoch wird ein Baum, der unkontrolliert bis zu derartigen Abbruchbedingungen des Algorithmus wächst, sehr groß und unübersichtlich. Diesem Problem kann man damit begegnen und gegensteuern, indem der maximale Baum wieder um seine schwächsten Äste beschnitten wird (engl. *pruning*). Dies geschieht mit Hilfe des sogenannten Kosten-Komplexitätsmaßes eines Baumes T:

$$R_\alpha(T) = R(T) + \alpha|T|$$

Der Betrag von T bedeutet in diesem Fall die Anzahl der Endknoten, α wird als Kosten-Komplexitätsparameter bezeichnet: Je größer α gewählt wird, desto stärker wird die Größe des Baumes bestraft. Die Funktion $R(.)$ steht für eine beliebige Unreinheitsfunktion, wobei in der Praxis meistens die Devianz oder die Resubstitutionsfehlerrate herangezogen wird. Da es Sinn macht, die Beschneidung nicht auf das gleiche Kriterium wie das Wachstum zu stützen, wird im Allgemeinen empfohlen, das Wachstum auf der Devianz basieren zu lassen und die Resubstitutionsfehlerrate dann später zur Beschneidung zu verwenden.

Vorgegangen wird nun so, daß Schritt für Schritt die bezüglich $R(.)$ schwächsten Äste beschnitten werden, um $R_\alpha(T)$ zu minimieren. Häufig bedient man sich hierbei der Methode der Kreuzvalidierung, um zuvor den optimalen Wert des willkürlich zu wählenden Kosten-Komplexitätsparameters α bzw. die optimale Anzahl von Endknoten zu schätzen. Die *R*-Algorithmen liefern dann den kleinsten α-optimalen Baum oder den optimalen Baum zu einer vorgegebenen Endknotenzahl bzw. Hierarchietiefe. Unter der Hierarchietiefe versteht man dabei die maximal zugelassene Anzahl von Splits, die beginnend in der Wurzel bis zu den Endknoten durchgeführt werden dürfen.

Aus Gründen der Rechenzeit wurde im Rahmen dieser Arbeit meist darauf verzichtet, tatsächlich eine optimale Anzahl von Endknoten zu ermitteln. Stattdessen kann bereits das Wachstum des Baumes auf eine a priori festgelegte Hierarchietiefe beschränkt werden. Dieser Ansatz führt zwar nicht zu optimalen Bäumen, spart aber enorm an Rechenzeit.

2.3 Bagging

Nachdem also die Entscheidung für ein Klassifikationsverfahren als Basis gefallen ist (nämlich CART, siehe Abschnitt 2.2.2), kann man sich nun dem eigentlichen Schwerpunkt dieses Kapitels, nämlich dem Aggregieren von Klassifikationsverfahren, zuwenden. Die wohl einfachste Variante stellt hier das Bagging dar. Der Begriff leitet sich als Kurzform von *Bootstrap Aggregating* ab.

Wie beim gewöhnlichen Bootstrapping werden aus der Lernstichprobe L mehrere Unterstichproben L_m nach dem Design SRSWR (*Simple Random Sampling With Replacement*) von gleichem Umfang n_L gezogen. Die Auswahlwahrscheinlichkeiten lauten also pro Untersuchungseinheit bei jeder Teilziehung $1/n_L$ und jede Einheit kann so mehrfach in die Unterstichprobe gelangen. Die Umfänge der einzelnen Stichproben könnten auch allgemeiner als n_m bezeichnet werden, so daß unterschiedliche Größen pro Ziehung m zulässig sind. Bei Bagging verwendet man aber üblicherweise immer den gleichen Stichprobenumfang, nämlich n_L, also den Gesamtumfang der Lernstichprobe.

Breiman (1996) entdeckte den Nutzen, der sich daraus ergibt, solche durchmischte (engl.

perturbed) Versionen der Lernstichprobe zu bilden und die zugehörigen Prädiktoren über Mehrheitsentscheidungen miteinander zu verknüpfen. In der Bagging-Prozedur werden also derartige Lernstichproben L_m des Umfangs n_L durch Zufallsziehung aus L gewonnen. Auf jedem solchen Bootstrap-Datensatz kann dann ein Prädiktor $C(., L_m)$ entwickelt werden, der später für die finale Abstimmung herangezogen wird. Der Klassifikator $C(., L_m)$ wird also anhand der m-ten Bootstrap-Stichprobe gebildet. Alternativen zu diesem einfachen nichtparametrischen Bootstrap sind der parametrische Bootstrap (vgl. zum Beispiel Dudoit, Fridlyand & Speed (2002)) und Lernstichproben basierend auf konvexen Pseudodaten (vgl. zum Beispiel Breiman (1998)), die im Rahmen dieser Arbeit aber nicht weiter behandelt werden.

Für jeden dieser Datensätze L_m liegt schließlich ein eigener Basis-Klassifikator, in diesem Fall ein CART-Baum, vor. Eine zu klassifizierende Untersuchungseinheit durchläuft dann alle so gebildeten Bäume und erhält letztendlich als Prognose diejenige Klasse, die am häufigsten auftritt. Diese auf einfachem Mehrheitsentscheid beruhende Aggregationsmethode kann auch als Abstimmung (engl. *voting*) bezeichnet werden, woraus sich die in der Literatur gängige Bezeichnung *Voting-Verfahren* ableitet.

Ziel dieser Vorgehensweise ist es, die Fehlklassifikationsraten, die ein einzelner Baum erzielen würde, zu verbessern. Voraussetzung für eine solche Verbesserung ist aber in jedem Fall ein Basis-Klassifikationsverfahren, das selbst instabil ist. Dies bedeutet, daß bereits kleine Änderungen in der Datengrundlage zu großen Auswirkungen auf den Klassifikator führen können. Nur ein Verfahren, das in diesem Sinne eine starke Variabilität aufweist, kann durch Bootstrapping nachträglich stabilisiert werden. Typische instabile Verfahren sind CART und neuronale Netze, während lineare Diskriminanzfunktionen und Nächste-Nachbarn-Verfahren auf geringe Veränderungen in den Daten so wenig reagieren, daß eine Verbesserung der Prognose durch Verringerung der Variabilität der Resultate kaum möglich ist. Zusammengefaßt kann man also sagen, daß gute, aber instabile Verfahren ihrer optimalen Lösung durch Varianzreduzierung angenähert werden. Der größte Vorteil der Bäume, nämlich die einfache Interpretierbarkeit der binären Struktur, entfällt natürlich auf der anderen Seite, denn am Abstimmungsergebnis kann man die Wirkungsweise der relevanten Kovariablen nicht mehr erkennen. Dies kann jedoch zu Gunsten einer höheren Vorhersagegüte in Kauf genommen werden. Eine ausführlichere Beschreibung der Wirkungsweise von Bagging sowie alternative Erklärungsansätze folgen in einem späteren Abschnitt über Bagging in der Literatur.

Formal betrachtet kann diese Technik der Abstimmung mit einfachem Mehrheitsentscheid wie folgt dargestellt werden:

$$C_{bag}(x, L) = \text{argmax}_j \left(\sum_{m=1}^{M} I(C(x, L_m) = j) \right) \quad . \tag{2.1}$$

Hier steht L_m für die m-te Bootstrap-Version des Lerndatensatzes, $I(.)$ stellt die Indika-

torfunktion dar, die den Wert 1 annimmt, wenn die Bedingung in Klammern erfüllt ist, und ansonsten gleich Null ist. Die Anzahl der verschiedenen Bootstrap-Ziehungen wurde mit M bezeichnet. Da es sich dabei auch gleichzeitig um die Anzahl der zyklischen Durchläufe des Algorithmus handelt, spricht man hier oft von M als der gewählten Anzahl von Bagging-Zyklen.

Auch eine gewichtete Mehrheitsabstimmung für die vorherzusagende Klasse anhand einer Beobachtung x ist denkbar, macht aber beim Bagging, wo die einzelnen Klassifikatoren auf gleichberechtigten und unabhängig gezogenen Bootstrap-Stichproben basieren, wohl wenig Sinn. Sie lautet in Formeln

$$\text{argmax}_j \left(\sum_{m=1}^{M} c_m I(C(x, L_m) = j) \right) \quad , \tag{2.2}$$

wobei c_m das Gewicht darstellt, das der m-ten Lernstichprobe zugeordnet ist.

Bei einer Abstimmung, egal ob ungewichtet oder gewichtet, könnte man sich auch für den Grad der Eindeutigkeit interessieren, also wie deutlich die Entscheidung zugunsten der Vorhersage letztendlich ausgefallen ist. Diese Aussagekraft der Prädiktion kann beispielsweise anhand von sogenannten *prediction votes* (vgl. Dudoit, Fridlyand & Speed (2002)) gemessen werden, die für eine Beobachtung x wie folgt definiert sind:

$$\text{PV}(x) = \frac{\max_j \left(\sum_{m=1}^{M} I(C(x, L_m) = j) \right)}{M} \quad , \tag{2.3}$$

beziehungsweise für die gewichtete Version der Abstimmung

$$\text{PV}(x) = \frac{\max_j \left(\sum_{m=1}^{M} c_m I(C(x, L_m) = j) \right)}{\sum_{m=1}^{M} c_m} \quad . \tag{2.4}$$

Zum Abschluß dieses Abschnitts über Bagging wird das Verfahren noch in Abbildung 2.2 in seiner allgemeinen algorithmischen Struktur dargestellt.

2.4 Boosting

Auch beim Boosting, das ursprünglich aus der Informatik, genauer gesagt dem Bereich des maschinellen Lernens, stammt, handelt es sich um ein Voting-Verfahren, das aber im Gegensatz zum eher statischen Bootstrapping von Bagging adaptive Resampling-Techniken einsetzt und mit gewichteter Abstimmung im Aggregationsschritt arbeitet.

Bagging

1. Aus der Lernstichprobe L werden M Bootstrap-Stichproben L_m vom Umfang n_L mit Zurücklegen gezogen.
2. Ein CART-Klassifizierer $C(., L_m)$ wird für jede solche Bootstrap-Stichprobe L_m entwickelt.
3. Die Prädiktoren werden über einen Mehrheitsentscheid aggregiert, wobei die letztendlich vorhergesagte Klassenzugehörigkeit diejenige ist, die von der Mehrheit dieser Prädiktoren vorhergesagt wird:

$$C_{bag}(x, L) = \text{argmax}_j \left(\sum_{m=1}^{M} I(C(x, L_m) = j) \right) \quad .$$

Abbildung 2.2: Algorithmus für Bagging

Im ersten Schritt wird wie beim Bagging eine gewöhnliche Bootstrap-Stichprobe mit identischen Ziehungswahrscheinlichkeiten für alle Untersuchungseinheiten gezogen, auf die das Basis-Klassifikationsverfahren CART aufsetzen kann. Anschließend prognostiziert man den gesamten Lerndatensatz L mit Hilfe dieses Baumes. Im zweiten Schritt wird wiederum eine Unterstichprobe gezogen. Allerdings haben die Untersuchungseinheiten nicht mehr die gleichen Ziehungswahrscheinlichkeiten. Die Auswahlchancen der im ersten Schritt fehlklassifizierten Einheiten werden nun höher gewichtet. Ansonsten wird genauso verfahren wie im ersten Schritt. Ebenso werden bei jedem weiteren Schritt die Ziehungswahrscheinlichkeiten immer weiter verändert. Die Modelle sollen auf diese Art und Weise so verändert werden, daß schwer zu klassifizierende Einheiten besonders berücksichtigt werden. Diese Schritte werden einige Male wiederholt, je nachdem, welche Anzahl von Zyklen gewählt wurde. Am Ende werden die Prognosen aller so gebildeten Bäume wie beim Bagging per Abstimmung zu einer Gesamtprognose kombiniert.

Für die finale Abstimmung gilt nun aber, daß die Kombination der beteiligten CARTs gewichtet vorgenommen wird. Diese Gewichte werden unter Zuhilfenahme der jeweiligen Vorhersagegüte der Bäume bestimmt: Je besser die aktuelle Prognose, desto höher wird dieser Basis-Klassifikator gewichtet.

Ferner muß noch auf eine alternative Umsetzung des Boosting-Ansatzes kurz eingegangen werden: Ein Vorschlag lautet, anstatt mit Ziehungswahrscheinlichkeiten (gewichtetes Resampling) in jedem Schritt mit allen Datenpunkten zu rechnen, diese aber stattdessen gewichtet in die CART-Klassifikation eingehen zu lassen. Die Verwendung von solchen Gewichten statt Resampling hängt selbstverständlich von der Verfügbarkeit einer Funktion ab, die gewichtete Inputdaten verarbeiten kann. Die gängigen CART-Algorithmen von R ermöglichen dies. Auf diese Art und Weise entfällt der zufällige Aspekt der Ziehung und die scheinbare Verwandtschaft zum oben erwähnten Bagging verschwindet. Jeder einzelne

Klassifikator kann dann als dasjenige Modell angesehen werden, das mit Hilfe der aktuellen Gewichte des m-ten Schrittes gebildet wurde.

Diese interessante Variante wurde hier im Rahmen zahlreicher empirischer Versuche ausprobiert, doch schon bald wieder verworfen, da sich der Umgang mit Situationen, in denen unbrauchbare Gewichtungsfaktoren resultieren (wenn beispielsweise mehr als die Hälfte der Einheiten falsch klassifiziert wurden) und ein Neustart durchgeführt werden muß, als kompliziert herausstellt. Startet man nämlich hier wieder mit gleichverteilten Ziehungswahrscheinlichkeiten neu, so wiederholt sich die in diesem Falle deterministische Reihenfolge der Gewichtungen, das heißt, die neuen Zyklen liefern letztendlich keine neuen Beiträge zur Abstimmung. Auch andere Varianten, bei denen das Gewichtungskriterium durch geringfügige Veränderungen an die numerischen Probleme angepaßt wurde, lieferten etwas schlechtere Ergebnisse als die entsprechenden Bootstrap-Versionen, was eine weitere Begründung für das Arbeiten mit Ziehungswahrscheinlichkeiten liefert. Das Phänomen, daß der Zufallsaspekt die Resultate verbessern kann, wurde laut Dettling (2004) bereits von einer Reihe von Forschern unabhängig voneinander empirisch festgestellt. Auch Bauer & Kohavi (1999) weisen darauf hin, daß in der Literatur je nach Implementation und verwendetem Datensatz hier von keiner einheitlichen Überlegenheit der einen oder anderen Variante gesprochen werden kann.

Nach diesen allgemeinen Betrachtungen über Boosting folgt nun eine Beschreibung der wichtigsten praktischen statistischen Umsetzungen dieses Konzepts aus der Informatik.

2.4.1 Discrete AdaBoost

Die gängigste Variante, der *Discrete AdaBoost*-Algorithmus, wie er ursprünglich von Freund & Schapire (1996) eingeführt wurde, beginnt im ersten Schritt mit den Auswahlwahrscheinlichkeiten $w_1 = \cdots = w_{n_L} = 1/n_L$. Der Faktor, mit dem diese Wahrscheinlichkeiten der jeweils letzten Ziehung in den folgenden Schritten multipliziert werden, lautet

$$\exp\left(\log\left(\frac{1 - E_w[I(Y \neq C(x, L_m))]}{E_w[I(Y \neq C(x, L_m))]}\right) I(Y_i \neq C(x_i, L_m))\right) \quad .$$

Anschließend muß die so veränderte Verteilung renormiert werden. Der Erwartungswert E_w in dieser Formel steht für die (mit den jeweils aktuellen Auswahlchancen) gewichtete Fehlklassifikationsrate

$$\sum_{i=1}^{n_L} w_i I(Y_i \neq C(x_i, L_m)) \quad .$$

Betrachtet man den Term genauer, so erkennt man, daß einfach alle falsch klassifizierten Einheiten um den im Logarithmus stehenden Bruch multiplikativ hochgewichtet werden, während die Auswahlwahrscheinlichkeiten der korrekt klassifizierten Einheiten zunächst unverändert bleiben und sich erst bei der Renormierung der neuen Verteilung anpassen.

Algorithmisch betrachtet funktioniert Discrete AdaBoost also wie in Abbildung 2.3 dargestellt.

Discrete AdaBoost

1. Starte mit Gewichten $w_1 = \cdots = w_{n_L} = 1/n_L$ für die Beobachtungen der Lernstichprobe L.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Die aktuellen Gewichte $w_1, \ldots, w_{n_L}$ werden für den Resampling-Vorgang verwendet. Basierend auf diesen Wahrscheinlichkeiten wird eine Lernstichprobe L_m vom Umfang n_L aus L mit Zurücklegen gezogen.
   (b) Anhand von L_m wird ein Klassifizierer $C(., L_m)$ entwickelt.
   (c) Die Lernstichprobe durchläuft den Klassifizierer $C(., L_m)$, woraus man den Indikator $\epsilon_i = 1$ erhält, falls die i-te Beobachtung falsch klassifiziert wurde. Ansonsten gilt $\epsilon_i = 0$.
   (d) Mit $e_m = \sum_{i=1}^{n_L} w_i \epsilon_i$, $b_m = (1 - e_m)/e_m$ und $c_m = \log((1 - e_m)/e_m)$ werden die Resampling-Gewichte für den nächsten Schritt aktualisiert:

$$w_{i,new} = \frac{w_i b_m^{\epsilon_i}}{\sum_{j=1}^{n_L} w_j b_m^{\epsilon_j}} = \frac{w_i \exp(c_m \epsilon_i)}{\sum_{j=1}^{n_L} w_j \exp(c_m \epsilon_j)}$$

3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über

$$\text{argmax}_j \left(\sum_{m=1}^{M} c_m I(C(x, L_m) = j) \right) \quad .$$

Abbildung 2.3: Algorithmus für Discrete AdaBoost

Während es sich bei e_m um eine gewichtete Summe von Fehlern handelt, bezeichnet die Größe

$$c_m = \log \left(\frac{1 - e_m}{e_m} \right)$$

das logarithmierte Verhältnis (engl. *Odds*) von Treffern zu Fehlern.

Die Wahl genau dieses Terms als Gewichtungsfaktor ist im Übrigen nicht willkürlich: Er sorgt nämlich dafür, daß im nächsten Schritt je 50 Prozent der Auswahlwahrscheinlichkeitsmasse auf die zuletzt fehlklassifizierten sowie auf die zuletzt korrekt klassifizierten Untersuchungseinheiten entfallen. Für die aktualisierten Gewichte gilt also:

$$\sum_{\epsilon_i=1} w_{i,new} = \sum_{\epsilon_i=0} w_{i,new} = 0.5 \quad .$$

Der Term c_m, der als Faktor verwendet wird, kann in zwei Extremfällen problematisch werden: Gilt für die Fehlklassifikationsrate $e_m = 0$, so gilt für den logarithmierten Odds

$c_m \to \infty$, er ist also nicht mehr definiert. Ist andererseits die Prognose dagegen sogar schlechter als einfaches Raten im binären Fall, also die Fehlklassifikationsrate e_m größer als 0.5, so wird der Multiplikator kleiner als 1 und falsch klassifizierte Einheiten würden geringer statt höher gewichtet werden.

In beiden Fällen könnte man so vorgehen, daß man die Auswahlwahrscheinlichkeiten wieder auf die anfängliche Gleichverteilung zurücksetzt und diesen fehlerhaften Zyklus nicht in die Abstimmung mit einbezieht. Alternativ kann aber das Kriterium auch geringfügig geändert werden, um derartige numerische Probleme zu vermeiden. Dies erscheint sicherlich als die elegantere Lösung des Problems. In der endgültigen Version wird deshalb in jedem Schritt geprüft, ob die Konsistenzbedingungen $e_m > 0$ und $b_m > 1$ erfüllt sind. Ist dies nicht der Fall, so setzt man $e_m = 1/n_L$ bzw. $b_m = 1 + 1/n_L$, womit die numerischen Schwierigkeiten vermieden sind.

Bisher wurde das Kriterium ausschließlich für den binären Fall konzipiert: Das Konzept des schwachen Lernverfahrens, auf dessen Basis Boosting als maschinelles Lernverfahren entwickelt wurde, bezeichnet nämlich einen binären Klassifikator, der garantiert eine geringfügig bessere Fehlerrate als 0.5 erzielt. Im Mehrklassenfall kann dies jedoch definitiv nicht mehr garantiert werden. Da die Gewichte

$$c_m = \log\left(\frac{1 - e_m}{e_m}\right)$$

demnach grundsätzlich für Zweiklassenprobleme konstruiert sind, wird für k-Klassenprobleme eine Anpassung vorgeschlagen. Diese Idee, wie Discrete AdaBoost durch kleine Änderungen auf diesen allgemeineren Fall erweitert werden kann, funktioniert, ohne daß das Mehrklassenproblem auf mehrere binäre Probleme (wie zum Beispiel beim *one-against-all*-Ansatz) zurückgeführt werden muß:

Als Fehlerrate kann in k-Klassenproblemen stets $(k-1)/k$ erreicht werden, einfach indem alle Beobachtungen in die größte Klasse prognostiziert werden. Die darauf basierende Anpassung lautet

$$c_m = \log\left(\frac{(1 - e_m)/(1/k)}{e_m/(1 - 1/k)}\right) = \log\left(\frac{(1 - e_m)(k-1)}{e_m}\right) \quad .$$

So erhält man für den Fall $e_m \to (k-1)/k$ einen Wert von $c_m \to 0$. Diese neue und vielversprechende Idee, mit der also aufwendige *one-against-all*-Ansätze vermieden werden können, kann als Korrektur verwendet werden und ist relevant für die Entwicklung einer Mehrklassenversion des gewöhnlichen AdaBoost, die in Abbildung 2.4 algorithmisch dargestellt ist.

2.4.2 Real AdaBoost

Im Fall von lediglich zwei Klassen wird gewöhnlich eine binäre Darstellung, wie $\check{Y} \in \{0, 1\}$ oder $\tilde{Y} \in \{-1, 1\}$, für den Klassenindikator gewählt. Letztere Darstellungsweise findet

Discrete AdaBoost (Mehrklassenvariante)

1. Starte mit Gewichten $w_1 = \cdots = w_{n_L} = 1/n_L$ für die Beobachtungen der Lernstichprobe L.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Die aktuellen Gewichte $w_1, \ldots, w_{n_L}$ werden für den Resampling-Vorgang verwendet. Basierend auf diesen Wahrscheinlichkeiten wird eine Lernstichprobe L_m vom Umfang n_L aus L mit Zurücklegen gezogen.
   (b) Anhand von L_m wird ein Klassifizierer $C(., L_m)$ entwickelt.
   (c) Die Lernstichprobe durchläuft den Klassifizierer $C(., L_m)$, woraus man den Indikator $\epsilon_i = 1$ erhält, falls die i-te Beobachtung falsch klassifiziert wurde. Ansonsten gilt $\epsilon_i = 0$.
   (d) Mit $e_m = \sum_{i=1}^{n_L} w_i \epsilon_i$, $b_m = ((1 - e_m)(k - 1))/e_m$ und $c_m = \log(((1 - e_m)(k - 1))/e_m)$ werden die Resampling-Gewichte für den nächsten Schritt aktualisiert:

$$w_{i,new} = \frac{w_i b_m^{\epsilon_i}}{\sum_{j=1}^{n_L} w_j b_m^{\epsilon_j}} = \frac{w_i \exp(c_m \epsilon_i)}{\sum_{j=1}^{n_L} w_j \exp(c_m \epsilon_j)}$$

3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über

$$\text{argmax}_j \left(\sum_{m=1}^{M} c_m I(C(x, L_m) = j) \right) \quad .$$

Abbildung 2.4: Algorithmus für Discrete AdaBoost (Mehrklassenvariante)

häufig Anwendung im Forschungsfeld des maschinellen Lernens. Sie wird hier anstelle der bisher verwendeten Darstellung $Y \in \{1, 2\}$ auch beim im folgenden beschriebenen *Real AdaBoost*-Algorithmus für binäre Probleme herangezogen. Man gewinnt die Darstellung $\tilde{Y} \in \{-1, 1\}$ aus $Y \in \{1, 2\}$ leicht durch die Transformation $\tilde{Y} = 2Y - 3$.

Real AdaBoost, das von Friedman, Hastie & Tibshirani (2000) eingeführt wurde, verwendet anstelle der strikten 0-1-Klassifikation eines Prädiktors $C(x, L)$ reelwertige Klassifikationsfunktionen $f(x, L)$ mit der Übereinkunft, daß $f(x, L) \geq 0$ einer Klassifikation $C(x, L) = 1$ und umgekehrt $f(x, L) < 0$ einer Prognose $C(x, L) = -1$ entspricht. Für gewöhnlich basieren diese Terme $f(x, L)$ auf den Klassifikationswahrscheinlichkeiten von $C(x, L)$. In Abbildung 2.5 ist der schematische Ablauf dieses Algorithmus dargestellt.

Der entscheidende Term im Aktualisierungsschritt der Gewichte ist $w_i \exp(-\tilde{Y}_i f(x_i, L_m))$, der auch in Abhängigkeit von einem Indikator ϵ für Treffer ($\epsilon_i = 0$) und Fehlklassifikationen ($\epsilon_i = 1$) dargestellt werden kann:

$$w_i \exp(-\tilde{Y}_i f(x_i, L_m)) = \begin{cases} w_i \exp(-|f(x_i, L_m)|) & \text{für } \epsilon_i = 0 \\ w_i \exp(|f(x_i, L_m)|) & \text{für } \epsilon_i = 1 \end{cases} \quad .$$

Real AdaBoost

1. Starte mit Gewichten $w_1 = \cdots = w_{n_L} = 1/n_L$ für die Beobachtungen der Lernstichprobe L.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Die aktuellen Gewichte $w_1, \ldots, w_{n_L}$ werden für den Resampling-Vorgang verwendet. Basierend auf diesen Wahrscheinlichkeiten wird eine Lernstichprobe L_m vom Umfang n_L aus L mit Zurücklegen gezogen.
   (b) Anhand von L_m wird der Klassifizierer $C(., L_m)$ entwickelt.
   (c) Die Lernstichprobe durchläuft den Klassifizierer $C(., L_m)$, woraus man die Zuordnungswahrscheinlichkeiten $p(x_i) = \hat{P}(\tilde{Y}_i = 1|x_i)$ erhält.
   (d) Basierend auf diesen Wahrscheinlichkeiten wird ein reelwertiger Klassifizierer entwickelt
   $$f(x_i, L_m) = 0.5 \cdot \log \frac{p(x_i)}{1 - p(x_i)}$$
   und die Gewichte werden für den nächsten Schritt aktualisiert:
   $$w_{i,new} = \frac{w_i \exp(-\tilde{Y}_i f(x_i, L_m))}{\sum_{j=1}^{n_L} w_j \exp(-\tilde{Y}_j f(x_j, L_m))}$$
3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über
   $$\text{sign}\left(\sum_{m=1}^{M} f(x, L_m)\right) \quad .$$

Abbildung 2.5: Algorithmus für Real AdaBoost

Es ist ersichtlich, daß auch hier bei fehlklassifizierten Beobachtungen die Gewichte vergrößert werden, während sie bei korrekt klassifizierten Einheiten sinken.

Da der Original-Algorithmus in dieser Form nur für Zweiklassenprobleme anwendbar ist, folgt nun eine Variante, die für Mehrklassenprobleme zum Einsatz kommen kann. Sie ist in Abbildung 2.6 schematisch dargestellt.

In beiden Varianten des Algorithmus ist die Existenz der Terme $f(x_i, L_m)$ bzw. $f_j(x_i, L_m)$ nicht gesichert, wenn eine der dazu verwendeten Klassenwahrscheinlichkeiten gleich Null ist. Auch hier wird aber eine kleine Modifikation vorgeschlagen, um die Existenz dieser Terme zu garantieren: In jedem Schritt des Algorithmus wird geprüft, ob alle $p_j(x_i)$ von Null verschieden sind. Ist dies nicht der Fall, so werden die betroffenen Wahrscheinlichkeiten gleich $1/n_L$ gesetzt.

Real AdaBoost (Mehrklassenvariante)

1. Starte mit Gewichten $w_1 = \cdots = w_{n_L} = 1/n_L$ für die Beobachtungen der Lernstichprobe L.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Die aktuellen Gewichte $w_1, \ldots, w_{n_L}$ werden für den Resampling-Vorgang verwendet. Basierend auf diesen Wahrscheinlichkeiten wird eine Lernstichprobe L_m vom Umfang n_L aus L mit Zurücklegen gezogen.
   (b) Anhand von L_m wird der Klassifizierer $C(., L_m)$ entwickelt.
   (c) Die Lernstichprobe durchläuft den Klassifizierer $C(., L_m)$, woraus man die Zuordnungswahrscheinlichkeiten $p_j(x_i) = \hat{P}(Y_i = j|x_i)$ erhält.
   (d) Basierend auf diesen Wahrscheinlichkeiten wird ein reelwertiger Klassifizierer entwickelt
   $$f_j(x_i, L_m) = 0.5 \cdot \log \frac{p_j(x_i)}{(\prod_{l \neq j}^{k} p_l(x_i))^{\frac{1}{k-1}}}$$
   und die Gewichte werden für den nächsten Schritt aktualisiert:
   $$w_{i,new} = \frac{w_i \exp(-f_{Y_i}(x_i, L_m))}{\sum_{j=1}^{n_L} w_j \exp(-f_{Y_j}(x_j, L_m))}$$
3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über
   $$\text{argmax}_j \left(\sum_{m=1}^{M} f_j(x, L_m) \right) \quad .$$

Abbildung 2.6: Algorithmus für Real AdaBoost (Mehrklassenvariante)

2.4.3 Gentle AdaBoost

In Friedman, Hastie & Tibshirani (2000) wird *Gentle AdaBoost* als eine Variante von Real AdaBoost vorgeschlagen, die eine andere Aktualisierungsfunktion für die Gewichtung verwendet und damit empirisch in vielen Fällen zu besseren Ergebnissen führt. Der Grund dafür ist, daß die Klassifikationswahrscheinlichkeiten in einem anderen, numerisch stabileren Term als bei Real AdaBoost verwendet werden (siehe Abbildung 2.7).

Dieses Verfahren, bei dem anstelle des numerisch problematischen Quotienten zwischen zwei Wahrscheinlichkeiten auf Differenzen zurückgegriffen wird, erweist sich als numerisch deutlich stabiler. Dies läßt sich schon daran erahnen, daß eine Korrektur, um Definitionslücken der Kriterien abzufangen, im Gegensatz zu den beiden anderen AdaBoost-Algorithmen nicht nötig ist.

In Abbildung 2.8 ist noch eine Mehrklassenvariante von Gentle AdaBoost dargestellt, die

Gentle AdaBoost

1. Starte mit Gewichten $w_1 = \cdots = w_{n_L} = 1/n_L$ für die Beobachtungen der Lernstichprobe L.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Die aktuellen Gewichte $w_1, \ldots, w_{n_L}$ werden für den Resampling-Vorgang verwendet. Basierend auf diesen Wahrscheinlichkeiten wird eine Lernstichprobe L_m vom Umfang n_L aus L mit Zurücklegen gezogen.
   (b) Anhand von L_m wird der Klassifizierer $C(., L_m)$ entwickelt.
   (c) Die Lernstichprobe durchläuft den Klassifizierer $C(., L_m)$, woraus man die Zuordnungswahrscheinlichkeiten $p(x_i) = \hat{P}(\tilde{Y}_i = 1|x_i)$ erhält.
   (d) Basierend auf diesen Wahrscheinlichkeiten wird ein reellwertiger Klassifizierer entwickelt
   $$f(x_i, L_m) = p(x_i) - (1 - p(x_i))$$
   und die Gewichte werden für den nächsten Schritt aktualisiert:
   $$w_{i,new} = \frac{w_i \exp(-\tilde{Y}_i f(x_i, L_m))}{\sum_{j=1}^{n_L} w_j \exp(-\tilde{Y}_j f(x_j, L_m))}$$
3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über
   $$\text{sign}\left(\sum_{m=1}^{M} f(x, L_m)\right) \quad .$$

Abbildung 2.7: Algorithmus für Gentle AdaBoost

ebenso wie bei Real AdaBoost auf einer Durchschnittsbildung der Wahrscheinlichkeiten für alle nicht prognostizierten Klassen beruht.

2.4.4 LogitBoost

LogitBoost ist eine Technik, die ein unmittelbar interessierendes Kriterium, nämlich die binäre Log-Likelihood eines additiven logistischen Modells, optimiert (siehe dazu auch Friedman, Hastie & Tibshirani (2000) im Abschnitt zur Literatur in diesem Kapitel). Der Algorithmus basiert dabei auf gewichteten Daten, verwendet also im Gegensatz zu den bisher vorgestellten Verfahren kein Bootstrapping, um die benötigte Neugewichtung der Daten vorzunehmen. Ebenfalls ist es von Bedeutung, die binäre Zielgröße nicht als $\tilde{Y}$ mit Werten -1 und 1 wie bei Real und Gentle AdaBoost, sondern als $\check{Y}$ mit Werten 0 und 1 zu kodieren. Außerdem weist der Algorithmus eine weitere Besonderheit auf: Im Gegensatz zu allen bisher betrachteten Methoden basiert er nämlich nicht auf Klassifikations-

Gentle AdaBoost (Mehrklassenvariante)

1. Starte mit Gewichten $w_1 = \cdots = w_{n_L} = 1/n_L$ für die Beobachtungen der Lernstichprobe L.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Die aktuellen Gewichte $w_1, \ldots, w_{n_L}$ werden für den Resampling-Vorgang verwendet. Basierend auf diesen Wahrscheinlichkeiten wird eine Lernstichprobe L_m vom Umfang n_L aus L mit Zurücklegen gezogen.
   (b) Anhand von L_m wird der Klassifizierer $C(., L_m)$ entwickelt.
   (c) Die Lernstichprobe durchläuft den Klassifizierer $C(., L_m)$, woraus man die Zuordnungswahrscheinlichkeiten $p_j(x_i) = \hat{P}(Y_i = j|x_i)$ erhält.
   (d) Basierend auf diesen Wahrscheinlichkeiten wird ein reellwertiger Klassifizierer entwickelt
   $$f_j(x_i, L_m) = p_j(x_i) - \left(\frac{1}{k-1} \sum_{l \neq j}^{k} p_l(x_i) \right)$$
   und die Gewichte werden für den nächsten Schritt aktualisiert:
   $$w_{i,new} = \frac{w_i \exp(-f_{Y_i}(x_i, L_m))}{\sum_{j=1}^{n_L} w_j \exp(-f_{Y_j}(x_j, L_m))}$$
3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über
   $$\text{argmax}_j \left(\sum_{m=1}^{M} f_j(x, L_m) \right) \quad .$$

Abbildung 2.8: Algorithmus für Gentle AdaBoost (Mehrklassenvariante)

sondern auf Regressionsbäumen. Da das bei CART optimierte Devianzkriterium für metrische Zielgrößen genau der KQ-Methode entspricht (vgl. z.B. Tutz (2000)), muß an den entsprechenden Algorithmen keine Änderung vorgenommen werden.

Im Gegensatz zu allen bisher geschilderten Verfahren wurde hier der Ansatz, mit Gewichtung statt Zufallsziehungen zu arbeiten, ebenso wie beim folgenden L_2-Boost, beibehalten, da er sich im speziellen Fall von Regressionsbäumen empirisch als die stabilere Variante herausgestellt hat. Die Vorgehensweise läßt sich wie in Abbildung 2.9 dargestellt beschreiben.

Auf diese Art und Weise werden in jedem Schritt *working response* und Gewichte so gewählt, daß Fälle, die nahe an der Entscheidungsgrenze (engl. *decision boundary*) liegen, deren Zuordnung in die zu prognostizierende Klasse also noch sehr zweifelhaft ist, ein besonders hohes Gewicht bekommen.

LogitBoost

1. Starte mit Gewichten $w_1 = \cdots = w_{n_L} = 1/n_L$ für die Beobachtungen der Lernstichprobe L, $F(x_i) = 0$ und Wahrscheinlichkeitsschätzungen $p(x_i) = \hat{P}(\check{Y}_i = 1|x_i) = \frac{1}{2}$.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Berechne einen *working response* z_i und Gewichte w_i gemäß

$$\begin{aligned} z_i &= \frac{\check{Y}_i - p(x_i)}{p(x_i)(1-p(x_i))} \\ w_i &= p(x_i)(1-p(x_i)) \end{aligned}$$

   (b) Anhand von L_m wird eine gewichtete Kleinst-Quadrat-Regression $f(x, L_m)$ von x auf z mit den Gewichten w bestimmt.
   (c) Anschließend wird $F(x_i)$ durch $F(x_i) + \frac{1}{2} f(x_i, L_m)$ ersetzt und die Wahrscheinlichkeitsschätzungen für den nächsten Schritt werden aktualisiert:

$$p_{new}(x_i) = \frac{\exp(F(x_i))}{\exp(F(x_i)) + \exp(-F(x_i))}$$

3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über

$$\text{sign}\left(\sum_{m=1}^{M} f(x, L_m)\right) \quad .$$

Abbildung 2.9: Algorithmus für LogitBoost

Friedman, Hastie & Tibshirani (2000) schlagen ursprünglich einige Beschränkungen für den Wertebereich der Wahrscheinlichkeiten sowie der Gewichte vor, um den Algorithmus numerisch stabil zu halten. Hier wurde auf die Implementierung dieser willkürlich gewählten Schranken als Untergrenzen verzichtet, da Definitionslücken aufgrund der Neugewichtungsschritte nicht auftreten können und nur solche gravierenden Probleme bei den anderen Algorithmen bedacht wurden. So erscheint die unmittelbare und objektive Vergleichbarkeit der verschiedenen Verfahren besser gewährleistet, was ja das Hauptziel dieser Arbeit darstellt.

Die Autoren stellen bei zahlreichen Experimenten mit realen und simulierten Daten fest, daß dieser Algorithmus in Verbindung mit Klassifikationsbäumen am besten mit *Stumps*, also mit Bäumen mit nur zwei Endknoten (Hierarchietiefe 1), arbeitet. Daher wird auch im Rahmen dieser Arbeit in den später folgenden empirischen Vergleichsstudien speziell darauf geachtet, für die Berechnung der LogitBoost-Prognosen tendenziell kleinere Bäume heranzuziehen.

Es exisitert auch eine Variante, die unmittelbar mit Mehrklassenproblemen umgehen kann.

Da der nominale Mehrklassenfall in dieser Arbeit jedoch keine große Rolle spielt und in der Literatur bereits ausführlich behandelt ist (Dettling & Bühlmann (2003) stellen fest, daß die Mehrklassenvariante empirisch nicht besonders gut arbeitet), wurde auf eine algorithmische Umsetzung dieses Ansatzes verzichtet.

2.4.5 L_2-Boost

Schließlich soll noch eine Boosting-Variante vorgestellt werden, die ursprünglich entwickelt wurde, um entweder binäre Variablen zu klassifizieren oder (als Regressionsvariante) reellwertige Zielgrößen vorherzusagen: *L_2-Boost* (Bühlmann & Yu (2002b)), ein Spezialfall des allgemeineren *Gradient-Descent*-Algorithmus, wie er auch von Friedman (2001) vorgestellt wurde, funktioniert ohne jede Art von Gewichtung. Grundidee bei diesen Techniken ist das schrittweise Anpassen der Residuen, anstatt in jedem Schritt dieselbe Zielgröße zu verwenden. Damit weist der Algorithmus (siehe Abbildung 2.10) ebenfalls die Besonderheit auf, nicht auf Klassifikations- sondern auf Regressionsbäumen zu basieren. Erst im letzten Schritt wird die reellwertige Prognose wieder auf die prognostizierte Klasse abgebildet. Die binäre Zielgröße wird hier als $\tilde{Y} \in \{-1, 1\}$ kodiert.

Die Autoren schlagen eine Beschränkung des Wertebereichs von $\hat{F}$ auf das Intervall $[-1; 1]$ vor. Diese Restriktion, die einen zu gravierenden Einfluß einzelner Prädiktoren verhindert, kommt auch hier zur Anwendung.

Neben dem quadratischen Verlust, der bei diesem Algorithmus minimiert wird (siehe dazu Friedman (2001) im Literaturteil dieses Kapitels), sind auch diverse andere Verlustfunktionen möglich.

Auch dieses Verfahren scheint empirisch, ebenso wie LogitBoost, mit Stumps zu deutlich besseren Resultaten zu führen, so daß in Verbindung mit L_2 stets tendenziell kleinere Bäume mit nur wenigen Endknoten als Prädiktoren zum Einsatz kommen.

L_2-Boost

1. Im ersten Schritt wird ein reellwertiger Initial-Klassifikator $\hat{F}_0(x) = \hat{f}(x)$ über die Minimierung des Kleinst-Quadrat-Fehlers

$$\min \left(\sum_{i=1}^{n_L} (\tilde{Y}_i - \hat{f}(x_i))^2 \right)$$

   berechnet. Dann startet die Iteration mit $m = 1$.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Der negative Gradienten-Vektor

$$u_i = \tilde{Y}_i - \hat{F}_{m-1}(x_i)$$

   wird berechnet.
   (b) Der reellwertige Klassifikator $\hat{f}_m(x)$ wird nun an die aktuellen Residuen angepaßt, wiederum über die Minimierung des Kleinst-Quadrat-Fehlers

$$\min \left(\sum_{i=1}^{n_L} (u_i - \hat{f}_m(x_i))^2 \right) \quad .$$

   (c) Die Vorhersage $\hat{F}_m(x)$ wird aktualisiert durch

$$\hat{F}_m(x) = \hat{F}_{m-1}(x) + \hat{f}_m(x) \quad .$$

3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über

$$\text{sign} \left(\hat{F}_M(x) \right) \quad .$$

Abbildung 2.10: Algorithmus für L_2-Boost

2.5 Bagging und Boosting in der Literatur

Bagging wurde wie bereits erwähnt zunächst von Breiman (1996) eingeführt. Er sieht die Stabilität eines Klassifikationsverfahrens dabei als den entscheidenden Faktor an. Eine Verbesserung ist nur dann möglich, wenn bereits kleine Änderungen in der Stichprobe zu großen Veränderungen in der Prognose führen können. Er führt den Effekt der Verbesserung durch Bagging auf die Anwendung des Bootstrap zurück, es handelt sich also demnach einfach um eine Technik der Varianzreduktion bei instabilen Verfahren, kein statistisches Modell kommt zum tragen. Als in diesem Sinne instabil erachtet er Verfahren wie CART, neuronale Netze oder die Variablenauswahl im linearen Modell, stabil und damit für Bagging ungeeignet erscheinen dagegen die lineare Diskriminanzanalyse oder Nächste-Nachbarn-Verfahren.

Formal betrachtet lautet die Wahrscheinlichkeit für eine richtige Klassifikation, gegeben x

$$\sum_{r=1}^{k} \mathrm{P}(C(x,L)=r|x)\mathrm{P}(Y=r|x) \quad ,$$

wobei immer

$$\sum_{r=1}^{k} \mathrm{P}(C(x,L)=r|x)\mathrm{P}(Y=r|x) \leq \mathrm{max}_r \mathrm{P}(Y=r|x)$$

gelten muß. Gleichheit ergibt sich exakt für die Bayes-Zuordnung. Für die gesamte Wahrscheinlichkeit einer richtigen Klassifikation gilt

$$\int \left(\sum_{r=1}^{k} \mathrm{P}(C(x,L)=r|x)\mathrm{P}(Y=r|x) \right) P_X dx \quad ,$$

wobei P_X die Wahrscheinlichkeitsverteilung von X darstellt. Weiter heißt ein Klassifizierer $C(.,L)$ ordnungserhaltend (engl. *order-correct*) in x, wenn

$$\mathrm{argmax}_r \mathrm{P}(C(x,L)=r|x) = \mathrm{argmax}_r \mathrm{P}(Y=r|x)$$

gilt. Zu dieser Eigenschaft muß angemerkt werden, daß sie nicht unbedingt auf einen besonders guten Klassifizierer hinweisen muß. Sie bedeutet lediglich, daß die Klasse r bei der Beobachtung x häufiger als jede andere Klasse vorhergesagt wird, wenn sie auch die häufigste Klasse bei Beobachtung x darstellt. Für den durch Bagging aggregierten Klassifikator

$$C_{agg}(x,L) = \mathrm{argmax}_r \mathrm{P}(C(x,L_m)=r|x) \quad \text{mit } m=1,\ldots,M$$

lautet dann die Wahrscheinlichkeit für eine korrekte Klassifikation gegeben x

$$\sum_{r=1}^{k} I(\mathrm{argmax}_i \mathrm{P}(C(x,L_m)=i|x)=r)\mathrm{P}(Y=r|x) \quad .$$

Sie stimmt genau dann mit $\max_r \mathrm{P}(Y = r|x)$ überein, falls $C(., L)$ ordnungserhaltend ist. Die daraus resultierende gesamte Wahrscheinlichkeit für eine richtige Klassifikation kann in zwei Summanden zerlegt werden, basierend auf der Menge A aller ordnungserhaltenden sowie der Menge $\bar{A}$ aller nicht ordnungserhaltenden x:

$$\int_{x\in A} (\max_r \mathrm{P}(Y = r|x))\, P_X dx + \int_{x\in \bar{A}} \left(\sum_{r=1}^{k} I(C_{agg}(x, L) = r)\mathrm{P}(Y = r|x) \right) P_X dx$$

Da der erste Summand dem Fehler der optimalen Bayes-Zuordnung entspricht, wird der durch Bagging aggregierte Klassifikator $C_{agg}(., L)$ optimal, falls der Basis-Klassifizierer $C(., L)$ für alle x ordnungserhaltend ist.

In seinen Studien anhand von insgesamt elf empirischen Datensätzen und mit CART als Basis-Klassifikationstechnik erhält Breiman eine Reduzierung der Fehlklassifikationsrate zwischen 6% und 77% . Neben der gewöhnlichen Abstimmungsmethode experimentiert er auch mit Entscheidungen anhand der Durchschnittsbildung von Klassifikationswahrscheinlichkeiten. Dieser Ansatz liefert aber annähernd identische Ergebnisse. Insgesamt hält er das Arbeiten mit mehr als 50 Zyklen oder mehr als n_L Einheiten pro Bootstrap-Ziehung für überflüssig, da jenseits dieser Schwellen so gut wie keine Verbesserung mehr auftritt. Zusammenfassend hält er fest, daß auf Kosten der einfachen Interpretierbarkeit von Klassifikationsbäumen ein teilweise erheblicher Gewinn an Genauigkeit erzielt werden kann.

Friedman & Hall (2000) beschäftigen sich mit weiteren Überlegungen zur Wirkungsweise von Bagging. Dazu zerlegen sie den Klassifikator in einen linearen Teil, sowie einen Teil höherer Ordnung bzw. äquivalenterweise die zu optimierende Zielfunktion in einen quadratischen Teil, sowie einen Teil höherer Ordnung. Sie kommen dabei zu dem Ergebnis, daß vor allem nichtlineare Komponenten von Schätzern durch Varianzreduktion verbessert werden können, weshalb Bagging auch vorrangig bei hochgradig nichtlinearen Verfahren wie Klassifikationsbäumen oder neuronalen Netzen Wirkung zeigt. Die Varianzreduzierung basiert darauf, daß die nichtlineare Komponente durch eine Schätzung ihres Erwartungswerts ersetzt wird, während der lineare Teil unverändert bleibt. Dies entspricht einer Glättung der zu optimierenden Zielfunktion und führt dazu, daß lokale Optima durch Mittelwerte ersetzt werden, was das Auffinden des globalen Optimums erleichtert. Die Überlegung, daß lineare Funktionen typischerweise weniger variabel sind als hochgradige Polynome, liefert dann letztendlich auch eine Erklärung dafür, daß sich bei sehr stabilen Verfahren kaum Verbesserungen durch Bagging ergeben. Als besonders gut erweist sich Bagging deshalb auch bei recht komplexen Aufgabenstellungen mit vielen *nuisance*-Parametern, da erst hier solche nichtlinearen Komponenten eine bedeutende Rolle spielen. In einer abschließenden Simulationsstudie wurde noch untersucht, welche optimalen Ziehungsumfänge m aus n_L für die Modelle mit und ohne Zurücklegen unter verschiedenen Bedingungen gefunden werden können. Eine Wahl von $m/n_L = 1$ für das Modell mit Zurücklegen sowie $m/n_L = 1/2$ für das Modell ohne Zurücklegen scheinen dabei eine gute Wahl zu sein. Allgemein ähneln sich die Resultate bei diesen beiden Ansätzen sehr stark, so daß sie als in etwa gleichwertig angesehen werden können.

Auch Bühlmann & Yu (2002a) suchen nach Erklärungen für den positiven Effekt von Bagging. Ihre Definition eines stabilen Verfahrens impliziert lediglich, daß das Verfahren gegen irgendeinen stabilen Grenzwert strebt, der nicht identisch mit dem wahren Parameter sein muß. Würde es sich um den wahren Parameter handeln, so entspräche diese Forderung genau der Konsistenz. Diese Definition ist präziser als die von Breiman (1996) gegebene, steht aber nicht im Widerspruch dazu. Die Autoren leiten theoretische Grenzwerte für diese Verbesserung ab und stellen fest, daß Bagging als Glättungsverfahren angesehen werden kann, das strikte Entscheidungen (engl. *hard decisions*) in weiche Entscheidungen (engl. *soft decisions*) umformt. Dies ist insofern entscheidend, da Instabilität in erster Linie bei solchen auf Indikatorfunktionen basierenden *hard decisions* auftritt, zum Beispiel eben bei Klassifikationsbäumen. Ebenfalls beschäftigen sich die Autoren mit der Variante von Ziehen ohne Zurücklegen, die sie als *Subagging* (abgeleitet aus *subsample aggregating*) bezeichnen. Bei Versuchen mit simulierten Daten erhalten sie bei geringen Ziehungsumfängen so etwas bessere Resultate als mit gewöhnlichem Bagging. Zusätzlich ist der Rechenaufwand deutlich kleiner und das Verfahren hat den Vorteil, daß es theoretisch etwas leichter zugänglich ist, vor allem im Regressionsfall. Bei einem empirischen Datensatz liefert allerdings Bagging die besseren Ergebnisse, wobei Subagging mit $m/n_L = 1/2$ (*half subagging*) diesem Resultat noch am nächsten kommt. Abschließend stellen die Autoren fest, daß Bagging sowohl mit großen Klassifikationsbäumen als auch mit Stumps als Basis-Verfahren zu guten Resultaten führen kann, die günstigste Anzahl an Endknoten scheint also von den konkreten Daten abzuhängen. Insgesamt stehen die empirischen Ergebnisse im Widerspruch zu Friedman & Hall (2000), da die vor allem bei großen Stichprobenumfängen dramatischen Verbesserungen deutlicher sind, als es durch die Beeinflussung des nichtlinearen Anteils allein erklärt werden kann.

In einer Folgeveröffentlichung beschäftigt sich Bühlmann (2003) mit weiteren Bagging-Varianten. Für den Fall der Regression führt er *Bragging* (abgeleitet aus *bootstrap robust aggregating*) ein, bei dem anstatt des Mittelwertes der einzelnen Vorhersagen der robustere Median herangezogen wird. Da dieser Ansatz allerdings zur hier interessierenden Klassifikation nicht genutzt werden kann, sei lediglich festgehalten, daß Bragging empirisch zu deutlich besseren Resultaten führt, wenn als Basis-Regressionsverfahren bereits ein sehr glattes Modell vorliegt, das durch den Glättungseffekt von Bagging oder Subagging kaum mehr beeinflußt wird.

Breiman (2001) führt den Oberbegriff *Random Forests* ein, der alle baumbasierenden und aggregierten Verfahren, die auf unabhängigen Zufallsvektoren (also unabhängigem Sampling aus einer identischen Verteilung für alle Bäume) basieren, zusammenfaßt. Die Aggregation der Bäume muß dann durch eine gleichberechtigte, also ungewichtete, Abstimmung erfolgen. Beispiele hierfür sind Bagging oder Ansätze, die pro Knoten eines Baumes zufällig eine der q besten Aufteilungen auswählen (*Random Split Selection*). Die theoretische Herleitung einer oberen Schranke für die Fehlerrate von Random Forests ist zwar nicht von praktischer Relevanz, da sie sehr weite Grenzen liefert, die Tendenz der entscheidenden Größen, die diese Schranke beeinflussen, wird aber klar: Je stärker die Vorhersagegüte der Einzelbäume und je geringer ihre Korrelation (dies kann durch den unabhängigen Zufalls-

aspekt bei der Entstehung der Bäume erreicht werden) untereinander, desto geringer wird die obere Fehlerschranke. Als Maß für das Vertrauen in eine Vorhersage verwendet Breiman die sogenannte *Margin*, was soviel wie Spielraum oder Spanne bedeutet. Sie wird im Rahmen des Abstimmungsprozesses definiert als die Anzahl der korrekten Stimmen minus der Anzahl der Stimmen für eine falsche Klasse:

$$mg(x, Y = r) = \left(\frac{1}{M} \sum_{m=1}^{M} I(C_m(x) = r) \right) - \max_{s \neq r} \left(\frac{1}{M} \sum_{m=1}^{M} I(C_m(x) = s) \right)$$

Je größer dieser Wert, desto zuverlässiger ist eine Prognose. Diese Größe kann, ebenso wie Fehlerraten, Korrelationen oder Wichtigkeit der Kovariaten bei Bagging leicht intern anhand der *Out-of-bag*-Beobachtungen geschätzt werden. Dabei handelt es sich um die Beobachtungen, die aufgrund der zufälligen Ziehung nicht in die für die Modellbildung relevante Substichprobe gefallen sind. Bezieht man nämlich die Prognose nur für diejenigen Beobachtungen in die Abstimmung mit ein, die sich nicht in der jeweiligen Lernstichprobe L_m befinden, so erhält man eine Schätzung für den Generalisierungsfehler, ohne daß eine Aufteilung der Beobachtungen in Lern- und Testdaten nötig ist.

Ein besonders ausführlich untersuchter Spezialfall, nämlich die Kombination aus Bagging (ohne Pruning der resultierenden Bäume) und einer Variante von Random Split Selection, bei der der Zufallsaspekt in der Wahl von Inputvariablen oder Linearkombinationen von Inputvariablen (mit zufälligen Gewichten) pro Knoten steckt, führt zu Resultaten, die ebenso gut sind wie bei Boosting und zeigt empirisch ebenso die Tendenz zu Resistenz gegen Überanpassung (engl. *Overfitting*). Darunter versteht man, daß ein Verfahren (auf Kosten der Generalisierungsfähigkeit auf neue Daten) die Lerndaten zu präzise anpaßt. Die erwähnte Methode ist allerdings deutlich unkomplizierter und leichter zu implementieren, benötigt weniger Rechenzeit und reagiert robuster auf fehlerbehaftete Daten und Ausreißer. Speziell bei Fehlern in der Klassenvariable wird die Prognose deutlich weniger gestört, da im Gegensatz zu Boosting keine Konzentration der Auswahlwahrscheinlichkeit auf einzelne, häufig fehlklassifizierte Beobachtungen stattfinden kann. Als überraschendes Ergebnis stellt Breiman sogar fest, daß die zufällige Auswahl einer einzigen Variable pro Split bereits zu sehr guten Resultaten führt, es bestehen kaum Unterschiede zu größeren Anzahlen von verwendeten Kovariaten.

Als letzte Weiterentwicklung sollte noch die Arbeit von Hothorn & Lausen (2003) genannt werden, die die *Out-of-bag*-Stichprobe beim Bootstrap als Datengrundlage für eine lineare Diskriminanzfunktion verwenden. Diese kanonischen Diskriminanzvariablen werden im anschließend zu entwickelnden Klassifikationsbaum als zusätzliche Prädiktoren genutzt. Der Ansatz, den sie als *Double-Bagging* bezeichnen, kann als Aggregationsverfahren zwischen Kombinationen von Methoden verstanden werden. Da diese *Out-of-bag*-Stichprobe nur einen kleinen Datensatz darstellt, ist eine lineare Diskriminanzanalyse hier relativ unstabil und liefert aufgrund der Unabhängigkeit der Daten keine durch Overfitting verfälschten kanonischen Variablen als Basis für die Bagging-Prozedur. Vergleiche mit linearer Diskriminanzanalyse, Klassifikationsbäumen und Bagging anhand von sechs simulierten und vier

empirischen Datensätzen zeigen, daß Double-Bagging meist am besten abschneidet. Es scheint die Vorzüge der beiden extrem verschiedenen Verfahren von linearer Diskriminanzanalyse und Bagging in sich zu vereinen und ähnelt je nach Datenlage mehr oder weniger einer dieser beiden Techniken.

Unabhängig von der Entwicklung von Bagging wurde von Freund (1995) Boosting eingeführt, zunächst allerdings nicht in Verbindung mit statistischen Aufgabenstellungen, sondern vielmehr im Forschungsfeld des maschinellen Lernens. Im sogenannten PAC-Lernmodell (engl. *probably approximately correct*) erwartet man von einem Lernverfahren, daß es mindestens eine Präzision von $1 - \epsilon$ mit mindestens der Wahrscheinlichkeit $1 - \delta$ erreicht, die als die Reliabilität des Verfahrens bezeichnet wird. Diese Reliabilität kann durch mehrmalige Ziehung von zufälligen Stichproben aus der Grundgesamtheit verbessert werden, indem pro Ziehung getestet wird, ob die geforderte Präzision eingehalten wurde. Ist dies nicht der Fall, so wird dieser Schritt verworfen und eine neue Ziehung durchgeführt. Boosting sieht der Autor dagegen als das adäquate Mittel zur Verbesserung der Präzision an. Ein schwaches Lernverfahren mit einer Trefferwahrscheinlichkeit von $1/2 + \epsilon$, das also nur geringfügig besser als Raten ist, kann dabei zu einem beliebig präzisen Verfahren aufgebaut werden. Dies gilt für den verteilungsfreien Fall, sprich für eine feste vorgegebene Menge von Untersuchungseinheiten. Statistisch formuliert wird der Resubstitutionsfehler auf Null gesenkt. Boosting wird über ein sogenanntes Mehrheitsabstimmungsspiel (engl. *majority-vote-game*) hergeleitet, das zwischen einem Gewichteverteiler (engl. *weightor*) und einem Auswähler (engl. *chooser*) gespielt wird. Während der *weightor* Gewichte w_i mit $w_i \geq 0$ und $\sum_i w_i = 1$ für die Beobachtungseinheiten mit dem Ziel auswählt, daß alle Objekte in der Mehrheit der Spieldurchläufe gewählt werden müssen, wählt der *chooser* derart Objekte aus, daß eine vorgegebene Gewichtsumme von mindestens $1/2 + \delta$ erreicht wird. Er verfolgt dabei das gegenteilige Ziel, nämlich daß die Objekte in weniger als der Hälfte der Fälle gewählt werden. Boosting steht dann insofern in Verbindung zu diesem Spiel, als es sich um die optimale Strategie des *weightors* handelt. Bei dieser theoretischen Herleitung bleibt die erste Veröffentlichung zunächst stehen, Ergebnisse mit simulierten oder gar empirischen Daten sind hier nicht geliefert und die Auswirkungen auf den für die Statistik deutlich relevanteren Test- oder Generalisierungsfehler werden nicht weiter beachtet.

Wenig später wurden diese theoretischen Boosting-Ansätze aber von Freund & Schapire (1996) auch algorithmisch umgesetzt und zum ersten mal als *AdaBoost*, eine praktikable und einfach zu implementierende Variante, vorgestellt. Hauptanliegen dieses Artikels ist es, die Qualitäten des Verfahrens anhand einer umfangreichen Studie mit 27 empirischen Datensätzen nachzuweisen. Im Gegensatz zu den theoretischen Aussagen über das *majority-vote-game*, die sich auf den Resubstitutionsfehler beziehen, liegt hier nun der Schwerpunkt des Interesses auf dem Generalisierungs- oder Testfehler. Dabei läßt sich feststellen, daß Boosting bei sehr einfachen Klassifikationsverfahren wie Stumps (Klassifikationsbäume mit nur zwei Endknoten) deutlich über Bagging dominiert (bei einer durchschnittlichen Verbesserung um ca. 55% im Vergleich zu ca. 11%). Bei komplexeren Basis-Klassifikationsverfahren wie großen Bäumen mit vielen Endknoten ist Boosting zumindest noch etwas besser, wobei der Unterschied nicht mehr so deutlich zu Tage tritt

(ca. 25% Verbesserung im Vergleich zu ca. 20%). Insgesamt stellen die Autoren fest, daß ein Datensatz günstig für die Anwendung von Boosting ist, wenn er stark unterschiedlich schwer zu klassifizierende Fälle aufweist und Instabilität der Prognose bereits bei leichten Variationen in der Lernstichprobe auftritt. Boosting legt dann höhere Gewichte auf schwer zu klassifizierende Einheiten und sorgt damit für die zur Verbesserung der Prognose nötige Variabilität in der Datengrundlage. Boosting kann sich also die für Bagging als Voraussetzung benötigte Instabilität durch die kontinuierliche Änderung der Verteilung selbst erzeugen.

Auch Breiman (1998) setzt sich daraufhin mit dem neuen Boosting-Verfahren auseinander. Er bezeichnet die Klasse von Verfahren, bei denen Gewichte für Objekte systematisch variiert werden, als *arcing*-Methoden (abgeleitet aus *adaptively resample and combine*), die wiederum noch allgemeiner zusammen mit Bagging zu den *P&C*-Techniken (engl. *perturb and combine*) gezählt werden können. Als Basisverfahren erscheinen instabile Verfahren deshalb am geeignetsten, da sie mit Variabilität auf die ständig fluktuierenden Gewichte (die nicht gegen eine stabile Grenzverteilung konvergieren) beim Boosting reagieren können. Als einen der großen Vorteile dieser Verfahren sieht er die Unabhängigkeit von subjektiv zu wählenden Parametern. Dazu meint der Autor: "*Furthermore, the arc-classifier is off-the-shelf. Its performance does not depend on any tuning or settings for particular problems. Just read in the data and press the start button.*" Er versucht, die Wirkungsweise von Boosting mit einer Bias-Varianz-Zerlegung für die Klassifikation zu erklären, eine Eigenschaft die keineswegs wie bei numerischen Zielgrößen als gängig zu bezeichnen ist und deshalb auch in anderen Definitionen von anderen Autoren vorliegt:

$$\text{Fehlklassifikationsrate}(C) = \text{Fehlklassifikationsrate}(C^*) + \text{Bias}(C) + \text{Varianz}(C)$$

Dabei steht C für ein beliebiges Klassifikationsverfahren und C^* mit

$$C^*(x) = \text{argmax}_j P(Y = j|x)$$

für den Bayes-Klassifizierer. Beim Bias handelt es sich demnach um den persistenten Fehler, die Varianz kann dagegen durch Aggregationstechniken (und unter Umständen auf Kosten eines geringfügig anwachsenden Bias) verringert werden. Eine große Varianz deutet dabei auf ein instabiles Klassifikationsverfahren hin. Hier ist das Potenzial von Boosting demnach am größten.

In einer Vergleichsstudie anhand der zehn Datensätze, die bereits bei der Einführung von Bagging zum Einsatz kamen, sowie vier simulierten und einem weiteren sehr umfangreichen empirischen Datensatz, zeigt der Autor, daß der Erfolg von Boosting nicht an der spezifischen Form des Algorithmus hängt, sondern nur auf das adaptive Resampling zurückzuführen ist. Dies ergibt sich aus dem Vergleich mit einer ad-hoc-Erfindung eines ähnlichen Algorithmus namens X4. Die neuen Gewichte ergeben sich hierbei über

$$\frac{1 + m(x_i)^4}{\sum_{j=1}^{n_L} (1 + m(x_j)^4)}$$

wobei $m(x_i)$ die Anzahl der Fehlklassifikationen von x_i bis zum m-ten Schritt beschreibt. Die abschließende Abstimmung der Teilprognosen erfolgt ungewichtet. Empirisch ergeben sich hier etwa gleichwertige Ergebnisse zum gewöhnlichen Boosting-Algorithmus, obwohl die Techniken doch in vielen Punkten grundverschieden sind. Voraussetzung für eine Verbesserung der Trefferrate ist demnach lediglich, daß die Gewichte der Untersuchungseinheiten schwanken ohne zu konvergieren. Entscheidend ist also der adaptive Resampling-Schritt mit einer Hochgewichtung der schwer zu klassifizierenden Untersuchungseinheiten. Außerdem kann man feststellen, daß eine Verbesserung des Generalisierungsfehlers auch dann noch eintritt, wenn der Resubstitutionsfehler bereits lange die Null erreicht hat. Desweiteren funktioniert Boosting sowohl über Resampling als auch mit gewichteten Daten, was zeigt, daß der Aspekt des Zufalls zwar für Bagging, aber nicht für Boosting nötig ist. Das einzige Problem des Verfahrens scheint nach Meinung des Autors in der Gefahr zu liegen, daß Ausreißer nie richtig klassifiziert werden und dadurch immer höhere Gewichte erhalten, was die Qualität des aggregierten Klassifikationsverfahrens erheblich beeinträchtigen kann. Dadurch ist nämlich die Fluktuation der Gewichte nicht mehr gegeben, vielmehr strebt deren Verteilung gegen eine Gleichverteilung mit den Ausreißern als Träger.

Zur Erklärung der Funktionsweise von Boosting ziehen Schapire, Freund, Bartlett & Lee (1998) den Begriff der Margin heran (siehe auch Breiman (2001)). Diese Größe kann als Vertrauensmaß für die Klassifikation angesehen werden, ein Wert größer als Null sagt aus, daß die Klassifikation richtig erfolgt ist. Die Autoren nutzen nun dieses Maß zu einer alternativen Erklärung anstelle der Bias-Varianz-Zerlegung: Boosting vergrößert, insbesondere im Lerndatensatz, diese Spanne, so daß immer mehr Fälle eindeutig der richtigen Klasse zugeordnet werden können. Dies erklärt auch, warum Boosting nicht oder nur wenig zu Overfitting neigt und warum die Ergebnisse selbst dann noch besser werden, wenn der Fehler in der Lernstichprobe bereits Null erreicht hat. Normalerweise würde man hier nämlich erwarten, daß der Testfehler für immer komplexere Modelle langsam wieder ansteigen müßte. Die Verteilung der Margins ändert sich allerdings auch dann noch weiter, wenn bereits jeder einzelne Wert positiv ist, die Sicherheit der Klassifikation nimmt also immer weiter zu. Ferner zeigen die Autoren einen Zusammenhang zwischen der Maximierung der Margins und einer Verringerung des Generalisierungsfehlers, unabhängig von der Anzahl der kombinierten Klassifizierer. Die Grenzen dieses theoretischen Zusammenhangs sind zwar sehr weich, so daß sie erst ab mehreren 10000 Fällen relevant werden können. Dennoch wird dadurch aber die Richtung des Zusammenhangs aufgezeigt. Boosting zielt speziell auf die Erhöhung von niedrigen Margins ab, so daß sich dieses Maß für die Gewißheit der Klassifikation immer weiter verbessert. Dieses Ziel von Boosting deutet darauf hin, daß zu Support Vector Machines (siehe Bishop & Tipping (2000)), die ebenfalls die Erhöhung der Margins anstreben, ein deutlich engerer Bezug besteht als zu Bagging.

Ein weiterer Meilenstein in der Entwicklung von Boosting liegt durch eine Veröffentlichung von Friedman, Hastie & Tibshirani (2000) vor. Sie stellen Boosting innerhalb des statistischen Forschungsfeldes als eine Methode zur schrittweisen Anpassung eines additiven

Modells

$$F(x) = \sum_{m=1}^{M} f_m(x)$$

dar. Die einfachste und gängigste Variante der Modellanpassung wäre hier die Minimierung des quadratischen Verlusts

$$\mathrm{E}\left(Y - \sum_{m=1}^{M} f_m(x)\right)^2 \quad .$$

Dies entspricht einer Anpassung der Residuen wie in der linearen Regression. Da der quadratische Verlust allerdings ein ungeeignetes Kriterium für Klassifikationsprobleme ist, optimiert AdaBoost eine geeignetere Verlustfunktion, nämlich

$$\mathrm{E}\left(e^{-YF(x)}\right) \quad ,$$

zur Anpassung eines additiven logistischen Regressionsmodells

$$\frac{1}{2}\log\left(\frac{\mathrm{P}(Y=1|x)}{\mathrm{P}(Y=-1|x)}\right) = \sum_{m=1}^{M} f_m(x) \quad .$$

Dieses Kriterium, das sowohl von Discrete als auch von Real AdaBoost, das auf den Klassifikationswahrscheinlichkeiten basiert, optimiert wird, ist nicht identisch, aber sehr ähnlich zur binomialen Log-Likelihood

$$-\log\left(1 + e^{-2YF(x)}\right) \quad ,$$

die das Modell

$$\log\left(\frac{\mathrm{P}(Y=1|x)}{\mathrm{P}(Y=-1|x)}\right) = \sum_{m=1}^{M} f_m(x)$$

anpaßt. Deshalb erscheinen diese beiden Boosting-Varianten als unnötig kompliziert. Näher liegt eine direkte Optimierung der Log-Likelihood, die schließlich in die natürlichere Variante des LogitBoost-Algorithmus mündet. Aufgrund von numerischen Problemen schlagen die Autoren vor, eine willkürlich gewählte Untergrenze für die resultierenden Gewichte heranzuziehen. Auf diese Möglichkeit wurde jedoch im Sinne der Vergleichbarkeit mit anderen Algorithmen, die nicht auf diese Art und Weise stabilisiert werden, im empirischen Teil dieser Arbeit verzichtet.

