Bachelorarbeit

Klassifikation und Variablenselektion bei höherdimensionalen Daten unter Verwendung von Nearest Neighbor Ensembles

Ludwig-Maximilians-Universität München
Institut für Statistik

Verfasser: Lena Schober

Gutachter: Prof. Dr. Gerhard Tutz

Eingereicht am: 27. September 2010

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während der Erstellung dieser Arbeit unterstützt haben.

Ein ganz besonderes Dankeschön an meinen Betreuer Jan Gertheiss. Vielen Dank für die fachliche Betreuung und deine hilfreichen Anregungen.

Außerdem möchte ich mich bei meinen Freunden bedanken, die sich immer geduldig alle Geschichten zur Bachelorarbeit angehört haben.

Ein herzliches Danke geht an meine Mutter, die mir eine große moralische Unterstützung war und beim Korrekturlesen der Arbeit hilfreich zur Seite stand.

Erklärung der Selbstständigkeit

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der erlaubten Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Publikationen entnommen sind, sind als solche kenntlich gemacht.

München, den 27. September 2010

Lena Schober

Inhaltsverzeichnis

1 Einführung

Klassifikationsverfahren sind in der Statistik eine viel behandeltes Themengebiet. Erst die Klassifizierung realer Informationen ermöglicht geordnete Verarbeitung, deswegen ist die Klassifizierung zentraler Bestandteil vieler Anwendungen.
Bei der Klassifikation werden Objekte anhand bestimmter Merkmale einer Klasse zugeordnet. Dies erfolgt über einen Klassifikator. Ziel der Methoden ist vor allem die Vorhersage einer Klassenzugehörigkeit für neue Objekte. Beispielsweise kann es sich dabei um die Bösartigkeit eines Tumors handelt, die mit Hilfe von Merkmalen wie Größe, Lage, ... prognostiziert werden soll. Die theoretisch optimale Entscheidung liefert die sogenannte Bayes-Klassifikation. Unter allen Entscheidungsregeln liefert sie die geringste Fehlerrate. Fehlerraten sind ein Gütekriterium um die Effizienz von Klassifikationsverfahren zu überprüfen und definieren sich als

$$\epsilon(\delta) = \frac{\text{Fehler}}{\text{Alle}} \quad \text{mit} \quad \delta = \text{Zuordnungsregel}$$

d.h. als Verhältnis falsch klassifizierter Objekte zur Anzahl insgesamt. Hierbei handelt es sich die globale empirische Fehlerrate. Weiter Fehlerraten die betrachtet werden können, in der Arbeit jedoch nicht weiter berücksichtigt werden, ist die individuelle Fehlerrate, also die Wahrscheinlichkeit, dass ein Objekt, dass der Klasse g angehört, der Klasse $\hat{g}$ zugeordnet wird. Sowie die bedingte Fehlerrate als die „Wahrscheinlichkeit einer Fehlklassifikation, wenn der Merkmalsvektor x beobachtet wird“ [Fahrmeir et al. (1996)]. Zusätzlich gibt es eine theoretische Fehlerrate, diese wird im Kapitel 2.4 genau erklärt.

Für die Bayes-Regel gilt der Ansatz eine neue Beobachtung x, deren Klassenzugehörigkeit unbekannt ist, durch gegebene Beobachtungen X, mit gegebenen Klassen, zu zuordnen. Es handelt sich dabei um ein parametrisches Verfahren. Es werden „Annahmen über [die] globale parametrische Form der klassenspezifischen Dichten“ [Nothnagel (1999)] gemacht. Die aposteriori Wahrscheinlichkeit, d.h. „die Wahrscheinlichkeit, dass ein Objekt mit

beobachtetem Merkmalsvektor x der Klasse [g] angehört“ [Fahrmeir et al. (1996)], ergibt sich dann über er den Satz von Bayes als

$$d_g(x_i) = p(g|x) = \frac{p(g)f(x|g)}{f(x)}$$

$d_g(x_i)$ heißt Diskriminanzfunktion und $p(g)$ bezeichnet die a priori Wahrscheinlichkeit der Klasse g, $f(x|g)$ die Klassenverteilung und $f(x)$ die unbedingte Verteilung von x. Als Entscheidungsregel für die Klassifikation folgt

$$\delta(x) = g \Leftrightarrow p(g|x) = \max_i p(i|x)$$

Äquivalent kann $p(g|x) \approx f(x|g)p(g)$ verwendet werden. Für den Fall gleicher apriori Wahrscheinlichkeiten, d.h $p(1) = ... = p(g)$, ergibt sich die Maximum-Likelihood-Regel. Diese ist definiert als

$$\delta(x) = g \Leftrightarrow f(x|g) = \max_i f(x|i)$$

Setzt man eine Normalverteilung

$$\frac{1}{(2\pi)^{\frac{1}{p}}|\Sigma|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}$$

mit p = Anzahl Prädiktoren und $|\Sigma|$ Determinante der Kovarianz, sowie klassenweise identischen Kovarianzen Σ voraus, spricht man von einer linearen Diskriminanzanalyse (LDA). Sind klassenweise verschiedenen Kovarianzen Grundlage handelt es sich um eine quadratische Diskriminanzanalyse (QDA).
Voraussetzung für die parametrischen Klassifikationsverfahren ist, dass die Dichte $f(x|g)$ und die apriori-Wahrscheinlichkeit $p(g)$ der neuen Beobachtung x gegeben sein müssen. [vgl. Fahrmeir et al. (1996)]
In der Realität ist dies jedoch häufig nicht der Fall. Um dieses Problem zu umgehen, können nonparametrische Klassifikationsverfahren verwendet werden.
Was ist unter dem Begriff nonparametrisch zu verstehen? Nonparametrisch

heißt wörtlich übersetzt „ohne Parameter“, d.h. es wird keine Annahme über den Parameter einer Verteilung gemacht. Häufig wird nonparametrisch mit verteilungsfrei gleichgesetzt. Verteilungsfrei ist hierbei irreführend, da eine Verteilung zugrunde liegen kann, diese aber nicht berücksichtigt wird.
Eine direkte Übersetzung für den Ausdruck „nonparametrische Klassifikation“ gibt es nicht, aber man kann sagen, es handelt sich um eine Klassifikationstechnik bei der mit gegebenen Daten und ohne Verteilungsannahmen, d.h. ohne bekannten Dichten gearbeitet wird.

Eine weitere Aspekt der für die Thematik Klassifikation interessant ist, ist der Umgang mit vielen Variablen. Wie verhält sich der Klassifikator wenn $x_1, ..., x_p$ Prädiktoren zugrunde liegen und p hohe Werte annimmt? Problematisch hierbei ist, dass die Dichte der Merkmale in der Nachbarschaft für höher Dimensionen nicht dicht besetzt ist. Häufig erfolgt daher eine Auswahl von relevanten Einflussgrößen.

In der folgenden Arbeit wird diese Problematik am Beispiel der nonparametrischen Methode der Nearest Neigbor Ensembles untersucht. Zunächst wird dazu das Prinzip Nearest Neighbor erklärt und im Anschluss die Nearest Neighbor Ensembles erläutert. Um die Effizienz der Klassifikationstechnik zu überprüfen, wird diese über ihre Prognosegenauigkeit mit dem einfachen Nearest Neighbor verglichen. Außerdem wird ein Vergleich mit einem anderen Klassifikationsverfahren, der regularisierten lineare Diskriminanzanalyse (RDA), vorgenommen.

2 Nearest Neighbor Verfahren

Das Nearest Neigbor Verfahren ist eine der ältesten Methoden zur Klassifikation. Begründer der Nearest Neigbor Regel (NN-Regel) waren E. Fix und J.L. Hodges. Sie beschrieben 1951 in einem unveröffentlichten Paper eine Methode zur Modellklassifikation. Diese ist heute als k Nearest Neighbor Regel bekannt. Ihr Konzept war Grundlage für viele weiter Verfahren, darunter z.B. distanzgewichtete Denkansätze oder auch die Fuzzy Methode. Veröffentlicht würde ihre Idee als Nachdruck erst 1989 im International Statistical Review.

2.1 Grundregel

In den folgenden Abschnitten zur Erklärung der Nearest Neighbor sind die Ausführungen von Fix and Hodges Jr (1989), Cover and Hart (1967) und Gertheiss and Tutz (2009b) Grundlage.
Das Prinzip der Nearest Neigbor Regel basiert auf der Berechnung der Distanzen zwischen einer neuen Beobachtung x und Merkmalsvektoren mit bekannten Klassenzugehörigkeiten $g = 1, ..., G$.
Im nächsten Schritt wird x derjenigen Klasse zugeordnet, die den nächsten Merkmalsvektor enthält.

Gegeben sei eine Lernstichprobe (Trainingsdaten)

$$L = (x_i, y_i), i = 1, ..., n$$

mit Merkmalsvektor x_i und Klassenzugehörigkeit y_i sowie eine neue Beobachtung x (Testdaten)
Gesucht wird die Klassenzugehörigkeit $\hat{y}$ der neuen Beobachtung

2.2 Einfache Nearest Neighbor Klassifikation (1NN)

Der einfachste Fall der Nearest Neigbor Klassifikation ist die 1NN-Regel, hierbei wird der erste nächste Nachbar des Merkmalvektors gesucht. Die Be-

stimmung des nächsten Nachbarn erfolgt über die Distanz

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

mit d ist ein Distanzmaß.

Allgemein ist das Distanzmaß z.B. durch die Minkowski-Distanz definiert:

$$d(x_i, x_j) = (\sum_{s=1}^{p} |x_{is} - x_{js}|^q)^{\frac{1}{q}} \quad \text{mit} \quad x_i, x_j \in R^p$$

Spezialfälle dieser Form sind die Absolut-Distanz mit $q = 1$, auch Manhattan-Distanz genannt

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

oder die euklidische Distanz mit $q = 2$

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

Die unterschiedliche Herangehensweise der beiden Distanzen ist in Abbildung 1 dargestellt. Die Strecke der Absolut-Distanz läuft über die x- und

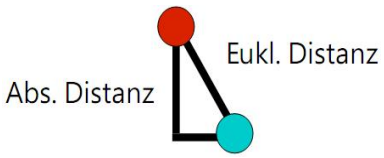

Abbildung 1: Wege der Distanzmaße

y-Achse. Die euklidische Distanz hingegen nimmt den direkten Weg zwischen den beiden Punkten. Meistens wird die euklidische Distanz verwendet, da diese häufig die besten Ergebnisse erzielt.

Als Beispiel für die 1NN-Regel seien zwei Klassen G = 2 (hellblau und dunkelblau) mit mehreren Merkmalsvektoren sowie einer neuen Beobachtung (rot), siehe Abbildung 2, gegeben. In welche Klasse wird diese neue Merkmalsausprägung klassifiziert? In diesem Beispiel stammt der nächste Nachbar aus

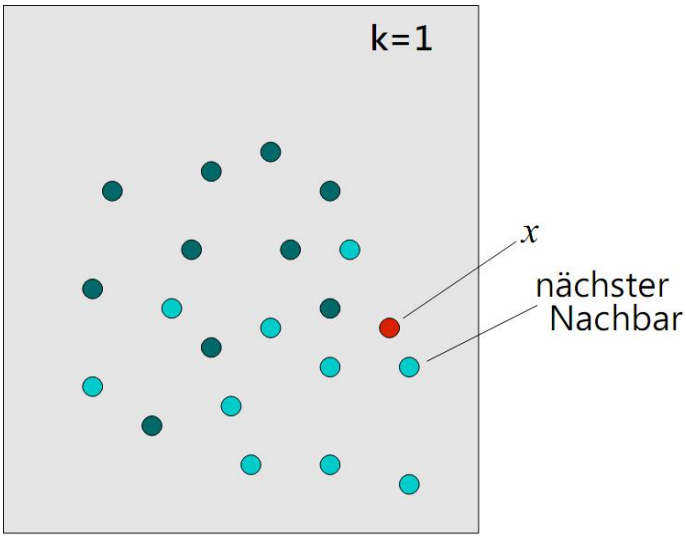

Abbildung 2: 1-NN Fall

der hellblauen Klasse. Die neue Beobachtung wird daher dieser zugeordnet.

Wird nur ein nächster Nachbar betrachtet, kann es leicht zu Fehlern in der Klassifikation kommen. Bei dem nächsten Nachbarn könnte es sich z.B. um einen Ausreißer handeln, und somit würde die falsche Klasse ausgewählt werden. Um diesen Fehler möglichst gering zu halten, werden mehrere Nachbarn miteinbezogen.

2.3 k Nearest Neigbor Klassifikation (kNN)

Bei der kNN-Regel wird jetzt nicht mehr nur der erste Nachbar ermittelt, sondern die k nächsten Nachbarn werden gesucht.

2.3.1 Prinzip

Die Bestimmung der k nächsten Nachbarn erfolgt ebenfalls über das Distanzmaß wobei $d(x, x_{(1)}) \leq d(x, x_{(2)}) \leq ... \leq d(x, x_{(k)})$. x wird dann derjenigen Klasse zugeordnet, die unter den k nächsten Nachbarn am häufigsten vorkommt. Haben mehrere Klassen dieselbe Häufigkeit wird eine dieser Klassen zufällig ausgewählt.
Betrachtet wird wieder das Beispiel aus dem vorherigen Fall. Jetzt werden jedoch die drei nächsten Nachbarn mit einbezogen (Abbildung 3). Die neue Be-

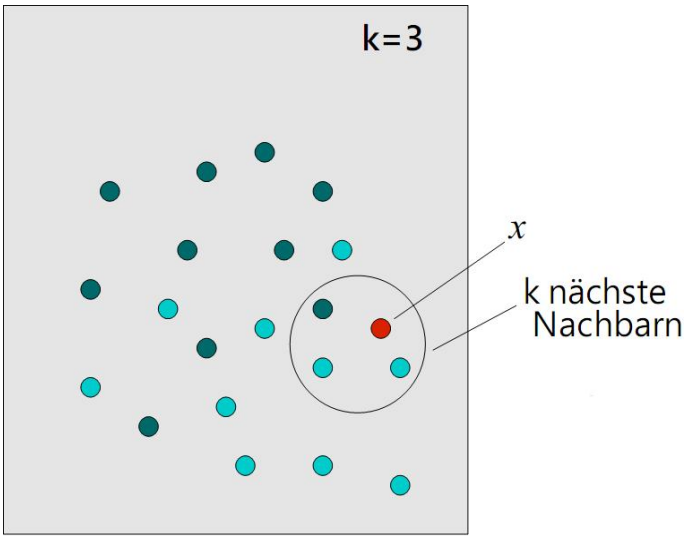

Abbildung 3: 3-NN Fall

obachtung wird auch hier der hellblauen Klasse zugeordnet, da zwei nächste Nachbarn aus der hellblauen Klasse stammen und nur einer aus der Dunkelblauen.

