LEHRBUCH

Stefan Richter

Statistisches und maschinelles Lernen

Gängige Verfahren im Überblick

Springer Spektrum

Statistisches und maschinelles Lernen

Stefan Richter

Statistisches und maschinelles Lernen

Gängige Verfahren im Überblick

Springer Spektrum

Stefan Richter
Institut für Angewandte Mathematik
Heidelberg University, Heidelberg
Baden-Württemberg, Deutschland

Ergänzendes Onlinematerial in der Programmiersprache R finden Sie auf https://www.springer.com/de/book/9783662593530.

ISBN 978-3-662-59353-0 ISBN 978-3-662-59354-7 (eBook)
https://doi.org/10.1007/978-3-662-59354-7

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Springer Spektrum

Planung/Lektorat: Iris Ruhmann

Springer Spektrum ist ein Imprint der eingetragenen Gesellschaft Springer-Verlag GmbH, DE und ist ein Teil von Springer Nature.
Die Anschrift der Gesellschaft ist: Heidelberger Platz 3, 14197 Berlin, Germany

Vorwort

Dieses Buch gibt einen Überblick über die bekanntesten Verfahren des maschinellen Lernens und betrachtet diese aus der Perspektive der mathematischen Statistik. Damit soll dem Leser ein Übergang zu der sehr weit verzweigten, spezialisierten Fachliteratur gegeben werden.

Viele Algorithmen des *maschinellen Lernens* wurden bereits vor einigen Jahrzehnten entwickelt; deren Popularität ist aber eng mit der Entwicklung entsprechend leistungsfähiger Computer verbunden. Die meisten Verfahren eint die Ausgangssituation, dass hochdimensionale Eingangsdaten vorliegen und daraus Strukturen erkannt (Beispiel: Komprimierung von Bildern) oder Vorhersagen für eine weitere Größe erhalten werden sollen (Beispiel: Lebenszufriedenheit von Personen in Abhängigkeit von Alter, Arztbesuche pro Woche, Körpergröße, …). Oft ist dabei nicht jede Komponente der Eingangsdaten relevant. Um zufriedenstellende Resultate zu liefern, müssen die Verfahren daher ohne weitere Anleitung in der Lage sein, unwichtige Komponenten zu verwerfen. Die Entwicklung erfolgte meist zunächst aus der Praxis heraus anhand konkreter Beispiele. In dieser ersten Phase erfolgt naturgemäß nur eine unzureichende theoretische Betrachtung. Das *statistische Lernen* hat zum Ziel, die wesentlichen Ideen der Verfahren aufzugreifen und diese aus statistischer Sicht zu analysieren und zu verbessern. Ganz konkret setzen wir uns mit der Konvergenzrate der Algorithmen in Abhängigkeit von der Anzahl der Trainingsdaten n und der Dimension d der Eingangsdaten auseinander. Dies ermöglicht einerseits eine genaue Analyse der Leistungsfähigkeit der Verfahren bei hochdimensionalen Eingangsdaten; andererseits sind die Ergebnisse auch ohne großes Vorwissen und ohne Verständnis der Beweise gut zu erfassen.

Wir behandeln die folgenden Teilbereiche des maschinellen Lernens:

- Supervised Learning
- Reinforcement Learning
- Unsupervised Learning

Der erste Teil des Buches beschäftigt sich mit Supervised und Reinforcement Learning (das hier didaktisch aus dem Supervised Learning hervorgeht), der zweite, kürzere Teil

behandelt das Unsupervised Learning. Bei dem Aufbau des Buches wurde darauf geachtet, die einzelnen Kapitel weitestgehend unabhängig voneinander behandeln zu können. Eine Ausnahme bilden die Kap. 1 und 3, welche grundlegende Notationen und Ansätze für die jeweils nachfolgenden Kapitel aufzeigen.

Um einen möglichst breiten Überblick geben zu können, der auch interessante und neue Themenbereiche mit einschließt, ist eine Reduktion in der Detailtiefe nötig. Dieses Buch hat daher ausdrücklich nicht das Ziel, jedes Verfahren und dessen Weiterentwicklungen für speziellere Probleme bis ins Kleinste zu erkunden. Stattdessen soll der Leser nach der Lektüre des Buches in der Lage sein, für ein gegebenes Problem Lösungsansätze mit Verfahren des maschinellen Lernens vorschlagen zu können, und sich über deren Vor- und Nachteile sowie die zugrunde gelegten Modellannahmen bewusst sein. Für die Anwendung in der Praxis ist dann eventuell ein genaueres Studium des jeweiligen Verfahrens nötig. Diesem Duktus folgend werden für die Herleitung wichtige Sätze aus verschiedensten mathematischen Teilgebieten nur zitiert und ohne genauere Auseinandersetzung angewandt; komplette Beweise sind aufgrund der Länge der Argumentationen nur selten möglich; in diesem Fall verweisen wir auf die entsprechenden Fachartikel.

Ein Kapitel arbeitet für das jeweilig untersuchte Verfahren Folgendes heraus:

- die aus mathematischer Sicht vorliegende Modellannahme des Problems, d. h., welche Annahmen stellen wir an den „wahren“ Zusammenhang der Eingangs- und Ausgangsdaten,
- eine anschaulich motivierte und mathematische Formulierung des für das Verfahren zu lösenden Optimierungsproblems,
- Lösung der Optimierungsprobleme in der Praxis,
- bisher bekannte Resultate für die Konvergenzraten unter geeigneten Bedingungen.

Dafür führen wir jeweils die in der wissenschaftlichen Literatur vorherrschende Notation ein. Um den Leser nicht zu verwirren und dem Dozenten eine einfachere Handhabung zu ermöglichen, haben wir diese allerdings soweit wie möglich vereinheitlicht und systematisiert. Mitunter führen wir auch unkonventionelle neue Bezeichnungen ein, um teilweise nur wörtlich vorhandene Definitionen auf ein sicheres mathematisches Fundament zu stellen. Dies erleichtert dann auch eine Formalisierung der Konvergenzresultate. Soweit didaktisch sinnvoll, stellen wir auch Beweise für die Aussagen oder zumindest Beweisskizzen zur Verfügung. Explizit mathematische Teile oder Beweise, die für das Verständnis des restlichen Kapitels nicht zwingend notwendig sind, sind separiert in den Abschnitten „Theoretische Resultate“ zu finden.

Aus Gründen der besseren Lesbarkeit verwenden wir in diesem Buch überwiegend das generische Maskulinum. Wir meinen damit stets alle Geschlechter.

Allgemeine Informationen zur Didaktik

Das Buch richtet sich an Studierende der Mathematik höheren Semesters, die bereits Vorkenntnisse in Wahrscheinlichkeitstheorie besitzen. Es kann als Basis für eine vierstündige Vorlesung in einem Semester genutzt werden. Dazu wird empfohlen, die teilweise länglichen mathematischen Definitionen mittels Folien zu präsentieren, die in jedem Fall auch die Anwendungen der vorgestellten Verfahren auf Beispiele enthalten sollten.

Begleitend zum Buch ermöglichen wir die intensivere Auseinandersetzung durch Bereitstellung von elementar formulierten Quelltexten in der Programmiersprache „R“, welche häufig in der Statistik genutzt wird (https://www.springer.com/de/book/9783662593530). Natürlich können unsere Programme in puncto Geschwindigkeit nicht mit maschinennäheren Programmiersprachen wie „Python“ mithalten. Dafür ist es mit der elementaren Formulierung möglich, den Algorithmus als Gesamtes zu begreifen und ihn nicht durch unnötig komplizierte Quelltexte länger als eine *black box* zu betrachten. „R“ bietet außerdem den Vorteil, dass mit wenig Aufwand Visualisierungen der Ergebnisse erhalten werden können. Das erworbene Wissen kann mittels der sich am Ende der Kapitel befindenden Übungsaufgaben überprüft werden, wobei hier der Fokus vor allem auf der Anwendung und Implementierung der Verfahren liegen sollte.

Kapitelübersicht und Empfehlungen

Wir geben hier eine kurze Kapitelübersicht und einige Anmerkungen über die gegenseitigen Abhängigkeiten. Auch wenn die Kapitel weitestgehend unabhängig voneinander gestaltet sind, so enthalten gerade die Kap. 1 und 3 einige wesentliche Grundlagen, welche für das Verständnis der Ansätze und Notationen in allen weiteren Kapiteln von Bedeutung sind. Naturgemäß sind jedoch nicht *alle* Inhalte dieser Kapitel für *alle* vorgestellten Verfahren notwendig. Es wird daher empfohlen, diese nicht komplett zu lesen, sondern bestimmte Teile davon erst an den Stellen zu konsultieren, wo diese das erste Mal benötigt werden.

- In Kap. 1 werden die grundlegenden Notationen für das Unsupervised Learning sowie einige gängige Techniken in der Praxis eingeführt. Es wird empfohlen, zunächst die Abschn. 1.1, 1.2 und 1.3 zu behandeln und sich nur einen Überblick über Abschn. 1.4 zu verschaffen. Die restlichen Abschnitte sollten dort genutzt werden, wo die Verfahren die dargestellten Hilfsmittel benötigen.
- Kap. 2 behandelt einige Standardverfahren für lineare Modelle für Regression. Es ist davon auszugehen, dass Personen mit einer grundlegenden Statistikausbildung den KQ-Schätzer kennen. Daher sollte hier vor allem darauf Wert gelegt werden, dass nicht länger die Schätzung der Parameter, sondern der daraus abgeleitete Algorithmus im Vordergrund steht. Durch Einführung des Lasso-Schätzers soll deutlich werden,

unter welchen Annahmen auch bei hochdimensionalen Eingangsdaten die Konstruktion eines sinnvollen Verfahrens möglich ist. Zum Verständnis der in Kap. 2 eingeführten Algorithmen ist die ausführliche Auseinandersetzung mit der Modellannahme und der damit verbundenen Standardisierung der Daten im Abschn. 2.1.1 nicht notwendig.