Als weitere Variante wird noch der sogenannte Gentle AdaBoost vorgestellt, bei dem die Gewichtsanpassungen nicht so extrem ausfallen wie bei dem ähnlichen Real AdaBoost, so daß ein konservativerer, aber numerisch stabilerer Algorithmus entsteht, der sehr gute Ergebnisse auch bei Ausreißern und anderen schwer zu klassifizierenden Fällen liefert. Optimiert wird dabei wiederum die gleiche Zielfunktion $\mathrm{E}(\exp(-YF(x))$.

Eine Vergleichsstudie dieser Ansätze anhand verschiedener simulierter Datensätze zeigt, daß keine der Varianten als die in jedem Fall Beste bezeichnet werden kann. Abhängig

ist die Qualität der Resultate von der Anzahl der Zyklen, der gewählten Komplexität der Basis-Klassifikationsverfahren (große Bäume oder Stumps) sowie der Art der simulierten Grenzziehung zwischen den Klassen. Bei einer zusätzlichen Untersuchung von acht empirischen Datensätzen können schließlich keine großen Unterschiede zwischen den Verfahren festgestellt werden. Die Feststellung, daß in vielen Beispielen wenig bis kein Overfitting auftritt, führen die Autoren darauf zurück, daß in einem additiven Modell die Basisfunktionen f_m von Schritt zu Schritt eine geringere Rolle für das Gesamtmodell spielen.

Zum Abschluß führen die Autoren noch Varianten zur Verkürzung der Rechenzeit ein. Beim Stutzen der Gewichte (engl. *weight trimming*) werden Beobachtungen mit unbedeutend kleinen Gewichten unterhalb einer vorgegebenen Schranke im Modellbildungsschritt komplett weggelassen. Beim Aktualisieren der Gewichte müssen sie dagegen wieder herangezogen werden, da die neuen Gewichte so wieder diesseits der Schranke liegen können. Dennoch kann durch diese Reduzierung des Datensatzes viel Rechenzeit eingespart werden. Ebenso kann das Baumwachstum von vornherein beschränkt werden, anstatt auf das aufwendige Pruning zurückzugreifen.

Schließlich fassen die Autoren zusammen, daß nun aufgrund des geschilderten Zusammenhangs von Boosting mit einem statistischen Modell die Verbesserungen unabhängig von der Frage, ob gewichtete Daten oder Zufallsziehungen verwendet werden, erklärt werden können, während Bagging lediglich ein Verfahren zur Varianzreduktion ist und in keinem Zusammenhang zu Boosting steht. Insgesamt handelt es sich daher sicherlich um eine der wichtigsten und richtungsweisenden Veröffentlichungen zum Thema Boosting, die die entscheidende Verknüpfung vom maschinellen Lernen zur Statistik darstellt.

Schapire (2002) faßt in einer weiteren Arbeit die bis dahin stattgefundenen wichtigsten Strömungen im Feld der Boosting-Algorithmen zusammen. Insbesondere befaßt er sich mit den gängigsten Erklärungsansätzen für die Verbesserung der Resubstitutions- und der Generalisierungsfehlerraten (über theoretische Schranken bzw. über die Margins-Erklärung), mit bis dahin veröffentlichten empirischen Vergleichsstudien, sowie mit den Zusammenhängen zu verwandten Themengebieten. Schwerpunkte liegen hierbei unter Anderem auf der Spieltheorie, der logistischen Regression (mit der konsequenten Entwicklung des LogitBoost) sowie dem Vergleich mit Support Vector Machines (siehe Bishop & Tipping (2000)), die als Ziel die Maximierung der minimalen Margin verfolgen. Alles in allem kann festgehalten werden, daß Boosting eine effiziente Methode ist, um leicht zu findende schwache Lernverfahren ohne zusätzliche Parameterwahl in gute und präzise Klassifikationsverfahren zu verwandeln. Allerdings weist der Autor auch auf die Schwäche hin, daß sich aufgrund der starken Datenabhängigkeit empirisch die Gewichte auf einige wenige vorhandene Ausreißer in den Daten konzentrieren können, die dann zu einer deutlichen Verschlechterung der Prognosefähigkeit des Modells führen. In diesem Fall tritt also auch das empirisch erstaunlich selten beobachtete Overfitting auf. Doch auch aus dieser Schwäche kann ein positiver Effekt gewonnen werden, indem die Gewichte, die im Laufe der Boosting-Zyklen entstehen, als Indikator für die Entdeckung von ebensolchen Ausreißern in den Daten genutzt werden. Zuguterletzt wird noch die Feststellung erwähnt, daß sich der Generalisierungsfehler auch

noch verbessern kann, wenn in der Lernstichprobe bereits eine optimale Anpassung erreicht ist, und daß dadurch die Tendenz von Boosting, nur selten zu Overfitting zu führen, erklärt werden kann.

Mit theoretischen Überlegungen zum Verhalten von aggregierten Klassifikationsverfahren unter unendlich großen Stichprobenräumen beschäftigt sich Breiman (2000). Er teilt dazu die Algorithmen in zwei Klassen ein: Verfahren, die deterministisch ablaufen, gewichtete Daten verwenden, tendenziell eher einfach strukturierte Bäume benötigen und mit gewichteter Abstimmung arbeiten, bezeichnet er wie bereits in früheren Arbeiten als *arcing*-Algorithmen. Sie zeichnen sich dadurch aus, daß sie als Gradienten-Verfahren zur Optimierung einer Zielfunktion angesehen werden können. Ist dagegen die Auswahl der Untersuchungseinheiten oder der Variablen zufallsbasiert, komplexe Bäume liefern tendenziell bessere Ergebnisse und die Abstimmung erfolgt ungewichtet, so spricht er von Randomisierungsalgorithmen. Hier basiert die Erklärung der Funktionsweise primär auf der Präzision der einzelnen Prädiktoren sowie auf deren Korrelationsstruktur (siehe auch Breiman (2001)). Beide Ansätze führen demnach zu etwa gleich guten Resultaten, arbeiten aber von Grund auf verschieden.

Geht man von einem unendlich großen Stichprobenraum aus, so läßt sich zeigen, daß der Generalisierungsfehler von AdaBoost, das zu den *arcing*-Algorithmen gezählt wird, gegen das Bayes-Risiko konvergiert. Außerdem legen Überlegungen zum zugehörigen Beweis nahe, daß beliebig viele andere Verfahren existieren, die zwar jeweils eine andere Verlustfunktion minimieren, aber ebenso gegen das Bayes-Risiko streben. Diese Eigenschaft bestätigt sich aber empirisch bei endlichen Stichproben nicht. Die Funktionsweise von *arcing* interpretiert der Autor so, daß der Schwerpunkt nicht auf der hohen Gewichtung von falsch klassifizierten Fällen, sondern auf der Angleichung der Margins aller Fälle liegt. Dies könnte als Begründung für die Bildung eines Plateaus bezüglich der Fehlerrate bei endlichen Stichproben herangezogen werden. Unklar bleibt aber weiter der Zusammenhang, warum diese Plateaubildung bezüglich der Lerndaten zu einer Verbesserung auf den Testdaten führt.

Ein neuer Blickwinkel auf Boosting wird in Friedman (2001) aufgezeigt. Er stellt eine allgemeine Äquivalenz zwischen der schrittweisen numerischen Optimierung durch die Anpassung in Richtung des steilsten Abfalls des Gradienten einer Funktion auf der einen und Boosting-Algorithmen auf der anderen Seite fest. Daraus lassen sich Gradient-Boosting-Techniken für verschiedenste Verlustfunktionen entwickeln. Wichtige Beispiele sind der quadratische Verlust, der äquivalent zum L_2-Boost ist, oder die logistische Verlustfunktion für den Zwei- oder Mehrklassenfall, die damit in die entsprechenden LogitBoost-Algorithmen mündet. Aber auch jede andere Verlustfunktion kann herangezogen werden, so daß über den bereits bekannten Term $e^{-YF(x)}$ beispielsweise auch die gewöhnliche AdaBoost-Methode in diese allgemeine Klasse von Algorithmen integrierbar ist.

Eine Besonderheit dieses Ansatzes ist es, daß für jede Art von Verlustfunktion und egal, ob es sich um Regressions- oder Klassifikationsprobleme handelt, stets gewöhnliche Regressionsbäume, die nach dem KQ-Prinzip arbeiten, als Basisverfahren verwendet werden

können. Dies liegt daran, daß der steilste Abstieg für eine Klasse von Funktionen nicht exakt bestimmt werden kann, da der Gradient nur in Datenpunkten definiert ist. Deshalb wählt man empirisch diejenige Funktion, die am nächsten zum optimalen Abfall liegt. Man bestimmt sie sinnvollerweise durch die KQ-Methode, wofür die Wahl von gewöhnlichen Regressionsbäumen als Basisverfahren immer geeignet ist.

Sei $h(x;\theta)$ ein Regressionsbaum, wobei θ die Parameter, sprich Splitwerte und -variablen zusammenfaßt. Dann wird im Initialisierungsschritt zunächst der Ausgangsschätzer

$$F_0(x) = \text{argmin}_\rho \sum_{i=1}^{N} L(Y_i, \rho)$$

anhand der gewählten Verlustfunktion L bestimmt. Bei quadratischem Verlust ist dieses ρ beispielsweise das arithmetische Mittel $\bar{Y}$, bei Betragsverlust der Median von Y.

Nun wird für jeden Zyklus $m = 1, \ldots, M$ des Algorithmus zunächst der negative Gradientenvektor

$$\tilde{Y}_i = -\left[\frac{\partial L(Y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

ermittelt. Anschließend paßt man einen neuen Regressionsbaum an diese Gradienten an. Dessen Parameter lauten

$$\theta_m = \text{argmin}_{\theta,\beta} \sum_{i=1}^{N} \left(\tilde{Y}_i - \beta h(x_i;\theta)\right)^2 \quad .$$

Im nächsten Schritt kann dann die Minimierung längs des steilsten Abstiegs der Verlustfunktion mit Hilfe des Gradienten vorgenommen werden. Die optimale Schrittweite wird über

$$\rho_m = \text{argmin}_\rho \sum_{i=1}^{N} L\left(Y_i, F_{m-1}(x_i) + \rho h(x_i, \theta_m)\right)$$

bestimmt. Der daraus aktualisierte Klassifikator lautet

$$F_m(x) = F_{m-1}(x) + \rho_m h(x, \theta_m) \quad .$$

Für derartige schrittweise Vorgehensweisen existieren zwei verschiedene englische Fachtermini, die unterschieden werden müssen: Während bei einem schrittweise (engl. *stepwise*) arbeitenden Algorithmus die Terme aus früheren Schritten immer wieder angepaßt werden, bedeutet phasenweise (engl. *stagewise*), daß frühere Terme unverändert bleiben und lediglich neue Terme hinzukommen. In diesem Sinne handelt es sich hier also um eine auf dem *stagewise*-Prinzip basierende Vorgehensweise.

Neu und interessant für Regressionsprobleme ist insbesondere ein Algorithmus, der auf dem Betragsabstand als Verlustfunktion basiert. Hier entsteht als Residuum lediglich das Vorzeichen $\text{sign}(Y - F(x))$ der Abweichung zwischen wahrer Klassenzugehörigkeit und Prognose. Dieser Ansatz ist die unkompliziertere Alternative dazu, tatsächlich Bäume bezüglich Betragsabständen zu optimieren.

Friedman (1999) verfeinert diesen eben geschilderten Ansatz durch zusätzlich eingebrachte Zufallsaspekte: Das Basis-Klassifikationsverfahren (hier wird wiederum mit Bäumen gearbeitet) erhält nur zufällige Teilstichproben anstelle aller Daten. Umfangreiche Simulationsstudien zeigen eine Verbesserung gegenüber der deterministischen Variante, die allerdings im Regressionsproblem deutlicher ausfällt als bei Klassifikation. Als eine mögliche Erklärung für diese Verringerung des Fehlers, die auch auf viele andere Aggregationstechniken, die mit Zufallsvektoren arbeiten (wie zum Beispiel Random Forests), angewendet werden kann, wird angeführt, daß der Zufall für eine größere Varianz zwischen den Basis-Klassifikationsverfahren sorgt, gleichzeitig aber auch für eine geringere Korrelation. Durch den Aggregationsschritt wird die Varianz des Gesamtmodells wieder verringert, aber pro Basis-Klassifikator kommt aufgrund der geringen Korrelationen weiterhin mehr neue Information hinzu. Dieser positive Effekt des Zufallsaspekts liefert also auch eine mögliche Erklärung dafür, daß deterministische AdaBoost-Algorithmen, die mit gewichteten Daten arbeiten, nicht unbedingt besser sein müssen als stochastische Varianten, die auf gewichteten Zufallsziehungen aus der Lernstichprobe basieren.

Aufbauend auf den beiden letztgenannten Arbeiten setzen Bühlmann & Yu (2002b) an: Aus der neuen Sichtweise von Boosting als Kleinst-Quadrat-Gradienten-Verfahren entwickeln sie ausführlich den L_2-Boost. Anstatt der bisherigen Verlustfunktionen

$$e^{-YF(x)} \quad \text{bzw.} \quad \log(1+e^{-2YF(x)})$$

widmen sie sich dem L_2-Kriterium, also

$$\frac{(Y-F(x))^2}{2} \quad \text{für } Y \in \{-1;1\} \quad .$$

Während die beiden ersteren Kriterien zu der bekannten Lösung

$$F(x) = \frac{1}{2}\log\left(\frac{\mathrm{P}(Y=1|x)}{\mathrm{P}(Y=-1|x)}\right)$$

führen, steuert das L_2-Kriterium auf

$$F(x) = \mathrm{E}(Y|x)$$

zu. Empirisch wird diese Lösung durch ein Gradienten-Abstiegsverfahren angenähert, was einem schrittweisen Anpassen der Residuen entspricht. Das Basis-Verfahren, auf dem der Boosting-Algorithmus aufsetzt, muß dabei jeweils bezüglich des verwendeten Kriteriums optimiert werden. Im Fall des L_2-Kriteriums ist dies besonders einfach, denn man kann mit gewöhnlichen Regressionsbäumen arbeiten, die den quadratischen Verlust minimieren. Außerdem ist der negative Gradient gerade der klassische Residuenvektor aus der gewöhnlichen linearen Regression. Die Besonderheit dieses Algorithmus, der in Abschnitt 2.4.5 beschrieben ist, besteht darin, daß er im Gegensatz zu allen anderen Boosting-Verfahren keine Gewichtung der Beobachtungen benötigt.

Da ein wiederholtes Anpassen der verbliebenen Residuen bei zu komplexen Basisverfahren unweigerlich zu Overfitting führen würde, empfehlen die Autoren pauschal mit Stumps zu arbeiten. Außerdem wird speziell durch die Struktur dieses Algorithmus klar, daß beim Hinzufügen eines weiteren Boosting-Schrittes die Komplexität des Gesamtverfahrens nicht um einen konstanten Betrag, sondern nur exponentiell verringert steigt, da jeder neue Term von den früheren abhängt. Grund dafür ist die Tatsache, daß die jeweils nächsten Residuen auf den Klassifikatoren zuvor basieren. Dies könnte die relative Resistenz gegenüber Overfitting erklären.

Da allgemein bei diesem Ansatz eine Funktion geschätzt wird, die die erwarteten Kosten einer gewählten Verlustfunktion minimiert, muß zur Anwendung auf das hier interessierende Feld der Klassifikation die Approximation in jedem Schritt zusätzlich auf das Intervall von -1 bis 1 eingeschränkt werden, denn auch $\mathrm{E}(Y|x)$ ist auf diesen Bereich beschränkt.

Die Autoren halten zum Abschluß einiger empirischer Betrachtungen fest, daß bei hochdimensionalen Problemen möglichst einfache Basis-Klassifizierer wie Stumps ideal sind und daß ganz allgemein zu gute Basis-Techniken zu Overfitting führen können. Dies führt wieder zu der ursprünglichen Entwicklung von Boosting zurück, die ja auf dem Begriff des schwachen Lernverfahrens basierte. Außerdem arbeitet Boosting, wie schon von mehreren Autoren festgestellt, sehr schlecht bei Fehlern in der Klassenvariable und führt zu besonders extremem Overfitting. Grund scheint hier zu sein, daß selbst sehr schwache Lernverfahren wie Stumps zu stark sind, das heißt, sie passen sich an die störenden Fehler in den Daten bereits zu gut an, so daß die Gesamtfehlerrate schnell ansteigt.

In einer weiteren Betrachtung seines L_2-Boost leitet Bühlmann (2002) unter anderem die bisher allgemeingültigste Kosistenzaussage her, nämlich die Bayes-Risiko-Konsistenz dieses Algorithmus mit CART als Basis-Verfahren und einer Endknotenzahl pro Baum, die die Anzahl der Prädiktoren um mindestens 1 übersteigt. Als einzige Voraussetzungen gehen hier die *iid*-Struktur der Daten, also die Annahme, daß alle Untersuchungseinheiten unabhängig aus einer identischen Verteilung stammen, sowie die Forderung, daß die Endknoten der beteiligten Bäume ausreichend viele Beobachtungen beinhalten, ein.

Zum Abschluß der Übersicht über Entwicklungen bezüglich Boosting sei noch ein Regressionsansatz namens LARS (abgeleitet aus *least angle regression*) von Efron, Hastie, Johnstone & Tibshirani (2002) erwähnt, der starke Ähnlichkeit zum Boosting aufweist. Die Methode basiert auf der Vorwärtsselektion von Variablen: Zunächst wird die Modellanpassung in Richtung derjenigen Dimension oder Kovariate X_i vorgenommen, die die höchste Korrelation zur Zielgröße Y aufweist. Dies wird so lange fortgesetzt, bis eine zweite Variable X_j eine genauso hohe Korrelation zu dem bis dahin erzielten Residuum aufweist. Ab hier wird zwischen den Richtungen dieser beiden Variablen gemittelt, also die Winkelhalbierende gewählt, bis wiederum eine dritte Kovariate die gleiche Korrelation aufweist. So wird weiter fortgefahren, das heißt, man wählt immer die *least-angle*-Richtung aller bis dahin beteiligten Kovariaten, daher der Name des Verfahrens. Bei dieser Vorgehensweise handelt es sich also um eine Art von adaptivem Anpassen wie beim Boosting, was sich vor allem beim Vergleich mit den Ansätzen von Friedman (2001) zeigt. Dieser effiziente

Algorithmus könnte demnach möglicherweise in Zukunft auch im Feld des Boosting zur Anwendung kommen, weshalb dieses Regressionsverfahren hier, allerdings nur am Rande, Erwähnung finden soll.

Nach diesem Überblick über die Anfänge und weiterführende Entwicklungen im Bereich von Bagging und Boosting folgt nun noch eine kurze Zusammenfassung einiger weiterer wichtiger Vergleichsstudien in der Literatur. Beginnen kann man hier mit einer Arbeit von Bauer & Kohavi (1999), die Vergleiche zwischen Bagging, Boosting und der oben geschilderten Ad-hoc-Variante zum herkömmlichen Boosting-Algorithmus von Breiman (1998) anstellen. Beide Boosting-Varianten werden dabei sowohl mit gewichteten Daten als auch mit Zufallsstichproben ausprobiert. Ferner kommen als Basis-Klassifikationsverfahren komplexe Bäume, Stumps und ein naiver Bayes-Ansatz zur Anwendung. Dieser basiert auf der Annahme von bedingter Unabhängigkeit der Kovariaten $x_1, \ldots, x_p$ gegeben Y und resultiert aus dem Zusammenhang

$$\mathrm{P}(Y|x) \propto \mathrm{P}(x|Y)\mathrm{P}(Y) = \prod_{i=1}^{p} \mathrm{P}(x_i|Y)\mathrm{P}(Y) \quad .$$

Die wichtigsten betrachteten Gütekriterien sind die absolute sowie die relative Reduktion des Fehlers, die dann jeweils über alle betrachteten Datensätze gemittelt werden. Versuche, den MSE auf Klassifikationswahrscheinlichkeiten als Gütekriterium anzuwenden, schlagen bei Boosting fehl, da dieses Verfahren als Wahrscheinlichkeitsschätzer nicht gut geeignet ist, sondern als reine 0-1-Klassifikationstechnik betrachtet werden muß. Pauschal wird bei allen Aggregationstechniken mit 25 Zyklen gearbeitet, damit langsame Verfahren, die viele Iterationen bis zu einem brauchbaren Ergebnis benötigen, nicht überbewertet werden. Außerdem spielt der Faktor Rechenzeit bei großen Datensätzen eine nicht zu vernachlässigende Rolle. Bei der Analyse von 14 großen Datensätzen mit 1000 bis 58000 Fällen stellen die Autoren fest, daß Boosting im Durchschnitt am besten abschneidet, jedoch keineswegs in jedem Fall das beste Verfahren ist. Speziell bei durch viele Ausreißer verrauschten Daten zeigen sich bei Boosting Probleme, da diese nicht korrekt klassifizierbaren Punkte sehr hohe Gewichte erhalten und so das Gesamtergebnis empfindlich stören. Bagging erweist sich hier als deutlich stabiler, erzielt aber ansonsten die geringere Verbesserung, da es lediglich auf Varianzreduktion basiert. Breimans Ad-hoc-Variante für Boosting funktioniert dagegen nur in Zusammenhang mit Zufallsstichproben und nicht bei gewichteten Daten. Manchmal führt aber auch keines der aggregierten Verfahren zu einer Verbesserung. Für die Behandlung von verrauschten Daten empfehlen die Autoren ein Verfahren, das sie als *Wagging* (von engl. *weighted aggregation*) bezeichnen und bei dem lediglich weißes Rauschen auf die (ursprünglich identischen) Ziehungswahrscheinlichkeiten beim Bagging addiert wird. Diese werden dann als Gewichte statt als Ziehungswahrscheinlichkeiten verwendet, um in jedem Schritt den kompletten Datensatz nutzen zu können. Außerdem wird eine probabilistische Version von Bagging (*p-Bagging*) vorgestellt, die statt der Abstimmung von 0-1-Klassifikatoren mit gemittelten Klassifikationswahrscheinlichkeiten arbeitet. Neben diesen und weiteren Variationen führen sie noch eine Reihe weiterer Details an, die die numerische Stabilität der Boosting-Algorithmen sichern sollen.

Dietterich (2000) vergleicht CART, Bagging und Boosting mit einer Methode, die er als *Randomisierung* bezeichnet. Hierbei werden interne Entscheidungen im Klassifikationsverfahren zufällig variiert. Im Speziellen bedeutet dies bei Bäumen, daß jeder Split zufällig aus den 20 besten ermittelt wird. Zum Vergleich werden 33 Datensätze herangezogen, bei denen teilweise auch Versuche mit verrauschten Klassenzugehörigkeiten, also einigen zufällig veränderten Werten der Zielgröße, unternommen werden. Es zeigt sich, daß ohne verrauschte Daten Boosting dominiert, während Bagging und Randomisierung etwa gleichauf liegen. Mit verrauschten Daten schneidet Boosting dagegen sehr schlecht ab, teilweise sogar schlechter als CART, während Bagging hier am besten ist. Dieser deutliche Abfall von Boosting dürfte wie schon oben erwähnt darauf zurückzuführen sein, daß sehr hohe Gewichte auf die nicht klassifizierbaren verrauschten Punkte gelegt werden, die dann das Gesamtergebnis des Verfahrens gravierend beeinträchtigen.

Dudoit, Fridlyand & Speed (2002) beschäftigen sich speziell mit Daten, die sehr viele Variablen, aber nur wenige Untersuchungseinheiten aufweisen, wie es im Forschungsfeld der Genetik oft der Fall ist. Verglichen werden dabei die lineare Diskriminanzanalyse, k-Nächste-Nachbarn-Verfahren, CART, Bagging, Boosting sowie einige andere Varianten von Klassifikationsverfahren, unter anderem auch Bagging-Versionen, die auf konvexen Pseudo-Daten oder einem parametrischen Bootstrap als Alternative zum gewöhnlichen nichtparametrischen Bootstrap basieren. Drei große Datensätze mit bereits vorselektierten Kovariaten führen zu dem Ergebnis, daß sich für diese komplexen Klassifikationsprobleme die Nächste-Nachbarn-Verfahren am besten eignen. Es folgen Boosting und Bagging, während CART und lineare Diskriminanzanalyse am schlechtesten abschneiden. Als Nebenprodukt wird in dieser Arbeit mit den sogenannten *prediction votes*, die in Abschnitt 2.3 bereits eingeführt wurden, ein Präzisionskriterium vorgestellt, das im Rahmen der Entwicklungen von ordinalen Boosting-Varianten in der vorliegenden Arbeit ebenfalls mit untersucht, allerdings bei der endgültigen Implementierung nicht mehr berücksichtigt wurde.

Auch Dettling & Bühlmann (2003) interessieren sich für die Klassifikation bei hochdimensionalen und verrauschten Microarray-Daten mit geringen Stichprobenumfängen. Unter verrauschten Daten verstehen sie dabei Fehlspezifikationen in der Klassenvariable, wie sie typisch für diese Art von biologischen Daten sind. In diesem Artikel wird versucht, Boosting möglichst gut an diese Gegebenheiten anzupassen: Nach einer automatischen Variablenselektion über eine nichtparametrische Scoring-Methode, die auf der Teststatistik des Zwei-Stichproben-Wilcoxon-Tests zum Vergleich der Mittelwerte in den beiden Klassen der Zielgröße basiert, wird LogitBoost mit Stumps, das nach Ansicht der Autoren am robustesten gegen derartige Verrauschung ist, verwendet. Um mit Mehrklassenproblemen umzugehen, bieten die Autoren verschiedene Modelle an, wobei sie sich letztendlich für den *one-against-all*-Ansatz entscheiden, der empirisch besser abschneidet als die direkte Mehrklassenvariante des LogitBoost. Empirisch zeigen sie die guten Resultate ihrer Methode anhand von sechs Datensätzen und einer Simulationsstudie.

Bühlmann (2003) entwickelt in seiner oben bereits erwähnten Arbeit eine weitere vielversprechende Kombination von Verfahren: Beim sogenannten *BagBoosting* verwendet er

als Basis-Klassifikationsverfahren für beliebige Boosting-Algorithmen statt einzelner Klassifikationsbäume bereits mit Hilfe von Bagging aggregierte Klassifikatoren, die wiederum Stumps als Basis verwenden. Hier werden also Boosting und Bagging in ein gemischtes Verfahren integriert.

In einer Folgeveröffentlichung arbeitet Dettling (2004) mit diesem heuristischen *BagBoosting*. Durch die Verwendung von Bagging mit Stumps als Basisklassifizierer für LogitBoost sollen die Vorteile von Bagging und Boosting (Reduzierung von Varianz bzw. Bias) kombiniert werden. Der Ansatz wird anhand einer Vergleichsstudie evaluiert, die sechs empirische Microarray-Datensätze sowie simulierte Daten, die jedoch aus realen Kovarianzstrukturen und Mittelwerten gebildet werden, umfaßt. Die simulierten Daten haben den Vorteil, daß aufgrund der Unabhängigkeit der einzelnen Beobachtungen Konfidenzintervalle und statistische Tests ermöglicht werden, was bei Kreuzvalidierung von realen Daten nicht möglich ist. Hier würde die Konfidenzintervallbreite aufgrund der Datenüberlappung in den verschiedenen Versuchswiederholungen unterschätzt. Der Vergleich mit CART, Bagging, LogitBoost, Random Forests, Support Vector Machines, Nächste-Nachbarn-Techniken sowie linearer Diskriminanzanalyse zeigt, daß BagBoosting etwas besser als die ohnehin schon sehr guten Random Forests und Support Vector Machines abschneidet, während Nächste-Nachbarn und lineare Diskriminanzanalyse zumindest bei nicht zu großen Datensätzen gerade noch zufriedenstellend arbeiten. Boosting und Bagging alleine, besonders jedoch CART, schneiden dagegen sehr schlecht ab. Tests anhand der simulierten Daten zeigen, daß die Überlegenheit von BagBoosting gegenüber den übrigen Verfahren tatsächlich statistisch signifikant ist. Zusätzlich wird anhand des integrierten MSE bestätigt, daß sowohl Varianz als auch Bias verringert werden. Abschließend werden noch zwei wichtige Aussagen getroffen: Erstens scheint ein Zufallsaspekt statt Determinismus in Ensemble-Methoden zu besseren Ergebnissen zu führen, was viele Forscher unabhängig voneinander feststellen konnten und was auch in dieser Arbeit augenscheinlich wurde. Zweitens können in Bezug auf Microarray-Daten aufgrund der extrem hohen Korrelationen der Gene untereinander nur sehr schwer die für eine Krankheit verantwortlichen Gene herausgefiltert werden, weshalb nach bisherigem Forschungsstand lediglich Prädiktion ohne Variablenselektion betrieben werden kann.

Kapitel 3

Ordinale Klassifikation

Nachdem sich das zweite Kapitel mit den Ensemble-Verfahren zur Verbesserung von Klassifikationsmethoden für nominale Zielgrößen beschäftigt hat, wird nun auf den zweiten Schwerpunkt dieser Arbeit, nämlich die ordinale Klassifikation, eingegangen.

3.1 Der sequenzielle Ansatz

Ein erster Ansatz für die Einbindung von Ordinalität, nämlich die sequenzielle Methode, stammt ursprünglich aus dem Bereich der Regression mit Hilfe von generalisierten linearen Modellen. Modelliert werden sollen dabei die bedingten Wahrscheinlichkeiten

$$\begin{aligned}
&\mathrm{P}(Y = 1|X = x)\\
&\mathrm{P}(Y = 2|Y \geq 2, X = x)\\
&\vdots\\
&\mathrm{P}(Y = k-1|Y \geq k-1, X = x)
\end{aligned}$$

Dabei wird die ordinale Größe Y in mehrere binäre Variablen zerlegt, wie es die Skizze 3.1 verdeutlichen soll: Man geht davon aus, daß die niedrigste Klasse sozusagen einen natürlichen Urzustand (zum Beispiel "gesund" im Gegensatz zu fortschreitenden Erkrankungsstufen) darstellt. Deshalb wird zunächst eine Variable betrachtet, die nur die erste von den restlichen Klassen trennt ($\{1\}$ gegen $\{2, 3, 4, \ldots, k\}$) und für diese binäre Zielgröße ein gewöhnliches nominales Verfahren angewendet. Für die Bestimmung des Modells für den nächsten Übergang von Stufe 2 in eine noch höhere Klasse ($\{2\}$ gegen $\{3, 4, \ldots, k\}$) werden dann nur noch diejenigen Untersuchungseinheiten aus der Lernstichprobe herangezogen, die nicht aus der ersten, bereits übersprungenen Klasse stammen. Auf diese Weise

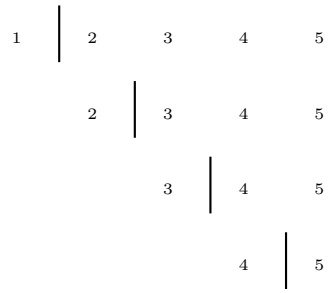

Abbildung 3.1: Skizze zur sequenziellen Zerlegung

wird für jeden Übergang verfahren, bis man schließlich beim Modell für den letzten Übergang ($\{k-1\}$ gegen $\{k\}$) nur noch diejenigen Daten verwendet, die aus einer der beiden höchsten Klassen stammen. Ein Nachteil dieser Methode ist damit offensichtlich, nämlich daß bei späteren Übergängen immer weniger Daten genutzt werden können, da immer nur diejenigen Fälle zur Modellbildung herangezogen werden, die nicht aus einer bereits übersprungenen Klasse stammen.

Formal ausgedrückt wird also die Klassenvariable $Y \in \{1, \ldots, k\}$ zerlegt in $k-1$ binäre Größen

$$Y^{(r)} = \begin{cases} -1 & \text{für } Y = r \\ +1 & \text{für } Y \in \{r+1, \ldots, k\} \end{cases}$$

mit $r = 1, \ldots, k-1$. Für jede dieser Variablen entwickelt man dann einen Klassifikator $C^{(r)}(x) = \hat{Y}^{(r)} \in \{-1, 1\}$.

Die eigentliche Klassifikation kann dann nach der Bildung der einzelnen Modelle wie folgt durchgeführt werden: Vom ersten Übergang von Klasse 1 nach 2 ausgehend, durchläuft das zu klassifizierende Objekt sequenziell die Modelle für die Übergänge, bis es zum ersten Mal in die jeweils tiefere Klasse eingeordnet wird, das heißt, bis alle höheren Klassen ausscheiden. Auf diese Weise wird jedes Objekt eindeutig einer Klasse zugeordnet.

3.2 Der kumulative Ansatz

Auch der kumulative Ansatz stammt in seiner parametrischen Variante, nämlich der Schwellenwert-Methode, ursprünglich aus dem Bereich der generalisierten linearen Modelle. Hier werden Wahrscheinlichkeiten für die Klassifikation der Zielgröße Y in Abhängigkeit von den Einflußgrößen x modelliert:

$$\begin{aligned} \mathrm{P}(Y \leq 1) &= \mathrm{F}(\beta_{0,1} + x'\beta_1) \\ \mathrm{P}(Y \leq 2) &= \mathrm{F}(\beta_{0,2} + x'\beta_2) \\ &\vdots \\ \mathrm{P}(Y \leq k-1) &= \mathrm{F}(\beta_{0,k-1} + x'\beta_{k-1}) \end{aligned}$$

Ein alternatives, etwas strenger restringiertes Modell ergibt sich, wenn die auf x multiplikativ einwirkenden Parameter $\beta_1, \ldots, \beta_{k-1}$ als identisch vorausgesetzt werden:

$$\begin{aligned} \mathrm{P}(Y \leq 1) &= \mathrm{F}(\theta_1 + x'\beta) \\ \mathrm{P}(Y \leq 2) &= \mathrm{F}(\theta_2 + x'\beta) \\ &\vdots \\ \mathrm{P}(Y \leq k-1) &= \mathrm{F}(\theta_{k-1} + x'\beta) \end{aligned}$$

Die Parameter $\theta_1, \ldots, \theta_{k-1}$ können dabei als Schwellenwerte angesehen werden, die festlegen, ob der jeweilige Übergang zu einer höheren Klasse noch stattgefunden hat oder nicht. Daher stammt auch die Bezeichnung des Ansatzes. Auf diese Art und Weise entsteht wiederum für jeden Übergang ein eigenes Modell in Form von sich überlagernden Wahrscheinlichkeitsverteilungen, wobei mit Hilfe der so bestimmten kumulierten Wahrscheinlichkeiten eine eindeutige Zuordnung zu einer der k Klassen möglich wird. Da die Übergänge nicht sequenziell betrachtet werden, sondern jeweils alle Klassen kumulativ berücksichtigt werden (also beispielsweise $\{1\}$ gegen $\{2, 3\}$ und $\{1, 2\}$ gegen $\{3\}$ bei drei ordinalen Klassen), können auch bei den späteren Übergängen alle Daten verwendet werden.

Da mit CART ein nichtparametrisches Verfahren als Basis für Aggregationstechniken verwendet wird, kann diese Idee jedoch hier nicht unmittelbar zur Anwendung kommen, denn bei binären Bäumen werden keine linearen Modellansätze für eine Schätzung der Klassifikationswahrscheinlichkeiten vorgenommen. Bei anderen parametrischen Klassifikationsverfahren, wie sie zu Beginn des zweiten Kapitels kurz vorgestellt wurden, könnten Ansätze mit Hilfe der Schwellenwert-Methode dagegen durchaus angewendet werden.

Bei der als *Voting-Methode* zu bezeichnenden nichtparametrischen Variante, die damit auch bei Klassifikationsbäumen angewendet werden kann und ebenso wie Aggregationsverfahren auf einer Abstimmung zwischen mehreren Modellen basiert, handelt es sich um einen viel versprechenden Ansatz. Hierbei wird wiederum jeder Übergang einzeln unter Einbeziehung aller Daten (also kumulativ, ebenso wie bei der Schwellenwert-Methode, beispielsweise $\{1\}$ gegen $\{2, 3\}$ oder $\{1, 2\}$ gegen $\{3\}$) modelliert. Da alle Klassen kumulativ berücksichtigt werden, können also auch für spätere Übergänge alle Daten verwendet werden. Die ordinale Zielgröße wird dazu auch hier in mehrere binäre Variablen zerlegt.

Formal wird also die Klassenvariable Y mit k Klassen wieder in $k-1$ binäre Größen zerlegt, die in diesem Fall

$$Y^{(r)} = \begin{cases} -1 & \text{für } Y \in \{1, \ldots, r\} \\ +1 & \text{für } Y \in \{r+1, \ldots, k\} \end{cases}$$

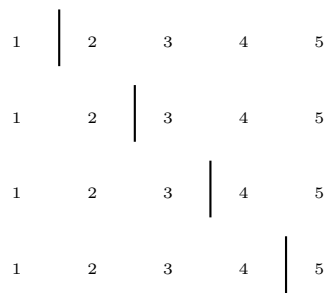

Abbildung 3.2: Skizze zur kumulativen Zerlegung

mit $r = 1, \ldots, k-1$ lauten. Auch hier kann dann für jede dieser Variablen ein eigener Klassifikator $C^{(r)}(x) = \hat{Y}^{(r)} \in \{-1, 1\}$ bestimmt werden.

Die Zuordnung erfolgt wiederum durch Abstimmung zwischen den einzelnen Modellen. Dabei kann man verschiedene Varianten des Votings unterscheiden. Es folgen einige typische Möglichkeiten, bei denen jeweils ein aus der Aggregation resultierender Scorevektor

$$S(x) = (S_1(x), \ldots, S_k(x))'$$

für die k Klassen der Zielgröße angegeben wird. Die endgültige Prognose erfolgt jeweils in die Klasse r, die den höchsten Score $S_r(x)$ aufweist.

- Bei strikten 0-1-Entscheidungen wird für jede zugeordnete Klasse eine Stimme vergeben (Beispiel: Modell für $\{1, 2\}$ gegen $\{3, 4, 5\}$ stimmt für die zweite, höhere Klassengruppe; dann erhalten die Klassen 3, 4 und 5 jeweils einen Punkt, die Klassen 1 und 2 dagegen keinen). Formal setzt man:

$$S_r(x) = \sum_{j=1}^{r-1} I(C^{(j)}(x) = 1) + \sum_{j=r}^{k-1} I(C^{(j)}(x) = -1)$$

- Jede Klasse erhält ihre Zuordnungswahrscheinlichkeit als Score (Beispiel: Modell für $\{1, 2\}$ gegen $\{3, 4, 5\}$ wie oben hat die Zuordnungswahrscheinlichkeiten 0.3 für die niedrigere und 0.7 für die höhere Klassengruppe; dann erhalten die Klassen 3, 4 und 5 jeweils 0.7 Punkte, die Klassen 1 und 2 dagegen nur 0.3). Formal setzt man:

$$S_r(x) = \sum_{j=1}^{r-1} \mathrm{P}(C^{(j)}(x) = 1) + \sum_{j=r}^{k-1} \mathrm{P}(C^{(j)}(x) = -1)$$

- Als Score kann bei der Verwendung von Aggregations-Verfahren wie Bagging oder Boosting die Anzahl der Stimmen für jede Klasse übernommen werden (Beispiel: Zehn Modelle für $\{1, 2\}$ gegen $\{3, 4, 5\}$ wie oben, wovon zweimal für die niedrigere

und achtmal für die höhere Klassengruppe gestimmt wird; dann erhalten die Klassen 3, 4 und 5 jeweils 8 Punkte, die Klassen 1 und 2 dagegen nur 2). Formal setzt man:

$$S_r(x) = \sum_{j=1}^{r-1} \left(\sum_{m=1}^{M} I(C^{(j)}(x, L_m) = 1) \right) + \sum_{j=r}^{k-1} \left(\sum_{m=1}^{M} I(C^{(j)}(x, L_m) = -1) \right)$$

3.3 Regressionsbasierte Ansätze

Wie bereits erwähnt, befindet sich das ordinale Skalenniveau zwischen der nominalen Struktur, bei der die Klassen keine Ordnung aufweisen, und der Kardinalskala, bei der Differenzen oder sogar Verhältnisse zwischen verschiedenen Ausprägungen erklärt sind. Die bisher geschilderten Möglichkeiten zur ordinalen Klassifikation nähern sich der Aufgabenstellung von der nominalen Seite her. Durch Umkodierung wird die ordinale Zielgröße in mehrere binäre Variablen, die selbstverständlich nur noch nominales Skalenniveau aufweisen, zerlegt. Anschließend werden Techniken für nominale Variablen verwendet und erst am Ende wieder zu einer ordinalen Prognose kombiniert.

Ebenso könnte man sich der Aufgabenstellung aber auch von der anderen Seite her nähern: Interpretiert man die ordinalen Klassen als gruppierte kardinalskalierte Größen, so erscheint es plausibel, Regressionstechniken, die die angemessene Vorgehensweise bei kardinalskalierten Zielgrößen darstellen, anzuwenden. Der Ansatz besteht also darin, so zu tun, als handle es sich bei den ordinalen Klassenlabels um Ausprägungen einer kardinalskalierten Variable und ein beliebiges Regressionsverfahren anzuwenden. In dieser Arbeit, in der sämtliche nominalen Aufgabenstellungen mit Klassifikationsbäumen angegangen werden, ist es hier naheliegend, das regressionsbasierte Äquivalent, nämlich Regressionsbäume, heranzuziehen. Die Prognosen, die ein solches Verfahren liefert, sind dann aber natürlich nicht mehr auf den zulässigen diskreten Wertebereich der ordinalen Zielgröße beschränkt. Daher bietet es sich an, diese Rückführung auf das ordinale Skalenniveau durch Runden auf die nächstgelegene ordinale Klasse durchzuführen.

Im Gegensatz zu den zuerst geschilderten sequenziellen und kumulativen Ansätzen weist diese regressionsbasierte Technik einige angreifbare Punkte auf. So erscheint es nicht angebracht, für eine Variable, bei der Differenzen eigentlich nicht definiert sind, Regressionsverfahren anzuwenden, und das Runden bedeutet ja implizit ebenfalls wieder, daß nicht erklärte Abstände zu bzw. zwischen Klassenlabels genutzt werden. Andererseits kann ein solches Runden aber auch so gerechtfertigt werden, daß man sich mit seiner Prognose eben näher an der einen als an der anderen Klasse befindet. Auch wenn dieser Ansatz also durchaus angreifbar erscheint, soll er zumindest einige wenige Male im Rahmen der später geschilderten ordinalen Verfahren zum Einsatz kommen.

3.4 Ordinale Klassifikation in der Literatur

Anderson & Philips (1981) interpretieren ordinale Variablen Y als Messung einer latenten metrischen Größe Z, die aufgrund beschränkter Meßmittel nicht besser erfaßbar ist. Um eine solche Variable als Zielgröße zu modellieren, wird ein ordinales Regressionsmodell für die bedingte Verteilung

$$P(Y \leq s|x) = \Psi(\theta_s - \beta' x)$$

mit $s = 1, \ldots, k-1$ herangezogen, wobei die Art des Modells von der Wahl von Ψ abhängt. Die gängigsten Varianten sind hier das logistische Modell

$$\Psi(\theta_s - \beta' x) = \frac{\exp(\theta_s - \beta' x)}{1 + \exp(\theta_s - \beta' x)}$$

sowie das Probitmodell

$$\Psi(\theta_s - \beta' x) = \Phi(\theta_s - \beta' x) \quad .$$

In dieser Arbeit zeigen die Autoren Methoden der Parameterschätzung für verschiedene Samplingmethoden auf. Als Prognose kann dann diejenige Kategorie gewählt werden, die die höchste geschätzte a posteriori-Wahrscheinlichkeit aufweist. Ein alternativer Ansatz wäre dagegen, die latente Größe Z selbst über einen gewöhnlichen Regressionsansatz zu modellieren und dann Schwellenwerte festzulegen, nach denen eine Eingruppierung in die ordinalen Klassen erfolgen kann. Diese Ansätze werden anhand eines empirischen Datensatzes aus der medizinischen Forschung untersucht.

Rudolfer, Watson & Lesaffre (1995) untersuchen in ihrer Arbeit eine ganze Reihe von Modellen für ordinale Daten. Zunächst betrachten sie die herkömmliche Diskriminanzanalyse mit der Diskriminanzfunktion

$$d_r(x) = \log\frac{\mathrm{P}(Y = r)}{\mathrm{P}(Y = 1)} - \frac{1}{2}\left(\mu_r' \Sigma^{-1} \mu_r - \mu_1' \Sigma^{-1} \mu_1\right) + \Sigma^{-1}(\mu_r - \mu_1)x \quad ,$$

sowie das multinomiale logistische Regressionsmodell

$$\mathrm{P}(Y = r|x) = \frac{\exp(\beta_{0r} + \beta_r' x)}{1 + \sum_{l=2}^{k} \exp(\beta_{0l} + \beta_l' x)} \quad ,$$

bei denen die Ordinalität jeweils nicht berücksichtigt wird. Als *assessed model* (beurteiltes Modell), das stark subjektiv geprägt ist, bezeichnen die Autoren ein ordinales logistisches Regressionsmodell

$$\log\frac{\mathrm{P}(Y = r|x)}{\mathrm{P}(Y = 1|x)} = \beta_{0r} + \phi_r \beta' x \quad ,$$

das dem obigen multinomialen entspricht. Zusätzlich werden die Konstanten ϕ_j mit $0 = \phi_1 \leq \phi_2 \leq \cdots \leq \phi_{k-1} \leq \phi_k = 1$ zur Trennung zwischen den ordinalen Klassen herangezogen. Zuguterletzt werden noch zwei Ansätze aus der *cumulative-odds*-Modellklasse herangezogen: Bei *proportional odds* wird die bedingte Wahrscheinlichkeit $\mathrm{P}(Y \leq s|x)$ über

$$\mathrm{P}(Y \leq s|x) = F(\beta_{0s} - \beta' x)$$

modelliert, zum Beispiel anhand eines logistischen Modells

$$\mathrm{P}(Y \leq s|x) = \frac{\exp(\beta_{0s} - \beta' x)}{1 + \exp(\beta_{0s} - \beta' x)}$$

oder eines Probitmodells

$$\mathrm{P}(Y \leq s|x) = \Phi(\beta_{0s} - \beta' x) \quad .$$

Dagegen modelliert man beim sog. *continuation ratio* das Verhältnis von $P(Y = s)$ gegenüber $P(Y > s)$ über

$$\log \frac{\mathrm{P}(Y = s|x)}{1 - \mathrm{P}(Y \leq s|x)} = \beta_{0s} - \beta' x$$

und damit

$$\mathrm{P}(Y = s|x) = \frac{\exp(\beta_{0s} - \beta' x)}{\prod_{r=1}^{s}(1 + \exp(\beta_{0r} - \beta' x))} \quad .$$

In beiden Fällen wird demnach eine Verbindung von einer latenten zu einer metrischen Variable hergestellt. Insgesamt kann man festhalten, daß alle genannten Modelle bis auf die Diskriminanzanalyse bei lediglich zwei Klassen der Zielgröße übereinstimmen. Bei allen Verfahren erfolgt die letztendliche Klassifikation jeweils über die maximale a posteriori-Wahrscheinlichkeit. Anhand von verschiedenen simulierten Datensätzen werden die genannten Verfahren bezüglich dreier Fehlerraten verglichen. Neben der rohen Fehlklassifikationsrate werden weiter die Abstände zwischen wahrer und prognostizierter Klasse sowie asymmetrische Abstände, bei denen ein Fehler in eine Richtung schwerwiegender ist als in die andere Richtung, betrachtet. Als Ergebnis kann man festhalten, daß ordinale Modelle bezüglich aller Gütemaße deutlich besser sind als die nominalen Modelle, wenn deren spezielle Struktur auch zur Simulation des Datensatzes verwendet wurde. Diese Einschränkung ist wichtig, da reale Datensätze nur selten exakte Modellannahmen erfüllen. Am robustesten gegen die Verletzung der jeweiligen Annahme scheint das ordinale logistische Regressionsmodell zu sein. Unter den beiden nominalen Modellen schneidet der multinomiale logistische Ansatz bei dichotomen Kovariaten besser ab, während die Diskriminanzanalyse bei normalverteilten Einflußgrößen dominiert. Insgesamt können die Autoren feststellen, daß ordinale Modelle definitiv nützlich sind, der Grad der Verbesserung gegenüber nominalen Ansätzen jedoch stark von der Wahl eines geeigneten Modells abhängt.

Was die Gütemaße zur Beurteilung von ordinalen Modellen betrifft, sehen Kramer, Widmer, Pfahringer & De Groeve (2001) einen Konflikt (oder *trade-off*) zwischen der Fehlklassifikationsrate einerseits und den Abstandsmaßen andererseits. Ferner bieten sie einige Verfahren zur ordinalen Klassifikation an, die auf Regressionsbäumen basieren. Einerseits kann in Form einer *post-processing*-Methode durch nachträgliches Runden der Prognose in jedem Endknoten die Transformation auf die ordinale Größe erfolgen. Hier darf also mit gewöhnlichen unmodifizierten Regressionsbäumen gerechnet werden, lediglich die Prognose wird am Ende auf die jeweils nächste Klasse gerundet. Andererseits kann bereits während der Entwicklung des Baumes in jedem Knoten der Median, ein auf die nächste Klasse gerundetes Mittel oder der Modus bestimmt und darauf basierend die Verzweigungen gewählt

werden. Als Gütemaße werden die Fehlklassifikationsrate, die Wurzel des mittleren quadratischen Fehlers, sowie der Korrelationskoeffizient von Spearman betrachtet. Anhand einer Studie mit fünf Datensätzen kommen die Autoren zu dem Ergebnis, daß alle genannten Verfahren in der Lage sind, die in den Daten vorhandene ordinale Struktur erfolgreich zu nutzen. Es läßt sich also festhalten, daß meist eine Verbesserung gegenüber nichtordinalen Ansätzen erzielt werden kann. Eine eindeutig dominierende Technik läßt sich jedoch nicht extrahieren.

Einen neuen Ansatz, der weder darauf basiert, durch Verwendung eines gewöhnlichen Klassifikationsverfahrens die ordinale Struktur einfach zu ignorieren, noch auf das Konzept von Regression mit anschließender Rücktransformation in eine ordinale Prognose setzt, wird von Frank & Hall (2001) vorgestellt. Die Grundidee ist dabei wieder eine kumulative Zerlegung der ordinalen Größe in $k-1$ binäre Variablen. Bestimmt man nun Prognosewahrscheinlichkeiten anhand von $k-1$ getrennten Modellen für diese binären Größen, so erhält man aggregierte Wahrscheinlichkeiten für die beiden Randklassen über

$$\begin{aligned} \mathrm{P}(Y=1) &= 1-\mathrm{P}(Y>1) \\ \mathrm{P}(Y=k) &= \mathrm{P}(Y>k-1) \quad , \end{aligned}$$

also jeweils nur aus einzelnen binären Modellen. Wahrscheinlichkeiten für die mittleren Klassen resultieren dagegen aus Differenzen der Klassifikationswahrscheinlichkeiten von je zwei Modellen

$$\mathrm{P}(Y=r) = \mathrm{P}(Y>r-1) - \mathrm{P}(Y>r) \quad .$$

Insgesamt ergibt sich dadurch nach Ansicht der Autoren eine *echte* Wahrscheinlichkeitsverteilung, die zur Prognose genutzt werden kann, indem die Klasse mit der höchsten Wahrscheinlichkeit vorhergesagt wird. Um die Güteeigenschaften dieses Ansatzes zu überprüfen, führen die Autoren eine umfangreiche Vergleichsstudie durch. Aufgrund ihrer Feststellung, daß kaum Standarddatensätze mit ordinalen Zielgrößen existieren (ein Phänomen, das auch bei der Bearbeitung dieses Dissertationsthemas auffiel), werden metrische Zielgrößen aus Regressionsdatensätzen durch Gruppierung diskretisiert. Als erwünschtes Nebenprodukt kann so erreicht werden, daß die Klassen gleich groß sind. Insgesamt 29 Datensätze konnten auf diese Weise untersucht werden. Als Gütemaß wird lediglich die Fehlklassifikationsrate betrachtet. Um sicherzustellen, ob ein Effekt dieses ordinalen Ansatzes nicht einfach auf die Dichotomisierung der Zielgröße, sondern tatsächlich auf die Berücksichtigung der Ordinalität zurückzuführen ist, wird zum Vergleich nicht nur der gewöhnliche nominale Klassifikationsansatz, sondern auch ein rein nominaler Mehrklassenansatz, der ebenfalls auf einer Dichotomisierung der Daten beruht, herangezogen. Die Autoren stellen dabei fest, daß ihr ordinaler Ansatz im Vergleich zu beiden anderen meist zu deutlichen Verbesserungen führt, wobei die Verbesserung umso deutlicher ausfällt, je mehr ordinale Klassen vorliegen, das heißt je deutlicher ausgeprägt die ordinale Struktur in den Daten ist.

Kapitel 4

Aggregierte Klassifikationsverfahren bei ordinalen Zielgrößen

In diesem Kapitel der Arbeit geht man nun von einem Mehrklassenproblem mit $Y \in \{1, \ldots, k\}$ aus, bei dem die Klassen als geordnet angenommen werden. Basierend auf beiden ordinalen Ansätzen (sequenziell und kumulativ) kann man nun versuchen, die Ergebnisse, die mit Hilfe von Bagging oder Boosting erzielt werden, noch zu verbessern. Jeder einen der Übergänge modellierende Baum kann dann durch das Abstimmungsergebnis mehrerer Bäume ersetzt werden, die auf Zufallsziehungen aus der jeweiligen Lernstichprobe des Übergangs basieren. So kommt es zu *Abstimmungen zwischen Abstimmungsergebnissen*, denn mehrere Klassifikationsmodelle liefern Stimmen, die dann für jede einzelne Klasse über alle diese Modelle hinweg aufaddiert werden.

Wir betrachten im Folgenden eine Reihe von Ansätzen, um die ordinale Struktur der Antwortkategorie zu nutzen. All diese Techniken lassen sich jedoch in zwei Kategorien einteilen: Der erste Ansatz, hier *Fixed Split Boosting* genannt, führt die Problemstellung auf binäre Klassifikationen zurück, indem die Antwortkategorien bezüglich jedes möglichen Trennpunktes zwischen benachbarten Klassen dichotomisiert werden. Nach dem Einsatz von Boosting-Techniken für die einzelnen binären Problemstellungen getrennt, werden die resultierenden Prognosen zu einem aggregierten Klassifizierer kombiniert. Natürlich können statt Boosting auch Bagging oder beliebige andere, nicht aggregierte Klassifikationsverfahren herangezogen werden, die zum Vergleich dann auch im empirischen Teil dieser Arbeit ausprobiert wurden. Der zweite Ansatz versucht dagegen, die ordinale Struktur explizit innerhalb jedes einzelnen Boosting-Zykluses auszunutzen, indem Kriterien zur Aktualisierung der Gewichte verwendet werden, die in der Lage sind, Ordinalität zu berücksichtigen.

4.1 Fixed Split Boosting

Im Folgenden wird die Klassifikationsprozedur in zwei Stufen aufgeteilt. In der ersten Stufe werden Aggregationstechniken verwendet, nachdem die Zielgröße dichotomisiert worden ist. In der zweiten Stufe werden die resultierenden binären Klassifikatoren dann zu einem Gesamtergebnis kombiniert. Da hier Boosting jeweils für festgelegte binäre Spaltungen (*fixed splits*) der Zielgröße getrennt durchgeführt wird, wird der Ansatz als *Fixed Split Boosting* bezeichnet.

Selbstverständlich können auch gewöhnliche, nicht aggregierte Techniken oder Bagging auf solche festen Aufteilungen angewendet werden. In diesem Fall spricht man dann beispielsweise von *Fixed Split CART* oder *Fixed Split Bagging*.

Kumulative Variante: Die Rückführung auf binäre Probleme wird also durchgeführt, indem die Zielgröße dichotomisiert wird. Dies geschieht beim kumulativen Ansatz über

$$Y^{(r)} = \begin{cases} 1 & \text{für } Y \in \{1, \ldots, r\} \\ 2 & \text{für } Y \in \{r+1, \ldots, k\} \end{cases}$$

für $r = 1, \ldots, k-1$.

Sei nun $C^{(r)}(., L)$ ein Klassifikator für das durch $Y^{(r)}$ gegebene binäre Klassifikationsproblem, also für die Aufteilung $\{1, \ldots, r\}$ gegen $\{r+1, \ldots, k\}$. Für ein fest gewähltes r erhält man durch die Anwendung eines beliebigen aggregierten Klassifikationsverfahrens die vorhergesagte Klasse einer Beobachtung x durch

$$C^{(r)}_{agg}(x) = \text{argmax}_j \left(\sum_{m=1}^{M} c^{(r)}_m I(C^{(r)}(x, L^{(r)}_m) = j) \right) \quad .$$

Dabei ist $L^{(r)}_m$ die m-te Lernstichprobe und $c^{(r)}_m$ der Gewichtungsfaktor für den m-ten Klassifikator innerhalb des Aggregationsverfahrens, jeweils für die Aufteilung der Zielgröße Y in r.

Dieser aggregierte Klassifikator $C^{(r)}_{agg}(.)$ der ersten Stufe wurde für eine festgelegte Aufteilung an der Stelle r konzipiert. Die Kombination der aggregierten Klassifikatoren

$$C^{(1)}_{agg}(.), \ldots, C^{(k-1)}_{agg}(.)$$

basiert dann auf einer zweiten Stufe der Aggregation, wobei nun die Ordnung innerhalb der Kategorien ausgenutzt wird. Dazu verwendet man, daß die Zielvariable $Y \in \{1, \ldots, k\}$ ebenso durch einen binären Response-Vektor $(Y_1, \ldots, Y_k)$ repräsentiert werden kann. Dabei gilt:

$$Y_j = \begin{cases} 1 & \text{für } Y = j \\ 0 & \text{sonst} \end{cases}$$

Der Klassifikator $C_{agg}^{(r)}(.)$ stellt eine Vorhersage des binären Klassenindikators $Y^{(r)}$ dar. Sei dazu das Ergebnis dieser Klassifizierer in die Sequenz $\hat{Y}_1^{(r)}, \ldots, \hat{Y}_k^{(r)}$ von binären Variablen transformiert.

Im Fall $C_{agg}^{(r)}(x) = 1$, der $\hat{Y}(x) \in \{1, \ldots, r\}$ entspricht, setzt man:

$$\hat{Y}_1^{(r)}(x) = \cdots = \hat{Y}_r^{(r)}(x) = \frac{1}{r}, \quad \hat{Y}_{r+1}^{(r)}(x) = \cdots = \hat{Y}_k^{(r)}(x) = 0$$

Im Fall $C_{agg}^{(r)}(x) = 2$, der $\hat{Y}(x) \in \{r+1, \ldots, k\}$ entspricht, setzt man:

$$\hat{Y}_1^{(r)}(x) = \cdots = \hat{Y}_r^{(r)}(x) = 0, \quad \hat{Y}_{r+1}^{(r)}(x) = \cdots = \hat{Y}_k^{(r)}(x) = \frac{1}{k-r}$$

Auf diese Weise liefert der Klassifizierer $C_{agg}^{(r)}(.)$ eine der binären Sequenzen

$$\frac{1}{r} \cdot (1, 1, \ldots, 1, 0, 0, \ldots, 0)$$

oder

$$\frac{1}{k-r} \cdot (0, 0, \ldots, 0, 1, 1, \ldots, 1) \quad ,$$

wobei der Wechsel von 1 nach 0 bzw. von 0 nach 1 nach der r-ten Komponente auftritt. Diese Sequenzen werden hier durch r bzw. $k-r$ dividiert, um der Tatsache Rechnung zu tragen, daß sich jeweils eine unterschiedliche Anzahl von ursprünglichen Kategorien in den Dichotomisierungen wiederfindet. Die Gruppen von Klassen müssen sich ihren Score also gleichmäßig untereinander aufteilen.

Der endgültige Klassifikator ist dann gegeben durch die Aggregation der zweiten Stufe:

$$C_{agg}(x) = \text{argmax}_j \left(\sum_{r=1}^{k-1} \hat{Y}_j^{(r)}(x) \right)$$

Auf diese Weise wird der Beobachtung x diejenige Klasse als Vorhersage zugeordnet, die von einer (gewichteten) Mehrheit der Aufteilungen favorisiert wird. Der Algorithmus zum kumulativen Fixed Split Boosting funktioniert also wie in Abbildung 4.1 zusammengefaßt.

Als Ergänzung kann desweiteren für jede Aufteilung r der Score

$$s^{(r)} = \sum_{r=1}^{k-1} \hat{Y}_j^{(r)}(x)$$

berechnet werden, der als ein Maß für die Genauigkeit der Vorhersage (nach Aggregierung über Bagging oder Boosting) herangezogen werden kann. Standardisiert erhält er die Form

$$\tilde{s}^{(r)} = \frac{s^{(r)}}{\sum_{t=1}^{k-1} s^{(t)}} \quad .$$

kumulatives Fixed Split Boosting

1. Definiere $Y^{(r)} = \begin{cases} 1 & \text{für } Y \in \{1, \ldots, r\} \\ 2 & \text{für } Y \in \{r+1, \ldots, k\} \end{cases}$ für $r = 1, \ldots, k-1$.
2. Sei $C^{(r)}(., L)$ der Klassifikator für das auf $Y^{(r)}$ basierende binäre Problem. Dann verwende irgendeine Ensemble-Technik, um

$$C^{(r)}_{agg}(x) = \text{argmax}_j \left(\sum_{m=1}^{M} c^{(r)}_m I(C^{(r)}(x, L^{(r)}_m) = j) \right)$$

   zu erhalten.
3. Transformiere $C^{(r)}_{agg}(x)$ in eine binäre Sequenz $\hat{Y}^{(r)}_1, \ldots, \hat{Y}^{(r)}_k$ gemäß

$$\hat{Y}^{(r)}_1(x) = \cdots = \hat{Y}^{(r)}_r(x) = \frac{1}{r}, \quad \hat{Y}^{(r)}_{r+1}(x) = \cdots = \hat{Y}^{(r)}_k(x) = 0 \quad \text{für} \quad C^{(r)}_{agg}(x) = 1$$
$$\hat{Y}^{(r)}_1(x) = \cdots = \hat{Y}^{(r)}_r(x) = 0, \quad \hat{Y}^{(r)}_{r+1}(x) = \cdots = \hat{Y}^{(r)}_k(x) = \frac{1}{k-r} \quad \text{für} \quad C^{(r)}_{agg}(x) = 2$$

4. Der aggregierte Klassifikator ist dann gegeben durch

$$C_{agg}(x) = \text{argmax}_j \left(\sum_{r=1}^{k-1} \hat{Y}^{(r)}_j \right) \quad .$$

Abbildung 4.1: Algorithmus für kumulatives Fixed Split Boosting

Schließlich sollte noch angemerkt werden, daß in der zweiten Stufe der Aggregation die Präzision der einzelnen binären Vorhersagen noch nicht benutzt wurde. Deshalb erscheint es sinnvoll, auch eine gewichtete Version dieses Algorithmus zu betrachten, die dann gegeben ist durch

$$C_{agg,w}(x) = \text{argmax}_j \left(\sum_{r=1}^{k-1} PV^{(r)}(x) \hat{Y}^{(r)}_j(x) \right) \quad ,$$

wobei die Gewichte

$$PV^{(r)}(x) = \frac{\max_j \sum_{m=1}^{M} c^{(r)}_m I(C^{(r)}(x, L^{(r)}_m) = j)}{\sum_{m=1}^{M} c^{(r)}_m}$$

aus den binären *prediction votes* in der ersten Aggregationsstufe gebildet werden. Die zugehörigen Scores, die die Genauigkeit in der ersten Stufe berücksichtigen, lauten

$$s^{(r)}_w = \sum_{r=1}^{k-1} PV^{(r)}(x) \hat{Y}^{(r)}_j(x) \quad .$$

Auch sie können wiederum normiert werden zu

$$\tilde{s}_w^{(r)} = \frac{s_w^{(r)}}{\sum_{t=1}^{k-1} s_w^{(t)}} \quad .$$

Empirische Versuche mit einem abgewandelten Algorithmus, der diese Präzisionsmaße als zusätzliche Gewichte verwendet, wurden durchgeführt, lieferten aber tendenziell eher schlechtere Ergebnisse als die oben beschriebene Version von Fixed Split Boosting. Deshalb wird auf diese Variante nicht weiter eingegangen.

Alternativ könnten die genannten Gewichte natürlich auch verwendet werden, um die substanzielle Bedeutung von einzelnen Aufteilungen statt der Genauigkeit der Vorhersage zu beurteilen.

Sequenzielle Variante: Auch sequenziell kann die ordinale Aufgabenstellung auf binäre Probleme zurückgeführt werden. Die Dichotomisierung der Zielgröße erfolgt in diesem Fall über

$$Y^{(r)} = \begin{cases} 1 & \text{für } Y = r \\ 2 & \text{für } Y \in \{r+1, \ldots, k\} \end{cases}$$

für $r = 1, \ldots, k-1$. Liegt also eine Klasse kleiner als der jeweilige Trennpunkt r vor, so erhält die entsprechende binäre Variable einen fehlenden Wert.

Sei wiederum $C^{(r)}(., L)$ ein Klassifikator für das durch $Y^{(r)}$ gegebene binäre Klassifikationsproblem, in diesem Fall also für die Aufteilung $\{r\}$ gegen $\{r+1, \ldots, k\}$. Dann erhält man für ein fest gewähltes r durch die Anwendung eines beliebigen aggregierten Klassifikationsverfahrens die vorhergesagte Klasse einer Beobachtung x durch

$$C_{agg}^{(r)}(x) = \text{argmax}_j \left(\sum_{m=1}^{M} c_m^{(r)} I(C^{(r)}(x, L_m^{(r)}) = j) \right) \quad .$$

Zur Modellbildung können selbstverständlich aber nur diejenigen Beobachtungen herangezogen werden, die bezüglich dieser Zielgröße keinen fehlenden Wert aufweisen. Deshalb werden bei der Ziehung von $L_m^{(r)}$ lediglich solche Beobachtungen berücksichtigt, für die $Y \geq r$ gilt.

Diese für spezielle Aufteilungen in r konzipierten Klassifikatoren $C_{agg}^{(r)}(.)$ der ersten Stufe müssen nun, ähnlich wie im kumulativen Ansatz, in einer zweiten Stufe wieder aggregiert werden, wobei nun die Ordnung der Kategorien mit eingeht. Hier soll dies aber nicht über das kumulative Aufaddieren von Scores, sondern stattdessen über sequenzielles Durchlaufen der Einzelklassifikatoren $C_{agg}^{(r)}(.)$ geschehen:

Im Fall $C_{agg}^{(1)}(x) = 1$ erfolgt die Klassifikation unmittelbar in die Klasse 1. Ansonsten (also im Fall $C_{agg}^{(1)}(x) = 2$) fährt man mit dem Klassifizierer für die nächste Dichotomisierung $C_{agg}^{(2)}(x)$ fort. Lautet die Prognose nun 1, so wird Klasse 2 prognostiziert, ansonsten fährt

man mit $C^{(3)}_{agg}(x)$ fort. Die Klassifikation erfolgt also demnach in die niedrigste Klasse r, für die $C^{(r)}_{agg}(x) = 1$ gilt. Gilt dagegen bis einschließlich der letzten Aufteilung in $k-1$ stets $C^{(r)}_{agg}(x) = 2$, so erfolgt die Klassifikation in die verbleibende Klasse k.

Der endgültige Klassifikator der zweiten Stufe ist also gegeben durch:

$$C_{agg}(x) = \min\left(\text{argmin}_r\left(C^{(r)}_{agg}(x) = 1\right), k\right) \quad .$$

Auf diese Weise wird der Beobachtung x die niedrigste Klasse als Vorhersage zugeordnet, die der Alternative einer beliebigen höheren Klasse vorgezogen wird. Sequenzielles Fixed Split Boosting läßt sich wie in Abbildung 4.2 als Algorithmus zusammenfassen.

---

sequenzielles Fixed Split Boosting

1. Definiere $Y^{(r)} = \begin{cases} 1 & \text{für } Y = r \\ 2 & \text{für } Y \in \{r+1, \ldots, k\} \end{cases}$ für $r = 1, \ldots, k-1$.
2. Sei $C^{(r)}(., L)$ der Klassifikator für das auf $Y^{(r)}$ basierende binäre Problem. Dann verwende irgendeine Ensemble-Technik, um

$$C^{(r)}_{agg}(x) = \text{argmax}_j \left(\sum_{m=1}^{M} c^{(r)}_m I(C^{(r)}(x, L^{(r)}_m) = j) \right)$$

   zu erhalten.
3. Durchlaufe die Klassifikatoren $C^{(r)}_{agg}(x)$ sequenziell für $r = 1, \ldots, k-1$ und fasse die Ergebnisse zusammen, indem diejenige Klasse r als Prognose gebildet wird, für die zum ersten Mal der Fall $C^{(r)}_{agg}(x) = 1$ auftritt.
4. Der aggregierte Klassifikator ist also gegeben durch

$$C_{agg}(x) = \min\left(\text{argmin}_r\left(C^{(r)}_{agg}(x) = 1\right), k\right) \quad .$$

Abbildung 4.2: Algorithmus für sequenzielles Fixed Split Boosting

Abschließend kann zu Fixed Split Boosting im Allgemeinen angemerkt werden, daß es auf den ersten Blick den Nachteil aufweist, daß die ordinale Struktur lediglich im finalen Abstimmungsschritt benutzt wird. Dies könnte zu dem naheliegenden Schluß führen, daß andere Verfahren, die die Ordinalität bereits in früheren Schritten nutzen, unweigerlich zu besseren Resultaten führen. Dabei darf aber nicht ein entscheidender Vorteil dieses Verfahrens übersehen werden, den die im folgenden Abschnitt vorgestellten ordinalen AdaBoost-Versionen nicht aufweisen: Jede dichotome Aufteilung der Zielgröße wird individuell durch Boosting getrennt, so daß die Prognosen für die einzelnen Splits explizit optimiert werden. Bei den folgenden Verfahren wird dagegen pro Boosting-Schritt mit identischen Gewichten für jede der Aufteilungen gearbeitet, die dann sicher nicht für alle Splits gleichzeitig optimal sein können.