Im Fall der kNN-Regel kann für die Auswahl einer Klasse die geschätzte

Klassenwahrscheinlichkeit berechnet werden. Diese ist definiert als

$$\hat{P}(y = g|x) = \frac{1}{k} \sum_{i=1}^{k} I(y_{(i)} = g)$$

mit $y_{(i)}$ entspricht der Klassenzugehörigkeit des i-ten Nachbarn, d.h. es ergibt sich die relative Häufigkeit der Klasse g unter den k nächsten Nachbarn.
Für den Fall $k = 7$ und $g \in \{1, 2, 3\}$ (s. Abbildung 4) ergeben sich die folgenden jeweiligen geschätzten Klassenwahrscheinlichkeiten: Für Klasse 1

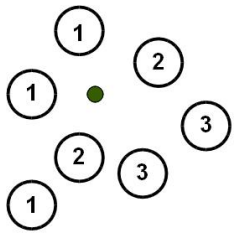

Abbildung 4: k nächste Nachbarn

beträgt die Wahrscheinlichkeit 3/7, für Klasse 2 und 3 jeweils 2/7.

2.3.2 Auswahl von k

Bei der kNN-Regel spielt die Auswahl von k eine entscheidende Rolle. Soll man k klein oder groß wählen? Für welches k erzielt man das beste Ergebnis?

Abbildung 5 und 6 zeigen ein Klassifikationsproblem aus Hastie et al. (2009). Hierbei sollen für drei Klassen eine geeignete Zuordnung gefunden werden. Setzt man k klein, erhält man eine instabile Klassifikationsentscheidung. Für

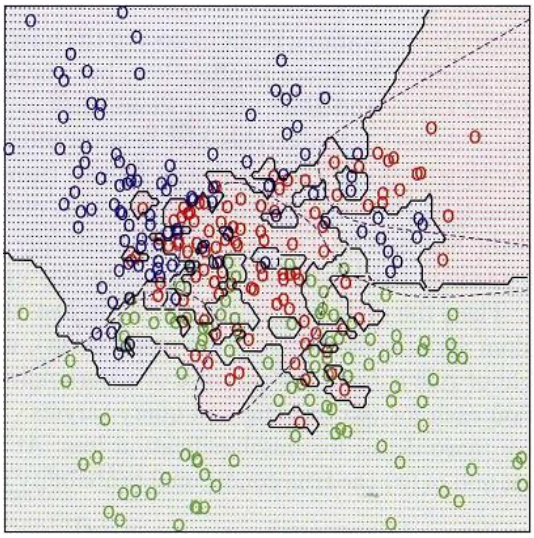

Abbildung 5: 1NN-Fall [Hastie et al. (2009)]

$k = 1$ (s. Abbildung 5) sind keine klaren Klassentrennungen zu erkennen. Man erhält einen geringen Bias, aber eine große Varianz. Ausreißer oder kleine Veränderungen der Lage eines Merkmalsvektor können leicht zu einer anderen Klassenzuordnung führen.

Erhöht man k erhält man eine stabilere Entscheidung. Es ist jedoch nicht immer sinnvoll weit entfernte Punkte mit einzubeziehen. Bei der Wahl $k = 15$

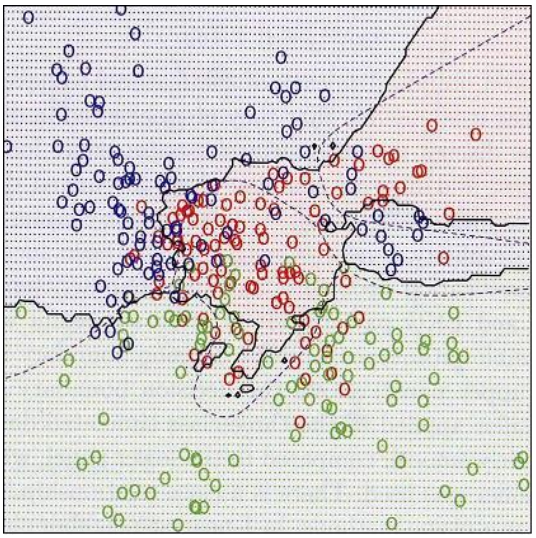

Abbildung 6: 15NN-Fall [Hastie et al. (2009)]

(Abbildung 6) „treten vermehrt Falschklassifikationen der Trainingsdaten auf, dafür ist die Klasseneinteilung übersichtlicher“. [Zierau (2007)]
Da k von den Daten abhängig ist, kann kein einfaches Kriterium angewendet werden um die optimale Größe der nächsten Nachbarn zu bestimmen.
Die einfachste Methode das „richtige“ k zu finden, ist es verschiedene auszuprobieren, die Fehlerraten zu vergleichen und das k mit der geringsten Fehlerrate zu verwenden.
Eine Methode um Fehlerraten zu berechnen und zu vergleichen, ist die leave-one-out Methode. Hierbei wird die Lernstichprobe in n-1 Trainingsdaten und einen Testdatenpunkt aufgeteilt. Nun wird für diesen Testdatenpunkt die kNN-Regel, bei festgelegtem k, durchgeführt.
Diesen Schritt wiederholt man für alle Merkmalsvektoren der Lernstichprobe, d.h. jeder Merkmalsvektor war einmal Testdatenpunkt und wurde mittels knn klassifiziert.
Da die wahren Klassenzugehörigkeiten bekannt sind, können für die verschie-

denen Testdatenpunkte die empirischen Fehlerraten, also

$$\epsilon(knn) = \frac{\text{Fehler}}{\text{Alle}}$$

berechnet werden.
Dieser Durchlauf wird für unterschiedliche k durchgeführt. Über die durchschnittliche Fehlerrate

$$\epsilon(\gamma) = \frac{1}{n} \sum_i I(y_i \neq \hat{y}_i)$$

können die Ergebnisse der verschiedenen k verglichen werden. Das k mit der niedrigsten Fehlerrate wird dann ausgewählt und benutzt.

2.4 Theoretische Fehlerrate

Im folgenden Abschnitt wird nicht mehr von der empirische Fehlerrate ausgegangen, sondern es wird die theoretische Fehlerrate betrachtet. [vgl. Cover and Hart (1967)]

2.4.1 Definition und Grenzen

Ausgehend von $x_{(n)}$ sei der nächste Nachbar von x mit Klassenzugehörigkeit $y_{(n)}$, wobei n den Stichprobenumfang bezeichnet. Sowie y ist die tatsächliche Klasse von x. Hierbei unterliegt die Klassifikation einer Verlustfunktion $L(y, \hat{y}_{(n)})$.

Seien nun die Tupel (x_i, y_i) Zufallsvariablen, dann definiert sich die theoretische Fehlerrate als

$$\epsilon(\gamma) = E(L(y, \hat{y}_{(n)}))$$

mit

$$L = \begin{cases} 1 & \text{für} \quad y \neq \hat{y}_{(n)} \\ 0 & \text{für} \quad y = \hat{y}_{(n)} \end{cases}$$

d.h. sie resultiert aus dem Erwartungswert der Verlustfunktion. Geht $n \to \infty$, so ergibt sich die asymptotische Fehlerrate $\lim_{n\to\infty} \epsilon(\gamma)$.

Diese asymptotische Fehlerrate der 1NN-Regel ist suboptimal. Es ist daher sinnvoll die Grenzen für die Abweichung von der optimalen Fehlerrate zu betrachten. Bei der optimalen Fehlerrate handelt es sich um die Bayes-Fehlerrate ϵ_{opt}. Wie in der Einführung erwähnt, basiert die Bayes-Klassifikation darauf x der Klasse zuzuordnen für die $P(y|x)$ maximal wird.
Für die Abweichung der beiden Fehlerraten gelten folgende Grenzen:

$$\epsilon_{opt} \leq \epsilon_{asy}(\gamma_1) \leq \epsilon_{opt}(2 - \frac{G}{G-1}\epsilon_{opt}) \leq 2\epsilon_{opt}$$

mit G = Anzahl der Klassen und $\epsilon_{asy}(\gamma_1)$ = Fehlerrate für den 1nn-Fall.
Die erste Ungleichung besagt, dass die Bayes-Regel besser ist als die asymptotische Fehlerrate für 1NN. Die Grenze nach oben wird in der zweiten Ungleichung in Abhängigkeit zur Klassenanzahl angegeben. Für den zwei Klassenfall, also $G = 2$, erhält man $\epsilon_{opt}(2-2\epsilon_{opt})$; für $G = 5$ folgt $\epsilon_{opt}(2-5/4\epsilon_{opt})$. Man kann erkennen, dass die Grenze nach oben hin beschränkt ist. Diese Beschränkung wird in der letzte Ungleichung gezeigt und besagt, dass die asymptotische Fehlerrate höchstens doppelt so hoch ist wie die Bayes-Fehlerrate.

2.4.2 Beeinflussung durch k

Nicht nur die Klassenanzahl, sondern auch die Größe von k, also die betrachteten nächsten Nachbarn, hat einen Einfluss auf die theoretische Fehlerrate. Im zwei Klassen-Fall ist die Fehlerrate für gerade und ungerade Anzahl von k identisch, d.h.

$$\epsilon_{asy}(\gamma_{2k}) = \epsilon_{asy}(\gamma_{2k-1})$$

Mit einer Erhöhung von k verbessert sich die Fehlerrate bei festen k so, dass folgt

$$\epsilon_{asy}(\gamma_k) \leq ... \leq \epsilon_{asy}(\gamma_1)$$

k darf asymptotisch jedoch nicht zu groß gewählt werden, da sonst die Information in der Stichprobe verloren geht.

Bei geeigneter Wahl von k wird der Abstand der beiden Fehlerrate geringer, d.h. die asymptotische nähert sich der optimalen Fehlerrate.
Für $k \to \infty$ wobei $\frac{k}{n} \to 0$, also n wächst schneller als k, gilt sogar:(schwache Konvergenz)

$$\epsilon_{asy}(\gamma_k) \to \epsilon_{opt}$$

Man muss hierbei aber im Hinterkopf behalten, dass in der Praxis n meistens nicht gegen unendlich geht und k somit auch nicht unendlich groß gewählt werden kann. [vgl. Fahrmeir et al. (1996)]

Für Vervollständigung wird im folgenden Kapitel die Regression durch Nearest Neighbor betrachtet.

2.5 Regression

Die Nearest Neighbor Regression stellt eine Erweiterung des Konzeptes gleitender Durchschnitte dar, die in Fahrmeir et al. (2007) genauer erklärt werden.
Hierbei liegt die Idee zugrunde, „als Schätzer $\hat{f}(x_i)$ an der Stelle x_i, $i = 1, ..., n$, den „Mittelwert“ der Responsebeobachtungen in einer Nachbarschaft von x_i zu verwenden“. [Sprung (2004)]
Allgemein ist der nächste Nachbar Schätzer definiert durch

$$\hat{f}(x_i) = MWO_{i \in N(x_i)} y_i$$

wobei MWO ein beliebiger Mittelwertsoperator ist und $N(x_i)$ eine Nachbarschaft um x_i bezeichnet. Der Schätzer ist somit durch den Mittelwertsoperator, als auch durch die Nachbarschaft definiert, die im folgenden genauer erklärt werden. [vgl. Fahrmeir et al. (2007) und Sprung (2004)]

2.5.1 Mittelwertsoperatoren

Häufig verwendete Mittelwertsoperatoren sind

- Arithmetisches Mittel (running mean)

Der Schätzer wird über das arithmetische Mittel der Beobachtungen in der Nachbarschaft bestimmt, d.h.

$$\hat{f}(x_i) = \frac{1}{|N(x_i)|} \sum_{j \in N(x_i)} y_i$$

- Median (running median)

  Der Median der Ausprägungen in der Nachbarschaft wird verwendet und der Schätzer ergibt sich durch

  $$\hat{f}(x_i) = \text{median}\{y_i, i \in N(x_i)\}$$

- lineare Einfachregression (running line)

  Basierend auf den Nachbarn von x_i wird eine lineare Einfachregression geschätzt. Die Vorhersage durch diese Modell wird als Schätzer verwendet, man erhält

  $$\hat{f}(x_i) = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

  mit den aus der Nachbarschaft bestimmten KQ-Schätzern $\hat{\beta}_0$ und $\hat{\beta}_1$

2.5.2 Definition Nachbarschaft

Die Annahme erfolgt über eine „Nachbarschaft bestehend aus den k nächsten Nachbarn“. [Fahrmeir et al. (2007)] Diese führt zu einer unsymmetrische Nachbarschaft und ist definiert als

$$N(x_i) = \{i : d_i \in \{d_{(1)}, ..., d_{(k)}\}\}$$

mit $d_{(1)}, ..., d_{(k)}$ bezeichnet die geordneten Distanzen $d_i = |x_i - x|$.
Hier ist k als Glättungsparameter anzusehen. Für kleine k erfolgt eine raue Schätzung und für ein großes, nah am Stichprobenumfang liegendes k wird eine glatte Schätzung hervorgerufen.

3 Nearest Neighbor Ensembles

Die Einführung in das Prinzip der Ensemble Techniken erfolgt auf Grundlage von Domeniconi and Yan (2004) und Bay (1999).
Um Klassifikationsverfahren zu verbessern existieren Ensemble Techniken, die dafür zuständig sind die Genauigkeit des Ganzen zu verbessern.
Ensemble Methoden sind eine Verbindung von einzelnen Klassifikatoren, die verschiedenartig sind, um sich gegenseitig zu unterstützen. Die Verschiedenartigkeit ist wichtig, da diese sicherstellt, dass die Klassifikatoren unkorrelierte Fehler machen.

