Kapitel 3
Klassifikationsverfahren

In diesem Kapitel werden die in dieser Arbeit verwendeten Klassifikationsverfahren vorgestellt. Bei den ersten Verfahren handelt es sich um rein auf den Nächsten Nachbarn basierende Ansätze. Im Anschluss daran wird zu Vergleichszwecken kurz auf die lineare sowie quadratische Diskriminanzanalyse eingegangen. Die logistische Regression markiert das erste Verfahren, welches sowohl als klassisches Klassifikationsverfahren (basierend auf Kovariablen), als auch als Nächste Nachbarn Verfahren eingesetzt wird. Das Hauptaugenmerk dieses Kapitels liegt auf dem Lasso-Verfahren, sowie den Random Forest und Boosting Ansätzen. Diese sollen konkret mit Hilfe der Informationen aus den Nächsten Nachbarn verbessert werden. Zu guter letzt wird noch ein Ensemble Ansatz basierend auf dem Lasso-Verfahren vorgestellt. Bei allen Ansätzen wurden jedoch nicht alle Aspekte erläutert, sondern nur diejenigen, die in einem direkten Bezug zur Analyse binärer Daten stehen. Darüber hinaus sind die einzelnen Verfahren mit Quellenangaben versehen um ausführlichere Informationen über die unterschiedlichen Ansätze zur Verfügung zu stellen.

Um dieses Kapitel nicht ganz so theorielastig zu halten, werden die Klassifikationsverfahren direkt auf einen Simulationsdatensatz angewandt. Es handelt sich um die zweidimensionalen mlbench Daten, welche in Kapitel 4.1 genauer beschrieben werden. Die Wahl ist auf diesen Datensatz gefallen, da durch den zweidimensionalen Charakter die Visualisierung einiger Verfahren ermöglicht wird, was zu einem besseren Verständnis beitragen soll. Darüber hinaus ist – im Gegensatz zu realen Datensätzen – die zu Grunde liegende Struktur bekannt, was die Interpretation der Ergebnisse erleichtert. Dies macht sich insbesondere bei der Variablenwichtigkeit bemerkbar, da nachgeprüft werden kann, wie sich die Gewichtung auf die unterschiedlichen Parameter verteilt.

D. Koch, *Verbesserung von Klassifikationsverfahren,* BestMasters,
DOI 10.1007/978-3-658-11476-3_2, © Springer Fachmedien Wiesbaden 2016

Neben der grundlegenden Theorie wird zu jedem Verfahren der Algorithmus als eine kurze Zusammenfassung des Verfahrens zur Verfügung gestellt. Darüber hinaus werden auch die Eigenschaften der Klassifikationsverfahren aufgeführt. Ein besonderes Augenmerk wird denjenigen Eigenschaften zuteil, welche sich auf die zusätzliche Aufnahme der Nächsten Nachbarn, bzw. deren Summen auswirken können.

Um die Reproduzierbarkeit zu gewährleisten wird zudem auf die konkrete Umsetzung der vorgestellten Verfahren eingegangen. Es werden sowohl die verwendeten R-Pakete, als auch die genutzen Funktionen aufgeführt. Zudem wird auf wichtige Änderungen der Defaulteinstellungen hingewiesen.

3.1 Nächste Nachbarn

Das von Fix und Hodges (1951, 1952) entwickelte Nächste Nachbarn Verfahren hat sich als eines der populärsten Klassifikationsmethoden etabliert. Trotz seiner relativ simplen Funktionsweise gehört es in vielen Klassifikationsproblemen zu den erfolgreichsten Verfahren (Friedman (1994) Seite 1). Die nachfolgend beschriebene Theorie dieses nicht-parametrischen Verfahrens ist in Agrawala (1977), Dasarathy (1991) und in der Arbeit von Silverman und Jones (1989) in etwas ausführlicherer Fassung nachzulesen.