- In Kap. 3 stellen wir die Grundlagen für die Behandlung von Klassifikationsproblemen zur Verfügung. Um auch hier den Fokus nicht zu sehr auf die Theorie zu legen, wird empfohlen, nur den Abschn. 3.1 und den Anfang von Abschn. 3.2 (ohne die Darstellung der Möglichkeiten 1 und 2) zu lesen und die weiteren Teile erst zu konsultieren, wenn dies im Buch ausgewiesen ist.
- In Kap. 4 werden einige Standardverfahren für lineare Modelle in Klassifikationsproblemen behandelt. Mit der Einführung der Support Vector Machine (SVM) ab Abschn. 4.3 wird hier das erste komplexere Modell aus dem Bereich des maschinellen Lernens behandelt.
- Das Kap. 5 zu nichtparametrischen Methoden ist als Übergangskapitel konzipiert. Es soll einerseits erste Methoden bereitstellen, die auch für nichtlineare Modellannahmen Ergebnisse liefern, andererseits aber auch den beschränkten Nutzen dieser Methoden bei hochdimensionalen Eingangsdaten deutlich machen. Als Motivation für nachfolgende Kapitel wird mit der Modellannahme des naiven Bayes-Klassifizierers eine erste Möglichkeit präsentiert, allgemeine (nichtlineare) Zusammenhänge zwischen Eingangs- und Ausgangsdaten zuzulassen und trotzdem gute Ergebnisse zu erhalten.
- In Kap. 6 werden Verfahren basierend auf sogenannten Regressions- und Klassifikationsbäumen eingeführt. Durch Bagging, Boosting und das Prinzip des Random Forests lernen wir allgemein formulierte Methoden zur Verbesserung dieser Verfahren kennen. Abschn. 6.2 ist hierbei theoretisch orientiert und stellt einen Zusammenhang der Konvergenzeigenschaften von Bäumen zu den nichtparametrischen Methoden aus Kap. 6 her. Er kann beim Lesen ausgelassen werden.
- Mit den neuronalen Netzwerken in Kap. 7 erreichen wir ein Gebiet des maschinellen Lernens aus der aktuellen Forschung. Wir geben eine didaktisch orientierte Motivation, stellen die Theorie auf ein mathematisch sicheres Fundament und geben am Ende des Kapitels einen Ausblick auf in der Praxis verwendete komplexere Strukturen.
- Das Kap. 8 gibt eine Einführung in das Teilgebiet des Reinforcement Learning. Die ersten drei Abschn. 8.1, 8.2 und 8.3 sind hierbei unabhängig vom bisher eingeführten Stoff im Buch. Erst im letzten Teil, Abschn. 8.4, wird eine Verbindung zu neuronalen Netzwerken und den Methoden des Supervised Learnings hergestellt.
- Die Kap. 9 und 10 geben eine Einführung in das Gebiet des Unsupervised Learning mittels der Techniken der Repräsentantenfindung und der Dimensionsreduktion. Die

besprochenen Techniken sind unabhängig von den übrigen Kapiteln des Buches. In Kap. 10 werden jedoch einige Ansätze verwendet, die auch bei der SVM (vgl. Kap. 4) genutzt werden.

Danksagung

Zunächst möchte ich Herrn Prof. Jens-Peter Kreiß und dem Institut für mathematische Stochastik an der Universität Braunschweig danken. Durch die mir angebotene Vertretungsstelle bekam ich die Möglichkeit, eine Vorlesung über das statistische und maschinelle Lernen zu konzipieren und umzusetzen. Wesentliche Teile der Rohfassung des Buches sind in dieser Zeit entstanden. Meiner Arbeitsgruppe in Heidelberg danke ich für das Korrekturlesen und die hilfreichen Diskussionen über die Struktur der einzelnen Kapitel, namentlich Christof Schötz, Nathawut Phandoidaen, Sandra Schluttenhofer, Sergio-Filipe Brenner-Miguel und Marilena Müller. Ein besonderer Dank gilt auch Johannes Vogt und Steffen Wolf, die mir bei den etwas angewandteren Teilen des Buches zu neuronalen Netzwerken und Reinforcement Learning geholfen haben, das Vorgehen in der Praxis korrekt abzubilden und mit der Theorie zu verbinden. Ich danke außerdem Iris Ruhmann, Veronika Rosteck und Anja Dochnal für die hervorragende und verständnisvolle Betreuung während des Entstehungsprozesses des Buchs.

Heidelberg Stefan Richter
12.06.2019

Inhaltsverzeichnis

Supervised Learning: Grundlagen

1

Inhaltsverzeichnis

1.1 Die Problemstellung des Supervised Learnings

Supervised Learning bzw. *Überwachtes Lernen* ist ein Teilgebiet des maschinellen Lernens. In Abschn. 1.1.1 geben wir zunächst einige typische Anwendungsbeispiele und leiten daraus eine mathematische Formulierung der Problemstellung ab.

1.1.1 Struktur der Daten

Mit Methoden des *Supervised Learning* können folgende Problemstellungen gelöst werden:

a) *Handschrifterkennung:* Bestimme die Postleitzahl des Zielorts eines Briefes aus einer von Hand geschriebenen Zahl,
b) *Aktienpreis:* Sage den Preis einer Aktie in 6 Monaten vorher,
c) *Spamerkennung:* Bestimme bei einer gerade empfangenen E-Mail, ob es sich um unerwünschte Werbung handelt oder nicht,
d) *Spracherkennung:* Ermittle den gesprochenen Buchstaben (oder ganze Wörter) in einer Tonaufnahme.

S. Richter, *Statistisches und maschinelles Lernen,*
https://doi.org/10.1007/978-3-662-59354-7_1

In obigen Beispielen liegt jeweils folgende Situation vor: Gegeben ist ein Objekt mit gewissen Eigenschaften X, und gesucht ist eine Zielgröße Y, im Einzelnen:

a) X ist ein Scan der Briefoberfläche, Y die gesuchte Postleitzahl,
b) X sind die gesamten bekannten Daten der Firma, anderer Aktien und allgemeiner ökonomischer Marker, Y ist der Preis der Aktie in 6 Monaten,
c) X ist die E-Mail, Y „Spam“ oder „kein Spam“,
d) X ist die Tonaufnahme, Y beschreibt den gesuchten Buchstaben.

Alle Algorithmen des *Supervised Learning* verlangen das Vorliegen einer solchen Problemstruktur. Für die Größen X und Y werden folgende Bezeichnungen verwendet:

- X heißt *Featurevektor* (oder *Input*), die einzelnen Komponenten heißen *Features.*
- Y heißt *Label* (oder *Output*).

In Anwendungsproblemen kann es sich bei X um Text, Bilder, Tonaufnahmen usw. handeln, d. h., X kann zunächst jede Art von Struktur besitzen. In vielen Fällen ist es jedoch möglich, durch eine Vorbearbeitung der Daten zu erreichen, dass X ein reellwertiger Vektor ist, d. h. $X \in \mathbb{R}^d$ mit geeignetem $d \in \mathbb{N}$ gilt:

- Bei Graustufenbildern können beispielsweise die Grauwerte der einzelnen Pixel in einem Vektor zusammengefasst werden. Ein 100×100-Pixel-Bild entspricht dann einem Vektor $B \in \mathbb{R}^{10.000}$.
- Text kann je nach Problemstellung durch reellwertige Kenngrößen, die für Y relevant sind, ersetzt werden. Im Kontext von Beispiel c) kann der Mailtext X umgewandelt werden zu einem Vektor $\tilde{X} = (a_1, \ldots, a_4)^T \in \mathbb{R}^4$, wobei a_i das Vorkommen der Wörter und Zeichen „!“, „gratis“, „,“, „Luxus“ zählt (natürlich können hier noch weitere Kenngrößen ergänzt werden).

Die Dimension d von X kann je nach Anwendungsproblem sehr groß sein.

Die Größe Y erfüllt in der Regel $Y \in \mathbb{R}$ oder $Y \in \{1, \ldots, K\}$, wobei $K \in \mathbb{N}$. Im ersten Fall spricht man von einem *Regressionsproblem* (vgl. Beispiel b)), im zweiten Fall von einem *Klassifikationsproblem* mit K Klassen (vgl. Beispiele a),c),d)).

Zur Wahrung einer möglichst allgemeinen Theorie bezeichnen wir die Räume, in welchen sich X und Y befinden, zunächst mit $\mathcal{X}$ bzw. $\mathcal{Y}$.

1.1.2 Ermittlung von Algorithmen

Ziel des *Supervised Learning* ist die Ermittlung eines Zusammenhangs zwischen X und Y, idealerweise ausgedrückt durch eine (deterministische) Funktion $Y = f^*(X)$ mit

$f^* : \mathcal{X} \to \mathcal{Y}$. Zur Bestimmung von f^* sind in der Regel eine bestimmte Anzahl $n \in \mathbb{N}$ von Beobachtungen für X und zugehörigem Y gegeben. Diese Beobachtungen werden mit (X_i, Y_i), $i = 1, \ldots, n$ bezeichnet und *Trainingsdaten* genannt.

Für die theoretische Betrachtung nehmen wir in diesem Buch an, dass jedes einzelne Trainingsdatum einer Wahrscheinlichkeitsverteilung $\mathbb{P}^{(X,Y)}$ entstammt. Das bedeutet, die Gesamtheit der (X_i, Y_i), $i = 1, \ldots, n$ ist repräsentativ für das gegebene Problem; man erwartet, dass bei erneuter Erzeugung von Beobachtungen (z. B., wenn der gefundene Zusammenhang in der Praxis verwendet werden soll) ähnliche Werte wie bei den bereits erhaltenen (X_i, Y_i) auftreten. Diese Annahme ist verletzt, wenn man eine willkürliche Auswahl der Trainingsdaten anhand bestimmter Eigenschaften getroffen hat. Auch eine systematische Beeinflussung der Messwerte während des Messvorgangs darf nicht auftreten.

Weiterhin gehen wir davon aus, dass die Trainingsdaten untereinander *stochastisch unabhängig* sind. Anschaulich bedeutet dies, dass sich die Werte der (X_i, Y_i) während der Sammlung nicht untereinander beeinflusst haben. Entsprechen die X_i z. B. mehreren über die Zeitpunkte $i = 1, \ldots, n$ gemessenen Größen, so muss gewährleistet sein, dass zeitlich nah beieinander liegende X_i nicht stärker übereinstimmen als zeitlich weit auseinander liegende X_i (Gleiches gilt für die Y_i).

In der Sprache der Wahrscheinlichkeitstheorie bedeuten beide Annahmen zusammen, dass $(X_i, Y_i) \overset{\text{iid}}{\sim} \mathbb{P}^{(X,Y)}$ unabhängig und identisch verteilt sind (kurz: i. i. d. für *independent and identically distributed*). Für die meisten der in diesem Buch vorgestellten Verfahren gibt es auch statistische Resultate für den Fall, dass diese Annahme verletzt ist; das Studium dieser Theorien geht jedoch über den Horizont dieser Einführung hinaus und wird nicht besprochen.

Eine exakte Angabe von f^* mittels der (X_i, Y_i), $i = 1, \ldots, n$ ist oft nicht möglich. Stattdessen versucht man, eine möglichst gute Schätzung von f^* zu finden. Eine solche Schätzung bezeichnen wir mit $\hat{f}_n : \mathcal{X} \to \mathcal{Y}$. Formal betrachtet ist $\hat{f}_n$ eine kompliziertere Abbildung: $\hat{f}_n = \hat{f}_n(X_1, Y_1, \ldots, X_n, Y_n)$ hängt von den gegebenen Trainingsdaten ab, d. h., für jede Ausprägung von (X_i, Y_i), $i = 1, \ldots, n$ liefert uns $\hat{f}_n$ eine Anleitung, wie aus den Beobachtungen eine Abbildung $\mathcal{X} \to \mathcal{Y}$ erhalten werden kann. Die Literatur gibt für $\hat{f}_n$ keine standardisierte Benennung; in diesem Buch verwenden wir für $\hat{f}_n$ den Begriff des *Algorithmus*. Damit entsteht eine Bezeichnungskollision mit dem namensgleichen Begriff aus der Informatik: Während uns eine mathematisch eindeutige Definition von $\hat{f}_n$ genügt, verlangt man dort, dass $\hat{f}_n$ in elementare Berechnungsschritte zerlegt werden kann.

Für die Bewertung von $\hat{f}_n$ gibt es zwei Qualitätskriterien:

- Interpretierbarkeit (qualitativ): Können die Ergebnisse von $\hat{f}_n$ genutzt werden, um ein tieferes Verständnis über den Zusammenhang zwischen X und Y zu erlangen?
- Vorhersagefähigkeit (quantitativ): Sagt $\hat{f}_n(X)$ für neue Beobachtungen X das richtige Y voraus bzw. wie gut ist diese Vorhersage? Die Messung der Qualität einer Abbildung $f : \mathcal{X} \to \mathcal{Y}$ erfolgt mittels einer *Verlustfunktion* L, welche den Abstand zwischen $f(X)$ und Y misst.