Die hier betrachtete Technik der Nearest Neighbor Ensembles, vorgestellt von Gertheiss and Tutz (2009a), hat nicht nur den Hintergrund eine Klassifikation über die kNN-Regel zu berechnen, sondern es werden zusätzlich eine Variablenselektion, sowie die Verwendung von Interaktionen durchgeführt.

3.1 Konzept

Angefangen mit dem einfachsten Fall bei dem der Klassifikator nur auf einer einzelnen Variable basiert. Demzufolge sei $\hat{\pi}_{ig(j)}$ die geschätzte Wahrscheinlichkeit, dass die Beobachtung i in die g-te Kategorie fällt, falls das Distanzmaß nur auf dem Prädiktor x_j basiert.
Der endgültige Schätzer $\hat{\pi}_{ig}$ wird durch das Ensemble konstruiert und es folgt

$$\hat{\pi}_{ig} = \sum_{j=1}^{p} c_j \hat{\pi}_{ig(j)} \text{ mit } c_j \geq 0 \; \forall j \text{ und } \sum_j c_j = 1$$

Die Gewichte c_j werden absichtlich benutzt um im Verlauf die einflussreichen Variablen herauszufiltern. Bisher sind diese Gewichte unbekannt und müssen noch bestimmt werden.
Es ist zu klären, ob die Gewichte von der Klasse abhängig sind, d.h.

$$\hat{\pi}_{ig} = \sum_{j=1}^{p} c_{gj} \hat{\pi}_{ig(j)} \text{ mit } c_{gj} \geq 0 \; \forall g, j \text{ und } \sum_j c_{gj} = 1 \; \forall g$$

Für die Schätzer $\hat{\pi}_{ig(j)}$ muss unter der Bedingung $\hat{\pi}_{ig(j)} \geq 0 \; \forall g, j$ und $\sum_g \hat{\pi}_{ig(j)} = 1 \; \forall j$ gelten, dass $\hat{\pi}_{ig} \geq 0 \; \forall g$ und $\sum_g \hat{\pi}_{ig} = 1$. Dies ist nur der Fall, falls $c_{1j} = ... = c_{Gj} = c_j$. Solange nicht bekannt ist, in welche Klasse die neue Beobachtung zugeordnet wird, ist der Schätzer $\hat{\pi}_{ig(j)}$ unbekannt. Folglich muss garantiert werden, dass sich der Schätzer $\hat{\pi}_{ig}$ zu eins summiert, egal welcher Schätzer $\hat{\pi}_{ig(j)}$ vorliegt. Der Beweis dazu ist im Appendix von Gertheiss and Tutz (2009a) zu finden.

3.2 Ermittlung der Gewichte

Zur Berechnung der oben erwähnen Gewichte wird eine Verlustfunktion $L(y, \hat{\Pi})$ definiert. Diese besagt wie gut die wahren Klassen $y = (y_1, ..., y_n)^T$ durch die geschätzten Auswahlwahrscheinlichkeiten $(\hat{\Pi})_{ig} = \hat{\pi}_{ig}, \; i = 1, ..., n$ und $g = 1, ..., G$ gefittet werden.
Da $\hat{\pi}_{ig}$ nach Definition von c bestimmt wird, kann die Matrix $\hat{\Pi}$ als eine Funktion von $c = (c_1, ..., c_p)^T$ angesehen werden. Die Gewichte erhält man, indem die Verlustfunktion für gegebenen Trainingsdaten über alle c minimiert wird. Für den Fall einer moderaten Anzahl an Prädiktoren und einer großen Stichprobe, kann es bei der einfachen Verlustfunktion passieren, dass eine Beobachtung i keinen Nachbarn aus der gleichen Klasse hat. Die Auswahlwahrscheinlichkeit für die richtig Klasse y_i wäre somit 0. Um diese Hypersensivität zu umgehen wird eine quadratische Verlustfunktion, auch „Brier Score“ [Gertheiss and Tutz (2009a)] genannt, verwendet. Dieser ist definiert als

$$Q(y, \Pi) = \sum_i \sum_g (z_{ig} - \pi_{ig})^2$$

mit

$$z_{ig} = \begin{cases} 1 & \text{für } y_i = g \\ 0 & \text{für sonst.} \end{cases}$$

z_{ig} sind hierbei Indikator-Variablen des Vektors $z_i = (z_{i1}, ..., z_{iG})^T$, der den Response der Klassen $y_i \in 1, ..., G$ darstellt.
Die Funktion $Q(y, \Pi)$ bestraft die Abweichung der Auswahlwahrscheinlichkeit von der tatsächlichen Auswahl und besagt je größer der Brier Score umso

schlechter ist die Vorhersage.

Zur Implementierung wird für jede Trainingsbeobachtung i eine Matrix P_i erstellt so, dass gilt:

$$(P_i)_{gj} = \hat{\pi}_{ig(j)}$$

Diese einzelnen Matrizen werden in einer großen Matrix $P = (P_1^T ... \ P_n^T)^T$ zusammengefasst. Dasselbe passiert mit dem Dummy-Vektor z_i. Man erhält $z = (z_1^T, ..., z_n^T)$.
Der Brier-Score kann jetzt als eine Funktion von c in Matrixschreibweise geschrieben werden

$$Q(c) = (z - Pc)^T(z - Pc)$$

Man erhält ein quadratisches Optimierungsproblem mit der Beschränkung $c_j \geq 0 \ \forall j$ und $\sum_j c_j = 1$ Die Lösung dieses Problems wird später im praktischen Beispiel genauer behandelt.

3.3 Variablenselektion

Ein weiterer Baustein des Nearest Neighbor Ensembles ist die Variablenselektion. Ihr Grundprinzip besteht darin, eine Variable x_j auszuwählen, falls das Gewicht $c_j \neq 0$ ist. Eine Möglichkeit dafür ist das sogenannte Thresholding, ein Schwellenwertverfahren. Um dieses Verfahren anwenden zu können, müssen die Gewichte der einzelnen Variablen zuvor bestimmt worden sein.

3.3.1 Threshold

Beim Thresholding kann man in die Varianten Hard- und Soft-Thresholding unterscheiden.Bei der Selektion mit Hard-Thresholding werden alle Variablen mit einem Gewicht das kleiner ist als eine bestimmte Schwelle t auf Null gesetzt. Für $t = 0.25 \max_j(c_j)$ ergibt sich beispielsweise:
Koeffizienten, die kleiner sind als $0.25 \max_j(c_j)$, d.h. die zugehörige Einflussgröße erreicht nicht 25 % des Gewichtes des „wichtigsten“ Prädiktors, werden auf Null gesetzt. Da die Bedingung $\sum_j c_j = 1$ weiter erfüllt sein muss, erfolgt nach dem Nullsetzen eine erneute Berechnung der Gewichte der selektierten

Variablen.
Als zweite Möglichkeit kann Soft-Thresholding verwendet werden. Hierbei werden die Gewichte durch $c_j = (\widetilde{c_j} - t)^+$ ersetzt, wobei $\widetilde{c_j}$ dem zuvor bestimmten Gewichten entspricht.

Die Schwelle t wird bei beiden Methoden als Tuningparameter angesehen. t soll datenbezogen bestimmt werden, häufig wird dafür die K-fache Kreuzvalidierung benutzt.
Die K-fache Kreuzvalidierung ist die verallgemeinerte Form der in Abschnitt 2.3.2 beschrieben leave-one-out Methode. Es wird hierbei n=K gesetzt, mit K entspricht K gleichgroßen Teildatensätzen.

Die Threshold Methoden stehen unter der Bedingung $\sum_j c_j = 1$, d.h. die Summe der Gewichte muss eins ergeben. Wird diese Voraussetzung nun gelockert auf $\sum_j |\widetilde{c_j}| \leq s$ erhält man ein Problem vom Typ Lasso.

3.3.2 Lasso

Lasso zielt darauf ab, die Koeffizienten zu schrumpfen und außerdem eine Variablenselektion vorzunehmen. „Auf diesem Weg soll gute Prognosegenauigkeit und Interpretierbarkeit erreicht werden.“ [Stuckart (2009)] Die Methode basiert darauf, ein lineares Modell durch Minimierung der Residuenquadratsumme an die Daten anzupassen. [vgl.Hornung (2005)]
Hierfür wird ein Strafterm auf $\widetilde{c_j}$ gelegt und zusätzlich wird das Bestrafungskriterium

$$Q_p(\widetilde{c}) = (z - P\widetilde{c})^T(z - P\widetilde{c}) + \lambda \sum_j |\widetilde{c_j}|$$

über alle c minimiert. Die Schwelle s aus der Bedingung $\sum_j |\widetilde{c_j}| \leq s$ kann als Tuningparameter angesehen werden. Er bestimmt die Stärke der Schrumpfung, genauer „je kleiner [s] umso stärker die Schrumpfung“. [Stuckart (2009)]

3.4 Interaktionen

Eine weitere Methode die durch die Nearest Neighbor Ensembles betrachtet wird, ist die Verwendung der Interaktionen. Dies ist nur im Rahmen geringer Dimensionen möglich. Grund dafür ist der starke Anstieg der Prädiktoren, so beträgt die Anzahl der Einflussgrößen mit Interaktionen $P = p + \binom{p}{2} + \dots$. Wird für die Daten zuerst die Variablenselektion angewendet, bleibt für gewöhnlich eine adäquate Anzahl von Variablen zurück, für die dann im zweiten Schritt Interaktionen mit betrachtet werden können.
Der Schätzer basiert nun nicht mehr nur auf einem sondern auf zwei Prädiktoren

$$\hat{\pi}_{ig} = \sum_{j} \sum_{s} c_{js} \hat{\pi}_{ig(js)}$$

Um die Effizienz der Nearest Neighbor Ensembles zu überprüfen, werden diese berechnet und mit dem einfachen Nearest Neighbor Klassifikator verglichen. Für den Vergleich wird die Prognosegenauigkeit, d.h. der Anteil richtig klassifizierter Objekte durch Anzahl insgesamt bzw.. 1-empirische Fehlerrate, verwendet.

4 Simulationsstudien

Zuerst erfolgt eine Überprüfung des Verfahrens über in R simulierte Datensätze. Diese bestehen aus einer Datenmatrix X mit den Merkmalsausprägungen und einem Vektor y mit den zugehörigen Klassen.

4.1 Datenerzeugung und betrachtete Methoden

4.1.1 Allgemeines

Für die Erzeugung der Trainings- und Testdaten wird die Funktion `mvrnorm` aus dem Paket `MASS` verwendet und generiert eine multivariate Normalverteilung. D.h. die Merkmale X sind multivariat normalverteilt.
Dabei liegt folgende Funktion zugrunde:

```
simData <- function(n, nclasses, mu, Sigma)
  {
    y <- NULL
    X <- NULL
    for (i in 1:nclasses)
      {
          yi <- c(rep(i,n))
          Xi <- mvrnorm(n=n, mu[[i]], Sigma[[i]])
          y <- c(y,yi)
          X <- rbind(X,Xi)
      }
      return(list("X"=X,"y"=y))
    }
```

Diese gibt die Klassenzugehörigkeiten `y` an, sowie die Merkmalsvektoren `X`. Für die Anwendung muss `n`= Anzahl der Beobachtungen, `nclasses` = Anzahl der Klassen, `mu` = Erwartungswert (μ) und `Sigma` = Kovarianzmatrix (Σ) angegeben werden. Für den Fall höherdimensionaler Daten werden insgesamt 100 Variablen für jeweils drei Klassen erzeugt. Die Größe der Trainingsdaten beträgt 200 Beobachtungen, die der Testdaten 500.

Für die Funktion zur Erzeugung der Daten ergibt sich somit `n` = 200 bzw. `n` = 500 und `nclasses` = 3. Man erhält für die Trainingsdaten eine Matrix mit 100 Variablen und 600 Beobachtungen, für die Testdaten eine Matrix mit 100 Prädiktoren und 1500 Beobachtungen.

4.1.2 Betrachtete Szenarien

Für μ und Σ werden verschiedene Annahmen getroffen.

Erzeugung des Erwartungswertes
Für μ werden zwei unterschiedliche Fälle angenommen.

Fall 1:
Die Erwartungswerte der drei Klassen sind unterschiedlich und sind gegeben als $\mu_1^T = (0, ..., 0), \mu_2^T = (2, ..., 2)$ und $\mu_3^T = (1, ..., 1)$ mit μ_1 Erwartungswert der ersten Klasse, μ_2 Erwartungswert der zweiten Klasse und μ_3 Erwartungswert der dritten Klasse.

Fall 2:
In den drei Klassen werden unterschiedliche Erwartungswerte angenommen, diese sind jetzt zusätzlich innerhalb der Klassen für einzelne Komponenten verschieden. So ergibt sich für $\mu_1^{*T} = (0, ..., 0)$, $\mu_2^{*T} = (2, ..., 2, 0, ..., 0)$ und $\mu_3^{*T} = (1, ..., 1, 0, ..., 0)$.