3.1.1 k-Nächste Nachbarn

Um eine neue Beobachtung $\widetilde{\mathbf{x}} \in \mathbb{R}^p$ zu klassifizieren, betrachtet man die Klassenzugehörigkeiten der k nächstgelegenen Punkte aus den Lerndaten $(y_i, \mathbf{x}_i), \quad i = 1, \ldots, n_L$ (sog. Nächsten Nachbarn) und entscheidet sich für die Klasse, welche am häufigsten unter den k-Nächsten Nachbarn vertreten ist. Hierfür müssen zuallererst die Nächsten Nachbarn bestimmt werden. Man ordne die Daten

$$(y_{(1)}, \mathbf{x}_{(1)}), \ldots, (y_{(n_L)}, \mathbf{x}_{(n_L)})$$

anhand von zunehmenden Distanzen $d(\widetilde{\mathbf{x}}, \mathbf{x})$ hinsichtlich der zu klassifizierenden Beobachtung $\widetilde{\mathbf{x}}$. Sollten identische Distanzen – sogenannte Bindungen – auftreten so werden Beobachtungen mit niedrigerem Index als näher definiert. Alternative Möglichkeiten der Bindungsbrechung wurden in Abschnitt 2.2.3 vorgestellt. Beim k-Nächste Nachbarn Verfahren *knn* ordnet man die neue Beobachtung $\widetilde{\mathbf{x}} \in \mathbb{R}^p$ der Klasse zu, die am häufigsten unter den k nächsten Nachbarn

$$\left(y_{(1)}(\widetilde{\mathbf{x}}), \mathbf{x}_{(1)}(\widetilde{\mathbf{x}})\right), \ldots, \left(y_{(k)}(\widetilde{\mathbf{x}}), \mathbf{x}_{(k)}(\widetilde{\mathbf{x}})\right)$$

vertreten ist (Mehrheitsentscheid).

$$\delta_{\text{knn}}(\widetilde{\mathbf{x}}) = r \Leftrightarrow \text{Klasse r tritt am häufisten in } \{y_{(1)}(\widetilde{\mathbf{x}}), \ldots, y_{(k)}(\widetilde{\mathbf{x}})\} \text{ auf}$$

Sollten mehrere Klassen gleich häufig auftreten, so entscheidet man sich zufällig für eine dieser Klassen.

Verwendet man hingegen das Nächste Nachbar Verfahren (*nn1*), so ist ausschließlich die Klassenzugehörigkeit des nächsten Nachbar $\left(y_{(1)}(\widetilde{\mathbf{x}}), \mathbf{x}_{(1)}(\widetilde{\mathbf{x}})\right)$ mit der geringsten Distanz

$$d(\widetilde{\mathbf{x}}, \mathbf{x}_{(1)}) = \min_{i=1,\ldots,n_L} (d(\widetilde{\mathbf{x}}, \mathbf{x}_i))$$

zur neuen Beobachtung für die Vorhersage von Bedeutung. Es wird die Klasse des Nächsten Nachbarn ermittelt und die Zuordnung erfolgt dann in die Klasse, welcher auch der Nächste Nachbar angehört.

$$\delta_{\text{nn1}}(\widetilde{\mathbf{x}}) = y_{(1)}(\widetilde{\mathbf{x}})$$

Für gewöhnlich wird für die Distanz $d(.,.)$ die euklidische Distanzmetrik verwendet, allerdings muss es sich hierbei nicht um die beste Metrik handeln. Denn je nach Datenlage können andere Metriken zu besseren Klassifikationsergebnissen führen. Weitere gängige Metriken sind in Abschnitt 2.2.1 aufgeführt. Generell ist zu berücksichtigen, dass vor der Distanzberechnung – mit Ausnahme der Mahalanobis Distanz – alle Kovariablen standardisiert werden müssen, sodass nicht einzelne Variablen einen dominierenden Einfluss besitzen. Letzten Endes hat die Wahl der Metrik einen großen Einfluss auf die Güte des Verfahrens (Ripley (1996) Seite 191). Die Auswirkungen der gewählten Metrik lassen sich ganz gut am Spezialfall $k = 1$ veranschaulichen. Für $k = 1$ wird eine Voronoi-Zerlegung des angegebenen Raumes anhand von sogenannten Zentren durchgeführt (vgl. Aurenhammer und Klein (1996)). Jede Region enthält exakt ein Zentrum – dieses entspricht einer Beobachtung aus den Lerndaten – zu welchem die Punkte der zugehörigen Region die kürzeste Distanz aufweisen. Die Auswirkungen der unterschiedlichen Metriken ist in Abbildung 3.1 dargestellt. Je nachdem für welche Metrik man sich entscheidet, wird die Klassifizierung der neuen Beobachtung (als $\mathbf{x}$ dargestellt) von einem anderen Punkt aus den Lerndaten beeinflusst.