4.2 Ordinaler AdaBoost

Die Fixed-Split-Boosting-Prozedur, die im letzten Abschnitt vorgeschlagen wurde, weist deutliche Ähnlichkeiten mit sogenannten *one-against-all*-Ansätzen auf, die häufig im Forschungsfeld des maschinellen Lernens zum Einsatz kommen. Beide Ansätze haben gemeinsam, daß jede Klasse der Zielgröße einzeln mit allen anderen Klassen verglichen wird und die über Bagging oder Boosting erhaltenen Ergebnisse dieser binären Problemstellungen am Ende kombiniert werden (vgl. beispielsweise Dettling & Bühlmann (2003)).

Fixed Split Boosting nutzt die ordinale Struktur der Zielgröße also nicht im Neugewichtungsschritt. Nur der finale Kombinationsschritt der binären Einzelresultate greift auf die Ordinalität zurück.

Diese getrennte Betrachtung der beiden Aggregationsstufen bedeutet, daß bei geordneten Antwortkategorien die Reihenfolge in der ersten Stufe, also beim Bagging oder Boosting selbst, nicht berücksichtigt werden kann. Deshalb werden im Folgenden alternative Algorithmen vorgeschlagen, die bereits eine Verknüpfung zwischen den Gewichten beim Boosting und der ordinalen Struktur des Klassifikators herstellen.

An dieser Stelle muß angemerkt werden, daß bei dieser Vorgehensweise der sequenzielle Ansatz nur in Verbindung mit dem Discrete AdaBoost, nicht jedoch mit Real oder Gentle AdaBoost, eingesetzt werden kann. Grund hierfür ist, daß ein sequenzieller Durchlauf zur Klassifikation eines Objekts definitiv auf strikten 0-1-Entscheidungen basieren muß: Entweder die Einheit verbleibt in einer niedrigeren Klasse und der Durchlauf endet, oder sie wird in eine höhere Klasse prognostiziert und durchläuft die weiteren Modelle. Hier können die feiner abgestimmten Klassifikationsscores von Real und Gentle AdaBoost nicht genutzt werden, lediglich die strikten 0-1-Prognosen von Discrete AdaBoost sind dazu kompatibel.

4.2.1 Ordinal Discrete AdaBoost

Kumulative Variante: Die erste Variante basiert von der Grundidee her auf dem gewöhnlichen Discrete-AdaBoost-Algorithmus. Allerdings werden die verwendeten Kriterien so angepaßt, daß sie sensitiv gegenüber der ordinalen Struktur in der Zielgröße sind.

Der grundlegende Unterschied zu Fixed Split Boosting liegt in Schitt 2 des in Abbildung 4.3 beschriebenen Algorithmus. Die Aggregation über die verschiedenen Aufteilungen ist nun Bestandteil der Boosting-Prozedur. Außerdem reflektieren die Fehlerindikatoren ϵ_i nun den Abstand zwischen der Vorhersage und der wahren Klasse und sind damit in der Lage, die ordinale Struktur in der Zielgröße beim Adjustieren der Gewichte zu berücksichtigen. Alternativen zu diesem standardisierten Abstandskriterium

$$\epsilon_i = \frac{|C(x_i, L_m) - Y_i|}{k-1}$$

kumulatives Ordinal Discrete AdaBoost

1. Starte mit Gewichten $w_1 = \cdots = w_{n_L} = 1/n_L$ für die Beobachtungen der Lernstichprobe L.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Die aktuellen Gewichte $w_1, \ldots, w_{n_L}$ werden für den Resampling-Vorgang verwendet. Basierend auf diesen Wahrscheinlichkeiten wird eine Lernstichprobe L_m vom Umfang n_L aus L mit Zurücklegen gezogen.
   (b) Anhand von L_m werden Klassifizierer $C^{(r)}(., L_m)$ für alle bezüglich r dichotomisierten Aufteilungen der ordinalen Klassenvariable entwickelt.
   (c) Die Lernstichprobe durchläuft jeden dieser Klassifizierer $C^{(r)}(., L_m)$, woraus man die Information erhält, ob die i-te Beobachtung in eine Klasse kleiner oder größer als r prognostiziert wird. Die Ergebnisse der Klassifizierer für verschiedene Aufteilungen in r werden über einen Mehrheitsentscheid kombiniert, woraus man den aggregierten Klassifizierer $C(., L_m)$ erhält.
   (d) Seien die Fehlerindikatoren nun gegeben als $\epsilon_i = \frac{|C(x_i, L_m) - Y_i|}{k-1}$. Darauf basierend erfolgt die Aktualisierung der Gewichte mit der selben Notation für e_m, b_m und c_m wie beim gewöhnlichen dichotomen Discrete AdaBoost über

$$w_{i,new} = \frac{w_i \exp(c_m \epsilon_i)}{\sum_{j=1}^{n} w_j \exp(c_m \epsilon_j)} \quad .$$

3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über

$$\text{argmax}_j \left(\sum_{m=1}^{M} c_m I(C(x, L_m) = j) \right) \quad .$$

Abbildung 4.3: Algorithmus für kumulatives Ordinal Discrete AdaBoost

wären beispielsweise die unstandardisierte Distanz

$$\epsilon_i = |C(x_i, L_m) - Y_i| \quad ,$$

die quadrierte unstandardisierte Distanz

$$\epsilon_i = (C(x_i, L_m) - Y_i)^2$$

oder die quadrierte standardisierte Distanz

$$\epsilon_i = \left(\frac{C(x_i, L_m) - Y_i}{k-1} \right)^2 .$$

Selbstverständlich könnte auch weiterhin der einfache 0-1-Fehler

$$\epsilon_i = I(C(x_i, L_m) \neq Y_i) \quad ,$$

also die rohe Fehlklassifikationsrate, verwendet werden. Diese Größe ist dann allerdings nicht in der Lage, die ordinale Struktur zu erkennen und zu berücksichtigen.

Wie bereits bei der Entwicklung einer Mehrklassenvariante für den gewöhnlichen Discrete AdaBoost beschrieben, sind die Gewichte

$$c_m = \frac{1 - e_m}{e_m}$$

grundsätzlich für Zweiklassenprobleme konstruiert. Ebenso wie eine Anpassung für diesen nominalen Mehrklassenfall vorgenommen wurde, kann man sich nun an einer Adjustierung von c_m bezüglich des hier verwendeten ordinalen Abstandsmaßes versuchen.

Man stellt jedoch leicht fest, daß bei diesem Kriterium keine Anpassung für den Mehrklassenfall durchgeführt werden muß. Für den einfachen Betragsabstand mit Wertebereich $[0, k-1]$ berechnet man für den Fall, daß alle Beobachtungen in eine der k Gruppen mit gleich großen Stichprobenumfängen klassifiziert werden, einen Fehlerwert von $(k-1)/2$ (unter der Annahme von gleichen Gewichten $w_i = 1/n_L$). So erhält eine geeignete Anpassung von c_m die altbekannte Form

$$c_m = \log\left(\frac{(1 - e_m)/((k-1)/2)}{e_m/((k-1)/2)}\right) = \log\left(\frac{1 - e_m}{e_m}\right) \quad .$$

Derselbe Wert resultiert auch für die standardisierte Abstandsvariante, wie sie in der Beschreibung des Algorithmus angeführt ist.

Da der Term c_m ebenso wie beim gewöhnlichen Discrete AdaBoost gegen unendlich streben kann, wird dieselbe Konsistenzbedingung angewendet. Dadurch ist gesichert, daß c_m stets wohldefiniert ist.

Sequenzielle Variante: Die im Folgenden beschriebene sequenzielle Variante des Ordinal Discrete AdaBoost unterscheidet sich lediglich in einem Schritt von dem oben angeführten kumulativen Algorithmus. In jedem Boosting-Zyklus werden die Prognosen für die einzelnen Splits nämlich sequenziell aggregiert. Das weitere Vorgehen anhand dieser aggregierten Prognosen ist ansonsten exakt identisch (siehe Abbildung 4.4).

Desweiteren gelten die gleichen Aussagen zu möglichen alternativen Abstandsmaßen sowie zu Konsistenzbedingungen wie beim kumulativen Ordinal Discrete AdaBoost.

Wie bereits oben erwähnt ist dies die einzige plausible sequenzielle AdaBoost-Erweiterung, da beim sequenziellen Durchlaufen der Splits unmittelbar strikte 0-1-Entscheidungen benötigt werden, ob in die niedrigere oder eine der höheren Klassen prognostiziert werden soll. Ein verfeinertes Kriterium auf Basis von Klassifikationswahrscheinlichkeiten müßte hier ebenfalls unmittelbar in eine 0-1-Entscheidung umgewandelt werden, so daß die Unterschiede zum Discrete-Ansatz nicht genutzt werden könnten. Desweiteren stünden nach der sequenziellen Aggregation auch keine Klassifikationswahrscheinlichkeiten mehr zur Verfügung, die für den Aktualisierungsschritt aber benötigt würden.

sequenzielles Ordinal Discrete AdaBoost

1. Starte mit Gewichten $w_1 = \cdots = w_{n_L} = 1/n_L$ für die Beobachtungen der Lernstichprobe L.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Die aktuellen Gewichte $w_1, \ldots, w_{n_L}$ werden für den Resampling-Vorgang verwendet. Basierend auf diesen Wahrscheinlichkeiten wird eine Lernstichprobe L_m vom Umfang n_L aus L mit Zurücklegen gezogen.
   (b) Anhand von L_m werden Klassifizierer $C^{(r)}(., L_m)$ für alle bezüglich r dichotomisierten Aufteilungen der ordinalen Klassenvariable entwickelt.
   (c) Die Lernstichprobe durchläuft diese Klassifizierer $C^{(r)}(., L_m)$ sequenziell für $r = 1, \ldots, k-1$, und die Ergebnisse werden so zusammengefaßt, daß diejenige Klasse r als Prognose gebildet wird, für die zum ersten Mal der Fall $C^{(r)}(x_i, L_m) = 1$ auftritt. Diese Prognose wird dann als $C(x_i, L_m)$ bezeichnet.
   (d) Seien die Fehlerindikatoren nun gegeben als $\epsilon_i = \frac{|C(x_i, L_m) - Y_i|}{k-1}$. Darauf basierend erfolgt die Aktualisierung der Gewichte mit der selben Notation für e_m, b_m und c_m wie beim gewöhnlichen dichotomen Discrete AdaBoost über

$$w_{i,new} = \frac{w_i \exp(c_m \epsilon_i)}{\sum_{j=1}^{n} w_j \exp(c_m \epsilon_j)} \quad .$$

3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über

$$\text{argmax}_j \left(\sum_{m=1}^{M} c_m I(C(x, L_m) = j) \right) \quad .$$

Abbildung 4.4: Algorithmus für sequenzielles Ordinal Discrete AdaBoost

4.2.2 Ordinal Real AdaBoost

Auf ähnliche Art und Weise wie bei Discrete AdaBoost erfolgt die Konstruktion einer ordinalen Variante von Real AdaBoost. Sie basiert wiederum auf dem dichotomen Problem der Trennung zwischen Klassen $\{1, \ldots, r\}$ und $\{r+1, \ldots, k\}$. Für jede solche Dichotomisierung stellt ein dichotomer Klassifikator Wahrscheinlichkeiten

$$p^{(r)}(x) = \hat{P}(Y \leq r | x)$$

zur Verfügung. Aus diesen Wahrscheinlichkeiten erhält man eine Folge von Scores $\hat{Y}^{(r)}(x)$ für jede Klasse über

$$\hat{Y}_1^{(r)}(x) = \cdots = \hat{Y}_r^{(r)}(x) = p^{(r)}(x) \quad \text{und} \quad \hat{Y}_{r+1}^{(r)}(x) = \cdots = \hat{Y}_k^{(r)}(x) = 1 - p^{(r)}(x) \quad .$$

Zur Aggregation über die verschiedenen Aufteilungen hinweg betrachtet man dann den Wert

$$\hat{Y}_j(x) = \sum_{r=1}^{k-1} \hat{Y}_j^{(r)}(x) \quad ,$$

der die Sicherheit der Vorhersage in Klasse j widerspiegelt. Eine Version für Ordinal Real AdaBoost, die auf diesem Term basiert, aber immer noch eine enge Beziehung zum gewöhnlichen Real AdaBoost aufweist, ist in Abbildung 4.5 angegeben.

Das Vorzeichen von $f_j(x_i, L_m)$ hängt von dem Verhältnis des Scores für Klasse j verglichen mit einem gewichteten Durchschnittsscore aller anderen Klassen ab. Diese Gewichtung der Klassenwahrscheinlichkeiten basiert auf dem Abstand zwischen betrachteter und wahrer Klasse. So werden hohe Wahrscheinlichkeiten für Klassen, die eine große Distanz zur wahren Klasse aufweisen, stärker bestraft als hohe Wahrscheinlichkeiten für Nachbarklassen.

Wiederum kann die Konsistenzbedingung für die Existenz von $f_j(x_i, L_m)$ unmittelbar von der nominalen Real AdaBoost-Variante übernommen werden.

4.2.3 Ordinal Gentle AdaBoost

Ebenso wie beim Vergleich zwischen den gewöhnlichen nominalen Real AdaBoost- und Gentle AdaBoost-Algorithmen soll im ordinalen Fall gelten, daß der einzige Unterschied auf der Verwendung eines alternativen Aktualisierungskriteriums basiert. Während im nominalen Fall die reinen Klassifikationswahrscheinlichkeiten als Basis dienen, stützt man sich nun wie bei Ordinal Real AdaBoost auf Scores, die wiederum anhand besagter Wahrscheinlichkeiten aller dichotomisierten Aufteilungen der Zielgröße zustande kommen. Darauf aufbauend basiert Ordinal Gentle AdaBoost nun auf den Differenzen solcher Scores, während Ordinal Real AdaBoost die für diese Gruppe von Verfahren typischen Quotienten zwischen Prognosewahrscheinlichkeiten verwendet. Der Algorithmus wurde so umgesetzt, wie er in Abbildung 4.6 dargestellt ist.

Das Vorzeichen von $f_j(x_i, L_m)$ hängt also im Gegensatz zu Real AdaBoost von der Differenz zwischen dem Score für Klasse j und einem mit Hilfe der Abstände gewichteten Durchschnittsscore aller anderen Klassen ab, was sich wiederum als numerisch stabiler als die diesbezüglich kritischen Quotienten erweisen sollte. Ebenso wie im nominalen Fall werden deshalb auch keine Konsistenzbedingungen für die Existenz von $f_j(x_i, L_m)$ benötigt.

kumulatives Ordinal Real AdaBoost

1. Starte mit Gewichten $w_1 = \cdots = w_{n_L} = 1/n_L$ für die Beobachtungen der Lernstichprobe L.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Die aktuellen Gewichte $w_1, \ldots, w_{n_L}$ werden für den Resampling-Vorgang verwendet. Basierend auf diesen Wahrscheinlichkeiten wird eine Lernstichprobe L_m vom Umfang n_L aus L mit Zurücklegen gezogen.
   (b) Anhand von L_m werden Klassifizierer $C^{(r)}(., L_m)$ für alle bezüglich r dichotomisierten Aufteilungen der ordinalen Klassenvariable entwickelt.
   (c) Die Lernstichprobe durchläuft jeden dieser Klassifizierer $C^{(r)}(., L_m)$, woraus man die Klassenwahrscheinlichkeiten $p^{(r)}(x) = \hat{P}(Y \leq r|x)$ erhält. Daraus berechnet man Scores

$$\begin{aligned} \hat{Y}_1^{(r)}(x) = \cdots = \hat{Y}_r^{(r)}(x) &= p^{(r)}(x) \\ \hat{Y}_{r+1}^{(r)}(x) = \cdots = \hat{Y}_k^{(r)}(x) &= 1 - p^{(r)}(x) \quad . \end{aligned}$$

   (d) Die Aufteilungen werden über

$$\hat{Y}_j(x) = \sum_{r=1}^{k-1} \hat{Y}_j^{(r)}(x)$$

   aggregiert.
   (e) Mit

$$f_j(x_i, L_m) = 0.5 \cdot \log \frac{\hat{Y}_j(x_i)}{\frac{1}{\sum_{l \neq j}^{k} d_{il}} \Sigma_{l \neq j}^{k} d_{il} \hat{Y}_l(x_i)}$$

   und d_{ij} = *Abstand zwischen wahrer Klasse und Klasse j für Beobachtung i* aktualisiere die Ziehungswahrscheinlichkeiten für den nächsten Schritt über

$$w_{i,new} = \frac{w_i \exp(-f_{Y_i}(x_i, L_m))}{\sum_{j=1}^{n_L} w_j \exp(-f_{Y_j}(x_j, L_m))} \quad .$$

3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über

$$\text{argmax}_j \left(\sum_{m=1}^{M} f_j(x, L_m) \right) \quad .$$

Abbildung 4.5: Algorithmus für kumulatives Ordinal Real AdaBoost

kumulatives Ordinal Gentle AdaBoost

1. Starte mit Gewichten $w_1 = \cdots = w_{n_L} = 1/n_L$ für die Beobachtungen der Lernstichprobe L.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Die aktuellen Gewichte $w_1, \ldots, w_{n_L}$ werden für den Resampling-Vorgang verwendet. Basierend auf diesen Wahrscheinlichkeiten wird eine Lernstichprobe L_m vom Umfang n_L aus L mit Zurücklegen gezogen.
   (b) Anhand von L_m werden Klassifizierer $C^{(r)}(., L_m)$ für alle bezüglich r dichotomisierten Aufteilungen der ordinalen Klassenvariable entwickelt.
   (c) Die Lernstichprobe durchläuft jeden dieser Klassifizierer $C^{(r)}(., L_m)$, woraus man die Klassenwahrscheinlichkeiten $p^{(r)}(x) = \hat{P}(Y \leq r|x)$ erhält. Daraus berechnet man Scores

$$\begin{aligned} \hat{Y}_1^{(r)}(x) = \cdots = \hat{Y}_r^{(r)}(x) &= p^{(r)}(x) \\ \hat{Y}_{r+1}^{(r)}(x) = \cdots = \hat{Y}_k^{(r)}(x) &= 1 - p^{(r)}(x) \quad . \end{aligned}$$

   (d) Die Aufteilungen werden über

$$\hat{Y}_j(x) = \sum_{r=1}^{k-1} \hat{Y}_j^{(r)}(x)$$

   aggregiert.
   (e) Mit

$$f_j(x_i, L_m) = \hat{Y}_j(x_i) - \frac{1}{\sum_{l \neq j}^{k} d_{il}} \sum_{l \neq j}^{k} d_{il} \hat{Y}_l(x_i)$$

   und d_{ij} = *Abstand zwischen wahrer Klasse und Klasse j für Beobachtung i* aktualisiere die Ziehungswahrscheinlichkeiten für den nächsten Schritt über

$$w_{i,new} = \frac{w_i \exp(-f_{Y_i}(x_i, L_m))}{\sum_{j=1}^{n_L} w_j \exp(-f_{Y_j}(x_j, L_m))} \quad .$$

3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über

$$\text{argmax}_j \left(\sum_{m=1}^{M} f_j(x, L_m) \right) \quad .$$

Abbildung 4.6: Algorithmus für kumulatives Ordinal Gentle AdaBoost

4.3 Ein ordinaler L_2-Boost-Algorithmus

Der L_2-Boost kann leicht in ein ordinales Verfahren umgewandelt werden, ohne daß kumulative oder sequenzielle Aufteilungen bezüglich der Zielgröße vorgenommen werden müssen.

Hier handelt es sich um eine Variante des Algorithmus von Bühlmann & Yu (2002b), der keine Dichotomisierung der ordinalen Zielgröße benötigt, sondern, ebenso wie der schon vorgestellte gewöhnliche L_2-Boost, auf Regressionstechniken basiert.

Da der mittlere quadratische Fehler einen sensitiven Indikator für ordinale Klassen darstellt, kann dieser Algorithmus durch eine einzige geringfügige Anpassung unmittelbar zur ordinalen Klassifikation genutzt werden: Die Werte von $\hat{F}_m(x)$ werden einfach auf das nächste Klassenlabel gerundet, um so den zulässigen Wertebereich von Y einzuhalten. Der Algorithmus lautet dann wie in Abbildung 4.7 angeführt.

Als Endergebnis gibt der Algorithmus also den reellen Wert der Regressionsprognose zurück, allerdings gerundet auf die nächstliegende ordinale Klasse. Analog zu den von Bühlmann & Yu (2002b) für den gewöhnlichen L_2-Boost vorgeschlagenen Beschränkungen des Wertebereichs von $\hat{F}$ auf das Intervall $[-1; 1]$, wird hier eine Einschränkung auf das Intervall $[0.5; k + 0.5]$ eingeführt, die eine zu extreme Abweichung der Prognose vom Wertebereich der ordinalen Klassen verhindert. Dies wird dadurch gesichert, daß zu keinem Zeitpunkt während des Ablaufs des Algorithmus die Prognose außerhalb dieses Intervalls liegen darf. Ist dies der Fall, so wird der Wert von $\hat{F}_m$ auf die Grenze 0.5 bzw. $k + 0.5$ gesetzt.

Ebenso wie beim nominalen L_2-Boost kann man empirisch feststellen, daß kleine Bäume mit geringer Hierarchietiefe als Basis zu deutlich besseren Generalisierungsergebnissen führen als größere Bäume mit vielen Endknoten, so daß im hierzu gehörigen empirischen Teil dieser Arbeit pauschal mit möglichst einfach strukturierten Bäumen gerechnet wird.

Ordinal L_2-Boost

1. Im ersten Schritt wird eine reellwertige Initial-Prognose $\hat{F}_0(x) = \hat{f}(x)$ über die Minimierung des Kleinst-Quadrat-Fehlers

$$\min \left(\sum_{i=1}^{n_L} (Y_i - \hat{f}(x_i))^2 \right)$$

   berechnet. Dabei gilt, daß Y_i in $\{1, \ldots, k\}$ liegt. Dann startet die Iteration mit $m = 1$.
2. Gehe im m-ten Schritt wie folgt vor:
   (a) Der negative Gradienten-Vektor

$$u_i = Y_i - \hat{F}_{m-1}(x_i)$$

   wird berechnet.
   (b) Die reellwertige Prognose $\hat{f}_m(x)$ wird nun an die aktuellen Residuen angepaßt. Dies erfolgt wiederum über die Minimierung des Kleinst-Quadrat-Fehlers

$$\min \left(\sum_{i=1}^{n_L} (u_i - \hat{f}_m(x_i))^2 \right) \quad .$$

   (c) Die Vorhersage $\hat{F}_m(x)$ wird aktualisiert durch

$$\hat{F}_m(x) = \hat{F}_{m-1}(x) + \hat{f}_m(x) \quad .$$

3. Nach M Schritten erhält man die aggregierte Abstimmung für die Beobachtung x über $\hat{F}_M(x)$, wobei auf die nächste ordinale Klasse gerundet wird.

Abbildung 4.7: Algorithmus für Ordinal L_2-Boost

Kapitel 5

Erweiterte Nächste-Nachbarn-Verfahren

Basierend auf der gängigen Nächste-Nachbarn-Methodik wird im Rahmen dieses Abschnitts ein flexibler Ansatz entwickelt, der das Basisverfahren in zwei Richtungen erweitert. Zunächst soll ein Gewichtungsschema eingeführt werden, das die nächsten Nachbarn anhand ihrer Ähnlichkeit zu einem neu zu klassifizierenden Fall gewichtet. Desweiteren kann leicht gezeigt werden, daß die Abstimmung der nächsten Nachbarn äquivalent zum Modus der Wahrscheinlichkeitsverteilung der prognostizierten Klasse ist. Auf dieser Feststellung basierend benutzt die zweite Erweiterung alternativ den Median oder das arithmetische Mittel dieser Verteilung, wenn die Zielgröße ordinales bzw. kardinales Skalenniveau aufweist. Ein *R*-Paket namens *kknn* mit den Implementierungen dieser Verfahren wurde im Rahmen dieser Arbeit im Zusammenwirken mit Dipl.-Stat. Klaus Schliep am *Allan Wilson Centre for Molecular Ecology and Evolution* der Massey University in Neuseeland erarbeitet.

Eine spezielle Kombination dieser Erweiterungen, nämlich eine Glättung anhand eines gewichteten arithmetischen Mittels, stellt dann die Verbindung zu der lokalen Regressionstechnik LOESS bzw. deren Spezialfall des Nadaraya-Watson-Schätzers her. Beide Verfahren sind beispielsweise in Chen, Härdle & Schulz (2004) oder Cleveland & Loader (1995) übersichtlich zusammengefaßt.

Zunächst muß hier ein Hinweis zur Nomenklatur erfolgen: Da die Bezeichnung k für die Anzahl der verschiedenen Klassen in der Zielgröße mit demjenigen k aus den k-Nächste-Nachbarn-Verfahren kollidieren würde, wird im Rahmen dieses fünften Kapitels ausnahmsweise das Kürzel c für die Anzahl der Klassen gewählt.

5.1 Einfache Nächste-Nachbarn-Klassifikation (NN)

Nächste-Nachbarn-Verfahren(Fix & Hodges (1951), siehe auch Cover & Hart (1967)) stellen eine der einfachsten und intuitivsten Techniken im Feld der Klassifikationsverfahren dar. Es handelt sich dabei, wie bereits kurz im zweiten Kapitel dieser Arbeit angedeutet, um ein nichtparametrisches Verfahren, bei dem die Klassifikation einer neuen Untersuchungseinheit in diejenige Klasse erfolgt, die jene Beobachtung aus der Lernstichprobe aufweist, die dem zu klassifizierenden Fall in Bezug auf die verwendeten Kovariablen am ähnlichsten ist. Die Bestimmung dieser Ähnlichkeit beruht auf Abstandsmaßen.

Formal läßt sich dieser einfache Sachverhalt wie folgt darstellen: Sei

$$L = \{(Y_i, x_i), i = 1, \ldots, n_L\}$$

der Lern- oder Trainingsdatensatz der beobachteten Daten. Dabei stehen wie in früheren Kapiteln die $Y_i \in \{1, \ldots, c\}$ für die Klassenzugehörigkeit, der Vektor $x_i' = (x_{i1}, \ldots, x_{ip})$ beschreibt die Werte der Kovariaten. Die Bestimmung des nächsten Nachbarn geschieht über eine beliebige Distanzfunktion $d(.,.)$. Für einen neu zu klassifizierenden Fall (Y, x) wird dann der nächste Nachbar $(Y_{(1)}, x_{(1)})$ aus der Lernstichprobe über

$$d(x, x_{(1)}) = \min_i \left(d(x, x_i)\right)$$

bestimmt und als Prognose $\hat{Y} = Y_{(1)}$, also die Klasse des nächsten Nachbarn, gewählt. Die Bezeichnungsweise $x_{(j)}$ bzw. $Y_{(j)}$ beschreibt dabei den j-ten Nachbarn zu x bzw. dessen Klassenzugehörigkeit.

Typische solche Distanzfunktionen sind beispielsweise der euklidische Abstand

$$d(x_i, x_j) = \left(\sum_{s=1}^{p} (x_{is} - x_{js})^2\right)^{\frac{1}{2}}$$

oder der Absolutabstand

$$d(x_i, x_j) = \sum_{s=1}^{p} |x_{is} - x_{js}| \quad .$$

Allgemein können beide Maße als Spezialfälle der sogenannten Minkowsky-Distanz

$$d(x_i, x_j) = \left(\sum_{s=1}^{p} |x_{is} - x_{js}|^q\right)^{\frac{1}{q}}$$

dargestellt werden. Der euklidische Abstand ergibt sich dabei für die Wahl $q = 2$, der Absolutabstand resultiert aus dem Parameterwert $q = 1$.

Die Funktionsweise dieses Verfahrens läßt sich, wie in Fahrmeir, Hamerle & Tutz (1996) beschrieben, am besten anhand des zufälligen Zustandekommens der Lernstichprobe erklären. Da es sich bei der Klasse $Y_{(1)}$ des nächsten Nachbarn $(Y_{(1)}, x_{(1)})$ zu einem neuen

Fall (Y, x) um eine Zufallsvariable handelt, ergibt sich die Klassifikationswahrscheinlichkeit von (Y, x) in Klasse $Y_{(1)}$ als $\mathrm{P}(Y_{(1)}|x_{(1)})$. Für sehr große Lernstichproben fallen x und $x_{(1)}$ sehr nahe zusammen, weshalb näherungsweise $\mathrm{P}(Y_{(1)}|x_{(1)}) \approx \mathrm{P}(Y|x)$ gilt. Damit wird die neue Beobachtung x also approximativ mit der Wahrscheinlichkeit $\mathrm{P}(Y|x)$ der wahren Klasse Y zugeordnet.

5.2 k-Nächste-Nachbarn-Klassifikation (kNN)

Eine erste Erweiterung dieses Ansatzes, die in der Praxis gängigerweise eingesetzt wird und weite Verbreitung gefunden hat, ist der sogenannte k-Nächste-Nachbarn-Ansatz. Hier zieht man nicht nur die nächste Beobachtung aus der Lernstichprobe zur Klassifikation heran, sondern die k ähnlichsten Fälle, wobei k vom Benutzer festzulegen ist. Dann fällt die Entscheidung zugunsten derjenigen Klasse, der die meisten dieser Nachbarn angehören.

Bezeichne dazu k_r die Anzahl der Beobachtungen aus der Gruppe der nächsten Nachbarn, die aus Klasse r stammen. Es gelte also

$$\sum_{r=1}^{c} k_r = k \quad .$$

Dann wird ein neuer Fall in diejenige Klasse l zugeordnet, für die gilt:

$$k_l = \max_r (k_r) \quad .$$

Auf diese Art und Weise wird verhindert, daß immer lediglich eine einzelne Beobachtung aus der Lernstichprobe alleine über die Prognose entscheidet. Wie man leicht sieht, wird über den Parameter k der Grad der Lokalität des Verfahrens gesteuert: Für $k = 1$ erhält man als maximal lokales Verfahren den einfachen Nächste-Nachbarn-Ansatz, für $k \to n_L$ ergibt sich dagegen ein globaler Mehrheitsentscheid der gesamten Lernstichprobe, also konstante Prognosen für alle neuen Beobachtungen, die klassifiziert werden sollen: Es wird immer die am häufigsten auftretende Klasse vorhergesagt.

5.3 Gewichtete k-Nächste-Nachbarn-Klassifikation (kkNN)

Diese zusätzliche Erweiterung basiert auf der Idee, daß solche Beobachtungen aus der Lernstichprobe, die besonders nahe an dem zu klassifizierenden Fall (Y, x) liegen, auch ein höheres Gewicht bei der Entscheidung erhalten sollen als solche Nachbarn, die weiter von (Y, x) entfernt sind. Dies ist bei kNN nicht der Fall: Zwar haben nur die k nächsten Nachbarn einen Einfluß auf die Prognose; Dieser Einfluß ist jedoch für jeden dieser Nachbarn gleich, egal wie stark die individuelle Ähnlichkeit zu (Y, x) ausgeprägt ist.

Um dieses Ziel zu erreichen, müssen die Distanzen, anhand denen im ersten Schritt die nächsten Nachbarn bestimmt werden, in Ähnlichkeiten umgewandelt werden, die dann als Gewichte verwendet werden können.

Standardisierung der Kovariaten

Wiederum werden zunächst die k nächsten Nachbarn anhand des Minkowsky-Abstandes bestimmt. Dazu benötigt man wie bisher zwei Parameter, nämlich die Anzahl der Nachbarn k und den Minkowsky-Parameter q zur Wahl der Art des Abstandsmaßes.

Um dafür zu sorgen, daß alle Kovariaten mit dem gleichen Gewicht in die Berechnung der Distanzen eingehen, müssen deren Werte standardisiert werden. Im Falle von verhältnis- oder differenzenskalierten Größen kann dieses Ziel einfach durch Division der Ausprägungen durch die Standardabweichung der jeweiligen Variable erreicht werden. Die zusätzliche Subtraktion des Mittelwertes, die noch zu einer vollständigen Standardisierung fehlen würde, ist in diesem Fall nicht notwendig, da diese Operation keinen Einfluß auf Abstände zwischen Beobachtungen hat.

Für ordinale Kovariaten mit m Kategorien werden im Programmpaket zwei Prozeduren angeboten: Man kann derartige Größen entweder so behandeln, als ob sie intervallskaliert wären, oder sie in $m-1$ Dummy-Variablen transformieren. Liegen beispielsweise 5 ordinale Klassen vor, so resultieren aus dieser Transformation die folgenden Ausprägungen der Dummy-Variablen $v_1, \ldots, v_4$:

class	v_1	v_2	v_3	v_4
1	1	1	1	1
2	-1	1	1	1
3	-1	-1	1	1
4	-1	-1	-1	1
5	-1	-1	-1	-1

Bei der Berechnung des Abstandes zwischen zwei Beobachtungen x_i und x_j entspricht dann die Anzahl der von Null verschiedenen Differenzen zwischen Dummy-Variablen dem Abstand an ordinalen Klassen zwischen x_i und x_j. Dieser Ansatz behandelt Abstände immer proportional und linear, unabhängig von der Wahl des Minkowsky-Parameters.

Auf ähnliche Art und Weise können Dummy-Variablen für nominale Kovariaten mit m Klassen entwickelt werden. Da es keine sinnvolle Definition einer ausgezeichneten Referenzkategorie gibt, wenn man mit Abständen arbeitet, benötigt man hier m Dummy-Variablen:

class	v_1	v_2	v_3	v_4	v_5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1

Dabei resultiert kein Unterschied, wenn zwei Beobachtungen identisch sind, sowie genau zwei Nicht-Null-Spalten, wenn die Beobachtungen unterschiedlichen Klassen angehören.

Nun entsteht das Problem, wie diese Dummy-Variablen standardisiert werden sollen. Dazu bietet das Programmpaket eine einheitliche Technik für beide Arten von Dummy-Variablen, die auf der Spur der Kovarianzmatrix einer Gruppe von zusammengehörigen Dummy-Größen basiert. Die Korrelationsstruktur wird dabei ignoriert und man nutzt den Ausdruck

$$\sqrt{\frac{1}{m}\sum_{i=1}^{m}\hat{\mathrm{Var}}(v_i)} \qquad \text{bzw.} \qquad \sqrt{\frac{1}{m-1}\sum_{i=1}^{m-1}\hat{\mathrm{Var}}(v_i)} \quad ,$$

um jeweils die zusammengehörigen Dummy-Variablen zu einer nominalen beziehungsweise ordinalen Kovariate durch diesen Wert zu dividieren. Dabei steht $\hat{\mathrm{Var}}(v_i)$ für die empirische Varianz der Dummy-Variable v_i. Diese Standardisierung aller zusammengehörigen Größen mit derselben durchschnittlichen Standardabweichung erscheint notwendig, da Unterschiede zwischen Klassen immer symmetrisch behandelt werden sollten, unabhängig von Unterschieden in den Standardabweichungen der einzelnen Dummy-Variablen.

Desweiteren würden Kovariaten mit vielen Klassen ohne eine zusätzliche Korrektur mehr Gewicht erhalten als andere, da dann einfach mehr Dummy-Variablen vorliegen, die jeweils gleichberechtigt zum Abstandsmaß beisteuern könnten und dabei jeweils das selbe Gewicht wie eine metrische Größe erhalten würden. Wenn also die Abstände berechnet werden, werden alle Differenzen von zusammengehörigen Dummy-Variablen mit den Konstanten $\frac{1}{m-1}$ beziehungsweise $\frac{1}{m}$ gewichtet, falls die ursprüngliche Kovariate ordinales beziehungsweise nominales Skalenniveau aufweist.

Natürlich unterscheidet sich diese Form der Standardisierung über die mittlere Varianz der Dummy-Variablen von der exakten Behandlung der metrischen Variablen. Ein derartiger Ansatz erscheint aber zumindest besser als kategoriale Variablen überhaupt nicht zu behandeln.

Die Standardisierung aller Arten von Kovariaten basiert ausschließlich auf den Beobachtungen der Lernstichprobe. Selbstverständlich könnte man die Werte der Kovariaten aller neuen Beobachtungen, die letztendlich klassifiziert werden sollen, zum Standardisierungsschritt hinzuziehen. Es erscheint aber als die konsistentere und vergleichbarere Herangehensweise, neue Beobachtungen stets mit demselben Faktor zu standardisieren. Die Resultate hängen also nur von den Werten in der Lernstichprobe ab.

Wenige Autoren beschäftigen sich mit dem Themenbereich, in welcher Form nominale und ordinale Kovariaten in Distanzmaße mit eingehen sollen. Alternative Ansätze zur Behandlung von kategorialen Variablen zu dem hier geschilderten finden sich zum Beispiel in Fahrmeir, Hamerle & Tutz (1996) oder Cost & Salzberg (1993).

Ein Gewichtungsschema der nächsten Nachbarn

Der Übergang von Distanzen zu Gewichten erfolgt dann im zweiten Schritt anhand von Kernfunktionen (daher das erste k in kkNN als Kürzel). Dabei handelt es sich um Funktionen $K(.)$ der Distanzen d, deren Maximum für $d = 0$ vorliegt und deren Funktionswerte umso kleiner werden, je weiter man sich von der Null entfernt. Es muß also gelten:

- $K(d) \geq 0$ für alle $d \in \mathbb{R}$
- $K(d)$ wird maximal für $d = 0$
- $K(d)$ fällt monoton für $d \to \pm\infty$

Typische Beispiele für derartige Funktionen sind:

- Rechteckskern $\frac{1}{2} \cdot \mathrm{I}(|d| \leq 1)$
- Dreieckskern $(1 - |d|) \cdot \mathrm{I}(|d| \leq 1)$
- Epanechnikov-Kern $\frac{3}{4}(1 - d^2) \cdot \mathrm{I}(|d| \leq 1)$
- Quartic- oder Biweight-Kern $\frac{15}{16}(1 - d^2)^2 \cdot \mathrm{I}(|d| \leq 1)$
- Triweight-Kern $\frac{35}{32}(1 - d^2)^3 \cdot \mathrm{I}(|d| \leq 1)$
- Kosinus-Kern $\frac{\pi}{4}\cos(\frac{\pi}{2}d) \cdot \mathrm{I}(|d| \leq 1)$
- Gauß-Kern $\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{d^2}{2}\right)$
- Inversions-Kern $\frac{1}{|d|} \cdot \mathrm{I}(|d| \leq 1)$

Beim letztgenannten Inversions-Kern muß zu den Abständen eine kleine Konstante $\epsilon > 0$ addiert werden, da er für einen Abstand von Null nicht definiert wäre.

Im Fall von Distanzen, die von ihrer Definition her stets positiv sein müssen, betrachtet man von diesen Funktionen selbstverständlich nur den positiven Teil des Definitionsbereichs (siehe auch Abbildung 5.1). Die Wahl der Art des Kerns stellt damit den dritten zu wählenden Parameter dar. Erfahrungsgemäß spielt aber die Wahl des speziellen Kerns (wenn man vom Sonderfall des Rechteckskerns absieht, der zu keiner Gewichtung der Beobachtungen führt) keine große Rolle.

Der Gauß-Kern wurde aus den später folgenden empirischen Untersuchungen weggelassen. Der Grund dafür ist, daß andere Normalverteilungen mit anderen σ^2-Parameterwerten für einen Definitionsbereich von $[0, 1]$ der Abstände D (siehe unten) geeigneter erscheinen. Eine gerechtfertigt beste Wahl ist aber nicht naheliegend gegeben.

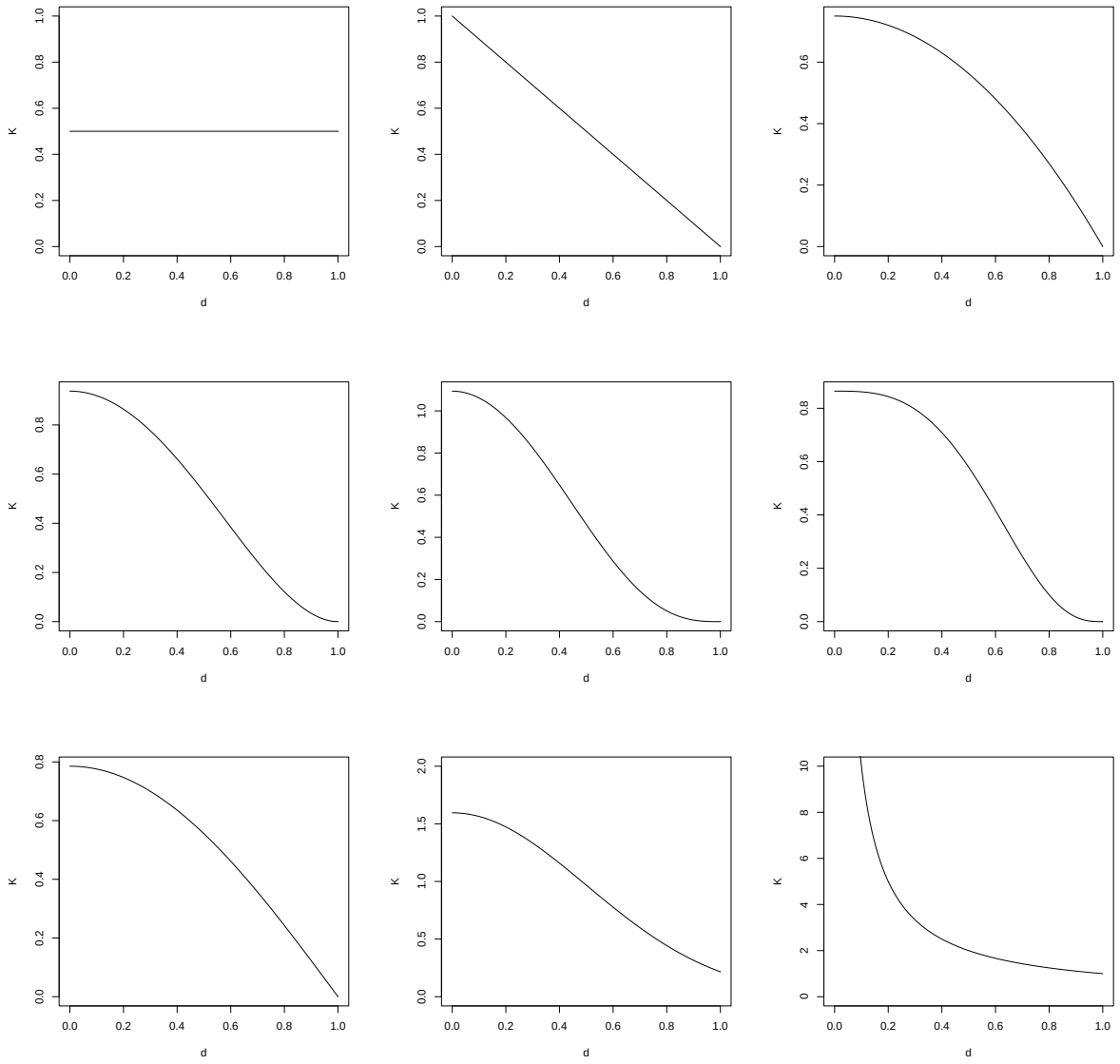

Abbildung 5.1: Übersicht über mögliche Kernfunktionen (von links oben nach rechts unten: Rechteck, Dreieck, Epanechnikov, Biweight, Triweight, Tricube, Cosinus, Gauß, Inversion. Der Tricube-Kern ist in der obigen Liste noch nicht aufgeführt, wird aber im späteren Vergleich der kkNN-Verfahren mit LOESS erwähnt; Der Gauß-Kern basiert auf einer Varianz σ^2 von 0.25 statt 1, um die Gestalt der Dichte deutlicher zu machen.)

Jede solche Kernfunktion benötigt eine Fensterbreite, so daß der Wertebereich außerhalb von gegebenen Schranken um das Maximum gleich Null beträgt. Diese Fensterbreiten werden bei kkNN automatisch aus der Distanz des ersten gerade nicht mehr berücksichtigten $(k+1)$-ten Nachbarn bestimmt. Dies geschieht indirekt durch die Normierung aller Distanzen mit der Distanz des $(k+1)$-ten Nachbarn:

$$D(x, x_{(i)}) = \frac{d(x, x_{(i)})}{d(x, x_{(k+1)})} \quad \text{für} \quad i = 1, \ldots, k \quad .$$

Die so normierten Distanzen liegen immer im Intervall $[0, 1]$. In der hier verwendeten Implementierung wird eine kleine Konstante $\epsilon > 0$ zu $d(x, x_{(k+1)})$ addiert, um für einige der nächsten Nachbarn Gewichte von 0 zu vermeiden. Dies könnte ansonsten passieren, wenn einer oder mehrere der Nachbarn exakt die gleiche Distanz wie der $(k+1)$-te Nachbar aufweisen, da die meisten Kerne am Fensterrand $D = 1$ den Wert 0 annehmen. Diese Bandweite von 1 ist insofern die angemessene Wahl, damit alle Beobachtungen, die weiter als der k-te Nachbar von x entfernt liegen, keine Rolle mehr spielen. Die Wahl der Bandweite erfolgt also adaptiv bezüglich der gegebenen Daten.

Zusammenfassung der kkNN-Methode

Sind auf diese Art und Weise Ähnlichkeitsmaße zu den Beobachtungen der Lernstichprobe ermittelt, so kann die Klassifikation eines neuen Falles (Y, x) in diejenige Klasse erfolgen, die die höchste aufaddierte Ähnlichkeit zu (Y, x) aufweist:

$$\max_r \left(\sum_{i=1}^{k} K(D(x, x_{(i)})) I(Y_{(i)} = r) \right) \quad .$$

Sowohl kNN als auch NN sind damit als Spezialfälle in kkNN enthalten: kNN resultiert bei Wahl des Rechteckskerns, NN ergibt sich unabhängig von der Wahl der Kernfunktion für $k = 1$.

Hauptziel des erweiterten Ansatzes ist es, das Verfahren bis zu einem gewissen Grad von einer falschen Wahl von k unabhängig zu machen, die zu hohen Fehlklassifikationsraten führt. Diese Anzahl der nächsten Nachbarn steckt bei kkNN nun implizit in den Gewichten: Ist k fälschlicherweise zu hoch gewählt, so wird k automatisch nach unten adjustiert. In diesem Fall dominiert nämlich eine kleine Anzahl von Nachbarn mit hohen Gewichten über die übrigen entfernteren Nachbarn, deren Klassen aufgrund der deutlich niedrigeren Gewichte keinen Einfluß auf die Prognose nehmen können.

In Abbildung 5.2 wird eine schematische Darstellung des Algorithmus in seiner allgemeinsten Form gegeben. Alle gängigen Nächste-Nachbarn-Techniken sind, wie bereits oben geschildert, als Spezialfälle enthalten.

Gewichtete k-Nächste-Nachbarn-Klassifikation (kkNN)

1. Gegeben seien eine Lernstichprobe $L = \{(Y_i, x_i), i = 1, \ldots, n_L\}$ von Beobachtungen x_i mit bekannter Klassenzugehörigkeit Y_i, sowie eine neu zu klassifizierende Beobachtung x, deren Klassenzugehörigkeit Y prognostiziert werden soll.
2. Bestimme anhand einer Abstandsfunktion $d(x_i, x)$ die $k+1$ nächsten Nachbarn zu x.
3. Der $(k+1)$-te Nachbar dient zur Normierung der k geringsten Distanzen auf Werte in $[0; 1]$ über
$$D_{(i)} = D(x, x_{(i)}) = \frac{d(x, x_{(i)})}{d(x, x_{(k+1)})} \quad .$$
4. Wandle die normierten Distanzen D_i mit Hilfe einer Kernfunktion $K(.)$ in Gewichte $w_{(i)} = K(D_{(i)})$ um.
5. Wähle als Prognose für die Klassenzugehörigkeit Y der Beobachtung x diejenige Klasse, die eine gewichtete Mehrheit der k nächsten Nachbarn aufweist:
$$\hat{Y} = \max_r \left(\sum_{i=1}^{k} w_{(i)} I(Y_{(i)} = r) \right) \quad .$$

Abbildung 5.2: Algorithmus für kkNN

Betrachtet man ganz allgemein die Vorgehensweise bei all diesen Verfahren, so kann man kkNN, und damit auch die einfacheren Nächste-Nachbarn-Verfahren, im folgenden Sinne als Voting- oder Ensemble-Techniken bezeichnen: Mehrere potenzielle Klassifizierer (nämlich die nächsten Nachbarn) stimmen ab, woraus ein Gesamtresultat per (gewichtetem) Mehrheitsentscheid gebildet wird. Dies zeigt eine gewisse Ähnlichkeit zu modernen Ensemble-Techniken wie Bagging oder Boosting, wo ebenfalls eine Abstimmung zwischen den Klassifizierern erfolgt.

5.4 Berücksichtigung ordinaler Strukturen

Median statt Modus

Eine zweite, vom Gewichtungsansatz unabhängige Erweiterung, stellt der Umgang mit Zielgrößen von unterschiedlichem Skalenniveau dar. Die Klassifikationsversion von kkNN, die ja eine nominale Zielgröße prognostiziert, arbeitet dabei mit einem gewichteten Mehrheitsentscheid der nächsten Nachbarn. Diese Vorgehensweise kann ebenso durch den Modus der geschätzten Wahrscheinlichkeitsverteilung der Prognosen beschrieben werden, die aus den in der Summe auf 1 normierten aufaddierten Gewichten für die Klassenzugehörigkeiten

resultiert:

$$\hat{P}(Y = r|x, L) = \frac{\sum_{i=1}^{k} w_{(i)} I(Y_{(i)} = r)}{\sum_{i=1}^{k} w_{(i)}} \quad .$$

Von dieser Feststellung ausgehend erscheint es naheliegend, für ordinale Zielgrößen entsprechend den Median

$$\text{argmax}_r \left(\sum_{i=1}^{r} \hat{P}(Y = i|x, L) \leq 0.5 \right)$$

dieser Verteilung heranzuziehen. Außerdem erscheint insofern auch die Prognose von kardinalskalierten Zielgrößen anhand des arithmetischen Mittels eben dieser Verteilung möglich.

Bei Verwendung des arithmetischen Mittels besteht auch ein enger Bezug zu der lokalen Regressionstechnik LOESS. Dabei wird die Quadratsumme der Residuen eines lokalisierten Regressionsproblems minimiert:

$$\min_\beta \sum_{i=1}^{n_L} (Y_i - \beta_0 - \beta_1(x_i - x))^2 \, K\left(\frac{x_i - x}{d(x, x_{(k)})} \right)$$

Wenn die Kovariaten lediglich zur Bestimmung der Nachbarschaft von Beobachtungen berücksichtigt werden, wenn also $\beta_1 = 0$ gewählt wird, erhält man den Spezialfall des Nadaraya-Watson-Schätzers, eine lokale Glättungtechnik, die stückweise konstante Basisfunktionen nutzt. Dies ist identisch zur Prognostizierung anhand eines gewichteten Mittelwerts aller Beobachtungen, die in das jeweilige Nächste-Nachbarn-Fenster fallen:

$$\hat{Y} = \text{E}(Y|x) = \frac{\sum_{i=1}^{n_L} K\left(\frac{x_i - x}{d(x, x_{(k)})} \right) Y_i}{\sum_{i=1}^{n_L} K\left(\frac{x_i - x}{d(x, x_{(k)})} \right)} = \frac{\sum_{i=1}^{k} w_{(i)} Y_{(i)}}{\sum_{i=1}^{k} w_{(i)}}$$

und stellt damit genau die Vorgehensweise von kkNN unter Verwendung des arithmetischen Mittels der Klassenverteilung dar.

In diesem Sinne besteht ein besonders enger Zusammenhang zwischen Nadaraya-Watson und kkNN, denn Nadaraya-Watson bildet gewissermaßen die Schnittstelle zwischen kkNN und LOESS. Die einzigen kleinen Unterschiede bestehen darin, daß kkNN eine Vielzahl von möglichen Kernfunktionen zur Verfügung stellt, um viele alternative Gewichtungsschemata zuzulassen, während bei LOESS speziell der Tricube-Kern

$$K(d) = \frac{70}{81}(1 - |d|^3)^3 \cdot \text{I}(|d| \leq 1)$$

verwendet wird (siehe auch Abbildung 5.1). Desweiteren basiert die Normierung der Abstände bei kkNN auf dem $(k+1)$-ten Nachbarn statt auf dem k-ten (wie bei LOESS). So hat der k-te Nachbar gerade noch einen Einfluß auf die Prognose, was den Ursprung dieses Ansatzes innerhalb der k-Nächste-Nachbarn-Techniken bekräftigt.

Da das Arbeiten mit dem arithmetischen Mittel der Klassenverteilung, in anderen Worten also mit Nadaraya-Watson bzw. LOESS ohne Kovariaten, zu keinen neuen Erkenntnissen führt, soll der Schwerpunkt der empirischen Untersuchungen auf jeden Fall im Klassifikationskontext zu finden sein, wobei selbstverständlich besonders die Verwendung des Medians der Klassenverteilung im Mittelpunkt steht. Diese Aufgabenstellung war bisher nicht durch lokale Regressionstechniken wie LOESS abgedeckt, die speziell auf metrische Variablen ausgerichtet sind. Das Arbeiten mit dem Median ist lediglich im Feld der lokalen Glättungstechniken gängig, wobei aber normalerweise keine ordinalen Zielgrößen vorkommen. Dennoch soll hier hervorgehoben werden, daß die Nadaraya-Watson-Schätzung als Spezialfall für metrische Variablen komplett im *kknn*-Paket zur Verfügung steht.

y-Kerne

Betrachtet man nochmal die Schätzung für die Verteilung der Klassenzugehörigkeit

$$\hat{P}(Y = r|x, L) = \frac{\sum_{i=1}^{k} w_{(i)} I(Y_{(i)} = r)}{\sum_{i=1}^{k} w_{(i)}} \quad ,$$

so bietet sich eine weitere, zusätzlich zum Median anwendbare Erweiterung an, um der ordinalen Struktur in der Zielgröße Rechnung zu tragen. In diesem Term kann die Indikatorfunktion $I(Y_{(i)} = r)$ selbst wieder als eine Kernfunktion angesehen werden, lediglich ein Punkt, nämlich $Y_{(i)}$, fällt dabei in das Beobachtungsfenster. Setzt man an dieser Stelle dagegen eine andere Kernfunktion, zum Beispiel einen Dreieckskern

$$\left(1 - \frac{|Y_{(i)} - r|}{\delta + 1}\right) \cdot \mathrm{I}(|Y_{(i)} - r| \leq \delta) \quad , \qquad \delta \in \{0, 1, \ldots, c-1\}$$

so wird der Score, der sich aus der Nähe der Nachbarschaft von $x_{(i)}$ zu x ergibt, nicht nur auf dessen Klassenzugehörigkeit $Y_{(i)}$, sondern in geringerem Maße auch auf Nachbarklassen vergeben. Wählt man δ beispielsweise gleich 0, so handelt es sich um eine gewöhnliche Indikatorfunktion, das heißt nichts ändert sich im Vergleich zur bisherigen Vorgehensweise. Für $\delta = 1$ erhält dagegen die Klassenzugehörigkeit $Y_{(i)}$ des Nachbarn das Gewicht 1, die beiden (ordinalen) Nachbarklassen immerhin aber noch einen Wert von 0.5. Je größer das (ganzzahlige) δ gewählt wird, desto mehr Nachbarklassen werden in diese Betrachtung mit einbezogen.

Der resultierende Klassifikator mit einem solchen y-Kern lautet dann also

$$\hat{P}(Y = r|x, L) \propto \sum_{i=1}^{k} w_{(i)} \left(1 - \frac{|Y_{(i)} - r|}{\delta + 1}\right) \cdot \mathrm{I}(|Y_{(i)} - r| \leq \delta) \quad .$$

Eine Erweiterung des nicht weiter untersuchten Ansatzes für metrische Größen mit Hilfe solcher y-Kerne ist prinzipiell möglich, aber deutlich komplizierter und anders algorithmisch umzusetzen, vor allem bei der Auswahl von geeigneten Parameterwerten δ. Anstatt

nur auszuwählen, wieviele der Nachbarklassen mit einbezogen werden sollen, müßten Kerne mit stetig variierbarer Bandweite implementiert werden, weil statt Nachbarklassen benachbarte Intervalle betrachtet werden müßten. Da sich diese Arbeit jedoch ausschließlich mit dem Klassifikationsaspekt beschäftigt, wurde auf die Erweiterung für metrische Größen verzichtet.

5.5 Nächste-Nachbarn-Klassifikation in der Literatur

Alternative Gewichtungstechniken zu der hier gewählten werden in anderen Arbeiten erwähnt, weisen jedoch entscheidende Unterschiede zum kkNN-Ansatz auf. Beispielsweise wird in Fahrmeir, Hamerle & Tutz (1996) eine alternative Methode vorgestellt, bei der eine feste Anzahl von nächsten Nachbarn pro Klasse r ($r = 1, \ldots, c$) bestimmt wird und dann die mittlere Distanz eines zu klassifizierenden Punktes zu diesen Klassenrepräsentanten herangezogen wird. Für Klasse r werden also jeweils die k_r nächsten Nachbarn x_{rj} mit $j = 1, \ldots, k_r$ bestimmt. Prognostiziert wird dann diejenige Klasse l, für die

$$\frac{k_l}{\sum_{j=1}^{k_l} d(x, x_{lj})} = \max_r \frac{k_r}{\sum_{j=1}^{k_r} d(x, x_{rj})}$$

gilt. Die Anzahl k_r der Nachbarn wird idealerweise proportional zum Klassenumfang gewählt.

Bei kkNN spielt dagegen immer noch die entscheidende Rolle, wieviele der nächsten Nachbarn des Gesamtdatensatzes jeweils auf die verschiedenen Klassen entfallen. Damit besteht eine wesentlich deutlicher ausgeprägte Ähnlichkeit zu kNN, welches alleine auf diesen Zählungen beruht. Die Gewichte kommen erst im finalen Klassifikationsschritt ins Spiel. Desweiteren wird darauf hingewiesen, daß der einfachste Nächste-Nachbarn-Klassifikator im Falle von endlichem Stichprobenumfang asymptotisch suboptimal ist. Die engen Grenzen für diese Abweichung sind jedoch bekannt und können durch die Erweiterung auf k-Nächste-Nachbarn weiter reduziert werden. Zuletzt wird im Rahmen von Glättungsverfahren ein gleitender Median (engl. *running median*) erwähnt, der mit der hier vorgestellten Technik zur ordinalen Klassifikation zusammenhängt, jedoch noch nicht im Rahmen von ordinalen Prognosen verwendet wurde.

Paik & Yang (2004) verwenden Kombinationen von vielen kNN-Klassifizierern mit unterschiedlichen Parameterwerten für k und verschiedenen Subgruppen von Kovariaten, um die Resultate eines einzelnen kNN-Prädiktors zu verbessern. Diesen Ansatz bezeichen sie als *Adaptive Klassifikation durch Mischung* (ACM). Er arbeitet ebenso mit Gewichten, aber anstatt die Beobachtungen aus der Lernstichprobe zu gewichten, wird ein Gewichtungsschema für die einzelnen Klassifikatoren selbst bestimmt, das auf deren Klassifikationswahrscheinlichkeiten basiert. Distanzen zwischen Beobachtungen kommen darin also nicht vor. Die Autoren erhalten so bessere Ergebnisse als durch die feste a priori-Wahl von k und den Kovariaten mittels Kreuzvalidierung, besonders dann, wenn diese Wahl sehr instabil ist.

Nächste-Nachbarn-Klassifikation mit flexibler Metrik von Friedman (1994) stellt eine dritte, komplett verschiedene Idee für ein Gewichtungsschema dar. Hier werden lokal flexible Gewichte für die Kovariaten vergeben, um deren lokale Relevanz berücksichtigen zu können, die wiederum mit Hilfe von rekursiven Partitionierungs-Techniken (wie bei CART) geschätzt wird. Dabei wird nach den lokal wichtigsten Variablen gesucht, wobei auch geeignete Linearkombinationen, die ihrerseits wieder über lineare Diskriminanzanalyse bestimmt werden, zugelassen sind. Es findet also wieder keine Gewichtung der Beobachtungen in der Lernstichprobe statt. Stattdessen ist nun aber die Metrik zur Bestimmung von Distanzen lokal flexibel, so daß man sich nicht mehr auf eine sehr unflexible Variablenselektion, die jede Kovariate nur entweder ausschließen oder identisch zu allen übrigen gewichten kann, verlassen muß.

Kapitel 6

Empirische Studien

6.1 Empirische Vergleiche bei nominaler Zielgröße

6.1.1 Studiendesign

Das gängige Gütemaß bei Klassifikationsverfahren ist die Fehlklassifikationsrate. Sie gibt den prozentualen Anteil von falsch klassifizierten Untersuchungseinheiten an. Dieses Maß wird gewöhnlich anhand einer separaten Kontrollstichprobe ermittelt, deren Einheiten nicht in der Lernstichprobe aufgetaucht sind. Man kann es zwar auch anhand der gleichen Daten ermitteln, die schon als Lernstichprobe verwendet wurden. Eine solche Resubstitutionsfehlerrate überschätzt aber die wahre Qualität des Verfahrens. Dies ändert jedoch nichts daran, daß auch ein solches Maß zum Vergleich verschiedener Verfahren herangezogen werden kann, um beispielsweise zu bestimmen, wie gut die Lerndaten selbst durch das Modell angepaßt werden. Beide Varianten, also das Arbeiten mit getrennten Lern- und Test-Stichproben sowie die Verwendung des gesamten Datensatzes zur Resubstitution, kommen hier zur Anwendung.

Nachdem im Rahmen dieser Arbeit bereits einige Klassifikationsvarianten vorgestellt wurden, nämlich neben diversen Nächste-Nachbarn-Techniken das Basis-Verfahren CART alleine sowie Bagging und verschiedene Versionen von Boosting, die jeweils auf diese Basis aufsetzen, wird nun die Qualität ihrer Prognosen anhand mehrerer empirischer Datensätze untersucht. Dazu muß zunächst auf die Parameter, die bei den einzelnen Verfahren variiert werden können, eingegangen werden.

Die Funktionen aller zu vergleichenden Ensemble-Methoden auf Basis von CART werden mit den Parametern *valid*, *runs* und *tree.depth* aufgerufen. Ist *valid* wie in der Voreinstellung gleich FALSE, so werden alle Daten sowohl zur Modellbildung als auch zur Bestimmung der Fehlklassifikationsrate herangezogen. Das Ergebnis ist also eine Resubstitutionsfehlerrate. Ist *valid* gleich TRUE, so wird mit getrennter Lern- und Test-Stichprobe

gearbeitet. Nur in diesem Fall erhält der Parameter *runs* eine Bedeutung: Er besagt, wieviele zufällige Aufteilungen der Daten in zwei Drittel Lernstichprobe und ein Drittel Validierungsstichprobe durchgeführt werden sollen. Der Parameter *tree.depth* gibt schließlich an, welche Hierarchietiefe alle an den Modellen beteiligten Bäume aufweisen sollen. Die Bäume werden also nicht individuell optimiert, sondern es wird von vornherein festgelegt, auf welche Größe sie maximal wachsen dürfen. Zwei Argumente sprechen für diese Vorgehensweise: Erstens erspart man sich Rechenzeit in hohem Umfang, wenn auf die aufwendige Kreuzvalidierung zur Bestimmung der optimalen Baumgröße und die Beschneidung der schwächsten Äste verzichtet werden kann. Zweitens ist der Gütevergleich zwischen den verschiedenen Verfahren transparenter, wenn alle Bäume in etwa die gleiche Anzahl von Endknoten aufweisen, wenn also die Güte der Resultate nicht auf zufällige Schwankungen der Baumgrößen zurückzuführen ist. Ein Nachteil dieser Vorgehensweise ist sicherlich die Tatsache, daß das Potenzial der untersuchten Verfahren nicht voll ausgeschöpft wird. Dies sollte aber zugunsten der Vergleichbarkeit der Ergebnisse akzeptiert werden.

Ein weiterer Parameter, der bei allen Voting-Verfahren, nicht jedoch beim reinen CART zur Anwendung kommt, ist *cycles*. Er legt fest, wieviele Zyklen die Algorithmen von Bagging bzw. Boosting durchlaufen sollen. Zu Beginn der Experimente wurde speziell beim Boosting noch der Parameter *log.weight* herangezogen, der festlegt, ob bei der gewichteten Abstimmung die rohen Gewichte b_m oder ihr Logarithmus c_m verwendet wird. Da in der Literatur beides Erwähnung findet, wurden beide Varianten ausprobiert. Die Resultate unterschieden sich aber nur in so geringem Maße, daß letztendlich die Algorithmen ausschließlich in der oben beschriebenen Form verwendet wurden.

Bei den Nächste-Nachbarn-Techniken werden in den empirischen Studien sowohl die Anzahl der nächsten Nachbarn k und der Minkowsky-Parameter q, als auch die verwendete Kernfunktion variiert. Da eine Resubstitution der Einheiten aus der Lernstichprobe bei einem derartigen Prototypen-Verfahren keinen Sinn machen würde - es würde stets die Fehlerrate Null erzielt, da der nächste Nachbar der Punkt selbst ist - basieren hier alle Ergebnisse auf getrennten Lern- und Teststichproben.

Desweiteren werden auch Resultate präsentiert, die auf kreuzvalidierten (siehe Abschnitt 2.1) Parameterwerten basieren. Dabei können gleichzeitig die optimalen Parameter für die Anzahl der Nachbarn k und die Kernfunktion bestimmt werden. Dies geschieht über die sogenannte *leave-one-out*-Methode: Je eine der Beobachtungen wird weggelassen, um dann anhand der verbliebenen Datenpunkte prognostiziert zu werden. Dies geschieht für alle Datenpunkte und anschließend wird über die einzelnen Fehlerraten gemittelt. Um eine solche Optimierung nicht zu komplex und instabil zu gestalten, werden aber einige Einschränkungen getroffen: k wird stets auf Werte zwischen 1 und 25 beschränkt und für die Kernfunktionen werden drei Varianten ausprobiert. Zuerst läßt man als Vergleichsresultat lediglich den Rechteckskern zu, als zweites nur einen der erfolgversprechenden Gewichtungskerne. Die letzte Variante stellt dann die eventuell zu komplexe und damit instabile Auswahl aus allen zur Verfügung stehenden Kernfunktionen dar.

Die Wahl, welche der möglichen Kernfunktionen in der zweiten Variante herangezogen

wird, basiert auf empirischen Erfahrungen. Als besonders konstant erwiesen sich bei Versuchen mit zahlreichen Datensätzen der Epanechnikov- sowie der Triweight-Kern. Deren Ergebnisse zählten fast immer zu den besten, was allerdings nicht besagen soll, daß sie stets über die anderen dominieren. Eine detailliertere Studie speziell dieser beiden Kerne zeigte, daß der Triweight-Kern dazu tendiert, mit wachsender Anzahl k von Nachbarn langsam zu immer geringeren Fehlerraten zu führen, bis irgendwann k so hoch ist, daß die Fehlerraten ebenso langsam wieder ansteigen. Der Epanechnikov-Kern liefert dagegen meist sehr schnell fallende Fehlerraten, die bereits bei einer deutlich kleineren Wahl von k ihr Minimum erreichen und dann auch schneller wieder ansteigen. Bei beiden liegt ein optimaler Wert für k jedoch fast immer über der idealen Wahl von k im Falle eines Rechtecks-Kerns.

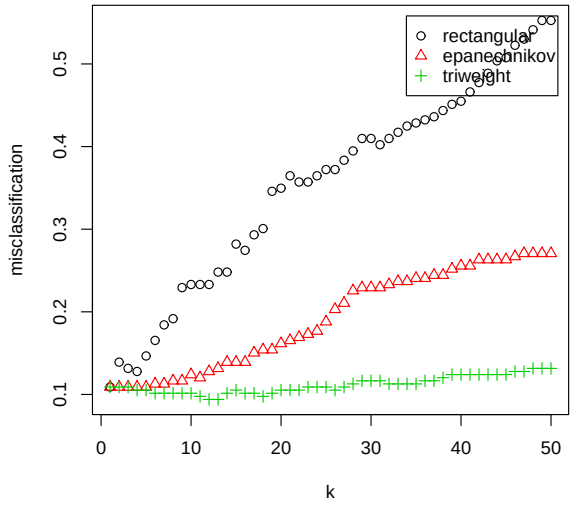

Abbildung 6.1: Typisches Beispiel (eine zufällige Aufteilung der Soybean-Daten) zum Vergleich der Testfehlerraten bei Verwendung von Epanechnikov- und Triweight-Kern

Muß man also davon ausgehen, daß dieses ideale k der gewöhnlichen Nächste-Nachbarn-Methode sehr klein ist (was bei den meisten der hier betrachteten nominalen Datensätzen der Fall ist) und man sich mit der Wahl der Anzahl von Nachbarn eventuell deutlich oberhalb dieses Wertes befindet, so bietet sich der Triweight-Kern an, da die Gefahr besteht, daß die Fehlerraten beim Epanechnikov-Kern ihr Optimum bereits hinter sich gelassen haben und für zu hohes k wieder deutlich angestiegen sind. Ist der ideale Parameterwert für k gleich 1, dann steigen bei Epanechnikov die Fehlerraten oft bereits ab $k = 3$ langsam wieder an (siehe Abbildung 6.1), während bei Triweight sogar noch eine geringfügige Verbesserung erzielt werden kann.