Erzeugung der Kovarianzmatrizen
Es werden insgesamt fünf verschiedene Kovarianzmatrizen angenommen. Diese haben jeweils folgende Korrelationsmatrizen zugrunde liegen:

- Autokorrelationsmatrix
- Block-Autokorrelationsmatrix
- Equikorrelationsmatrix
- Block-Equikorrelationsmatrix

- Block-Netzwerkmatrix

Die ***Autokorrelationsmatrix*** ist definiert als

$$\Sigma_A = \begin{pmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{p-1} \\ \rho & 1 & \rho & \cdots & \vdots \\ \rho^2 & \rho & 1 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \rho \\ \rho^{p-1} & \cdots & \cdots & \rho & 1 \end{pmatrix}$$

Hier sind die Merkmale voneinander abhängig. Je weiter sie in der Matrix voneinander entfernt sind, umso kleiner wird ihre Korrelation, genauer die Potenz erhöht sich bei jeder Stelle um eins.
Außerdem wird die zugehörige Block-Autokorrelationsmatrix betrachtet. Ihr Grundprinzip ist so aufgebaut, dass die Autokorrelationsmatrix blockhaft in eine große Nuller Matrix gesteckt wird. Daraus ergibt sich die Form:

$$\Sigma_{BA} = \begin{pmatrix} \Sigma_A & 0 & 0 \\ 0 & \Sigma_A & 0 \\ 0 & 0 & \Sigma_A \end{pmatrix}$$

Die ***Equikorrelationsmatrix*** ist strukturiert als

$$\Sigma_E = \begin{pmatrix} 1 & \rho & \rho & \cdots & \rho \\ \rho & 1 & \rho & \cdots & \vdots \\ \rho & \rho & 1 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \rho \\ \rho & \cdots & \cdots & \rho & 1 \end{pmatrix}$$

Die Korrelation ρ ist dabei an jeder Stelle identisch, d.h. jedes Merkmal korreliert gleich zu den anderen. Auch hier wird außerdem die Block-Equikorrelationsmatrix verwendet, aufgebaut nach demselben Muster wie der Block der Autokorre-

lationsmatrix und es ergibt sich

$$\Sigma_{BE} = \begin{pmatrix} \Sigma_E & 0 & 0 \\ 0 & \Sigma_E & 0 \\ 0 & 0 & \Sigma_E \end{pmatrix}$$

Als letztes wird die ***Netzwerkmatrix*** angewendet. Diese wird nur in der Form eines Blocks benutzt, d.h.

$$\Sigma_{BN} = \begin{pmatrix} \Sigma_N & 0 & 0 \\ 0 & \Sigma_N & 0 \\ 0 & 0 & \Sigma_N \end{pmatrix}$$

Die Netzwerkmatrix Σ_N selbst ist eine symmetrische Matrix und basiert auf der Korrelation zwischen Merkmalen. Sind diese miteinander verbunden, d.h. sie sind bedingt abhängig, gegeben den Rest, so ist deren Korrelation ungleich Null. Abbildung 7 zeigt ein Beispiel eines Netzwerkes. Es werden hierbei 10 Prädiktoren betrachtet. 1 und 2 sind beispielsweise direkt miteinander verbunden, wohin gegen 1 und 3 nur über die Variable 6 vernetzt ist. Die

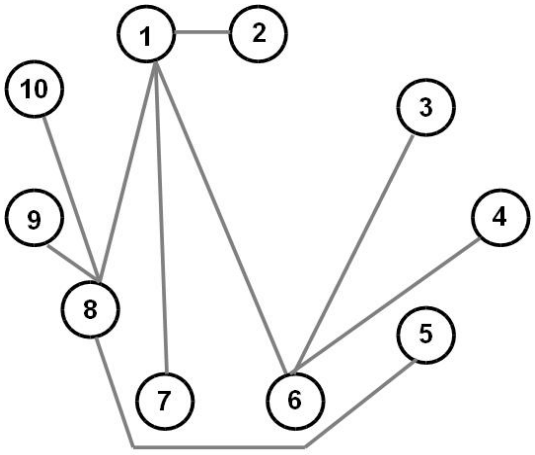

Abbildung 7: Darstellung eines Netzwerks

Elemente d_{ij} der Netzwerkmatrix [vgl. Whittaker (2009)] berechnen sich dann über

$$d_{ij} = -corr(x_i, x_j|rest)$$

Für die Kovarianzmatrix heißt das, bei direkter Vernetzung ist d_{ij} größer, als wenn eine andere Variable als Verbindung vorliegt.

Die Kovarianzmatrix selbst berechnet sich über die inversen Elemente der Choleskyzerlegung der Korrelationsmatrix. Für die 100 zu erzeugenden Variablen wird eine 100×100 Korrelationsmatrix bzw. Kovarianzmatrix benötigt.

Zur genauen Bestimmung der Korrelationsmatrizen werden folgende Werte angenommen:

Für die Auto- und Equikorrelationsmatrix muss ρ angegeben werden. In der ersten Klasse wird $\rho_1 = 0.7$, in der zweiten Klasse $\rho_2 = 0.6$ und in der dritten Klasse $\rho_3 = 0.8$ festgesetzt. D.h. in den Klassen wird eine mittlere bis hohe Korrelation angenommen.
Für die Blockmatrizen der Auto- und Equikorrelation wird ein Block von 20 Variablen erzeugt. Daraus ergibt sich ausschnittsweise für die Kovarianzmatrix mit Autokorrelation

$$\Sigma_{auto} = \begin{pmatrix} 1.0000 & 0.7000 & 0.4900 & 0.3430 & 0.2401 & 0.1681 & 0.1177 & ... & ... \\ & 1.0000 & 0.7000 & 0.4900 & 0.3430 & 0.2401 & 0.1681 & ... & ... \\ & & 1.0000 & 0.7000 & 0.4900 & 0.3430 & 0.2401 & ... & ... \\ & & & 1.0000 & 0.7000 & 0.4900 & 0.3430 & ... & ... \\ & & & & ... & ... & ... & ... & ... \end{pmatrix}$$

Für die Kovarianzmatrix mit Equikorrelation

$$\Sigma_{equi} = \begin{pmatrix} 1.0 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & ... & ... \\ & 1.0 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & ... & ... \\ & & 1.0 & 0.7 & 0.7 & 0.7 & 0.7 & ... & ... \\ & & & 1.0 & 0.7 & 0.7 & 0.7 & ... & ... \\ & & & & ... & ... & ... & ... & ... \end{pmatrix}$$

Der Block der Netzwerkmatrix besteht nur aus 10 Variablen, daraus ergibt sich

$$\Sigma_{netz} = \begin{pmatrix} 20.3306 & 12.1983 & 3.3268 & 6.6536 & -1.7384 & 11.0894 & ... & ... \\ & 8.3190 & 1.9961 & 3.9922 & -1.0430 & 6.6536 & ... & ... \\ & & 1.70803 & 1.4161 & -0.2845 & 2.3601 & ... & ... \\ & & & 3.8321 & -0.5689 & 4.7202 & ... & ... \\ & & & & ... & ... & ... & ... \end{pmatrix}$$

Das hier verwendete Netzwerk wurde bereits in Abbildung 7 dargestellt.

Szenarien

Insgesamt ergeben sich dadurch 10 Szenarien und somit 10 unterschiedliche Datensätze:

- Szenario 1:
  Blockkorrelationsmatrix mit μ
- Szenario 2:
  Block-Autokorrelationsmatrix mit μ
- Szenario 3:
  Equikorrelationsmatrix mit μ
- Szenario 4:
  Block-Equikorrelationsmatrix mit μ
- Szenario 5:
  Block-Netzwerkkorrelationsmatrix mit μ
- Szenario 6:
  Autokorrelationsmatrix mit μ^*
- Szenario 7:
  Block-Autokorrelationsmatrix mit μ^*

- Szenario 8:
  Equikorrelationsmatrix mit μ^*

- Szenario 9:
  Block-Equikorrelationsmatrix mit μ^*

- Szenario 10:
  Block-Netzwerkkorrelationsmatrix mit μ^*

Für jedes Szenario werden 100 Datensätze erzeugt und für die im nächsten Abschnitt beschriebenen Methoden benutzt.

4.1.3 Betrachtete Methoden

Zur Berechnung des Klassifikationsverfahrens der einfachen Nearest Neighbors wird die Funktion `knn` aus dem Paket `class` verwendet. Sie gibt die geschätzten Klassenwahrscheinlichkeiten sowie die prognostizierten Klassenzugehörigenkeiten der Testdaten aus.
`knn` wird in der Simulation für drei Fälle benutzt: $k = 1$, $k = 17$ und $k = 31$, d.h. es wird der erste, die 17 und die 31 nächsten Nachbarn betrachtet. Die Anzahl der nächsten Nachbarn ist nicht willkürlich, sondern über die Funktion `kknn` bestimmt. Diese gibt das beste k, also das mit der geringsten Fehlerrate, aus. Basieren auf diesem k wird zusätzlich entweder ein kleineres oder ein größeres k angenommen.
Die Nearest Neighbor Ensembles werden über die Funktion `nnEnsemble` [s. Anhang] berechnet. Hierfür wird eine Verlustfunktion, sowie eine Methode zur Variablenselektion benötigt, außerdem muss die Schwelle t des Thresholdings festgesetzt werden.
Als Verlustfunktion wird der in 3.2 beschrieben Brier-Score angenommen. Für die Variablenselektion wird „ipop“ benutzt. Diese Methode löst quadratische Probleme und hat die Form $\min(c^T x * 1/2 x^T H x)$ unter der Bedingung $b \leq Ax \leq b + r$ und $l \leq x \leq u$. ipop minimiert die Verlustfunktion um darüber die Gewichte zu bestimmen und somit die relevanten Variablen zu

erhalten. Angewendet auf die Formel des Brier-Score ergibt sich

$$(y - Pc)^T(y - Pc) = \underbrace{y^T y}_{const.} \underbrace{-2y^T P}_{c} x + x^T \underbrace{P^T P}_{H} x \underset{x}{\rightarrow} \min$$

unter der Bedingung $c_j \leq 0$ und $|c| = 1$. Als Schwelle für die Gewichtsbestimmung wird ein Vektor $t = c(0.05, 0.1, 0.15, 0.2)$ angenommen.
Auch `nnEnsemble` gibt wie die Funktion `knn` die prognostizierten Klassenzugehörigkeiten des Testdatensatzes aus. Außerdem erhält man die geschätzten Klassenwahrscheinlichkeiten für den Trainings- und Testdatensatz sowie die Gewichte der Variablen.
Die Nearest Neigbor Ensembles werden zweimal berechnet. Die erste Variante verwendet einen Prädiktor $\hat{\pi}_{(j)}$ und keine Interaktionen, die zweite Variante benutzt die selektieren Variablen und betrachtet zudem die zweifachen Interaktionen. Beide werden für ein kleines k (k=11) und ein großes k (k=35) berechnet. Auch hier wurde k über `kknn` bestimmt.

4.2 Ergebnisse

Die Methoden werden jeweils für die 100 simulierten Datensätze verwendet. Man erhält somit 100× die Prognosegenauigkeit. Für einen ersten Überblick wird im weiteren der Mittelwert der Genauigkeiten des jeweiligen Szenarios benutzt.

Um die Prognosegenauigkeit der Nearest Neighbor Ensembles in Relation zu setzen, wird für die simulierten Daten zuerst die Bayes-, also die theoretisch optimale, Klassifikation berechnet. Wie bereits in der Einleitung erläutert liegt dabei eine multivariaten Normalverteilung zugrunde

$$\frac{1}{(2\pi)^{\frac{1}{p}}|\Sigma|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}$$

mit p = Anzahl Prädiktoren und $|\Sigma|$ ist die Determinante der Kovarianz. Nach Anwendung der Bayes-Regel ergeben sich folgende durchschnittliche Prognosegenauigkeiten (s. Tabelle 1). Insgesamt erreicht Bayes über alle Kor-

	Auto	Blockauto	Equi	Blockequi	Blocknetz
identischer Ewert	0.9935	0.9959	0.9212	0.9773	0.9999
unterschiedl. Ewert	0.9716	0.9631	0.9999	0.9173	0.8478

Tabelle 1: Prognosegenauigkeit Bayes-Klassifikation

relationsmatrizen eine durchschnittliche Genauigkeit von ca. 0.96 %. Im besten Fall werden 99 %, im schlechtesten Fall 85 % Objekte richtig klassifiziert. Da die Verteilung der Szenarien bekannt ist, erzielt die Bayes-Klassifikation eine sehr gutes Ergebnis. Dies kann in der Praxis jedoch deutlich schlechter ausfallen, da dort die Verteilung geschätzt werden muss.

Die folgenden Tabellen zeigen die Prognosegenauigkeiten der einfachen Nearest Neighbor und der Nearest Neighbor Ensembles. 1nn, 17nn und 31nn stehen für die einfache NN-Regel mit $k = 1$, $k = 17$ und $k = 31$. kmin, kmin2, kmax und kmax2 bezeichnen die Ensemble Technik, wobei kmin und kmax nur einen Prädiktor und keine Interaktionen zur Klassifikation ver-

wenden. kmin2 und kmax2 benutzen die aus kmin und kmax ausgewählten Variablen, für die nun zusätzlich die Interaktionen miteinbezogen werden.

Für gleichen Erwartungswert innerhalb einer Klasse (Tabelle 2) werden durchschnittlich für die Nearest Neighbor 77,46 % und für die Nearest Neighbor Ensembles 77,43 % der Objekte richtig prognostiziert. Die Ensemble Methode

	Auto	Blockauto	Equi	Blockequi	Blocknetz
1nn	0.9014	0.9313	0.3937	0.7347	0.6595
17nn	0.9594	0.9736	0.4443	0.8398	0.7503
31nn	0.9635	0.9758	0.4843	0.8521	0.7584
kmin	0.7986	0.8056	0.6247	0.7585	0.7365
kmin2	0.8614	0.8741	0.6388	0.8061	0.8059
kmax	0.8118	0.8167	0.6292	0.7669	0.7511
kmax2	0.8672	0.8768	0.6371	0.8088	0.8099

Tabelle 2: Prognosegenauigkeit identische Erwartungswerte

ist nur unter Equi- und Blocknetzkorrelation besser als der einfache Nearest Neighbor. Wobei das Ergebnis für Equikorrelation insgesamt im Vergleich zu den anderen Korrelationsmatrizen gering ist. So schätzt die NN-Regel nur eine Prognosegenauigkeit von unter 50 %. Unter Auto- und Blockautokorrealtion erzielt der einfache Nearest Neighbor hingegen ein sehr gutes Ergebnis, die Genauigkeit beträgt bis zu 97 %. Das Ensemble gibt für die Autokorrelation maximal 86 % richtig prognostizierte Objekte an. Es zeigt sich, dass die zweite Variante, d.h. die Verwendung der selektieren Variablen unter Einbeziehung der Interaktionen, immer besser ist als die Erste. Ob nun ein kleines k oder ein großen k bei der Ensemble Technik verwendet wird, zeigt keinen Unterschied. Im Gegensatz zum einfachen Nearest Neighbor, für den die Anzahl miteinbezogener Nachbarn verschiedene Ergebnisse liefern. So steigt die Prognosegenauigkeit für mehrere betrachtete Nachbarn, und erreicht für 31nn unter allen Korrelationsmatrizen das beste Ergebnis der NN-Regel.