Darüber hinaus besitzt auch die Wahl der Anzahl an genutzten Nachbarn eine wichtige Rolle in Bezug auf die Güte des Verfahrens. Diesbezüglich konnten Cover und Hart (1967) nachweisen, dass die Nächste Nachbarn Regel *nn1* asymptotisch gesehen maximal doppelt so schlecht abschneidet wie die opti-

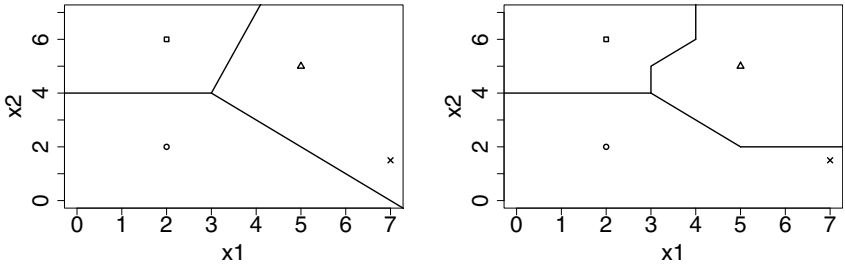

Abb. 3.1: Voronoi-Diagramm mit euklidischer Metrik (links) und Manhattan Metrik (rechts). Die drei ausgesparten Formen repräsentieren die Lerndaten und das Kreuz steht für eine neue zu klassifizierende Beobachtung $\widetilde{\mathbf{x}}$. Mittels der Linien wird veranschaulicht, welche Beobachtung aus den Lerndaten für die Klassenzuordnung relevant ist.

male Bayes-Regel. Wählt man k größer, so kann dies in vielen Fällen die Klassifikationsgüte verbessern. Grundsätzlich reduziert eine erhöhte Anzahl an betrachteten Nachbarn die Variabilität des Klassifikators, allerdings geht dies mit einer Eröhung des Bias einher. Demzufolge gilt es eine gute Balance zwischen Variabilität und Bias zu finden. Bei zu klein gewähltem k besitzt das Verfahren eine hohe Sensitivität gegenüber Ausreißern, was besonders in Grenzbereichen mit Überschneidungen mehrerer Klassen zu Fehlklassifikationen führt. Ein zu großes k hingegen verwendet voraussichtlich auch Beobachtungen aus anderen Clustern, was eine Klassifikation ebenso verfälscht. Im Extremfall $k = n_L$ wird sogar grundsätzlich die am häufigsten vertretene Klasse prognostiziert. Dementsprechend ist eine angemessene Wahl von k entscheidend um gute Ergebnisse zu erhalten. Eine Möglichkeit zur Optimierung dieses Hyperparameters bietet die Anwendung einer Kreuzvalidierung. Zur besseren Veranschaulichung sind die unterschiedlichen Fälle in Abbildung 3.2 dargestellt.

Ein weiterer wichtiger Faktor, der sich stark auf die Güte des Verfahrens auswirkt, ist die Auswahl der verwendeten Variablen. In diversen Klassifikationsproblemen stehen dem Nutzer eine Vielzahl an Variablen zur Verfügung. Nimmt man jedoch Variablen in das Modell auf, welche keine relevanten Informationen enthalten oder vergisst man wichtige Variablen, so verschlechtern sich die Klassifikationsergebnisse merklich (vgl. Paik und Yang (2004) Seite 2). Vor diesem Hintergrund ist zu erwähnen, dass bei den k-Nächste Nachbarn Verfahren der in Anhang A vorgestellte Fluch der Dimensionen besonders stark zuschlägt. Einen Ansatz um dieses Problem zu umgehen haben unter anderem Paik und Yang (2004) ausgearbeitet.