Zur Ermittlung geeigneter Algorithmen stellt man eine *Modellannahme* an den Zusammenhang f^* von X und Y. Dies geschieht beispielsweise durch Annahmen an die funktionale Struktur von f^*. Solch eine Annahme reduziert die Anzahl der möglichen Ausprägungen von f^* und ermöglicht damit eine systematische Ermittlung von $\hat{f}_n$ durch Maximierung der Vorhersagefähigkeit auf den Trainingsdaten. In diesem Buch stellen wir jeweils deutlich heraus, welche Modellannahme durch ein Verfahren getroffen wird. Es ist zu beachten, dass bei einer nicht der Realität entsprechenden Modellannahme das ermittelte $\hat{f}_n$ beliebig schlechte Resultate liefern kann; die gewissenhafte Auswahl ist daher von großer Wichtigkeit.

Über die letzten Jahrzehnte gab es eine Paradigmenverschiebung bei der Entwicklung von Algorithmen $\hat{f}_n$: Früher arbeiteten Experten sehr lange daran, bereits *vor* Ermittlung des Algorithmus die wesentlichen Komponenten von X für die Vorhersage von Y zu extrahieren (d. h., die sogenannte *feature extraction* wurde mithilfe von Expertenwissen durchgeführt). Bei der Modellannahme und der darauf basierenden Berechnung des Algorithmus konnte dann davon ausgegangen werden, dass alle Komponenten von X sehr wichtig für die Vorhersage von Y sind. Die Dimension des Vektors $X \in \mathbb{R}^d$ war nur moderat groß (oft $d < 20$). Die Anzahl der Trainingsdaten n war wesentlich größer als d (kurz: $n \gg d$).

Heutige Regressions- und Klassifikationsprobleme sind häufig hochdimensional, d. h., die gegebenen X sind Vektoren hoher Dimension. Die Vorbearbeitung von X mittels Expertenwissen ist entweder nicht mehr vorhanden oder mündet in einer nur unzureichenden Reduktion der Komponenten von X. Entsprechend ist nicht jede Komponente von X wichtig für die Vorhersage von Y. In Bezug auf die Anwendungen (a) und (c) oben ist beispielsweise nicht jedes Pixel des Scans der Briefoberfläche wichtig für die Vorhersage der Postleitzahl; genauso kann es sein, dass die Anzahl der Ausrufezeichen „! “ kein sinnvoller Indikator für eine Spammail ist. Ein Algorithmus aus dem Gebiet des maschinellen Lernens muss daher die für Y wichtigen Komponenten von X erkennen und dann darauf basierend Vorhersagen für Y treffen.

In Abschn. 1.2 formalisieren wir die hier eingeführten Begriffe und betten sie in die statistische Entscheidungstheorie ein.

1.2 Statistische Entscheidungstheorie

Seien $\mathcal{X}, \mathcal{Y}$ metrische Räume und $\mathcal{B}(\mathcal{X})$, $\mathcal{B}(\mathcal{Y})$ die zugehörigen Borel'schen σ-Algebren. Fordern wir im Folgenden *Messbarkeit* von Funktionen $f : \mathcal{X} \to \mathcal{Y}$, so meinen wir $\mathcal{B}(X)$-$\mathcal{B}(Y)$-Messbarkeit.

Sei $(\Omega, \mathcal{A}, \mathbb{P})$ ein Wahrscheinlichkeitsraum. Die Größen $X : \Omega \to \mathcal{X}$ und $Y : \Omega \to \mathcal{Y}$ seien messbare Abbildungen *(Zufallsvariablen).* Wir unterstellen damit, dass die Größen X, Y bei der Datensammlung zufällige Werte in $\mathcal{X}$ bzw. $\mathcal{Y}$ mit jeweils festgelegten, aber unbekannten Wahrscheinlichkeiten annehmen. Die dadurch resultierende Wahrscheinlichkeitsverteilung von (X, Y) auf $\mathcal{X} \times \mathcal{Y}$ entspricht der *induzierten Verteilung* $\mathbb{P}^{(X,Y)}$.

Die Modellierung von (X, Y) als zufällige Größe bedeutet, dass auch bei Beobachtung zweier Paare (X_1, Y_1) und (X_2, Y_2) mit gleichen Werten $X_1 = X_2$ möglicherweise zwei verschiedene $Y_1 \neq Y_2$ auftreten. Wir unterstellen also keinen vollständig deterministischen Zusammenhang der Form $Y = f^*(X)$ zwischen den Beobachtungen X und Y. Stattdessen erwarten wir, dass X, Y durch Umwelteinflüsse verrauschten Beobachtungen „wahrer" Werte $x \in \mathcal{X}$, $y \in \mathcal{Y}$ entsprechen, für welche eine deterministische Beziehung $y = f^*(x)$ gilt. Die nur in der Theorie existierende Wahrscheinlichkeitsverteilung $\mathbb{P}^{(X,Y)}$ enthält alle Informationen über mögliche Kombinationen von X, Y, ist aber in der Praxis unbekannt. Informationen über $\mathbb{P}^{(X,Y)}$ können jedoch aus Trainingsdaten abgeleitet werden.

Anschaulich steht (X, Y) für den Prototyp *eines* Trainingsdatums. Für die folgenden Definitionen nutzen wir (X, Y) auch als *neue* Beobachtung, die auftritt, nachdem ein Algorithmus $\hat{f}_n$ basierend auf Trainingsdaten festgelegt wurde. Die Qualität von $\hat{f}_n$ kann dann mittels (X, Y) bestimmt werden.

Wir definieren nun zunächst formal, welche grundlegenden Eigenschaften wir von einer Funktion $f : \mathcal{X} \to \mathcal{Y}$ erwarten und wie deren Qualität mittels einer Verlustfunktion gemessen werden kann:

Definition 1.1
Es bezeichne $\mathbb{E}Z$ den *Erwartungswert* einer Zufallsvariable Z.

- Eine *Entscheidungsregel* ist eine messbare Abbildung $f : \mathcal{X} \to \mathcal{Y}$.
- Eine *Verlustfunktion* ist eine messbare Abbildung $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$. Bei Anwendung der Entscheidungsregel f entsteht der *Verlust* $L(Y, f(X))$.
- Das *Risiko* von f ist $R(f) := \mathbb{E}L(Y, f(X))$. ♦

Während die Verlustfunktion zunächst nur das Abstandsmaß zwischen Vorhersage $f(X)$ und Y bereitstellt, drückt das Risiko den *im Mittel erwarteten Fehler* bei Nutzung von f aus. Daher ist uns vor allem an der Bestimmung von Entscheidungsregeln $f : \mathcal{X} \to \mathcal{Y}$ gelegen, für die $R(f)$ möglichst klein ist.

In diesem Buch nehmen wir nun stets $\mathcal{X} \subset \mathbb{R}^d$ und $\mathcal{Y} \subset \mathbb{R}$ (Regressionsproblem) oder $\mathcal{Y} = \{1, \ldots, K\}$ mit $K \in \mathbb{N}$, $K \geq 2$ (Klassifikationsproblem) an. Zwei häufig verwendete Verlustfunktionen sind:

Beispiel 1.2

- Falls $\mathcal{Y} \subset \mathbb{R}$, so heißt
$$L(y, s) = (y - s)^2$$
quadratische Verlustfunktion.

- Falls $\mathcal{Y} = \{1, \ldots, K\}$ mit $K \in \mathbb{N}$, $K \geq 2$, so heißt

$$L(y, s) = 1_{\{y \neq s\}} = \begin{cases} 1, & y \neq s, \\ 0, & y = s \end{cases}$$

0-1-Verlustfunktion. Die Funktion $1_{\cdot}$ heißt *Indikatorfunktion.*

Bemerkung Bei der 0-1-Verlustfunktion misst man nur, ob die richtige Klasse bestimmt wurde oder nicht. Je nach Problem kann diese aber genauer auf die Situation zugeschnitten definiert werden.

Auch wenn $\mathbb{P}^{(X,Y)}$ in der Praxis unbekannt ist, so kann die theoretische Bestimmung von $R(f)$ für eine beliebige Entscheidungsfunktion f weitere Hinweise auf einen geeigneten Algorithmus geben. Wir bestimmen $R(f)$ für den Fall der quadratischen und der 0-1-Verlustfunktion:

Lemma 1.3 (Theoretische Berechnung von $R(f)$ bei Kenntnis von $\mathbb{P}^{(X,Y)}$)

- Ist $\mathcal{Y} = \mathbb{R}$ und L die quadratische Verlustfunktion, so gilt

$$R(f) = \mathbb{E}L(Y, f(X)) = \mathbb{E}[(Y - f(X))^2] = \int_{\mathcal{X}} \int_{\mathcal{Y}} (y - f(x))^2 \, d\mathbb{P}^{Y|X=x}(y) \, d\mathbb{P}^X(x),$$

und $R(f)$ wird minimal für $f^*(x) = \mathbb{E}[Y|X = x]$. f^* bestimmt also den im Mittel erwarteten Wert für Y bei gegebener Realisierung x.

- Ist $\mathcal{Y} = \{1, \ldots, K\}$ und L die 0-1-Verlustfunktion, so gilt

$$R(f) = \mathbb{E}L(Y, f(X)) = \mathbb{P}(Y \neq f(X)) = \int_{\mathcal{X}} \mathbb{P}(Y \neq f(X)|X = x) \, d\mathbb{P}^X(x),$$

und $R(f)$ wird minimal für jede Funktion $f^* : \mathcal{X} \to \mathcal{Y}$ mit

$$f^*(x) \in \underset{k \in \{1,\ldots,K\}}{\arg\min} \, [1 - \mathbb{P}(Y = k|X = x)] = \underset{k \in \{1,\ldots,K\}}{\arg\max} \, \mathbb{P}(Y = k|X = x). \tag{1.1}$$

f^* wählt die Klasse aus, die bei gegebener Realisierung x am wahrscheinlichsten ist. f^* heißt auch *Bayes-Klassifizierer* oder *MAP-Klassifizierer,* wobei MAP eine Abkürzung für den englischen Begriff *maximum a posteriori* ist. f^* ist für diejenigen $x \in \mathcal{X}$, für welche mehrere $k \in \{1, \ldots, K\}$ Maximierer von Gl. (1.1) sind, nicht eindeutig bestimmt.

Die $R(f)$ minimierende Entscheidungsregel f^* hat eine herausgehobene Stellung und wird wie folgt bezeichnet:

Definition 1.4
Der Minimierer f^* von $R(f)$ heißt *Bayes-Regel* oder (im Klassifizierungsproblem) *Bayes-Klassifizierer.* Das zugehörige Risiko $R(f^*)$ heißt *Bayes-Risiko.* ♦

Genauso wie $\mathbb{P}^{(X,Y)}$ ist f^* in Anwendungen unbekannt, trotzdem ist das Ziel natürlich die Bestimmung einer Entscheidungsregel, die möglichst „nah“ an f^* liegt. Das Verfahren zur Ermittlung einer Entscheidungsregel basierend auf Trainingsdaten nennen wir Algorithmus:

Definition 1.5 (Algorithmus)
Beobachtungen $(X_i, Y_i) \overset{\text{iid}}{\sim} \mathbb{P}^{(X,Y)}$, $i = 1, \ldots, n$ heißen *Trainingsdaten.* Die Zusammenfassung aller Trainingsdaten bezeichnen wir mit $T_n := (X_i, Y_i)_{i=1,\ldots,n}$.