Tabelle 3 zeigt die Prognosegenauigkeit für den Fall, dass zwischen und innerhalb einer Klassen unterschiedliche Erwartungswerte zugrunde liegen. Die durchschnittliche Gesamtgenauigkeit aller Methoden des einfachen Nea-

	Auto	Blockauto	Equi	Blockequi	Blocknetz
1nn	0.5674	0.5696	0.8105	0.4416	0.3802
17nn	0.6681	0.6751	0.7654	0.5161	0.4098
31nn	0.6975	0.7042	0.7453	0.5445	0.4184
kmin	0.7129	0.7151	0.6183	0.6177	0.5991
kmin2	0.7484	0.7527	0.6385	0.6351	0.6722
kmax	0.7240	0.7273	0.6250	0.6257	0.6234
kmax2	0.7522	0.7561	0.6336	0.6341	0.6709

Tabelle 3: Prognosegenauigkeit unterschiedliche Erwartungswerte

rest Neighbor liegt nun bei 59.37 %, die der Nearest Neighbor Ensmble bei 67.47 % und ist somit um 17 %- bzw. 10 %-Punkte niedriger als für identische Erwartungswerte innerhalb der Klassen. Die Ensemble Technik schneidet hier jedoch besser ab. So ist das Ergebnis für alle Korrelationmatrizen außer der Equikorrelation für die Nearest Neighbor Ensembles besser als die einfache NN-Regel. Auch hier ist die zweite Variante mit Interaktionen der Ensemble-Methode besser als die Erste mit allen Variablen. Maximal werden unter Verwendung der Autokorrelationsmatrix 75 % der Objekte richtig prognostiziert. Der einfache Nearest Neighbor erzielt für die Equikorrelation unter Berücksichtigung des ersten nächsten Nachbars das beste Ergebnis und erreicht 81 %. Grund hierfür könnte die hohe Korrelation zwischen den Prädiktoren sein.

Zusammengefasst kann man sagen, dass die Nearest Neighbor Ensemble für identische Erwartungswerte innerhalb einer Klasse eine höhere Genauigkeit erreichen, für komponentenweise unterschiedliche Erwartungswerte aber im Vergleich besser abschneiden als die einfachen Nearest Neighbor. Egal welcher Erwartungswert zugrunde liegt, in beiden Fällen erzielt die zweite Variante der Ensembles mit selektieren Variablen und Interaktionen ein besseres Ergebnis.

Erhöht man die Variablenanzahl und setzt eine Größe von 200 Variablen voraus, so ergibt sich beispielsweise für die Verwendung der Block-Autokorrelation mit verschiedenen Erwartungswerten und der Equikorrelation mit identischen

Erwartungswerten innerhalb einer Klasse folgendes Ergebnis:

	Blockauto unter	Equi
1nn	0.6413	0.3632
17nn	0.7784	0.4299
31nn	0.8074	0.4718
kmin	0.7572	0.6265
kmin2	0.8076	0.6418
kmax	0.7704	0.6297
kmax2	0.8111	0.6373

Tabelle 4: Prognosegenauigkeit 200 Variablen

Die Prognosegenauigkeit für 200 Variablen der Block-Autokorrelationsmatrix mit komponentenweise unterschiedlichen Erwartungswerten verbessert sich im Vergleich zu 100 verwendeten Variablen deutlich. Die Genauigkeit für den einfachen Nearest Neighbor erhöht sich um ca. 10 %-, für die Ensemble-Methode um ca. 5 %-Punkte.
Unter Verwendung der Equikorrelation bleibt das Ergebnis relativ gleich.
Eine weitere Untersuchung für verschiedene Anzahlen von Prädiktoren wäre interessant, wird in dieser Arbeit aber nicht weiter behandelt.

Bisher wurde nur der Mittelwert der Prognosegenauigkeiten aus den 100 Datensätzen der verschiedenen Methode betrachtet. Im weiteren wird nun die Verteilung der Genauigkeit der simulierten Daten verglichen. Dies geschieht anhand von Boxplots. Dabei ist zu beachten, dass die Skala der Boxplots unterschiedlich ist. Da es vor allem um den Vergleich der Methoden für die jeweilig zugrunde liegende Korrelationsmatrix geht und nicht um den Vergleich der verschiedenen Korrelationsmatrizen, kann dies vernachlässigt werden.

Abbildung 8 zeigt den Boxplot der Prognosegenauigkeiten für die Autokorrelation. Auf der x-Achse sind die verschiedene Methoden abgetragen, die Bezeichnungen sind identische mit den oben beschriebenen. Wie bereits bei der durchschnittlichen Prognosegenauigkeit zu erkennen war, ist für identischen

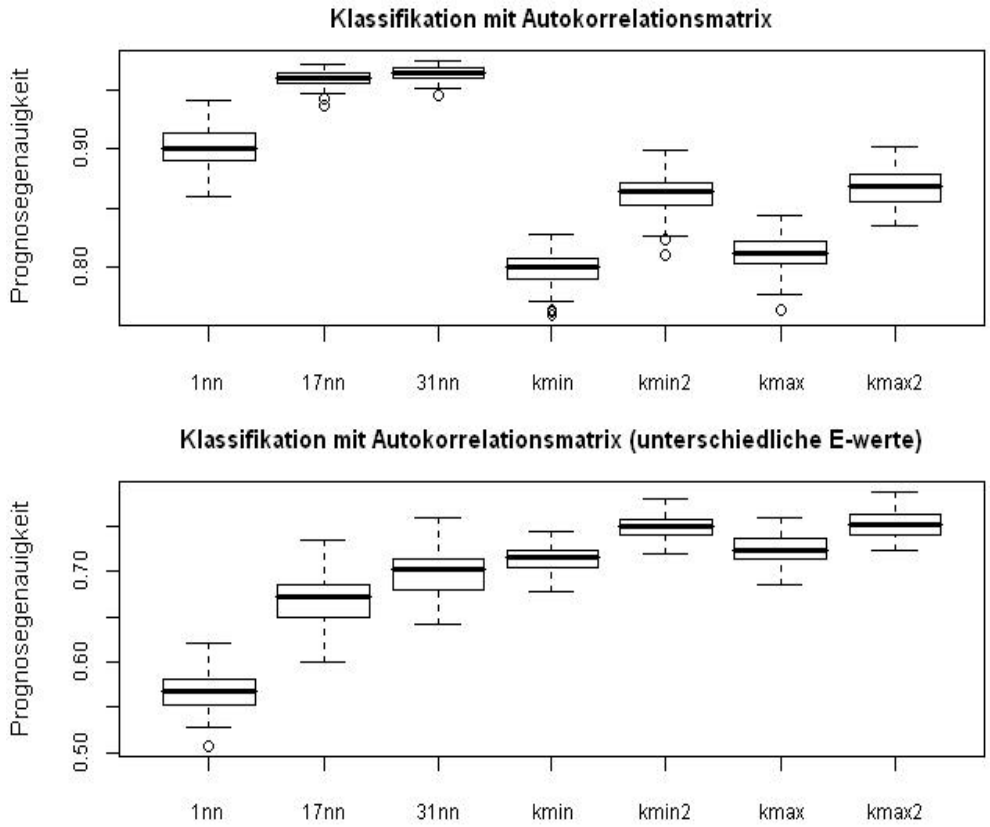

Abbildung 8: Prognosegenauigkeit unter Autokorrelation

Erwartungswert einer Klasse die Prognosegenauigkeit für den einfachen Nearest Neighbor klar besser. Das beste Ergebnis erzielt dabei die NN-Regel für 17 bzw. 31 nächste Nachbarn. Die Ensemble Technik liefert für alle Methoden eine niedrigere Genauigkeit. Erreicht aber im besten Fall für kmin2 und kmax2 das untere Quantil der 1nn Methode. Die Boxplots zeigen ebenfalls das schlechtere Abschneiden der Nearest Neighbor Ensembles bei Betrachtung alle Variablen und fehlenden Interaktionen. So erreicht kmin als bestes Ergebnis die geringste Prognosegenauigkeit von kmin2.
Für unterschiedliche Erwartungswerte liefert die Ensemble Technik die besten Ergebnisse. Im schlechtesten Fall erzielen sie dabei eine Genauigkeit von 69 %. Der Median für 31nn liegt unter dem der Methoden kmin und kmax, jedoch liegt die Streuung der Prognosegenauigkeiten über dem der beiden Ensemble Techniken und liegt im besten Fall bei 76 %.

Abbildung 9 betrachtet die Genauigkeit unter Block-Autokorrelation. Die

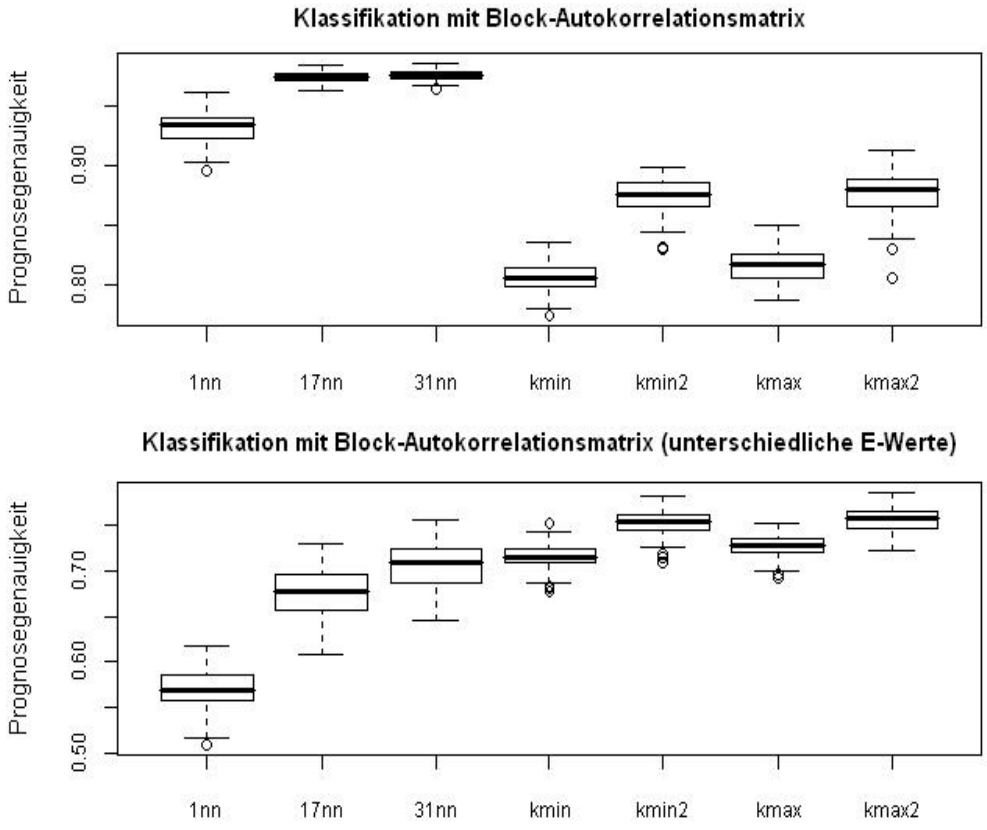

Abbildung 9: Prognosegenauigkeit unter Block-Autokorrelation

Boxplot sind denen unter Autokorrelation sehr ähnlich. Auch hier erzielen 17nn und 31nn für identischen Erwartungswert die besten Ergebnisse und erreichen zwischen 96 % und 98 % richtig klassifizierter Objekte. Die Ensemble Methoden prognostizieren im besten Fall für kmax2 90 % richtig und damit knapp die schlechteste Prognose der einfachen NN-Regel. Für unterschiedliche Erwartungswerte liegen die Prognosegenauigkeiten für 31nn, kmin und kmax nahe beieinander, so liegen ihre Mediane jeweils bei ca. 72 %. Im Vergleich zum gleichen Erwartungswert ist der Unterschied zwischen kmin, kmax und kmin2, kmax2 geringer.
Unter den beiden Autokorrelationsmatrizen ist jeweils bei unterschiedliche

Erwartungswerte die Genauigkeit für die Ensemble Technik höher. Da nur komponentenweise unterschiedliche Erwartungswerte vorliegen, kommt hierbei die Variablenselektion der Klassifikation zugute. Der Erwartungswert wird durch die Variablenselektion „ausgeglichen“.

Unter Equikorrelation (Abbildung 10) sind die Tendenzen entgegengesetzt zu Auto- und Blockautokorrelation. So erzielt hier für gleichen Erwartungswert

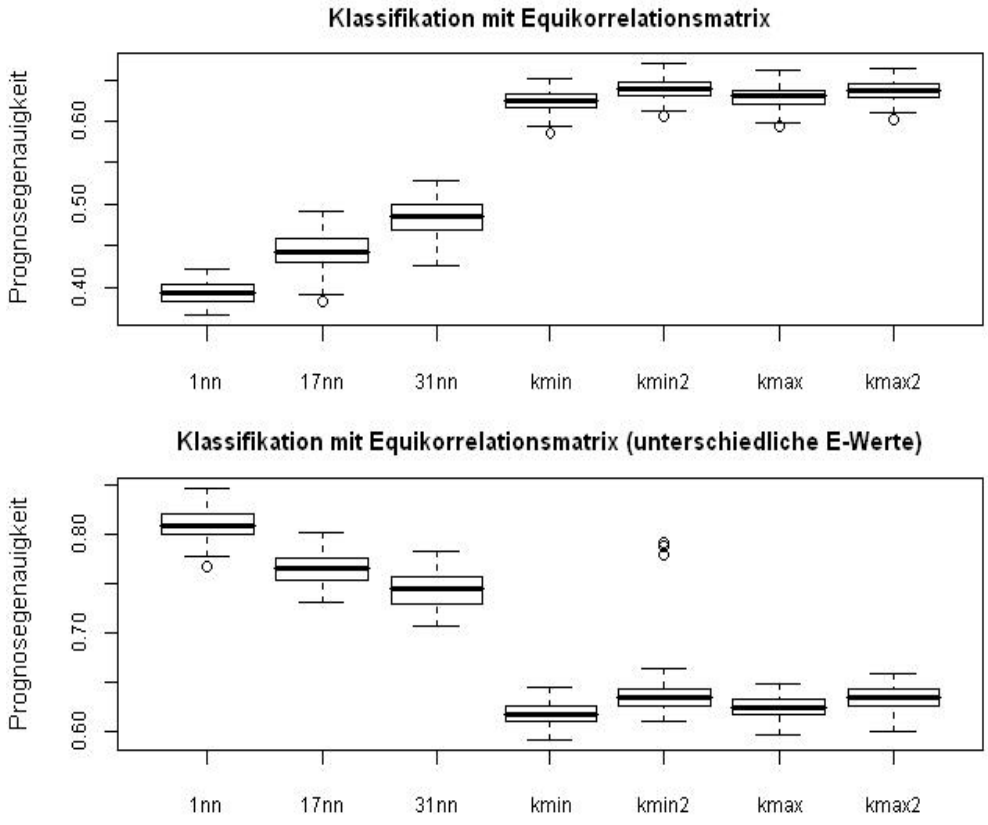

Abbildung 10: Prognosegenauigkeit unter Equikorrelation

die Ensemble Technik ein besseres und für komponentenweise unterschiedliche Erwartungswerte die einfachen Nearest Neighbor ein besseres Ergebnis. Für identischen Erwartungswert liegen die Prognosegenauigkeiten aller Ensemble Methoden fast gleich und streuen zwischen 60-67 %. Auch für unterschiedliche Erwartungswerte liegen die Genauigkeiten der Ensemble Techniken nah beieinander. Im Vergleich zur einfachen Nearest Neighbor Regel

schneidet sie nun deutlich niedriger ab und erreicht im besten Fall 5 %-Punkte weniger als der schlechteste Fall der NN-Regel. Auffällig sind jedoch die Ausreißer von kmin2. Diese erreichen eine Prognosegenauigkeit von 88 % und befinden sich damit auf dem Niveau von 17nn. Ebenfalls auffällig ist, dass die Prognosegenauigkeit mit steigenden betrachteten Nachbarn, im Gegensatz zu allen anderen Korrelationsmatrizen, kontinuierlich fällt.
Ursache für die besser Prognosegenauigkeit der Ensemble Technik für identische Erwartungswerte könnte sein, dass trotz Variablenselektion im erweiterten Sinn alle Variablen betrachtet werden, da diese durch relativ starke Korrelation zusammen hängen.
Betrachte man nun den Boxplot unter Block-Equikorrelation (Abbildung 11) ändert sich die Tendenz und bleibt nicht wie bei Auto- und Blockautokorrelation gleich. Unter gleichen Erwartungswert erzielen 17nn und 31nn die