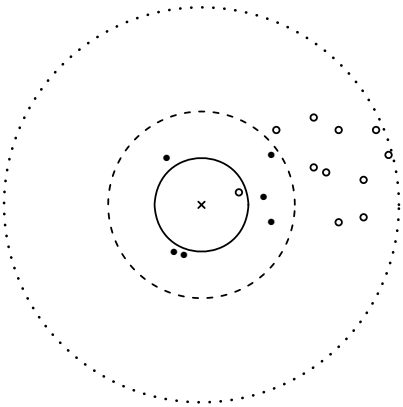

Abb. 3.2: Wahl des Parameters k. Relevante Nächste Nachbarn für den zu klassifizierenden Punkt $\mathbf{x}$. $k = 1$ (——) $k = 7$ (- - -) $k = 17$ ($\cdots$)

(Abbildung nach Aßfalg et al. (2003), S.89)

Algorithmus 2 k-Nächste Nachbarn

1. Standardisiere alle Kovariablen.
2. Berechne die Distanzen zwischen der neuen Beobachtung $\widetilde{\mathbf{x}}$ und jeder Beobachtung aus den Lerndaten
$$d(\widetilde{\mathbf{x}}, \mathbf{x}_{(i)}), \quad \text{für } i = 1, \ldots, n_L$$
mit Hilfe einer geeigneten Distanzmetrik $d(\cdot,\cdot)$.
3. Ordne die Distanzen der Größe nach an:
$$\left(y_{(1)}(\widetilde{\mathbf{x}}), \mathbf{x}_{(1)}(\widetilde{\mathbf{x}})\right), \ldots, \left(y_{(k)}(\widetilde{\mathbf{x}}), \mathbf{x}_{(k)}(\widetilde{\mathbf{x}})\right)$$
4. Klassifiziere $\widetilde{\mathbf{x}}$ basierend auf einem Mehrheitsentscheid der Klassenzugehörigkeit der k-Nächsten Nachbarn.
$$\delta_{\text{knn}}(\widetilde{\mathbf{x}}) = r \Leftrightarrow \text{Klasse r tritt am häufisten in } \{y_{(1)}(\widetilde{\mathbf{x}}), \ldots, y_{(k)}(\widetilde{\mathbf{x}})\} \text{ auf}$$

Eigenschaften

Alles in allem verfügt der k-Nächste Nachbarn Ansatz über diverse Vorteile. Es handelt sich um ein einfach anzuwendendes Verfahren, welches gegenüber Ausreißern weitestgehend robust ist. Dies gilt zwar nicht für das *nn1* Verfahren, allerdings liefert es trotz seiner Einfachheit in einigen Ansätzen erstaunlich gute Ergebnisse. Darüber hinaus wird kein Wissen über die Verteilung benötigt. Allerdings

hat das Verfahren auch eine Schattenseite. Es müssen genug Lerndaten vorhanden sein, um effektive Ergebnisse liefern zu können (siehe Parvin et al. (2008) Seite 1). Als weitere Nachteile sind der extrem hohe Berechnungs- sowie Speicheraufwand, welche mit steigender Anzahl an Lerndaten zunehmen, aufzuführen. Des Weiteren muss zusätzlich zur Distanzmetrik und der verwendeten Kovariablen die Anzahl der Nächsten Nachbarn sinnvoll gewählt werden. Und zu guter Letzt muss beachtet werden, dass die Klassen in den Lerndaten ungefähr gleich häufig auftreten, da das Verfahren ansonsten bei überschneidenden Klassengrenzen vorrangig die stärker vertretene Klasse prognostiziert.

Umsetzung

Die Umsetzung erfolgte mit Hilfe des R-Paketes *class* von Brian und Venables (2013). Die enthaltene Funktion *knn()* wurde für beide Ansätze verwendet. Das Hyperparametertuning für die Wahl des Parameters *k* bei *knn*, wurde durch eine 5-fache Kreuzvalidierung mit der Devianz als Schadensfunktion verwirklicht, um die optimale Anzahl an relevanten Nächsten Nachbarn in jedem Simulationsdurchlauf neu zu bestimmen.

Auswertung

Angewendet auf die Daten schneidet *nn1* (Missklassifikationsrate: Median = 0.151) eher schlecht ab (vgl. Abbildung 3.3). Die Verwendung mehrerer Nachbarn (*knn*) bewirkt eine deutliche Verbesserung der Ergebnisse, was vermutlich auf Unsicherheiten im Grenzgebiet zurückzuführen ist, welche bei *nn1* herrschen.