Eine messbare Abbildung $\hat{f}_n : \Omega \times \mathcal{X} \to \mathcal{Y}$ heißt *Algorithmus,* falls es eine messbare Abbildung $A : (\mathcal{X} \times \mathcal{Y})^n \times \mathcal{X} \to \mathcal{Y}$ gibt, so dass für alle $\omega \in \Omega$, $x \in \mathcal{X}$ gilt:

$$\hat{f}_n(\omega, x) = A(T_n(\omega), x) \tag{1.2}$$

Für jede Realisierung $\omega \in \Omega$ heißt $\hat{f}_n(\omega) : \mathcal{X} \to \mathcal{Y}$ die von $\hat{f}_n$ *induzierte Entscheidungsregel.* ♦

Bemerkung 1.6

- Die Darstellung in Gl. (1.2) bedeutet, dass die von $\hat{f}_n$ gelieferten Entscheidungsregeln nur auf Basis der vorliegenden Trainingsdaten bestimmt werden und keine weiteren zufälligen Informationen aus der Umwelt einfließen.
- Im Folgenden schreiben wir oft kurz $\hat{f}_n$ anstelle von $\hat{f}_n(\omega)$. Damit interpretieren wir einen Algorithmus als zufällige Entscheidungsregel, welche sich je nach vorliegenden Trainingsdaten ändert.

Die Qualität eines Algorithmus wird mittels des Generalisierungsfehlers gemessen:

Definition 1.7 (Generalisierungsfehler)
$\mathbb{E}R(\hat{f}_n)$ heißt *Generalisierungsfehler,* wobei

$$R(\hat{f}_n) := \mathbb{E}[L(Y, \hat{f}_n(X))|T_n]$$

mit $(X, Y) \sim \mathbb{P}^{(X,Y)}$ unabhängig von T_n. ♦

Bemerkung 1.8
Aus theoretischer Sicht findet die Bewertung der Qualität von $\hat{f}_n$ in zwei Schritten statt:

- Der bedingte Erwartungswert $R(\hat{f}_n) := \mathbb{E}[L(Y, \hat{f}_n(X))|T_n]$ berechnet das Risiko der von $\hat{f}_n$ induzierten Entscheidungsregel unter der Annahme, dass die zur Bewertung herangezogene Beobachtung (X, Y) unabhängig von den Trainingsdaten ist. $R(\hat{f}_n)$ führt aber keine Bewertung der Trainingsdaten durch, welche zur Ermittlung von $\hat{f}_n$ geführt haben. So kann es beispielsweise sein, dass durch „ungünstige“ Zufälle die Trainingsdaten T_n

nicht repräsentativ für den allgemeinen Zusammenhang zwischen X und Y waren. In diesem Fall ist $R(\hat{f}_n)$ groß, obwohl der Algorithmus $\hat{f}_n$ eventuell sehr gut ist.
- Erst der Generalisierungsfehler $\mathbb{E}R(\hat{f}_n)$ bewertet den Algorithmus $\hat{f}_n$, d. h. die *Vorschrift,* mit welcher die Entscheidungsregeln aus T_n gewonnen werden, indem das Risiko $R(\hat{f}_n)$ über alle Konfigurationen von theoretisch möglichen Trainingsdaten gemittelt wird.

Für theoretische Resultate und die Herleitung neuer Algorithmen sind daher vor allem $\mathbb{E}R(\hat{f}_n)$ oder statistische Aussagen über die Größe von $R(\hat{f}_n)$ von Interesse, wohingegen in der Praxis die Bestimmung von $R(\hat{f}_n)$ für die konkret erhaltene Entscheidungsregel $\hat{f}_n(\omega)$ im Vordergrund steht (vgl. Bemerkung 1.17).

Eine quantitative Bewertung von $\hat{f}_n$ ist in der statistischen Theorie mittels Vergleich zum Bayes-Risiko $R(f^*)$ möglich. In diesem Buch legen wir besonderen Wert auf den Einfluss der Dimension d auf die Qualität von $\hat{f}_n$, weshalb diese explizit in die folgende Definition einfließt:

Definition 1.9 (Excess Bayes Risk)
- $\mathbb{E}R(\hat{f}_n) - R(f^*) \geq 0$ heißt das *Excess Bayes Risk.* Der Ausdruck gibt an, wie stark das Bayes-Risiko durch den Generalisierungsfehler überschritten wird.
- Für jedes $d \in \mathbb{N}$ sei $(\psi_d(n))_{n\in\mathbb{N}}$ eine monoton fallende Folge. Ein Algorithmus *lernt im Durchschnitt mit Konvergenzrate* $\psi_d(n)$, falls eine von d, n unabhängige Konstante $C > 0$ existiert, sodass

$$\text{für alle } n \in \mathbb{N}: \quad \mathbb{E}R(\hat{f}_n) - R(f^*) \leq C \cdot \psi_d(n). \tag{1.3}$$

Ein Algorithmus *lernt mit hoher Wahrscheinlichkeit mit Konvergenzrate* $\psi_d(n)$, falls

$$\limsup_{c\to\infty} \limsup_{d,n\to\infty} \mathbb{P}\left(|R(\hat{f}_n) - R(f^*)| \geq c \cdot \psi_d(n)\right) = 0. \tag{1.4}$$

♦

In vielen Modellen ist ein Beweis von Gl. (1.3) nicht ohne Modifikation der in Abschn. 1.3 eingeführten Standardansätze zur Ermittlung von Algorithmen möglich. Wir werden daher oft nur Gl. (1.4) zeigen. Aussagen dieser Form benötigen üblicherweise noch eine weitere Einschränkung an das Verhalten von d im Vergleich zu n. Formal drücken wir dies aus, indem wir $d = d_n$ als von n abhängige Größe betrachten.

Sind Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$ gegeben, so scheint eine naheliegende Forderung an $\hat{f}_n$ zu sein:

$$\text{für alle } i \in \{1, \ldots, n\}: \quad L(\hat{f}_n(X_i), Y_i) = 0 \tag{1.5}$$

Für viele Verlustfunktionen ist dies gleichbedeutend mit

$$\text{für alle } i \in \{1, \dots, n\}: \quad \hat{f}_n(X_i) = Y_i,$$

d. h. der korrekten Wiedergabe der Trainingsdaten. Dieser Ansatz hat jedoch zwei Nachteile:

- Es wird verkannt, dass die X_i, Y_i durch zufällige Umwelteinflüsse verfälscht sind und einige Y_i gar nicht den „bestmöglichen" Antworten $f^*(X_i)$ entsprechen. Im schlimmsten Fall gibt es zwei Beobachtungen (X_{i_1}, Y_{i_1}) und (X_{i_2}, Y_{i_2}) mit $X_{i_1} = X_{i_2}$, aber $Y_{i_1} \neq Y_{i_2}$, so dass die Erfüllung von Gl. (1.5) gar nicht möglich ist. Ist Gl. (1.5) ganz oder zumindest für sehr viele Indizes i näherungsweise erfüllt, so sprechen wir von einer *Überanpassung* an die Trainingsdaten (engl. *overfitting*).
- Gl. (1.5) gibt keinerlei Information darüber, wie $\hat{f}_n(x)$ für $x \notin \{X_1, \dots, X_n\}$ zu wählen ist. Gerade in der Praxis soll $\hat{f}_n$ aber auch auf vorher nicht gesehene Fälle angewandt werden.

Zur Korrektur und Erweiterung des Ansatzes aus Gl. (1.5) ist eine Annahme an die Struktur von f^* nötig. Diese muss uns ermöglichen, den Wert von $f^*(x)$ auch für $x \in \mathcal{X}$ in einer Umgebung der Trainingsdaten $X_i, i = 1, \dots, n$ abschätzen zu können. Üblicherweise erfolgt die Formulierung einer Annahme an f^* durch Angabe einer Funktionenklasse:

Definition 1.10 (Modellannahme)
Ist $\mathcal{F} \subset \{f : \mathcal{X} \to \mathcal{Y} \text{ messbar}\}$ eine beliebige Menge von Funktionen, so nennen wir $f^* \in \mathcal{F}$ eine *Modellannahme* an f^*. ♦

Ein zugehöriger Algorithmus $\hat{f}_n$ wählt dann üblicherweise ebenfalls nur Elemente von $\mathcal{F}$ aus.

Oft ist die Modellannahme eine Vereinfachung des tatsächlichen Zusammenhangs zwischen X und Y und trifft in der Realität nicht zu (d. h. $f^* \notin \mathcal{F}$). In diesem Fall gilt folgende Zerlegung des Excess Bayes Risks:

$$\mathbb{E}R(\hat{f}_n) - R(f^*) = \underbrace{\left[\mathbb{E}R(\hat{f}_n) - \inf_{f \in \mathcal{F}} R(f)\right]}_{\text{Schätzfehler}} + \underbrace{\left[\inf_{f \in \mathcal{F}} R(f) - R(f^*)\right]}_{\text{Approximationsfehler}}$$

Der *Approximationsfehler* ist dann ein durch unsere Vereinfachung unvermeidbarer Fehler. Für die Bestimmung eines guten Algorithmus muss daher der *Schätzfehler* minimiert werden.

Das folgende einfache Regressionsproblem soll die Begriffe an einem Beispiel verdeutlichen:

Beispiel 1.11 Es seien $X : \Omega \to \mathcal{X} = \mathbb{R}$ und $Y : \Omega \to \mathcal{Y} = \mathbb{R}$, und es gelte („in der Realität")

$$Y = f_0(X) + \varepsilon$$

mit einer *unstetigen* Funktion $f_0 : \mathbb{R} \to \mathbb{R}$ und einem zufälligen Fehler ε. Es gelte $\mathbb{E}\varepsilon = 0$, und ε sei unabhängig von X.

Uns ist der wahre Zusammenhang unbekannt. Wir machen die *Modellannahme:*

$$Y = f_c(X) + \varepsilon$$

mit einer *stetigen* Funktion $f_c : \mathbb{R} \to \mathbb{R}$.

Wir verwenden die quadratische Verlustfunktion $L(y, s) = (y - s)^2$. Dann gilt:

$$f^*(x) = \mathbb{E}[Y|X = x] = f_0(x),$$

d. h., unsere Modellannahme lautet $f_0 = f^* \in \mathcal{F} := \{f : \mathbb{R} \to \mathbb{R} \text{ stetig}\}$ und ist offensichtlich verletzt. Das Bayes-Risiko ist

$$R(f^*) = \mathbb{E}L(Y, f^*(X)) = \mathbb{E}[\varepsilon^2].$$

Das Risiko für eine beliebige Entscheidungsfunktion $f \in \mathcal{F}$ lautet:

$$R(f) = \mathbb{E}\left[\left(\underbrace{Y}_{=f_0(X)+\varepsilon} - f(X)\right)^2\right] = \mathbb{E}\left[(f_0(X) - f(X))^2\right] + \mathbb{E}[\varepsilon^2],$$

d. h., der durch die Modellannahme verursachte *Approximationsfehler* lautet

$$\inf_{f\in\mathcal{F}} R(f) - R(f^*) = \inf_{f\in\mathcal{F}} \mathbb{E}\left[(f_0(X) - f(X))^2\right].$$

Für die Höhe des Approximationsfehlers kommt es also z. B. darauf an, wie häufig Beobachtungen X nahe der Unstetigkeitsstellen von f_0 (wo unsere Modellannahme die größten Fehler verursacht) auftreten.