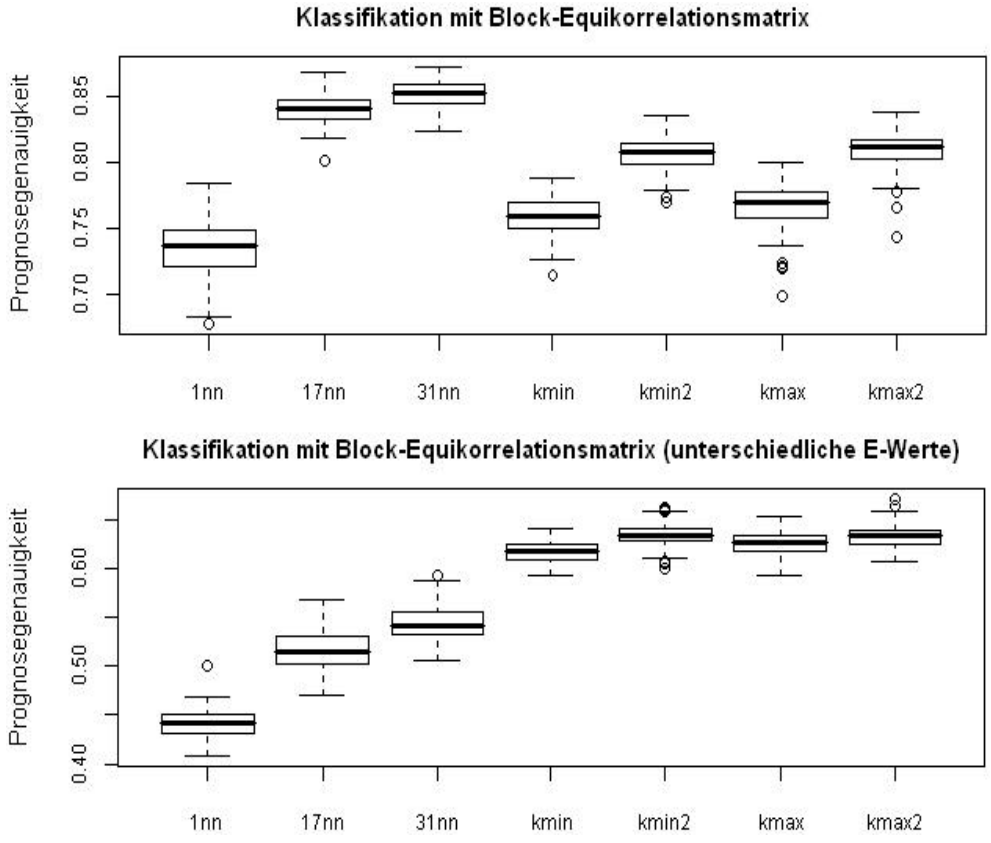

Abbildung 11: Prognosegenauigkeit unter Block-Equikorrelation

höchste Genauigkeit. 1nn hingegen erreicht im Vergleich aller Methoden die niedrigste Prognosegenauigkeit und überschreitet einzig mit einem Ausreißer 50 % richtig klassifizierter Objekte. Die Nearest Neighbor Ensembles erreichen im besten Fall für kmin2 und kmax2 das untere Quartil der 17nn-Regel und weisen im Vergleich von zuvor eine erhöhte Streuung auf. Außerdem sind mehrere Ausreißer nach unten zu erkennen. Liegt ein unterschiedlicher Erwartungswert einzelner Komponenten zugrunde erzielen die Ensembles die besten Ergebnisse. 31nn erreicht als bester Klassifikator unter den Nearest Neighbor gerade noch die niedrigste Rate an richtig prognostizierten Objekten der schlechtesten Ensemble Technik kmin.

Abbildung 12 zeigt die Genauigkeit unter Block-Netzwerkkorrelation. Für

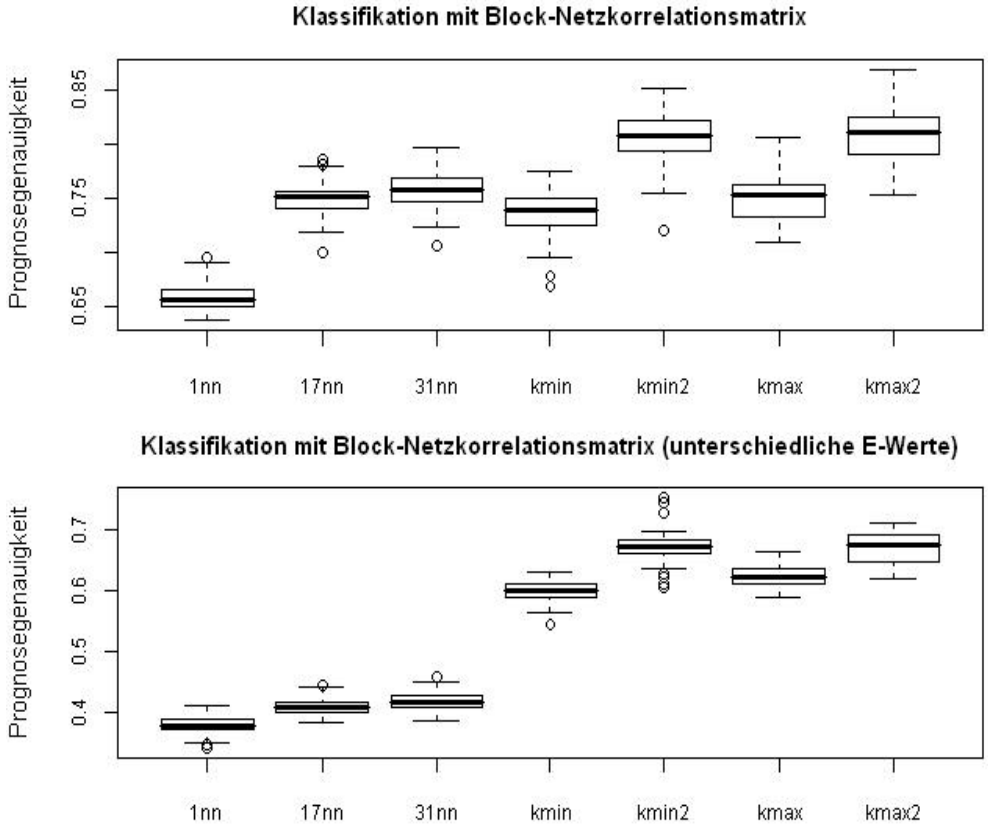

Abbildung 12: Prognosegenauigkeit unter Block-Netzwerkkorrelation

identischen Erwartungswert erzielen kmin2 und kmax2 das beste Ergebnis,

haben jedoch eine große Spannweite von fast 10 %. Auffällig ist, dass der Median des Nearest Neighbor mit 31 nächsten Nachbarn über dem der Ensembles kmin und kmax liegt. Für unterschiedliche Erwartungswerte erzielen die Ensemble Techniken das deutlich bessere Ergebnis. kmin2 zeigt hierbei zahlreiche Ausreißer nach unten und nach oben. Die Prognosegenauigkeit der einfachen Nearest Neighbor liegen nahe zusammen und erreicht bestenfalls für 31nn 44 % richtig klassifizierter Objekte.
Für die beiden Equi- und Blocknetzwerkkorrelationsmatrizen mit komponentenweisen unterschiedlichen Erwartungswert greift vermutlich ebenfalls die Variablenselektion, wie bereits bei Auto- und Blockautokorrelation.
Die Boxplots zeigen, dass die Streuung für die jeweilige Korrelationsmatrix unterschiedlich ist. Liegt ein identischer Erwartungswert zugrunde, haben, mit Ausnahme der Equikorrelation, alle Korrelationsmatrizen für die Ensemble Technik eine weitere Spannweite der Prognosegenauigkeiten als der einfache Nearest Neighbor. Im Vergleich dazu ist die Streuung für unterschiedliche Erwartungswerte, außer unter Block-Netzwerkkorrelation, für die Nearest Neighbor Ensemble geringer.

4.3 Vergleich mit Diskriminanzanalyse

Im folgenden wird die Methode der Nearest Neighbor Ensembles mit der regularisierten Diskriminanzanalyse (RDA) [vgl. Guo et al. (2007)] verglichen. Die regularisierte Diskriminanzanalyse ist eine Erweiterung der LDA. Anstelle von Σ wird nun eine gepoolte Kovarianzmatrix

$$\widetilde{\Sigma} = (1 - \lambda)I_p + \lambda\Sigma$$

eingesetzt. I_p entspricht einer p-dimensionalen Einheitsmatrix und $\lambda \in [0, 1]$ ist ein Tuningparameter, der Σ kontrolliert. Zusätzlich werden geschrumpfte Datenschwerpunkte verwendet. Diese bilden sich über

$$\overline{x}_g^{**} = sgn(\overline{x}_g^*)(\overline{x}_g^* - \Delta)_+$$

$\overline{x}_g^*$ bezeichnet dabei den Schwerpunkt jeder Variablen und wird über $\overline{x}_g^* = \widetilde{\Sigma}^{-1}\overline{x}_g$ berechnet. Für die Diskriminanzanalyse ergibt sich dann der Ausdruck

$$\widetilde{d}_g(x_i) = x_i^T\overline{x}_g^{**} - \frac{1}{2}\overline{x}_g^T\overline{x}_g^{**} + ln\pi_g$$

„Die Elemente von $\overline{x}_g^{**}$ sind dann [nach Definition] entweder gleich 0, wenn das entsprechende Element von $\overline{x}_g^{**}$ vom Betrag her kleiner als die Grenze Δ ist oder, wenn dies nicht der Fall ist, werden sie um den Betrag Δ gegen 0 geschrumpft“. [Thiemichen (2009)] Dies bedeutet für die Diskriminanzanalyse, dass Variablen, die auf 0 geschrumpft worden sind, keinen Einfluss auf die Klassifikation haben. [vgl. Fiessler (2010)]

Die Berechnung von λ und Δ erfolgt über die in R bereits implementierte Funktion `rda`. Die eigentliche Klassifikation erfolgt über den Befehl `rda.predict`, der die prognostizieren Klassen zurück gibt. Tabelle 5 zeigt die Prognosege-

	gleicher Ewert	unterschiedl. Ewert
Auto	0.9525	0.7716
Blockauto	0.9703	0.9702
Equi	0.5638	0.5633
Blockequi	0.8359	0.5616
Blotznetz	0.9904	0.7857

Tabelle 5: Prognosegenauigkeit RDA

nauigkeiten der regularisierte Diskriminanzanalyse. Für gleichen Erwartungswert wird durchschnittlich ein sehr gutes Ergebnis erzielt, für unterschiedlichen Erwartungswert erreicht sie eine niedrigere Genauigkeit. Vergleicht man die Werte mit der besten Methode der Nearest Neighbor Ensembles kmax2, liegt die Genauigkeit der RDA meistens höher. Nur unter Equikorrelation erreichen die Ensemble Technik ein besseres Ergebnis.

5 Verwendung von realen Datensätzen

In den nächsten Abschnitten werden zur Untersuchung der Methoden reale Daten benutzt. Grundlage dafür sind Microarray Datensätze, diese haben zur Eigenschaft sehr viele Variablen, aber nur eine geringe Anzahl an Ausprägungen, zu haben. Bei den Prädiktoren handelt es sich um sogenannte Genexpressionen. "Genexpression bezeichnet hierbei die Transkription von DNA in mRNA"[Schillinger (2007)] von Genen. Microarrays werden zum Beispiel dafür eingesetzt Krebstypen zu klassifizieren oder den Verlauf von Krankheiten vorher zusagen.