Zum Abschluß dieser Betrachtungen der Ansätze zum Vergleich von verschiedenen Verfahren sei noch auf eine Arbeit von Salzberg (1999) hingewiesen, der die in der wissenschaftlichen Praxis übliche Vorgehensweise kritisiert. Vergleiche von mehreren Verfahren anhand

von einigen Datensätzen erlauben demnach keinesfalls Aussagen zur globalen Überlegenheit von einer Technik über andere. Vielmehr gibt es definitiv keine besten Algorithmen für alle beliebigen Situationen. Dies ist auch als kritischer Punkt der ansonsten wichtigen und guten Idee, Sammlungen von geeigneten Benchmark-Datensätzen anzulegen, zum Beispiel dem Archiv für Klassifikationsdatensätze der University of California, festzuhalten.

Speziell was das Testen von signifikanten Unterschieden zwischen Verfahren betrifft, sieht der Autor einige angreifbare Punkte: So führen Hunderte von empirischen Studien anhand ein und desselben Datensatzes unweigerlich zu einigen falsch positiven Testergebnissen, die dann auch prompt dokumentiert werden, während die Vielzahl der nicht signifikanten Resultate für gewöhnlich unter den Tisch fällt. Werden desweiteren mehrere Lern-Test-Paare aus einem Datensatz gezogen, so bestehen aufgrund der Überschneidungen von Beobachtungen Abhängigkeiten zwischen den einzelnen Teststichproben. Eine einfache Bonferroni-Korrektur der p-Werte wegen multiplen Testens, wie sie bei unabhängigen Beobachtungen üblich wäre, reicht dann nicht mehr aus. Statt der Verwendung von t-Tests, wie sie in der Praxis meist herangezogen werden, führt der Autor zum Vergleich von zwei Verfahren einen Binomialtest an, der auf der Zählung von Beobachtungen aus der Teststichprobe basiert, die von je einem der Verfahren richtig, vom anderen jedoch falsch klassifiziert worden sind. Ein Problem ist hier allerdings, daß die Anzahl der von beiden Verfahren gleich richtig oder gleich falsch klassifizierten Beobachtungen nicht mit eingeht, obwohl viele gleiche Resultate gegen Unterschiede in den Verfahren sprechen. Zuletzt wird noch auf Randomisierungstests eingegangen. Hier werden die Klassenlabels zufällig variiert und geprüft, wie stark ein Verfahren dann von dem hier natürlich zu erwartenden Zufallsergebnis abweicht. Dieser Wert kann als Bias-Korrektur für die jeweilige Technik verwendet werden.

Andererseits muß angemerkt werden, daß der Vergleich der Ergebnisse von verschiedenen Verfahren durchaus anhand von gewöhnlichen Tests zum Mittelwertvergleich durchgeführt werden kann, wenn man mit simulierten Daten arbeitet. Hier können nämlich beliebig viele unabhängige Wiederholungen von Ziehungen vorgenommen werden. Allerdings kranken simulierte Daten stets an dem Problem, daß sie auf einem festgelegten Modell basieren. Wie gut Verfahren bei solchen künstlichen Daten abschneiden, läßt deshalb nur wenig Schlüsse auf reale, empirische Datensituationen zu, was ja das eigentliche Ziel jeder Vergleichsstudie sein sollte.

Zuletzt ist noch ein wichtiger Punkt empirischer Vergleichsstudien zwischen verschiedenen Verfahren anzumerken: Ein fairer Vergleich ist nur dann möglich, wenn die Parameter aller beteiligten Techniken optimal gewählt werden (engl. *parameter tuning*). Beschäftigt man sich dagegen lediglich mit der eigenen Methode ausführlicher, während bei den anderen nur die Basiseinstellungen herangezogen werden, so ist eine Überschätzung des Potenzials des eigenen Verfahrens die logische Konsequenz. Im Idealfall findet das *tuning* a priori an einem unabhängigen Datensatz oder per Kreuzvalidierung statt.

6.1.2 Datensätze

Im Folgenden werden die verwendeten Datensätze kurz vorgestellt. Einige weisen lediglich eine dichotome Zielgröße auf, einige andere stellen ein nominales Mehrklassenproblem dar. Alle Datensätze aus dem Datenarchiv der University of California stehen im Internet über
`http://www.ics.uci.edu/pub/machine-learning-databases.html`
zur Verfügung.

Wisconsin-Breast-Cancer-Datensatz

Dieser Datensatz, der ursprünglich an den University of Wisconsin Hospitals in Madison erhoben wurde, stammt aus dem umfangreichen Datenarchiv der University of California in Berkeley. Es handelt sich um einen der Standard-Datensätze, die von Breiman als Beispiele bei der Entwicklung von Bagging verwendet wurden. Seitdem wird er in einer Vielzahl von Veröffentlichungen zum Thema Ensemble-Techniken herangezogen.

Die Daten wurden inhaltlich dazu verwendet, um mit Hilfe einiger Variablen, die Brustkrebserkrankungen mit Hilfe von zellulären Charakteristiken des Tumors beschreiben, ein Modell zu bilden, das die Art des Tumors in gut- bzw. bösartig klassifiziert. Die Zielgröße ist also eine binäre Variable, die die Gut- bzw. Bösartigkeit des Tumors darstellt. Als Einflußgrößen werden insgesamt neun Variablen verwendet, die jeweils zehn ordinale Klassen annehmen können.

Alle Fälle, die mindestens einen fehlenden Wert aufweisen, wurden nachträglich entfernt, so daß von den ursprünglich 699 Fällen 683 übrigbleiben, die sich im Verhältnis 444 zu 239 auf die beiden Klasen aufteilen.

Glass-Identification-Datensatz

Auch dieser Datensatz stammt aus dem Datenarchiv der University of California und kann zu den von Breiman verwendeten Standard-Datensätzen gezählt werden.

Anhand von chemischen Indikatoren soll hier ein Modell gebildet werden, das Glas-Material bezüglich des industriellen Verwendungszwecks klassifiziert. Insgesamt weist diese Zielgröße sieben Klassen auf, wobei in dieser Version des Datensatzes eine der Klassen nicht besetzt ist, so daß von sechs nominalen Klassen ausgegangen werden kann. Die Umfänge der vier kleineren Klassen variieren zwischen 9 und 29 Beobachtungen, die beiden großen Klassen weisen 70 bzw. 76 Einheiten auf.

Insgesamt setzt sich der Datensatz aus 214 Fällen zusammen, die in sämtlichen Variablen keine fehlenden Werte aufweisen. Neun metrische Größen, die in erster Linie die chemische Zusammensetzung des Materials beschreiben, werden als unabhängige Variablen verwendet.

Ionosphere-Datensatz

Beim nächsten Datensatz, der ebenfalls dem Datenarchiv aus Berkeley entnommen ist, handelt es sich erneut um eines von Breimans Standardbeispielen.

Hier liegen Radardaten zur Untersuchung von freien Elektronen in der Ionosphäre vor. Die erhaltenen Signale werden aufgrund ihrer komplexen elektromagnetischen Struktur in einer ganzen Reihe von Variablen festgehalten. Ziel ist es, anhand dieser Daten zu klassifizieren, ob es sich um ein gutes Resultat (die Struktur in der Ionosphäre ist erkennbar) oder um ein schlechtes (das Signal liefert keine klare Struktur) handelt.

Insgesamt liegen 351 Meßpunkte vor, die sich im Verhältnis 126 zu 225 auf die beiden Klassen aufteilen und keine fehlenden Werte aufweisen. Die Radardaten werden in Form von 34 metrischen Variablen registriert. Als Zielgröße dient eine binäre Größe, die angibt, ob das Radarbild gut oder schlecht ist.

Soybean-Datensatz

Hierbei handelt es sich wiederum um einen der Standard-Datensätze von Breiman, der dem Datenarchiv der University of California entnommen ist. Er unterscheidet sich von allen übrigen verwendeten Datensätzen durch die sehr hohe Anzahl von verschiedenen Klassen in der Zielgröße.

Mit Hilfe von Variablen, die Beobachtungen an den Pflanzen selbst, sowie klimatische Rahmenbedingungen beschreiben, soll ein Modell gebildet werden, das die Art der Erkrankung von Sojabohnenpflanzen prognostiziert. Insgesamt 19 verschiedene Krankheiten kommen dabei in Frage.

Der Datensatz besteht ursprünglich aus 307 Fällen, die jedoch sehr viele fehlende Werte aufweisen. Reduziert man ihn auf die vollständigen Fälle, so sinkt der Stichprobenumfang auf 266. Auf diese Art fallen aber gerade Pflanzen mit bestimmten Erkrankungen weg, so daß sich auch die Anzahl der Kategorien in der Zielgröße von 19 auf 15 reduziert. Jede der verbliebenen Klassen weist dabei zwischen 10 und 40 Beobachtungen auf.

Zusammengefaßt liegt also im hier verwendeten dezimierten Datensatz eine Zielgröße mit 15 verschiedenen nominalen Klassen vor, die mit Hilfe von 35 nominalen Einflußgrößen prognostiziert werden soll.

Microarray-Datensätze

Die Verwendung von zwei Microarray-Datensätzen zum Vergleich der diversen Ensemble-Techniken erfolgt nicht zufällig. Vielmehr handelt es sich hier um ein besonders wichtiges und aktuelles Gebiet der Klassifikation, für das die Entwicklung von neuen Methoden eine große Rolle spielt. Die Besonderheit derartiger Datensätze besteht einheitlich darin, daß

anhand von wenigen Untersuchungseinheiten (typischerweise deutlich unter 100) eine Vielzahl von Variablen, nämlich die Expressionswerte verschiedener Gene (typischerweise bis zu 10000), bestimmt werden. Diese augenscheinliche Diskrepanz zwischen Stichprobenumfang und Variablenzahl, mit der viele herkömmliche statistische Verfahren große Schwierigkeiten haben, ist derzeit einer der Forschungsschwerpunkte im Feld der genetischen Statistik.

Leukemia Dieser Standard-Datensatz findet sich in verschiedenen Libraries des Programmpakets *R*. Er besteht aus den Expressionswerten von 7129 Genen von Leukämie-Patienten. Der Datensatz teilt sich auf in einen Lerndatensatz, bestehend aus 27 ALL- und 11 AML-Patienten, und einen Testdatensatz aus 20 ALL- und 14 AML-Patienten. Diese beiden Teile wurden zusammengesetzt und Filtertechniken für die Gene benutzt, so daß lediglich 3571 Gene als Kovariaten für die binäre Zielgröße übrig bleiben.

SRBCT Dieser Datensatz von Kahn, Wei, Ringner, Saal, Ladanyi, Westermann, Berthold, Schwab, Antonescu, Peterson & Meltzer (2001) enthält die Expressionswerte von 2308 Genen für 83 *Single-Round-Blue-Cells-Tumor*-Patienten aus vier verschiedenen Klassen (EWS, BL, NB, und RMS). Fünf weitere Proben stehen ebenfalls zur Verfügung, wurden aber hier weggelassen. Die Daten werden öffentlich unter
`http://www.thep.lu.se/pub/Preprints/01/lu_tp_01_06_supp.html`
zur Verfügung gestellt. Zur Vorbereitung wurden die bereits vorausgewählten Gene nur noch standardisiert. Die Klassenumfänge liegen zwischen 11 und 29.

Waveform-Datensatz

Der letzte Datensatz, der hier zum Vergleich herangezogen wird, ist gleichzeitig der einzige künstlich erzeugte. Der Algorithmus zur Generierung dieser simulierten Daten sowie ein größerer Beispieldatensatz sind ebenso wie der Großteil der hier verwendeten realen Datensätze dem Datenarchiv der University of California entnommen.

Als Basis für die Berechnung der Variablen dienen drei phasenverschobene Wellenverläufe. Jede Untersuchungseinheit des Datensatzes entsteht durch die Überlagerung von zwei dieser drei Wellen. Insgesamt gibt es drei verschiedene Kombinationsmöglichkeiten, die damit die drei möglichen Klassenzugehörigkeiten darstellen. Auf diese Art und Weise basiert also die Zielgröße Wellenkombination auf der zufälligen Auswahl von zwei der drei Wellen. Die Werte der unabhängigen Variablen sind dementsprechend die y-Koordinaten der jeweiligen sich überlagernden Wellen, gemessen an verschiedenen x-Positionen, wobei dazu noch weißes Rauschen addiert wird. Aus diesen verrauschten Koordinaten soll letztendlich mit Hilfe eines Modells vorhergesagt werden, welche Wellenkombination der Entstehung der Daten zugrunde liegt.

Zur Analyse wurde hier eine Stichprobe von 999 Fällen, von denen je etwa ein Drittel den drei Klassen zuzuordnen ist, aus dem vorliegenden Datensatz ausgewählt. Für jeden Fall

liegen 21 Kovariablen, die Meßstellen der Wellenüberlagerungen, vor, die Aufgrund der künstlichen Datenkonstruktion keine fehlenden Werte aufweisen.

Neben dem eben geschilderten Datensatz liegt noch eine zweite Variante vor, die das Auffinden eines geeigneten Prognosemodells noch künstlich erschweren soll: Zusätzlich zu den Meßwerten werden nämlich weitere Attribute verwendet, die überhaupt keine Information über die Klassenzugehörigkeit enthalten, da es sich um reine Zufallszahlen handelt. So soll geklärt werden, ob die Verfahren in der Lage sind, diese eigentlich sinnlosen Prädiktoren aus dem Modell auszuschließen. Insgesamt werden 19 solche Störvariablen in dieser zweiten Variante des Datensatzes verwendet.

6.1.3 Ergebnisse

Die folgenden Resultate zu den einzelnen Datensätzen werden jeweils in tabellarischer Form angegeben und kurz interpretiert. Alle dargestellten Ergebnisse, die auf Bäumen basieren, wurden anhand einer Hierarchietiefe von 3 (also maximal 8 Endknoten) bestimmt, sofern für spezielle Datensätze kein anderer Wert angegeben ist. Größere Bäume werden so weit als möglich vermieden, um kein Overfitting zu produzieren. In der Teststichprobenvariante wurde über die Resultate von je 50 Aufteilungen in Lern- und Validierungsstichprobe gemittelt. Ensemble-Techniken arbeiten jeweils mit 100 Zyklen.

Wisconsin-Breast-Cancer-Datensatz

Verfahren	Resubstitution	Test
CART	0.047	0.056
Bagging	0.025	0.040
Discrete AdaBoost	0.000	0.034
Real AdaBoost	0.000	0.034
Gentle AdaBoost	0.000	0.033
LogitBoost	0.016	0.036
L_2-Boost	0.022	0.039

Tabelle 6.1: Fehlklassifikationsraten bei den Cancer-Daten

Zunächst muß erwähnt werden, daß für Logit- und L_2-Boost Bäume der Hierarchietiefen 1, 2 und 3 ausprobiert wurden. Das jeweils beste Ergebnis (Tiefe 3 bei Logit- bzw. Tiefe 1 bei L_2-Boost) wird hier präsentiert.

Betrachtet man die Resultate bezüglich der CART-basierten Ensemble-Techniken (Tabelle 6.1), so sieht man, wie im Fall der Resubstitution das einfache CART-Ergebnis durch Bagging verbessert wird, wobei gilt, daß die Fehlklassifikationsraten tendenziell um so kleiner werden, je mehr Zyklen herangezogen werden (vgl. Abb. 6.2). Die meisten Boosting-

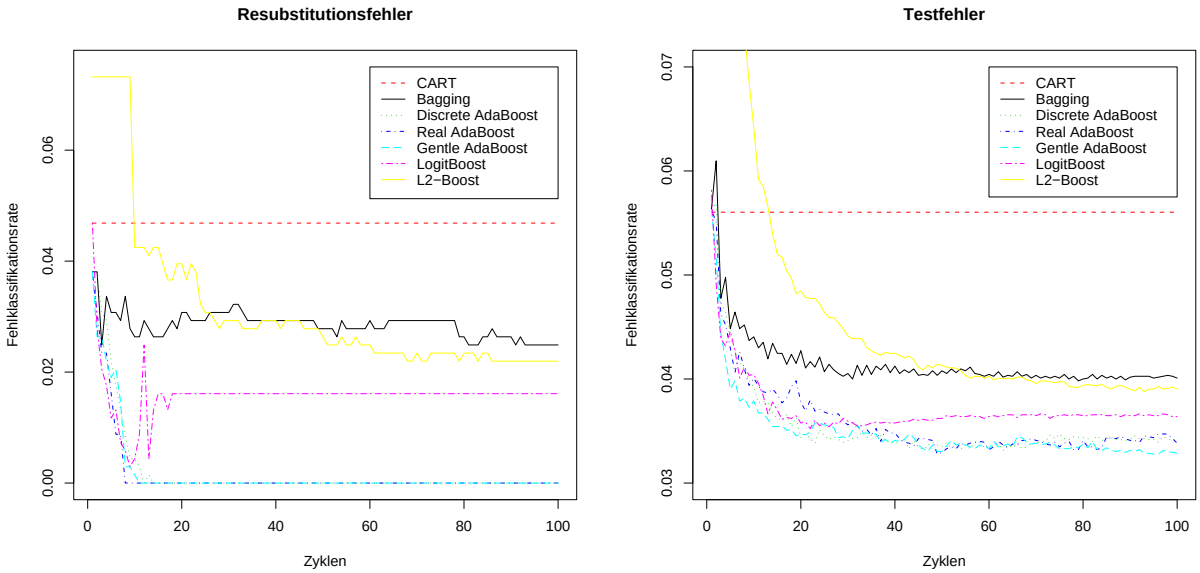

Abbildung 6.2: Entwicklung von Resubstitutions- und Testfehler bei den Cancer-Daten

Varianten pendeln sich dagegen bereits nach sehr wenigen Zyklen auf eine fehlerlose Klassifikation ein. Dies gilt nicht für Logit- und L_2-Boost, die hier etwas schlechter abschneiden.

Bei der Teststichprobe bleiben diese Beobachtungen im Großen und Ganzen erhalten. Lediglich der Fehler beim Boosting erreicht natürlich nicht die Null. Dennoch sind die Resultate stets besser als beim Bagging, das wiederum bereits zu einer deutlichen Verbesserung gegenüber CART führt. Zwischen den verschiedenen AdaBoost-Ansätzen lassen sich so gut wie keine Unterschiede feststellen, alle Varianten scheinen in etwa gleich gut zu arbeiten, während Logit- und L_2-Boost ebenso wie bei Resubstitution etwas abfallen.

Insgesamt kann man also festhalten, daß sowohl Bagging als auch Boosting die CART-Fehlerrate mit der Anzahl der verwendeten Zyklen immer weiter absenken und daß Boosting dabei die noch etwas besseren Resultate liefert.

Betrachtet man die Fehlklassifikationsraten bezüglich der Nächste-Nachbarn-Techniken (Tabelle 6.2), so kann man feststellen, daß sich bei diesem Datensatz wenig an den Fehlerraten ändert, egal welche Parameterkombination herangezogen wird. Das Klassifikationsproblem scheint hier zu stabil zu sein, als daß sich klare Unterschiede zwischen verschiedenen Verfahren ergeben könnten. Das konstante Niveau des Fehlers reicht aber durchaus an die besten Resultate von Boosting heran.

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	max. 25	rectangular	0.035	2	max. 25	rectangular	0.034
1	max. 25	triweight	0.035	2	max. 25	triweight	0.035
1	max. 25	all kernels	0.035	2	max. 25	all kernels	0.034

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	1	rectangular	0.035	2	1	rectangular	0.043
1	3	rectangular	0.032	2	3	rectangular	0.035
1	5	rectangular	0.033	2	5	rectangular	0.032
1	7	rectangular	0.034	2	7	rectangular	0.031
1	11	rectangular	0.034	2	11	rectangular	0.033
1	21	rectangular	0.039	2	21	rectangular	0.033
1	1	triweight	0.035	2	1	triweight	0.043
1	3	triweight	0.035	2	3	triweight	0.043
1	5	triweight	0.034	2	5	triweight	0.042
1	7	triweight	0.034	2	7	triweight	0.040
1	11	triweight	0.033	2	11	triweight	0.038
1	21	triweight	0.030	2	21	triweight	0.033

Tabelle 6.2: Fehlklassifikationsraten bei den Cancer-Daten (die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

Verfahren	Resubstitution	Test
CART	0.252	0.339
Bagging	0.229	0.304
Discrete AdaBoost	0.000	0.256
Real AdaBoost	0.000	0.253
Gentle AdaBoost	0.000	0.248

Tabelle 6.3: Fehlklassifikationsraten bei den Glass-Daten

Glass-Identification-Datensatz

Aufgrund der höheren Anzahl von verschiedenen Klassen wurden alle hier beschriebenen Ergebnisse mit Hilfe von Bäumen erzielt, für die eine maximale Hierarchietiefe von 4 zugelassen wurde. Dies erscheint für ein solches Mehrklassenproblem angemessen.

Die Ergebnisse der Ensemble-Techniken (Tabelle 6.3) entsprechen in etwa denjenigen beim Cancer-Datensatz, nur ist die Abstufung der Qualität der Verfahren zueinander noch deutlicher ausgeprägt (siehe auch Abbildung 6.3). Dementsprechend gilt sowohl unter Resubstitution als auch bei getrennten Teststichproben, daß Bagging und Boosting die CART-Resultate verbessern. Allerdings ist die durch die Boosting-Varianten erzielte Steigerung der Präzision noch gravierender. Hier zeigt sich besonders deutlich eine Überlegenheit von Boosting gegenüber Bagging.

Betrachtet man die *kk*NN-Ergebnisse (Tabelle 6.4), so scheint hier die Wahl eines kleinen k die geeignetste. Ohne Gewichte werden die Fehlerraten mit wachsendem k größer. Mit Gewichtung pendeln sich dagegen die Resultate für höhere k unabhängig vom Kern wieder auf die optimalen Ergebnisse mit nur einem nächsten Nachbarn ein. Der Vorteil dieses Ansatzes besteht also darin, daß sozusagen eine automatische Adjustierung von k vorgenommen wird: Ist k nämlich fälschlicherweise zu hoch gewählt, wird durch die Wahl der Gewichte der Einfluß der zu viel gewählten Nachbarn annulliert bzw. so stark gesenkt, daß sie keine Auswirkungen mehr auf die Prognose haben. Diese bei der Entwicklung des

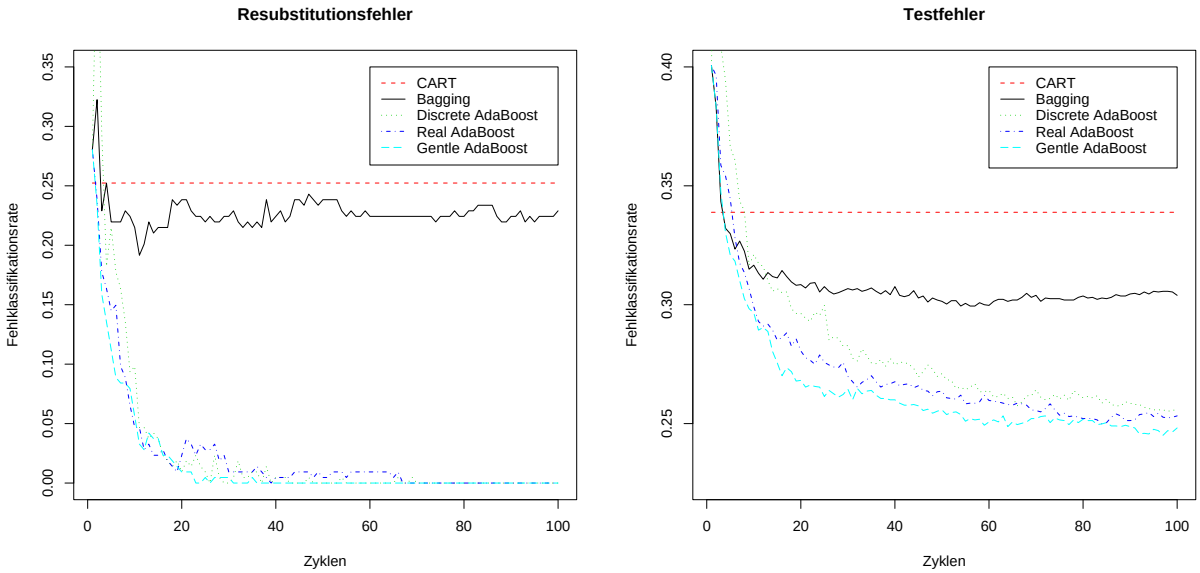

Abbildung 6.3: Entwicklung von Resubstitutions- und Testfehler bei den Glass-Daten

Verfahrens beabsichtigte Eigenschaft bestätigt sich also in der Empirie.

Die Fehlerraten streuen vom Niveau her deutlich, so daß schlechte Resultate (hohes k, keine Gewichtung) in der Nähe von CART liegen, während die besten kkNN-Ergebnisse etwas besser als Bagging, jedoch nicht so gut wie Boosting sind.

Ionosphere-Datensatz

Auch hier wurden für Logit- und L_2-Boost mehrere Hierarchietiefen ausprobiert, wobei in beiden Fällen Stumps zu den besten Resultaten führten, die auch in der Übersicht dargestellt sind.

Für die Resultate (Tabelle 6.5) gelten die gleichen Feststellungen wie bei den anfangs vorgestellten Cancer-Daten, die ja ebenfalls ein Zweiklassenproblem darstellen: Wiederum liefern sowohl Bagging als auch Boosting Verbesserungen gegenüber der CART-Prognose, wobei die verschiedenen AdaBoost-Varianten dabei deutlich besser als Bagging abschneiden, untereinander jedoch kaum zu unterscheiden sind. Ebenso wiederholt sich als Ergebnis die Dominanz der AdaBoost-Varianten gegenüber Logit- und L_2-Boost.

Die Ergebnisse von kkNN (Tabelle 6.6) weisen dagegen eher eine Ähnlichkeit zu denjenigen bezüglich der Glass-Daten auf. Wiederum führen kleine k zu den geringsten Fehlerraten, und mit Gewichtung pendeln sich die schlechten Resultate für höhere k wieder auf das optimale Niveau ein.

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	max. 25	rectangular	0.296	2	max. 25	rectangular	0.319
1	max. 25	triweight	0.277	2	max. 25	triweight	0.305
1	max. 25	all kernels	0.285	2	max. 25	all kernels	0.307

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	1	rectangular	0.276	2	1	rectangular	0.306
1	3	rectangular	0.303	2	3	rectangular	0.320
1	5	rectangular	0.327	2	5	rectangular	0.357
1	7	rectangular	0.341	2	7	rectangular	0.356
1	11	rectangular	0.339	2	11	rectangular	0.377
1	21	rectangular	0.379	2	21	rectangular	0.408
1	1	triweight	0.276	2	1	triweight	0.306
1	3	triweight	0.278	2	3	triweight	0.304
1	5	triweight	0.271	2	5	triweight	0.303
1	7	triweight	0.272	2	7	triweight	0.302
1	11	triweight	0.269	2	11	triweight	0.297
1	21	triweight	0.273	2	21	triweight	0.305

Tabelle 6.4: Fehlklassifikationsraten bei den Glass-Daten (die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

Verfahren	Resubstitution	Test
CART	0.085	0.121
Bagging	0.080	0.098
Discrete AdaBoost	0.000	0.073
Real AdaBoost	0.000	0.075
Gentle AdaBoost	0.000	0.072
LogitBoost	0.000	0.088
L_2-Boost	0.006	0.082

Tabelle 6.5: Fehlklassifikationsraten bei den Ionosphere-Daten

Bezüglich der Höhe der Fehlerraten ist kkNN hier aber nicht so erfolgreich. Während die besten Resultate lediglich an Bagging herankommen, schneidet das Verfahren bei einer ungünstigen Kombination von Parametern sogar schlechter als CART ab.

Bei den kkNN-Resultaten mit kreuzvalidierten Parametern kommt es zu der auf den ersten Blick verwunderlichen Situation, daß die Fehlerraten bei Variante 1 (nur Rechteckskern) und Variante 3 (alle Kerne) wesentlich besser sind als alle berechneten Einzelergebnisse ohne Kreuzvalidierung. Dies ist jedoch darauf zurückzuführen, daß beim gewöhnlichen Nächste-Nachbarn-Verfahren mit geradzahliger Anzahl von Nachbarn häufig Stimmengleichheiten auftreten, die dann pauschal in die jeweils niedrigere Klasse zugeteilt werden. Solche Ergebnisse sind aber mit Vorsicht zu genießen. In diesem Fall weisen mehrere Punkte den gleichen euklidischen Abstand zueinander auf, so daß speziell bei der Parameterkombination $k = 2$, $q = 2$ und Rechteckskern viele unentschiedene Abstimmungen auftreten, die hier nur zufällig in die richtige Klasse zugeordnet werden. Die Tatsache, daß lediglich diese besagte Parameterkonstellation problematisch ist, läßt sich auch deutlich aus Abbildung 6.5 erkennen.

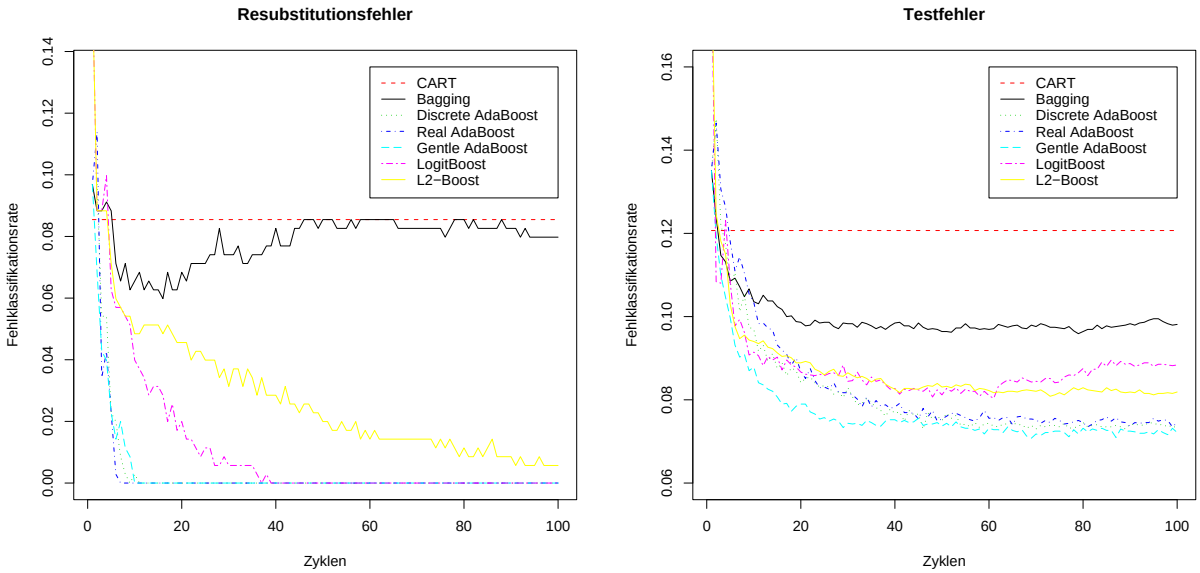

Abbildung 6.4: Entwicklung von Resubstitutions- und Testfehler bei den Ionosphere-Daten

Soybean-Datensatz

Zunächst muß darauf hingewiesen werden, daß bei diesem Datensatz eine Veränderung an den Standardeinstellungen der CART-Algorithmen vorgenommen werden mußte. Da 8 Endknoten (Hierarchietiefe 3) selbst im optimalen Fall nicht ausreichen können, um eine Zielgröße mit 15 Klassen korrekt zu klassifizieren, werden hier ausnahmsweise 16 (Hierarchietiefe 4) Endknoten zugelassen.

Betrachtet man die Ergebnisse in Tabelle 6.7 bei Resubstitution, so verhalten sich die Resultate so, wie es zu erwarten war: Die CART-Fehlerrate wird bereits durch Bagging deutlich verbessert, während Boosting sie sogar sehr schnell auf ein Niveau nahe Null senkt (siehe auch Abbildung 6.6).

Bei der Validierung bestätigt sich diese Tendenz aus der Resubstitution deutlicher als bei jedem anderen der hier betrachteten Datensätze. Die Fehlklassifikationsrate von CART sinkt durch die Anwendung der Boosting-Varianten auf ein Viertel des Ausgangswertes ab, während Bagging zu vergleichsweise geringen Verbesserungen führt. Außerdem kann man eine Tendenz erahnen, daß die beiden Boosting-Ansätze, die mit den Klassifikationswahrscheinlichkeiten statt der unflexiblen Punktprognose arbeiten, zumindest geringfügig besser abschneiden.

Als Grund für die besonders positiven Effekte der Ensemble-Techniken kann man vermuten, daß die Unterschiede zwischen den Verfahren umso deutlicher zutage treten, je schwerer das Prognoseproblem ist. Da Mehrklassenprobleme sicher die deutlich anspruchsvollere Aufga-

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	max. 25	rectangular	0.094	2	max. 25	rectangular	0.106
1	max. 25	triweight	0.099	2	max. 25	triweight	0.130
1	max. 25	all kernels	0.094	2	max. 25	all kernels	0.107

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	1	rectangular	0.096	2	1	rectangular	0.135
1	3	rectangular	0.111	2	3	rectangular	0.156
1	5	rectangular	0.119	2	5	rectangular	0.162
1	7	rectangular	0.125	2	7	rectangular	0.169
1	11	rectangular	0.143	2	11	rectangular	0.178
1	21	rectangular	0.169	2	21	rectangular	0.228
1	1	triweight	0.096	2	1	triweight	0.135
1	3	triweight	0.098	2	3	triweight	0.134
1	5	triweight	0.099	2	5	triweight	0.133
1	7	triweight	0.098	2	7	triweight	0.129
1	11	triweight	0.100	2	11	triweight	0.126
1	21	triweight	0.099	2	21	triweight	0.128

Tabelle 6.6: Fehlklassifikationsraten bei den Ionosphere-Daten (die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

Verfahren	Resubstitution	Test
CART	0.365	0.428
Bagging	0.169	0.320
Discrete AdaBoost	0.030	0.128
Real AdaBoost	0.038	0.112
Gentle AdaBoost	0.023	0.115

Tabelle 6.7: Fehlklassifikationsraten bei den Soybean-Daten

be darstellen, sind besonders hier, aber auch bei den Glass-Daten die augenscheinlichsten Verbesserungen zu erkennen.

Bezüglich der Nächste-Nachbarn-Techniken scheint wiederum eine möglichst geringe Wahl von k zur geringsten Fehlerrate zu führen (siehe Tabelle 6.8). Außerdem zeigt sich erneut der positive Effekt der Gewichtung im Fall von zu hoch gewähltem k, die automatische Adjustierung greift also auch bei diesen Daten.

Im Vergleich zu den Ensemble-Techniken schneidet kkNN auch sehr gut ab: Lediglich die besten Boosting-Algorithmen erreichen die optimale Fehlerrate der Nächste-Nachbarn-Ansätze.

Microarray-Datensätze

Bei den Microarray-Daten soll der Schwerpunkt auf einem umfangreichen Vergleich der CART-basierten Ensemble-Methoden sowie des modifizierten Nächste-Nachbarn-Ansatzes mit verschiedensten anderen Klassifikationstechniken liegen. Deshalb wurde eine Reihe von weiteren klassischen und modernen Verfahren herangezogen. Alle Klassifikationsbäume arbeiten mit einer Hierarchietiefe von 2 (Leukemia) bzw. 4 (SRBCT). Als Kern für die gewichteten Nächste-Nachbarn-Techniken wurde die einfachste Variante, nämlich der Dreieckskern, sowie die euklidische Distanz ($q = 2$) gewählt. Die weiteren hier beschriebenen

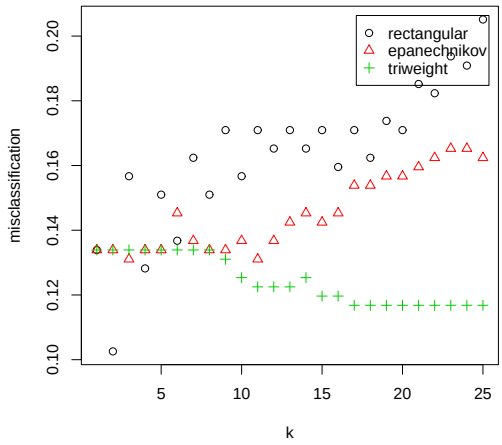

Abbildung 6.5: Typisches Beispiel für den kreuzvalidierten Testfehler bei den Ionosphere-Daten (eine zufällige Aufteilung); Man sieht deutlich das Minimum bei $k = 2$ und dem Rechteckskern.

Ergebnisse wurden bereits in Boulesteix (2004) veröffentlicht.

Zunächst wurde eine neuere Methode namens *Prediction Analysis of Microarray* (kurz: *PAM*) angewendet, die speziell für hochdimensionale Microarray-Daten konzipiert wurde (Tibshirani, Hastie, Narasimhan & Chu (2002)) und ohne eine vorherige Variablenselektion arbeitet. PAM basiert auf kupierten (engl. *shrunken*) Zentroiden und benötigt die Wahl eines Shrinkage-Parameters δ. Die Anzahl der Gene, die zur Berechnung der Zentroide herangezogen wird, hängt von diesem Parameter ab. Eine mögliche Wahl ist $\delta = 0$: Es gehen alle Gene in die Berechnung ein. Zusätzlich wird jedoch ein auf Kreuzvalidierung basierendes Auswahlverfahren zur Bestimmung eines optimalen Wertes für δ angeboten, das hier ebenfalls ausprobiert wird. Die PAM-Methode liegt als *R*-Paket *pamr* implementiert vor.

Desweiteren wird ein vielversprechendes Verfahren namens *Partial Least Squares* (kurz: *PLS*) angewendet. Neue Komponenten werden anhand einer PLS-Dimensionsreduktion ermittelt und mit diesen Komponenten als Kovariaten wird anschließend eine lineare Diskriminanzanalyse durchgeführt (Nguyen & Rocke (2002)).

Zusätzlich kommt die moderne Technik der *Support Vector Machines* (*SVM*, siehe Furey, Cristianini, Duffy, Bednarski, Schummer & Haussler (2000)) zum Einsatz, ebenso wie die klassische lineare Diskriminanzanalyse (*LDA*). Für Support Vector Machines wurde die Implementierung des *R*-Pakets *e1071* herangezogen.

Für die meisten dieser Techniken benötigt man eine Variablen- oder Genselektionsmethode, da die Behandlung von mehr als 2000 Variablen nur schwer möglich ist. Hierzu wird eine

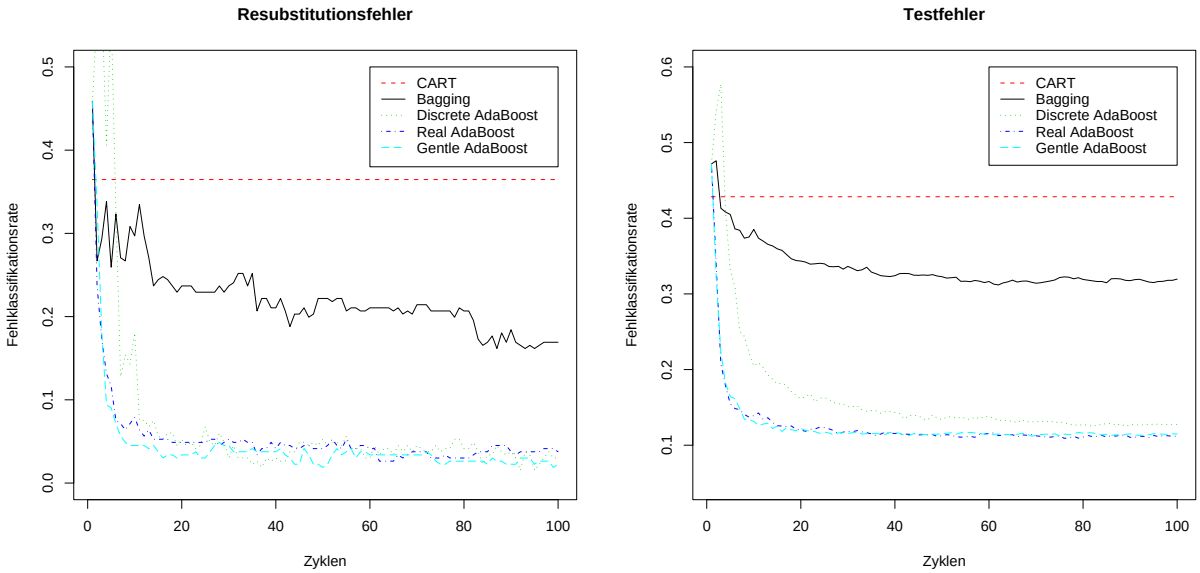

Abbildung 6.6: Entwicklung von Resubstitutions- und Testfehler bei den Soybean-Daten

Rangliste der Gene anhand der $\frac{BSS}{WSS}$-Statistik gebildet. Sie lautet für Gen s wie folgt:

$$\frac{\text{BSS}_s}{\text{WSS}_s} = \frac{\sum_{r=1}^{c} \sum_{i:Y_i=r} (\hat{\mu}_{rs} - \hat{\mu}_s)^2}{\sum_{r=1}^{c} \sum_{i:Y_i=r} (x_{is} - \hat{\mu}_{rs})^2}$$

BSS bedeutet dabei *between groups sum of squares*, WSS *within groups sum of squares*. Desweiteren beschreibt $\hat{\mu}_s$ das Stichprobenmittel von x_s, während $\hat{\mu}_{rs}$ das Stichprobenmittel von x_s innerhalb der Klasse r darstellt. Diese Art der Variablenselektion wird separat für jeden der 50 zufällig erzeugten Lerndatensätze durchgeführt.

Die Vergleichsübersicht in Tabelle 6.9 zeigt, daß CART und sämtliche darauf aufbauenden Ensemble-Techniken nicht besonders gut abschneiden. Immerhin führen Bagging und Boosting aber zu deutlichen Verbesserungen gegenüber einem einzelnen Klassifikationsbaum. LDA und PAMR führen bei SRBCT zu etwas geringeren Fehlerraten und SVM reduziert die Fehlklassifikation immerhin bereits unter 2 %. Bei Leukemia verhält es sich ähnlich, lediglich LDA liefert hier deutlich schlechtere Ergebnisse. PLS mit wenigen Komponenten schneidet bei SRBCT extrem schlecht ab. Mit wachsender Anzahl von Komponenten sinkt die Fehlerrate aber dramatisch bis auf unter 1 %. Die Anzahl der Komponenten spielt bei Leukemia dagegen kaum eine Rolle, da die Resultate von Anfang an sehr gut sind. Die Nächste-Nachbarn-Techniken liefern bei SRBCT schließlich die besten Resultate und speziell kkNN mit Verwendung von Gewichten weist die geringste Fehlerrate aller verglichenen Verfahren auf. So deutlich ist die Überlegenheit bei Leukemia nicht, die entsprechenden Fehlerraten zählen dennoch mit zu den geringsten aller Verfahren. Dies ist ein wichtiges Ergebnis, da speziell Microarray-Analysen in der modernen Biostatistik immer

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	max. 25	rectangular	0.127	2	max. 25	rectangular	0.139
1	max. 25	triweight	0.119	2	max. 25	triweight	0.124
1	max. 25	all kernels	0.122	2	max. 25	all kernels	0.125

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	1	rectangular	0.122	2	1	rectangular	0.133
1	3	rectangular	0.164	2	3	rectangular	0.172
1	5	rectangular	0.195	2	5	rectangular	0.197
1	7	rectangular	0.238	2	7	rectangular	0.232
1	11	rectangular	0.308	2	11	rectangular	0.310
1	21	rectangular	0.426	2	21	rectangular	0.437
1	1	triweight	0.122	2	1	triweight	0.133
1	3	triweight	0.121	2	3	triweight	0.130
1	5	triweight	0.119	2	5	triweight	0.126
1	7	triweight	0.118	2	7	triweight	0.124
1	11	triweight	0.117	2	11	triweight	0.120
1	21	triweight	0.132	2	21	triweight	0.124

Tabelle 6.8: Fehlklassifikationsraten bei den Soybean-Daten (die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

mehr an Bedeutung gewinnen. Hier erscheinen die Nächste-Nachbarn-Verfahren also sehr vielversprechend, baumbasierte Verfahren dagegen eher ungeeignet.

Waveform-Datensatz

Betrachtet man die CART-basierten Ergebnisse in Tabelle 6.10, so fällt zunächst auf, daß zwischen den Resultaten mit Störgrößen und denjenigen ohne Störgrößen kaum Unterschiede bestehen. Die uninformativen Variablen werden als solche also scheinbar erkannt und nicht oder kaum in die erzielten Prognosemodelle mit aufgenommen.

Ansonsten erhält man wieder die Resultate, die zu erwarten waren: Boosting-Varianten liefern die besseren Fehlklassifikationsraten als Bagging, beide führen aber gegenüber dem einfachen CART-Modell zu deutlichen Verbesserungen. Diese Aussagen gelten sowohl für die Resubstitution als auch für getrennte Lern- und Teststichproben.

Anders verhält es sich hier bei den Nächste-Nachbarn-Ansätzen (Tabelle 6.11): Sind Störgrößen vorhanden, so wird die Prognose durchgehend schlechter. Dieses Phänomen ist aber durchaus zu erwarten, da die Abstandsberechnungen alle Variablen mit einbeziehen, egal wie stark ihr Einfluß auf die Zielgröße ist. Ohne eine vorherige Variablenselektion sollte kkNN also nicht angewendet werden, was aber für die meisten Klassifikations- und Regressionsverfahren gilt und damit keinen besonderen Nachteil dieses Ansatzes darstellt.

Außerdem greift hier die Gewichtung zumindest bei den untersuchten Parametereinstellungen noch nicht, da mit zunehmendem k die Fehlerrate bereits ohne Gewichtung sinkt. Eine Korrektur von k nach unten ist in diesem Wertebereich des Parameters also nicht nötig und führt deshalb auch eher zu Verschlechterungen in der Vorhersage. Die Versuche zeigen, daß die Vorzüge des Gewichtungsansatzes erst bei einer sehr hohen Wahl von k (größer als 50) auftreten. Das Potenzial der Gewichtung kann also bei diesen beiden Datensätzen nicht voll ausgeschöpft werden. Deutlich zeigen sich dagegen wieder die Eigenschaften der beiden Kerne Epanechnikov und Triweight: Während Triweight mit wachsendem k eher langsam

	Leukemia	SRBCT
Verfahren	Testfehlerrate	Testfehlerrate
PAMR ($\delta = 0$)	0.030	0.066
PAMR (optimales δ)	0.052	0.024
LDA (10 Gene)	0.052	0.085
LDA (20 Gene)	0.064	0.041
SVM (50 Gene)	0.033	0.019
SVM (100 Gene)	0.028	0.013
1 PLS	0.025	0.362
3 PLS	0.034	0.052
5 PLS	0.032	0.008
CART (100 Gene)	0.099	0.177
Bagging	0.075	0.070
Discrete AdaBoost	0.047	0.069
Gentle AdaBoost	0.049	0.065
NN (100 Gene)	0.033	0.009
5NN (100 Gene)	0.031	0.013
k5NN (100 Gene)	0.033	0.006

Tabelle 6.9: Fehlklassifikationsraten bei den Microarray-Daten

fallende Fehlerraten aufweist, folgt Epanechnikov sehr nahe an der Rechteckskern-Variante und überholt diese für sehr hohe k-Werte dann auch.

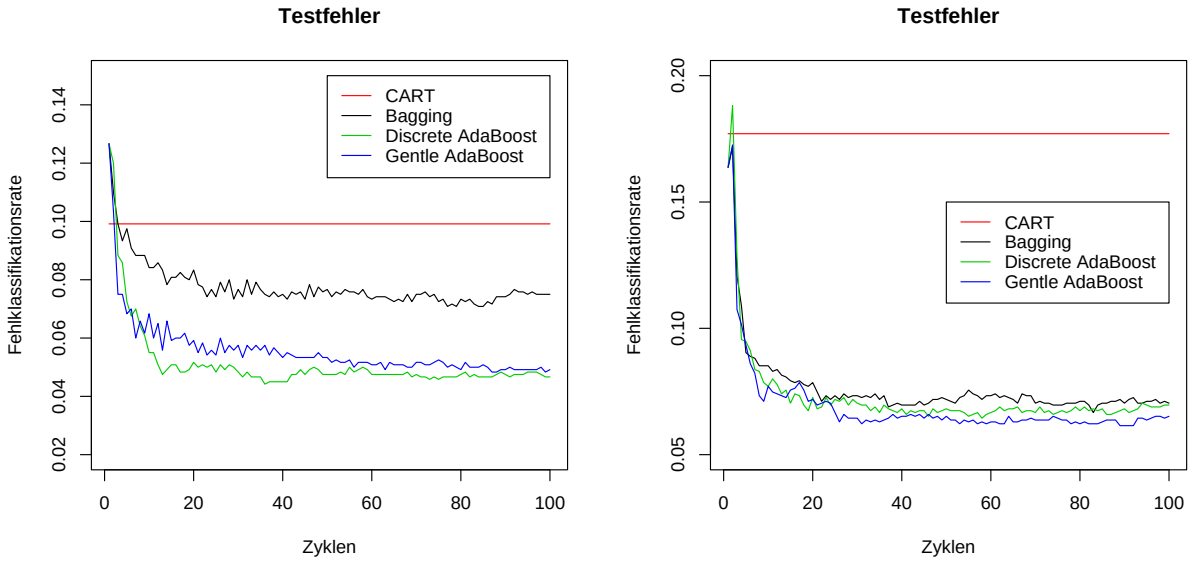

Abbildung 6.7: Entwicklung des Testfehlers bei den Microarray-Daten (links Leukemia, rechts SRBCT)

Datensatz	Verfahren	Resubstitution	Test
ohne Störgrößen	CART	0.231	0.290
	Bagging	0.141	0.195
	Discrete AdaBoost	0.000	0.164
	Real AdaBoost	0.061	0.155
	Gentle AdaBoost	0.002	0.158
mit Störgrößen	CART	0.230	0.271
	Bagging	0.158	0.206
	Discrete AdaBoost	0.000	0.167
	Real AdaBoost	0.047	0.163
	Gentle AdaBoost	0.000	0.165

Tabelle 6.10: Fehlklassifikationsraten bei den Waveform-Daten

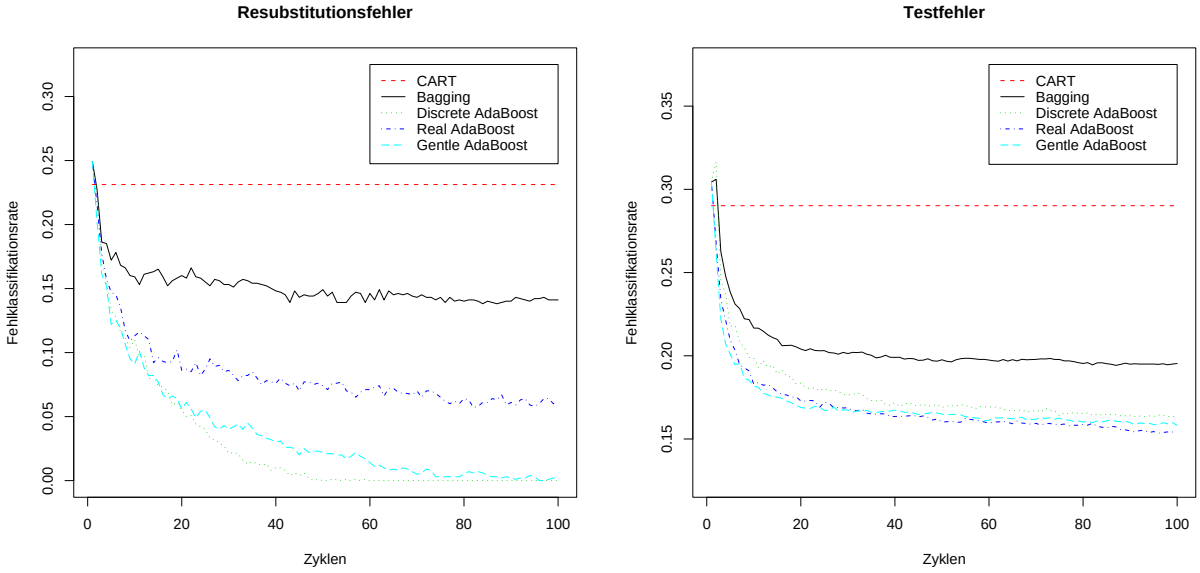

Abbildung 6.8: Entwicklung von Resubstitutions- und Testfehler bei den Waveform-Daten ohne Störgrößen

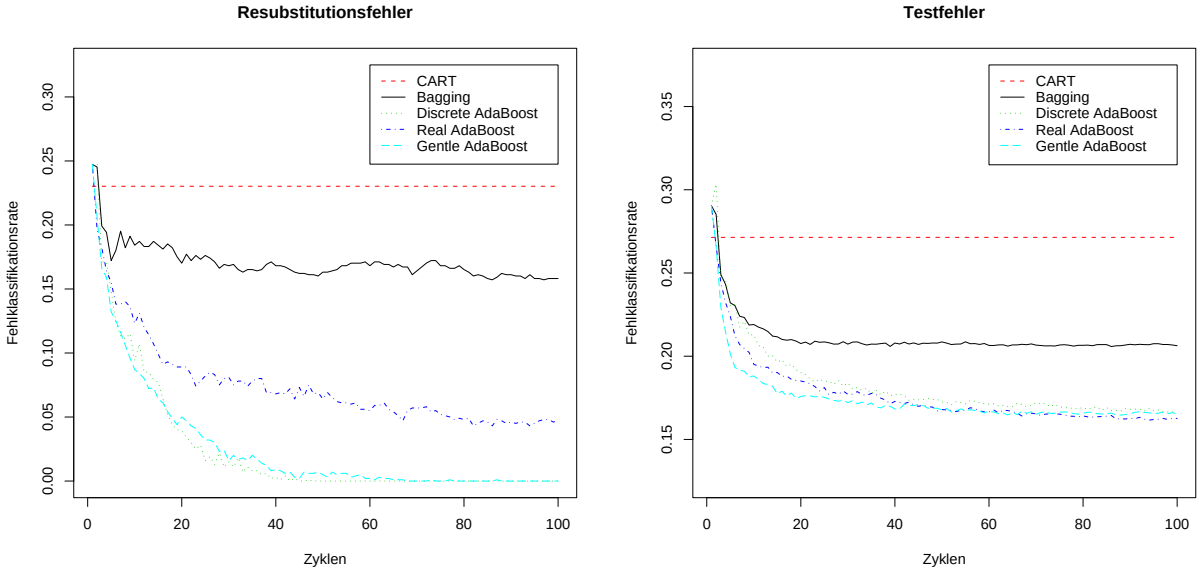

Abbildung 6.9: Entwicklung von Resubstitutions- und Testfehler bei den Waveform-Daten mit Störgrößen

	q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
ohne Störgrößen	1	max. 25	rectangular	0.175	2	max. 25	rectangular	0.172
	1	max. 25	triweight	0.205	2	max. 25	triweight	0.202
	1	max. 25	all kernels	0.173	2	max. 25	all kernels	0.176
mit Störgrößen	1	max. 25	rectangular	0.182	2	max. 25	rectangular	0.213
	1	max. 25	triweight	0.223	2	max. 25	triweight	0.239
	1	max. 25	all kernels	0.184	2	max. 25	all kernels	0.210

	q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
ohne Störgrößen	1	1	rectangular	0.248	2	1	rectangular	0.242
	1	3	rectangular	0.208	2	3	rectangular	0.214
	1	5	rectangular	0.202	2	5	rectangular	0.201
	1	7	rectangular	0.195	2	7	rectangular	0.190
	1	11	rectangular	0.184	2	11	rectangular	0.184
	1	21	rectangular	0.173	2	21	rectangular	0.170
	1	31	rectangular	0.164	2	31	rectangular	0.166
	1	41	rectangular	0.161	2	41	rectangular	0.163
	1	51	rectangular	0.162	2	51	rectangular	0.162
	1	101	rectangular	0.165	2	101	rectangular	0.162
	1	1	epanechnikov	0.248	2	1	epanechnikov	0.242
	1	3	epanechnikov	0.232	2	3	epanechnikov	0.234
	1	5	epanechnikov	0.215	2	5	epanechnikov	0.215
	1	7	epanechnikov	0.207	2	7	epanechnikov	0.205
	1	11	epanechnikov	0.196	2	11	epanechnikov	0.195
	1	21	epanechnikov	0.181	2	21	epanechnikov	0.181
	1	31	epanechnikov	0.175	2	31	epanechnikov	0.173
	1	41	epanechnikov	0.170	2	41	epanechnikov	0.168
	1	51	epanechnikov	0.165	2	51	epanechnikov	0.165
	1	101	epanechnikov	0.161	2	101	epanechnikov	0.161
	1	1	triweight	0.248	2	1	triweight	0.242
	1	3	triweight	0.245	2	3	triweight	0.239
	1	5	triweight	0.239	2	5	triweight	0.236
	1	7	triweight	0.233	2	7	triweight	0.233
	1	11	triweight	0.224	2	11	triweight	0.223
	1	21	triweight	0.208	2	21	triweight	0.205
	1	31	triweight	0.198	2	31	triweight	0.197
	1	41	triweight	0.191	2	41	triweight	0.191
	1	51	triweight	0.187	2	51	triweight	0.187
	1	101	triweight	0.172	2	101	triweight	0.171
mit Störgrößen	1	1	rectangular	0.280	2	1	rectangular	0.290
	1	3	rectangular	0.232	2	3	rectangular	0.254
	1	5	rectangular	0.220	2	5	rectangular	0.238
	1	7	rectangular	0.212	2	7	rectangular	0.229
	1	11	rectangular	0.199	2	11	rectangular	0.222
	1	21	rectangular	0.186	2	21	rectangular	0.212
	1	31	rectangular	0.178	2	31	rectangular	0.199
	1	41	rectangular	0.181	2	41	rectangular	0.186
	1	51	rectangular	0.175	2	51	rectangular	0.182
	1	101	rectangular	0.173	2	101	rectangular	0.182
	1	1	epanechnikov	0.280	2	1	epanechnikov	0.290
	1	3	epanechnikov	0.262	2	3	epanechnikov	0.275
	1	5	epanechnikov	0.236	2	5	epanechnikov	0.255
	1	7	epanechnikov	0.221	2	7	epanechnikov	0.242
	1	11	epanechnikov	0.209	2	11	epanechnikov	0.226
	1	21	epanechnikov	0.193	2	21	epanechnikov	0.214
	1	31	epanechnikov	0.187	2	31	epanechnikov	0.206
	1	41	epanechnikov	0.181	2	41	epanechnikov	0.198
	1	51	epanechnikov	0.178	2	51	epanechnikov	0.193
	1	101	epanechnikov	0.171	2	101	epanechnikov	0.181
	1	1	triweight	0.280	2	1	triweight	0.290
	1	3	triweight	0.278	2	3	triweight	0.285
	1	5	triweight	0.270	2	5	triweight	0.278
	1	7	triweight	0.260	2	7	triweight	0.271
	1	11	triweight	0.247	2	11	triweight	0.262
	1	21	triweight	0.226	2	21	triweight	0.245
	1	31	triweight	0.214	2	31	triweight	0.232
	1	41	triweight	0.205	2	41	triweight	0.223
	1	51	triweight	0.198	2	51	triweight	0.217
	1	101	triweight	0.184	2	101	triweight	0.202

Tabelle 6.11: Fehlklassifikationsraten bei den Waveform-Daten (die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

6.2 Empirische Vergleiche bei ordinaler Zielgröße

6.2.1 Studiendesign

In diesem Abschnitt werden die verschiedenen ordinalen Ansätze mit einfacheren Alternativmethoden, die entweder nicht die ordinale Information in den Daten nutzen oder keine Aggregationstechniken verwenden, verglichen. Die einfachste solche Variante ist ein gewöhnlicher Klassifikationsbaum, der im folgenden als nominaler CART bezeichnet wird. Hier wird ein Baum mit Hilfe des Devianz-Kriteriums gebildet, wobei das Wachstum bis zu einer vom Benutzer festgelegten maximalen Tiefe durchgeführt wird, die von Datensatz zu Datensatz variieren kann. Ein Ansatz, der zwar die Ordnung der Klassen in Betracht zieht, aber immer noch auf Bagging oder Boosting verzichtet, ist es, einen Baum für jede Dichotomisierung getrennt zu entwickeln und diese Einzelergebnisse dann gemäß eines Fixed-Split-Verfahrens zu aggregieren. Diese Methode wird als Fixed Split CART bezeichnet. Auf die selbe Art und Weise können zwei Bagging-Varianten herangezogen werden: Ein nominaler Ansatz, bei dem jeder Baum die als nominal angesehene mehrklassige Zielvariable prognostiziert und das endgültige Ergebnis aus einer Mehrheitsabstimmung dieser Vorhersagen resultiert, und Fixed Split Bagging, bei dem Bagging für alle festen Aufteilungen herangezogen wird. Diese Ergebnisse werden dann sowohl über die Dichotomisierungen als auch über die Bagging-Zyklen hinweg aggregiert. Unter den hier geschilderten Ansätzen sind die nominalen Techniken als Baseline für das Auffinden möglicher Verbesserungen durch ordinales Bagging oder Boosting zu verstehen.

Desweiteren betrachten wir eine Reihe von verschiedenen Boosting-Versionen: Die einfachen nominalen Mehrklassen-Varianten von Discrete, Real und Gentle AdaBoost, die die ordinale Struktur in den Daten nicht nutzen, werden wiederum lediglich zum Vergleich mit neuen Techniken herangezogen. Diese neuen Methoden sind alle beschriebenen ordinalen Boosting-Techniken: Einerseits unterscheiden wir zwischen den verschiedenen Boosting-Algorithmen, andererseits zwischen zwei Arten der Aggregation. Fixed Split bedeutet dabei, daß jede einzelne Dichotomisierung getrennt der Boosting-Prozedur unterzogen wird und die Kombination dieser Ergebnisse erst am Ende stattfindet. Ordinal AdaBoost, das die Ergebnisse bezüglich der einzelnen Dichotomisierungen bereits in jedem Boosting-Zyklus kombiniert, arbeitet dagegen mit standardisierten Distanzen als Gewichtungskriterium. In jedem Fall können kumulative und sequenzielle Aufteilungen der Zielgröße betrachtet werden. Zusätzlich werden noch die Resultate des ordinalen L_2-Boost präsentiert, der ein Beispiel für ein regressionsbasiertes Verfahren darstellt.

Letztendlich können im ordinalen Fall so insgesamt 24 verschiedene Ensemble-Methoden miteinander verglichen werden:

Verfahren
nominaler CART
nominales Bagging
nominales Discrete AdaBoost
nominales Real AdaBoost
nominales Gentle AdaBoost
kumulativer Fixed Split CART
kumulatives Fixed Split Bagging
kumulatives Fixed Split Discrete AdaBoost
kumulatives Fixed Split Real AdaBoost
kumulatives Fixed Split Gentle AdaBoost
kumulatives Fixed Split LogitBoost
kumulatives Fixed Split L_2-Boost
kumulatives Ordinal Discrete AdaBoost
kumulatives Ordinal Real AdaBoost
kumulatives Ordinal Gentle AdaBoost
sequenzieller Fixed Split CART
sequenzielles Fixed Split Bagging
sequenzielles Fixed Split Discrete AdaBoost
sequenzielles Fixed Split Real AdaBoost
sequenzielles Fixed Split Gentle AdaBoost
sequenzielles Fixed Split LogitBoost
sequenzielles Fixed Split L_2-Boost
sequenzielles Ordinal Discrete AdaBoost
Ordinal L_2-Boost

Bei allen aggregierten Klassifikationsverfahren muß der Anwender die Anzahl der Zyklen wählen, also die Anzahl der verschiedenen Klassifikatoren, die bei einer Anwendung von Bagging oder Boosting kombiniert werden sollen. In dieser Studie verwenden wir als Standard eine Anzahl von 100 Zyklen. Der zweite Parameter, der dann noch zu wählen ist, ist die konstante Hierarchietiefe aller beteiligten Bäume, also indirekt die maximale Anzahl der Endknoten. Die von uns gewählten Werte dieses Parameters variieren von Datensatz zu Datensatz und werden bei den Ergebnissen mit angegeben.

Damit sind die Parameter, die bei den einzelnen Verfahren variiert werden können, dieselben, die bereits bei der Untersuchung der gewöhnlichen Voting-Verfahren für nominale Größen erwähnt werden, denn die ordinalen Ansätze benötigen keine zusätzlichen Parameter, so daß jegliche Manipulationsmöglichkeit auf die eingesetzten Bagging- und Boosting-Methoden zurückzuführen ist.

Neben diesen Voting-Verfahren sollen auch wieder die Nächste-Nachbarn-Techniken eingesetzt werden. Jede Parametereinstellung kann dabei sowohl als nominale als auch als ordinale Variante eingesetzt werden, je nachdem, ob man den Modus oder den Median der Klassenverteilung betrachtet. Bei der ordinalen Variante kann zusätzlich ein y-Kern herangezogen werden.

Auch bei den Nächste-Nachbarn-Verfahren treten im ordinalen Fall zunächst keine zusätzlichen Parameter auf, so daß es bei der Variation von k, q und der Art des Kernes bleibt. Verwendet man jedoch zusätzlich noch einen y-Kern, so muß dessen Breite, die angibt, wieviele Nachbarklassen der Prognose zusätzlich Scores erhalten sollen, ebenfalls gewählt werden. Hier wird pauschal mit einem Wert von 2 gearbeitet, das heißt, je zwei Klassen nach oben und unten werden mit berücksichtigt.

Arbeitet man mit kreuzvalidierten Parameterwerten, so werden wieder die gleichen Einschränkungen wie im nominalen Fall getroffen, also k zwischen 1 und 25 und die besagten drei Varianten für die Menge der zulässigen Kernfunktionen. Wichtig ist, hierbei darauf hinzuweisen, daß das per Kreuzvalidierung optimierte Gütekriterium bei ordinalen Daten nicht die Fehlklassifikationsrate ist. Stattdessen wird hier der absolute Abstand zwischen

wahrer und vorhergesagter Klasse genutzt, bei metrischen Größen dagegen der quadrierte Abstand.

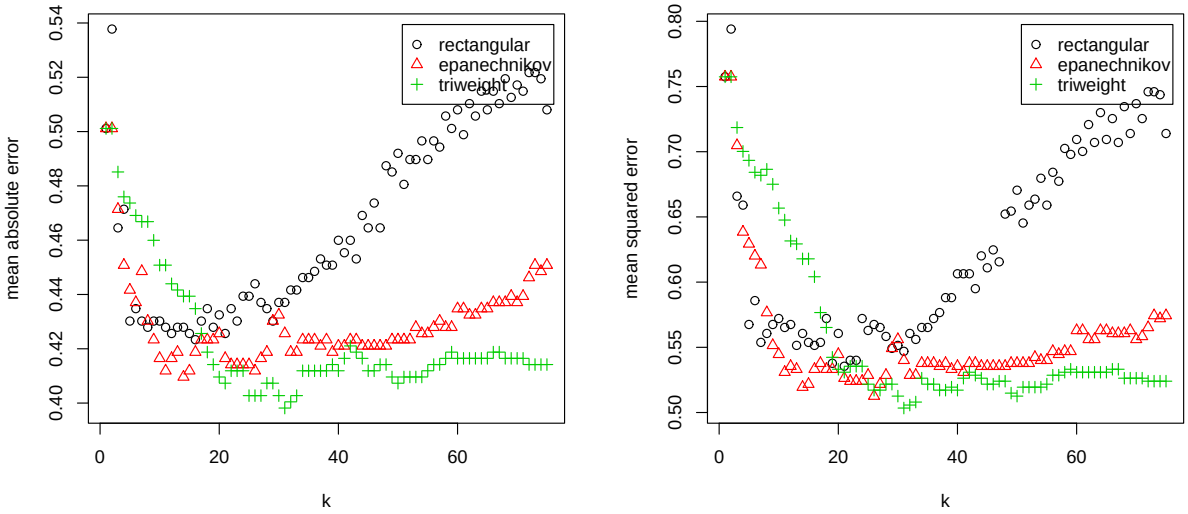

Abbildung 6.10: Typisches Beispiel (eine zufällige Aufteilung der Herzerkrankungs-Daten) zum Vergleich der Fehlerraten bei Verwendung von Epanechnikov- und Triweight-Kern (links Betragsabstände, rechts euklidische Abstände zwischen wahrer und prognostizierter Klasse)

Die Wahl, welche der möglichen Kernfunktionen bei der zweiten Kreuzvalidierungs-Variante verwendet wird, basiert auf den selben empirischen Erfahrungen, die bereits im nominalen Teil der Vergleichstudien geschildert wurden. Da das ideale k der gewöhnlichen Nächste-Nachbarn-Methode ohne Kernfunktionen bei den meisten der hier untersuchten ordinalen Datensätze allerdings sehr hoch ist, kann der Epanechnikov-Kern gewählt werden, der etwas zuverlässiger zu niedrigen Fehlerraten führt, wenn man mit der Wahl von k nicht zu weit über dem idealen k für den Rechtecks-Kern liegt (siehe Abbildung 6.10 für $k < 20$).

Desweiteren könnte in einer Kreuzvalidierung für ordinale Zielgrößen zusätzlich noch die Breite eines y-Kerns mit einbezogen werden. Dies wurde auch versucht, erwies sich aber als wenig erfolgreich, da eine derart komplexe Kreuzvalidierung in drei Dimensionen sehr instabil wird. Stattdessen soll der y-Kern, ebenso wie das gesamte kkNN-Verfahren, möglichst einfach gehalten werden, indem stets eine Breite von 2 (Scores für zwei Nachbarklassen nach oben und nach unten) für den Dreiecks-y-Kern verwendet wird.