Bei der Wahl der Modellannahme $\mathcal{F} \subset \{f : \mathcal{X} \to \mathcal{Y} \text{ messbar}\}$ gibt es ein natürliches Dilemma: Da $\hat{f}_n$ die bestmögliche Entscheidungsfunktion aus ganz $\mathcal{F}$ liefern soll, dafür aber nur n Trainingsdaten zur Verfügung stehen, ist die Qualität von $\hat{f}_n$ abhängig von der Größe von $\mathcal{F}$:

- Ist $\mathcal{F}$ groß, so ist der Approximationsfehler klein, aber der Schätzfehler groß.
- Ist $\mathcal{F}$ klein, so ist der Approximationsfehler groß, aber der Schätzfehler klein.

Bei vielen Modellen wird ein Element von $\mathcal{F}$ durch Angabe von Parametern festgelegt, d. h. $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$, wobei Θ ein Parameterraum ist. Im maschinellen Lernen gilt häufig $\Theta \subset \mathbb{R}^p$ mit Dimension $p \in \mathbb{N}$. Mit dem Begriff *Komplexität* von $\mathcal{F}$ verweist man lose auf den Wert von $\frac{p}{d}$ (wie viele Parameter werden zum Beschreiben einer Komponente genutzt?) oder darauf, wie „weit" $\mathcal{F}$ noch von $\{f : \mathcal{X} \to \mathcal{Y}$ messbar$\}$ entfernt ist.

Das Studium des Approximationsfehlers ist oft eine rein analytische Aufgabe und steht bei uns nicht im Vordergrund. Wir werden daher oft annehmen, dass

$$f^* \in \mathcal{F} \quad \text{und damit} \quad \inf_{f \in \mathcal{F}} R(f) - R(f^*) = 0.$$

Für jedes in diesem Buch vorgestellte Verfahren des *Supervised Learning* stellen wir die Modellannahme $f^* \in \mathcal{F}$ deutlich heraus, so dass in konkreten Anwendungsproblemen auch eine Bewertung des Approximationsfehlers möglich ist.

Liegt ein Regressionsproblem vor, so gibt es unabhängig von obiger Diskussion eine weitere Zerlegung des Excess Bayes Risks:

Lemma 1.12 (Excess Bayes Risk für die quadratische Verlustfunktion) Ist $\mathcal{Y} \subset \mathbb{R}$, L die quadratische Verlustfunktion und $\hat{f}_n$ ein beliebiger Algorithmus, so gilt

$$\mathbb{E}R(\hat{f}_n) - R(f^*) = \mathbb{E}\,\mathrm{MSE}(\hat{f}_n(X)),$$

wobei

$$\mathrm{MSE}(\hat{f}_n(x)) := \mathbb{E}[(\hat{f}_n(x) - f^*(x))^2], \quad x \in \mathcal{X}$$

der *mittlere quadratische Fehler* (engl. mean squared error) des Algorithmus $\hat{f}_n$ ist.

Es gilt die sogenannte *Bias-Varianz-Zerlegung*

$$\mathrm{MSE}(\hat{f}_n(x)) = \mathrm{Var}(\hat{f}_n(x)) + \left|\mathrm{Bias}(\hat{f}_n(x))\right|^2, \tag{1.6}$$

wobei

- $\mathrm{Bias}(\hat{f}_n(x)) := \mathbb{E}\hat{f}_n(x) - f^*(x)$ der *Bias* von $\hat{f}_n(x)$ genannt wird und
- $\mathrm{Var}(\hat{f}_n(x)) = \mathbb{E}\left[\left(\hat{f}_n(x) - \mathbb{E}\hat{f}_n(x)\right)^2\right]$ die *Varianz* von $\hat{f}_n(x)$ bezeichnet.

Beweis Ist L die quadratische Verlustfunktion, so gilt $f^*(X) = \mathbb{E}[Y|X]$ (vgl. Lemma 1.3), und daher:

$$
\begin{aligned}
\mathbb{E}R(\hat{f}_n) &= \mathbb{E}\left[\left(Y - f^*(X) + f^*(X) - \hat{f}_n(X)\right)^2\right] \\
&= \mathbb{E}\left[\left(Y - f^*(X)\right)^2\right] + 2\underbrace{\mathbb{E}\left[\mathbb{E}\left[(Y - f^*(X)) \cdot (f^*(X) - \hat{f}_n(X))\middle| X, T_n\right]\right]}_{=\mathbb{E}\left[(f^*(X)-\hat{f}_n(X))\cdot\mathbb{E}\left[(Y-f^*(X))\middle| X\right]\right]=0} \\
&\quad + \mathbb{E}\left[(f^*(X) - \hat{f}_n(X))^2\right] \\
&= R(f^*) + \mathbb{E}\left[\mathbb{E}\left[\left(\hat{f}_n(X) - f^*(X)\right)^2 \middle| X\right]\right]
\end{aligned}
$$

Die Berechnung der beiden Terme $\mathbb{E}R(\hat{f}_n)$ und $R(f^*)$ kann also verlagert werden auf die Untersuchung, wie sich die Differenz $\hat{f}_n(x) - f^*(x)$ für alle $x \in \mathcal{X}$ verhält. Gl. (1.6) bewertet den Algorithmus anhand zweier Merkmale:

- Die Varianz misst, wie stark der Algorithmus von den Trainingsdaten abhängt, d. h. wie stark sich eine Änderung einzelner Trainingsdatenpunkte auf die vom Algorithmus induzierte Entscheidungsregel auswirkt.
- Der Bias beschreibt den durchschnittlich erwarteten systematischen Fehler der von $\hat{f}_n$ erzeugten Entscheidungsregeln. Ein systematischer Fehler entsteht, wenn der Algorithmus bestimmte Entscheidungsregeln in $\mathcal{F}$ durch zusätzlich eingeführte Zwangsbedingungen bevorzugt. Dies kann durch Bestrafung unerwünschter Eigenschaften von $\hat{f}_n$ geschehen oder durch Einschränkung der Suche von $\hat{f}_n$ auf eine kleinere Funktionenklasse $\hat{\mathcal{F}} \subset \mathcal{F}$ (obwohl man nur weiß bzw. annimmt, dass $f^* \in \mathcal{F}$).

Es scheint zunächst nicht sinnvoll, mutwillig einen systematischen Fehler durch oben genannte Möglichkeiten einzuführen. Es hat sich jedoch gezeigt, dass ein moderater systematischer Fehler zu einer erheblichen Einsparung bei der Varianz führen kann. Tatsächlich geht ein kleiner Bias häufig mit hoher Varianz einher und ein großer Bias mit einer kleinen Varianz; man spricht von einem *Bias-Varianz-Tradeoff*. In diesem Sinne sind Algorithmen mit sehr kleinem Bias nicht notwendig diejenigen mit dem kleinsten Generalisierungsfehler. In Abb. 1.1 ist ein typischer Verlauf des Excess Bayes Risks $\mathbb{E}R(\hat{f}_n) - R(f^*)$ in Abhängigkeit von der Größe der Funktionenklasse $\hat{\mathcal{F}} \subset \mathcal{F}$ zu sehen.

1.3 Standardansätze zur Ermittlung von Algorithmen

In diesem Abschnitt stellen wir einen systematischen Ansatz zur Bestimmung von Algorithmen vor. Wir gehen davon aus, dass eine Modellannahme $f^* \in \mathcal{F}$ mit geeigneter Funktionenklasse $\mathcal{F} \subset \{f : \mathcal{X} \to \mathcal{Y} \text{ messbar}\}$ formuliert wurde und eine Verlustfunktion L vorliegt. Dann sollte auch $\hat{f}_n \in \mathcal{F}$ gelten und $R(\hat{f}_n)$ möglichst gering sein; idealerweise sollte

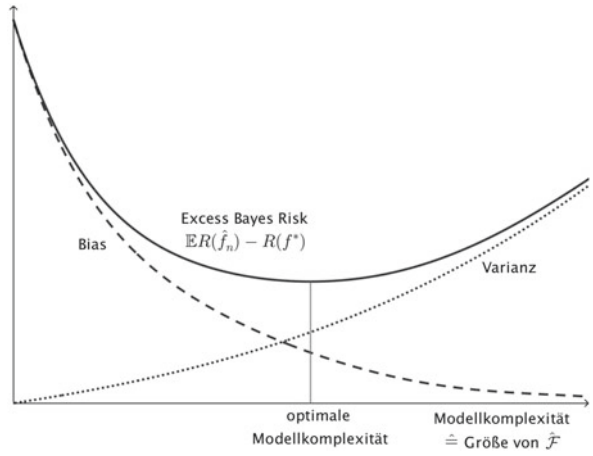


Abb. 1.1 Grafische Darstellung des Excess Bayes Risks $\mathbb{E}R(\hat{f}_n) - R(f^*)$ und der beiden Summanden Varianz und Bias in Abhängigkeit von der Größe der Funktionenklasse $\hat{\mathcal{F}} \subset \mathcal{F}$

$$\hat{f}_n \in \arg\min_{f \in \mathcal{F}} R(f) \tag{1.7}$$

gelten. Für eine beliebige Entscheidungsregel f kann das Risiko $R(f)$ wie folgt geschätzt werden:

Definition 1.13
Sind Trainingsdaten $(X_i, Y_i), i = 1, \ldots, n$ gegeben und ist $f : \mathcal{X} \to \mathcal{Y}$ eine Entscheidungsregel, so heißt

$$\hat{R}_n(f) := \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f(X_i))$$

das *empirische Risiko* von f auf den Trainingsdaten. ♦

Ersetzen wir $R(f)$ in Gl. (1.7) durch $\hat{R}_n(f)$, erhalten wir einen Standardansatz zur Bestimmung von Algorithmen.

Bemerkung 1.14 (Ansatz zur Bestimmung von Algorithmen 1)

$$\hat{f}_n \in \arg\min_{f \in \mathcal{F}} \hat{R}_n(f), \qquad \hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f(X_i)).$$

Hängen Elemente von $\mathcal{F}$ von sehr vielen oder sogar unendlich vielen frei wählbaren Parametern ab, so nennen wir $\mathcal{F}$ *hochdimensional* oder von *hoher Komplexität.* In diesem Fall ist das obige Minimierungsproblem nicht eindeutig lösbar und eine Lösung $\hat{f}_n$ ist überangepasst an die Trainingsdaten (vgl. Gl. (1.5)). Durch Bestrafung unerwünschter Eigenschaften von $\hat{f}_n$ kann die Menge der möglichen Lösungen reduziert werden. Dazu führen wir einen

Bestrafungsterm $J : \mathcal{F} \to \mathbb{R}$ (engl. *penalty term*) ein, welcher kleine Werte für erwünschte $f \in \mathcal{F}$ und große Werte für unerwünschte $f \in \mathcal{F}$ liefert. J wird als zusätzlicher Summand in das Minimierungsproblem eingefügt:

Bemerkung 1.15 (Ansatz zur Bestimmung von Algorithmen 2)

$$\hat{f}_{n,\lambda} \in \arg\min_{f \in \mathcal{F}} \left\{ \hat{R}_n(f) + \lambda \cdot J(f) \right\}, \tag{1.8}$$

wobei $\lambda \geq 0$ *Komplexitätsparameter* oder *Bestrafungsparameter* genannt wird.