5.1 Datensätze

Die hier verwendeten Datensätze sind in R bereits vorliegend und in den Paketen `rda` und `spls` zu finden.

colon Der Datensatz colon basiert auf Untersuchungen über Dickdarmkrebs. Er beinhaltet 6500 Genexpression von 40 Krebs- und 22 normalen (ohne Krebszellen) Gewebeproben. [s. Alon et al. (1999)] Effektiv verwendet werden nur 2000 Expressionen, die mit der höchsten minimal Intensität. Die Gewebeproben werden von 40 Patienten genommen, wobei von 22 sowohl Krebs- als auch normale Proben genommen werden. [vgl. Parmigiani et al. (2003)]

prostate Der Prostatakrebs Datensatz [s. Singh et al. (2002)] umfasst 6033 Genexpressionsdaten. Es sind 52 Krebsgewebeproben, sowie von 50 Personen zusätzlich gesunde Proben vorliegend. Aufgrund der sehr großen Datenmenge wird prostate auf 2000 Variablen gekürzt. Dies erfolgt mit Hilfe eines t-Tests. Der t-Test überprüft über den Erwartungswert den Wert des Effekts zwischen zwei Population jeder Variablen einzeln. Dabei vergleicht er für den gegebenen Datenfall die kranken mit den gesunden Gewebeproben. Daraus ergeben sich die Hypothesen H_0 : Gen hat keinen Effekt und H_1 : Gen hat einen Effekt. Anders ausgedrückt $H_0: \ \mu_1 = \mu_2$ und $H_1: \ \mu_1 \neq \mu_2$. μ_1 bezeichnet

dabei den Erwartungswert der ersten Population, also der Krebsgewebenproben, und μ_2 den der gesunden Proben. Als Teststatistik wird

$$t = \frac{\overline{x}_1 + \overline{x}_2}{S\sqrt{\frac{1}{n} + \frac{1}{m}}}$$

mit $S = \frac{(n-1)S_1^2+(m-1)S_2^2}{n+m-2}$ verwendet. Sortiert man nun die Gene nach ihre t-Werten werden für die weiteren Analysen die Gene mit den größten Werten verwendet. Die t-Werte könne hier demnach als Gewicht angesehen werden. Für den Datensatz prostate heißt das, dass 2000 Prädiktoren mit den größten t-Werten ausgewählt werden.

5.2 Ergebnisse

Die vorgestellten Methoden, Nearest Neighbor, Nearest Neighbor Ensemble und RDA werden auf diese Daten angewendet. Als Unterschied zu den Simulationen liegen hier bei den Datensätzen keine getrennten Trainings- und Testdaten vor. Deshalb erfolgt eine Aufteilung so, dass ein Testdatensatz mit 17 Beobachtungen entsteht und die restlichen Beobachtungen als Trainingsdaten verwendet werden. Diese Aufspaltung wird zufällig hundert mal wiederholt. Man erhält demnach 100 verschiedene Datensätze mit jeweils 17 Testdaten. Zusätzlich wird für die Bestimmung der Parameter λ und γ der regularisierten Diskriminanzanalyse eine Kreuzvalidierung vorgenommen.
Die Anzahl der betrachteten Nachbarn wird wie zuvor über die Funktion `kknn` bestimmt. Es wird sowohl für colon, als auch prostate $k = 3$ und $k = 6$ festgesetzt.
Die Methoden werden für alle 100 Datensätze angewendet. Als Ergebnis wird im weiteren wieder die durchschnittliche Prognosegenauigkeit betrachtet. Tabelle 6 zeigt die Prognosegenauigkeiten der beiden vorgestellten Datensätze. Für colon sind die Genauigkeiten für alle Methoden relativ gut. Sie liegen, mit Ausnahme der RDA, im Bereich von 75 %. Die regularisierten Diskriminanzanalyse erreicht sogar 85 % richtig klassifizierter Objekte. Wie sich bereits bei den Simulationen gezeigt hat, erzielt die Ensemble Technik mit selektierten Variablen und unter Berücksichtigung von Interaktionen ein

	colon	prostate
1nn	0.7512	0.8341
3nn	0.7835	0.8506
6nn	0.7853	0.8429
nnekmin	0.7276	0.8241
nnekmin2	0.7629	0.8294
nnekmax	0.7529	0.8829
nnekmax2	0.7900	0.8453
rda	0.8488	0.3041

Tabelle 6: Prognosegenauigkeit reale Daten

besseres Ergebnis als ohne. Die Genauigkeit von kmax2 liegt sogar über den Ergebnissen des einfachen Nearest Neighbor. Dieser erreicht unter 6 betrachteten Nachbarn sein bestes Ergebnis.
Für den prostate Datensatz liegt die Prognosegenauigkeit höher. So erzielt die NN-Regel durchschnittlich 0.84 % und die Ensemble Technik 85 % richtig klassifizierte Objekte. Die regularisierte Diskriminanzanalyse erreicht nur einen Wert von 30 %. Das beste Ergebnis weist der Nearest Neighbor Ensembles kmax auf. Auffällig ist hier, dass mit selektierten Variablen und Interaktionen keine Verbesserung der Prognosegenauigkeit auftritt.

5.3 Genlisten

Eine weitere Möglichkeit die im Zusammenhang mit Microarrays häufig angewandt wird, sind Genlisten.

5.3.1 Konzept

Da Genexpressionen eine sehr große Dimension haben, ist es sinnvoll einige Gene herauszufiltern, die für die Klassifikation ausschlaggebend sind. Dabei werden die Gene gewichtet und basierend darauf in einer Reihenfolge sortiert. Die „höchstbewertesten Gene können dann als Ausgangsbasis für weiterführende Analysen verwendet werden“. [Schillinger (2007)]
Problematisch bei diesem Ansatz ist die biologische Relevanz der Gene. So können mehrere gute Klassifikatoren auftreten, die jedoch unterschiedliche

Gene selektieren. Eine allgemeingültige Relevanz der Gene wäre somit nicht gewährleistet und vom Klassifikator abhängig. Dieses Phänomen wird in zahlreichen Abhandlungen behandelt. In dieser Arbeit wird die Problematik außer acht gelassen.

Die Methode der Nearest Neighbor Ensembles führt bereits eine Variablenselektion durch. Wie gut ist diese jedoch, wenn man zum Vergleich eine andere Technik zur Selektion verwendet? Dies wird mit Hilfe des t-Tests und dem Wilcoxon-Rangsummentest behandelt.
Der t-Test funktioniert wie bereits in Abschnitt 5.1 erklärt. Es geht dabei darum die Gene nach ihren t-Werten zu ordnen und die mit den größten Werten weiter zu verwenden.

Der Wilcoxon-Rangsummen- oder auch Mann-Whitney-Test gehört zur Familie der nonparametrischen Tests. Er arbeitet mit dem Rang der Expressionsdaten und nicht mit den eigentlichen Ausprägungen. Wilcoxon überprüft, ob sich die Verteilungen der Grundgesamtheiten zweier Stichproben bezüglich ihrer Lage unterscheiden. Dabei werden die Mediane der beiden Größen verglichen. Eine Verteilungsannahme liegt nicht zugrunde. Die Hypothesen ergeben sich dadurch als $H_0 : \ x_{1,med} = x_{2,med}$ und $H_1 : \ x_{1,med} \neq x_{2,med}$. Zur Berechnung der Wilcoxon-Prüfgröße werden die Ränge der Elemente aufsummiert, d.h. $W = \sum_{i=1}^{n} R(x_i)$. [vgl. Klausl and Zuber (2009)] Auch hier werden die Variablen anschließend nach ihren W-Werten sortiert und die mit den größten Werten selektiert.

5.3.2 Resultat

Grundlage der folgenden Auswertungen sind wieder die beiden Microarray Datensätze colon und prostate. Diesmal erfolgt jedoch keine Aufteilung der Daten, sondern es wird ein Datensatz angenommen. Um die Gewicht aller Variablen über die Nearest Neighbor Ensembles zu erhalten, wird die Threshold-Schwelle t gleich Null gesetzt. Für die beiden Tests werden die gesunden gegen die kranken Gewebeproben getestet. Man erhält demnach für

alle 2000 Prädiktoren des jeweiligen Datensatzes drei unterschiedliche Gewichte, die der Nearest Neighbor Ensembles, des t-Tests und des Wilcoxon-Rangsummen-Tests.
Fürs erste werden die Variablen mit den 100 größten Gewichten ermittelt. Vergleicht man nun die vorkommenden Prädiktoren der Selektionsverfahren mit der jeweils anderen, also die Übereinstimmungen, ergibt sich folgendes Ergebnis, siehe Tabelle 7. Dabei steht t für den t-Test, w für den Wilcoxon-

	colon	prostate
t&w	87	87
kmin&kmax	62	58
kmin&t	30	38
kmin&w	34	38
kmax&t	32	46
kmax&w	34	46

Tabelle 7: Übereinstimmungen an Variablen

Rangsummen-Test. kmin bzw. kmax beziehen sich auf die Nearest Neighbor Ensembles mit einer kleineren und einer größeren Anzahl an betrachteten Nachbarn. Auch hier wurde $k = 3$ und $k = 6$ verwendet.
Vergleicht man jeweils die artverwandten Methoden, also t und w oder kmin und kmax, kann man erkennen, dass der t- und Wilcoxon-Test sowohl für colon, als auch für prostate hohe Übereinstimmungen vorweist. So werden 87 gleiche Gene selektiert. Bei der Ensemble Technik ist das Ergebnis etwas niedriger, die Daten zeigen 62 bzw. 58 Übereinstimmungen. Für die anderen vier Vergleiche werden durchschnittlich 37 identische Gene selektiert, d.h knapp 1/3 der Prädiktoren werden durch die Verfahren gleich selektiert. Wobei für den Datensatz prostate die Anzahl etwas höher liegt. Außerdem fällt auf, dass kmin und kmax verglichen mit t oder w dieselbe Anzahl an Übereinstimmungen liefert.

Bisher wurden die ersten 100 Variablen mit den größten Gewichten betrachtet. Werden nun wieder alle Prädiktoren mit einbezogen, ist es interessant, wie die Gene für die verschiedenen Selektionsverfahren eingestuft werden.

Dafür bekommt jedes Gene je nach Gewicht einen Rang zugeordnet. Das Gene mit dem größten Gewicht bekommt demnach Rang 1, das mit den zweitgrößten Rang 2 und so weiter. Nun werden die Differenzen der Ränge von jeweils zwei Selektionsverfahren gebildet. Man erhält dadurch die Abweichungen der Position des jeweiligen Genes. Für den Vergleich zwischen t-Test und den Nearest Neighbor Ensembles kmax des prostate Datensatzes bekommt man ausschnittsweise für die ersten 15 Gene:
$0, 0, -382, -668, 0, -56, -8, -32, -84, -178, -706, -24, -40, -40, -35, \ldots$.
0 bedeutet das Gen hat in beide Verfahren die gleiche Position bekommen, alle anderen Zahlen zeigen an, ob das Gen weiter vorne bzw. weiter hinten platziert worden ist.
Insgesamt zeigen sich für die Datensätze erheblicher Differenzen. D.h. die Ränge der Variablen sind für die verschiedenen Verfahren sehr unterschiedlich. Die wenigsten Rangdifferenzen ergeben sich dabei noch für den Vergleich von t und w bzw. kmin und kmax. Berechnet man den Spearman-Rangkorrelationskoeffizient für die beiden Vergleiche, also ein Maß für den monotonen Zusammenhang, ergibt sich ein Wert von ca. 0.76. Dies bedeutet, dass ein deutlicher Zusammenhang zwischen t-Test und Wilcoxon-Test, sowie den beiden Nearest Neighbor Ensembles zu erkennen ist.
Die anderen Vergleiche zeigen Rangdifferenzen von beispielsweise bis zu 1998 Stellen für den Datensatz colon auf. D.h. ein Gene wurde für das eine Verfahren als sehr wichtig, und für das andere als eher unwichtig eingestuft. Dies veranschaulicht das bereits erwähnte Problem, der Relevanz der Gene.

6 Zusammenfassung

Zusammenfassend kann man sagen, dass die Nearest Neighbor Ensembles eine gute Methode zur Klassifikation im Fall höherdimensionaler Daten darstellt. Gerade an realen Datensätzen zeigt sich ein gutes Resultat. Für beide verwendeten Datensätze wird durch die Nearest Neighbor Ensembles die höchste Prognosegenauigkeit erreicht. Mit einer Ausnahme ist das Ergebnis mit selektieren Prädiktoren und Interaktionen besser, als unter Miteinbeziehung aller Variablen.

Die Ergebnisse der Simulationen sind genauer zu betrachten. Hier zeigt sich je nach zugrunde liegender Korrelationsmatrix eine höhere oder niedrigere Prognosegenauigkeit als für die einfache NN-Regel. Für die Ensemble Technik gilt in jedem Fall, dass die Miteinbeziehung der Variablenselektion und Interaktionen ein besseres Ergebnis hervorruft.
Die Variablenselektion erhöht die Genauigkeit der Nearest Neighbor Ensembles, ist jedoch als kritisch anzusehen. Da in der Klassifikationsentscheidung die wissenschaftliche Relevanz der Prädiktoren nicht berücksichtigt wird. Die hohen Rangdifferenzen der Genlisten zeigen diese Problematik.

Im Vergleich zum einfachen Nearest Neighbor ist die Ensemble Methode weniger von der Größe von k abhängig. Die NN-Regel zeigt stark schwankende Ergebnisse für unterschiedliche Anzahl betrachteter Nachbarn. Die Nearest Neighbor Ensembles verändern ihre Prognosegenauigkeit hingegen nur geringfügig. Auffällig dabei ist, auch $k = 1$ weist eine gute Genauigkeit auf. Meistens zeigt sich, dass ein größeres k ein besseres Ergebnis liefert.
Diese Arbeit beschäftigt sich ausschließlich mit dem 2- bzw. 3-Klassenfall. Eine Überprüfung der Nearest Neighbor Ensembles für einen mehr Klassenfall wäre eine interessante Themenstellung für die weitere Forschung.

Nonparametrische Klassfikationsverfahren werden immer häufiger verwendet. Ein großer Vorteil der Methode besteht darin, dass rein mit den Daten gearbeitet wird. Es müssen keine Annahmen über die Verteilung getroffen werden. Im Fall höherdimensionaler Daten führt die datenbezogenen Herangehensweise zum Teil zu lange Berechnungszeiten, gerade dann, wenn Interaktionen mit betrachtet werden.
Diese Arbeit zeigt, dass Nearest Neighbor Ensembles gerade an realen Daten ein gutes Ergebnis liefern und als vielversprechender Ansatz für weitere Entwicklungen angenommen werden können.

7 Literaturverzeichnis

Literatur

Alon, U., N. Barkai, D. Notterman, and et al (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences 96*(12), 6745.

Bay, S. (1999). Nearest neighbor classification from multiple feature subsets. *Intelligent Data Analysis 3*(3), 191–209.

Cover, T. and P. Hart (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory 13*(1), 21–27.