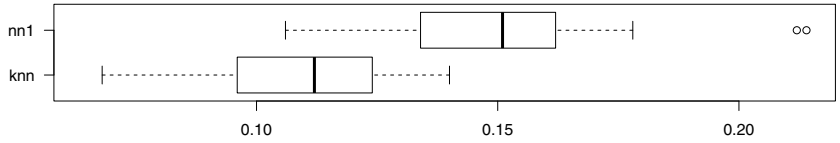

Abb. 3.3: Missklassifikationsraten von *nn1* und *knn*.

3.1.2 Gewichteter k-Nächste Nachbarn Algorithmus

Bei dem gewichteten k-Nächste Nachbarn Algorithmus handelt es sich um einen Ansatz, der die Auswirkung der Wahl von k etwas abschwächen soll. Diesem Algorithmus liegt die Idee zu Grunde, dass mit steigender Distanz des Nachbarn zur neuen Beobachtung $\widetilde{\mathbf{x}}$ der Beitrag zur Klassifikation geringer ausfällt. Um dies umzusetzen verwendet man Kernfunktionen $K(\cdot)$, welche die Eigenschaft besitzen, dass sie bei einer Distanz von 0 ihren maximalen Wert annehmen, der mit zunehmender Distanz d abnimmt. Im nachfolgenden Algorithmus werden der Dreieckskern

$$K(d) = (1 - |d|) \cdot I(|d| \leq 1)$$

als auch der Gaußkern

$$K(d) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{d^2}{2}\right)$$

verwendet. Weitere Kernfunktionen und ein etwas abgewandelter Algorithmus wurden im R-Paket *kknn* von Schliep und Hechenbichler (2013) umgesetzt. Ausführlichere Informationen hierzu finden sich in der Arbeit von Hechenbichler und Schliep (2004).

Zuallererst müssen wieder die Distanzen

$$d(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}})), \quad \text{für } j = 1, \ldots, k$$

der neuen Beobachtung $\widetilde{\mathbf{x}}$ zu den k Nächsten Nachbarn ermittelt werden. Diese Distanzen werden daraufhin standardisiert um die Stärke der Gewichtung beeinflussen zu können. Hierfür bieten sich zwei unterschiedliche Ansätze an.

$$D(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}})) = \frac{d(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}}))}{\gamma} \tag{3.1}$$

$$D(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}})) = \frac{d(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}}))}{d(\widetilde{\mathbf{x}}, \mathbf{x}_{(k)}(\widetilde{\mathbf{x}}))} \cdot \tau \tag{3.2}$$

Der erste Ansatz standardisiert alle Distanzen mit dem selben Wert. Ansatz (3.2) hingegen standardisiert die Distanzen mit Hilfe der Distanz zum k-ten Nächsten Nachbarn. Dadurch beschränkt man den Wertebereich vorerst auf das Intervall $[0,1]$, welches durch den Hyperparameter τ wieder vergrößert werden kann. Dieser Ansatz bietet sich mit dem Zusatz $\tau = 1$ zum Beispiel bei der Verwendung des Dreieckkerns an, da dieser ausschließlich im Bereich $[0,1]$ echt positive Werte liefert. Die standardisierten Distanzen werden an die gewählte Kernfunk-

tion übergeben um die einzelnen Gewichte

$$w(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}})) = \frac{K\left(D(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}}))\right)}{\sum_{l=1}^{k} K\left(D(\widetilde{\mathbf{x}}, \mathbf{x}_{(l)}(\widetilde{\mathbf{x}}))\right)}$$

der k-Nächsten Nachbarn zu bestimmen. Die Klassenzugehörigkeiten der k-Nächsten Nachbarn gehen mit der ermittelten Gewichtung in die Prognose ein.

$$\delta_{\text{wknn}}(\widetilde{\mathbf{x}}) = 1 \Leftrightarrow \hat{\pi}(\widetilde{\mathbf{x}}) = \sum_{j=1}^{k} w(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}}))y_{(j)}(\widetilde{\mathbf{x}}) \geq 0.5$$