Die Evaluation sämtlicher Methoden kann auf verschiedenen Präzisionsmaßen basieren. Als

ein rohes Kriterium für die Genauigkeit der Vorhersage wird die Fehlklassifikationsrate

$$\frac{1}{n}\sum_{i=1}^{n} I(Y_i \neq \hat{Y}_i)$$

herangezogen. Im hier interessierenden Fall von ordinaler Klassenstruktur sollten Kriterien allerdings in der Lage sein, explizit auf die ordinale Struktur der Klassen einzugehen, denn natürlich ist beispielsweise eine Fehlklassifikation in eine Nachbarklasse der wahren Zugehörigkeit weit weniger gravierend als ein größerer Abstand zwischen wahrem Wert und Prognose. Daher werden noch zwei weitere Maße herangezogen, die solche Abstände auf verschiedene Art bestrafen: Der mittlere absolute Wert der Differenzen

$$\frac{1}{n}\sum_{i=1}^{n} |Y_i - \hat{Y}_i|$$

und der mittlere quadrierte Abstand

$$\frac{1}{n}\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \quad ,$$

der große Unterschiede sogar noch härter bestraft. Starke Abweichungen von der ordinalen Struktur sollen so erkannt werden.

Eine Alternative stellt die Berechnung von Korrelationen zwischen den wahren und den durch das Modell prognostizierten Variablenausprägungen dar. Das im Fall von ordinalen Daten geeignete Maß ist hier der Korrelationskoeffizient von Spearman, der speziell zur Bestimmung des Zusammenhangs von rangskalierten Größen entwickelt wurde. Da sich die Unterschiede dieses Maßes für verschiedene Modelle als zu gering herausgestellt haben, wird auf die Darstellung dieses Gütekriteriums allerdings verzichtet.

Bei jedem dieser Präzisionsmaße muß wiederum zwischen Resubstitutionsfehler und Testfehler unterschieden werden. Resubstitution bedeutet, daß die Entwicklung einer Klassifikationsregel und die Bestimmung der Genauigkeit ihrer Vorhersage auf den gleichen Daten, nämlich dem kompletten Datensatz, beruhen. Dieser Resubstitutionsfehler kann verwendet werden, um zu untersuchen, wie schnell eine Fehlerrate von den verschiedenen Methoden abgesenkt werden kann. Wegen der deutlich vorhandenen Verzerrung ist er als Schätzung für die wahre Präzision der Prognose aber ungeeignet. Für Prototypen-Ansätze wie die Nächste-Nachbarn-Verfahren ist eine Resubstitution dagegen wie oben beschrieben gar nicht anwendbar. Daher teilt man den Datensatz zufallsbasiert in zwei Teile, die aus einem bzw. zwei Dritteln der Daten bestehen. Anhand der Einheiten des größeren Lerndatensatzes wird das Klassifikationsmodell gebildet und damit können dann die Beobachtungen des kleineren Testdatensatzes vorhergesagt werden. Hier werden 50 verschiedene Zufallsaufteilungen in Lern- und Testdaten verwendet und als Ergebnis der Mittelwert dieser Einzelergebnisse betrachtet.

Sämtliche angeführte Gütemaße erscheinen sinnvoll, sind aber in der Praxis nicht ausreichend, da nicht explizit hervorgeht, welche Art von Fehlklassifikationen überwiegend vorliegen, also beispielsweise, ob es sich um wenige gravierende oder viele geringfügige Abweichungen handelt, zwei Situationen, die in der Praxis durchaus unterschiedlich bewertet werden müssen. Deshalb sollte man immer zusätzlich eine einfache Häufigkeitsverteilung dieser Abstände (beispielsweise auch in Form von Diagrammen) betrachten. Hier handelt es sich zwar nicht um eine unmittelbar zum Vergleich geeignete, einzelne Maßzahl, dafür werden die Verhältnisse aber am besten wiedergegeben. Es ist also in der Anwendung immer empfehlenswert, sich nicht nur auf die etablierten Maßzahlen zu stützen, sondern zusätzlich einen Gesamteindruck mit Hilfe weiterer deskriptiver Verfahren zu bilden. Aus Platzgründen wird in dieser Arbeit jedoch weitgehend darauf verzichtet.

Bei den Ensemble-Methoden werden neben den skalaren Gütemaßen auch Abbildungen zu der Entwicklung der Fehlerraten mit wachsender Zyklenzahl präsentiert. Da aus Platzgründen unmöglich alle Verfahren in diesem empirischen Teil der Arbeit abgebildet werden können, mußte man sich auf eine Teilmenge festlegen. Die Wahl fiel auf die folgenden Verfahren: Der einfache CART sowie Real AdaBoost werden zum Vergleich herangezogen. Von den eigentlichen ordinalen Techniken werden Fixed Split Bagging sowie Fixed Split Real AdaBoost und Ordinal Gentle AdaBoost herangezogen. Jeweils wird die kumulative Variante verwendet, da sich herausgestellt hat, daß diese Version durchgehend besser abschneidet als die sequenzielle Aggregation. Zuletzt wird noch der ordinale L_2-Boost in den Vergleich mit einbezogen, so daß insgesamt sechs Techniken dargestellt werden. Alle diese Methoden zeichnen sich dadurch aus, daß sie relativ stabil und gut arbeiten. Sie sind aber keinesfalls durchgängig die Besten, manchmal tritt sogar bei einigen dieser Techniken Overfitting auf, so daß sie für den jeweiligen Datensatz als ungeeignet erscheinen. Um zu einem gerechten Urteil über die Algorithmen zu kommen, erscheint aber eine solche objektive Auswahl sinnvoller, als für jeden Datensatz stets nur die besten Verfahren auszuwählen und das Overfitting-Problem so gezielt zu umschiffen. Zusätzlich sind aber im Anhang D zumindest die Testfehler-Entwicklungen aller übrigen Verfahren bezüglich der fünf betrachteten Datensätze in einer ausführlichen Übersicht dargestellt.

6.2.2 Datensätze

Nachdem alle zu untersuchenden Verfahren dargestellt wurden und weiter geklärt ist, welche ordinalen Gütemaße zum Einsatz kommen sollen, kann nun ein Vergleich der Prognosegüte einer Vielzahl von alternativen Modellen vorgenommen werden. Dazu wird eine Reihe von empirischen Datensätzen herangezogen, die sich in der Anzahl der ordinalen Klassen teilweise deutlich unterscheiden. Sie werden im Folgenden kurz vorgestellt.

Verzichtet wurde hier auf simulierte ordinale Daten, da eine zu starke Abhängigkeit zwischen der Wahl des zugrunde liegenden Modells der Daten und dem Zielmodell des Klassifikationsverfahrens besteht (vgl. Rudolfer, Watson & Lesaffre (1995)). Ein fairer Vergleich der Ergebnisse von verschiedenen Techniken erscheint so nicht gegeben.

Scapula-Datensatz

Die Schulterblatt-Daten wurden im Rahmen einer Doktorarbeit von W. Feistl am rechtsmedizinischen Institut der LMU München, betreut von Prof. Dr. Penning, erhoben, bisher aber noch nicht veröffentlicht. Thema ist die Identifikation unbekannter Leichen, bei der neben Geschlecht und ethnischer Herkunft vor allem die Alterseinschätzung eine wichtige Rolle spielt. Bei stark beeinträchtigten Leichen stehen jedoch nicht alle hierfür nutzbaren Informationen zur Verfügung. In dieser Arbeit soll deshalb untersucht werden, ob eine Altersschätzung alleine anhand von Schulterblättern, die in den meisten Fällen vorhanden sind, möglich ist und welche Genauigkeit dabei erzielt werden kann.

Dazu stehen die Daten von 153 Leichen zur Verfügung. Insgesamt wurden je 79 Größen erhoben, die Vermessungen, wie Strecken oder Winkel, sowie morphologische Beurteilungen wie Oberflächenbeschaffenheit des Schulterblattes darstellen. In einer Vorauswahl wurden 15 besonders wichtige Variablen als Kovariaten identifiziert. Zielgröße ist die eigentlich metrische Größe Alter. Da eine auf das Jahr genaue Prognose des Alters zu schwierig, wenn nicht gar unmöglich erschien, wurde eine Kategorisierung in 10-Jahres-Intervalle vorgenommen, so daß eine Zielgröße mit acht ordinalen Klassen entsteht. Die daraus resultierenden Klassenumfänge sind stark unterschiedlich. Neben zwei extrem gering besetzten Randklassen (2 bzw. 6 Fälle) variieren die restlichen Klassengrößen zwischen 9 und 45 Beobachtungen.

Umfrage unter Kunden einer Automobilfirma

Die Daten stammen aus der Umfrage einer deutschen Automobilfirma und wurden in Fahrmeir, Hamerle & Tutz (1996) veröffentlicht. Im Jahr 1983 wurden Fragebögen an 2000 Kunden versandt, die ein neues Auto vor etwa drei Monaten erworben hatten. Die Hauptinteressen waren der Grad der Zufriedenheit, Gründe für die jeweilige Wahl und Kundenprofile. Die Teilnahme an der Studie war selbstverständlich freiwillig. Nur 1182 Personen beantworteten den Fragebogen und nach der Entfernung von Bögen, die fehlende Werte aufwiesen, blieben lediglich 793 Fragebögen übrig. Jeder Bogen besteht aus 46 Fragen. Der Grad der Zufriedenheit, der ursprünglich in zehn ordinalen Klassen gemessen wurde, wird in dieser Untersuchung als Zielvariable verwendet. Da viele der Klassen nur sehr wenig Fälle aufweisen, wurden sie zu vier verschiedenen ordinalen Klassen zusammengefaßt, die schließlich zwischen 136 und 292 Beobachtung enthalten.

Mietspiegel-Datensatz

Zahlreiche deutsche Städte erstellen sogenannte Mietspiegel, um Mietern, Vermietern, Mietberatungsstellen und Sachverständigen eine objektive Entscheidungshilfe in Mietfragen zur Verfügung zu stellen. Die Mietspiegel werden dabei insbesondere zur Ermittlung der ortsüblichen Vergleichsmiete (Nettomiete in Abhängigkeit von Wohnungsgröße,

-ausstattung, -alter, etc.) herangezogen. Bei der Erstellung von Mietspiegeln (im Fall der Stadt München durch Infratest) wird aus der Gesamtheit aller in Frage kommenden Wohnungen eine repräsentative Zufallsstichprobe gezogen, und die interessierenden Daten von Interviewern anhand von Fragebögen ermittelt. Der vorliegende Datensatz stellt einen Ausschnitt aus dem Mietspiegel München des Jahres 1994 dar und enthält die Daten von 1082 Wohnungen. Um ordinale Klassifikation anwenden zu können, wurden die Nettomieten in 5 Klassen von in etwa gleicher Klassengröße (zwischen 208 und 230) gruppiert. Dieser Datensatz wurde bei Fahrmeir, Künstler, Pigeot & Tutz (1997) als Beispiel herangezogen.

Herzerkrankungs-Datensatz

Diese Daten aus dem Archiv der University of California in Berkeley enthalten vier gleichartige Datensätze, die sich mit der Diagnose von Herzerkrankungen befassen und an den folgenden Instituten erhoben wurden: Cleveland Clinic Foundation (Robert Detrano, M.D., Ph.D.), Hungarian Institute of Cardiology (Budapest; Andras Janosi, M.D.), V.A. Medical Center (Long Beach, CA; Robert Detrano, M.D., Ph.D.) und University Hospital (Zurich, Switzerland; William Steinbrunn, M.D.). Sie weisen 76 Attribute auf, allerdings wurden in bisherigen Veröffentlichungen lediglich 14 davon genutzt. Die interessierende Größe betrifft das Auftreten von Herzerkrankungen bei den beobachteten Patienten. Die entsprechende Variable stellt den angiographischen Erkrankungsstatus dar und wird gemessen anhand einer Variable mit Werten zwischen 0 (keine Erkrankung) und 4. Aufgrund der Beschreibung der Variablen gehen wir von einer ordinalen Struktur dieser Zielgröße aus. Da drei der Kovariablen eine große Menge von fehlenden Werten aufweisen, werden sie ebenso aus dem Datensatz entfernt wie Beobachtungen, die in den verbleibenden Kovariaten fehlende Werte aufweisen. Insgesamt können so 437 Fälle untersucht werden. Letztendlich werden noch die Klassen 3 und 4 der Zielgröße zu einer Klasse zusammengefaßt, da sie jeweils nur wenige Beobachtungen enthalten, so daß ein Vier-Klassen-Problem mit 11 Kovariaten vorliegt. Die Klassenumfänge variieren zwischen 43 und 193 Beobachtungen.

Wachheits-Datensatz

In diesem Datensatz wird anhand eines ordinalen Scores mit sieben Klassen beurteilt, wie wach ein Kind zu einem bestimmten Zeitpunkt nach einer Operation mit Vollnarkose ist. Da nur 60 Fälle vorliegen und sich deshalb teilweise nur sehr geringe Klassengrößen ergeben, wurden besonders kleine benachbarte Klassen zusammengefaßt, so daß die Zielgröße letztendlich fünf ordinale Klassen mit Umfängen zwischen 8 und 24 Fällen aufweist. Als unabhängige Variablen werden das Alter, die Dauer der Operation sowie die Dosierung des Narkosemittels herangezogen.

6.2.3 Ergebnisse

Scapula-Datensatz

Evaluation	Verfahren	Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Resubstitution	nominaler CART	0.405	0.647	1.418
	nominales Bagging	0.222	0.379	0.889
	nominales Discrete AdaBoost	0.052	0.052	0.052
	nominales Real AdaBoost	0.078	0.078	0.078
	nominales Gentle AdaBoost	0.007	0.007	0.007
	kumulativer Fixed Split CART	0.444	0.542	0.830
	kumulatives Fixed Split Bagging	0.386	0.431	0.562
	kumulatives Fixed Split Discrete AdaBoost	0.000	0.000	0.000
	kumulatives Fixed Split Real AdaBoost	0.000	0.000	0.000
	kumulatives Fixed Split Gentle AdaBoost	0.000	0.000	0.000
	kumulatives Fixed Split LogitBoost	0.013	0.013	0.013
	kumulatives Fixed Split L_2-Boost	0.098	0.105	0.118
	kumulatives Ordinal Discrete AdaBoost	0.131	0.131	0.131
	kumulatives Ordinal Real AdaBoost	0.412	0.425	0.451
	kumulatives Ordinal Gentle AdaBoost	0.327	0.327	0.327
	sequenzieller Fixed Split CART	0.412	0.529	0.895
	sequenzielles Fixed Split Bagging	0.196	0.229	0.307
	sequenzielles Fixed Split Discrete AdaBoost	0.092	0.118	0.170
	sequenzielles Fixed Split Real AdaBoost	0.235	0.301	0.458
	sequenzielles Fixed Split Gentle AdaBoost	0.222	0.281	0.425
	sequenzielles Fixed Split LogitBoost	0.007	0.007	0.007
	sequenzielles Fixed Split L_2-Boost	0.059	0.059	0.059
	sequenzielles Ordinal Discrete AdaBoost	0.144	0.144	0.144
	Ordinal L_2-Boost	0.353	0.359	0.373
Test	nominaler CART	0.667	1.111	2.546
	nominales Bagging	0.645	1.019	2.196
	nominales Discrete AdaBoost	0.656	0.942	1.753
	nominales Real AdaBoost	0.641	0.870	1.456
	nominales Gentle AdaBoost	0.644	0.886	1.544
	kumulativer Fixed Split CART	0.677	0.958	1.793
	kumulatives Fixed Split Bagging	0.643	0.839	1.360
	kumulatives Fixed Split Discrete AdaBoost	0.650	0.815	1.208
	kumulatives Fixed Split Real AdaBoost	0.645	0.808	1.218
	kumulatives Fixed Split Gentle AdaBoost	0.647	0.820	1.247
	kumulatives Fixed Split LogitBoost	0.645	0.894	1.623
	kumulatives Fixed Split L_2-Boost	0.611	0.768	1.166
	kumulatives Ordinal Discrete AdaBoost	0.638	0.863	1.475
	kumulatives Ordinal Real AdaBoost	0.693	0.890	1.370
	kumulatives Ordinal Gentle AdaBoost	0.651	0.815	1.230
	sequenzieller Fixed Split CART	0.711	1.113	2.331
	sequenzielles Fixed Split Bagging	0.696	1.014	1.942
	sequenzielles Fixed Split Discrete AdaBoost	0.673	0.960	1.749
	sequenzielles Fixed Split Real AdaBoost	0.706	1.048	2.051
	sequenzielles Fixed Split Gentle AdaBoost	0.708	1.047	2.045
	sequenzielles Fixed Split LogitBoost	0.661	0.987	1.980
	sequenzielles Fixed Split L_2-Boost	0.647	0.919	1.666
	sequenzielles Ordinal Discrete AdaBoost	0.629	0.906	1.618
	Ordinal L_2-Boost	0.663	0.870	1.358

Tabelle 6.12: Fehlerraten bei den Scapula-Daten

Bei den nominalen Ansätzen werden Bäume mit einer maximalen Endknotenzahl von 16 (Hierarchietiefe 4) verwendet. Dies erscheint notwendig für eine Aufgabenstellung mit acht Klassen und insgesamt 15 Kovariablen. Alle ordinalen Ansätze werden dagegen mit Bäumen der maximalen Größe 8 (Hierarchietiefe 3) durchgeführt, da hier nur jeweils zwei Klassen behandelt werden müssen. Eine Ausnahme bilden die auf Logit- oder L_2-Boost basierenden Fixed-Split-Varianten, die bei zu großen Bäumen schnell zu Overfitting führen. Deshalb beschränkt man sich hier auf Stumps (Hierarchietiefe 1), ebenso wie beim regressionsbasierten ordinalen L_2-Boost.

Zunächst kann man feststellen, daß bei diesen Daten speziell zwei Verfahren, nämlich die beiden auf dem LogitBoost basierenden Fixed-Split-Techniken, bereits nach wenigen Zyklen

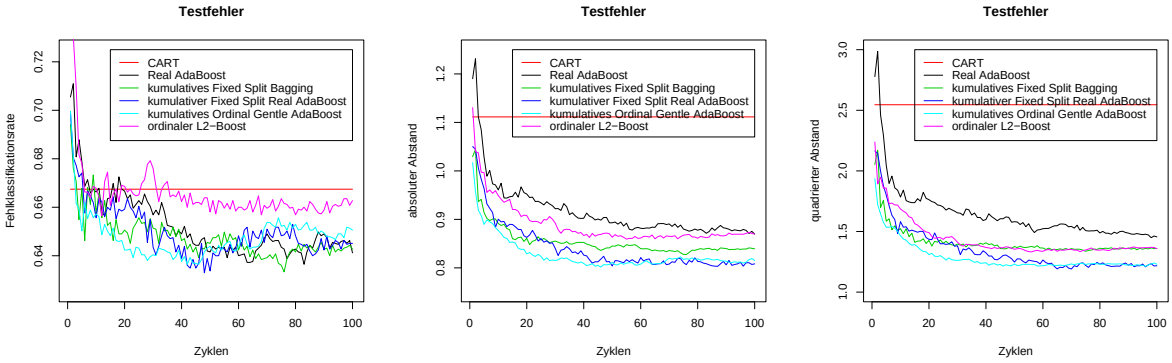

Abbildung 6.11: Entwicklung der Testfehler bei den Scapula-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand)

in Overfitting münden, was bei der Beurteilung dieser Methoden berücksichtigt werden sollte. Würde hier bereits früher, zum Beispiel nach 25 Zyklen, abgebrochen, so würde man quadrierte Abstände von 1.695 beim sequenziellen, sowie 1.177 beim kumulativen Ansatz erhalten. Das letztgenannte Ergebnis wäre eines der besten insgesamt. Da aber ansonsten bei keiner der untersuchten Techniken Overfitting auftrat, kann man zusammenfassend festhalten, daß dieser Datensatz sehr geeignet für die Anwendung von Ensemble-Techniken zu sein scheint. Eine vollständige Übersicht über die Verlaufsdiagramme der Fehlerraten mit wachsender Zyklenzahl aller Verfahren ist dem Anhang D zu entnehmen. Dies gilt ebenso für alle weiteren Beispieldaten dieses Abschnitts.

Die Interpretation von Tabelle 6.12 führt zu den folgenden Schlüssen: Bei der Resubstitution werden alle Arten der Fehlermessung durch die Verwendung von aggregierten Verfahren verbessert. Im Vergleich zu Fixed Split Boosting, wo der Fehler auf Null reduziert wird, und nominalen Boosting-Varianten, die der Null zumindest sehr nahe kommen, fallen die Resultate der anderen ordinalen Methoden etwas ab. Dennoch schneiden sie stets besser ab als Bagging.

Die interessanteren Ergebnisse basieren sicherlich auf der Validierung mit Hilfe von getrennten Testdatensätzen, da hier die eigentliche Vorhersagegüte eines Klassifikators direkt getestet wird. Betrachtet man die Fehlklassifikationsraten, so findet man kaum nennenswerte Unterschiede zwischen den betrachteten Verfahren. Die für Probleme mit ordinalen Klassen ohnehin wichtigeren Kriterien sind allerdings die Abstandsmaße. Sowohl die absolute als auch die quadrierte Distanz zeigen ein ähnliches Verhalten, aber die Unterschiede zwischen den zu vergleichenden Verfahren treten bei den quadrierten Abständen noch deutlicher zu Tage. Man kann festhalten, daß die Ergebnisse von CART mit allen aggregierten Ansätzen verbessert werden. Besonders Fixed Split Boosting, aber auch einige andere ordinale Boosting-Techniken sowie Fixed Split Bagging, liefern sehr gute Ergebnisse.

Verfahren	k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Modus, kein y-Kern	max. 25	rectangular	0.661	1.035	2.117
	max. 25	epanechnikov	0.632	0.911	1.664
	max. 25	all kernels	0.653	0.980	1.893
Median, kein y-Kern	max. 25	rectangular	0.621	0.849	1.432
	max. 25	epanechnikov	0.608	0.804	1.293
	max. 25	all kernels	0.613	0.809	1.304
Median und y-Kern	max. 25	rectangular	0.648	0.831	1.296
	max. 25	epanechnikov	0.637	0.805	1.214
	max. 25	all kernels	0.632	0.800	1.212

Verfahren	k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Modus, kein y-Kern	1	rectangular	0.631	0.916	1.691
	3	rectangular	0.656	1.018	2.036
	5	rectangular	0.651	0.976	1.876
	7	rectangular	0.649	1.001	1.988
	11	rectangular	0.657	1.025	2.091
	21	rectangular	0.651	1.116	2.540
	1	epanechnikov	0.631	0.916	1.691
	3	epanechnikov	0.633	0.915	1.685
	5	epanechnikov	0.619	0.897	1.652
	7	epanechnikov	0.611	0.875	1.577
	11	epanechnikov	0.626	0.882	1.544
	21	epanechnikov	0.637	0.925	1.691
Median, kein y-Kern	1	rectangular	0.631	0.916	1.691
	3	rectangular	0.617	0.849	1.431
	5	rectangular	0.618	0.826	1.344
	7	rectangular	0.621	0.827	1.346
	11	rectangular	0.606	0.823	1.382
	21	rectangular	0.614	0.871	1.557
	1	epanechnikov	0.631	0.916	1.691
	3	epanechnikov	0.630	0.888	1.573
	5	epanechnikov	0.617	0.833	1.385
	7	epanechnikov	0.603	0.796	1.280
	11	epanechnikov	0.603	0.787	1.245
	21	epanechnikov	0.595	0.798	1.303
Median und y-Kern	1	rectangular	0.624	0.894	1.585
	3	rectangular	0.626	0.809	1.263
	5	rectangular	0.642	0.804	1.208
	7	rectangular	0.633	0.800	1.223
	11	rectangular	0.665	0.856	1.349
	21	rectangular	0.698	0.945	1.628
	1	epanechnikov	0.624	0.894	1.585
	3	epanechnikov	0.616	0.820	1.325
	5	epanechnikov	0.620	0.781	1.173
	7	epanechnikov	0.626	0.779	1.153
	11	epanechnikov	0.638	0.792	1.169
	21	epanechnikov	0.664	0.841	1.286

Tabelle 6.13: Fehlerraten bei den Scapula-Daten (Dargestellt ist eine Auswahl der Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte.)

Abbildung 6.11 zeigt die Güte von sechs ausgewählten Klassifikationstechniken. Während auf der y-Achse jeweils eines der drei Präzisionsmaße abgetragen wird, stellt die x-Achse die Anzahl der berechneten Zyklen dar. Die Abbildungen zeigen also die Entwicklung der Vorhersagequalität während des Laufs der Algorithmen. Wiederum sieht man, daß kein großer Unterschied zwischen den Werten der Fehlklassifikationsraten besteht (man vergleiche die Skala der y-Achse) und daß die ordinalen Boosting-Verfahren bezüglich der Distanzmaße deutlich überlegen sind.

Auch die Abbildung 6.12, die aus Platzgründen nur für diesen Datensatz gezeigt wird, gibt die Güte einiger ausgewählter Methoden wieder. Hier wird das Hauptaugenmerk allerdings auf die einzelnen Beobachtungen im Datensatz gerichtet. Auf der x-Achse sind die Untersuchungseinheiten geordnet nach ihrer Klassenzugehörigkeit abgetragen. Die y-Achse zeigt den Median der vorhergesagten Klasse über alle 50 verschiedenen Aufteilungen in Lern-

und Teststichprobe an. Man sieht, daß die großen Unterschiede zwischen Vorhersage und wahrem Wert meist bei den gleichen Beobachtungen auftreten. Dies weist darauf hin, daß einige Untersuchungseinheiten dieses Datensatzes offensichtlich besonders schwer zu klassifizieren sind, egal welche Technik dazu verwendet wird. Dieser Hinweis auf Ausreißer ist für empirische Daten nicht ungewöhnlich und zeigt sich auch bei den anderen betrachteten Beispielen.

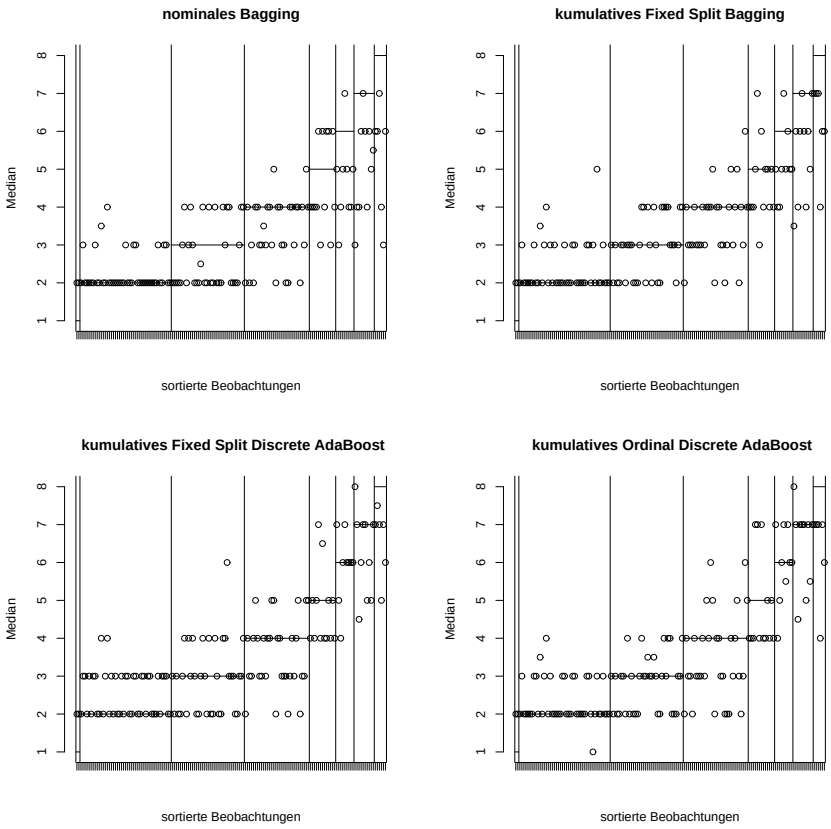

Abbildung 6.12: Median der vorhergesagten Klasse jeder Beobachtung des Datensatzes (Vertikale Linien trennen zwischen den Klassen, horizontale Linien zeigen die wahre Klasse innerhalb dieser Grenzen; Angewandte Methoden von links oben nach rechts unten: nominales Bagging, kumulatives Fixed Split Bagging, kumulatives Fixed Split Discrete AdaBoost, kumulatives Ordinal Discrete AdaBoost)

Bei der Interpretation der kkNN-Ergebnisse (Tabelle 6.13) sind ebenfalls die Distanzmaße zwischen wahrem Wert und Prognose von besonderem Interesse, da bei diesen Maßzahlen die Fähigkeit zur Berücksichtigung der ordinalen Struktur deutlich wird. Die Wahl $k = 1$ erweist sich als besonders gut, wenn man auf die Berücksichtigung der ordinalen Struktur zunächst verzichtet, also die Modi statt die Mediane der Klassenverteilung betrachtet. Für höhere k verbessern Gewichtungen stets die immer schlechter werdenden ungewichteten

kNN-Lösungen. Im Vergleich zu nominalen Boosting-Techniken befinden sich die optimalen Resultate aber immerhin im oberen Mittelfeld. Verwendet man dagegen den Median, um der Ordinalität in der Zielgröße Rechnung zu tragen, so stellt man fest, daß sich die Fehler für jede einzelne untersuchte Parameterkombination weiter verringern. Im Vergleich zu ordinalen Ensemble-Techniken sind die optimalen Ergebnisse jetzt sogar im Spitzenfeld zu finden. Zieht man zuletzt auch noch einen y-Kern hinzu, so tritt nochmal eine geringfügige Verbesserung auf. Dies führt bei sehr geringer Rechenzeit zu herausragend guten Ergebnissen.

Umfrage unter Kunden einer Automobilfirma

Evaluation	Verfahren	Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Resubstitution	nominaler CART	0.585	0.788	1.219
	nominales Bagging	0.583	0.743	1.063
	nominales Discrete AdaBoost	0.294	0.377	0.549
	nominales Real AdaBoost	0.262	0.313	0.414
	nominales Gentle AdaBoost	0.213	0.270	0.386
	kumulativer Fixed Split CART	0.594	0.711	0.948
	kumulatives Fixed Split Bagging	0.570	0.639	0.778
	kumulatives Fixed Split Discrete AdaBoost	0.077	0.078	0.081
	kumulatives Fixed Split Real AdaBoost	0.014	0.014	0.014
	kumulatives Fixed Split Gentle AdaBoost	0.010	0.010	0.010
	kumulatives Fixed Split LogitBoost	0.472	0.562	0.752
	kumulatives Fixed Split L_2-Boost	0.469	0.546	0.707
	kumulatives Ordinal Discrete AdaBoost	0.570	0.794	1.243
	kumulatives Ordinal Real AdaBoost	0.591	0.593	0.595
	kumulatives Ordinal Gentle AdaBoost	0.574	0.574	0.574
	sequenzieller Fixed Split CART	0.589	0.753	1.081
	sequenzielles Fixed Split Bagging	0.589	0.762	1.107
	sequenzielles Fixed Split Discrete AdaBoost	0.549	0.733	1.121
	sequenzielles Fixed Split Real AdaBoost	0.552	0.733	1.096
	sequenzielles Fixed Split Gentle AdaBoost	0.561	0.739	1.095
	sequenzielles Fixed Split LogitBoost	0.455	0.591	0.884
	sequenzielles Fixed Split L_2-Boost	0.460	0.588	0.860
	sequenzielles Ordinal Discrete AdaBoost	0.638	0.943	1.554
	Ordinal L_2-Boost	0.562	0.605	0.694
Test	nominaler CART	0.651	0.851	1.268
	nominales Bagging	0.627	0.798	1.144
	nominales Discrete AdaBoost	0.618	0.801	1.191
	nominales Real AdaBoost	0.605	0.778	1.145
	nominales Gentle AdaBoost	0.608	0.782	1.149
	kumulativer Fixed Split CART	0.650	0.798	1.101
	kumulatives Fixed Split Bagging	0.634	0.739	0.949
	kumulatives Fixed Split Discrete AdaBoost	0.603	0.760	1.099
	kumulatives Fixed Split Real AdaBoost	0.622	0.801	1.189
	kumulatives Fixed Split Gentle AdaBoost	0.610	0.777	1.139
	kumulatives Fixed Split LogitBoost	0.607	0.761	1.090
	kumulatives Fixed Split L_2-Boost	0.603	0.741	1.033
	kumulatives Ordinal Discrete AdaBoost	0.668	0.986	1.713
	kumulatives Ordinal Real AdaBoost	0.632	0.721	0.898
	kumulatives Ordinal Gentle AdaBoost	0.625	0.712	0.886
	sequenzieller Fixed Split CART	0.637	0.833	1.244
	sequenzielles Fixed Split Bagging	0.618	0.800	1.166
	sequenzielles Fixed Split Discrete AdaBoost	0.632	0.817	1.208
	sequenzielles Fixed Split Real AdaBoost	0.615	0.797	1.166
	sequenzielles Fixed Split Gentle AdaBoost	0.618	0.799	1.167
	sequenzielles Fixed Split LogitBoost	0.615	0.808	1.223
	sequenzielles Fixed Split L_2-Boost	0.617	0.802	1.200
	sequenzielles Ordinal Discrete AdaBoost	0.724	1.177	2.239
	Ordinal L_2-Boost	0.613	0.704	0.893

Tabelle 6.14: Fehlerraten bei der Kundenumfrage

Beim nominalen Ansatz werden hier Bäume mit maximal 8 Endknoten herangezogen, da dieses Klassifikationsproblem lediglich auf vier Kategorien beruht. Alle ordinalen Ansätze werden ebenfalls auf Basis der gleichen Hierarchietiefe durchgeführt. Ausnahme bilden

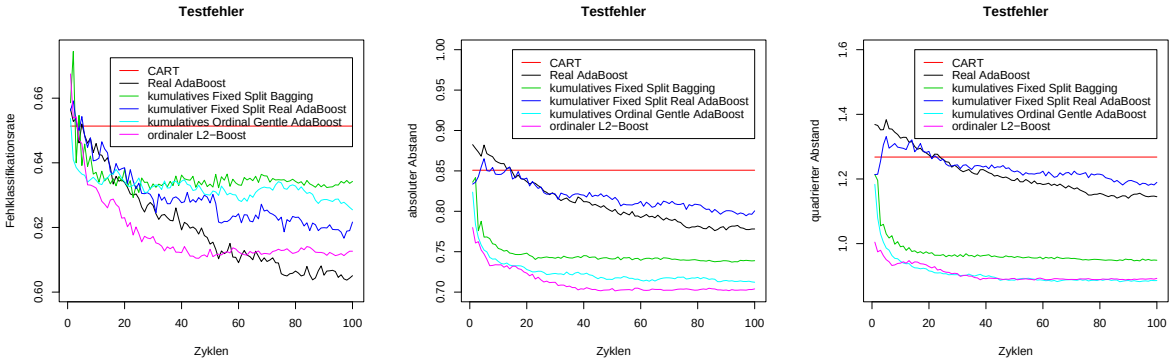

Abbildung 6.13: Entwicklung der Testfehler bei der Kundenumfrage (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand)

wieder die Logit- und L_2-Boost-Varianten, die auf Stumps (Hierarchietiefe 1) beschränkt werden.

Bei diesen Daten kann man feststellen, daß häufiger Overfitting auftritt als bei den Scapula-Daten. Bereits bei der Resubstitution entstehen bei Ordinal Discrete AdaBoost in beiden Varianten, also kumulativ und sequenziell, Probleme. Beispielsweise lauten die quadrierten Abstände nur 0.682 bzw. 0.905, wenn man die Algorithmen bereits nach 25 Zyklen abbrechen würde. Die selben beiden Techniken führen auch bei Betrachtung der Testfehler zu den gravierendsten Problemen (siehe Anhang D). Die nach 25 Zyklen erreichten quadrierten Abstände von 1.121 und 1.649 sind zwar wesentlich geringer als die in der Tabelle angegebenen nach 100 Zyklen, zählen aber dennoch nicht zu den niedrigsten im Vergleich zu den übrigen Techniken. Desweiteren tritt Overfitting auch bei vier Varianten des Fixed Split Boosting auf, nämlich bei sämtlichen Kombinationen aus kumulativer bzw. sequenzieller Aggregation und Logit- bzw. L_2-Boost. Hier ist das Anwachsen der Fehlerraten vom Betrag her jedoch so klein, daß es keine große Rolle spielt. Alle sonstigen Verfahren arbeiten jedoch einwandfrei.

Tabelle 6.14 kann folgendermaßen interpretiert werden: Auf die gleiche Art und Weise wie bei den Scapula-Daten werden die Fehlermaße bei Resubstitution durch die Verwendung von aggregierten Klassifikationsverfahren durchwegs verbessert. Insbesondere Fixed Split Boosting senkt den Fehler beinahe auf Null, während andere ordinale Boosting-Ansätze hier deutlich schlechter abschneiden. Speziell die beiden ordinalen Discrete AdaBoost-Varianten liefern sehr hohe Fehlerraten, die aber wie bereits erwähnt auf Overfitting zurückzuführen sind. Insgesamt erscheint die Verbesserung durch Verwendung von aggregierten Klassifikatoren nicht so überzeugend zu sein wie bei den Scapula-Daten.

Die Validierungsergebnisse bezüglich getrennter Testdatensätze, die einen deutlich besseren Indikator für die Vorhersagegüte eines Klassifikationsverfahrens darstellen, zeigen wiederum ähnliche Effekte wie bei den Scapula-Daten. Der wichtigste Unterschied zwischen den

Ergebnissen bezüglich dieser beiden Datensätze ist aber die Feststellung, daß verschiedene ordinale Methoden als die jeweils besten erscheinen. Die Fehlklassifikationsrate zeigt erneut nur geringe Differenzen zwischen den Methoden, weshalb kein überlegenes Verfahren bezüglich dieses Maßes herausstich. Bezüglich der interessanteren Abstandsmaße werden die Ergebnisse von CART aber zumindest geringfügig von den meisten der betrachteten aggregierten Verfahren verbessert. Allerdings erweist sich Fixed Split Boosting, das zu optimalen Ergebnissen bei den Scapula-Daten geführt hat, hier bei Weitem nicht als die beste Prozedur. Stattdessen liefern kumulativer Ordinal Real und Gentle AdaBoost sehr gute Ergebnisse, ebenso wie Fixed Split Bagging. Andere Verfahren wie die bereits oben genannten ordinalen Discrete AdaBoost-Varianten weisen dagegen sogar eine deutliche Verschlechterung gegenüber einem einfachen CART-Baum auf. Dieses Phänomen ist aber auf eine deutliche Tendenz zu Overfitting zurückzuführen, wie man an den Abbildungen D.1 bis D.4 im Anhang erkennen kann. Insgesamt scheint das Verbesserungspotenzial durch Ensemble-Techniken deutlich geringer als bei den Scapula-Daten zu sein.

Auch bei den kkNN-Techniken (Tabelle 6.15) sind ähnliche, wenn auch schwächere, Resultate wie bei den Scapula-Daten zu beobachten. Sämtliche auf dem Modus basierende Ergebnisse werden durch Verwendung des Medians deutlich verbessert. Eine zusätzliche Einbeziehung des y-Kerns senkt die Fehlerraten noch einmal, allerdings nicht mehr wesentlich. Im Gegensatz zu Scapula ist aber eine sehr hohe Anzahl k von Nachbarn bei Verwendung des Rechteckskerns optimal, so daß der Epanechnikov-Kern erst für die höchsten ausprobierten k-Werte zu Verbesserungen gegenüber der ungewichteten Variante führen kann. Das beste Einzelresultat ergibt sich demnach für die Wahl $k = 101$, Epanechnikov-Kern und Verwendung von Median und y-Kern, und erreicht in dieser Kombination auch die besten Boosting-Ergebnisse.

Mietspiegel-Datensatz

Ebenso wie bei den Scapula-Daten basieren alle nominalen Ansätze auf Bäumen mit einer maximalen Hierarchietiefe von 4 und alle ordinalen Ansätze auf einer Hierarchietiefe von 3. Wie immer werden die auf Logit- oder L_2-Boost basierenden Fixed-Split-Varianten, wie der regressionsbasierte ordinale L_2-Boost, mit Stumps, also einer Hierarchietiefe von 1, durchgeführt.

Bei diesem Datensatz zeigt sich bereits anhand der Resubstitutionsfehler, daß Ensemble-Techniken nicht ganz so gut arbeiten wie bei den beiden bisher beschriebenen Beispielen. Zwar tritt nur einmal deutliches Overfitting auf, nämlich bei sequenziellem Ordinal Discrete AdaBoost, aber bei vielen weiteren Techniken resultiert mit wachsender Zyklenzahl nur eine geringe Wirkung auf die Fehlerraten. Betrachtet man die Testfehlerraten (siehe auch Anhang D), so erkennt man Overfitting bei kumulativem Fixed Split Boosting in den Varianten Real und Gentle sowie bei sämtlichen Ordinal AdaBoost-Varianten, besonders

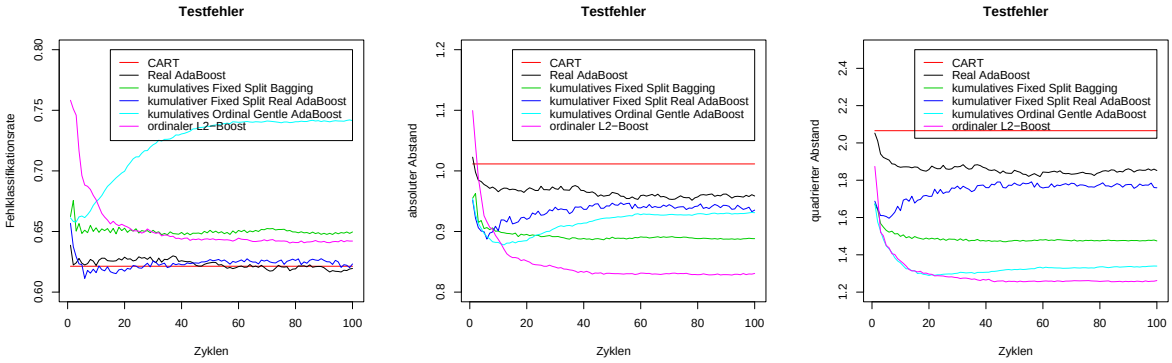

Abbildung 6.14: Entwicklung der Testfehler bei den Mietspiegel-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand)

aber in Verbindung mit Discrete AdaBoost.

Faßt man hier die Resubstitutions- und die Test-Ergebnisse der Ensemble-Techniken aus Tabelle 6.16 zusammen, so kann man festhalten, daß beinahe alle kumulativen Verfahren zufriedenstellend abschneiden. Lediglich ordinales Discrete AdaBoost, das sich auch bei anderen Datensätzen als problematisch erweist, fällt mit einer deutlichen Erhöhung der Fehlerraten völlig aus der Reihe. In der Validierung erweist sich jedoch der ordinale L_2-Boost als das beste Verfahren. Dagegen schneiden sämtliche sequenzielle Verfahren sehr schlecht ab. Diese Tendenz läßt sich zwar ebenfalls bei anderen Datensätzen erahnen, ist hier aber besonders deutlich ausgeprägt.

Betrachtet man die Nächste-Nachbarn-Varianten (Tabelle 6.17), so kann man wieder eine sehr deutliche Verbesserung der Resultate erkennen, wenn man vom Modus zum Median übergeht. Die zusätzliche Verwendung des y-Kerns führt dagegen nicht mehr zu verringerten Fehlerraten, im Durchschnitt bleibt das Niveau hier in etwa gleich. Der Epanechnikov-Kern führt fast durchgängig zu etwas geringeren Abstandsmaßen im Vergleich zur ungewichteten Variante. Die besten kkNN-Resultate liegen wiederum in etwa auf dem Niveau der guten kumulativen Ensemble-Techniken.

Im Anhang C dieser Arbeit befinden sich zusätzlich die Nächste-Nachbarn-Resultate für eine Gruppierung der Zielgröße in 10 ordinale Klassen. Wiederum wurde auf in etwa gleich große Klassenumfänge geachtet, die sich hier nun zwischen 90 und 132 bewegen. Wie zu erwarten war, sind hier die Verbesserungen durch Berücksichtigung der Ordinalität via Median und y-Kern noch etwas deutlicher ausgeprägt.

Verfahren	k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Modus, kein y-Kern	max. 25	rectangular	0.648	0.862	1.329
	max. 25	epanechnikov	0.632	0.839	1.293
	max. 25	all kernels	0.635	0.836	1.273
Median, kein y-Kern	max. 25	rectangular	0.652	0.778	1.031
	max. 25	epanechnikov	0.629	0.764	1.044
	max. 25	all kernels	0.632	0.756	1.007
Median und y-Kern	max. 25	rectangular	0.655	0.769	0.996
	max. 25	epanechnikov	0.651	0.770	1.007
	max. 25	all kernels	0.649	0.759	0.979

Verfahren	k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Modus, kein y-Kern	1	rectangular	0.663	0.941	1.575
	3	rectangular	0.679	0.992	1.714
	5	rectangular	0.652	0.899	1.452
	7	rectangular	0.649	0.888	1.422
	11	rectangular	0.642	0.852	1.313
	21	rectangular	0.642	0.829	1.222
	31	rectangular	0.643	0.818	1.181
	41	rectangular	0.644	0.809	1.144
	51	rectangular	0.640	0.805	1.137
	101	rectangular	0.638	0.807	1.146
	1	epanechnikov	0.663	0.941	1.575
	3	epanechnikov	0.663	0.939	1.566
	5	epanechnikov	0.654	0.911	1.491
	7	epanechnikov	0.645	0.888	1.430
	11	epanechnikov	0.631	0.852	1.342
	21	epanechnikov	0.624	0.819	1.245
	31	epanechnikov	0.619	0.801	1.192
	41	epanechnikov	0.618	0.795	1.168
	51	epanechnikov	0.619	0.787	1.137
	101	epanechnikov	0.628	0.794	1.131
Median, kein y-Kern	1	rectangular	0.663	0.941	1.575
	3	rectangular	0.633	0.834	1.273
	5	rectangular	0.635	0.800	1.149
	7	rectangular	0.637	0.783	1.088
	11	rectangular	0.641	0.774	1.044
	21	rectangular	0.649	0.768	1.006
	31	rectangular	0.652	0.767	0.998
	41	rectangular	0.657	0.770	0.996
	51	rectangular	0.652	0.766	0.994
	101	rectangular	0.652	0.778	1.031
	1	epanechnikov	0.663	0.941	1.575
	3	epanechnikov	0.659	0.918	1.503
	5	epanechnikov	0.635	0.845	1.305
	7	epanechnikov	0.628	0.811	1.207
	11	epanechnikov	0.620	0.776	1.104
	21	epanechnikov	0.630	0.758	1.020
	31	epanechnikov	0.636	0.755	0.992
	41	epanechnikov	0.641	0.756	0.986
	51	epanechnikov	0.641	0.753	0.976
	101	epanechnikov	0.642	0.751	0.969
Median und y-Kern	1	rectangular	0.656	0.875	1.352
	3	rectangular	0.650	0.784	1.055
	5	rectangular	0.657	0.777	1.018
	7	rectangular	0.662	0.780	1.018
	11	rectangular	0.658	0.774	1.006
	21	rectangular	0.659	0.768	0.985
	31	rectangular	0.657	0.761	0.969
	41	rectangular	0.661	0.762	0.964
	51	rectangular	0.656	0.754	0.949
	101	rectangular	0.662	0.764	0.968
	1	epanechnikov	0.656	0.875	1.352
	3	epanechnikov	0.651	0.786	1.058
	5	epanechnikov	0.646	0.773	1.026
	7	epanechnikov	0.648	0.771	1.015
	11	epanechnikov	0.651	0.766	0.996
	21	epanechnikov	0.652	0.763	0.984
	31	epanechnikov	0.651	0.759	0.976
	41	epanechnikov	0.651	0.756	0.966
	51	epanechnikov	0.650	0.751	0.955
	101	epanechnikov	0.650	0.743	0.929

Tabelle 6.15: Fehlerraten bei der Kundenumfrage (Dargestellt ist eine Auswahl der Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte.)

Evaluation	Verfahren	Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Resubstitution	nominaler CART	0.557	0.917	1.913
	nominales Bagging	0.523	0.848	1.739
	nominales Discrete AdaBoost	0.425	0.663	1.299
	nominales Real AdaBoost	0.422	0.623	1.128
	nominales Gentle AdaBoost	0.372	0.562	1.070
	kumulativer Fixed Split CART	0.603	0.833	1.406
	kumulatives Fixed Split Bagging	0.582	0.774	1.246
	kumulatives Fixed Split Discrete AdaBoost	0.397	0.544	0.916
	kumulatives Fixed Split Real AdaBoost	0.268	0.348	0.534
	kumulatives Fixed Split Gentle AdaBoost	0.257	0.331	0.501
	kumulatives Fixed Split LogitBoost	0.477	0.651	1.065
	kumulatives Fixed Split L_2-Boost	0.498	0.673	1.094
	kumulatives Ordinal Discrete AdaBoost	0.534	0.771	1.351
	kumulatives Ordinal Real AdaBoost	0.723	0.838	1.075
	kumulatives Ordinal Gentle AdaBoost	0.734	0.867	1.137
	sequenzieller Fixed Split CART	0.546	0.880	1.786
	sequenzielles Fixed Split Bagging	0.550	0.924	1.948
	sequenzielles Fixed Split Discrete AdaBoost	0.560	0.963	2.072
	sequenzielles Fixed Split Real AdaBoost	0.560	0.943	1.994
	sequenzielles Fixed Split Gentle AdaBoost	0.562	0.945	1.996
	sequenzielles Fixed Split LogitBoost	0.472	0.750	1.500
	sequenzielles Fixed Split L_2-Boost	0.479	0.768	1.555
	sequenzielles Ordinal Discrete AdaBoost	0.547	0.957	2.085
	Ordinal L_2-Boost	0.608	0.753	1.075
Test	nominaler CART	0.621	1.012	2.066
	nominales Bagging	0.592	0.950	1.917
	nominales Discrete AdaBoost	0.611	0.941	1.808
	nominales Real AdaBoost	0.620	0.959	1.852
	nominales Gentle AdaBoost	0.601	0.919	1.757
	kumulativer Fixed Split CART	0.664	0.949	1.652
	kumulatives Fixed Split Bagging	0.650	0.888	1.475
	kumulatives Fixed Split Discrete AdaBoost	0.602	0.862	1.529
	kumulatives Fixed Split Real AdaBoost	0.623	0.935	1.759
	kumulatives Fixed Split Gentle AdaBoost	0.617	0.910	1.671
	kumulatives Fixed Split LogitBoost	0.591	0.832	1.432
	kumulatives Fixed Split L_2-Boost	0.593	0.824	1.396
	kumulatives Ordinal Discrete AdaBoost	0.617	1.072	2.357
	kumulatives Ordinal Real AdaBoost	0.741	0.960	1.438
	kumulatives Ordinal Gentle AdaBoost	0.742	0.931	1.340
	sequenzieller Fixed Split CART	0.627	1.026	2.117
	sequenzielles Fixed Split Bagging	0.618	1.015	2.097
	sequenzielles Fixed Split Discrete AdaBoost	0.626	1.012	2.060
	sequenzielles Fixed Split Real AdaBoost	0.621	1.025	2.127
	sequenzielles Fixed Split Gentle AdaBoost	0.622	1.026	2.131
	sequenzielles Fixed Split LogitBoost	0.597	0.919	1.765
	sequenzielles Fixed Split L_2-Boost	0.596	0.923	1.789
	sequenzielles Ordinal Discrete AdaBoost	0.640	1.210	2.859
	Ordinal L_2-Boost	0.642	0.831	1.262

Tabelle 6.16: Fehlerraten bei den Mietspiegel-Daten

Verfahren	k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Modus, kein y-Kern	max. 25	rectangular	0.594	0.944	1.869
	max. 25	epanechnikov	0.598	0.946	1.864
	max. 25	all kernels	0.594	0.939	1.849
Median, kein y-Kern	max. 25	rectangular	0.628	0.861	1.419
	max. 25	epanechnikov	0.618	0.850	1.414
	max. 25	all kernels	0.620	0.853	1.417
Median und y-Kern	max. 25	rectangular	0.748	0.952	1.393
	max. 25	epanechnikov	0.749	0.952	1.394
	max. 25	all kernels	0.747	0.949	1.388

Verfahren	k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Modus, kein y-Kern	1	rectangular	0.640	1.045	2.143
	3	rectangular	0.639	1.087	2.330
	5	rectangular	0.619	0.987	1.966
	7	rectangular	0.611	0.982	1.966
	11	rectangular	0.600	0.954	1.894
	21	rectangular	0.592	0.938	1.850
	31	rectangular	0.601	0.959	1.908
	41	rectangular	0.602	0.964	1.930
	51	rectangular	0.607	0.969	1.940
	101	rectangular	0.613	0.983	1.971
	1	epanechnikov	0.640	1.045	2.143
	3	epanechnikov	0.630	1.025	2.095
	5	epanechnikov	0.623	1.002	2.013
	7	epanechnikov	0.615	0.979	1.943
	11	epanechnikov	0.604	0.954	1.875
	21	epanechnikov	0.594	0.937	1.840
	31	epanechnikov	0.588	0.928	1.825
	41	epanechnikov	0.588	0.931	1.840
	51	epanechnikov	0.587	0.932	1.849
	101	epanechnikov	0.594	0.944	1.879
Median, kein y-Kern	1	rectangular	0.640	1.045	2.143
	3	rectangular	0.624	0.936	1.728
	5	rectangular	0.616	0.894	1.580
	7	rectangular	0.614	0.873	1.501
	11	rectangular	0.620	0.859	1.437
	21	rectangular	0.631	0.883	1.501
	31	rectangular	0.653	0.870	1.384
	41	rectangular	0.663	0.871	1.358
	51	rectangular	0.677	0.879	1.345
	101	rectangular	0.742	0.945	1.399
	1	epanechnikov	0.640	1.045	2.143
	3	epanechnikov	0.634	0.998	1.967
	5	epanechnikov	0.624	0.936	1.726
	7	epanechnikov	0.616	0.903	1.615
	11	epanechnikov	0.612	0.869	1.496
	21	epanechnikov	0.619	0.849	1.404
	31	epanechnikov	0.627	0.845	1.367
	41	epanechnikov	0.635	0.847	1.354
	51	epanechnikov	0.642	0.849	1.341
	101	epanechnikov	0.687	0.881	1.327
Median und y-Kern	1	rectangular	0.717	1.075	2.008
	3	rectangular	0.728	0.969	1.519
	5	rectangular	0.747	0.957	1.416
	7	rectangular	0.743	0.948	1.391
	11	rectangular	0.747	0.947	1.379
	21	rectangular	0.754	0.949	1.363
	31	rectangular	0.759	0.961	1.389
	41	rectangular	0.763	0.971	1.410
	51	rectangular	0.765	0.980	1.429
	101	rectangular	0.781	1.027	1.534
	1	epanechnikov	0.717	1.075	2.008
	3	epanechnikov	0.735	0.983	1.558
	5	epanechnikov	0.746	0.966	1.461
	7	epanechnikov	0.747	0.957	1.421
	11	epanechnikov	0.745	0.944	1.374
	21	epanechnikov	0.751	0.947	1.366
	31	epanechnikov	0.753	0.949	1.368
	41	epanechnikov	0.755	0.949	1.362
	51	epanechnikov	0.757	0.952	1.365
	101	epanechnikov	0.772	0.984	1.429

Tabelle 6.17: Fehlerraten bei den Mietspiegel-Daten (Dargestellt ist eine Auswahl der Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte.)

Herzerkrankungs-Datensatz

Evaluation	Verfahren	Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Resubstitution	nominaler CART	0.355	0.471	0.741
	nominales Bagging	0.314	0.446	0.776
	nominales Discrete AdaBoost	0.149	0.158	0.176
	nominales Real AdaBoost	0.229	0.236	0.249
	nominales Gentle AdaBoost	0.146	0.146	0.146
	kumulativer Fixed Split CART	0.359	0.442	0.625
	kumulatives Fixed Split Bagging	0.330	0.368	0.451
	kumulatives Fixed Split Discrete AdaBoost	0.002	0.002	0.002
	kumulatives Fixed Split Real AdaBoost	0.000	0.000	0.000
	kumulatives Fixed Split Gentle AdaBoost	0.000	0.000	0.000
	kumulatives Fixed Split LogitBoost	0.176	0.192	0.229
	kumulatives Fixed Split L_2-Boost	0.227	0.259	0.327
	kumulatives Ordinal Discrete AdaBoost	0.334	0.368	0.437
	kumulatives Ordinal Real AdaBoost	0.476	0.476	0.476
	kumulatives Ordinal Gentle AdaBoost	0.373	0.373	0.373
	sequenzieller Fixed Split CART	0.341	0.428	0.625
	sequenzielles Fixed Split Bagging	0.297	0.403	0.641
	sequenzielles Fixed Split Discrete AdaBoost	0.275	0.368	0.584
	sequenzielles Fixed Split Real AdaBoost	0.286	0.407	0.709
	sequenzielles Fixed Split Gentle AdaBoost	0.291	0.407	0.691
	sequenzielles Fixed Split LogitBoost	0.153	0.217	0.382
	sequenzielles Fixed Split L_2-Boost	0.190	0.270	0.476
	sequenzielles Ordinal Discrete AdaBoost	0.394	0.462	0.600
	Ordinal L_2-Boost	0.330	0.350	0.391
Test	nominaler CART	0.413	0.580	0.978
	nominales Bagging	0.393	0.535	0.869
	nominales Discrete AdaBoost	0.410	0.523	0.790
	nominales Real AdaBoost	0.418	0.525	0.772
	nominales Gentle AdaBoost	0.413	0.525	0.787
	kumulativer Fixed Split CART	0.428	0.526	0.744
	kumulatives Fixed Split Bagging	0.402	0.467	0.605
	kumulatives Fixed Split Discrete AdaBoost	0.374	0.474	0.706
	kumulatives Fixed Split Real AdaBoost	0.385	0.497	0.758
	kumulatives Fixed Split Gentle AdaBoost	0.376	0.483	0.730
	kumulatives Fixed Split LogitBoost	0.383	0.495	0.747
	kumulatives Fixed Split L_2-Boost	0.378	0.480	0.709
	kumulatives Ordinal Discrete AdaBoost	0.426	0.580	0.954
	kumulatives Ordinal Real AdaBoost	0.514	0.580	0.714
	kumulatives Ordinal Gentle AdaBoost	0.447	0.515	0.656
	sequenzieller Fixed Split CART	0.417	0.553	0.878
	sequenzielles Fixed Split Bagging	0.398	0.530	0.840
	sequenzielles Fixed Split Discrete AdaBoost	0.390	0.518	0.818
	sequenzielles Fixed Split Real AdaBoost	0.398	0.536	0.866
	sequenzielles Fixed Split Gentle AdaBoost	0.399	0.534	0.858
	sequenzielles Fixed Split LogitBoost	0.386	0.515	0.820
	sequenzielles Fixed Split L_2-Boost	0.379	0.501	0.788
	sequenzielles Ordinal Discrete AdaBoost	0.448	0.623	1.065
	Ordinal L_2-Boost	0.404	0.475	0.624

Tabelle 6.18: Fehlerraten bei den Herzerkrankungs-Daten

Ebenso wie bei der Kundenumfrage werden im nominalen Ansatz Bäume mit maximal 8 Endknoten verwendet, die für ein Klassifikationsproblem mit vier Klassen ausreichend erscheinen. Alle ordinalen Techniken, die ja auf Zweiklassenproblemen beruhen, werden ebenfalls mit Bäumen dieser Größe durchgeführt. Auf Logit- oder L_2-Boost basierende Techniken arbeiten mit Stumps.

Eine Betrachtung des Overfitting-Problems zeigt, daß bereits bei Resubstitution sämtliche sequenzielle Verfahren schlecht abschneiden. Entweder es wird kaum Verbesserung erzielt oder es tritt bereits hier Overfitting auf. Bei den Testfehlerraten (siehe Abbildungen D.5 bis D.8) erweist sich dieses Problem dann allerdings als gar nicht so gravierend, wie es die Resubstitutionsfehler vermuten lassen würden. Lediglich kumulatives und sequenzielles Ordinal Discrete AdaBoost zeigen jetzt deutlich steigende Fehlerraten, während einige andere Techniken, die größtenteils bereits bei der Kundenumfrage problematisch waren,

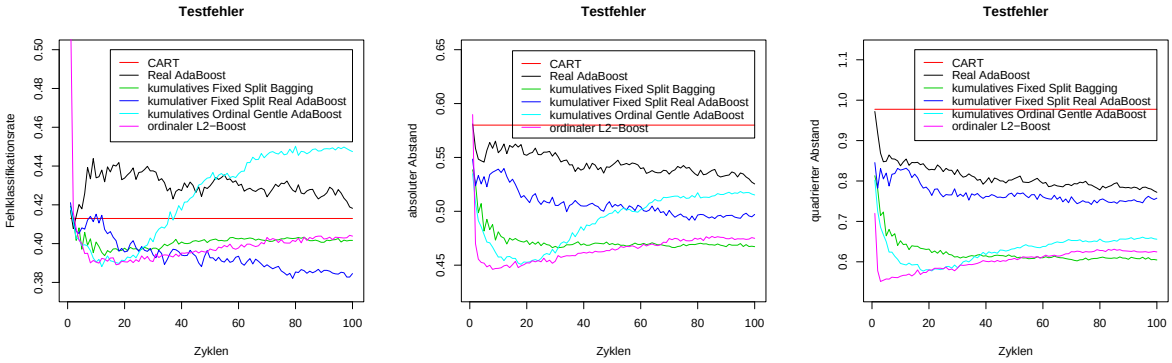

Abbildung 6.15: Entwicklung der Testfehler bei den Herzerkrankungs-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand)

lediglich in geringem Maße zu Overfitting führen.

Tabelle 6.18 kann folgendermaßen zusammengefaßt werden: Wiederum werden bei Resubstitution alle Arten von Fehlermessungen durch die Verwendung von aggregierten Klassifizierern verbessert. Eine Ausnahme bildet allerdings Ordinal Real AdaBoost, wo die Abstandsmaße lediglich auf Kosten von erhöhter Fehlklassifikation sinken. Fixed Split Boosting schafft es andererseits wieder, sämtliche Fehler auf Null zu reduzieren.

Bezüglich der Test-Ergebnisse findet man dagegen deutlich andere Effekte als bei den bisher betrachteten Datensätzen: Obwohl ordinale Varianten aller untersuchten Algorithmen Verbesserungen im Vergleich zu den entsprechenden nominalen Verfahren mit sich bringen, scheint Boosting in Kombination mit kumulativem oder sequenziellem Ansatz nicht zu einer Optimierung der Resultate beitragen zu können. Bagging und manchmal sogar CART führen in dieser Kombination nämlich zu etwas geringeren Fehlerraten. Kumulatives Fixed Split Bagging ist demnach auch hier das beste Verfahren, dicht gefolgt von ordinalem L_2-Boost. So ist dieser Datensatz als ein Beispiel anzusehen, bei dem die Verwendung von Boosting nicht uneingeschränkt empfohlen werden kann. Stattdessen scheint Bagging hier die deutlich angebrachtere Technik zu sein. Der Grund für diese Unterschiede in den Resultaten bei den verschiedenen betrachteten Beispielen bleibt allerdings unklar, möglicherweise deuten relativ gute Ergebnisse von Bagging im Vergleich zu sehr schlechtem Boosting auf Ausreißer in den Daten hin (siehe zum Beispiel Breiman (1998)).

Dagegen erweisen sich Nächste-Nachbarn-Techniken allgemein als hervorragend für diesen Datensatz geeignet (siehe Tabelle 6.19). Schon die gewöhnliche ungewichtete Methode ohne Verwendung von ordinalen Hilfsmitteln wie Median oder y-Kern liefert im Vergleich zu den Ensemble-Techniken sehr gute Resultate. Die Hinzunahme des Medians führt dann sowohl mit Epanechnikov-Kern als auch ohne Gewichtung zu Abstandsmaßen, die unterhalb der besten Ensemble-Ergebnisse liegen. Aus einer zusätzlichen Verwendung des y-Kerns ergeben sich dagegen geringfügig schlechtere Resultate, die im Vergleich zu Boo-

sting aber immernoch im Spitzenfeld liegen. Da das optimale k der gewöhnlichen Nächste-Nachbarn-Methode nicht besonders hoch ist, führen etwas höhere k in Verbindung mit dem Epanechnikov-Kern erwartungsgemäß zu den besten Ergebnissen. Das beobachtete Optimum liegt bei einem quadrierten Abstand von 0.54 im Vergleich zu etwa 0.60 bei kumulativem Fixed Split Bagging.

Wachheits-Datensatz

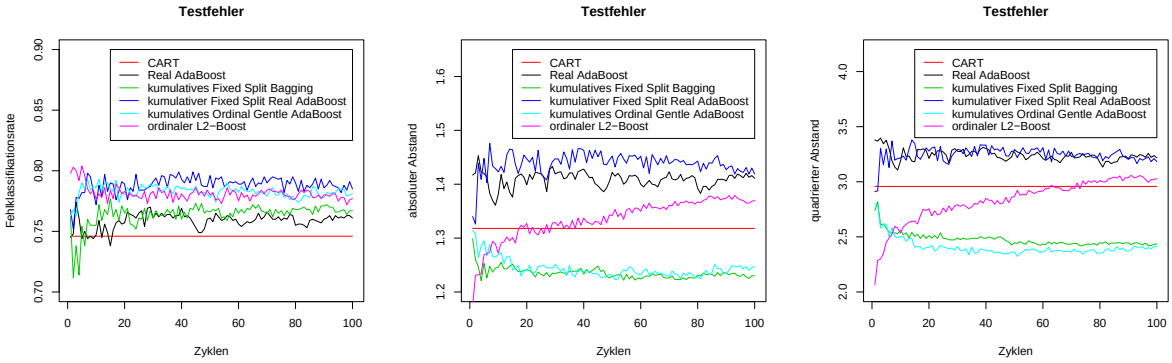

Abbildung 6.16: Entwicklung der Testfehler bei den Wachheits-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand)

Im nominalen Ansatz werden Bäume mit maximal 8 Endknoten verwendet. Alle ordinalen Techniken basieren dagegen lediglich auf einer maximal zulässigen Hierarchietiefe von 2, da ja auch nur drei Kovariaten herangezogen werden. Auf Logit- oder L_2-Boost basierende Techniken arbeiten wie immer mit Stumps.

Während sämtliche Verfahren bei Resubstitution zufriedenstellend, wenn auch nicht mit allzu großen Verbesserungen arbeiten, zeigt sich bei den Testfehlern ein völlig anderes Bild (siehe Abbildungen D.17 bis D.20). Sämtliche Varianten von Fixed Split Boosting weisen ein deutliches Overfitting auf, ebenso wie beide Ordinal Discrete AdaBoost-Methoden. Sogar der ordinale L_2-Boost, der sich bei allen anderen Datensätzen als zuverlässig und konstant gut herausgestellt hat, führt mit steigender Zyklenanzahl zu deutlich wachsenden Fehlerraten. Aber auch fast alle übrigen Techniken arbeiten sehr schlecht, so daß kaum Verbesserungen durch Anwendung der Ensemble-Ansätze erzielt werden können.

Die Wachheits-Daten sind bekannterweise ein Beispiel, das vielen statistischen Verfahren erhebliche Probleme bereitet. Dies zeigt sich deutlich an Tabelle 6.20: Bereits bei den Resubstitutionsfehlern fällt auf, daß zwar einige Verfahren (sowohl nominale, als auch kumulative und sequenzielle) zu Fehlerraten von Null führen, während jedoch andere Techniken sogar Verschlechterungen im Vergleich zu einem einzelnen CART-Baum bewirken.

Betrachtet man die Test-Ergebnisse, so fällt sofort auf, daß fast alle Verfahren zu Verschlechterungen statt Verbesserungen führen. Lediglich einige kumulative Varianten sind in der Lage, den einzelnen CART-Baum bezüglich der Abstandsmaße wenigstens geringfügig zu unterbieten. Wie schlecht die Ensemble-Techniken insgesamt bei diesem Datensatz greifen, ist wie gesagt auch den Abbildungen D.17 bis D.20 im Anhang zu entnehmen. Insgesamt scheint der Datensatz beinahe gar nicht auf diese Art der Klassifikation anzusprechen.

Ganz anders sieht es bei den kkNN-Techniken aus: Während bereits die gewöhnliche Nächste-Nachbarn-Methode besser als sämtliche Ensemble-Techniken abschneidet, treten sowohl durch Hinzunahme des Medians als auch unter Zuhilfenahme von y-Kernen nochmal deutliche Verbesserungen auf. Diese Resultate (in Verbindung mit einer Gewichtung anhand des Epanechnikov-Kerns) mit quadrierten Abständen von zum Teil unter 1.8 dominieren dann deutlich über Ensemble-Techniken, deren Fehlerraten alle größer als 2.2 sind.