λ gibt an, wie stark die Bestrafung $J(f)$ von f im Vergleich zu $\hat{R}_n(f)$ ausfällt. Im Falle hochdimensionaler $\mathcal{F}$ ist eine gute Wahl von λ notwendig, um einen Algorithmus $\hat{f}_n$ mit niedrigem Generalisierungsfehler zu erhalten.

Bemerkung 1.16 (Verbindung zwischen Bestrafungsterm und Bias) Unter Konvexitätsannahmen an $\hat{R}_n(f)$ und $J(f)$ in f kann man zeigen (vgl. Definition 2.32): Gl. (1.8) ist äquivalent dazu, dass $\hat{f}_{n,\lambda}$ Lösung des Minimierungsproblems

$$\hat{f}_{n,\lambda} \in \arg\min_{f \in \hat{\mathcal{F}}_\lambda} \hat{R}_n(f)$$

ist, wobei $\hat{\mathcal{F}}_\lambda = \{f \in \mathcal{F} : J(f) \leq \hat{t}(\lambda)\}$ und $\hat{t}(\lambda)$ eine reelle Zahl ist, die von λ und den Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$ abhängt. Üblicherweise ist $\lambda \mapsto \hat{t}(\lambda)$ monoton fallend, d. h., für wachsendes λ findet eine systematische Verkleinerung des Raums $\hat{\mathcal{F}}_\lambda$ statt. Gilt $\hat{f}_n \notin \hat{\mathcal{F}}_\lambda$, so entsteht durch $\hat{f}_{n,\lambda}$ eine systematische Verfälschung des allein auf der ursprünglichen Modellannahme basierenden Algorithmus $\hat{f}_n$, d. h., ein Bias entsteht. Im Allgemeinen führt ein großes λ daher zu einem großen Bias, aber aufgrund der geringeren Größe von $\hat{\mathcal{F}}_\lambda$ zu einer geringeren Varianz von $\hat{f}_{n,\lambda}$.

Parameter, welche vor der Berechnung von $\hat{f}_n$ gewählt werden müssen, werden auch allgemein ohne Angabe des Zwecks *Hyperparameter* oder *Tuningparameter* genannt. In Bemerkung 1.15 beispielsweise verwendet man für λ auch diese Bezeichnungen.

1.4 Trainieren, Validieren und Testen

Der Vorgang der Berechnung einer Entscheidungsregel $\hat{f}_n(\omega)$ aus einem vorgegebenen Algorithmus wird als *Training* bezeichnet.

Für Qualitätsvergleiche und -angaben wird in Publikationen das Risiko $R(\hat{f}_n)$ der ermittelten Entscheidungsregel angegeben, was dann gleichzeitig als Approximation des Generalisierungsfehlers $\mathbb{E}R(\hat{f}_n)$ fungiert. Für die Angabe von $R(\hat{f}_n)$ ist jedoch die Kenntnis von

$\mathbb{P}^{(X,Y)}$ nötig, die in der Praxis nicht gegeben ist. Werden jedoch nach der Berechnung von $\hat{f}_n$ weitere, von den Trainingsdaten unabhängige *Testdaten* zur Verfügung gestellt, so kann $R(\hat{f}_n)$ durch Mittelung der erhaltenen Verluste geschätzt werden:

Bemerkung 1.17 (Schätzung von $R(\hat{f}_n)$) Seien $(\tilde{X}_i, \tilde{Y}_i) \overset{\text{iid}}{\sim} \mathbb{P}^{(X,Y)}$, $i = 1, \ldots, m$ Beobachtungen *(Testdaten)* unabhängig von T_n. Dann gilt aufgrund des starken Gesetzes der großen Zahlen:

$$\text{empRT}(\hat{f}_n) := \frac{1}{m} \sum_{i=1}^{m} L\left(\tilde{Y}_i, \hat{f}_n(\tilde{X}_i)\right) \overset{m\to\infty}{\to} \mathbb{E}\left[L(Y, \hat{f}_n(X))|T_n\right] = R(\hat{f}_n) \quad f.s.$$

$\text{empRT}(\hat{f}_n)$ heißt *empirisches Risiko* von $\hat{f}_n$ auf den Testdaten oder *Testfehler.*

In Anwendungsproblemen sind Testdaten nicht immer verfügbar. Bemerkung 1.17 kann dann genutzt werden, indem die zur Verfügung gestellten Beobachtungen in Trainingsdaten $(X_i, Y_i)_{i=1,\ldots,n}$ und Testdaten $(\tilde{X}_i, \tilde{Y}_i)_{i=1,\ldots,m}$ aufgeteilt werden (zum Beispiel im Verhältnis $n : m = 90 : 10$) und die Berechnung von $\hat{f}_n$ *nur* mittels der Trainingsdaten erfolgt.

Bemerkung 1.18 Das empirische Risiko von $\hat{f}_n$ über die Trainingsdaten,

$$\hat{R}_n(\hat{f}_n) = \frac{1}{n} \sum_{i=1}^{n} L\left(Y_i, \hat{f}_n(X_i)\right),$$

wird auch *Trainingsfehler* genannt. Zur Schätzung von $R(\hat{f}_n)$ darf *nicht* $\hat{R}_n(\hat{f}_n)$ genutzt werden. Bei der Ermittlung von $\hat{f}_n$ sind Informationen aus den Trainingsdaten eingeflossen; wurde zum Beispiel Bemerkung 1.14 genutzt, so wurde die Entscheidungsregel $\hat{f}_n(\omega)$ darauf trainiert, sich besonders gut an $(X_i, Y_i)_{i=1,\ldots,n}$ anzupassen. Damit ist die Voraussetzung der Unabhängigkeit in Bemerkung 1.17 verletzt, und im Allgemeinen gilt $\hat{R}_n(\hat{f}_n) \leq R(\hat{f}_n)$. Im Extremfall der Überanpassung Gl. (1.5) gilt $\hat{R}_n(\hat{f}_n) = 0$ (obwohl die Qualität von $\hat{f}_n$ gerade in diesem Fall oft sehr schlecht ist).

Hängt $\hat{f}_{n,\lambda}$ von einem Tuningparameter $\lambda \geq 0$ ab, so wird in der Praxis auch eine Anleitung für eine möglichst fundierte Wahl von λ benötigt. Wir konzentrieren uns hier nur auf die Tuningparameter des Typs aus Gl. (1.8), die nachfolgend vorgestellten Ansätze sind aber auch bei allen anderen Arten anwendbar.

Ein üblicher Ansatz arbeitet mit der Schätzung des Risikos $R(\hat{f}_{n,\lambda})$ und einer darauf basierenden Wahl von λ. Wie bereits in Bemerkung 1.17 motiviert, teilen wir dazu die gegebenen Beobachtungen in eine Menge $(X_i, Y_i)_{i=1,\ldots,n}$ (Trainingsdaten) und eine Menge $(\bar{X}_i, \bar{Y}_i)_{i=1,\ldots,m}$ (genannt *Validierungsdaten*) auf und verfahren wie folgt:

Bemerkung 1.19 (Ermittlung von geeigneten Hyperparametern: Validierung) Seien $(\bar{X}_i, \bar{Y}_i) \overset{\text{iid}}{\sim} \mathbb{P}^{(X,Y)}$, $i = 1, \ldots, m$ Beobachtungen *(Validierungsdaten)* unabhängig von T_n. Ermittle $\hat{f}_{n,\lambda}$ auf Basis der Trainingsdaten $(X_i, Y_i)_{i=1,\ldots,n}$ und wähle

$$\hat{\lambda}^{std} :\in \operatorname*{arg\,min}_{\lambda \geq 0} \text{empVR}(\hat{f}_{n,\lambda}), \tag{1.9}$$

wobei

$$\text{empRV}(\hat{f}_{n,\lambda}) := \frac{1}{m} \sum_{i=1}^{m} L\left(\bar{Y}_i, \hat{f}_{n,\lambda}(\bar{X}_i)\right)$$

das *empirische Risiko* auf den Validierungsdaten genannt wird.

Bemerkungen

- Das Vorgehen in Bemerkung 1.19 heißt *Validierung*. Der Name bezieht sich darauf, dass verschiedene Werte für λ anhand der Validierungsdaten geprüft (validiert) werden und danach der am meisten geeignete Wert ausgewählt wird.
- Wir erwarten, dass $\hat{f}_{n,\hat{\lambda}^{std}}$ wegen Bemerkung 1.17 und

$$R\left(\hat{f}_{n,\hat{\lambda}^{std}}\right) \approx \text{empVR}\left(\hat{f}_{n,\hat{\lambda}^{std}}\right) = \min_{\lambda \geq 0} \text{empVR}\left(\hat{f}_{n,\lambda}\right) \approx \min_{\lambda \geq 0} R\left(\hat{f}_{n,\lambda}\right)$$

  zumindest näherungsweise das minimal mögliche Risiko unter allen $\hat{f}_{n,\lambda}$ erreicht.
- Zur Durchführung des obigen Ansatzes muss $\hat{f}_{n,\lambda}$ nicht für alle $\lambda \geq 0$ berechnet werden. Oft genügt die Berechnung zum Beispiel auf einem geometrischen Gitter $\lambda \in \{a^k : k \in \mathbb{Z}\}$ mit $0 < a < 1$, wobei noch besonders große und kleine Werte von λ ausgeschlossen werden müssen.

Beachte: Wurde $\hat{f}_n = \hat{f}_{n,\hat{\lambda}^{std}}$ durch Validierung ermittelt, so darf weder das empirische Risiko über die Trainingsdaten $\hat{R}_n(\hat{f}_n)$ noch das empirische Risiko über die Validierungsdaten $\text{empVR}(\hat{f}_n)$ zum Schätzen von $R(\hat{f}_n)$ genutzt werden. Als Faustregel gilt: Sobald die Daten in irgendeiner Form in die Berechnung von $\hat{f}_n$ eingeflossen sind, dürfen sie nicht mehr zur Ermittlung einer Schätzung des Risikos $R(\hat{f}_n)$ genutzt werden.

Eine korrekte Kombination von Training, Validierung und Risikoberechnung ergibt sich durch das folgende Vorgehen:

Bemerkung 1.20 (Trainieren, Validieren und Testen) Teile die ursprünglichen Beobachtungen in Trainingsdaten $(X_i, Y_i)_{i=1,\ldots,n}$, Validierungsdaten $(\bar{X}_i, \bar{Y}_i)_{i=1,\ldots,m_1}$ (falls nötig) und Testdaten $(\tilde{X}_i, \tilde{Y}_i)_{i=1,\ldots,m_2}$ auf, zum Beispiel im Verhältnis $n : m_1 : m_2 = 70 : 20 : 10$. Führe dann folgende Schritte aus:

1. Ermittle $\hat{f}_{n,\lambda}$ mit $(X_i, Y_i)_{i=1,\dots,n}$ für verschiedene $\lambda \geq 0$.
2. Ermittle $\hat{f}_n := \hat{f}_{n,\hat{\lambda}^{std}}$ gemäß Gl. (1.9).
3. Eine Schätzung von $R(\hat{f}_n)$ ist gegeben durch den Testfehler (vgl. Bemerkung 1.17)

$$\text{empRT}(\hat{f}_n) = \frac{1}{m_2} \sum_{i=1}^{m_2} L\left(\tilde{Y}_i, \hat{f}_n(\tilde{X}_i)\right).$$

   Der Trainingsfehler ist gegeben durch $\hat{R}_n(\hat{f}_n) = \frac{1}{n} \sum_{i=1}^{n} L(Y_i, \hat{f}_n(X_i))$ (vgl. Bemerkung 1.18).