Eine Zuordnung in Klasse 1 wird vorgenommen, wenn die geschätzte Wahrscheinlichkeit $\hat{\pi}(\widetilde{\mathbf{x}}) = \hat{P}(Y = 1|\widetilde{\mathbf{x}})$ für eine Klassenzugehörigkeit zur Klasse 1 über 50% beträgt. Alternativ kann die Entscheidungsregel auch für transformierte Daten

$$y_i^* = 2y_i - 1 \Leftrightarrow y_i = (y_i^* + 1)/2, \quad y_i^* \in \{-1, 1\}$$

aufgestellt werden. Diese lautet

$$\delta_{\text{wknn}}(\widetilde{\mathbf{x}}) = 1 \Leftrightarrow 2\hat{\pi}(\widetilde{\mathbf{x}}) - 1 = \sum_{j=1}^{k} w(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}}))y_{(j)}^*(\widetilde{\mathbf{x}}) \geq 0.$$

Eigenschaften

Der größte Vorteil des gewichteten k-Nächste Nachbarn Algorithmus ist die abgeschwächte Auswirkung der Wahl des Hyperparameters k. Selbst wenn die Anzahl an Nächsten Nachbarn groß gewählt wird, so haben weiter entfernte Nachbarn je nach Wahl von k und der Kernfunktion eine deutlich geringere Gewichtung als Nachbarn, welche sich in direkter Nachbarschaft zum neuen Beobachtungspunkt befinden. Demzufolge wählt man k lieber etwas zu groß, als zu klein. Des Weiteren sind sowohl der k-Nächste Nachbar Klassifikator – bei Wahl eines Rechteckkerns – als auch die Nächste Nachbar Regel als Spezialfälle im gewichteten k-Nächste Nachbarn Algorithmus enthalten. Allerdings kann der gewichtete k-Nächste Nachbarn Algorithmus im Gegensatz zum k-Nächste Nachbar Klassifikator ausschließlich auf binäre Zielvariablen angewendet werden.

Umsetzung

Die gewichteten k-Nächste Nachbarn Ansätze wurden mit Hilfe einer selbst geschriebenen Prozedur verwirklicht. Zum Vergleich wurde jedoch auch die Funktion *kknn()* aus dem gleichnahmigen Paket von (Schliep und Hechenbichler (2013)) verwendet. Dieses Vergleichsverfahren wird nachfolgend als "wknn" bezeichnet. Ein Hyperparametertuning hinsichtlich der Anzahl an verwendeten Nachbarn hat bei keinem der Verfahren stattgefunden. Der Defaultwert von *kknn()* fällt mit $k = 7$ relativ klein aus, weshalb dieser auf 25 erhöht wurde. Die selbst programmierten Verfahren werden hingegen sowohl mit 25 als auch 50 Nächsten Nachbarn vorgestellt. Das einzig durchgeführte Hyperparametertuning bezieht sich auf die beiden Gewichtungsparameter γ und τ.

Auswertung

Nachfolgend wird das soeben vorgestellte Verfahren in unterschiedlichen Konfigurationen auf die mlbench Daten angewendet. Anhand der Namensgebung der einzelnen Klassifikatoren lassen sich die gewählten Einstellungen ablesen.

wknn.Gewichtung.k.Kern.Parameter

Das Präfix "wknn" deutet an, dass es sich um ein Verfahren handelt, welches auf den gewichteten Nächsten Nachbarn basiert. Anhand von "Gewichtung" lässt sich ablesen, ob die Gewichtung (3.1) oder (3.2) angewendet wurde. "k" hingegen gibt an, wie viele Nächste Nachbarn einen Einfluss besitzen. Nachfolgend werden 25 oder 50 Nachbarn verwendet. Als "Kern" stehen dem Nutzer der Dreieckskern (t = triangular) und der Gaußkern (g = gaussian) zur Wahl. Zu guter Letzt gibt "Parameter" je nach Wahl des Gewichtungsverfahrens den Wert von τ bzw. γ an. Sollte dieser Parameter per Hyperparametertuning bestimmt worden sein, so ist dies durch ein "opt" gekennzeichnet.