Eine mögliche Begründung, warum hier die Ensemble-Techniken so gravierend schlechter als kkNN abschneiden, während sie bei allen anderen Datensätzen zumindest teilweise zu besseren Fehlerraten führen, liefert der besonders geringe Stichprobenumfang. Lediglich 40 Untersuchungseinheiten in der Lernstichprobe sind für derart komplexe Techniken wie Boosting zu gering, während ein sehr einfaches Verfahren wie kkNN doch deutlich besser funktioniert. Dies wurde in der Literatur auch schon des öfteren in Verbindung mit Microarray-Daten festgestellt (vgl. zum Beispiel Dudoit, Fridlyand & Speed (2002)), bei denen auch nur sehr wenig Beobachtungen vorliegen. Diese Beobachtung deckt sich ebenfalls mit den Microarray-Ergebnissen in dieser Arbeit.

Verfahren	k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Modus, kein y-Kern	max. 25	rectangular	0.372	0.486	0.739
	max. 25	epanechnikov	0.376	0.490	0.746
	max. 25	all kernels	0.365	0.473	0.715
Median, kein y-Kern	max. 25	rectangular	0.367	0.435	0.574
	max. 25	epanechnikov	0.371	0.443	0.593
	max. 25	all kernels	0.368	0.434	0.569
Median und y-Kern	max. 25	rectangular	0.415	0.478	0.603
	max. 25	epanechnikov	0.424	0.484	0.606
	max. 25	all kernels	0.414	0.475	0.596

Verfahren	k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Modus, kein y-Kern	1	rectangular	0.388	0.504	0.766
	3	rectangular	0.400	0.545	0.882
	5	rectangular	0.379	0.488	0.730
	7	rectangular	0.378	0.493	0.749
	11	rectangular	0.364	0.473	0.720
	21	rectangular	0.375	0.495	0.754
	31	rectangular	0.390	0.531	0.834
	41	rectangular	0.399	0.551	0.879
	51	rectangular	0.398	0.550	0.874
	101	rectangular	0.407	0.565	0.900
	1	epanechnikov	0.388	0.504	0.766
	3	epanechnikov	0.380	0.492	0.743
	5	epanechnikov	0.376	0.484	0.724
	7	epanechnikov	0.372	0.481	0.723
	11	epanechnikov	0.363	0.464	0.685
	21	epanechnikov	0.359	0.461	0.684
	31	epanechnikov	0.365	0.472	0.704
	41	epanechnikov	0.373	0.489	0.740
	51	epanechnikov	0.385	0.515	0.792
	101	epanechnikov	0.401	0.558	0.889
Median, kein y-Kern	1	rectangular	0.388	0.504	0.766
	3	rectangular	0.371	0.461	0.658
	5	rectangular	0.367	0.440	0.594
	7	rectangular	0.364	0.433	0.576
	11	rectangular	0.361	0.423	0.550
	21	rectangular	0.369	0.433	0.562
	31	rectangular	0.379	0.458	0.618
	41	rectangular	0.391	0.482	0.666
	51	rectangular	0.396	0.495	0.697
	101	rectangular	0.398	0.524	0.781
	1	epanechnikov	0.388	0.504	0.766
	3	epanechnikov	0.377	0.482	0.715
	5	epanechnikov	0.367	0.456	0.649
	7	epanechnikov	0.368	0.445	0.609
	11	epanechnikov	0.362	0.426	0.558
	21	epanechnikov	0.362	0.420	0.539
	31	epanechnikov	0.364	0.422	0.540
	41	epanechnikov	0.367	0.428	0.550
	51	epanechnikov	0.376	0.443	0.579
	101	epanechnikov	0.395	0.504	0.723
Median und y-Kern	1	rectangular	0.413	0.492	0.661
	3	rectangular	0.415	0.475	0.595
	5	rectangular	0.447	0.497	0.598
	7	rectangular	0.461	0.512	0.615
	11	rectangular	0.510	0.554	0.642
	21	rectangular	0.615	0.664	0.761
	31	rectangular	0.642	0.700	0.815
	41	rectangular	0.656	0.723	0.858
	51	rectangular	0.661	0.738	0.892
	101	rectangular	0.662	0.778	1.009
	1	epanechnikov	0.413	0.492	0.661
	3	epanechnikov	0.418	0.475	0.589
	5	epanechnikov	0.444	0.494	0.593
	7	epanechnikov	0.461	0.509	0.606
	11	epanechnikov	0.509	0.554	0.644
	21	epanechnikov	0.611	0.654	0.740
	31	epanechnikov	0.639	0.684	0.774
	41	epanechnikov	0.645	0.692	0.786
	51	epanechnikov	0.647	0.697	0.798
	101	epanechnikov	0.662	0.747	0.918

Tabelle 6.19: Fehlerraten bei den Herzerkrankungs-Daten (Dargestellt ist eine Auswahl der Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte.)

Evaluation	Verfahren	Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Resubstitution	nominaler CART	0.517	0.900	1.967
	nominales Bagging	0.467	0.917	2.117
	nominales Discrete AdaBoost	0.017	0.017	0.017
	nominales Real AdaBoost	0.000	0.000	0.000
	nominales Gentle AdaBoost	0.033	0.067	0.167
	kumulativer Fixed Split CART	0.550	0.917	1.833
	kumulatives Fixed Split Bagging	0.500	0.783	1.517
	kumulatives Fixed Split Discrete AdaBoost	0.000	0.000	0.000
	kumulatives Fixed Split Real AdaBoost	0.000	0.000	0.000
	kumulatives Fixed Split Gentle AdaBoost	0.000	0.000	0.000
	kumulatives Fixed Split LogitBoost	0.000	0.000	0.000
	kumulatives Fixed Split L_2-Boost	0.017	0.033	0.067
	kumulatives Ordinal Discrete AdaBoost	0.300	0.417	0.650
	kumulatives Ordinal Real AdaBoost	0.733	0.800	0.933
	kumulatives Ordinal Gentle AdaBoost	0.517	0.600	0.767
	sequenzieller Fixed Split CART	0.683	1.400	3.400
	sequenzielles Fixed Split Bagging	0.400	0.817	2.050
	sequenzielles Fixed Split Discrete AdaBoost	0.267	0.500	1.133
	sequenzielles Fixed Split Real AdaBoost	0.450	0.933	2.333
	sequenzielles Fixed Split Gentle AdaBoost	0.450	0.917	2.283
	sequenzielles Fixed Split LogitBoost	0.000	0.000	0.000
	sequenzielles Fixed Split L_2-Boost	0.033	0.083	0.217
	sequenzielles Ordinal Discrete AdaBoost	0.333	0.433	0.633
	Ordinal L_2-Boost	0.450	0.467	0.500
Test	nominaler CART	0.746	1.318	2.958
	nominales Bagging	0.705	1.205	2.575
	nominales Discrete AdaBoost	0.765	1.422	3.274
	nominales Real AdaBoost	0.761	1.411	3.217
	nominales Gentle AdaBoost	0.757	1.408	3.234
	kumulativer Fixed Split CART	0.773	1.266	2.560
	kumulatives Fixed Split Bagging	0.767	1.230	2.436
	kumulatives Fixed Split Discrete AdaBoost	0.786	1.431	3.225
	kumulatives Fixed Split Real AdaBoost	0.785	1.419	3.183
	kumulatives Fixed Split Gentle AdaBoost	0.789	1.425	3.181
	kumulatives Fixed Split LogitBoost	0.751	1.407	3.251
	kumulatives Fixed Split L_2-Boost	0.739	1.380	3.206
	kumulatives Ordinal Discrete AdaBoost	0.777	1.518	3.678
	kumulatives Ordinal Real AdaBoost	0.800	1.219	2.211
	kumulatives Ordinal Gentle AdaBoost	0.781	1.247	2.415
	sequenzieller Fixed Split CART	0.841	1.622	3.822
	sequenzielles Fixed Split Bagging	0.841	1.631	3.855
	sequenzielles Fixed Split Discrete AdaBoost	0.826	1.579	3.679
	sequenzielles Fixed Split Real AdaBoost	0.838	1.607	3.771
	sequenzielles Fixed Split Gentle AdaBoost	0.840	1.610	3.782
	sequenzielles Fixed Split LogitBoost	0.800	1.566	3.746
	sequenzielles Fixed Split L_2-Boost	0.761	1.481	3.581
	sequenzielles Ordinal Discrete AdaBoost	0.789	1.565	3.871
	Ordinal L_2-Boost	0.777	1.370	3.030

Tabelle 6.20: Fehlerraten bei den Wachheits-Daten

Verfahren	k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Modus, kein y-Kern	max. 25	rectangular	0.648	1.115	2.347
	max. 25	epanechnikov	0.694	1.199	2.541
	max. 25	all kernels	0.654	1.121	2.361
Median, kein y-Kern	max. 25	rectangular	0.744	1.184	2.264
	max. 25	epanechnikov	0.768	1.173	2.155
	max. 25	all kernels	0.769	1.186	2.202
Median und y-Kern	max. 25	rectangular	0.778	1.126	1.900
	max. 25	epanechnikov	0.785	1.143	1.931
	max. 25	all kernels	0.795	1.137	1.885

Verfahren	k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
Modus, kein y-Kern	1	rectangular	0.820	1.453	3.195
	3	rectangular	0.737	1.317	2.989
	5	rectangular	0.758	1.391	3.197
	7	rectangular	0.712	1.280	2.866
	11	rectangular	0.685	1.186	2.528
	21	rectangular	0.632	1.098	2.330
	31	rectangular	0.602	1.050	2.240
	1	epanechnikov	0.820	1.453	3.195
	3	epanechnikov	0.818	1.452	3.216
	5	epanechnikov	0.797	1.462	3.330
	7	epanechnikov	0.780	1.421	3.229
	11	epanechnikov	0.757	1.352	3.014
	21	epanechnikov	0.692	1.186	2.492
	31	epanechnikov	0.627	1.094	2.330
Median, kein y-Kern	1	rectangular	0.820	1.453	3.195
	3	rectangular	0.794	1.289	2.537
	5	rectangular	0.793	1.248	2.358
	7	rectangular	0.784	1.240	2.362
	11	rectangular	0.758	1.171	2.167
	21	rectangular	0.755	1.166	2.168
	31	rectangular	0.720	1.165	2.299
	1	epanechnikov	0.820	1.453	3.195
	3	epanechnikov	0.809	1.377	2.867
	5	epanechnikov	0.795	1.297	2.551
	7	epanechnikov	0.796	1.280	2.472
	11	epanechnikov	0.773	1.215	2.297
	21	epanechnikov	0.763	1.164	2.140
	31	epanechnikov	0.736	1.154	2.188
Median und y-Kern	1	rectangular	0.812	1.344	2.718
	3	rectangular	0.715	1.059	1.845
	5	rectangular	0.782	1.130	1.888
	7	rectangular	0.794	1.119	1.825
	11	rectangular	0.814	1.145	1.843
	21	rectangular	0.845	1.161	1.801
	31	rectangular	0.862	1.175	1.809
	1	epanechnikov	0.812	1.344	2.718
	3	epanechnikov	0.777	1.172	2.072
	5	epanechnikov	0.773	1.095	1.803
	7	epanechnikov	0.763	1.085	1.785
	11	epanechnikov	0.768	1.080	1.748
	21	epanechnikov	0.811	1.125	1.765
	31	epanechnikov	0.836	1.149	1.785

Tabelle 6.21: Fehlerraten bei den Wachheits-Daten (Dargestellt ist eine Auswahl der Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte.)

Kapitel 7

Zusammenfassung und Ausblick

Faßt man nun am Ende dieser Arbeit alle Ergebnisse zusammen, so läßt sich zunächst festhalten, daß sowohl Bagging als auch Boosting bei gewöhnlichen (nominalen) Klassifikationsproblemen im Großen und Ganzen sehr gut abschneiden. Bis auf wenige Ausnahmen zeigt sich, daß durch derartige Ensemble-Techniken das Ergebnis eines einfachen Klassifikationsbaumes zum Teil deutlich verbessert werden kann. Ferner erscheint Boosting als die etwas bessere Technik, da nicht nur die Fehlklassifikationsraten bei der Resubstitution bereits nach relativ wenigen Zyklen gegen Null gedrückt werden, sondern auch die Validierungsresultate anhand getrennter Teststichproben fast immer zumindest geringfügig besser sind als beim Bagging.

Das Konzept, aggregierte Klassifikationsverfahren mit Techniken zum Umgang mit ordinalen Datenstrukturen zu kombinieren, führt zu einer neuen Klasse von Methoden, die hier als Ordinal AdaBoost bezeichnet werden. Verschiedene Kriterien zur Aktualisierung der Gewichte innerhalb der Boosting-Methoden können hierzu herangezogen werden. Ein weiterer Vorschlag ist es, jede Dichotomisierung der Zielgröße getrennt per Boosting zu optimieren und diese Einzelergebnisse dann zu kombinieren, was hier als Fixed Split Boosting bezeichnet wird. Ebenso wie Boosting kann auch Bagging auf diese Art mit ordinaler Struktur kombiniert werden. Insgesamt werden also viele verschiedene Kombinationen von Ensemble- und ordinalen Techniken vorgestellt und empirisch verglichen. Diese Vergleiche werden anhand von mehreren Präzisionsmaßen durchgeführt. Neben der rohen Fehlklassifikationsrate werden auch Abstandsmaße betrachtet, um die ordinale Struktur in den Daten besser berücksichtigen zu können.

Die Ergebnisse bei diesen Versuchen zur ordinalen Klassifikation erweisen sich als gut, aber uneindeutig: Bei Datensätzen, die nur wenige ordinale Klassen aufweisen (Datensätze mit lediglich 3 Klassen wurden untersucht, in dieser Arbeit aber weggelassen), scheint die Verwendung von sequenziellem oder kumulativem Ansatz meist keine deutliche Verbesserung zu bringen. Im Gegenteil, manchmal resultiert sogar eine geringfügige Verschlechterung in den Gütemaßen. Während Bagging zumindest meistens zu relevanten Verbesserungen

führt, werden mit Boosting teilweise schlechtere Ergebnisse erzielt als mit einzelnen Klassifikationsbäumen. Bei Resubstitution zeigt sich die Eigenschaft von Boosting, Fehlerraten gegen Null zu drücken, in Kombination mit ordinalen Ansätzen wesentlich deutlicher als ohne Berücksichtigung der Ordnung. Dies gilt vor allem für Fixed Split Boosting.

Betrachtet man dagegen Datensätze mit mehreren (hier bis zu acht) ordinalen Klassen, so zeigen sich sowohl durch ordinale Ansätze als auch durch Voting-Verfahren positive Effekte. Speziell bei den Scapula-Daten erweist sich die Kombination aus Boosting und kumulativem Ansatz als recht effektiv. Insgesamt sind aber auch hier die Resultate eher verschwommen. Eindeutige Aussagen, daß bestimmte ordinale Ansätze stets zu Verbesserungen führen, mehr Zyklen bei den Ensemble-Verfahren auch günstigere Ergebnisse bringen oder Bagging und Boosting einzelne Bäume immer übertreffen, lassen sich hier nicht treffen. Zwar finden wir insofern vielversprechende Ergebnisse, als bei einigen Datensätzen alle ordinalen Ansätze die Qualität eines einfachen CART-Baumes übertreffen. Dabei scheint allerdings keine dominierende Methode vorzuliegen, denn bei unterschiedlichen Datensätzen führen jeweils andere Verfahren zu den besten Resultaten. Obwohl Fixed Split Bagging sehr zufriedenstellende Ergebnisse erzielt, wird es meist von mindestens einer der ordinalen Boosting-Techniken übertroffen. Während bei einigen Datensätzen der Ansatz von Fixed Split Boosting zu den günstigsten Resultaten führt, weist bei anderen Daten eine Variante von Ordinal AdaBoost oder der ordinale L_2-Boost die besten Ergebnisse auf.

Wieder andere Resultate zeigen sich zum Beispiel bei den Herzerkrankungs-Daten. Zwar werden die Ergebnisse durch die Anwendung von ordinalen Ansätzen verbessert, Boosting als Ensemble-Technik scheint allerdings nicht angebracht zu sein. Hier produzieren nämlich Bagging und manchmal sogar CART in Verbindung mit dem Fixed-Split-Ansatz bessere Ergebnisse.

Die Tatsache, daß die Fehlklassifikationsrate bei allen Datensätzen so gut wie nicht beeinflußt wird, zeigt, daß die ordinalen Techniken die Basismethode lediglich insofern verbessern, als die falsch klassifizierten Fälle weniger weit von der wahren Klasse abweichen, also weniger schwere Fehler aufweisen.

Insgesamt zeigt sich demnach, daß die Kombination von ordinalen Ansätzen und Ensemble-Verfahren bei wenigen ordinalen Klassen der Zielgröße weniger nützlich erscheint, während bei mehreren ordinalen Klassen, wenn also die ordinale Struktur deutlicher ausgeprägt ist, zumindest einzelne, teilweise aber auch deutliche, Verbesserungen resultieren. Der Grund für diese Unterschiede liegt vermutlich darin, daß sieben oder acht Klassen mehr ordinale Information beinhalten als drei Klassen. Bei drei Klassen dürfte es keine allzu große Rolle spielen, ob man sie als ordinal erkennt oder von ungeordnet nebeneinander stehenden nominalen Werten ausgeht. Nimmt dagegen die Klassenzahl zu, so spielt es eine immer größere Rolle, welche Klassen zueinander benachbart liegen und zwischen welchen ein großer Unterschied besteht. Hier wird sich allgemein die Verwendung von ordinalen Ansätzen zur Klassifikation mehr auszahlen.

Was das Auftreten von Overfitting betrifft, so zeigt sich, daß die ordinalen Ansätze hier

anfälliger zu sein scheinen als die doch recht stabilen gewöhnlichen nominalen Boosting-Verfahren. Wie soll man in der Praxis also mit diesem Problem umgehen, wenn in einem Anwendungsproblem lediglich ein Lerndatensatz vorliegt? Da neue Beobachtungen, deren Klassenzugehörigkeit unbekannt ist und vorhergesagt werden soll, nicht in die Modellbildung einbezogen werden können, liegt es nahe, den vorliegenden Lerndatensatz ebenso in mehrere Lern- und Testpaare aufzuteilen und anhand der Verlaufsdiagramme zu entscheiden, bei welcher Anzahl von Zyklen abgebrochen werden soll. Dies könnte ebenso anhand einer Kreuzvalidierung geschehen, deren Vorgehensweise bis auf die Art der Aufteilung der Daten identisch zu dem hier gewählten Verfahren ist. Anschließend wird für die Klassifikation der neuen Beobachtungen diese empirisch optimale Anzahl von Boosting-Zyklen herangezogen. Ein Problem bleibt hierbei in jedem Fall aber weiterhin der hohe Rechenaufwand, der durch die Ermittlung der optimalen Zyklenzahl nochmal gesteigert wird.

Diese Feststellungen legen nahe, daß weitere Forschung in diesem Feld eine lohnende Aufgabe darstellt, da viele Variationen der bisher vorgestellten Techniken vorstellbar sind. Besonders für den Transfer der AdaBoost-Algorithmen von nominalen zu ordinalen Datenstrukturen könnten auch andere plausible Kriterien entwickelt und ausprobiert werden. Zudem wäre es möglich, direkt ordinale Kriterien während des Wachstums der Klassifikationsbäume zu optimieren. Dieser Ansatz war für die hier entwickelten Verfahren nicht nötig, da Fixed Split Boosting und Ordinal AdaBoost intern lediglich binär klassifizieren, während Logit- und L_2-Boost bezüglich des KQ-Kriteriums optimierte Regressionsbäume benötigen. Zuletzt konnte die Frage, warum ein derartig ausgeprägter Unterschied zwischen den verschiedenen Ansätzen in der Verbesserung der Resultate von Datensatz zu Datensatz auftritt, noch nicht endgültig beantwortet werden, was Anlaß für zusätzliche Studien sein könnte.

Für die zweite Neuentwicklung, nämlich die kkNN-Verfahren, kann man als Zusammenfassung festhalten, daß die Ergebnisse bei der Klassifikation von nominalskalierten Zielgrößen insofern viel versprechend sind, als ein zu hoch gewählter Parameter k durch das Gewichtungsschema implizit wieder nach unten korrigiert wird. Besonders beim aktuellen und schwierigen Problem von Microarray-Daten liefern Nächste-Nachbarn-Techniken allgemein, besonders aber kkNN, hervorragende Resultate. Zuletzt kann festgestellt werden, daß bei ordinaler Klassifikation durch die Verwendung des Medians statt des Modus der Klassenverteilung die Fehlerraten entscheidend gesenkt werden können und ein zusätzlicher y-Kern nochmal zu einer Verbesserung führt. Die Benutzung des arithmetischen Mittels statt des Medians oder des Modus führt dagegen zum bekannten Nadaraya-Watson-Schätzer für lokale Glättung, der damit ebenfalls im kkNN-Paket enthalten ist.

Eigentlich gelten lokale Verfahren bei hoher Dimension der Kovariaten als ungeeignet. In vielen praktischen Problemen, vor allem im Bereich von Microarray-Daten, die bekanntlich besonders hochdimensional sind, liefern kNN-Verfahren dagegen besonders gute Ergebnisse. Eine mögliche Erklärung hierfür ist, daß die Krux der hohen Dimensionalität speziell bei der Schätzung von Modellen, also von Parametern zur Beschreibung des Einflusses von Kovariaten, zum Tragen kommt. Nächste-Nachbarn-Techniken bilden dagegen in diesem

Sinne keine statistischen Modelle. Es findet lediglich eine Prognose anhand von Prototypen statt. Deshalb erscheint auch ein Versuch der Prognose von kardinalskalierten Zielgrößen bei hochdimensionalen Daten anhand des (gewichteten) Mittelwertes der nächsten Nachbarn viel versprechend, während lokale Regressionsverfahren, die ein statistisches Modell schätzen, hier schlecht abschneiden.

Der große Vorteil der Nächste-Nachbarn-Techniken im Vergleich zu den Ensemble-Methoden ist aber die Laufzeit: Während Simulationen mit ordinalem Boosting oft mehrere Stunden benötigen, lassen sich die entsprechenden kkNN-Resultate in wenigen Sekunden erhalten.

Ein kritisches Problem ist dagegen nach wie vor die Variablenselektion, woran sich durch die Einführung von Gewichten nichts geändert hat. Die Aufnahme vieler Kovariaten, die rein zufällig streuen und deren Variabilität nichts zur Trennung zwischen den Klassen beiträgt, kann die Prognose empfindlich stören. Hier benötigt man also nach wie vor geeignete Variablenselektionsverfahren, die möglichst vor der eigentlichen Klassifikation zum Einsatz kommen sollten. Trotz dieses Problems kann das Klassifikationsverfahren kkNN aber als erfolgversprechender neuer Ansatz angesehen werden.

Anhang A

R-Funktionen zu ordinalem Boosting

nominalEnsemble

Verwendung:

```
nominalEnsemble <- function(formula = formula(data), data,
technique, tree.depth=1, cycles=100, valid=FALSE, runs=1, ...)
```

Argumente:

formula: Ein Formel-Objekt.

data: Ein Datensatz.

technique: Das anzuwendende Aggregationsverfahren. Mögliche Techniken sind “cart“, “bagging“, “discrete“, “real“, “gentle“, “logit“ und “l2“.

tree.depth: Die Hierarchietiefe der verwendeten Klassifikationsbäume.

cycles: Die Anzahl der Iterationen der Aggregationsverfahren.

valid: Resubstitution (0) oder geteilte Lern- und Test-Datensätze (1).

runs: Anzahl der Aufteilungen in Lern- und Test-Datensätze.

Resultat: Ein *list*-Objekt der Klasse *ensemble*, bestehend aus den Komponenten

call: Der Funktionsaufruf.

$MISCLASS$: Eine Matrix der Fehlklassifikationsraten; In den Zeilen sind die Ergebnisse der Zyklen, in den Spalten die Ergebnisse der verschiedenen Aufteilungen in Lern- und Test-Datensatz enthalten.

$MEAN.ABS$: Eine Matrix der absoluten Abstände zwischen wahrer und prognostizierter Klasse; In den Zeilen sind die Ergebnisse der Zyklen, in den Spalten die Ergebnisse der verschiedenen Aufteilungen in Lern- und Test-Datensatz enthalten.

MEAN.SQU: Eine Matrix der quadrierten Abstände zwischen wahrer und prognostizierter Klasse; In den Zeilen sind die Ergebnisse der Zyklen, in den Spalten die Ergebnisse der verschiedenen Aufteilungen in Lern- und Test-Datensatz enthalten.

Autor: Klaus Hechenbichler

fsEnsemble

Verwendung:

```
fsEnsemble <- function(formula = formula(data), data, technique,
aggregation, tree.depth=1, cycles=100, valid=FALSE, runs=1, ...)
```

Argumente:

formula: Ein Formel-Objekt.

data: Ein Datensatz.

technique: Das anzuwendende Aggregationsverfahren. Mögliche Techniken sind "cart", "bagging", "discrete", "real", "gentle", "logit" und "l2".

aggregation: Kumulative ("cum") oder sequenzielle ("seq") Aggregation.

tree.depth: Die Hierarchietiefe der verwendeten Klassifikationsbäume.

cycles: Die Anzahl der Iterationen der Aggregationsverfahren.

valid: Resubstitution (0) oder geteilte Lern- und Test-Datensätze (1).

runs: Anzahl der Aufteilungen in Lern- und Test-Datensätze.

Resultat: Ein *list*-Objekt der Klasse *ensemble*, bestehend aus den Komponenten

call: Der Funktionsaufruf.

MISCLASS: Eine Matrix der Fehlklassifikationsraten; In den Zeilen sind die Ergebnisse der Zyklen, in den Spalten die Ergebnisse der verschiedenen Aufteilungen in Lern- und Test-Datensatz enthalten.

MEAN.ABS: Eine Matrix der absoluten Abstände zwischen wahrer und prognostizierter Klasse; In den Zeilen sind die Ergebnisse der Zyklen, in den Spalten die Ergebnisse der verschiedenen Aufteilungen in Lern- und Test-Datensatz enthalten.

MEAN.SQU: Eine Matrix der quadrierten Abstände zwischen wahrer und prognostizierter Klasse; In den Zeilen sind die Ergebnisse der Zyklen, in den Spalten die Ergebnisse der verschiedenen Aufteilungen in Lern- und Test-Datensatz enthalten.

Autor: Klaus Hechenbichler

ordinalEnsemble

Verwendung:

```
ordinalEnsemble <- function(formula = formula(data), data, technique,
aggregation, tree.depth=1, cycles=100, valid=FALSE, runs=1, ...)
```

Argumente:

formula: Ein Formel-Objekt.

data: Ein Datensatz.

technique: Das anzuwendende Aggregationsverfahren. Mögliche Techniken sind "discrete", "real" und "gentle".

aggregation: Kumulative ("cum") oder sequenzielle ("seq") Aggregation. Nicht zulässig ist die Kombination von "seq" mit "real" oder "gentle".

tree.depth: Die Hierarchietiefe der verwendeten Klassifikationsbäume.

cycles: Die Anzahl der Iterationen der Aggregationsverfahren.

valid: Resubstitution (0) oder geteilte Lern- und Test-Datensätze (1).

runs: Anzahl der Aufteilungen in Lern- und Test-Datensätze.

Resultat: Ein *list*-Objekt der Klasse *ensemble*, bestehend aus den Komponenten

call: Der Funktionsaufruf.

$MISCLASS$: Eine Matrix der Fehlklassifikationsraten; In den Zeilen sind die Ergebnisse der Zyklen, in den Spalten die Ergebnisse der verschiedenen Aufteilungen in Lern- und Test-Datensatz enthalten.

$MEAN.ABS$: Eine Matrix der absoluten Abstände zwischen wahrer und prognostizierter Klasse; In den Zeilen sind die Ergebnisse der Zyklen, in den Spalten die Ergebnisse der verschiedenen Aufteilungen in Lern- und Test-Datensatz enthalten.

$MEAN.SQU$: Eine Matrix der quadrierten Abstände zwischen wahrer und prognostizierter Klasse; In den Zeilen sind die Ergebnisse der Zyklen, in den Spalten die Ergebnisse der verschiedenen Aufteilungen in Lern- und Test-Datensatz enthalten.

Autor: Klaus Hechenbichler

ordinalL2

Verwendung:

```
ordinalL2 <- function(formula = formula(data), data, wc=0,
tree.depth=1, cycles=100, valid=FALSE, runs=1, ...)
```

Argumente:

formula: Ein Formel-Objekt.

data: Ein Datensatz.

wc: *with constraints*; Verwendung (1) oder Weglassen (0) von Beschränkungen der Score-Werte innerhalb des Aktualisierungsalgorithmus.

tree.depth: Die Hierarchietiefe der verwendeten Klassifikationsbäume.

cycles: Die Anzahl der Iterationen der Aggregationsverfahren.

valid: Resubstitution (0) oder geteilte Lern- und Test-Datensätze (1).

runs: Anzahl der Aufteilungen in Lern- und Test-Datensätze.

Resultat: Ein *list*-Objekt der Klasse *ensemble*, bestehend aus den Komponenten

call: Der Funktionsaufruf.

$MISCLASS$: Eine Matrix der Fehlklassifikationsraten; In den Zeilen sind die Ergebnisse der Zyklen, in den Spalten die Ergebnisse der verschiedenen Aufteilungen in Lern- und Test-Datensatz enthalten.

$MEAN.ABS$: Eine Matrix der absoluten Abstände zwischen wahrer und prognostizierter Klasse; In den Zeilen sind die Ergebnisse der Zyklen, in den Spalten die Ergebnisse der verschiedenen Aufteilungen in Lern- und Test-Datensatz enthalten.

$MEAN.SQU$: Eine Matrix der quadrierten Abstände zwischen wahrer und prognostizierter Klasse; In den Zeilen sind die Ergebnisse der Zyklen, in den Spalten die Ergebnisse der verschiedenen Aufteilungen in Lern- und Test-Datensatz enthalten.

Autor: Klaus Hechenbichler

Zusätzliche Ausgabefunktionen

print.ensemble

summary.ensemble

plot.ensemble

Anhang B

Das *R*-Package *kkNN*

Weighted k-Nearest Neighbor Classifier (kknn)

Description:

Performs k-nearest neighbor classification of a test set using a training set. For each row of the test set, the k nearest training set vectors (according to Minkowsky distance) are found, and the classification is done via the maximum of summed kernel densities. In addition even ordinal and continuous variables can be predicted.

Usage:

```
kknn(formula = formula(train), train, test, na.action =
 na.omit(), k = 7, distance = 2, kernel = "triangular",
 ykernel = 0, contrasts = c('unordered' = "contr.dummy",
 'ordered' = "contr.ordinal"))
```

Arguments:

formula:	A formula object.
train:	Matrix or data frame of training set cases.
test:	Matrix or data frame of test set cases.
na.action:	A function which indicates what should happen when the data contain 'NA's.
k:	Number of neighbors considered.
distance:	Parameter of Minkowsky distance.
kernel:	Kernel to use. Possible choices are "rectangular" (which is standard unweighted knn) , "triangular", "epanechnikov" (or beta(2,2)), "biweight" (or beta(3,3)), "triweight" (or beta(4,4)), "cos", "inv" and "gaussian".
ykernel:	Window width of an y-kernel, especially for prediction of ordinal classes.

contrasts: A vector containing the 'unordered' and 'ordered' contrasts to use.

Details:

This nearest neighbor method expands knn in several directions. First it can be used not only for classification, but also for regression and ordinal classification. Second it uses kernel functions to weight the neighbors according to their distances. In fact, not only kernel functions but every monotonic decreasing function $f(x)$ for $x > 0$ will work fine.

Value: kknn returns a list-object of class kknn including the components

fitted.values: Vector of predictions.

CL: Matrix of classes of the k nearest neighbors.

W: Matrix of weights of the k nearest neighbors.

D: Matrix of distances of the k nearest neighbors.

prob: Matrix of predicted class probabilities.

response: Type of response variable, one of *continuous*, *nominal* or *ordinal*.

distance: Parameter of Minkowsky distance.

call: The matched call.

terms: The 'terms' object used.

Authors:

Klaus P. Schliep (*K.P.Schliep@massey.ac.nz*)
Klaus Hechenbichler (*hechen@stat.uni-muenchen.de*)

References:

Hechenbichler K. and Schliep K.P. (2004) *Weighted k-Nearest-Neighbor Techniques and Ordinal Classification*, Discussion Paper 399, SFB 386, Ludwig-Maximilians University Munich (*http://www.stat.uni-muenchen.de/sfb386/papers/dsp/paper399.ps*)

SeeAlso: train.kknn, simulation, knn and knn1

Examples:

```
library(kknn)

data(iris)
m <- dim(iris)[1]
val <- sample(1:m, size = round(m/3), replace = FALSE,
   prob = rep(1/m, m))
iris.learn <- iris[-val,]
iris.valid <- iris[val,]
iris.kknn <- kknn(Species~., iris.learn, iris.valid,
```

```
   distance = 1, kernel = "triangular")
summary(iris.kknn)
fit <- fitted(iris.kknn)
table(iris.valid$Species, fit)
pcol <- as.character(as.numeric(iris.valid$Species))
pairs(iris.valid[1:4], pch = pcol, col = c("green3", "red")
   [(iris.valid$Species != fit)+1])

data(ionosphere)
ionosphere.learn <- ionosphere[1:200,]
ionosphere.valid <- ionosphere[-c(1:200),]
fit.kknn <- kknn(class ~ ., ionosphere.learn, ionosphere.valid)
table(ionosphere.valid$class, fit.kknn$fit)
(fit.train1 <- train.kknn(class ~ ., ionosphere.learn,
   kmax = 15, kernel = c("triangular", "rectangular",
   "epanechnikov"), distance = 1))
table(predict(fit.train1, ionosphere.valid),
   ionosphere.valid$class)
(fit.train2 <- train.kknn(class ~ ., ionosphere.learn,
   kmax = 15, kernel = c("triangular", "rectangular",
   "epanechnikov"), distance = 2))
table(predict(fit.train2, ionosphere.valid),
   ionosphere.valid$class)
```

Training kknn (train.kknn)

Description: Training of kknn method via leave-one-out crossvalidation.

Usage:

```
train.kknn(formula, data, kmax = 11, kernel = NULL,
 distance = 2, ykernelmax = 0, contrasts = c('unordered' =
 "contr.dummy", 'ordered' = "contr.ordinal"), ...)
```

Arguments:

formula:	A formula object.
data:	Matrix or data frame.
kmax:	Maximum number of k.
kernel:	Kernel to use. Possible choices are “rectangular“ (which is standard unweighted knn) , “triangular“, “epanechnikov“ (or beta(2,2)), “biweight“ (or beta(3,3)), “triweight“ (or beta(4,4)), “cos“, “inv“ and “gaussian“.

distance: Parameter of Minkowsky distance.

ykernelmax: Maximal window width of an y-kernel for prediction of ordinal classes.

contrasts: A vector containing the 'unordered' and 'ordered' contrasts to use.

...: Further arguments passed to or from other methods.

Value: train.kknn returns a list-object of class train.kknn including the components

MISCLASS: Matrix of misclassification errors.

MEAN.ABS: Matrix of mean absolute errors.

MEAN.SQU: Matrix of mean squared errors.

fitted.values: List of predictions for all combinations of kernel and k.

best.parameters: List containing the best parameter value for kernel and k.

response: Type of response variable, one of *continuous*, *nominal* or *ordinal*.

distance: Parameter of Minkowsky distance.

call: The matched call.

terms: The 'terms' object used.

Author: Klaus P. Schliep (*K.P.Schliep@massey.ac.nz*)

References:

Hechenbichler K. and Schliep K.P. (2004) *Weighted k-Nearest-Neighbor Techniques and Ordinal Classification*, Discussion Paper 399, SFB 386, Ludwig-Maximilians University Munich (*http://www.stat.uni-muenchen.de/sfb386/papers/dsp/paper399.ps*)

SeeAlso: kknn and simulation

Examples:

```
library(kknn)

data(miete)
(train.con <- train.kknn(nmqm ~ wfl + bjkat + zh, data =
   miete, kmax = 25, kernel = c("rectangular", "triangular",
   "epanechnikov", "gaussian", "rank")))
plot(train.con)
(train.ord <- train.kknn(wflkat ~ nm + bjkat + zh, miete,
   kmax = 25, kernel = c("rectangular", "triangular",
   "epanechnikov", "gaussian", "rank")))
plot(train.ord)
(train.nom <- train.kknn(zh ~ wfl + bjkat + nmqm, miete,
   kmax = 25, kernel = c("rectangular", "triangular",
```

```
   "epanechnikov", "gaussian", "rank")))
plot(train.nom)

data(glass)
glass <- glass[,-1]
(fit.glass1 <- train.kknn(Type ~ ., glass, kmax = 15,
   kernel = c("triangular", "rectangular", "epanechnikov"),
   distance = 1))
(fit.glass2 <- train.kknn(Type ~ ., glass, kmax = 15,
   kernel = c("triangular", "rectangular", "epanechnikov"),
   distance = 2))
plot(fit.glass1)
plot(fit.glass2)
```

Crossvalidation procedure to test prediction accuracy (simulation)

Description:

simulation tests prediction accuracy of regression and/or classification techniques via simulation of different test sets.

Usage:

```
simulation(formula, data, runs = 10, train = TRUE,
 k = 11, ykernel = 0 ...)
```

Arguments:

formula:	A formula object.
data:	Matrix or data frame.
runs:	Number of crossvalidation runs.
train:	A logical value. If TRUE the training procedure for selecting optimal values of k and kernel is performed.
k:	Number or maximal number of neighbors considered, dependent of choice for train.
ykernel:	Window width or maximal window width of an y-kernel for ordinal classes, dependent of choice for train.
...:	Further arguments passed to or from other methods.

Value:

A matrix, containing the mean and variance of the misclassification error, the absolute and the squared distances.

Author: Klaus P. Schliep (*K.P.Schliep@massey.ac.nz*)

References:

Hechenbichler K. and Schliep K.P. (2004) *Weighted k-Nearest-Neighbor Techniques and Ordinal Classification*, Discussion Paper 399, SFB 386, Ludwig-Maximilians University Munich (*http://www.stat.uni-muenchen.de/sfb386/papers/dsp/paper399.ps*)

SeeAlso: kknn and train.kknn

Examples:

```
library(kknn)

data(miete)
simulation(nmqm ~ wfl + bjkat + zh, data = miete, runs = 5,
   kernel = "triangular", k = 15)
simulation(wflkat ~ nm + bjkat + zh, data = miete, runs = 5)
simulation(zh ~ wfl + bjkat + nmqm, data = miete, runs = 5)
```

Contrast Matrices (contr.dummy, contr.metric, contr.ordinal)

Description: Returns a matrix of contrasts.

Usage:

```
contr.dummy(n, contrasts = TRUE)
contr.ordinal(n, contrasts = TRUE)
contr.metric(n, contrasts =TRUE)
```

Arguments:

n:	A vector containing levels of a factor, or the number of levels.
contrasts:	A logical value indicating whether contrasts should be computed.

Details:

contr.dummy is standard dummy-coding, contr.metric is the same as as.numeric (makes sense of course only for ordered variables). contr.ordinal computes contrasts for ordinal variables.

Value:

A matrix with n rows and *n-1* columns for contr.ordinal, a matrix with n rows and n columns for contr.dummy and a vector of length n for contr.metric.

Author: Klaus P. Schliep (*K.P.Schliep@massey.ac.nz*)

References:

Hechenbichler K. and Schliep K.P. (2004) *Weighted k-Nearest-Neighbor Techniques and Ordinal Classification*, Discussion Paper 399, SFB 386, Ludwig-Maximilians University Munich (*http://www.stat.uni-muenchen.de/sfb386/papers/dsp/paper399.ps*)

SeeAlso: contrasts, contr.poly and contr.sdif

Examples:

```
contr.metric(5)
contr.ordinal(5)
contr.dummy(5)
```

Additional output functions

print.kknn

summary.kknn

predict.kknn

print.train.kknn

summary.train.kknn

predict.train.kknn

plot.train.kknn

Anhang C

Fehlerraten von $kkNN$ (ausführliche Übersicht)

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	max. 25	rectangular	0.035	2	max. 25	rectangular	0.034
1	max. 25	triweight	0.035	2	max. 25	triweight	0.035
1	max. 25	all kernels	0.035	2	max. 25	all kernels	0.034

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	1	rectangular	0.035	2	1	rectangular	0.043
1	3	rectangular	0.032	2	3	rectangular	0.035
1	5	rectangular	0.033	2	5	rectangular	0.032
1	7	rectangular	0.034	2	7	rectangular	0.031
1	11	rectangular	0.034	2	11	rectangular	0.033
1	21	rectangular	0.039	2	21	rectangular	0.033
1	1	triangular	0.035	2	1	triangular	0.043
1	3	triangular	0.035	2	3	triangular	0.040
1	5	triangular	0.033	2	5	triangular	0.036
1	7	triangular	0.030	2	7	triangular	0.032
1	11	triangular	0.031	2	11	triangular	0.031
1	21	triangular	0.034	2	21	triangular	0.032
1	1	epanechnikov	0.035	2	1	epanechnikov	0.043
1	3	epanechnikov	0.035	2	3	epanechnikov	0.040
1	5	epanechnikov	0.032	2	5	epanechnikov	0.035
1	7	epanechnikov	0.030	2	7	epanechnikov	0.032
1	11	epanechnikov	0.031	2	11	epanechnikov	0.031
1	21	epanechnikov	0.034	2	21	epanechnikov	0.032
1	1	biweight	0.035	2	1	biweight	0.043
1	3	biweight	0.034	2	3	biweight	0.042
1	5	biweight	0.035	2	5	biweight	0.040
1	7	biweight	0.033	2	7	biweight	0.037
1	11	biweight	0.031	2	11	biweight	0.034
1	21	biweight	0.032	2	21	biweight	0.031
1	1	triweight	0.035	2	1	triweight	0.043
1	3	triweight	0.035	2	3	triweight	0.043
1	5	triweight	0.034	2	5	triweight	0.042
1	7	triweight	0.034	2	7	triweight	0.040
1	11	triweight	0.033	2	11	triweight	0.038
1	21	triweight	0.030	2	21	triweight	0.033
1	1	cos	0.035	2	1	cos	0.043
1	3	cos	0.035	2	3	cos	0.040
1	5	cos	0.032	2	5	cos	0.036
1	7	cos	0.030	2	7	cos	0.032
1	11	cos	0.031	2	11	cos	0.031
1	21	cos	0.034	2	21	cos	0.032
1	1	inv	0.035	2	1	inv	0.043
1	3	inv	0.031	2	3	inv	0.035
1	5	inv	0.032	2	5	inv	0.031
1	7	inv	0.033	2	7	inv	0.031
1	11	inv	0.034	2	11	inv	0.033
1	21	inv	0.037	2	21	inv	0.033

Tabelle C.1: Fehlerraten bei den Cancer-Daten (die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	max. 25	rectangular	0.296	2	max. 25	rectangular	0.319
1	max. 25	triweight	0.277	2	max. 25	triweight	0.305
1	max. 25	all kernels	0.285	2	max. 25	all kernels	0.307

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	1	rectangular	0.276	2	1	rectangular	0.306
1	3	rectangular	0.303	2	3	rectangular	0.320
1	5	rectangular	0.327	2	5	rectangular	0.357
1	7	rectangular	0.341	2	7	rectangular	0.356
1	11	rectangular	0.339	2	11	rectangular	0.377
1	21	rectangular	0.379	2	21	rectangular	0.408
1	1	triangular	0.276	2	1	triangular	0.306
1	3	triangular	0.276	2	3	triangular	0.306
1	5	triangular	0.275	2	5	triangular	0.305
1	7	triangular	0.278	2	7	triangular	0.307
1	11	triangular	0.286	2	11	triangular	0.306
1	21	triangular	0.295	2	21	triangular	0.325
1	1	epanechnikov	0.276	2	1	epanechnikov	0.306
1	3	epanechnikov	0.276	2	3	epanechnikov	0.306
1	5	epanechnikov	0.277	2	5	epanechnikov	0.307
1	7	epanechnikov	0.283	2	7	epanechnikov	0.309
1	11	epanechnikov	0.292	2	11	epanechnikov	0.310
1	21	epanechnikov	0.304	2	21	epanechnikov	0.338
1	1	biweight	0.276	2	1	biweight	0.306
1	3	biweight	0.277	2	3	biweight	0.304
1	5	biweight	0.270	2	5	biweight	0.303
1	7	biweight	0.270	2	7	biweight	0.297
1	11	biweight	0.272	2	11	biweight	0.301
1	21	biweight	0.281	2	21	biweight	0.312
1	1	triweight	0.276	2	1	triweight	0.306
1	3	triweight	0.278	2	3	triweight	0.304
1	5	triweight	0.271	2	5	triweight	0.303
1	7	triweight	0.272	2	7	triweight	0.302
1	11	triweight	0.269	2	11	triweight	0.297
1	21	triweight	0.273	2	21	triweight	0.305
1	1	cos	0.276	2	1	cos	0.306
1	3	cos	0.276	2	3	cos	0.306
1	5	cos	0.277	2	5	cos	0.306
1	7	cos	0.280	2	7	cos	0.307
1	11	cos	0.289	2	11	cos	0.308
1	21	cos	0.299	2	21	cos	0.333
1	1	inv	0.276	2	1	inv	0.306
1	3	inv	0.287	2	3	inv	0.308
1	5	inv	0.288	2	5	inv	0.314
1	7	inv	0.300	2	7	inv	0.317
1	11	inv	0.301	2	11	inv	0.335
1	21	inv	0.325	2	21	inv	0.369

Tabelle C.2: Fehlerraten bei den Glass-Daten (die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	max. 25	rectangular	0.094	2	max. 25	rectangular	0.106
1	max. 25	triweight	0.099	2	max. 25	triweight	0.130
1	max. 25	all kernels	0.094	2	max. 25	all kernels	0.107

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	1	rectangular	0.096	2	1	rectangular	0.135
1	3	rectangular	0.111	2	3	rectangular	0.156
1	5	rectangular	0.119	2	5	rectangular	0.162
1	7	rectangular	0.125	2	7	rectangular	0.169
1	11	rectangular	0.143	2	11	rectangular	0.178
1	21	rectangular	0.169	2	21	rectangular	0.228
1	1	triangular	0.096	2	1	triangular	0.135
1	3	triangular	0.098	2	3	triangular	0.134
1	5	triangular	0.099	2	5	triangular	0.134
1	7	triangular	0.102	2	7	triangular	0.135
1	11	triangular	0.102	2	11	triangular	0.146
1	21	triangular	0.118	2	21	triangular	0.171
1	1	epanechnikov	0.096	2	1	epanechnikov	0.135
1	3	epanechnikov	0.098	2	3	epanechnikov	0.134
1	5	epanechnikov	0.099	2	5	epanechnikov	0.135
1	7	epanechnikov	0.102	2	7	epanechnikov	0.137
1	11	epanechnikov	0.104	2	11	epanechnikov	0.148
1	21	epanechnikov	0.121	2	21	epanechnikov	0.174
1	1	biweight	0.096	2	1	biweight	0.135
1	3	biweight	0.099	2	3	biweight	0.134
1	5	biweight	0.099	2	5	biweight	0.131
1	7	biweight	0.101	2	7	biweight	0.128
1	11	biweight	0.101	2	11	biweight	0.129
1	21	biweight	0.101	2	21	biweight	0.139
1	1	triweight	0.096	2	1	triweight	0.135
1	3	triweight	0.098	2	3	triweight	0.134
1	5	triweight	0.099	2	5	triweight	0.133
1	7	triweight	0.098	2	7	triweight	0.129
1	11	triweight	0.100	2	11	triweight	0.126
1	21	triweight	0.099	2	21	triweight	0.128
1	1	cos	0.096	2	1	cos	0.135
1	3	cos	0.099	2	3	cos	0.135
1	5	cos	0.099	2	5	cos	0.134
1	7	cos	0.102	2	7	cos	0.136
1	11	cos	0.102	2	11	cos	0.147
1	21	cos	0.119	2	21	cos	0.172
1	1	inv	0.096	2	1	inv	0.135
1	3	inv	0.110	2	3	inv	0.155
1	5	inv	0.114	2	5	inv	0.159
1	7	inv	0.123	2	7	inv	0.168
1	11	inv	0.139	2	11	inv	0.175
1	21	inv	0.169	2	21	inv	0.211

Tabelle C.3: Fehlerraten bei den Ionosphere-Daten (die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	max. 25	rectangular	0.127	2	max. 25	rectangular	0.139
1	max. 25	triweight	0.119	2	max. 25	triweight	0.124
1	max. 25	all kernels	0.122	2	max. 25	all kernels	0.125

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	1	rectangular	0.122	2	1	rectangular	0.133
1	3	rectangular	0.164	2	3	rectangular	0.172
1	5	rectangular	0.195	2	5	rectangular	0.197
1	7	rectangular	0.238	2	7	rectangular	0.232
1	11	rectangular	0.308	2	11	rectangular	0.310
1	21	rectangular	0.426	2	21	rectangular	0.437
1	1	triangular	0.122	2	1	triangular	0.133
1	3	triangular	0.119	2	3	triangular	0.126
1	5	triangular	0.120	2	5	triangular	0.122
1	7	triangular	0.129	2	7	triangular	0.128
1	11	triangular	0.147	2	11	triangular	0.144
1	21	triangular	0.199	2	21	triangular	0.180
1	1	epanechnikov	0.122	2	1	epanechnikov	0.133
1	3	epanechnikov	0.119	2	3	epanechnikov	0.127
1	5	epanechnikov	0.123	2	5	epanechnikov	0.124
1	7	epanechnikov	0.135	2	7	epanechnikov	0.133
1	11	epanechnikov	0.162	2	11	epanechnikov	0.152
1	21	epanechnikov	0.235	2	21	epanechnikov	0.198
1	1	biweight	0.122	2	1	biweight	0.133
1	3	biweight	0.120	2	3	biweight	0.129
1	5	biweight	0.116	2	5	biweight	0.122
1	7	biweight	0.118	2	7	biweight	0.119
1	11	biweight	0.127	2	11	biweight	0.122
1	21	biweight	0.160	2	21	biweight	0.141
1	1	triweight	0.122	2	1	triweight	0.133
1	3	triweight	0.121	2	3	triweight	0.130
1	5	triweight	0.119	2	5	triweight	0.126
1	7	triweight	0.118	2	7	triweight	0.124
1	11	triweight	0.117	2	11	triweight	0.120
1	21	triweight	0.132	2	21	triweight	0.124
1	1	cos	0.122	2	1	cos	0.133
1	3	cos	0.119	2	3	cos	0.126
1	5	cos	0.122	2	5	cos	0.122
1	7	cos	0.132	2	7	cos	0.130
1	11	cos	0.156	2	11	cos	0.148
1	21	cos	0.221	2	21	cos	0.187
1	1	inv	0.122	2	1	inv	0.133
1	3	inv	0.131	2	3	inv	0.152
1	5	inv	0.150	2	5	inv	0.173
1	7	inv	0.171	2	7	inv	0.196
1	11	inv	0.205	2	11	inv	0.242
1	21	inv	0.259	2	21	inv	0.332

Tabelle C.4: Fehlerraten bei den Soybean-Daten (die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	max. 25	rectangular	0.175	2	max. 25	rectangular	0.172
1	max. 25	triweight	0.205	2	max. 25	triweight	0.202
1	max. 25	all kernels	0.173	2	max. 25	all kernels	0.176

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	1	rectangular	0.248	2	1	rectangular	0.242
1	3	rectangular	0.208	2	3	rectangular	0.214
1	5	rectangular	0.202	2	5	rectangular	0.201
1	7	rectangular	0.195	2	7	rectangular	0.190
1	11	rectangular	0.184	2	11	rectangular	0.184
1	21	rectangular	0.173	2	21	rectangular	0.170
1	31	rectangular	0.164	2	31	rectangular	0.166
1	41	rectangular	0.161	2	41	rectangular	0.163
1	51	rectangular	0.162	2	51	rectangular	0.162
1	101	rectangular	0.165	2	101	rectangular	0.162
1	1	triangular	0.248	2	1	triangular	0.242
1	3	triangular	0.232	2	3	triangular	0.234
1	5	triangular	0.216	2	5	triangular	0.216
1	7	triangular	0.208	2	7	triangular	0.206
1	11	triangular	0.198	2	11	triangular	0.196
1	21	triangular	0.182	2	21	triangular	0.182
1	31	triangular	0.176	2	31	triangular	0.174
1	41	triangular	0.171	2	41	triangular	0.169
1	51	triangular	0.167	2	51	triangular	0.166
1	101	triangular	0.161	2	101	triangular	0.160
1	1	epanechnikov	0.248	2	1	epanechnikov	0.242
1	3	epanechnikov	0.232	2	3	epanechnikov	0.234
1	5	epanechnikov	0.215	2	5	epanechnikov	0.215
1	7	epanechnikov	0.207	2	7	epanechnikov	0.205
1	11	epanechnikov	0.196	2	11	epanechnikov	0.195
1	21	epanechnikov	0.181	2	21	epanechnikov	0.181
1	31	epanechnikov	0.175	2	31	epanechnikov	0.173
1	41	epanechnikov	0.170	2	41	epanechnikov	0.168
1	51	epanechnikov	0.165	2	51	epanechnikov	0.165
1	101	epanechnikov	0.161	2	101	epanechnikov	0.161
1	1	biweight	0.248	2	1	biweight	0.242
1	3	biweight	0.243	2	3	biweight	0.239
1	5	biweight	0.233	2	5	biweight	0.232
1	7	biweight	0.223	2	7	biweight	0.222
1	11	biweight	0.212	2	11	biweight	0.210
1	21	biweight	0.195	2	21	biweight	0.193
1	31	biweight	0.186	2	31	biweight	0.188
1	41	biweight	0.181	2	41	biweight	0.181
1	51	biweight	0.175	2	51	biweight	0.175
1	101	biweight	0.164	2	101	biweight	0.165
1	1	triweight	0.248	2	1	triweight	0.242
1	3	triweight	0.245	2	3	triweight	0.239
1	5	triweight	0.239	2	5	triweight	0.236
1	7	triweight	0.233	2	7	triweight	0.233
1	11	triweight	0.224	2	11	triweight	0.223
1	21	triweight	0.208	2	21	triweight	0.205
1	31	triweight	0.198	2	31	triweight	0.197
1	41	triweight	0.191	2	41	triweight	0.191
1	51	triweight	0.187	2	51	triweight	0.187
1	101	triweight	0.172	2	101	triweight	0.171
1	1	cos	0.248	2	1	cos	0.242
1	3	cos	0.232	2	3	cos	0.234
1	5	cos	0.216	2	5	cos	0.216
1	7	cos	0.208	2	7	cos	0.206
1	11	cos	0.197	2	11	cos	0.196
1	21	cos	0.182	2	21	cos	0.182
1	31	cos	0.176	2	31	cos	0.174
1	41	cos	0.171	2	41	cos	0.169
1	51	cos	0.166	2	51	cos	0.165
1	101	cos	0.161	2	101	cos	0.161
1	1	inv	0.248	2	1	inv	0.242
1	3	inv	0.208	2	3	inv	0.214
1	5	inv	0.203	2	5	inv	0.201
1	7	inv	0.195	2	7	inv	0.190
1	11	inv	0.184	2	11	inv	0.184
1	21	inv	0.172	2	21	inv	0.170
1	31	inv	0.164	2	31	inv	0.167
1	41	inv	0.161	2	41	inv	0.163
1	51	inv	0.162	2	51	inv	0.162
1	101	inv	0.164	2	101	inv	0.162

Tabelle C.5: Fehlerraten bei den Waveform-Daten ohne Störgrößen (die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	max. 25	rectangular	0.182	2	max. 25	rectangular	0.213
1	max. 25	triweight	0.223	2	max. 25	triweight	0.239
1	max. 25	all kernels	0.184	2	max. 25	all kernels	0.210

q	k	Kern	Testfehlerrate	q	k	Kern	Testfehlerrate
1	1	rectangular	0.280	2	1	rectangular	0.290
1	3	rectangular	0.232	2	3	rectangular	0.254
1	5	rectangular	0.220	2	5	rectangular	0.238
1	7	rectangular	0.212	2	7	rectangular	0.229
1	11	rectangular	0.199	2	11	rectangular	0.222
1	21	rectangular	0.186	2	21	rectangular	0.212
1	31	rectangular	0.178	2	31	rectangular	0.199
1	41	rectangular	0.181	2	41	rectangular	0.186
1	51	rectangular	0.175	2	51	rectangular	0.182
1	101	rectangular	0.173	2	101	rectangular	0.182
1	1	triangular	0.280	2	1	triangular	0.290
1	3	triangular	0.262	2	3	triangular	0.274
1	5	triangular	0.237	2	5	triangular	0.256
1	7	triangular	0.222	2	7	triangular	0.242
1	11	triangular	0.210	2	11	triangular	0.228
1	21	triangular	0.194	2	21	triangular	0.214
1	31	triangular	0.186	2	31	triangular	0.206
1	41	triangular	0.182	2	41	triangular	0.198
1	51	triangular	0.178	2	51	triangular	0.193
1	101	triangular	0.170	2	101	triangular	0.181
1	1	epanechnikov	0.280	2	1	epanechnikov	0.290
1	3	epanechnikov	0.262	2	3	epanechnikov	0.275
1	5	epanechnikov	0.236	2	5	epanechnikov	0.255
1	7	epanechnikov	0.221	2	7	epanechnikov	0.242
1	11	epanechnikov	0.209	2	11	epanechnikov	0.226
1	21	epanechnikov	0.193	2	21	epanechnikov	0.214
1	31	epanechnikov	0.187	2	31	epanechnikov	0.206
1	41	epanechnikov	0.181	2	41	epanechnikov	0.198
1	51	epanechnikov	0.178	2	51	epanechnikov	0.193
1	101	epanechnikov	0.171	2	101	epanechnikov	0.181
1	1	biweight	0.280	2	1	biweight	0.290
1	3	biweight	0.273	2	3	biweight	0.281
1	5	biweight	0.256	2	5	biweight	0.271
1	7	biweight	0.248	2	7	biweight	0.264
1	11	biweight	0.230	2	11	biweight	0.249
1	21	biweight	0.208	2	21	biweight	0.226
1	31	biweight	0.197	2	31	biweight	0.219
1	41	biweight	0.191	2	41	biweight	0.212
1	51	biweight	0.186	2	51	biweight	0.206
1	101	biweight	0.176	2	101	biweight	0.190
1	1	triweight	0.280	2	1	triweight	0.290
1	3	triweight	0.278	2	3	triweight	0.285
1	5	triweight	0.270	2	5	triweight	0.278
1	7	triweight	0.260	2	7	triweight	0.271
1	11	triweight	0.247	2	11	triweight	0.262
1	21	triweight	0.226	2	21	triweight	0.245
1	31	triweight	0.214	2	31	triweight	0.232
1	41	triweight	0.205	2	41	triweight	0.223
1	51	triweight	0.198	2	51	triweight	0.217
1	101	triweight	0.184	2	101	triweight	0.202
1	1	cos	0.280	2	1	cos	0.290
1	3	cos	0.262	2	3	cos	0.274
1	5	cos	0.237	2	5	cos	0.255
1	7	cos	0.222	2	7	cos	0.242
1	11	cos	0.210	2	11	cos	0.228
1	21	cos	0.194	2	21	cos	0.214
1	31	cos	0.186	2	31	cos	0.206
1	41	cos	0.182	2	41	cos	0.198
1	51	cos	0.178	2	51	cos	0.193
1	101	cos	0.171	2	101	cos	0.181
1	1	inv	0.280	2	1	inv	0.290
1	3	inv	0.232	2	3	inv	0.254
1	5	inv	0.220	2	5	inv	0.239
1	7	inv	0.212	2	7	inv	0.230
1	11	inv	0.198	2	11	inv	0.222
1	21	inv	0.186	2	21	inv	0.211
1	31	inv	0.178	2	31	inv	0.199
1	41	inv	0.181	2	41	inv	0.186
1	51	inv	0.176	2	51	inv	0.182
1	101	inv	0.174	2	101	inv	0.183

Tabelle C.6: Fehlerraten bei den Waveform-Daten mit Störgrößen (die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.658	0.874	1.338
max. 25	epanechnikov	0.643	0.856	1.318
max. 25	all kernels	0.638	0.837	1.263

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.659	0.929	1.536
3	rectangular	0.687	0.994	1.689
5	rectangular	0.653	0.900	1.449
7	rectangular	0.651	0.895	1.438
11	rectangular	0.649	0.867	1.343
21	rectangular	0.651	0.844	1.248
31	rectangular	0.647	0.820	1.174
41	rectangular	0.639	0.806	1.145
51	rectangular	0.643	0.802	1.121
101	rectangular	0.642	0.812	1.152
1	triangular	0.659	0.929	1.536
3	triangular	0.665	0.932	1.526
5	triangular	0.658	0.909	1.467
7	triangular	0.651	0.887	1.410
11	triangular	0.643	0.861	1.337
21	triangular	0.631	0.823	1.236
31	triangular	0.628	0.809	1.193
41	triangular	0.626	0.801	1.167
51	triangular	0.626	0.795	1.146
101	triangular	0.633	0.797	1.126
1	epanechnikov	0.659	0.929	1.536
3	epanechnikov	0.664	0.930	1.522
5	epanechnikov	0.658	0.909	1.466
7	epanechnikov	0.648	0.883	1.402
11	epanechnikov	0.640	0.855	1.324
21	epanechnikov	0.634	0.827	1.241
31	epanechnikov	0.633	0.816	1.203
41	epanechnikov	0.630	0.807	1.175
51	epanechnikov	0.629	0.799	1.149
101	epanechnikov	0.635	0.799	1.129
1	biweight	0.659	0.929	1.536
3	biweight	0.661	0.930	1.533
5	biweight	0.662	0.925	1.511
7	biweight	0.660	0.916	1.488
11	biweight	0.654	0.893	1.424
21	biweight	0.639	0.852	1.314
31	biweight	0.635	0.835	1.266
41	biweight	0.631	0.821	1.229
51	biweight	0.628	0.813	1.205
101	biweight	0.625	0.792	1.136
1	triweight	0.659	0.929	1.536
3	triweight	0.659	0.929	1.534
5	triweight	0.661	0.927	1.523
7	triweight	0.661	0.924	1.512
11	triweight	0.659	0.914	1.481
21	triweight	0.653	0.889	1.409
31	triweight	0.644	0.862	1.338
41	triweight	0.639	0.847	1.297
51	triweight	0.636	0.837	1.272
101	triweight	0.622	0.800	1.175
1	cos	0.659	0.929	1.536
3	cos	0.664	0.931	1.525
5	cos	0.658	0.909	1.467
7	cos	0.649	0.885	1.405
11	cos	0.643	0.860	1.335
21	cos	0.632	0.825	1.237
31	cos	0.633	0.817	1.207
41	cos	0.629	0.806	1.175
51	cos	0.629	0.800	1.154
101	cos	0.633	0.798	1.128
1	inv	0.659	0.929	1.536
3	inv	0.657	0.911	1.478
5	inv	0.639	0.864	1.360
7	inv	0.636	0.847	1.309
11	inv	0.632	0.831	1.261
21	inv	0.630	0.809	1.186
31	inv	0.628	0.792	1.128
41	inv	0.617	0.775	1.098
51	inv	0.622	0.774	1.079
101	inv	0.624	0.785	1.106

Tabelle C.7: Fehlerraten bei der Kundenumfrage ohne Verwendung von Median oder y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.655	0.782	1.037
max. 25	epanechnikov	0.639	0.769	1.034
max. 25	all kernels	0.636	0.762	1.016

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.659	0.929	1.536
3	rectangular	0.642	0.832	1.238
5	rectangular	0.636	0.802	1.151
7	rectangular	0.641	0.788	1.091
11	rectangular	0.649	0.782	1.049
21	rectangular	0.655	0.777	1.022
31	rectangular	0.655	0.770	1.001
41	rectangular	0.654	0.766	0.989
51	rectangular	0.654	0.764	0.983
101	rectangular	0.655	0.781	1.034
1	triangular	0.659	0.929	1.536
3	triangular	0.658	0.904	1.449
5	triangular	0.644	0.849	1.291
7	triangular	0.632	0.812	1.193
11	triangular	0.623	0.775	1.091
21	triangular	0.634	0.756	1.002
31	triangular	0.642	0.759	0.994
41	triangular	0.644	0.757	0.984
51	triangular	0.645	0.756	0.977
101	triangular	0.647	0.755	0.971
1	epanechnikov	0.659	0.929	1.536
3	epanechnikov	0.657	0.901	1.440
5	epanechnikov	0.643	0.843	1.275
7	epanechnikov	0.632	0.808	1.181
11	epanechnikov	0.626	0.775	1.084
21	epanechnikov	0.640	0.764	1.013
31	epanechnikov	0.645	0.762	0.997
41	epanechnikov	0.646	0.759	0.986
51	epanechnikov	0.645	0.756	0.978
101	epanechnikov	0.648	0.755	0.971
1	biweight	0.659	0.929	1.536
3	biweight	0.658	0.920	1.502
5	biweight	0.653	0.890	1.413
7	biweight	0.648	0.864	1.338
11	biweight	0.637	0.824	1.223
21	biweight	0.625	0.772	1.076
31	biweight	0.630	0.760	1.023
41	biweight	0.636	0.757	1.000
51	biweight	0.638	0.756	0.992
101	biweight	0.647	0.756	0.974
1	triweight	0.659	0.929	1.536
3	triweight	0.658	0.925	1.522
5	triweight	0.657	0.911	1.477
7	triweight	0.652	0.892	1.424
11	triweight	0.644	0.859	1.332
21	triweight	0.635	0.812	1.189
31	triweight	0.629	0.785	1.111
41	triweight	0.626	0.768	1.061
51	triweight	0.627	0.758	1.028
101	triweight	0.640	0.754	0.982
1	cos	0.659	0.929	1.536
3	cos	0.658	0.903	1.446
5	cos	0.643	0.848	1.288
7	cos	0.631	0.810	1.189
11	cos	0.624	0.775	1.089
21	cos	0.638	0.761	1.010
31	cos	0.643	0.761	0.998
41	cos	0.645	0.758	0.985
51	cos	0.646	0.757	0.981
101	cos	0.647	0.755	0.972
1	inv	0.659	0.929	1.536
3	inv	0.635	0.825	1.230
5	inv	0.626	0.789	1.133
7	inv	0.628	0.771	1.067
11	inv	0.634	0.762	1.021
21	inv	0.638	0.756	0.993
31	inv	0.639	0.750	0.972
41	inv	0.637	0.744	0.958
51	inv	0.638	0.742	0.951
101	inv	0.639	0.757	0.993

Tabelle C.8: Fehlerraten bei der Kundenumfrage mit Verwendung des Medians, aber ohne y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.660	0.772	0.996
max. 25	epanechnikov	0.652	0.765	0.991
max. 25	all kernels	0.650	0.758	0.973

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.655	0.871	1.340
3	rectangular	0.661	0.793	1.059
5	rectangular	0.656	0.778	1.021
7	rectangular	0.659	0.772	0.999
11	rectangular	0.662	0.774	0.999
21	rectangular	0.661	0.767	0.981
31	rectangular	0.659	0.760	0.962
41	rectangular	0.661	0.760	0.957
51	rectangular	0.657	0.752	0.941
101	rectangular	0.669	0.770	0.973
1	triangular	0.655	0.871	1.340
3	triangular	0.656	0.790	1.059
5	triangular	0.651	0.778	1.033
7	triangular	0.650	0.771	1.013
11	triangular	0.649	0.763	0.991
21	triangular	0.648	0.756	0.972
31	triangular	0.651	0.757	0.968
41	triangular	0.650	0.753	0.958
51	triangular	0.648	0.747	0.943
101	triangular	0.652	0.743	0.925
1	epanechnikov	0.655	0.871	1.340
3	epanechnikov	0.656	0.789	1.058
5	epanechnikov	0.651	0.778	1.032
7	epanechnikov	0.650	0.770	1.009
11	epanechnikov	0.651	0.765	0.993
21	epanechnikov	0.650	0.757	0.971
31	epanechnikov	0.653	0.758	0.969
41	epanechnikov	0.651	0.754	0.958
51	epanechnikov	0.650	0.749	0.946
101	epanechnikov	0.654	0.746	0.930
1	biweight	0.655	0.871	1.340
3	biweight	0.653	0.788	1.060
5	biweight	0.656	0.785	1.043
7	biweight	0.653	0.781	1.036
11	biweight	0.651	0.773	1.017
21	biweight	0.648	0.762	0.990
31	biweight	0.651	0.761	0.981
41	biweight	0.649	0.758	0.974
51	biweight	0.649	0.755	0.966
101	biweight	0.650	0.746	0.939
1	triweight	0.655	0.871	1.340
3	triweight	0.651	0.786	1.060
5	triweight	0.655	0.785	1.047
7	triweight	0.655	0.784	1.040
11	triweight	0.654	0.781	1.034
21	triweight	0.651	0.771	1.010
31	triweight	0.649	0.765	0.999
41	triweight	0.648	0.761	0.988
51	triweight	0.647	0.758	0.980
101	triweight	0.651	0.754	0.960
1	cos	0.655	0.871	1.340
3	cos	0.655	0.789	1.058
5	cos	0.651	0.778	1.033
7	cos	0.650	0.770	1.012
11	cos	0.651	0.765	0.994
21	cos	0.649	0.757	0.972
31	cos	0.652	0.758	0.969
41	cos	0.651	0.754	0.959
51	cos	0.649	0.748	0.946
101	cos	0.653	0.745	0.928
1	inv	0.655	0.871	1.340
3	inv	0.652	0.783	1.046
5	inv	0.648	0.768	1.008
7	inv	0.647	0.757	0.979
11	inv	0.653	0.762	0.981
21	inv	0.649	0.754	0.963
31	inv	0.650	0.748	0.946
41	inv	0.649	0.746	0.939
51	inv	0.646	0.739	0.924
101	inv	0.654	0.750	0.943