In diesem Buch legen wir bei der Ermittlung von Algorithmen den Fokus stets nur auf die Techniken für Schritt (1).

1.4.1 Überwachung des Validierungsfehlers bei iterativen Verfahren

Einige Algorithmen $\hat{f}_{n,\lambda}$ wie zum Beispiel neuronale Netzwerke (vgl. Abschn. 7.3) oder das Boosten von Bäumen (vgl. Abschn. 6.5) werden durch Iterationsverfahren bestimmt, die in jedem Schritt $m \in \mathbb{N}$ einen Algorithmus $\hat{f}_{n,\lambda}^{(m)}$ liefern. Bei den genannten Beispielen ist der als Grenzwert $\lim_{m\to\infty} \hat{f}_{n,\lambda}^{(m)} = \hat{f}_{n,\lambda}$ des Iterationsverfahrens definierte Algorithmus *überangepasst* an die Trainingsdaten, wobei die Überanpassung durch das Iterationsverfahren „angesammelt“ wurde. Dies geschieht dadurch, dass mit jedem neuen Iterationsschritt weitere Parameter der Modellannahme aktiviert werden. Der Algorithmus $\hat{f}_{n,\lambda}^{(m)}$ erfüllt somit $\hat{f}_{n,\lambda}^{(m)} \in \hat{\mathcal{F}}_\lambda^{(m)} \subset \mathcal{F}$, und für $m' > m$ gilt $\hat{\mathcal{F}}_\lambda^{(m)} \subset \hat{\mathcal{F}}_\lambda^{(m')}$. Entsprechend erwartet man, dass die Kurve der Risiken

$$m \mapsto R\left(\hat{f}_{n,\lambda}^{(m)}\right)$$

für kleines m zunächst absinkt (anfängliche Anpassung an die Trainingsdaten, kleiner werdender Bias), bei einem bestimmten Iterationsschritt $m^* \in \mathbb{N}$ ein Minimum erreicht und danach wieder ansteigt (aufgrund der wachsenden Varianz), vgl. Abb. 1.1.

Durch Überwachung des empirischen Risikos auf den Validierungsdaten kann somit eine geeignete Abbruchbedingung für das Iterationsverfahren bestimmt werden. Dieses Vorgehen heißt *early stopping:*

Bemerkung 1.21 (Überwachung des Validierungsfehlers bei iterativen Verfahren) Teile die ursprünglichen Beobachtungen in Trainingsdaten $(X_i, Y_i)_{i=1,\dots,n}$ und Validierungsdaten $(\bar{X}_i, \bar{Y}_i)_{i=1,\dots,m_1}$ auf. Sei $M \in \mathbb{N}$ eine vorgegebene maximal mögliche Anzahl an durchgeführten Iterationen.

1. Für verschiedene $\lambda \geq 0$:
   (1a) Bestimme für jede Iteration $m \in \{1, \ldots, M\}$ die Abbildung $\hat{f}_{n,\lambda}^{(m)}$ und das zugehörige empirische Risiko

$$\text{empRV}(\hat{f}_{n,\lambda}^{(m)}) = \frac{1}{n} \sum_{i=1}^{n} L\left(\bar{Y}_i, \hat{f}_{n,\lambda}^{(m)}(\bar{X}_i)\right).$$

   (1b) Bestimme $m^*(\lambda) := \arg\min_{m \in \{1,\ldots,M\}} \text{empRV}\left(\hat{f}_{n,\lambda}^{(m)}\right)$.
2. Ermittle $\hat{f}_n := \hat{f}_{n,\hat{\lambda}^{std}}^{m^*(\hat{\lambda}^{std})}$ gemäß Gl. (1.9) aus $\hat{f}_{n,\lambda}^{m^*(\lambda)}$.

Bemerkung Für die Anwendung obigen Verfahrens muss $\hat{f}_n$ nicht notwendig von λ abhängen; der Schritt (2) ändert sich dann zu $\hat{f}_n := \hat{f}_n^{(m^*)}$.

1.5 Alternative Validierungsmethoden

Hängt $\hat{f}_{n,\lambda}$ von einem Hyperparameter λ ab, müssen zur Ermittlung eines finalen Algorithmus $\hat{f}_n = \hat{f}_{n,\hat{\lambda}}$ gemäß Bemerkung 1.20 die ursprünglichen Beobachtungen in Trainings- und Validierungsdaten aufgeteilt werden. Nur die n Trainingsdaten fließen in die Berechnung von $\hat{f}_{n,\lambda}$ ein; die Validierungsdaten dienen nur der Wahl eines geeigneten λ.

Für einen niedrigen Generalisierungsfehler von $\hat{f}_n$ bzw. ein niedriges Risiko der zugehörigen Entscheidungsfunktion ist vor allem die Anzahl der verwendeten Trainingsdaten n entscheidend, und eine Verwendung von zu vielen Beobachtungen zur Validierung muss vermieden werden. Andererseits dürfen nicht zu wenige Daten zur Validierung genutzt werden, weil dann die Qualität der Auswahl von λ leidet. Dieses Dilemma tritt vor allem dann auf, wenn insgesamt nur wenige Beobachtungen zur Ermittlung von $\hat{f}_n$ vorliegen.

Wir stellen hier zwei Methoden vor, welche es erlauben, trotz eines Hyperparameters alle Beobachtungen zur Bestimmung von $\hat{f}_{n,\lambda}$ zu benutzen. Im Gegensatz zu Bemerkung 1.20 erfolgt die Auswahl von λ nicht auf Basis des Risikos $R(\hat{f}_{n,\lambda})$, sondern auf Basis des Generalisierungsfehlers $\mathbb{E}R(\hat{f}_{n,\lambda})$ (bzw. einer Schätzung davon). Dieses Vorgehen ist mit Vorsicht zu genießen. Im Gegensatz zum empirischen Risiko berücksichtigt der Generalisierungsfehler nicht die gerade vorliegenden Trainingsdaten T_n, sondern trifft die Wahl auf Basis des gemittelten Risikos über alle möglichen beobachtbaren Konfigurationen von Trainingsdaten. Das bedeutet, das gewählte $\hat{\lambda}$ ist zwar für viele Konfigurationen von Trainingsdaten eine gute Wahl, aber eventuell nicht für die gerade vorliegende. Den Einfluss dieses Sachverhalts auf die Qualität der Verfahren sowie weitere Vor- und Nachteile diskutieren wir jeweils kurz in den Bemerkungen 1.23 und 1.28.

1.5.1 Kreuzvalidierung (Cross Validation)

Bei der Kreuzvalidierung (engl. *Cross Validation*) werden keine Validierungsdaten $(\bar{X}_i, \bar{Y}_i)_{i=1,\dots,m_1}$ von den Beobachtungen abgespalten. Stattdessen werden die Trainingsdaten in M gleich große Gruppen aufgeteilt (vgl. Abb. 1.2) und selbst untereinander zur Validierung genutzt.

Zur Durchführung des Verfahrens darf der Algorithmus nicht nur für genau n Trainingsdaten berechenbar sein, sondern man braucht eine allgemeine Bildungsvorschrift $\hat{f}_{N,\lambda} = A_{N,\lambda}(T_N)$ des Algorithmus für eine beliebige Anzahl $N \in \mathbb{N}$ von Trainingsdaten T_N. Insbesondere muss gewährleistet sein, dass λ bei dieser Bildungsvorschrift unabhängig von der Anzahl der Trainingsdaten dieselbe Bedeutung und den gleichen Einfluss auf das Aussehen des Algorithmus hat. Die Bildungsvorschrift Gl. (1.8) beispielsweise erlaubt direkt die Bestimmung von $\hat{f}_{N,\lambda}$ für eine beliebige Anzahl von Trainingsdaten, und λ steht stets für die Bestrafung einer unerwünschten Eigenschaft (in gleicher Höhe).

Um die Notationslast zu verringern, unterdrücken wir in der folgenden Definition die Abhängigkeit von A von N, d. h., wir schreiben $A_\lambda(T_N) = A_{N,\lambda}(T_N)$.

Definition 1.22 (M-fold Cross Validation)
Sei $M \in \{2, \dots, n\}$.

(0) Teile die Trainingsdaten zufällig in M gleich große Teile auf. Wir erhalten eine disjunkte Zerlegung

$$\{1, \dots, n\} = \dot{\bigcup_{m=1}^{M}} I_m, \qquad I_m \subset \{1, \dots, n\} \quad (m = 1, \dots, M).$$

(1') Berechne den Algorithmus $\tilde{f}_{n,\lambda}^{(-m)} := A_\lambda(T_n^{(-m)})$ basierend auf den Trainingsdaten $T_n^{(-m)} := (X_i, Y_i)_{i \in \{1,\dots,n\} \setminus I_m}$ für verschiedene $\lambda \geq 0$.

(2') Sei $\hat{f}_n = \hat{f}_{n,\hat{\lambda}^{cv}}(T_n)$ der Algorithmus basierend auf allen Trainingsdaten, wobei

$$\hat{\lambda}^{cv} :\in \operatorname*{arg\,min}_{\lambda \geq 0} \mathrm{CV}_n(\lambda),$$

und

$$\mathrm{CV}_n(\lambda) := \frac{1}{n} \sum_{i=1}^{n} L\left(Y_i, \tilde{f}_{n,\lambda}^{(-m(i))}(X_i)\right) = \frac{1}{n} \sum_{m=1}^{M} \sum_{i \in I_m} L\left(Y_i, \tilde{f}_{n,\lambda}^{(-m)}(X_i)\right)$$

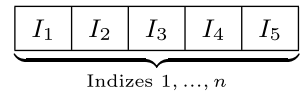


Abb. 1.2 Aufteilung der Trainingsdaten bei 5-fold Cross Validation

und $m : \{1, \dots, n\} \to \{1, \dots, M\}$ jedem Index i den Index m zuordnet mit $i \in I_m$.