In Abbildung 3.4 kann man erkennen, dass Verfahren basierend auf der Gewichtung (3.1) etwas schlechter abschneiden als diejenigen der alternativen Gewichtung (3.2). Hierbei erlangt man durch die Verdoppelung der betrachteten Nächsten Nachbarn Anzahl von 25 auf 50 eine deutliche Verbesserung. Bei Verwendung der zweiten Gewichtung spielt die Anzahl der verwendeten Nachbarn eine untergeordnete Rolle, solange eine gewisse Untergrenze nicht unterschritten wird. *wknn.2.25.g.1* und *wknn.2.50.g.1*, welche auf der alternativen Gewichtung sowie Verwendung des Gaußkernes und einem festen Gewichtungsparameter basieren, schneiden genauso gut ab wie das ebenfalls ungetunte *wknn*. Er-

Algorithmus 3 Gewichtete k-Nächste Nachbarn

1. Standardisiere alle Kovariablen.
2. Berechne die Distanzen zwischen der neuen Beobachtung $\widetilde{\mathbf{x}}$ und jeder Beobachtung aus den Lerndaten
$$d(\widetilde{\mathbf{x}}, \mathbf{x}_{(i)}), \quad \text{für } i = 1, \ldots, n_L$$
mit Hilfe einer geeigneten Distanzmetrik $d(\cdot, \cdot)$.
3. Ordne die Distanzen der Größe nach an:
$$\left(y_{(1)}(\widetilde{\mathbf{x}}), \mathbf{x}_{(1)}(\widetilde{\mathbf{x}})\right), \ldots, \left(y_{(k)}(\widetilde{\mathbf{x}}), \mathbf{x}_{(k)}(\widetilde{\mathbf{x}})\right)$$
4. Standardisiere die Distanzen anhand einer der nachfolgenden Ansätze:
$$D(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}})) = \frac{d(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}}))}{\gamma}, \quad \text{für } j = 1, \ldots, k$$
$$D(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}})) = \frac{d(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}}))}{d(\widetilde{\mathbf{x}}, \mathbf{x}_{(k)}(\widetilde{\mathbf{x}}))} \cdot \tau, \quad \text{für } j = 1, \ldots, k$$
5. Berechne die Gewichte der k Nächsten Nachbarn
$$w(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}})) = \frac{K\left(D(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}}))\right)}{\sum_{l=1}^{k} K\left(D(\widetilde{\mathbf{x}}, \mathbf{x}_{(l)}(\widetilde{\mathbf{x}}))\right)}, \quad \text{für } j = 1, \ldots, k$$
wobei $K(\cdot)$ eine geeignete Kernfunktion darstellt.
6. Klassifiziere $\widetilde{\mathbf{x}}$ basierend auf
$$\delta_{\text{wknn}}(\widetilde{\mathbf{x}}) = 1 \Leftrightarrow \hat{\pi}(\widetilde{\mathbf{x}}) = \sum_{j=1}^{k} w(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}})) y_{(j)}(\widetilde{\mathbf{x}}) \geq 0.5$$
Bei alternativer Kodierung $y^* \in \{-1, 1\}$ sieht die Entscheidungsregel folgendermaßen aus:
$$\delta_{\text{wknn}}(\widetilde{\mathbf{x}}) = 1 \Leftrightarrow 2\hat{\pi}(\widetilde{\mathbf{x}}) - 1 = \sum_{j=1}^{k} w(\widetilde{\mathbf{x}}, \mathbf{x}_{(j)}(\widetilde{\mathbf{x}})) y^*_{(j)}(\widetilde{\mathbf{x}}) \geq 0.$$

staunlicherweise führt die Optimierung des Gewichtungsparameters zu schlechteren Ergebnissen. Die Verwendung des Dreieckskerns anstatt des Gaußkerns hat bei festem Gewichtungsparameter vergleichbare Missklassifikationsraten zur Folge. Beim Dreieckskern wurde hinsichtlich der Gewichtung absichtlich kein Hyperparametertuning durchgeführt, da bei der Wahl von $\tau = 1$ auf Grund der Eigenschaften des Dreieckskerns der k-te Nachbar sich gerade an der Grenze des Bereiches befindet, an der einer Beobachtung ein echt positives Gewicht zuteil wird.

http://www.springer.com/978-3-658-11475-6

Verbesserung von Klassifikationsverfahren
Informationsgehalt der k-Nächsten-Nachbarn nutzen
Koch, D.
2016, XXII, 224 S. 278 Abb., Softcover
ISBN: 978-3-658-11475-6