Tabelle C.9: Fehlerraten bei der Kundenumfrage mit Verwendung von Median und y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.648	0.862	1.329
max. 25	epanechnikov	0.632	0.839	1.293
max. 25	all kernels	0.635	0.836	1.273

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.663	0.941	1.575
3	rectangular	0.679	0.992	1.714
5	rectangular	0.652	0.899	1.452
7	rectangular	0.649	0.888	1.422
11	rectangular	0.642	0.852	1.313
21	rectangular	0.642	0.829	1.222
31	rectangular	0.643	0.818	1.181
41	rectangular	0.644	0.809	1.144
51	rectangular	0.640	0.805	1.137
101	rectangular	0.638	0.807	1.146
1	triangular	0.663	0.941	1.575
3	triangular	0.664	0.939	1.566
5	triangular	0.655	0.913	1.492
7	triangular	0.646	0.889	1.432
11	triangular	0.632	0.855	1.350
21	triangular	0.625	0.822	1.251
31	triangular	0.617	0.798	1.189
41	triangular	0.613	0.787	1.157
51	triangular	0.613	0.778	1.125
101	triangular	0.626	0.790	1.121
1	epanechnikov	0.663	0.941	1.575
3	epanechnikov	0.663	0.939	1.566
5	epanechnikov	0.654	0.911	1.491
7	epanechnikov	0.645	0.888	1.430
11	epanechnikov	0.631	0.852	1.342
21	epanechnikov	0.624	0.819	1.245
31	epanechnikov	0.619	0.801	1.192
41	epanechnikov	0.618	0.795	1.168
51	epanechnikov	0.619	0.787	1.137
101	epanechnikov	0.628	0.794	1.131
1	biweight	0.663	0.941	1.575
3	biweight	0.661	0.939	1.571
5	biweight	0.663	0.937	1.559
7	biweight	0.662	0.930	1.533
11	biweight	0.652	0.905	1.471
21	biweight	0.638	0.865	1.367
31	biweight	0.629	0.841	1.307
41	biweight	0.625	0.826	1.266
51	biweight	0.620	0.813	1.234
101	biweight	0.611	0.779	1.132
1	triweight	0.663	0.941	1.575
3	triweight	0.662	0.942	1.576
5	triweight	0.663	0.939	1.568
7	triweight	0.661	0.935	1.556
11	triweight	0.660	0.928	1.533
21	triweight	0.653	0.907	1.475
31	triweight	0.646	0.885	1.416
41	triweight	0.640	0.868	1.372
51	triweight	0.635	0.854	1.337
101	triweight	0.618	0.808	1.220
1	cos	0.663	0.941	1.575
3	cos	0.664	0.939	1.566
5	cos	0.655	0.912	1.491
7	cos	0.646	0.888	1.430
11	cos	0.632	0.855	1.350
21	cos	0.625	0.822	1.252
31	cos	0.618	0.800	1.192
41	cos	0.616	0.792	1.164
51	cos	0.617	0.784	1.135
101	cos	0.628	0.794	1.130
1	inv	0.663	0.941	1.575
3	inv	0.651	0.911	1.501
5	inv	0.637	0.863	1.366
7	inv	0.629	0.842	1.313
11	inv	0.621	0.816	1.242
21	inv	0.625	0.802	1.172
31	inv	0.623	0.788	1.133
41	inv	0.626	0.782	1.101
51	inv	0.622	0.779	1.097
101	inv	0.622	0.783	1.105

Tabelle C.10: Fehlerraten bei der Kundenumfrage ohne Verwendung von Median oder y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.652	0.778	1.031
max. 25	epanechnikov	0.629	0.764	1.044
max. 25	all kernels	0.632	0.756	1.007

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.663	0.941	1.575
3	rectangular	0.633	0.834	1.273
5	rectangular	0.635	0.800	1.149
7	rectangular	0.637	0.783	1.088
11	rectangular	0.641	0.774	1.044
21	rectangular	0.649	0.768	1.006
31	rectangular	0.652	0.767	0.998
41	rectangular	0.657	0.770	0.996
51	rectangular	0.652	0.766	0.994
101	rectangular	0.652	0.778	1.031
1	triangular	0.663	0.941	1.575
3	triangular	0.659	0.919	1.504
5	triangular	0.637	0.850	1.314
7	triangular	0.628	0.813	1.212
11	triangular	0.619	0.777	1.109
21	triangular	0.624	0.753	1.018
31	triangular	0.630	0.749	0.989
41	triangular	0.638	0.752	0.981
51	triangular	0.640	0.750	0.969
101	triangular	0.641	0.748	0.964
1	epanechnikov	0.663	0.941	1.575
3	epanechnikov	0.659	0.918	1.503
5	epanechnikov	0.635	0.845	1.305
7	epanechnikov	0.628	0.811	1.207
11	epanechnikov	0.620	0.776	1.104
21	epanechnikov	0.630	0.758	1.020
31	epanechnikov	0.636	0.755	0.992
41	epanechnikov	0.641	0.756	0.986
51	epanechnikov	0.641	0.753	0.976
101	epanechnikov	0.642	0.751	0.969
1	biweight	0.663	0.941	1.575
3	biweight	0.660	0.932	1.547
5	biweight	0.657	0.911	1.481
7	biweight	0.652	0.887	1.407
11	biweight	0.642	0.843	1.281
21	biweight	0.626	0.784	1.121
31	biweight	0.621	0.761	1.054
41	biweight	0.622	0.752	1.019
51	biweight	0.624	0.748	1.002
101	biweight	0.635	0.747	0.969
1	triweight	0.663	0.941	1.575
3	triweight	0.662	0.938	1.565
5	triweight	0.661	0.927	1.527
7	triweight	0.657	0.914	1.493
11	triweight	0.651	0.889	1.416
21	triweight	0.640	0.839	1.274
31	triweight	0.633	0.804	1.173
41	triweight	0.630	0.787	1.120
51	triweight	0.626	0.773	1.081
101	triweight	0.623	0.746	0.997
1	cos	0.663	0.941	1.575
3	cos	0.659	0.919	1.503
5	cos	0.637	0.849	1.313
7	cos	0.628	0.812	1.211
11	cos	0.620	0.777	1.109
21	cos	0.628	0.757	1.021
31	cos	0.635	0.755	0.995
41	cos	0.641	0.756	0.986
51	cos	0.641	0.752	0.973
101	cos	0.642	0.751	0.969
1	inv	0.663	0.941	1.575
3	inv	0.625	0.824	1.256
5	inv	0.623	0.785	1.126
7	inv	0.623	0.766	1.062
11	inv	0.627	0.756	1.018
21	inv	0.636	0.752	0.983
31	inv	0.637	0.749	0.973
41	inv	0.641	0.750	0.968
51	inv	0.635	0.745	0.966
101	inv	0.637	0.757	0.995

Tabelle C.11: Fehlerraten bei der Kundenumfrage mit Verwendung des Medians, aber ohne y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.655	0.769	0.996
max. 25	epanechnikov	0.651	0.770	1.007
max. 25	all kernels	0.649	0.759	0.979

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.656	0.875	1.352
3	rectangular	0.650	0.784	1.055
5	rectangular	0.657	0.777	1.018
7	rectangular	0.662	0.780	1.018
11	rectangular	0.658	0.774	1.006
21	rectangular	0.659	0.768	0.985
31	rectangular	0.657	0.761	0.969
41	rectangular	0.661	0.762	0.964
51	rectangular	0.656	0.754	0.949
101	rectangular	0.662	0.764	0.968
1	triangular	0.656	0.875	1.352
3	triangular	0.651	0.787	1.059
5	triangular	0.647	0.774	1.029
7	triangular	0.647	0.770	1.015
11	triangular	0.649	0.764	0.994
21	triangular	0.650	0.761	0.983
31	triangular	0.649	0.758	0.977
41	triangular	0.650	0.755	0.965
51	triangular	0.648	0.750	0.953
101	triangular	0.649	0.742	0.928
1	epanechnikov	0.656	0.875	1.352
3	epanechnikov	0.651	0.786	1.058
5	epanechnikov	0.646	0.773	1.026
7	epanechnikov	0.648	0.771	1.015
11	epanechnikov	0.651	0.766	0.996
21	epanechnikov	0.652	0.763	0.984
31	epanechnikov	0.651	0.759	0.976
41	epanechnikov	0.651	0.756	0.966
51	epanechnikov	0.650	0.751	0.955
101	epanechnikov	0.650	0.743	0.929
1	biweight	0.656	0.875	1.352
3	biweight	0.650	0.788	1.066
5	biweight	0.652	0.784	1.049
7	biweight	0.652	0.782	1.043
11	biweight	0.648	0.772	1.020
21	biweight	0.649	0.767	1.003
31	biweight	0.648	0.762	0.991
41	biweight	0.649	0.761	0.985
51	biweight	0.651	0.760	0.980
101	biweight	0.646	0.749	0.953
1	triweight	0.656	0.875	1.352
3	triweight	0.648	0.788	1.069
5	triweight	0.652	0.786	1.054
7	triweight	0.653	0.784	1.047
11	triweight	0.652	0.782	1.041
21	triweight	0.648	0.773	1.023
31	triweight	0.650	0.771	1.013
41	triweight	0.650	0.768	1.005
51	triweight	0.648	0.765	0.999
101	triweight	0.647	0.758	0.978
1	cos	0.656	0.875	1.352
3	cos	0.651	0.786	1.058
5	cos	0.647	0.774	1.029
7	cos	0.648	0.770	1.016
11	cos	0.650	0.765	0.995
21	cos	0.651	0.762	0.984
31	cos	0.651	0.760	0.977
41	cos	0.651	0.755	0.965
51	cos	0.649	0.751	0.954
101	cos	0.650	0.743	0.930
1	inv	0.656	0.875	1.352
3	inv	0.640	0.775	1.046
5	inv	0.646	0.766	1.007
7	inv	0.652	0.767	0.997
11	inv	0.649	0.762	0.987
21	inv	0.647	0.754	0.969
31	inv	0.649	0.751	0.956
41	inv	0.648	0.748	0.946
51	inv	0.645	0.740	0.930
101	inv	0.652	0.751	0.950

Tabelle C.12: Fehlerraten bei der Kundenumfrage mit Verwendung von Median und y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.372	0.486	0.739
max. 25	epanechnikov	0.376	0.490	0.746
max. 25	all kernels	0.365	0.473	0.715

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.388	0.504	0.766
3	rectangular	0.400	0.545	0.882
5	rectangular	0.379	0.488	0.730
7	rectangular	0.378	0.493	0.749
11	rectangular	0.364	0.473	0.720
21	rectangular	0.375	0.495	0.754
31	rectangular	0.390	0.531	0.834
41	rectangular	0.399	0.551	0.879
51	rectangular	0.398	0.550	0.874
101	rectangular	0.407	0.565	0.900
1	triangular	0.388	0.504	0.766
3	triangular	0.380	0.492	0.742
5	triangular	0.374	0.482	0.724
7	triangular	0.374	0.481	0.721
11	triangular	0.363	0.462	0.679
21	triangular	0.357	0.458	0.680
31	triangular	0.362	0.466	0.695
41	triangular	0.370	0.481	0.723
51	triangular	0.379	0.502	0.765
101	triangular	0.401	0.556	0.884
1	epanechnikov	0.388	0.504	0.766
3	epanechnikov	0.380	0.492	0.743
5	epanechnikov	0.376	0.484	0.724
7	epanechnikov	0.372	0.481	0.723
11	epanechnikov	0.363	0.464	0.685
21	epanechnikov	0.359	0.461	0.684
31	epanechnikov	0.365	0.472	0.704
41	epanechnikov	0.373	0.489	0.740
51	epanechnikov	0.385	0.515	0.792
101	epanechnikov	0.401	0.558	0.889
1	biweight	0.388	0.504	0.766
3	biweight	0.385	0.498	0.752
5	biweight	0.377	0.488	0.738
7	biweight	0.375	0.483	0.725
11	biweight	0.372	0.476	0.708
21	biweight	0.356	0.454	0.672
31	biweight	0.354	0.456	0.681
41	biweight	0.359	0.461	0.683
51	biweight	0.365	0.472	0.706
101	biweight	0.397	0.541	0.845
1	triweight	0.388	0.504	0.766
3	triweight	0.386	0.500	0.757
5	triweight	0.380	0.493	0.748
7	triweight	0.375	0.485	0.734
11	triweight	0.373	0.481	0.722
21	triweight	0.362	0.462	0.686
31	triweight	0.356	0.455	0.677
41	triweight	0.352	0.450	0.668
51	triweight	0.354	0.454	0.672
101	triweight	0.380	0.503	0.766
1	cos	0.388	0.504	0.766
3	cos	0.380	0.492	0.743
5	cos	0.375	0.484	0.725
7	cos	0.373	0.480	0.719
11	cos	0.365	0.465	0.685
21	cos	0.357	0.459	0.683
31	cos	0.363	0.469	0.699
41	cos	0.371	0.485	0.731
51	cos	0.383	0.509	0.779
101	cos	0.402	0.558	0.889
1	inv	0.388	0.504	0.766
3	inv	0.370	0.477	0.717
5	inv	0.375	0.478	0.705
7	inv	0.363	0.464	0.689
11	inv	0.354	0.454	0.676
21	inv	0.363	0.473	0.714
31	inv	0.377	0.502	0.771
41	inv	0.387	0.522	0.811
51	inv	0.388	0.532	0.837
101	inv	0.400	0.555	0.882

Tabelle C.13: Fehlerraten bei den Herzerkrankungs-Daten ohne Verwendung von Median oder y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.367	0.435	0.574
max. 25	epanechnikov	0.371	0.443	0.593
max. 25	all kernels	0.368	0.434	0.569

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.388	0.504	0.766
3	rectangular	0.371	0.461	0.658
5	rectangular	0.367	0.440	0.594
7	rectangular	0.364	0.433	0.576
11	rectangular	0.361	0.423	0.550
21	rectangular	0.369	0.433	0.562
31	rectangular	0.379	0.458	0.618
41	rectangular	0.391	0.482	0.666
51	rectangular	0.396	0.495	0.697
101	rectangular	0.398	0.524	0.781
1	triangular	0.388	0.504	0.766
3	triangular	0.378	0.484	0.720
5	triangular	0.367	0.458	0.657
7	triangular	0.368	0.447	0.613
11	triangular	0.362	0.427	0.561
21	triangular	0.358	0.416	0.534
31	triangular	0.364	0.423	0.541
41	triangular	0.364	0.423	0.543
51	triangular	0.370	0.433	0.562
101	triangular	0.395	0.497	0.702
1	epanechnikov	0.388	0.504	0.766
3	epanechnikov	0.377	0.482	0.715
5	epanechnikov	0.367	0.456	0.649
7	epanechnikov	0.368	0.445	0.609
11	epanechnikov	0.362	0.426	0.558
21	epanechnikov	0.362	0.420	0.539
31	epanechnikov	0.364	0.422	0.540
41	epanechnikov	0.367	0.428	0.550
51	epanechnikov	0.376	0.443	0.579
101	epanechnikov	0.395	0.504	0.723
1	biweight	0.388	0.504	0.766
3	biweight	0.383	0.494	0.741
5	biweight	0.375	0.478	0.707
7	biweight	0.369	0.463	0.669
11	biweight	0.368	0.445	0.607
21	biweight	0.360	0.420	0.542
31	biweight	0.356	0.414	0.530
41	biweight	0.360	0.418	0.537
51	biweight	0.363	0.422	0.541
101	biweight	0.387	0.470	0.635
1	triweight	0.388	0.504	0.766
3	triweight	0.384	0.497	0.749
5	triweight	0.379	0.486	0.723
7	triweight	0.373	0.474	0.697
11	triweight	0.372	0.463	0.661
21	triweight	0.362	0.433	0.582
31	triweight	0.359	0.421	0.549
41	triweight	0.356	0.413	0.529
51	triweight	0.357	0.415	0.534
101	triweight	0.372	0.439	0.573
1	cos	0.388	0.504	0.766
3	cos	0.378	0.484	0.721
5	cos	0.368	0.458	0.653
7	cos	0.368	0.447	0.613
11	cos	0.361	0.425	0.556
21	cos	0.360	0.418	0.536
31	cos	0.363	0.422	0.539
41	cos	0.365	0.425	0.545
51	cos	0.373	0.438	0.568
101	cos	0.396	0.501	0.714
1	inv	0.388	0.504	0.766
3	inv	0.367	0.454	0.643
5	inv	0.364	0.437	0.590
7	inv	0.362	0.432	0.578
11	inv	0.356	0.419	0.547
21	inv	0.366	0.428	0.551
31	inv	0.371	0.441	0.582
41	inv	0.382	0.458	0.610
51	inv	0.388	0.474	0.647
101	inv	0.395	0.516	0.762

Tabelle C.14: Fehlerraten bei den Herzerkrankungs-Daten mit Verwendung des Medians, aber ohne y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.415	0.478	0.603
max. 25	epanechnikov	0.424	0.484	0.606
max. 25	all kernels	0.414	0.475	0.596

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.413	0.492	0.661
3	rectangular	0.415	0.475	0.595
5	rectangular	0.447	0.497	0.598
7	rectangular	0.461	0.512	0.615
11	rectangular	0.510	0.554	0.642
21	rectangular	0.615	0.664	0.761
31	rectangular	0.642	0.700	0.815
41	rectangular	0.656	0.723	0.858
51	rectangular	0.661	0.738	0.892
101	rectangular	0.662	0.778	1.009
1	triangular	0.413	0.492	0.661
3	triangular	0.418	0.475	0.589
5	triangular	0.443	0.493	0.594
7	triangular	0.459	0.508	0.605
11	triangular	0.509	0.555	0.648
21	triangular	0.612	0.655	0.742
31	triangular	0.637	0.682	0.771
41	triangular	0.643	0.689	0.781
51	triangular	0.646	0.695	0.793
101	triangular	0.659	0.734	0.886
1	epanechnikov	0.413	0.492	0.661
3	epanechnikov	0.418	0.475	0.589
5	epanechnikov	0.444	0.494	0.593
7	epanechnikov	0.461	0.509	0.606
11	epanechnikov	0.509	0.554	0.644
21	epanechnikov	0.611	0.654	0.740
31	epanechnikov	0.639	0.684	0.774
41	epanechnikov	0.645	0.692	0.786
51	epanechnikov	0.647	0.697	0.798
101	epanechnikov	0.662	0.747	0.918
1	biweight	0.413	0.492	0.661
3	biweight	0.419	0.476	0.591
5	biweight	0.439	0.492	0.596
7	biweight	0.459	0.511	0.613
11	biweight	0.511	0.558	0.654
21	biweight	0.613	0.656	0.743
31	biweight	0.637	0.681	0.769
41	biweight	0.643	0.687	0.775
51	biweight	0.645	0.690	0.781
101	biweight	0.655	0.713	0.831
1	triweight	0.413	0.492	0.661
3	triweight	0.421	0.480	0.598
5	triweight	0.441	0.497	0.608
7	triweight	0.457	0.511	0.618
11	triweight	0.509	0.560	0.660
21	triweight	0.613	0.657	0.746
31	triweight	0.640	0.683	0.768
41	triweight	0.644	0.687	0.773
51	triweight	0.643	0.687	0.775
101	triweight	0.646	0.697	0.798
1	cos	0.413	0.492	0.661
3	cos	0.418	0.475	0.589
5	cos	0.444	0.494	0.594
7	cos	0.460	0.509	0.606
11	cos	0.510	0.555	0.647
21	cos	0.612	0.655	0.740
31	cos	0.638	0.683	0.772
41	cos	0.644	0.691	0.784
51	cos	0.647	0.696	0.794
101	cos	0.661	0.742	0.904
1	inv	0.413	0.492	0.661
3	inv	0.416	0.473	0.586
5	inv	0.444	0.493	0.591
7	inv	0.457	0.505	0.600
11	inv	0.508	0.551	0.637
21	inv	0.614	0.659	0.748
31	inv	0.637	0.688	0.790
41	inv	0.646	0.702	0.813
51	inv	0.651	0.712	0.834
101	inv	0.664	0.763	0.962

Tabelle C.15: Fehlerraten bei den Herzerkrankungs-Daten mit Verwendung von Median und y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.381	0.495	0.749
max. 25	epanechnikov	0.384	0.499	0.751
max. 25	all kernels	0.373	0.481	0.719

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.392	0.512	0.793
3	rectangular	0.392	0.532	0.863
5	rectangular	0.378	0.484	0.720
7	rectangular	0.373	0.480	0.720
11	rectangular	0.378	0.492	0.743
21	rectangular	0.375	0.493	0.745
31	rectangular	0.387	0.524	0.820
41	rectangular	0.397	0.551	0.883
51	rectangular	0.398	0.552	0.878
101	rectangular	0.409	0.560	0.879
1	triangular	0.392	0.512	0.793
3	triangular	0.382	0.498	0.769
5	triangular	0.369	0.481	0.743
7	triangular	0.365	0.472	0.716
11	triangular	0.363	0.464	0.691
21	triangular	0.364	0.464	0.685
31	triangular	0.363	0.466	0.691
41	triangular	0.369	0.480	0.720
51	triangular	0.375	0.493	0.749
101	triangular	0.397	0.539	0.840
1	epanechnikov	0.392	0.512	0.793
3	epanechnikov	0.381	0.496	0.767
5	epanechnikov	0.369	0.483	0.750
7	epanechnikov	0.364	0.470	0.711
11	epanechnikov	0.364	0.464	0.690
21	epanechnikov	0.366	0.468	0.692
31	epanechnikov	0.365	0.471	0.702
41	epanechnikov	0.373	0.487	0.732
51	epanechnikov	0.380	0.505	0.774
101	epanechnikov	0.400	0.546	0.852
1	biweight	0.392	0.512	0.793
3	biweight	0.388	0.507	0.786
5	biweight	0.376	0.489	0.752
7	biweight	0.371	0.480	0.734
11	biweight	0.366	0.473	0.719
21	biweight	0.362	0.465	0.697
31	biweight	0.362	0.463	0.687
41	biweight	0.360	0.459	0.678
51	biweight	0.358	0.460	0.683
101	biweight	0.385	0.514	0.790
1	triweight	0.392	0.512	0.793
3	triweight	0.390	0.510	0.792
5	triweight	0.381	0.499	0.775
7	triweight	0.377	0.491	0.757
11	triweight	0.373	0.482	0.735
21	triweight	0.366	0.473	0.720
31	triweight	0.362	0.463	0.690
41	triweight	0.361	0.461	0.687
51	triweight	0.357	0.456	0.679
101	triweight	0.366	0.477	0.718
1	cos	0.392	0.512	0.793
3	cos	0.383	0.498	0.769
5	cos	0.370	0.483	0.744
7	cos	0.365	0.472	0.718
11	cos	0.363	0.464	0.693
21	cos	0.365	0.466	0.689
31	cos	0.363	0.467	0.696
41	cos	0.369	0.481	0.721
51	cos	0.378	0.499	0.760
101	cos	0.399	0.544	0.851
1	inv	0.392	0.512	0.793
3	inv	0.368	0.480	0.738
5	inv	0.373	0.479	0.719
7	inv	0.367	0.467	0.687
11	inv	0.367	0.474	0.712
21	inv	0.368	0.479	0.720
31	inv	0.381	0.507	0.776
41	inv	0.390	0.530	0.831
51	inv	0.393	0.538	0.846
101	inv	0.402	0.551	0.864

Tabelle C.16: Fehlerraten bei den Herzerkrankungs-Daten ohne Verwendung von Median oder y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.371	0.440	0.583
max. 25	epanechnikov	0.372	0.442	0.588
max. 25	all kernels	0.372	0.442	0.588

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.392	0.512	0.793
3	rectangular	0.371	0.462	0.659
5	rectangular	0.373	0.448	0.602
7	rectangular	0.368	0.436	0.576
11	rectangular	0.364	0.427	0.558
21	rectangular	0.374	0.436	0.562
31	rectangular	0.382	0.459	0.616
41	rectangular	0.393	0.484	0.668
51	rectangular	0.397	0.502	0.713
101	rectangular	0.396	0.520	0.772
1	triangular	0.392	0.512	0.793
3	triangular	0.381	0.488	0.733
5	triangular	0.368	0.463	0.675
7	triangular	0.364	0.447	0.627
11	triangular	0.365	0.434	0.577
21	triangular	0.364	0.425	0.547
31	triangular	0.362	0.419	0.535
41	triangular	0.367	0.427	0.548
51	triangular	0.371	0.435	0.565
101	triangular	0.395	0.494	0.695
1	epanechnikov	0.392	0.512	0.793
3	epanechnikov	0.379	0.484	0.727
5	epanechnikov	0.366	0.461	0.673
7	epanechnikov	0.367	0.447	0.618
11	epanechnikov	0.366	0.434	0.572
21	epanechnikov	0.365	0.425	0.547
31	epanechnikov	0.365	0.423	0.540
41	epanechnikov	0.368	0.431	0.557
51	epanechnikov	0.376	0.445	0.582
101	epanechnikov	0.395	0.501	0.716
1	biweight	0.392	0.512	0.793
3	biweight	0.387	0.501	0.766
5	biweight	0.374	0.477	0.713
7	biweight	0.368	0.463	0.676
11	biweight	0.366	0.451	0.638
21	biweight	0.365	0.432	0.569
31	biweight	0.366	0.426	0.549
41	biweight	0.364	0.420	0.534
51	biweight	0.361	0.418	0.534
101	biweight	0.381	0.457	0.608
1	triweight	0.392	0.512	0.793
3	triweight	0.388	0.506	0.782
5	triweight	0.379	0.490	0.747
7	triweight	0.375	0.478	0.714
11	triweight	0.370	0.465	0.677
21	triweight	0.367	0.447	0.619
31	triweight	0.365	0.436	0.582
41	triweight	0.365	0.429	0.562
51	triweight	0.363	0.423	0.544
101	triweight	0.363	0.424	0.546
1	cos	0.392	0.512	0.793
3	cos	0.382	0.488	0.732
5	cos	0.368	0.463	0.674
7	cos	0.364	0.447	0.624
11	cos	0.366	0.434	0.574
21	cos	0.365	0.425	0.547
31	cos	0.363	0.420	0.536
41	cos	0.368	0.429	0.552
51	cos	0.373	0.438	0.570
101	cos	0.395	0.497	0.702
1	inv	0.392	0.512	0.793
3	inv	0.372	0.463	0.662
5	inv	0.373	0.449	0.609
7	inv	0.370	0.440	0.584
11	inv	0.364	0.427	0.558
21	inv	0.372	0.433	0.558
31	inv	0.374	0.445	0.587
41	inv	0.388	0.470	0.636
51	inv	0.393	0.485	0.673
101	inv	0.392	0.514	0.762

Tabelle C.17: Fehlerraten bei den Herzerkrankungs-Daten mit Verwendung des Medians, aber ohne y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.415	0.475	0.597
max. 25	epanechnikov	0.425	0.484	0.603
max. 25	all kernels	0.410	0.467	0.584

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.415	0.497	0.678
3	rectangular	0.410	0.467	0.581
5	rectangular	0.441	0.493	0.596
7	rectangular	0.462	0.511	0.609
11	rectangular	0.513	0.559	0.653
21	rectangular	0.611	0.661	0.761
31	rectangular	0.648	0.705	0.820
41	rectangular	0.658	0.726	0.864
51	rectangular	0.663	0.742	0.898
101	rectangular	0.658	0.774	1.005
1	triangular	0.415	0.497	0.678
3	triangular	0.411	0.465	0.575
5	triangular	0.432	0.486	0.593
7	triangular	0.456	0.506	0.606
11	triangular	0.506	0.553	0.647
21	triangular	0.602	0.648	0.739
31	triangular	0.640	0.684	0.774
41	triangular	0.646	0.692	0.785
51	triangular	0.647	0.696	0.794
101	triangular	0.660	0.736	0.888
1	epanechnikov	0.415	0.497	0.678
3	epanechnikov	0.410	0.465	0.575
5	epanechnikov	0.433	0.486	0.592
7	epanechnikov	0.456	0.505	0.604
11	epanechnikov	0.507	0.553	0.646
21	epanechnikov	0.604	0.649	0.741
31	epanechnikov	0.641	0.687	0.777
41	epanechnikov	0.646	0.693	0.787
51	epanechnikov	0.648	0.699	0.800
101	epanechnikov	0.662	0.744	0.909
1	biweight	0.415	0.497	0.678
3	biweight	0.412	0.470	0.585
5	biweight	0.432	0.487	0.595
7	biweight	0.457	0.509	0.613
11	biweight	0.506	0.556	0.658
21	biweight	0.602	0.647	0.738
31	biweight	0.639	0.683	0.771
41	biweight	0.642	0.685	0.773
51	biweight	0.645	0.690	0.780
101	biweight	0.651	0.706	0.817
1	triweight	0.415	0.497	0.678
3	triweight	0.412	0.472	0.592
5	triweight	0.435	0.491	0.604
7	triweight	0.455	0.510	0.622
11	triweight	0.505	0.558	0.663
21	triweight	0.602	0.650	0.747
31	triweight	0.637	0.683	0.775
41	triweight	0.641	0.685	0.774
51	triweight	0.643	0.687	0.776
101	triweight	0.646	0.693	0.787
1	cos	0.415	0.497	0.678
3	cos	0.410	0.465	0.575
5	cos	0.432	0.486	0.592
7	cos	0.457	0.507	0.607
11	cos	0.506	0.553	0.646
21	cos	0.603	0.648	0.740
31	cos	0.641	0.686	0.776
41	cos	0.647	0.693	0.786
51	cos	0.648	0.698	0.797
101	cos	0.662	0.740	0.896
1	inv	0.415	0.497	0.678
3	inv	0.412	0.467	0.578
5	inv	0.437	0.488	0.590
7	inv	0.459	0.506	0.600
11	inv	0.511	0.556	0.647
21	inv	0.607	0.653	0.746
31	inv	0.642	0.695	0.800
41	inv	0.649	0.708	0.827
51	inv	0.657	0.725	0.860
101	inv	0.656	0.764	0.979

Tabelle C.18: Fehlerraten bei den Herzerkrankungs-Daten mit Verwendung von Median und y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.660	1.068	2.253
max. 25	epanechnikov	0.641	0.948	1.794
max. 25	all kernels	0.647	0.999	2.005

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.654	0.975	1.868
3	rectangular	0.666	1.046	2.131
5	rectangular	0.645	0.980	1.910
7	rectangular	0.653	0.990	1.920
11	rectangular	0.653	1.011	2.031
21	rectangular	0.643	1.090	2.432
1	triangular	0.654	0.975	1.868
3	triangular	0.650	0.961	1.824
5	triangular	0.644	0.945	1.761
7	triangular	0.633	0.917	1.667
11	triangular	0.633	0.895	1.587
21	triangular	0.639	0.930	1.716
1	epanechnikov	0.654	0.975	1.868
3	epanechnikov	0.649	0.961	1.829
5	epanechnikov	0.638	0.938	1.746
7	epanechnikov	0.627	0.906	1.641
11	epanechnikov	0.633	0.901	1.607
21	epanechnikov	0.641	0.936	1.731
1	biweight	0.654	0.975	1.868
3	biweight	0.651	0.967	1.846
5	biweight	0.643	0.939	1.756
7	biweight	0.647	0.942	1.745
11	biweight	0.633	0.913	1.670
21	biweight	0.631	0.898	1.610
1	triweight	0.654	0.975	1.868
3	triweight	0.651	0.969	1.857
5	triweight	0.648	0.957	1.818
7	triweight	0.641	0.937	1.753
11	triweight	0.642	0.933	1.717
21	triweight	0.625	0.893	1.604
1	cos	0.654	0.975	1.868
3	cos	0.648	0.960	1.823
5	cos	0.640	0.941	1.755
7	cos	0.631	0.911	1.645
11	cos	0.635	0.901	1.603
21	cos	0.642	0.931	1.710
1	inv	0.654	0.975	1.868
3	inv	0.636	0.937	1.736
5	inv	0.634	0.909	1.628
7	inv	0.640	0.903	1.589
11	inv	0.642	0.927	1.706
21	inv	0.642	0.986	1.962

Tabelle C.19: Fehlerraten bei den Scapula-Daten ohne Verwendung von Median oder y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.613	0.838	1.416
max. 25	epanechnikov	0.602	0.798	1.296
max. 25	all kernels	0.610	0.809	1.315

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.654	0.975	1.868
3	rectangular	0.610	0.847	1.447
5	rectangular	0.607	0.820	1.350
7	rectangular	0.623	0.821	1.316
11	rectangular	0.615	0.831	1.380
21	rectangular	0.603	0.860	1.546
1	triangular	0.654	0.975	1.868
3	triangular	0.647	0.937	1.743
5	triangular	0.632	0.874	1.510
7	triangular	0.618	0.820	1.339
11	triangular	0.613	0.804	1.283
21	triangular	0.598	0.798	1.307
1	epanechnikov	0.654	0.975	1.868
3	epanechnikov	0.642	0.927	1.719
5	epanechnikov	0.628	0.865	1.487
7	epanechnikov	0.616	0.818	1.332
11	epanechnikov	0.611	0.800	1.278
21	epanechnikov	0.593	0.805	1.341
1	biweight	0.654	0.975	1.868
3	biweight	0.649	0.961	1.827
5	biweight	0.642	0.916	1.673
7	biweight	0.634	0.877	1.538
11	biweight	0.621	0.825	1.350
21	biweight	0.612	0.800	1.270
1	triweight	0.654	0.975	1.868
3	triweight	0.648	0.964	1.844
5	triweight	0.647	0.942	1.753
7	triweight	0.644	0.918	1.671
11	triweight	0.629	0.862	1.484
21	triweight	0.621	0.818	1.327
1	cos	0.654	0.975	1.868
3	cos	0.644	0.931	1.729
5	cos	0.631	0.869	1.496
7	cos	0.618	0.821	1.341
11	cos	0.608	0.797	1.274
21	cos	0.597	0.803	1.323
1	inv	0.654	0.975	1.868
3	inv	0.608	0.842	1.432
5	inv	0.603	0.811	1.330
7	inv	0.622	0.812	1.288
11	inv	0.607	0.802	1.298
21	inv	0.594	0.823	1.415

Tabelle C.20: Fehlerraten bei den Scapula-Daten mit Verwendung des Medians, aber ohne y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.645	0.832	1.301
max. 25	epanechnikov	0.639	0.808	1.229
max. 25	all kernels	0.641	0.808	1.224

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.649	0.953	1.756
3	rectangular	0.623	0.816	1.296
5	rectangular	0.633	0.801	1.223
7	rectangular	0.645	0.813	1.233
11	rectangular	0.656	0.850	1.344
21	rectangular	0.689	0.929	1.595
1	triangular	0.649	0.953	1.756
3	triangular	0.631	0.853	1.427
5	triangular	0.636	0.818	1.275
7	triangular	0.621	0.783	1.189
11	triangular	0.636	0.793	1.188
21	triangular	0.660	0.837	1.291
1	epanechnikov	0.649	0.953	1.756
3	epanechnikov	0.633	0.854	1.425
5	epanechnikov	0.631	0.811	1.262
7	epanechnikov	0.620	0.780	1.184
11	epanechnikov	0.637	0.793	1.185
21	epanechnikov	0.663	0.845	1.314
1	biweight	0.649	0.953	1.756
3	biweight	0.634	0.882	1.532
5	biweight	0.635	0.846	1.395
7	biweight	0.634	0.813	1.264
11	biweight	0.627	0.786	1.184
21	biweight	0.651	0.818	1.242
1	triweight	0.649	0.953	1.756
3	triweight	0.637	0.903	1.600
5	triweight	0.636	0.865	1.464
7	triweight	0.637	0.844	1.380
11	triweight	0.622	0.793	1.215
21	triweight	0.635	0.795	1.196
1	cos	0.649	0.953	1.756
3	cos	0.633	0.855	1.425
5	cos	0.635	0.816	1.272
7	cos	0.620	0.782	1.187
11	cos	0.636	0.793	1.186
21	cos	0.658	0.838	1.304
1	inv	0.649	0.953	1.756
3	inv	0.624	0.811	1.275
5	inv	0.622	0.789	1.201
7	inv	0.637	0.788	1.168
11	inv	0.650	0.829	1.283
21	inv	0.684	0.900	1.478

Tabelle C.21: Fehlerraten bei den Scapula-Daten mit Verwendung von Median und y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.661	1.035	2.117
max. 25	epanechnikov	0.632	0.911	1.664
max. 25	all kernels	0.653	0.980	1.893

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.631	0.916	1.691
3	rectangular	0.656	1.018	2.036
5	rectangular	0.651	0.976	1.876
7	rectangular	0.649	1.001	1.988
11	rectangular	0.657	1.025	2.091
21	rectangular	0.651	1.116	2.540
1	triangular	0.631	0.916	1.691
3	triangular	0.632	0.913	1.680
5	triangular	0.625	0.906	1.672
7	triangular	0.614	0.882	1.598
11	triangular	0.616	0.870	1.525
21	triangular	0.625	0.901	1.638
1	epanechnikov	0.631	0.916	1.691
3	epanechnikov	0.633	0.915	1.685
5	epanechnikov	0.619	0.897	1.652
7	epanechnikov	0.611	0.875	1.577
11	epanechnikov	0.626	0.882	1.544
21	epanechnikov	0.637	0.925	1.691
1	biweight	0.631	0.916	1.691
3	biweight	0.628	0.908	1.672
5	biweight	0.632	0.916	1.693
7	biweight	0.628	0.912	1.684
11	biweight	0.621	0.890	1.604
21	biweight	0.612	0.865	1.529
1	triweight	0.631	0.916	1.691
3	triweight	0.631	0.913	1.680
5	triweight	0.631	0.911	1.676
7	triweight	0.630	0.911	1.679
11	triweight	0.628	0.909	1.667
21	triweight	0.617	0.885	1.603
1	cos	0.631	0.916	1.691
3	cos	0.634	0.916	1.684
5	cos	0.624	0.905	1.673
7	cos	0.612	0.880	1.594
11	cos	0.621	0.877	1.539
21	cos	0.631	0.911	1.657
1	inv	0.631	0.916	1.691
3	inv	0.622	0.898	1.635
5	inv	0.632	0.904	1.620
7	inv	0.631	0.900	1.607
11	inv	0.636	0.934	1.755
21	inv	0.640	1.024	2.160

Tabelle C.22: Fehlerraten bei den Scapula-Daten ohne Verwendung von Median oder y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.621	0.849	1.432
max. 25	epanechnikov	0.608	0.804	1.293
max. 25	all kernels	0.613	0.809	1.304

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.631	0.916	1.691
3	rectangular	0.617	0.849	1.431
5	rectangular	0.618	0.826	1.344
7	rectangular	0.621	0.827	1.346
11	rectangular	0.606	0.823	1.382
21	rectangular	0.614	0.871	1.557
1	triangular	0.631	0.916	1.691
3	triangular	0.631	0.890	1.580
5	triangular	0.624	0.852	1.442
7	triangular	0.607	0.804	1.296
11	triangular	0.605	0.787	1.240
21	triangular	0.596	0.791	1.275
1	epanechnikov	0.631	0.916	1.691
3	epanechnikov	0.630	0.888	1.573
5	epanechnikov	0.617	0.833	1.385
7	epanechnikov	0.603	0.796	1.280
11	epanechnikov	0.603	0.787	1.245
21	epanechnikov	0.595	0.798	1.303
1	biweight	0.631	0.916	1.691
3	biweight	0.629	0.900	1.631
5	biweight	0.633	0.888	1.569
7	biweight	0.625	0.865	1.500
11	biweight	0.618	0.827	1.357
21	biweight	0.607	0.788	1.242
1	triweight	0.631	0.916	1.691
3	triweight	0.631	0.908	1.663
5	triweight	0.630	0.899	1.623
7	triweight	0.632	0.887	1.571
11	triweight	0.627	0.863	1.481
21	triweight	0.611	0.807	1.295
1	cos	0.631	0.916	1.691
3	cos	0.633	0.891	1.581
5	cos	0.618	0.844	1.425
7	cos	0.605	0.802	1.290
11	cos	0.604	0.788	1.246
21	cos	0.596	0.796	1.292
1	inv	0.631	0.916	1.691
3	inv	0.613	0.843	1.420
5	inv	0.614	0.817	1.325
7	inv	0.616	0.817	1.325
11	inv	0.602	0.805	1.330
21	inv	0.607	0.844	1.461

Tabelle C.23: Fehlerraten bei den Scapula-Daten mit Verwendung des Medians, aber ohne y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.648	0.831	1.296
max. 25	epanechnikov	0.637	0.805	1.214
max. 25	all kernels	0.632	0.800	1.212

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.624	0.894	1.585
3	rectangular	0.626	0.809	1.263
5	rectangular	0.642	0.804	1.208
7	rectangular	0.633	0.800	1.223
11	rectangular	0.665	0.856	1.349
21	rectangular	0.698	0.945	1.628
1	triangular	0.624	0.894	1.585
3	triangular	0.613	0.819	1.330
5	triangular	0.617	0.780	1.173
7	triangular	0.622	0.774	1.146
11	triangular	0.636	0.788	1.160
21	triangular	0.662	0.836	1.273
1	epanechnikov	0.624	0.894	1.585
3	epanechnikov	0.616	0.820	1.325
5	epanechnikov	0.620	0.781	1.173
7	epanechnikov	0.626	0.779	1.153
11	epanechnikov	0.638	0.792	1.169
21	epanechnikov	0.664	0.841	1.286
1	biweight	0.624	0.894	1.585
3	biweight	0.602	0.831	1.402
5	biweight	0.613	0.809	1.294
7	biweight	0.622	0.794	1.210
11	biweight	0.626	0.782	1.158
21	biweight	0.647	0.801	1.182
1	triweight	0.624	0.894	1.585
3	triweight	0.604	0.842	1.444
5	triweight	0.606	0.824	1.368
7	triweight	0.613	0.811	1.298
11	triweight	0.616	0.790	1.212
21	triweight	0.626	0.779	1.153
1	cos	0.624	0.894	1.585
3	cos	0.615	0.822	1.332
5	cos	0.616	0.778	1.167
7	cos	0.625	0.777	1.149
11	cos	0.636	0.789	1.163
21	cos	0.663	0.839	1.280
1	inv	0.624	0.894	1.585
3	inv	0.629	0.804	1.233
5	inv	0.631	0.791	1.182
7	inv	0.633	0.789	1.176
11	inv	0.658	0.835	1.285
21	inv	0.694	0.920	1.531

Tabelle C.24: Fehlerraten bei den Scapula-Daten mit Verwendung von Median und y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.594	0.944	1.869
max. 25	epanechnikov	0.598	0.946	1.864
max. 25	all kernels	0.594	0.939	1.849

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.640	1.045	2.143
3	rectangular	0.639	1.087	2.330
5	rectangular	0.619	0.987	1.966
7	rectangular	0.611	0.982	1.966
11	rectangular	0.600	0.954	1.894
21	rectangular	0.592	0.938	1.850
31	rectangular	0.601	0.959	1.908
41	rectangular	0.602	0.964	1.930
51	rectangular	0.607	0.969	1.940
101	rectangular	0.613	0.983	1.971
1	triangular	0.640	1.045	2.143
3	triangular	0.631	1.026	2.095
5	triangular	0.625	1.005	2.020
7	triangular	0.617	0.986	1.965
11	triangular	0.605	0.956	1.879
21	triangular	0.593	0.937	1.843
31	triangular	0.590	0.931	1.832
41	triangular	0.586	0.926	1.823
51	triangular	0.585	0.925	1.828
101	triangular	0.586	0.925	1.829
1	epanechnikov	0.640	1.045	2.143
3	epanechnikov	0.630	1.025	2.095
5	epanechnikov	0.623	1.002	2.013
7	epanechnikov	0.615	0.979	1.943
11	epanechnikov	0.604	0.954	1.875
21	epanechnikov	0.594	0.937	1.840
31	epanechnikov	0.588	0.928	1.825
41	epanechnikov	0.588	0.931	1.840
51	epanechnikov	0.587	0.932	1.849
101	epanechnikov	0.594	0.944	1.879
1	biweight	0.640	1.045	2.143
3	biweight	0.633	1.030	2.106
5	biweight	0.627	1.012	2.047
7	biweight	0.619	0.992	1.986
11	biweight	0.615	0.980	1.946
21	biweight	0.603	0.954	1.881
31	biweight	0.594	0.940	1.855
41	biweight	0.590	0.933	1.837
51	biweight	0.587	0.927	1.827
101	biweight	0.586	0.926	1.826
1	triweight	0.640	1.045	2.143
3	triweight	0.636	1.037	2.121
5	triweight	0.629	1.018	2.065
7	triweight	0.624	1.005	2.027
11	triweight	0.618	0.987	1.968
21	triweight	0.607	0.964	1.906
31	triweight	0.602	0.953	1.878
41	triweight	0.595	0.945	1.869
51	triweight	0.592	0.939	1.854
101	triweight	0.586	0.927	1.828
1	cos	0.640	1.045	2.143
3	cos	0.631	1.025	2.094
5	cos	0.624	1.005	2.023
7	cos	0.617	0.985	1.959
11	cos	0.605	0.956	1.879
21	cos	0.594	0.938	1.843
31	cos	0.588	0.928	1.825
41	cos	0.587	0.927	1.826
51	cos	0.587	0.931	1.847
101	cos	0.591	0.937	1.858
1	inv	0.640	1.045	2.143
3	inv	0.627	1.014	2.054
5	inv	0.611	0.974	1.938
7	inv	0.606	0.954	1.873
11	inv	0.595	0.935	1.838
21	inv	0.584	0.916	1.793
31	inv	0.586	0.923	1.819
41	inv	0.581	0.914	1.803
51	inv	0.580	0.910	1.791
101	inv	0.581	0.914	1.805

Tabelle C.25: Fehlerraten bei den Mietspiegel-Daten mit 5 Klassen ohne Verwendung von Median oder y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.628	0.861	1.419
max. 25	epanechnikov	0.618	0.850	1.414
max. 25	all kernels	0.620	0.853	1.417

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.640	1.045	2.143
3	rectangular	0.624	0.936	1.728
5	rectangular	0.616	0.894	1.580
7	rectangular	0.614	0.873	1.501
11	rectangular	0.620	0.859	1.437
21	rectangular	0.631	0.883	1.501
31	rectangular	0.653	0.870	1.384
41	rectangular	0.663	0.871	1.358
51	rectangular	0.677	0.879	1.345
101	rectangular	0.742	0.945	1.399
1	triangular	0.640	1.045	2.143
3	triangular	0.634	1.007	2.003
5	triangular	0.626	0.948	1.774
7	triangular	0.618	0.913	1.651
11	triangular	0.613	0.875	1.520
21	triangular	0.616	0.848	1.411
31	triangular	0.622	0.842	1.374
41	triangular	0.628	0.841	1.353
51	triangular	0.634	0.842	1.338
101	triangular	0.674	0.867	1.314
1	epanechnikov	0.640	1.045	2.143
3	epanechnikov	0.634	0.998	1.967
5	epanechnikov	0.624	0.936	1.726
7	epanechnikov	0.616	0.903	1.615
11	epanechnikov	0.612	0.869	1.496
21	epanechnikov	0.619	0.849	1.404
31	epanechnikov	0.627	0.845	1.367
41	epanechnikov	0.635	0.847	1.354
51	epanechnikov	0.642	0.849	1.341
101	epanechnikov	0.687	0.881	1.327
1	biweight	0.640	1.045	2.143
3	biweight	0.635	1.023	2.065
5	biweight	0.628	0.978	1.899
7	biweight	0.623	0.943	1.761
11	biweight	0.614	0.898	1.602
21	biweight	0.615	0.860	1.460
31	biweight	0.617	0.847	1.404
41	biweight	0.622	0.843	1.377
51	biweight	0.626	0.843	1.364
101	biweight	0.650	0.850	1.318
1	triweight	0.640	1.045	2.143
3	triweight	0.638	1.035	2.104
5	triweight	0.632	0.997	1.968
7	triweight	0.627	0.972	1.874
11	triweight	0.618	0.924	1.697
21	triweight	0.614	0.877	1.525
31	triweight	0.616	0.858	1.449
41	triweight	0.618	0.848	1.408
51	triweight	0.621	0.847	1.394
101	triweight	0.633	0.842	1.341
1	cos	0.640	1.045	2.143
3	cos	0.634	1.002	1.980
5	cos	0.625	0.942	1.747
7	cos	0.618	0.909	1.633
11	cos	0.613	0.872	1.507
21	cos	0.618	0.848	1.406
31	cos	0.625	0.843	1.368
41	cos	0.633	0.846	1.355
51	cos	0.639	0.847	1.339
101	cos	0.681	0.874	1.318
1	inv	0.640	1.045	2.143
3	inv	0.622	0.950	1.792
5	inv	0.614	0.911	1.658
7	inv	0.612	0.890	1.579
11	inv	0.612	0.865	1.487
21	inv	0.617	0.846	1.402
31	inv	0.624	0.840	1.355
41	inv	0.630	0.839	1.334
51	inv	0.638	0.842	1.322
101	inv	0.683	0.876	1.315

Tabelle C.26: Fehlerraten bei den Mietspiegel-Daten mit 5 Klassen mit Verwendung des Medians, aber ohne y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.748	0.952	1.393
max. 25	epanechnikov	0.749	0.952	1.394
max. 25	all kernels	0.747	0.949	1.388

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.717	1.075	2.008
3	rectangular	0.728	0.969	1.519
5	rectangular	0.747	0.957	1.416
7	rectangular	0.743	0.948	1.391
11	rectangular	0.747	0.947	1.379
21	rectangular	0.754	0.949	1.363
31	rectangular	0.759	0.961	1.389
41	rectangular	0.763	0.971	1.410
51	rectangular	0.765	0.980	1.429
101	rectangular	0.781	1.027	1.534
1	triangular	0.717	1.075	2.008
3	triangular	0.734	0.984	1.568
5	triangular	0.745	0.968	1.474
7	triangular	0.746	0.957	1.427
11	triangular	0.744	0.943	1.376
21	triangular	0.750	0.945	1.365
31	triangular	0.752	0.947	1.364
41	triangular	0.754	0.948	1.363
51	triangular	0.755	0.949	1.362
101	triangular	0.770	0.976	1.409
1	epanechnikov	0.717	1.075	2.008
3	epanechnikov	0.735	0.983	1.558
5	epanechnikov	0.746	0.966	1.461
7	epanechnikov	0.747	0.957	1.421
11	epanechnikov	0.745	0.944	1.374
21	epanechnikov	0.751	0.947	1.366
31	epanechnikov	0.753	0.949	1.368
41	epanechnikov	0.755	0.949	1.362
51	epanechnikov	0.757	0.952	1.365
101	epanechnikov	0.772	0.984	1.429
1	biweight	0.717	1.075	2.008
3	biweight	0.734	0.997	1.620
5	biweight	0.747	0.984	1.528
7	biweight	0.747	0.967	1.465
11	biweight	0.744	0.948	1.398
21	biweight	0.747	0.943	1.369
31	biweight	0.749	0.945	1.367
41	biweight	0.751	0.947	1.367
51	biweight	0.753	0.948	1.364
101	biweight	0.767	0.964	1.380
1	triweight	0.717	1.075	2.008
3	triweight	0.733	1.004	1.652
5	triweight	0.747	0.997	1.579
7	triweight	0.748	0.983	1.521
11	triweight	0.748	0.962	1.443
21	triweight	0.745	0.944	1.380
31	triweight	0.746	0.943	1.372
41	triweight	0.749	0.945	1.369
51	triweight	0.750	0.946	1.368
101	triweight	0.760	0.953	1.363
1	cos	0.717	1.075	2.008
3	cos	0.735	0.984	1.564
5	cos	0.745	0.968	1.468
7	cos	0.747	0.958	1.426
11	cos	0.745	0.944	1.376
21	cos	0.750	0.946	1.366
31	cos	0.753	0.949	1.369
41	cos	0.755	0.948	1.360
51	cos	0.757	0.951	1.361
101	cos	0.771	0.981	1.421
1	inv	0.717	1.075	2.008
3	inv	0.729	0.971	1.532
5	inv	0.739	0.954	1.435
7	inv	0.741	0.948	1.403
11	inv	0.743	0.942	1.375
21	inv	0.748	0.941	1.356
31	inv	0.749	0.941	1.354
41	inv	0.749	0.942	1.355
51	inv	0.752	0.947	1.361
101	inv	0.768	0.981	1.421

Tabelle C.27: Fehlerraten bei den Mietspiegel-Daten mit 5 Klassen mit Verwendung von Median und y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.612	0.987	1.979
max. 25	epanechnikov	0.611	0.986	1.982
max. 25	all kernels	0.611	0.979	1.959

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.649	1.079	2.264
3	rectangular	0.646	1.106	2.398
5	rectangular	0.626	1.014	2.061
7	rectangular	0.616	1.005	2.047
11	rectangular	0.606	0.974	1.943
21	rectangular	0.613	0.979	1.944
31	rectangular	0.617	0.991	1.990
41	rectangular	0.616	1.010	2.076
51	rectangular	0.616	1.010	2.083
101	rectangular	0.614	0.987	1.990
1	triangular	0.649	1.079	2.264
3	triangular	0.643	1.060	2.208
5	triangular	0.632	1.030	2.113
7	triangular	0.628	1.013	2.052
11	triangular	0.615	0.988	1.984
21	triangular	0.609	0.978	1.955
31	triangular	0.608	0.977	1.956
41	triangular	0.609	0.981	1.973
51	triangular	0.608	0.983	1.985
101	triangular	0.606	0.972	1.959
1	epanechnikov	0.649	1.079	2.264
3	epanechnikov	0.643	1.063	2.218
5	epanechnikov	0.629	1.025	2.102
7	epanechnikov	0.623	1.004	2.027
11	epanechnikov	0.611	0.980	1.964
21	epanechnikov	0.610	0.981	1.962
31	epanechnikov	0.609	0.980	1.965
41	epanechnikov	0.612	0.986	1.985
51	epanechnikov	0.611	0.988	2.000
101	epanechnikov	0.607	0.976	1.977
1	biweight	0.649	1.079	2.264
3	biweight	0.645	1.066	2.226
5	biweight	0.638	1.050	2.180
7	biweight	0.634	1.035	2.125
11	biweight	0.625	1.010	2.049
21	biweight	0.614	0.987	1.982
31	biweight	0.611	0.983	1.969
41	biweight	0.610	0.983	1.974
51	biweight	0.610	0.985	1.978
101	biweight	0.609	0.979	1.968
1	triweight	0.649	1.079	2.264
3	triweight	0.646	1.071	2.241
5	triweight	0.643	1.059	2.202
7	triweight	0.638	1.044	2.157
11	triweight	0.632	1.026	2.098
21	triweight	0.622	1.001	2.020
31	triweight	0.617	0.993	2.000
41	triweight	0.613	0.988	1.987
51	triweight	0.611	0.986	1.981
101	triweight	0.610	0.981	1.965
1	cos	0.649	1.079	2.264
3	cos	0.643	1.063	2.220
5	cos	0.630	1.030	2.119
7	cos	0.625	1.008	2.039
11	cos	0.613	0.985	1.980
21	cos	0.610	0.982	1.967
31	cos	0.610	0.980	1.963
41	cos	0.612	0.985	1.978
51	cos	0.611	0.988	1.998
101	cos	0.606	0.973	1.968
1	inv	0.649	1.079	2.264
3	inv	0.639	1.048	2.170
5	inv	0.623	1.000	2.016
7	inv	0.613	0.981	1.965
11	inv	0.602	0.961	1.920
21	inv	0.602	0.961	1.922
31	inv	0.599	0.964	1.944
41	inv	0.600	0.965	1.944
51	inv	0.599	0.962	1.942
101	inv	0.591	0.936	1.869

Tabelle C.28: Fehlerraten bei den Mietspiegel-Daten mit 5 Klassen ohne Verwendung von Median oder y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.646	0.887	1.468
max. 25	epanechnikov	0.638	0.892	1.514
max. 25	all kernels	0.638	0.881	1.469

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.649	1.079	2.264
3	rectangular	0.635	0.967	1.827
5	rectangular	0.631	0.923	1.651
7	rectangular	0.628	0.904	1.588
11	rectangular	0.641	0.862	1.390
21	rectangular	0.652	0.886	1.454
31	rectangular	0.655	0.883	1.433
41	rectangular	0.669	0.893	1.421
51	rectangular	0.685	0.900	1.403
101	rectangular	0.742	0.945	1.395
1	triangular	0.649	1.079	2.264
3	triangular	0.644	1.033	2.089
5	triangular	0.633	0.974	1.858
7	triangular	0.631	0.948	1.755
11	triangular	0.629	0.911	1.616
21	triangular	0.636	0.892	1.518
31	triangular	0.640	0.882	1.472
41	triangular	0.642	0.877	1.449
51	triangular	0.646	0.872	1.423
101	triangular	0.666	0.871	1.355
1	epanechnikov	0.649	1.079	2.264
3	epanechnikov	0.644	1.028	2.065
5	epanechnikov	0.635	0.967	1.824
7	epanechnikov	0.632	0.939	1.716
11	epanechnikov	0.631	0.905	1.581
21	epanechnikov	0.639	0.892	1.512
31	epanechnikov	0.642	0.881	1.466
41	epanechnikov	0.645	0.879	1.448
51	epanechnikov	0.650	0.875	1.422
101	epanechnikov	0.679	0.882	1.357
1	biweight	0.649	1.079	2.264
3	biweight	0.646	1.052	2.163
5	biweight	0.638	1.007	1.988
7	biweight	0.634	0.971	1.848
11	biweight	0.631	0.936	1.715
21	biweight	0.635	0.904	1.568
31	biweight	0.638	0.894	1.519
41	biweight	0.641	0.887	1.489
51	biweight	0.642	0.882	1.470
101	biweight	0.653	0.872	1.405
1	triweight	0.649	1.079	2.264
3	triweight	0.648	1.064	2.209
5	triweight	0.644	1.030	2.077
7	triweight	0.639	1.001	1.964
11	triweight	0.635	0.963	1.813
21	triweight	0.633	0.924	1.655
31	triweight	0.636	0.910	1.585
41	triweight	0.637	0.899	1.542
51	triweight	0.639	0.894	1.518
101	triweight	0.645	0.879	1.450
1	cos	0.649	1.079	2.264
3	cos	0.646	1.035	2.088
5	cos	0.635	0.973	1.848
7	cos	0.632	0.943	1.732
11	cos	0.631	0.908	1.594
21	cos	0.637	0.891	1.512
31	cos	0.640	0.880	1.468
41	cos	0.644	0.878	1.450
51	cos	0.649	0.876	1.428
101	cos	0.674	0.877	1.356
1	inv	0.649	1.079	2.264
3	inv	0.632	0.979	1.887
5	inv	0.626	0.937	1.724
7	inv	0.625	0.920	1.661
11	inv	0.623	0.888	1.548
21	inv	0.629	0.872	1.461
31	inv	0.631	0.860	1.414
41	inv	0.637	0.859	1.388
51	inv	0.645	0.858	1.361
101	inv	0.691	0.887	1.329

Tabelle C.29: Fehlerraten bei den Mietspiegel-Daten mit 5 Klassen mit Verwendung des Medians, aber ohne y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.750	0.965	1.430
max. 25	epanechnikov	0.750	0.961	1.424
max. 25	all kernels	0.754	0.964	1.421

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.721	1.100	2.099
3	rectangular	0.734	0.987	1.571
5	rectangular	0.745	0.965	1.447
7	rectangular	0.745	0.957	1.423
11	rectangular	0.752	0.960	1.410
21	rectangular	0.757	0.960	1.391
31	rectangular	0.758	0.965	1.401
41	rectangular	0.763	0.981	1.441
51	rectangular	0.769	0.990	1.454
101	rectangular	0.784	1.046	1.585
1	triangular	0.721	1.100	2.100
3	triangular	0.736	1.004	1.630
5	triangular	0.743	0.981	1.522
7	triangular	0.745	0.967	1.463
11	triangular	0.746	0.956	1.417
21	triangular	0.753	0.960	1.407
31	triangular	0.755	0.955	1.385
41	triangular	0.755	0.952	1.375
51	triangular	0.758	0.956	1.379
101	triangular	0.770	0.981	1.423
1	epanechnikov	0.721	1.100	2.099
3	epanechnikov	0.737	1.000	1.612
5	epanechnikov	0.745	0.978	1.504
7	epanechnikov	0.746	0.967	1.454
11	epanechnikov	0.746	0.956	1.414
21	epanechnikov	0.754	0.958	1.400
31	epanechnikov	0.756	0.956	1.385
41	epanechnikov	0.756	0.953	1.375
51	epanechnikov	0.761	0.961	1.389
101	epanechnikov	0.771	0.988	1.442
1	biweight	0.721	1.100	2.099
3	biweight	0.738	1.015	1.675
5	biweight	0.747	0.998	1.579
7	biweight	0.748	0.984	1.520
11	biweight	0.748	0.967	1.452
21	biweight	0.749	0.959	1.417
31	biweight	0.751	0.957	1.404
41	biweight	0.754	0.956	1.392
51	biweight	0.759	0.958	1.386
101	biweight	0.766	0.968	1.399
1	triweight	0.721	1.100	2.099
3	triweight	0.735	1.020	1.703
5	triweight	0.746	1.010	1.627
7	triweight	0.750	0.998	1.571
11	triweight	0.752	0.981	1.500
21	triweight	0.749	0.966	1.442
31	triweight	0.749	0.961	1.421
41	triweight	0.749	0.958	1.412
51	triweight	0.754	0.961	1.408
101	triweight	0.761	0.962	1.393
1	cos	0.721	1.100	2.099
3	cos	0.737	1.002	1.624
5	cos	0.745	0.980	1.513
7	cos	0.746	0.968	1.458
11	cos	0.746	0.956	1.414
21	cos	0.753	0.959	1.404
31	cos	0.755	0.955	1.383
41	cos	0.756	0.954	1.378
51	cos	0.760	0.960	1.386
101	cos	0.771	0.985	1.435
1	inv	0.721	1.100	2.099
3	inv	0.730	0.987	1.579
5	inv	0.740	0.966	1.474
7	inv	0.742	0.958	1.436
11	inv	0.749	0.954	1.405
21	inv	0.749	0.948	1.375
31	inv	0.748	0.944	1.362
41	inv	0.748	0.945	1.365
51	inv	0.752	0.950	1.372
101	inv	0.768	0.990	1.452

Tabelle C.30: Fehlerraten bei den Mietspiegel-Daten mit 5 Klassen mit Verwendung von Median und y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.769	2.212	9.189
max. 25	epanechnikov	0.759	2.139	8.722
max. 25	all kernels	0.768	2.197	9.063

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.807	2.326	9.742
3	rectangular	0.797	2.525	11.382
5	rectangular	0.775	2.365	10.387
7	rectangular	0.768	2.207	9.120
11	rectangular	0.765	2.213	9.221
21	rectangular	0.763	2.174	9.006
31	rectangular	0.768	2.210	9.228
41	rectangular	0.768	2.246	9.514
51	rectangular	0.768	2.242	9.495
101	rectangular	0.774	2.287	9.809
1	triangular	0.807	2.326	9.742
3	triangular	0.800	2.310	9.679
5	triangular	0.789	2.263	9.386
7	triangular	0.780	2.230	9.207
11	triangular	0.767	2.166	8.815
21	triangular	0.757	2.127	8.622
31	triangular	0.754	2.117	8.622
41	triangular	0.754	2.130	8.753
51	triangular	0.755	2.138	8.813
101	triangular	0.756	2.163	9.011
1	epanechnikov	0.807	2.326	9.742
3	epanechnikov	0.799	2.308	9.661
5	epanechnikov	0.783	2.250	9.335
7	epanechnikov	0.774	2.209	9.092
11	epanechnikov	0.764	2.159	8.775
21	epanechnikov	0.758	2.140	8.738
31	epanechnikov	0.757	2.131	8.718
41	epanechnikov	0.756	2.143	8.812
51	epanechnikov	0.757	2.150	8.876
101	epanechnikov	0.760	2.199	9.247
1	biweight	0.807	2.326	9.742
3	biweight	0.803	2.316	9.698
5	biweight	0.793	2.282	9.496
7	biweight	0.784	2.248	9.307
11	biweight	0.775	2.209	9.071
21	biweight	0.761	2.152	8.740
31	biweight	0.758	2.131	8.646
41	biweight	0.754	2.119	8.660
51	biweight	0.754	2.128	8.735
101	biweight	0.756	2.155	8.917
1	triweight	0.807	2.326	9.742
3	triweight	0.805	2.320	9.699
5	triweight	0.797	2.295	9.566
7	triweight	0.790	2.274	9.454
11	triweight	0.783	2.234	9.194
21	triweight	0.765	2.186	8.976
31	triweight	0.761	2.157	8.789
41	triweight	0.758	2.133	8.671
51	triweight	0.756	2.131	8.715
101	triweight	0.752	2.128	8.758
1	cos	0.807	2.326	9.742
3	cos	0.799	2.309	9.667
5	cos	0.786	2.259	9.372
7	cos	0.775	2.219	9.165
11	cos	0.765	2.165	8.815
21	cos	0.757	2.138	8.708
31	cos	0.756	2.128	8.705
41	cos	0.755	2.141	8.809
51	cos	0.755	2.146	8.856
101	cos	0.758	2.183	9.138
1	inv	0.807	2.326	9.742
3	inv	0.796	2.291	9.559
5	inv	0.779	2.210	9.070
7	inv	0.777	2.171	8.799
11	inv	0.770	2.146	8.700
21	inv	0.762	2.116	8.612
31	inv	0.760	2.116	8.658
41	inv	0.759	2.138	8.856
51	inv	0.758	2.131	8.816
101	inv	0.760	2.154	9.000

Tabelle C.31: Fehlerraten bei den Mietspiegel-Daten mit 10 Klassen ohne Verwendung von Median oder y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.832	1.912	6.114
max. 25	epanechnikov	0.821	1.876	6.020
max. 25	all kernels	0.822	1.881	6.024

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.807	2.326	9.742
3	rectangular	0.795	2.053	7.504
5	rectangular	0.795	1.964	6.810
7	rectangular	0.806	1.925	6.462
11	rectangular	0.816	1.897	6.192
21	rectangular	0.838	1.892	5.925
31	rectangular	0.848	1.914	5.939
41	rectangular	0.855	1.920	5.900
51	rectangular	0.858	1.929	5.900
101	rectangular	0.879	2.020	6.120
1	triangular	0.807	2.326	9.742
3	triangular	0.800	2.236	9.036
5	triangular	0.799	2.095	7.862
7	triangular	0.796	2.010	7.188
11	triangular	0.802	1.933	6.558
21	triangular	0.815	1.874	6.046
31	triangular	0.827	1.862	5.864
41	triangular	0.835	1.859	5.760
51	triangular	0.837	1.860	5.705
101	triangular	0.852	1.893	5.709
1	epanechnikov	0.807	2.326	9.742
3	epanechnikov	0.800	2.213	8.828
5	epanechnikov	0.799	2.062	7.573
7	epanechnikov	0.796	1.985	6.988
11	epanechnikov	0.802	1.917	6.429
21	epanechnikov	0.818	1.870	5.996
31	epanechnikov	0.831	1.865	5.836
41	epanechnikov	0.837	1.866	5.758
51	epanechnikov	0.843	1.873	5.732
101	epanechnikov	0.857	1.914	5.774
1	biweight	0.807	2.326	9.742
3	biweight	0.803	2.276	9.351
5	biweight	0.799	2.169	8.489
7	biweight	0.798	2.086	7.807
11	biweight	0.799	1.983	6.965
21	biweight	0.810	1.901	6.283
31	biweight	0.818	1.876	6.034
41	biweight	0.824	1.863	5.889
51	biweight	0.831	1.859	5.800
101	biweight	0.844	1.873	5.692
1	triweight	0.807	2.326	9.742
3	triweight	0.805	2.302	9.550
5	triweight	0.803	2.219	8.879
7	triweight	0.800	2.154	8.377
11	triweight	0.800	2.047	7.491
21	triweight	0.801	1.934	6.583
31	triweight	0.810	1.899	6.272
41	triweight	0.816	1.883	6.102
51	triweight	0.820	1.873	5.998
101	triweight	0.836	1.859	5.718
1	cos	0.807	2.326	9.742
3	cos	0.800	2.220	8.897
5	cos	0.799	2.077	7.702
7	cos	0.797	1.996	7.069
11	cos	0.801	1.922	6.478
21	cos	0.816	1.871	6.024
31	cos	0.830	1.865	5.853
41	cos	0.836	1.864	5.760
51	cos	0.840	1.867	5.720
101	cos	0.854	1.904	5.748
1	inv	0.807	2.326	9.742
3	inv	0.798	2.106	7.954
5	inv	0.793	2.008	7.231
7	inv	0.800	1.967	6.849
11	inv	0.804	1.914	6.436
21	inv	0.822	1.869	5.975
31	inv	0.829	1.854	5.798
41	inv	0.832	1.848	5.702
51	inv	0.835	1.849	5.663
101	inv	0.857	1.896	5.677

Tabelle C.32: Fehlerraten bei den Mietspiegel-Daten mit 10 Klassen mit Verwendung des Medians, aber ohne y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.863	1.943	5.859
max. 25	epanechnikov	0.861	1.921	5.759
max. 25	all kernels	0.861	1.922	5.762