Domeniconi, C. and B. Yan (2004). Nearest neighbor ensemble. *Pattern Recognition 1*, 228–231.

Fahrmeir, L., A. Hamerle, and G. Tutz (1996). *Multivariate statistische Verfahren*. Walter de Gruyter.

Fahrmeir, L., T. Kneib, and S. Lang (2007). *Regression: Modelle, Methoden und Anwendungen*. Springer.

Fiessler, C. (2010). Regularisierte Diskriminanzanalyse. *Seminararbeit*.

Fix, E. and J. Hodges Jr (1989). Discriminatory analysis. Nonparametric discrimination: Consistency properties. *International Statistical Review*, 238–247.

Gertheiss, J. and G. Tutz (2009a). Feature selection and weighting by nearest neighbor ensembles. *Chemometrics and Intelligent Laboratory Systems 99*(1), 30–38.

Gertheiss, J. and G. Tutz (2009b). Variable scaling and nearest neighbor methods. *Journal of Chemometrics 23*(3), 149–151.

Guo, Y., T. Hastie, and R. Tibshirani (2007). Regularized linear discriminant analysis and its application in microarrays. *Biostatistics 8*(1), 86.

Hastie, T., R. Tibshirani, and J. Friedman (2009). The elements of statistical learning: data mining, inference and prediction. *Journal of Chemometrics 99*, 30–38.

Hornung, M. (2005). Klassifikation hochdimensionaler Daten unter Anwendung von Box-Cox Transformationen. *Diplomarbeit*.

Klausl, B. and V. Zuber (2009). Eine Einführung in R:Statistische Tests. *Präsentation Universität Leipzig*.

Nothnagel, M. (1999). Klassifationsverfahren der Diskriminanzanalyse. *Diplomarbeit*.

Parmigiani, G., E. Garrett, R. Irizarry, and S. Zeger (2003). *The analysis of gene expression data: an overview of methods and software*. Springer.

Schillinger, C. (2007). Robuste Merkmalsselektion aus Bipartitioen von Microarry-Expressionsdaten mit Anwendung auf aggregierte Tumordaten. *Diplomarbeit*.

Singh, D., P. Febbo, K. Ross, and et al (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer cell 1*(2), 203–209.

Sprung, S. (2004). Generalisierte additive Modelle. *Seminararbeit*, 9.

Stuckart, C. (2009). L1-Penalisierung im multiplen Regressionsmodell: Lasso. *Seminararbeit*.

Thiemichen, S. (2009). Regularisierte Lineare Diskriminanzanalyse. *Seminararbeit*.

Whittaker, J. (2009). Graphical models in applied multivariate statistics. pp. 162–164.

Zierau, H. (2007). Andere Methoden zur Klassifkation undn Objekterkennung. pp. 5–12.

8 Anhang

```
## Funktion nearest neighbor Ensemble

library(lars)
library(quadprog)
library(kernlab)



fitWeights <- function(ydata,Xdata,loss="quadscore",
method="ridge",threshold=0.5,categorial=T,maxinter=1,
k=1,ordered=F,K=3,printFold=T,neworder=NULL) {

# a function for classification as described in
"Feature selection and # weighting by nearest neighbor
ensembles" bei Gertheiss & Tutz (2009)
 if (loss=="logscore" | !categorial) {
  stop("currently only categorial outcomes and
  quadscore are supported") }
 n <- length(ydata)
 if (categorial) {
  categories <- as.numeric(levels(factor(ydata)))
  ncat <- length(categories)
  if (loss=="logscore") {
   yd <- rep(2,n)
   ica <- (1:n)[ydata==categories[1]]
   for (ca in 2:ncat) {
    ica <- c(ica,(ca-1)*n + (1:n)[ydata==categories[ca]])
 }}
 else {
  yd <- rep(0,n)
  yd[ydata==categories[1]] <- 1
  ica <- 1:(ncat*n)
```

```
  for (ca in 2:ncat) {
   ydca <- rep(0,n)
   ydca[ydata==categories[ca]] <- 1
   yd <- c(yd,ydca)
  }}}
  Xplus <- cbind(1:n,Xdata)
  yD <- 0
  NN <- apply(Xplus,1,which.nn,XX=Xdata,maxinter=maxinter,
neighbors=k, ordered=ordered)
nnseq <- seq(0,(dim(NN)[1]-k),by=k)
  for (nei in 1:k) {
   yData <- apply(NN[nei+nnseq,],1,predict.allNN,y=ydata)
if (categorial) {
     yDatadummy <- matrix(0,n,dim(yData)[2])
     yDatadummy[yData==categories[1]] <- 1
      for (ca in 2:ncat) {
       yDatadummyca <- matrix(0,n,dim(yData)[2])
       yDatadummyca[yData==categories[ca]] <- 1
       yDatadummy <- rbind(yDatadummy,yDatadummyca) }
      yD <- yD + yDatadummy }
     else {
      yD <- yD + yData
    }}
   yD <- yD/k
   if (categorial) {
    if (length(threshold) > 1) {
     nK <- floor(n/K) }
    if (length(neworder)==0) {
     neworder <- sample(n,n) }
    cvscore <- numeric(length(threshold))

  for (k in 1:K) {
## Training and test data
```

```
 if (printFold)
  cat("Fold",k,"\n")
 if (k < K) {
  ik <- neworder[(k-1)*nK + (1:nK)] }
   else {
    ik <- neworder[((K-1)*nK+1):n] }
   nik <- length(ik)
   ik <- rep(ik,ncat) + sort(rep((0:(ncat-1))*n,nik))

for (th in seq(along=threshold)) {
 if (method=="lasso") {
  object <- lars(x=yD[-ik,],y=yd[-ik],type="lasso",
  use.Gram=F,intercept=F, normalize=F)
  maxnorm <- sum(abs(predict.lars(object,s=ncol(yD),
  type="coef",mode="step")$coef))
  w <- coef(object,s=min(1,maxnorm),mode="norm") }
 else if (method=="ridge") {
  w <- ridge.coef(yd[-ik],yD[-ik,]) }
 else if (method=="nnls") {
  w <- nnls(rbind(yD[-ik,],1),c(yd[-ik],1)) }
 else if (method=="quadprog") {
  Dmat <- crossprod(yD[-ik,],yD[-ik,])
  dvec <- crossprod(yD[-ik,],yd[-ik])
  Amat <- cbind(1,diag(1,ncol(yD)))
  bvec <- c(1,rep(0,ncol(yD)))
  w <- solve.QP(Dmat=Dmat, dvec=dvec, Amat=Amat,
  bvec=bvec, meq=1)$solution }
 else if (method=="ipop") {
  A <- matrix(1,1,ncol(yD))
  l <- rep(0,ncol(A))
  u <- rep(1,ncol(A))
  b <- 1
  r <- 0
```

```
    result <- ipop(c=-t(yD[-ik,])%*%yd[-ik],
    H=t(yD[-ik,])%*%yD[-ik,], A=A, b=b, l=l, u=u, r=r, verb=F)
    w <- result@primal }
   else if (method=="optim") {
    w <- optim(par=rep(1/ncol(yD),ncol(yD)-1),fn=kl,
    method="L-BFGS-B",gr=grad,lower=0.01,upper=0.99,
    P=yD[ica,])$par
    w <- c(w,1-sum(w))
    print(w) }
   else {
    lm.train <- lm(yd[-ik] ~ 0 + yD[-ik,])
    w <- lm.train$coef }
   w[w<0] <- 0
   w <- w/sum(w)
   w[w<(threshold[th]*max(w))] <- 0
   w <- w/sum(w)
   cvscore[th] <- cvscore[th] + sum((yd[ik] - yD[ik,]%*%w)^2)
}}

  threshold <- threshold[which.min(cvscore)]
#print(threshold) }
  else {
   cvscore <- NA
   neworder <- NA }
  if (method=="lasso") {
   object <- lars(x=yD[ica,],y=yd,type="lasso",
   use.Gram=F,intercept=F,normalize=F)
   maxnorm <- sum(abs(predict.lars(object,s=ncol(yD),
   type="coef",mode="step")$coef))
   w <- coef(object,s=min(1,maxnorm),mode="norm") }
  else if (method=="ridge") {
   w <- ridge.coef(yd,yD[ica,]) }
  else if (method=="nnls") {
```

```
 w <- nnls(rbind(yD[ica,],1),c(yd,1)) }
else if (method=="quadprog") {
 Dmat <- crossprod(yD[ica,],yD[ica,])
 dvec <- crossprod(yD[ica,],yd)
 Amat <- cbind(1,diag(1,ncol(yD)))
 bvec <- c(1,rep(0,ncol(yD)))
 w <- solve.QP(Dmat=Dmat, dvec=dvec, Amat=Amat,
 bvec=bvec, meq=1)$solution }
else if (method=="ipop") {
 A <- matrix(1,1,ncol(yD))
 l <- rep(0,ncol(A))
 u <- rep(1,ncol(A))
 b <- 1
 r <- 0
 result <- ipop(c=-t(yD[ica,])%*%yd, H=t(yD[ica,])%*%yD[ica,],
 A=A, b=b, l=l, u=u, r=r, verb=F)
 w <- result@primal }
else if (method=="optim") {
 w <- optim(par=rep(1/ncol(yD),ncol(yD)-1),fn=kl,
 method="L-BFGS-B",gr=grad,lower=0.01,upper=0.99,
 P=yD[ica,])$par
 w <- c(w,1-sum(w))
 print(w) }
else {
 lm.train <- lm(yd ~ 0 + yD[ica,])
 w <- lm.train$coef}
 w[w<0] <- 0
 w <- w/sum(w)
 w[w<(threshold*max(w))] <- 0
 w <- w/sum(w)
 yPred <- matrix(yD%*%w,n,ncat)
 ypred <- apply(yPred,1,which.max)
 ypred <- categories[ypred] }
```

```
  else {
   if (method=="lasso") {
    object <- lars(x=yD,y=ydata,type="lasso",
    use.Gram=F,normalize=F, intercept=F)
    w <- coef(object,s=1,mode="norm") }
   else if (method=="ridge") {
    w <- ridge.coef(ydata,yD) }
   else {
    lm.train <- lm(ydata ~ 0 + yD)
    w <- lm.train$coef}
    w[w<0] <- 0
    w <- w/sum(w)
    w[w<(threshold*max(w))] <- 0
    w <- w/sum(w)
    ypred <- yData%*%w
    yPred <- NULL }
   return(list("coefficients"=w,"prediction"=ypred,
   "plearn"=yPred, "cvScore"=cvscore,"newOrder"=neworder)) }


nnEnsemble <- function(ytrain,Xtrain,Xtest,loss="quadscore",
method="ridge",threshold=0.5,categorial=T,maxinter=1,ordered=F,
k=1, K=3,printFold=F) {
  n <- dim(Xtest)[1]
  test <- fitWeights(ytrain,Xtrain,loss=loss,method=method,
  threshold=threshold, maxinter=maxinter,categorial=categorial,
  ordered=ordered,k=k,K=K, printFold=printFold)
  if (categorial) {
   categories <- as.numeric(levels(factor(ytrain)))
   ncat <- length(categories) }
  yD <- 0
  NN <- apply(Xtest,1,which.nn,XX=Xtrain,train=F,
  maxinter=maxinter, neighbors=k,ordered=ordered)
```

```
 nnseq <- seq(0,(dim(NN)[1]-k),by=k)
 for (nei in 1:k) {
  yData <- apply(NN[nei+nnseq,],1,predict.allNN,y=ytrain)
  if (categorial) {
   yDatadummy <- matrix(0,dim(yData)[1],dim(yData)[2])
   yDatadummy[yData==categories[1]] <- 1
   for (ca in 2:ncat) {
    yDatadummyca <- matrix(0,dim(yData)[1],dim(yData)[2])
    yDatadummyca[yData==categories[ca]] <- 1
    yDatadummy <- rbind(yDatadummy,yDatadummyca) }
   yD <- yD + yDatadummy }
   else {
    yD <- yD + yData }}
  yD <- yD/k
 if (categorial) {
  yPred <- matrix(yD%*%test$coefficients,n,ncat)
  ypred <- apply(yPred,1,which.max)
  ypred <- categories[ypred] }
 else {
  ypred <- yD%*%test$coefficients
  yPred <- NULL }
 return(list("fit"=ypred,"ptest"=yPred,"plearn"=test$plearn,
 "coefficients"=test$coefficients))}

which.nn <- function(x,XX,train=T,maxinter=1,neighbors=1,ordered=T){
 if (train) {
  xXX <- (t(t(XX) - x[-1]))^2 }
 else {
  xXX <- (t(t(XX) - x))^2 }
 if (maxinter>1) {
  xXXX <- NULL
  for (l in 2:maxinter) {
   if (ordered) {
```

```
    Comb <- matrix(rep(1:(dim(XX)[2]-l+1),l),l,
    dim(XX)[2]-l+1,byrow=T) + 0:(l-1) }
   else {
    Comb <- combn(dim(XX)[2],l) }
   xXXl <- apply(X=Comb,MARGIN=2,FUN=add.cols,M=xXX)
   xXXX <- cbind(xXXX,xXXl) }
  xXX <- cbind(xXX,xXXX)´}
 if (train) {
  xXX[x[1],] <- max(xXX) + 1 }
return(apply(xXX,2,which.neighbors,lastneighbor=neighbors))
}

add.cols <- function(cols,M) {
 return(rowSums(M[,cols])) }

which.neighbors <- function(x,lastneighbor=1) {
 return(order(x)[1:lastneighbor]) }

predict.allNN <- function(nn,y) {
 ypred <- y[nn]
 return(ypred) }

ridge.coef <- function(y,X,lambda=1) {
 p <- dim(X)[2]
 cholhelp <- chol(t(X)%*%X + lambda*diag(1,p))
 invhelp <- chol2inv(cholhelp)
 return(invhelp%*%t(X)%*%y) }

kl <- function(w,P) {
 #print(w)
 #print(log(1/crossprod(t(P),c(w,1-sum(w)))))
 return(sum(log(1/crossprod(t(P),c(w,1-sum(w)))))) }
```

```
grad <- function(w,P) {
 k <- ncol(P)
 pn <- crossprod(t(P),c(w,1-sum(w)))
 return(as.numeric(-0.1*crossprod(1/pn,P[,-k]-P[,k]))) }
```