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.836	2.318	9.470
3	rectangular	0.849	2.028	6.754
5	rectangular	0.849	1.961	6.187
7	rectangular	0.854	1.938	5.934
11	rectangular	0.864	1.934	5.778
21	rectangular	0.873	1.958	5.750
31	rectangular	0.876	1.987	5.845
41	rectangular	0.877	2.002	5.903
51	rectangular	0.880	2.016	5.951
101	rectangular	0.891	2.101	6.282
1	triangular	0.836	2.318	9.470
3	triangular	0.848	2.104	7.460
5	triangular	0.845	2.007	6.657
7	triangular	0.846	1.963	6.285
11	triangular	0.852	1.926	5.901
21	triangular	0.863	1.908	5.643
31	triangular	0.869	1.921	5.617
41	triangular	0.871	1.936	5.632
51	triangular	0.873	1.948	5.656
101	triangular	0.878	1.992	5.791
1	epanechnikov	0.836	2.318	9.470
3	epanechnikov	0.846	2.092	7.350
5	epanechnikov	0.846	1.997	6.544
7	epanechnikov	0.848	1.957	6.191
11	epanechnikov	0.853	1.919	5.834
21	epanechnikov	0.867	1.916	5.649
31	epanechnikov	0.870	1.933	5.647
41	epanechnikov	0.873	1.952	5.689
51	epanechnikov	0.874	1.956	5.683
101	epanechnikov	0.880	2.012	5.883
1	biweight	0.836	2.318	9.470
3	biweight	0.848	2.148	7.856
5	biweight	0.848	2.057	7.073
7	biweight	0.846	2.003	6.611
11	biweight	0.847	1.946	6.120
21	biweight	0.860	1.917	5.749
31	biweight	0.864	1.916	5.663
41	biweight	0.868	1.925	5.643
51	biweight	0.869	1.935	5.663
101	biweight	0.875	1.966	5.706
1	triweight	0.836	2.318	9.470
3	triweight	0.849	2.177	8.127
5	triweight	0.851	2.103	7.469
7	triweight	0.849	2.052	7.029
11	triweight	0.847	1.983	6.449
21	triweight	0.856	1.928	5.899
31	triweight	0.861	1.917	5.750
41	triweight	0.864	1.919	5.693
51	triweight	0.866	1.920	5.653
101	triweight	0.871	1.946	5.657
1	cos	0.836	2.318	9.470
3	cos	0.846	2.097	7.395
5	cos	0.846	2.004	6.607
7	cos	0.846	1.960	6.235
11	cos	0.853	1.921	5.846
21	cos	0.865	1.914	5.649
31	cos	0.870	1.929	5.633
41	cos	0.872	1.947	5.674
51	cos	0.874	1.953	5.676
101	cos	0.879	2.001	5.838
1	inv	0.836	2.318	9.470
3	inv	0.849	2.049	6.954
5	inv	0.849	1.973	6.354
7	inv	0.849	1.932	6.016
11	inv	0.859	1.914	5.782
21	inv	0.866	1.904	5.586
31	inv	0.872	1.921	5.609
41	inv	0.874	1.933	5.614
51	inv	0.876	1.942	5.623
101	inv	0.884	1.998	5.806

Tabelle C.33: Fehlerraten bei den Mietspiegel-Daten mit 10 Klassen mit Verwendung von Median und y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.777	2.252	9.391
max. 25	epanechnikov	0.776	2.228	9.245
max. 25	all kernels	0.777	2.237	9.325

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.816	2.401	10.274
3	rectangular	0.801	2.570	11.746
5	rectangular	0.779	2.424	10.823
7	rectangular	0.776	2.280	9.577
11	rectangular	0.771	2.247	9.425
21	rectangular	0.772	2.220	9.220
31	rectangular	0.772	2.253	9.548
41	rectangular	0.777	2.290	9.762
51	rectangular	0.777	2.310	9.960
101	rectangular	0.773	2.286	9.798
1	triangular	0.816	2.401	10.274
3	triangular	0.809	2.373	10.120
5	triangular	0.795	2.321	9.843
7	triangular	0.787	2.276	9.529
11	triangular	0.776	2.226	9.205
21	triangular	0.768	2.191	9.062
31	triangular	0.766	2.198	9.146
41	triangular	0.765	2.212	9.297
51	triangular	0.763	2.217	9.374
101	triangular	0.762	2.219	9.422
1	epanechnikov	0.816	2.401	10.274
3	epanechnikov	0.809	2.374	10.128
5	epanechnikov	0.792	2.308	9.762
7	epanechnikov	0.780	2.246	9.337
11	epanechnikov	0.772	2.205	9.093
21	epanechnikov	0.768	2.204	9.146
31	epanechnikov	0.766	2.214	9.258
41	epanechnikov	0.764	2.219	9.341
51	epanechnikov	0.763	2.226	9.426
101	epanechnikov	0.764	2.241	9.565
1	biweight	0.816	2.401	10.274
3	biweight	0.813	2.383	10.170
5	biweight	0.803	2.353	10.015
7	biweight	0.796	2.317	9.791
11	biweight	0.783	2.261	9.443
21	biweight	0.773	2.218	9.197
31	biweight	0.769	2.204	9.163
41	biweight	0.769	2.207	9.185
51	biweight	0.767	2.209	9.231
101	biweight	0.763	2.216	9.370
1	triweight	0.816	2.401	10.274
3	triweight	0.815	2.393	10.224
5	triweight	0.808	2.368	10.088
7	triweight	0.802	2.344	9.966
11	triweight	0.792	2.299	9.678
21	triweight	0.780	2.252	9.402
31	triweight	0.773	2.222	9.235
41	triweight	0.770	2.209	9.189
51	triweight	0.768	2.201	9.159
101	triweight	0.764	2.205	9.264
1	cos	0.816	2.401	10.274
3	cos	0.809	2.374	10.128
5	cos	0.793	2.318	9.829
7	cos	0.783	2.248	9.350
11	cos	0.772	2.208	9.121
21	cos	0.769	2.203	9.134
31	cos	0.767	2.215	9.257
41	cos	0.764	2.220	9.357
51	cos	0.763	2.226	9.416
101	cos	0.764	2.234	9.510
1	inv	0.816	2.401	10.274
3	inv	0.803	2.349	10.003
5	inv	0.785	2.257	9.413
7	inv	0.780	2.205	9.036
11	inv	0.772	2.185	9.040
21	inv	0.769	2.184	9.085
31	inv	0.769	2.202	9.271
41	inv	0.769	2.218	9.374
51	inv	0.768	2.213	9.390
101	inv	0.763	2.199	9.304

Tabelle C.34: Fehlerraten bei den Mietspiegel-Daten mit 10 Klassen ohne Verwendung von Median oder y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.844	1.939	6.178
max. 25	epanechnikov	0.831	1.950	6.443
max. 25	all kernels	0.834	1.919	6.170

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.816	2.401	10.274
3	rectangular	0.800	2.125	8.047
5	rectangular	0.807	2.037	7.216
7	rectangular	0.814	1.995	6.864
11	rectangular	0.827	1.956	6.514
21	rectangular	0.847	1.934	6.138
31	rectangular	0.846	1.930	6.072
41	rectangular	0.852	1.942	6.078
51	rectangular	0.856	1.952	6.047
101	rectangular	0.881	2.031	6.152
1	triangular	0.816	2.401	10.274
3	triangular	0.807	2.289	9.426
5	triangular	0.802	2.151	8.268
7	triangular	0.802	2.076	7.656
11	triangular	0.810	2.006	7.050
21	triangular	0.827	1.954	6.513
31	triangular	0.837	1.930	6.256
41	triangular	0.841	1.916	6.108
51	triangular	0.842	1.907	6.028
101	triangular	0.854	1.910	5.855
1	epanechnikov	0.816	2.401	10.274
3	epanechnikov	0.808	2.273	9.265
5	epanechnikov	0.802	2.128	8.071
7	epanechnikov	0.803	2.057	7.485
11	epanechnikov	0.813	1.992	6.884
21	epanechnikov	0.834	1.954	6.453
31	epanechnikov	0.841	1.931	6.215
41	epanechnikov	0.844	1.921	6.108
51	epanechnikov	0.847	1.917	6.045
101	epanechnikov	0.858	1.928	5.904
1	biweight	0.816	2.401	10.274
3	biweight	0.811	2.338	9.803
5	biweight	0.804	2.227	8.901
7	biweight	0.803	2.138	8.176
11	biweight	0.805	2.059	7.519
21	biweight	0.818	1.985	6.813
31	biweight	0.828	1.955	6.494
41	biweight	0.832	1.940	6.346
51	biweight	0.838	1.931	6.243
101	biweight	0.845	1.910	5.991
1	triweight	0.816	2.401	10.274
3	triweight	0.814	2.369	10.039
5	triweight	0.809	2.284	9.369
7	triweight	0.807	2.214	8.786
11	triweight	0.804	2.118	8.007
21	triweight	0.810	2.026	7.222
31	triweight	0.818	1.989	6.857
41	triweight	0.824	1.971	6.656
51	triweight	0.829	1.958	6.514
101	triweight	0.838	1.923	6.168
1	cos	0.816	2.401	10.274
3	cos	0.808	2.286	9.374
5	cos	0.802	2.141	8.183
7	cos	0.802	2.066	7.553
11	cos	0.811	1.997	6.953
21	cos	0.831	1.955	6.482
31	cos	0.841	1.933	6.237
41	cos	0.843	1.921	6.118
51	cos	0.845	1.917	6.054
101	cos	0.856	1.921	5.893
1	inv	0.816	2.401	10.274
3	inv	0.802	2.163	8.417
5	inv	0.801	2.077	7.642
7	inv	0.809	2.029	7.219
11	inv	0.816	1.966	6.729
21	inv	0.829	1.907	6.195
31	inv	0.833	1.884	5.973
41	inv	0.835	1.877	5.879
51	inv	0.840	1.878	5.822
101	inv	0.860	1.914	5.758

Tabelle C.35: Fehlerraten bei den Mietspiegel-Daten mit 10 Klassen mit Verwendung des Medians, aber ohne y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.869	1.989	6.088
max. 25	epanechnikov	0.867	1.966	6.003
max. 25	all kernels	0.865	1.948	5.898

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.842	2.388	9.971
3	rectangular	0.851	2.085	7.178
5	rectangular	0.853	2.013	6.475
7	rectangular	0.859	1.984	6.209
11	rectangular	0.867	1.973	6.008
21	rectangular	0.882	1.985	5.880
31	rectangular	0.883	1.997	5.901
41	rectangular	0.878	2.020	6.007
51	rectangular	0.879	2.036	6.055
101	rectangular	0.894	2.121	6.342
1	triangular	0.842	2.388	9.971
3	triangular	0.852	2.157	7.848
5	triangular	0.850	2.059	7.021
7	triangular	0.851	2.014	6.617
11	triangular	0.853	1.970	6.219
21	triangular	0.868	1.958	5.946
31	triangular	0.873	1.957	5.848
41	triangular	0.874	1.961	5.811
51	triangular	0.877	1.967	5.795
101	triangular	0.879	1.999	5.862
1	epanechnikov	0.842	2.388	9.971
3	epanechnikov	0.852	2.148	7.751
5	epanechnikov	0.852	2.052	6.918
7	epanechnikov	0.851	2.004	6.505
11	epanechnikov	0.854	1.966	6.147
21	epanechnikov	0.871	1.963	5.927
31	epanechnikov	0.877	1.969	5.870
41	epanechnikov	0.878	1.974	5.848
51	epanechnikov	0.880	1.981	5.840
101	epanechnikov	0.880	2.015	5.916
1	biweight	0.842	2.388	9.971
3	biweight	0.851	2.203	8.266
5	biweight	0.851	2.112	7.479
7	biweight	0.851	2.057	7.012
11	biweight	0.851	2.002	6.524
21	biweight	0.862	1.971	6.129
31	biweight	0.871	1.966	5.973
41	biweight	0.875	1.965	5.901
51	biweight	0.876	1.968	5.877
101	biweight	0.876	1.983	5.846
1	triweight	0.842	2.388	9.971
3	triweight	0.851	2.233	8.552
5	triweight	0.851	2.156	7.878
7	triweight	0.851	2.101	7.408
11	triweight	0.850	2.041	6.884
21	triweight	0.856	1.989	6.366
31	triweight	0.863	1.976	6.160
41	triweight	0.870	1.975	6.059
51	triweight	0.873	1.968	5.968
101	triweight	0.876	1.968	5.859
1	cos	0.842	2.388	9.971
3	cos	0.852	2.152	7.794
5	cos	0.852	2.057	6.978
7	cos	0.851	2.008	6.550
11	cos	0.853	1.967	6.172
21	cos	0.870	1.961	5.936
31	cos	0.875	1.964	5.854
41	cos	0.877	1.971	5.835
51	cos	0.879	1.977	5.835
101	cos	0.881	2.010	5.896
1	inv	0.842	2.388	9.971
3	inv	0.853	2.099	7.355
5	inv	0.855	2.021	6.645
7	inv	0.859	1.981	6.295
11	inv	0.861	1.954	6.036
21	inv	0.869	1.933	5.778
31	inv	0.869	1.932	5.705
41	inv	0.873	1.949	5.724
51	inv	0.876	1.961	5.732
101	inv	0.886	2.014	5.870

Tabelle C.36: Fehlerraten bei den Mietspiegel-Daten mit 10 Klassen mit Verwendung von Median und y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.655	1.124	2.376
max. 25	epanechnikov	0.684	1.194	2.556
max. 25	all kernels	0.652	1.129	2.405

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.827	1.497	3.349
3	rectangular	0.735	1.333	3.069
5	rectangular	0.747	1.376	3.144
7	rectangular	0.705	1.256	2.760
11	rectangular	0.690	1.185	2.505
21	rectangular	0.626	1.090	2.320
31	rectangular	0.602	1.050	2.240
1	triangular	0.827	1.497	3.349
3	triangular	0.825	1.485	3.323
5	triangular	0.805	1.485	3.373
7	triangular	0.780	1.441	3.287
11	triangular	0.751	1.360	3.066
21	triangular	0.695	1.196	2.538
31	triangular	0.632	1.099	2.335
1	epanechnikov	0.827	1.497	3.349
3	epanechnikov	0.823	1.488	3.338
5	epanechnikov	0.796	1.477	3.379
7	epanechnikov	0.762	1.401	3.183
11	epanechnikov	0.744	1.324	2.922
21	epanechnikov	0.686	1.181	2.483
31	epanechnikov	0.618	1.081	2.309
1	biweight	0.827	1.497	3.349
3	biweight	0.823	1.493	3.359
5	biweight	0.815	1.488	3.358
7	biweight	0.792	1.484	3.420
11	biweight	0.758	1.392	3.176
21	biweight	0.718	1.246	2.672
31	biweight	0.661	1.144	2.418
1	triweight	0.827	1.497	3.349
3	triweight	0.826	1.495	3.347
5	triweight	0.815	1.484	3.350
7	triweight	0.808	1.496	3.424
11	triweight	0.785	1.453	3.315
21	triweight	0.734	1.312	2.934
31	triweight	0.688	1.194	2.546
1	cos	0.827	1.497	3.349
3	cos	0.823	1.487	3.337
5	cos	0.802	1.485	3.391
7	cos	0.764	1.406	3.196
11	cos	0.745	1.325	2.937
21	cos	0.691	1.190	2.512
31	cos	0.623	1.087	2.317
1	inv	0.827	1.497	3.349
3	inv	0.828	1.517	3.433
5	inv	0.801	1.487	3.393
7	inv	0.777	1.418	3.214
11	inv	0.750	1.364	3.068
21	inv	0.687	1.194	2.566
31	inv	0.628	1.083	2.299

Tabelle C.37: Fehlerraten bei den Wachheits-Daten ohne Verwendung von Median oder y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.726	1.157	2.243
max. 25	epanechnikov	0.772	1.204	2.280
max. 25	all kernels	0.740	1.186	2.302

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.827	1.497	3.349
3	rectangular	0.785	1.303	2.625
5	rectangular	0.798	1.279	2.473
7	rectangular	0.767	1.208	2.302
11	rectangular	0.764	1.178	2.190
21	rectangular	0.733	1.139	2.139
31	rectangular	0.713	1.158	2.288
1	triangular	0.827	1.497	3.349
3	triangular	0.804	1.425	3.109
5	triangular	0.813	1.377	2.845
7	triangular	0.800	1.314	2.604
11	triangular	0.789	1.248	2.384
21	triangular	0.767	1.172	2.166
31	triangular	0.733	1.148	2.192
1	epanechnikov	0.827	1.497	3.349
3	epanechnikov	0.802	1.415	3.057
5	epanechnikov	0.808	1.349	2.729
7	epanechnikov	0.792	1.285	2.501
11	epanechnikov	0.774	1.221	2.321
21	epanechnikov	0.764	1.169	2.167
31	epanechnikov	0.737	1.161	2.221
1	biweight	0.827	1.497	3.349
3	biweight	0.815	1.464	3.232
5	biweight	0.816	1.429	3.059
7	biweight	0.805	1.345	2.723
11	biweight	0.797	1.280	2.494
21	biweight	0.774	1.188	2.210
31	biweight	0.740	1.148	2.158
1	triweight	0.827	1.497	3.349
3	triweight	0.822	1.471	3.233
5	triweight	0.809	1.436	3.124
7	triweight	0.813	1.405	2.961
11	triweight	0.798	1.319	2.647
21	triweight	0.779	1.220	2.316
31	triweight	0.757	1.158	2.142
1	cos	0.827	1.497	3.349
3	cos	0.798	1.417	3.093
5	cos	0.808	1.361	2.791
7	cos	0.796	1.295	2.535
11	cos	0.776	1.234	2.362
21	cos	0.768	1.175	2.179
31	cos	0.733	1.156	2.212
1	inv	0.827	1.497	3.349
3	inv	0.812	1.415	3.025
5	inv	0.822	1.381	2.837
7	inv	0.797	1.301	2.579
11	inv	0.776	1.220	2.318
21	inv	0.762	1.144	2.116
31	inv	0.747	1.157	2.203

Tabelle C.38: Fehlerraten bei den Wachheits-Daten mit Verwendung des Medians, aber ohne y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.778	1.137	1.953
max. 25	epanechnikov	0.798	1.158	1.962
max. 25	all kernels	0.791	1.147	1.955

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.819	1.374	2.806
3	rectangular	0.716	1.092	1.986
5	rectangular	0.790	1.145	1.929
7	rectangular	0.773	1.092	1.798
11	rectangular	0.810	1.135	1.815
21	rectangular	0.853	1.175	1.841
31	rectangular	0.860	1.180	1.834
1	triangular	0.819	1.374	2.806
3	triangular	0.791	1.209	2.205
5	triangular	0.775	1.163	2.061
7	triangular	0.774	1.114	1.880
11	triangular	0.783	1.093	1.773
21	triangular	0.817	1.128	1.770
31	triangular	0.835	1.149	1.789
1	epanechnikov	0.819	1.374	2.806
3	epanechnikov	0.785	1.188	2.144
5	epanechnikov	0.776	1.150	2.012
7	epanechnikov	0.770	1.102	1.842
11	epanechnikov	0.779	1.091	1.765
21	epanechnikov	0.821	1.135	1.777
31	epanechnikov	0.839	1.154	1.796
1	biweight	0.819	1.374	2.806
3	biweight	0.782	1.222	2.288
5	biweight	0.767	1.179	2.147
7	biweight	0.780	1.149	1.989
11	biweight	0.783	1.105	1.827
21	biweight	0.803	1.107	1.745
31	biweight	0.819	1.135	1.783
1	triweight	0.819	1.374	2.806
3	triweight	0.773	1.221	2.315
5	triweight	0.770	1.193	2.199
7	triweight	0.773	1.168	2.086
11	triweight	0.780	1.129	1.931
21	triweight	0.792	1.098	1.760
31	triweight	0.812	1.118	1.750
1	cos	0.819	1.374	2.806
3	cos	0.784	1.191	2.155
5	cos	0.779	1.154	2.024
7	cos	0.773	1.107	1.853
11	cos	0.778	1.093	1.781
21	cos	0.820	1.131	1.769
31	cos	0.836	1.152	1.796
1	inv	0.819	1.374	2.806
3	inv	0.784	1.198	2.170
5	inv	0.786	1.145	1.975
7	inv	0.788	1.110	1.846
11	inv	0.809	1.117	1.781
21	inv	0.852	1.167	1.819
31	inv	0.865	1.189	1.855

Tabelle C.39: Fehlerraten bei den Wachheits-Daten mit Verwendung von Median und y-Kern (Dargestellt sind die Ergebnisse für q=1; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.648	1.115	2.347
max. 25	epanechnikov	0.694	1.199	2.541
max. 25	all kernels	0.654	1.121	2.361

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.820	1.453	3.195
3	rectangular	0.737	1.317	2.989
5	rectangular	0.758	1.391	3.197
7	rectangular	0.712	1.280	2.866
11	rectangular	0.685	1.186	2.528
21	rectangular	0.632	1.098	2.330
31	rectangular	0.602	1.050	2.240
1	triangular	0.820	1.453	3.195
3	triangular	0.821	1.456	3.218
5	triangular	0.805	1.491	3.415
7	triangular	0.792	1.457	3.337
11	triangular	0.770	1.402	3.192
21	triangular	0.710	1.239	2.669
31	triangular	0.646	1.121	2.379
1	epanechnikov	0.820	1.453	3.195
3	epanechnikov	0.818	1.452	3.216
5	epanechnikov	0.797	1.462	3.330
7	epanechnikov	0.780	1.421	3.229
11	epanechnikov	0.757	1.352	3.014
21	epanechnikov	0.692	1.186	2.492
31	epanechnikov	0.627	1.094	2.330
1	biweight	0.820	1.453	3.195
3	biweight	0.821	1.456	3.220
5	biweight	0.809	1.467	3.307
7	biweight	0.796	1.459	3.331
11	biweight	0.782	1.451	3.345
21	biweight	0.739	1.323	2.945
31	biweight	0.670	1.149	2.419
1	triweight	0.820	1.453	3.195
3	triweight	0.822	1.456	3.204
5	triweight	0.814	1.479	3.329
7	triweight	0.808	1.472	3.328
11	triweight	0.800	1.477	3.385
21	triweight	0.761	1.393	3.185
31	triweight	0.706	1.232	2.670
1	cos	0.820	1.453	3.195
3	cos	0.820	1.456	3.222
5	cos	0.799	1.475	3.379
7	cos	0.781	1.425	3.235
11	cos	0.761	1.369	3.079
21	cos	0.701	1.219	2.611
31	cos	0.637	1.108	2.356
1	inv	0.820	1.453	3.195
3	inv	0.822	1.472	3.284
5	inv	0.823	1.539	3.547
7	inv	0.797	1.484	3.402
11	inv	0.739	1.339	3.005
21	inv	0.692	1.214	2.616
31	inv	0.620	1.069	2.267

Tabelle C.40: Fehlerraten bei den Wachheits-Daten ohne Verwendung von Median oder y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.744	1.184	2.264
max. 25	epanechnikov	0.768	1.173	2.155
max. 25	all kernels	0.769	1.186	2.202

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.820	1.453	3.195
3	rectangular	0.794	1.289	2.537
5	rectangular	0.793	1.248	2.358
7	rectangular	0.784	1.240	2.362
11	rectangular	0.758	1.171	2.167
21	rectangular	0.755	1.166	2.168
31	rectangular	0.720	1.165	2.299
1	triangular	0.820	1.453	3.195
3	triangular	0.816	1.407	2.979
5	triangular	0.810	1.335	2.655
7	triangular	0.801	1.300	2.532
11	triangular	0.772	1.233	2.371
21	triangular	0.767	1.168	2.148
31	triangular	0.745	1.150	2.150
1	epanechnikov	0.820	1.453	3.195
3	epanechnikov	0.809	1.377	2.867
5	epanechnikov	0.795	1.297	2.551
7	epanechnikov	0.796	1.280	2.472
11	epanechnikov	0.773	1.215	2.297
21	epanechnikov	0.763	1.164	2.140
31	epanechnikov	0.736	1.154	2.188
1	biweight	0.820	1.453	3.195
3	biweight	0.811	1.416	3.052
5	biweight	0.810	1.354	2.762
7	biweight	0.804	1.301	2.539
11	biweight	0.784	1.265	2.451
21	biweight	0.762	1.187	2.223
31	biweight	0.750	1.143	2.097
1	triweight	0.820	1.453	3.195
3	triweight	0.816	1.446	3.168
5	triweight	0.819	1.424	3.020
7	triweight	0.806	1.339	2.705
11	triweight	0.805	1.313	2.581
21	triweight	0.769	1.226	2.354
31	triweight	0.758	1.169	2.175
1	cos	0.820	1.453	3.195
3	cos	0.814	1.399	2.947
5	cos	0.803	1.306	2.568
7	cos	0.799	1.287	2.489
11	cos	0.773	1.219	2.309
21	cos	0.765	1.167	2.143
31	cos	0.736	1.147	2.161
1	inv	0.820	1.453	3.195
3	inv	0.814	1.402	2.970
5	inv	0.814	1.364	2.772
7	inv	0.792	1.291	2.543
11	inv	0.770	1.227	2.355
21	inv	0.765	1.158	2.142
31	inv	0.745	1.154	2.194

Tabelle C.41: Fehlerraten bei den Wachheits-Daten mit Verwendung des Medians, aber ohne y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
max. 25	rectangular	0.778	1.126	1.900
max. 25	epanechnikov	0.785	1.143	1.931
max. 25	all kernels	0.795	1.137	1.885

k	Kern	Testfehlerraten Fehlklassifikation	absoluter Abstand	quadrierter Abstand
1	rectangular	0.812	1.344	2.718
3	rectangular	0.715	1.059	1.845
5	rectangular	0.782	1.130	1.888
7	rectangular	0.794	1.119	1.825
11	rectangular	0.814	1.145	1.843
21	rectangular	0.845	1.161	1.801
31	rectangular	0.862	1.175	1.809
1	triangular	0.812	1.344	2.718
3	triangular	0.774	1.176	2.114
5	triangular	0.773	1.113	1.873
7	triangular	0.766	1.084	1.778
11	triangular	0.776	1.083	1.745
21	triangular	0.800	1.108	1.738
31	triangular	0.830	1.142	1.776
1	epanechnikov	0.812	1.344	2.718
3	epanechnikov	0.777	1.172	2.072
5	epanechnikov	0.773	1.095	1.803
7	epanechnikov	0.763	1.085	1.785
11	epanechnikov	0.768	1.080	1.748
21	epanechnikov	0.811	1.125	1.765
31	epanechnikov	0.836	1.149	1.785
1	biweight	0.812	1.344	2.718
3	biweight	0.787	1.199	2.169
5	biweight	0.775	1.135	1.945
7	biweight	0.770	1.105	1.841
11	biweight	0.765	1.082	1.770
21	biweight	0.784	1.087	1.725
31	biweight	0.811	1.117	1.739
1	triweight	0.812	1.344	2.718
3	triweight	0.775	1.199	2.213
5	triweight	0.768	1.148	2.010
7	triweight	0.771	1.126	1.932
11	triweight	0.767	1.115	1.895
21	triweight	0.767	1.072	1.734
31	triweight	0.790	1.089	1.699
1	cos	0.812	1.344	2.718
3	cos	0.773	1.171	2.089
5	cos	0.774	1.101	1.821
7	cos	0.761	1.080	1.770
11	cos	0.774	1.083	1.747
21	cos	0.806	1.123	1.773
31	cos	0.831	1.143	1.777
1	inv	0.812	1.344	2.718
3	inv	0.783	1.169	2.059
5	inv	0.785	1.143	1.957
7	inv	0.789	1.119	1.861
11	inv	0.808	1.122	1.796
21	inv	0.856	1.177	1.839
31	inv	0.867	1.186	1.840

Tabelle C.42: Fehlerraten bei den Wachheits-Daten mit Verwendung von Median und y-Kern (Dargestellt sind die Ergebnisse für q=2; Die obere Tabelle beinhaltet die Ergebnisse für kreuzvalidierte Parameterwerte)

Anhang D

Verlaufsdiagramme von ordinalem Boosting (ausführliche Übersicht)

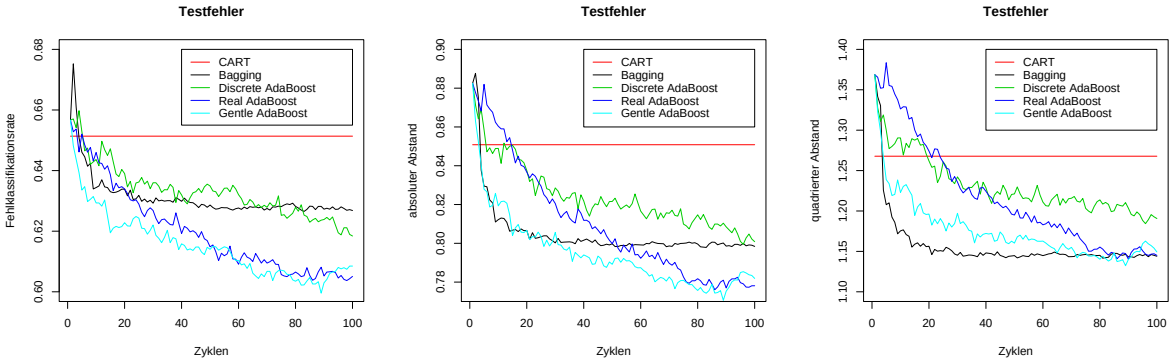

Abbildung D.1: Entwicklung der Testfehler bei der Kundenumfrage (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind nominale Ensemble-Techniken.

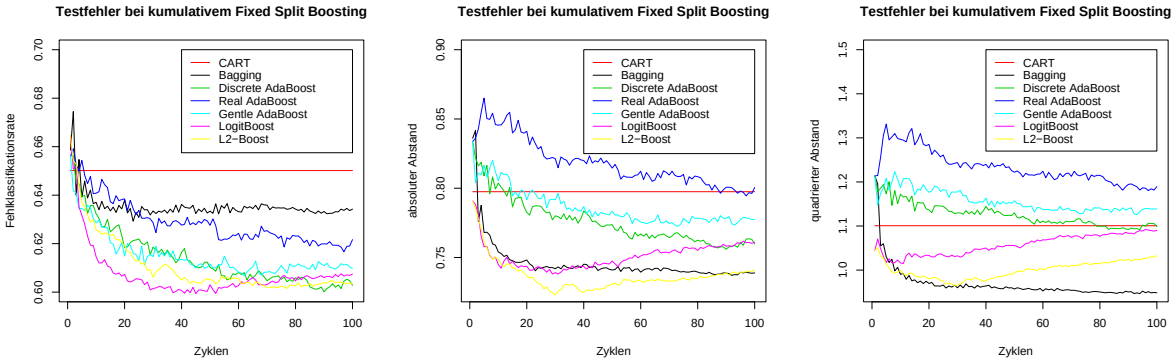

Abbildung D.2: Entwicklung der Testfehler bei der Kundenumfrage (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind kumulative Fixed-Split-Techniken.

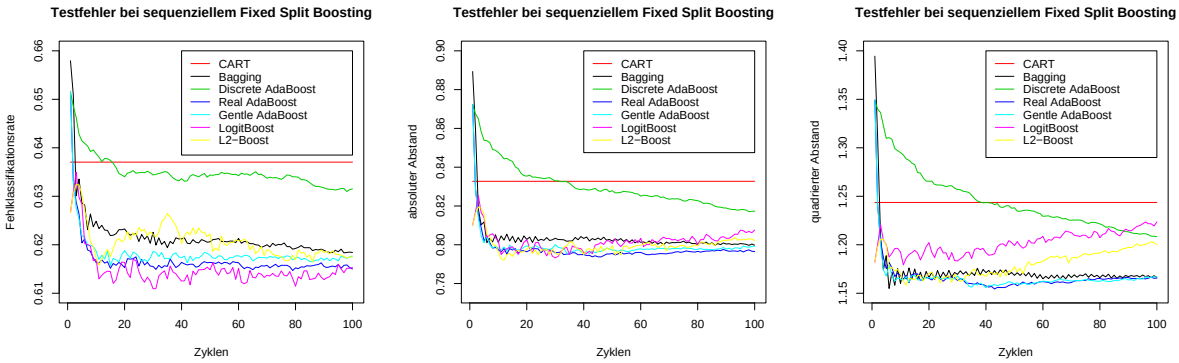

Abbildung D.3: Entwicklung der Testfehler bei der Kundenumfrage (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind sequenzielle Fixed-Split-Techniken.

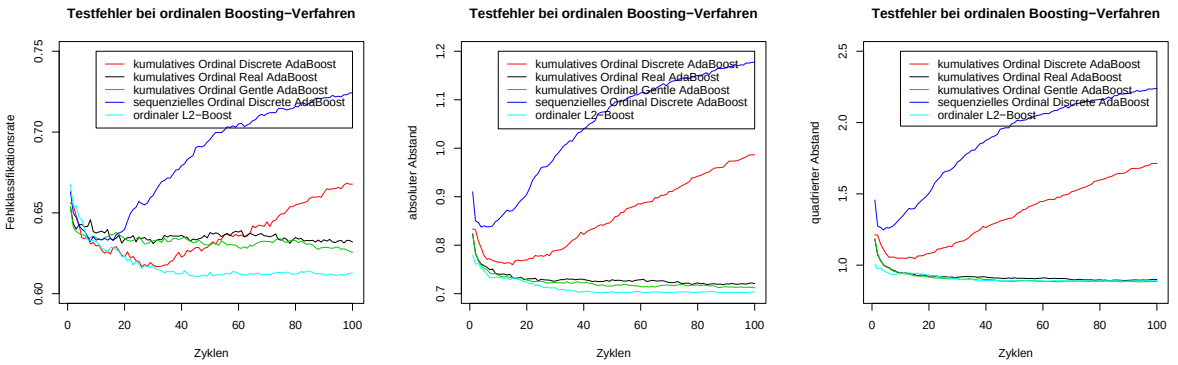

Abbildung D.4: Entwicklung der Testfehler bei der Kundenumfrage (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind ordinale Boosting-Techniken.

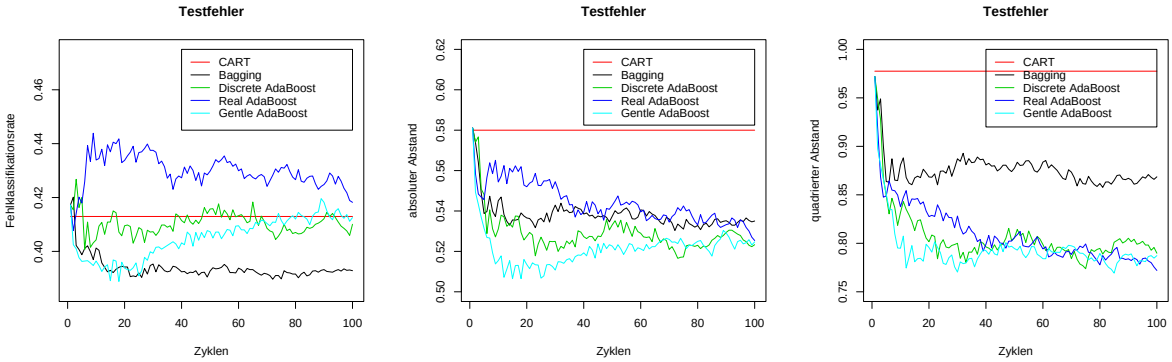

Abbildung D.5: Entwicklung der Testfehler bei den Herzerkrankungs-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind nominale Ensemble-Techniken.

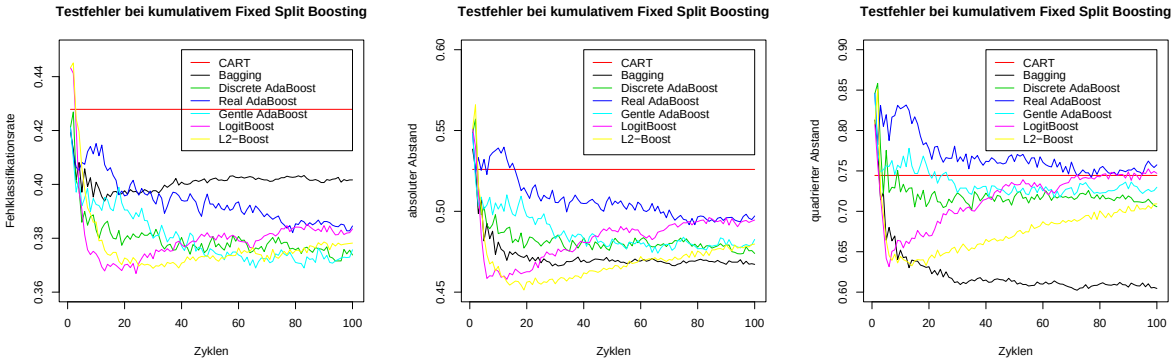

Abbildung D.6: Entwicklung der Testfehler bei den Herzerkrankungs-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind kumulative Fixed-Split-Techniken.

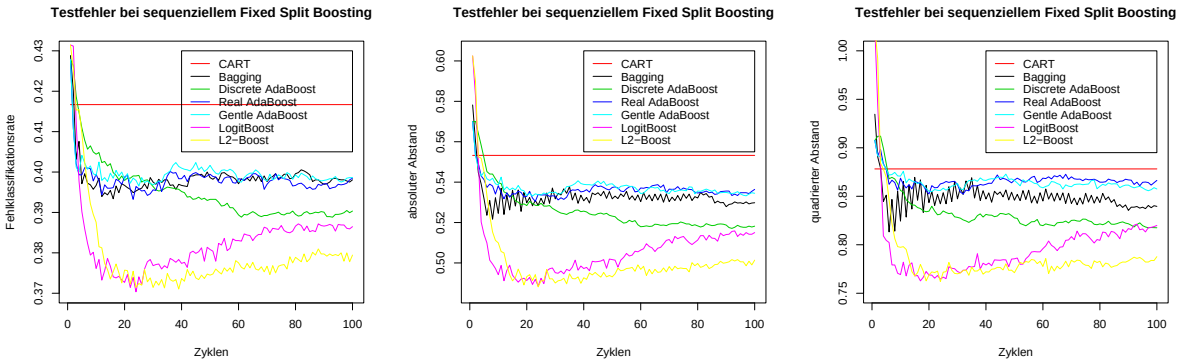

Abbildung D.7: Entwicklung der Testfehler bei den Herzerkrankungs-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind sequenzielle Fixed-Split-Techniken.

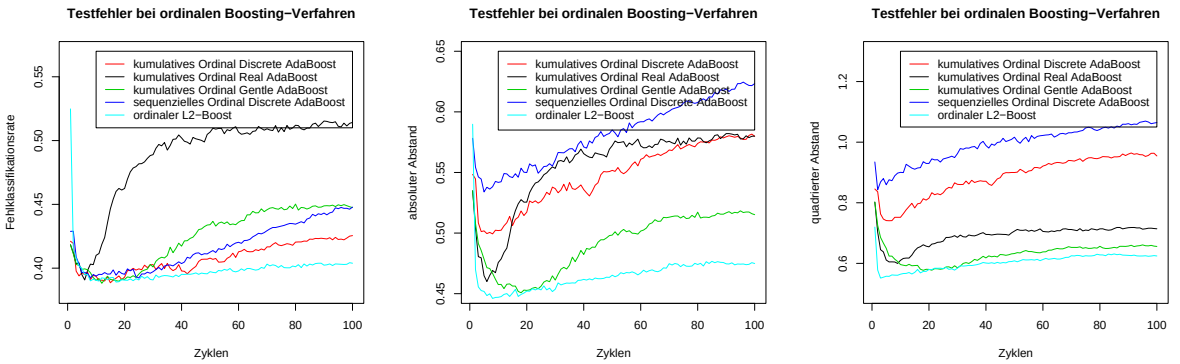

Abbildung D.8: Entwicklung der Testfehler bei den Herzerkrankungs-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind ordinale Boosting-Techniken.

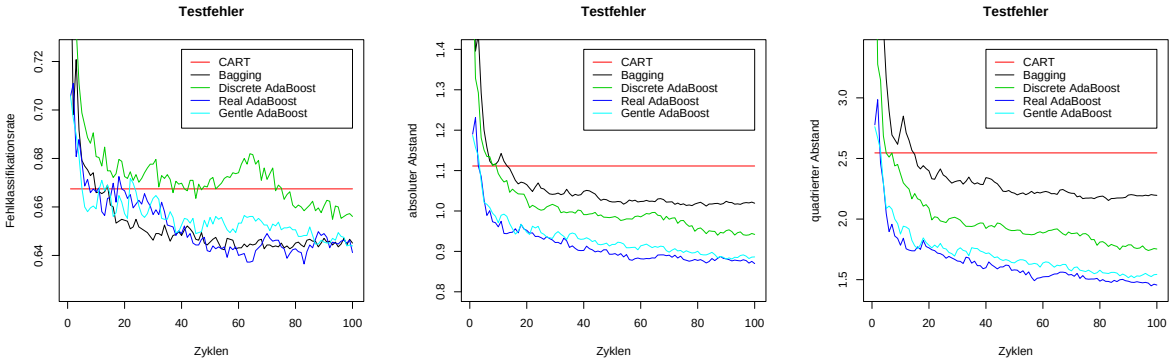

Abbildung D.9: Entwicklung der Testfehler bei den Scapula-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind nominale Ensemble-Techniken.

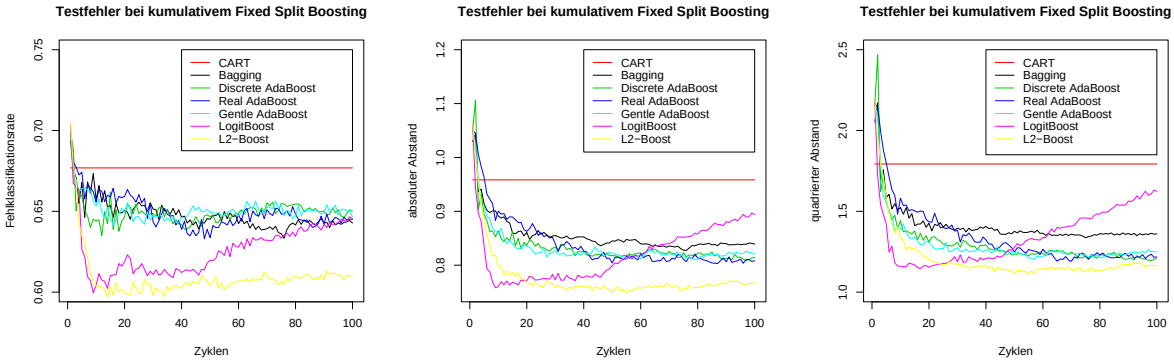

Abbildung D.10: Entwicklung der Testfehler bei den Scapula-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind kumulative Fixed-Split-Techniken.

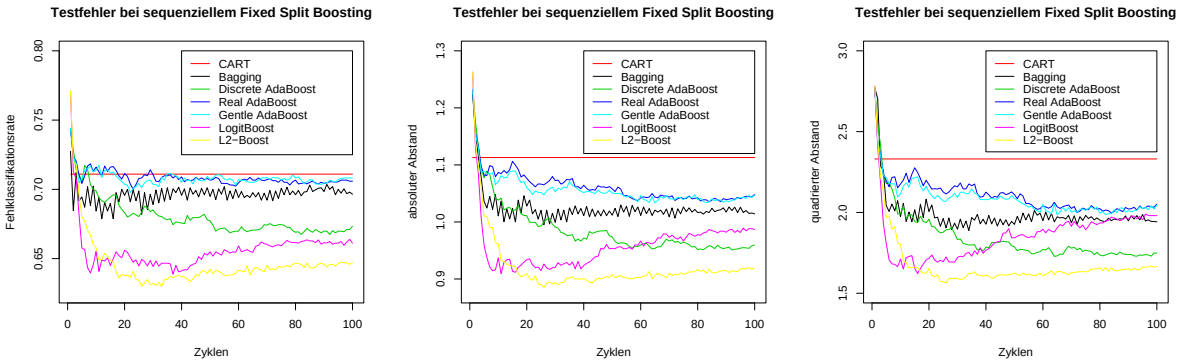

Abbildung D.11: Entwicklung der Testfehler bei den Scapula-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind sequenzielle Fixed-Split-Techniken.

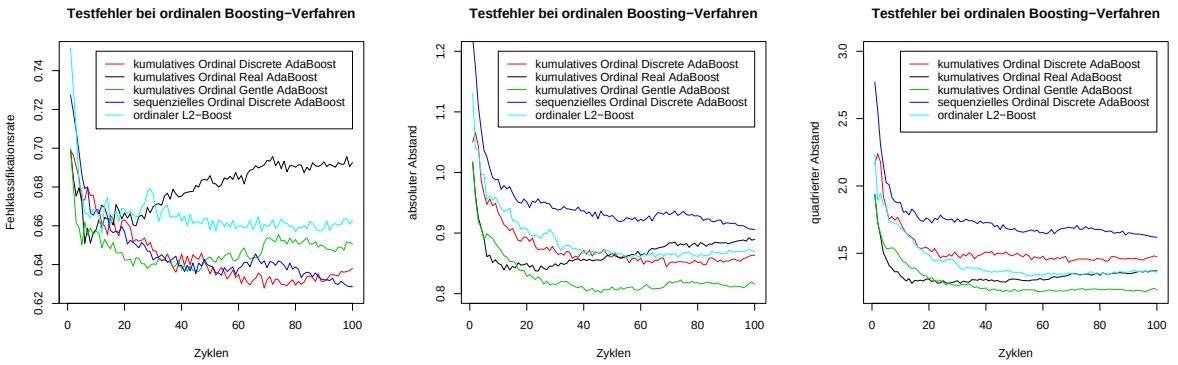

Abbildung D.12: Entwicklung der Testfehler bei den Scapula-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind ordinale Boosting-Techniken.

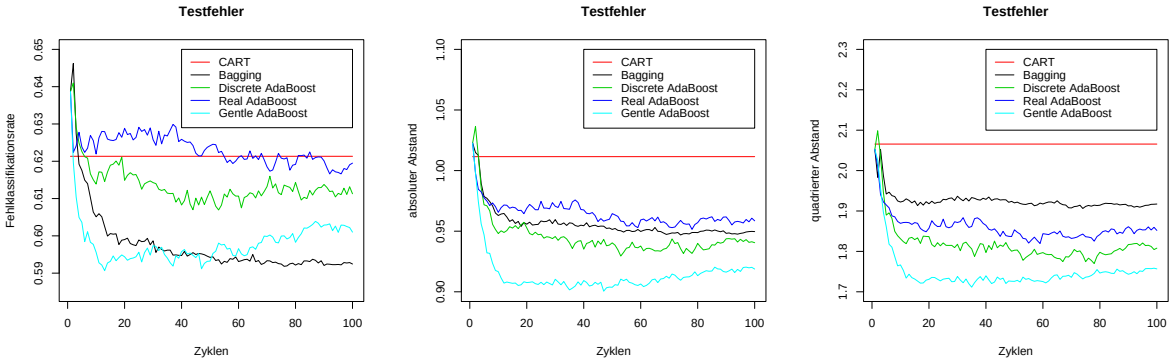

Abbildung D.13: Entwicklung der Testfehler bei den Mietspiegel-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind nominale Ensemble-Techniken.

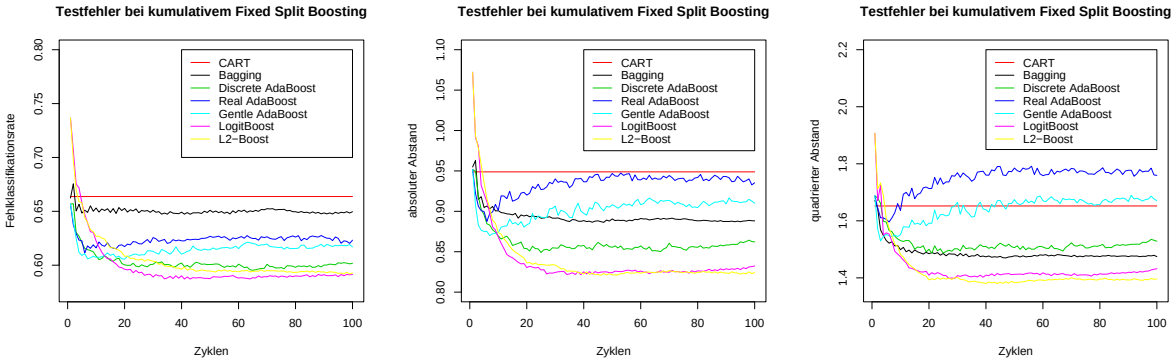

Abbildung D.14: Entwicklung der Testfehler bei den Mietspiegel-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind kumulative Fixed-Split-Techniken.

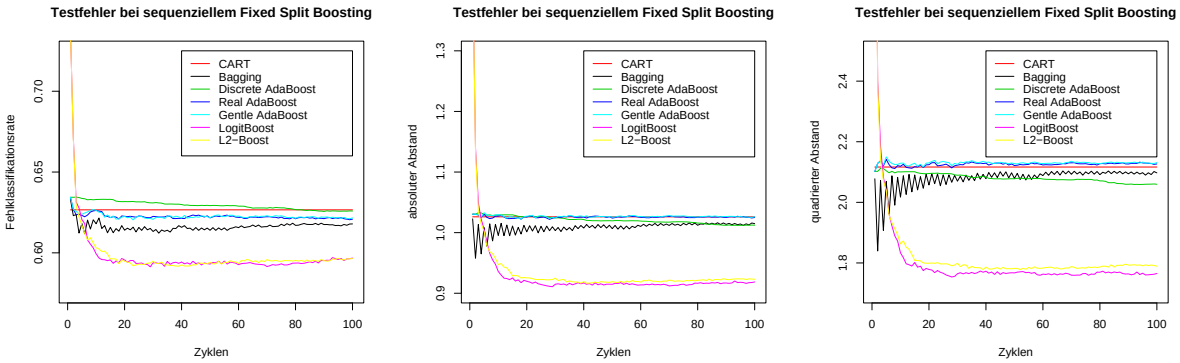

Abbildung D.15: Entwicklung der Testfehler bei den Mietspiegel-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind sequenzielle Fixed-Split-Techniken.

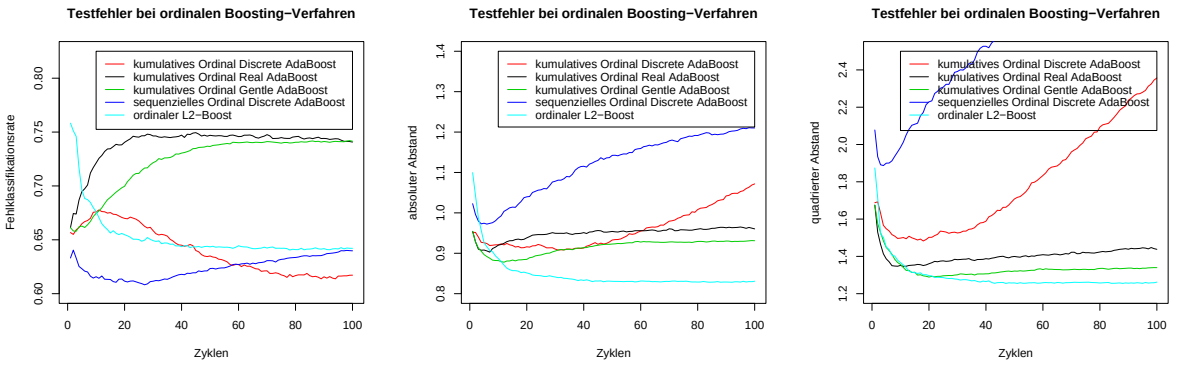

Abbildung D.16: Entwicklung der Testfehler bei den Mietspiegel-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind ordinale Boosting-Techniken.

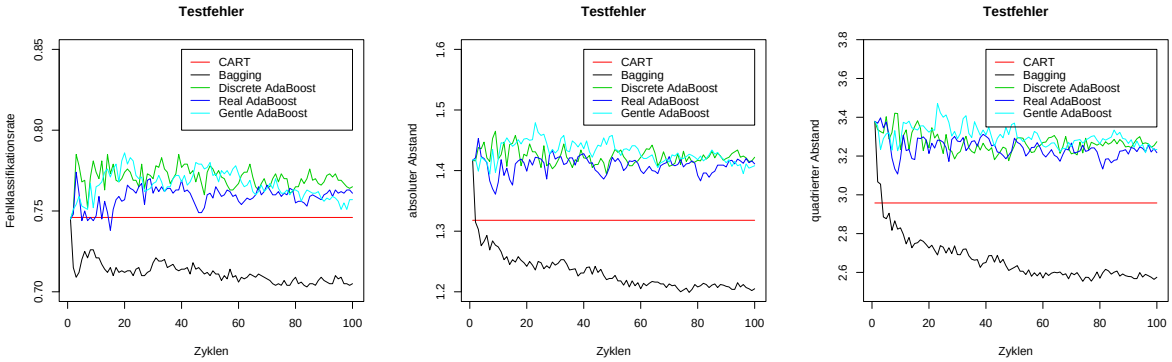

Abbildung D.17: Entwicklung der Testfehler bei den Wachheits-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind nominale Ensemble-Techniken.

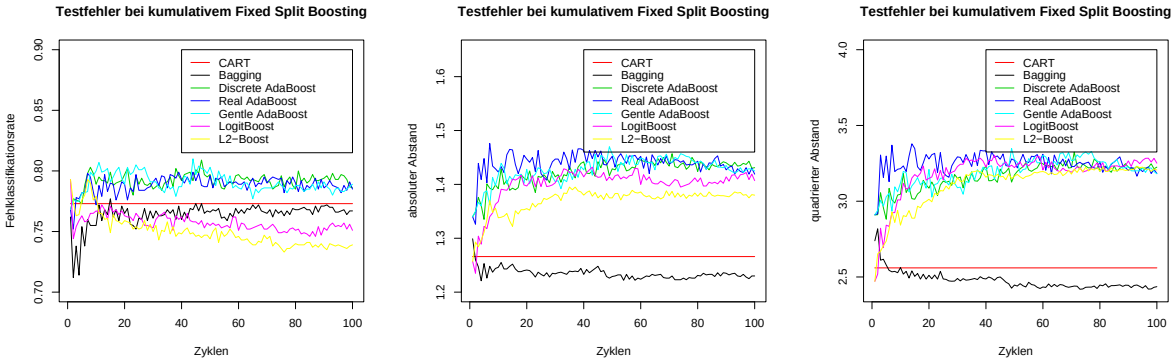

Abbildung D.18: Entwicklung der Testfehler bei den Wachheits-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind kumulative Fixed-Split-Techniken.

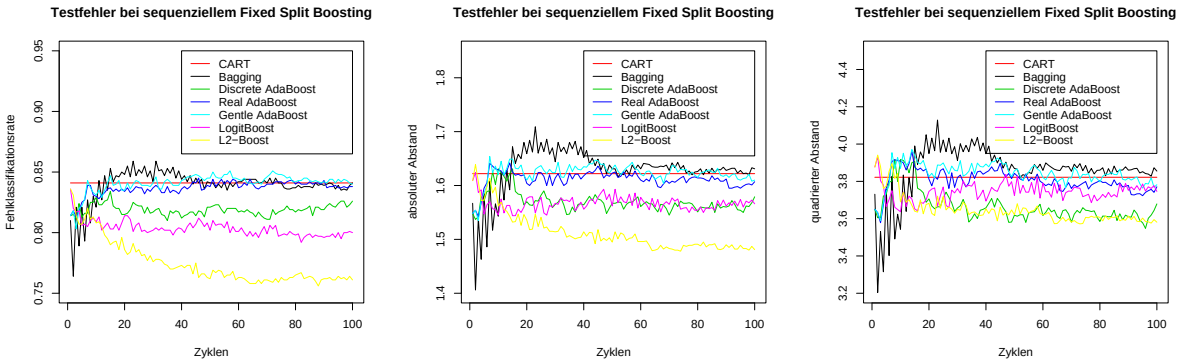

Abbildung D.19: Entwicklung der Testfehler bei den Wachheits-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind sequenzielle Fixed-Split-Techniken.

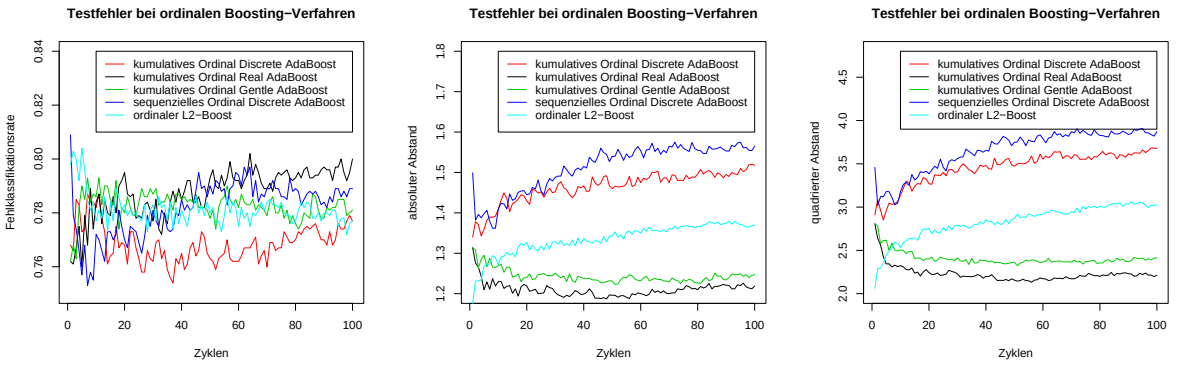

Abbildung D.20: Entwicklung der Testfehler bei den Wachheits-Daten (links: Fehlklassifikationsrate, Mitte: absoluter Abstand, rechts: quadrierter Abstand); Dargestellt sind ordinale Boosting-Techniken.

Literatur

Anderson, J. und Philips, P. (1981). Regression, discrimination and measurement models for ordered categorical variables. *Applied Statistics* **30**, 22–31.

Bauer, E. und Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning* **36**, 105–142.

Bühlmann, P. (2002). Consistency for l_2-boosting and matching pursuit with trees and tree-type basis functions. Research Report 109, ETH Zürich.

Bühlmann, P. (2003). Bagging, subagging and bragging for improving some prediction algorithms. To appear in: Recent Advances and Trends in Nonparametric Statistics.

Bühlmann, P. und Yu, B. (2002a). Analyzing bagging. *Annals of Statistics* **30**, 927–961.

Bühlmann, P. und Yu, B. (2002b). Boosting with the l_2-loss: Regression and classification. To appear in: Journal of the American Statistical Association.

Bishop, C. und Tipping, M. (2000). Variational relevance vector machines. Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence.

Boulesteix, A.-L. (2004). Pls dimension reduction for classification with microarray data. *Statistical Applications in Genetics and Molecular Biology* **3**.

Breiman, L. (1996). Bagging predictors. *Machine Learning* **24**, 123–140.

Breiman, L. (1998). Arcing classifiers. *Annals of Statistics* **26**, 801–849.

Breiman, L. (2000). Some infinity theory for predictor ensembles. Technical Report 577, University of California, Berkeley.

Breiman, L. (2001). Random forests. *Machine Learning* **45**, 5–32.

Breiman, L., Friedman, J., Olshen, R., und Stone, C. (1984). *Classification and Regression Trees*. Boca Raton - London - New York - Washington D.C.: Chapman & Hall.

Chen, Y., Härdle, W., und Schulz, R. (2004). Prognose mit nichtparametrischen Verfahren. Technical Report, Humboldt-Universität zu Berlin.

Clark, L. und Pregibon, D. (1992). Tree-based models. In J. Chambers & T. Hastie (Hrsg.), *Statistical Models in S*, S. 377–419. Pacific Grove: Wadsworth & Brooks.

Cleveland, W. und Loader, C. (1995). Smoothing by local regression: Principles and methods. Technical Report, AT&T Bell Laboratories, Murray Hill, NY.

Cost, S. und Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning* **10**, 57–78.

Cover, T. und Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* **13**, 21–27.

Dettling, M. (2004). Bagboosting for tumor classification with gene expression data. Technical Report No. 122, Seminar für Statistik, ETH Zürich.

Dettling, M. und Bühlmann, P. (2003). Boosting for tumor classification with gene expression data. *Bioinformatics* **19**, 1061–1069.

Dietterich, T. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning* **40**, 139–157.

Dudoit, S., Fridlyand, J., und Speed, T. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association* **97**, 77–87.

Efron, B., Hastie, T., Johnstone, I., und Tibshirani, R. (2002). Least angle regression. Technical Report 220, Stanford University.

Fahrmeir, L., Hamerle, A., und Tutz, G. (1996). *Multivariate statistische Verfahren*. Berlin - New York: de Gruyter.

Fahrmeir, L., Künstler, R., Pigeot, I., und Tutz, G. (1997). *Statistik: der Weg zur Datenanalyse*. Berlin: Springer.

Fix, E. und Hodges, J. (1951). Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical Report 4, U.S. Air Force, School of Aviation Medicine, Randolph Field, TX.

Frank, E. und Hall, M. (2001). A simple approach to ordinal classification. Working Paper 01/05, Department of Computer Science, University of Waikato.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information And Computation* **121**, 256–285.

Freund, Y. und Schapire, R. (1996). Experiments with a new boosting algorithm. Machine Learning: Proceedings of the Thirteenth International Conference.

Friedman, J. (1994). Flexible metric nearest neighbor classification. Technical Report 113, Stanford University, Statistics Department.

Friedman, J. (1999). Stochastic gradient boosting. Technical Report, Stanford University.

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. Technical Report, Stanford University.

Friedman, J. und Hall, P. (2000). On bagging and nonlinear estimation. Technical Report, Stanford University.

Friedman, J., Hastie, T., und Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *Annals of Statistics* **28**, 337–374.

Furey, T., Cristianini, N., Duffy, N., Bednarski, D., Schummer, M., und Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics* **16**, 906–914.

Hastie, T., Tibshirani, R., und Friedman, J. (2001). *The Elements of Statistical Learning*. New York: Springer.

Hechenbichler, K. und Schliep, K. (2004). Weighted k-nearest-neighbor techniques and ordinal classification. Discussion Paper 399, SFB 386 der Ludwig-Maximilians-Universität München.

Hothorn, T. und Lausen, B. (2003). Double-bagging: Combining classifiers by bootstrap

aggregation. *Pattern Recognition* **36**, 1303–1309.

Kahn, J., Wei, J., Ringner, M., Saal, L., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C., Peterson, C., und Meltzer, P. (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine* **7**, 673–679.

Kramer, S., Widmer, G., Pfahringer, B., und De Groeve, M. (2001). Prediction of ordinal classes using regression trees. *Fundamenta Informaticae* **47**, 1–15.

McLachlan, G. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley.

Nguyen, D. und Rocke, D. (2002). Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics* **18**, 39–50.

Paik, M. und Yang, Y. (2004). Combining nearest neighbor classifiers versus cross-validation selection. *Statistical Applications in Genetics and Molecular Biology* **3**, Article 12.

Ripley, B. (1996). *Pattern Recognition and Neural Networks*. Cambridge: University Press.

Rudolfer, S., Watson, P., und Lesaffre, E. (1995). Are ordinal models useful for classification? a revised analysis. *Journal of Statistical Computation and Simulation* **52**, 105–132.

Salzberg, S. (1999). On comparing classifiers: A critique of current research and methods. *Data Mining and Knowledge Discovery* **1**, 1–12.

Schapire, R. (2002). The boosting approach to machine learning: An overview. MSRI Workshop on Nonlinear Estimation and Classification.

Schapire, R., Freund, Y., Bartlett, P., und Lee, W. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics* **26**, 1651–1686.

Tibshirani, R., Hastie, T., Narasimhan, B., und Chu, G. (2002). Diagnosis of multiple cancer types by shrunken centroids of gene expression. *PNAS* **99**, 6567–6572.

Tutz, G. (2000). *Die Analyse kategorialer Daten*. München: Oldenbourg.

Tutz, G. und Hechenbichler, K. (2005). Aggregating classifiers with ordinal response structure. *Journal of Statistical Computation and Simulation* **75**, 391–408.

Danksagung

Wenn ich einem meiner liebsten Hobbys nachgehe und mir neue Alben, bevorzugt von eher unbekannten Bands aus dem Bereich der alternativen Rockmusik, kaufe, dann gibt es etwas, das mir genauso wichtig ist, wie das Anhören der Musik: Aus den Grüßen und Danksagungen der Bands an Kollegen, Produzenten oder Plattenlabels lassen sich nämlich die vielen interessanten Verbindungen und Netzwerke rekonstruieren, die entscheidend für Stilprägungen und musikalische Entwicklungen sind und die einen tieferen Einblick in die Szene erlauben. Erfahrungsgemäß sind diese Danksagungen meist sehr umfangreiche Listen, aus denen man sich die wenigen wichtigen Einträge mühsam heraussuchen muß.

Ähnlich umfangreiche Listen hatte ich im Kopf, als ich über diese Danksagung im Rahmen meiner Dissertation nachdachte. Da ich mein gesamtes bisheriges Leben in München verbracht habe und demzufolge in einem sehr konstanten und gefestigten Netzwerk aus sozialen Kontakten eingebunden bin, droht eine solche Danksagung natürlich zu einer schnöden Aufzählung zu verkommen. Trotzdem möchte ich hier viele Personen aus meinem Umfeld aufzählen, die in den fünf Jahren, in denen ich mit dem Thema meiner Arbeit beschäftigt war, eine wichtige Rolle für mich gespielt haben und spielen, auch wenn sie zum Teil mit der Arbeit an dieser Dissertation nicht unmittelbar in Verbindung stehen.

Zunächst darf ich hier natürlich meinen Eltern und meiner Oma danken, die mir den langen und beschwerlichen Ausbildungsweg über Abitur, Studium und Promotion überhaupt ermöglicht haben. Das “Hotel Mama“ während des Großteils des Studiums war mir eine große Hilfe.

Ebenso wichtig war und ist mein Schul-Freundeskreis, den ich mehr oder weniger direkt über das Gymnasium kennen gelernt habe. Nennen möchte ich hier Mirsch, List, Timm, Frank, Till, Meisen, Franz, Wolfgang und Pete, die ich zum Teil schon länger als 20 Jahre kenne. Zu ihnen ist der Kontakt nie abgerissen und mit diversen Freizeitaktivitäten konnten sie mich jederzeit von Sackgassen und Schwierigkeiten in der Forschung ablenken und wieder auf andere Gedanken bringen. Besonders hervorheben darf ich hier aber Regina und Flo, die mir in verschiedenen Abschnitten in den letzten Jahren besonders nahe gestanden sind.

Bevor ich mich den Statistikern zuwende, möchte ich noch den folgenden “fachfremden“ Gruppierungen danken: Den Eberwurzern (und hier vor allem dem Ex-Sportdirektor Gary)

für die körperliche Ertüchtigung, unserer Hütten- und Brettspielrunde für die strategische Grundausbildung, den beiden Münchner Alternative-Tempeln Backstage und Keller für ihre musikalischen Fortbildungsseminare, sowie dem FC Bayern München und dem VfL Bochum für zahlreiche entspannende (und noch viel mehr nervenstrapazierende) Stadionbesuche.

Wenn ich nun also auf das Institut für Statistik zu sprechen komme, sei zunächst die angenehme und familiäre Arbeitsatmosphäre genannt, die ich ja insgesamt immerhin über 10 Jahre lang genießen durfte. Was die eigentliche Dissertation betrifft, so möchte ich mich in erster Linie natürlich bei Herrn Professor Tutz für Thema und Betreuung bedanken. Aber auch alle Kollegen, egal ob Professoren oder Assistenten, hatten immer ein offenes Ohr für Fragen oder Probleme. Stellvertretend nennen will ich hier Volker und Holger, da ich sie bereits unmittelbar seit Beginn meines Sudiums kenne, und sie in der langen gemeinsamen Zeit zu sehr engen Freunden geworden sind. In die gleiche Kategorie fällt Klaus (zur Unterscheidung von mir in Fachkreisen auch manchmal "der kleine Klaus" genannt), dem wegen unserer gemeinsamen Entwicklung der gewichteten Nächste-Nachbarn-Technik im Speziellen und seiner Hilfe bei der R-Programmierung im Allgemeinen ein besonders großes Dankeschön zusteht.

Desweiteren durfte ich im Rahmen meiner Arbeit am Institut noch viele nette Menschen kennen lernen, die mir sicherlich auch nach meiner Zeit an der Universität noch erhalten bleiben werden. Deren Nennung allein würde den Rahmen dieser Danksagung sprengen, also seien stellvertretend Flo, Ingrid, Inga und besonders Ela und Verena, genannt, denen allensamt ein großer Anteil daran gebührt, daß ich die Zeit meiner Promotion immer in positiver Erinnerung behalten werde.

Zuguterletzt geht ein besonderer Dank an Ursula, die mich auch in stressigen Zeiten mit viel Geduld ertragen hat, mir bei der Fertigstellung der Arbeit eine große Hilfe war, und mich hoffentlich noch in vielen weiteren Lebensabschnitten begleiten wird.

München, den 20. November 2005

Klaus Hechenbichler

Lebenslauf

Klaus Hechenbichler

Schleißheimer Str. 515
80933 München

10.11.1973	geboren in München
1980 bis 1984	Besuch der Bad-Soden-Grundschule in München
1984 bis 1993	Besuch des Lion-Feuchtwanger-Gymnasiums in München
1993	Abitur (Note 1,6)
1993 bis 1994	Absolvierung des Zivildienstes beim Bund Naturschutz in München sowie im Krankenhaus München-Schwabing
1994 bis 2000	Studium der Statistik an der LMU München
2000	Diplom (Note 1,15)
2000 bis 2005	Beschäftigung als wissenschaftlicher Mitarbeiter am Institut für Statistik der LMU München sowie am Max-Planck-Institut für Psychiatrie in München
seit 2005	Beschäftigung als Biostatistiker am Institut Dr. Schauerte in Oberhaching