In diesem Fall schreiben wir, dass $\hat{f}_n$ bzw. $\hat{\lambda}^{cv}$ durch *M-fache Kreuzvalidierung* (engl. *M-fold Cross Validation*) gewählt wurde. ♦

Bemerkung 1.23 (Zur Theorie von Cross Validation)

- Im Fall $M = n$ entspricht dem Training von n Algorithmen, bei welchem jeweils eine Beobachtung ausgelassen wird. Dieser Spezialfall heißt *Leave-one-out Cross Validation.*
- Bei der M-fachen Kreuzvalidierung wird das empirische Risiko der auf $T_n^{(-m)}$ basierenden Algorithmen $\tilde{f}_{n,\lambda}^{(-m)}$, $m = 1, \dots, M$ mit den jeweils ausgelassenen Trainingsdaten bestimmt. $\mathrm{CV}_n(\lambda)$ mittelt diese Ausdrücke für $m = 1, \dots, M$. Daher stellt $\mathrm{CV}_n(\lambda)$ im Gegensatz zu $\mathrm{empVR}(\hat{f}_{n,\lambda})$ in Bemerkung 1.19 keine direkte Schätzung von $R(\hat{f}_{n,\lambda})$ dar. Nehmen wir der Einfachheit halber an, dass $\#I_m = \frac{n}{M} \in \mathbb{N}$ gilt, so erwarten wir aufgrund des Gesetzes der großen Zahlen:

$$\begin{aligned} \mathrm{CV}_n(\lambda) &= \frac{1}{M} \sum_{m=1}^{M} \frac{1}{\#I_m} \sum_{i \in I_m} L\left(Y_i, \tilde{f}_{n,\lambda}^{(-m)}(X_i)\right) \\ &\approx \frac{1}{M} \sum_{m=1}^{M} R\left(\tilde{f}_{n,\lambda}^{(-m)}\right) \approx \mathbb{E}R\left(\tilde{f}_{n,\lambda}^{(-1)}\right) = \mathbb{E}R(\hat{f}_{n-\frac{n}{M},\lambda}), \end{aligned}$$

  d. h., $\mathrm{CV}_n(\lambda)$ schätzt den Generalisierungsfehler von $\hat{f}_{n-\frac{n}{M},\lambda}$ anstelle des Risikos $R(\hat{f}_{n,\lambda})$. In den Extremfällen $M \in \{2, n\}$ wird also eigentlich ein optimales λ für einen Algorithmus basierend auf $\frac{n}{2}$ bzw. $n - 1$ Trainingsdaten gewählt; daher ist aus theoretischer Sicht ein hohes M zu bevorzugen.
- Die Tatsache, dass $\mathrm{CV}_n(\lambda)$ einen Generalisierungsfehler schätzt, ist hier kein Problem, da die Schätzung ohne weitere Modellannahme mit den Trainingsdaten erfolgt. In diesem Sinne ist zu erwarten, dass eine Minimierung von $\mathrm{CV}_n(\lambda)$ ähnlich gute, die Trainingsdaten berücksichtigende Resultate liefert wie die Minimierung von $R(\hat{f}_{n,\lambda})$ in Bemerkung 1.19.

Bemerkung 1.24 (Zur Anwendung von Cross Validation) In der Praxis wählt man häufig kleine M, z. B. $M = 5$ oder $M = 10$. Der Grund ist, dass für die Durchführung von Cross Validation der Algorithmus jeweils M-mal mit verschiedenen Trainingsdaten und für verschiedene λ bestimmt werden muss. Falls die Bestimmung der betrachteten Algorithmen sehr zeitaufwendig ist, kann Cross Validation in der Praxis oft nicht umgesetzt werden.

1.5.2 AIC – Akaike Information Criterion

Das *Akaike Information Criterion* (kurz AIC) wurde von [1] entwickelt. Es schätzt den Generalisierungsfehler $\mathbb{E}R(\hat{f}_{n,\lambda})$ durch eine Korrektur des Trainingsfehlers $\hat{R}_n(\hat{f}_{n,\lambda})$. Wir

besprechen das Verfahren hier nur heuristisch und für ein spezielles Regressionsproblem, da es in der Praxis des maschinellen Lernens selten genutzt wird:

Modellannahme 1.25 (AIC) $X_1, \dots, X_n \in \mathcal{X} \subset \mathbb{R}^d$ seien vorher festgelegte, nicht zufällige Messpunkte. Es sei $f^* : \mathcal{X} \to \mathbb{R}$ eine messbare Funktion. Für die Daten $(X_i, Y_i)_{i=1,\dots,n}$ gelte:

$$\varepsilon_i := Y_i - f^*(X_i), \quad i = 1, \dots, n \text{ sind i. i. d. Zufallsvariablen in } \mathbb{R} \text{ mit } \mathbb{E}\epsilon_i = 0. \tag{1.10}$$

Beachte: Da die X_i deterministisch sind, sind $(X_i, Y_i), i = 1, \dots, n$ *nicht* i. i. d., die Konvergenzresultate für Trainings- und Generalisierungsfehler sind also nicht direkt übertragbar. Durch die Modellannahme gilt jedoch $f^*(X_i) = \mathbb{E}Y_i$, d. h., obiges f^* bildet ein Analogon zur Bayes-Regel $f^*(x) = \mathbb{E}[Y|X = x]$.

In diesem Kontext lautet der Trainingsfehler eines beliebigen $\hat{f}_{n,\lambda}$:

$$\hat{R}_n(\hat{f}_{n,\lambda}) = \frac{1}{n} \sum_{i=1}^{n} L\left(Y_i, \hat{f}_{n,\lambda}(X_i)\right).$$

Seien $(X_i, \tilde{Y}_i), i = 1, \dots, n$ neue, von $(X_i, Y_i), i = 1, \dots, n$ unabhängige Beobachtungen, welche Gl. (1.10) folgen. Der Generalisierungsfehler von $\hat{f}_{n,\lambda}$ ist dann $\mathbb{E}R(\hat{f}_{n,\lambda})$ mit

$$R(\hat{f}_{n,\lambda}) = \frac{1}{n} \sum_{i=1}^{n} L\left(\tilde{Y}_i, \hat{f}_{n,\lambda}(X_i)\right).$$

Zwischen diesen beiden Größen besteht folgender Zusammenhang:

Lemma 1.26 (Trainingsfehler vs. Generalisierungsfehler) Sei $L(y, s) = (y - s)^2$ die quadratische Verlustfunktion. Dann gilt:

$$\mathbb{E}R\left(\hat{f}_{n,\lambda}\right) - \mathbb{E}\hat{R}_n\left(\hat{f}_{n,\lambda}\right) = \frac{2}{n} \sum_{i=1}^{n} \mathrm{Cov}\left(Y_i, \hat{f}_{n,\lambda}(X_i)\right)$$

Beweis Es gilt

$$\begin{aligned}
&\left(\tilde{Y}_i - \hat{f}_{n,\lambda}(X_i)\right)^2 - \left(Y_i - \hat{f}_{n,\lambda}(X_i)\right)^2 \\
&= \left[\tilde{Y}_i^2 - 2\tilde{Y}_i \hat{f}_{n,\lambda}(X_i) + \hat{f}_{n,\lambda}(X_i)^2\right] - \left[Y_i^2 - 2Y_i \hat{f}_{n,\lambda}(X_i) + \hat{f}_{n,\lambda}(X_i)^2\right] \\
&= \tilde{Y}_i^2 - Y_i^2 + 2(Y_i \hat{f}_{n,\lambda}(X_i) - \tilde{Y}_i \hat{f}_{n,\lambda}(X_i)).
\end{aligned} \tag{1.11}$$

Damit folgt

$$\begin{aligned}\mathbb{E}R(\hat{f}_{n,\lambda}) - \mathbb{E}\hat{R}_n(\hat{f}_{n,\lambda}) &= \frac{1}{n}\sum_{i=1}^{n}\left\{\mathbb{E}(\tilde{Y}_i - \hat{f}_{n,\lambda}(X_i))^2 - \mathbb{E}(Y_i - \hat{f}_{n,\lambda}(X_i))^2\right\} \\ &\overset{(2.11)}{=} \frac{2}{n}\sum_{i=1}^{n}\left\{\mathbb{E}[Y_i\hat{f}_{n,\lambda}(X_i)] - \underbrace{\mathbb{E}[\tilde{Y}_i\hat{f}_{n,\lambda}(X_i)]}_{=\mathbb{E}\tilde{Y}_i\mathbb{E}\hat{f}_{n,\lambda}(X_i)}\right\} \\ &\overset{\mathbb{E}Y_i=\mathbb{E}\tilde{Y}_i}{=} \frac{2}{n}\sum_{i=1}^{n}\mathrm{Cov}(Y_i, \hat{f}_{n,\lambda}(X_i)).\end{aligned}$$

Nutzen wir die Approximation $\mathbb{E}\hat{R}_n(\hat{f}_{n,\lambda}) \approx \hat{R}_n(\hat{f}_{n,\lambda})$, so erhalten wir aus Lemma 1.26:

$$\mathbb{E}R\left(\hat{f}_{n,\lambda}\right) \approx \hat{R}_n\left(\hat{f}_{n,\lambda}\right) + \frac{2}{n}\sum_{i=1}^{n}\mathrm{Cov}\left(Y_i, \hat{f}_{n,\lambda}(X_i)\right) \tag{1.12}$$

Das bedeutet, eine Schätzung des Generalisierungsfehlers kann erhalten werden durch eine Korrektur des Trainingsfehlers um einen zusätzlichen Summanden. Die Terme $\mathrm{Cov}(Y_i, \hat{f}_{n,\lambda}(X_i))$ repräsentieren die Abhängigkeit zwischen Y_i und der Vorhersage $\hat{f}_{n,\lambda}(X_i)$ und werden am größten für $Y_i = \hat{f}_{n,\lambda}(X_i)$. In diesem Sinne bestraft der zusätzliche Summand die Überanpassung an die Trainingsdaten.

Gl. (1.12) ist ohne eine detailliertere Modellannahme in der Praxis wertlos, da keine einfachen Schätzer für die theoretische Größe $\mathrm{Cov}(Y_i, \hat{f}_{n,\lambda}(X_i))$ existieren. Für einfach strukturierte Modellannahmen kann $\sum_{i=1}^{n}\mathrm{Cov}(Y_i, \hat{f}_{n,\lambda}(X_i))$ jedoch explizit berechnet und geschätzt werden. Eine häufig auftretende Form ist

$$\sum_{i=1}^{n}\mathrm{Cov}\left(Y_i, \hat{f}_{n,\lambda}(X_i)\right) = \mathrm{df}(\lambda)\cdot\sigma_\varepsilon^2, \tag{1.13}$$

mit $\mathrm{df}(\lambda) \geq 0$ nur abhängig von λ und $\sigma_\varepsilon^2 := \mathrm{Var}(\varepsilon_1)$. Dann heißt $\mathrm{df}(\lambda)$ die *Anzahl der Freiheitsgrade* bzw. die *effektive Anzahl der Parameter* im Modell. In diesem Fall kann die Validierung von λ wie folgt durchgeführt werden:

Definition 1.27 (AIC-Validierung)
Es gelte die Modellannahme 1.25 und Gl. (1.13). Es sei $\hat{\sigma}_n^2$ ein Schätzer für σ_ε^2. Dann heißt

$$\mathrm{AIC}_n(\lambda) := \hat{R}_n\left(\hat{f}_{n,\lambda}\right) + \frac{2}{n}\cdot\mathrm{df}(\lambda)\cdot\sigma_\varepsilon^2 \tag{1.14}$$

Akaike Information Criterion und $\hat{\lambda}^{aic} := \arg\min_{\lambda\geq 0}\mathrm{AIC}_n(\lambda)$. In diesem Fall schreiben wir, dass $\hat{\lambda}^{aic}$ durch *AIC-Validierung* gewählt wurde. ♦

Bemerkung 1.28

- Im Gegensatz zu Cross Validation ist für die Ermittlung von AIC(λ) wegen Gl. (1.13) eine konkrete Modellannahme nötig. Trifft diese zu, liefert das Verfahren ein gutes $\hat{\lambda}^{aic}$, welches aber möglicherweise nicht optimal auf die gerade aus den Trainingsdaten ermittelte Entscheidungsfunktion $\hat{f}_{n,\lambda}(\omega)$ angepasst ist, vgl. die Bemerkungen am Anfang von Abschn. 1.5.
- Ein allgemeiner Ansatz zur Wahl von $\hat{\sigma}_n^2$ ist $\hat{\sigma}_n^2 = \hat{R}_n(\hat{f}_{n,\tilde{\lambda}})$, wobei $\tilde{\lambda} \geq 0$ so gewählt sein muss, dass durch $\hat{f}_{n,\tilde{\lambda}}$ zumindest noch keine Überanpassung eingetreten ist. Bei der Nutzung dieses Schätzers muss man also bereits einen groben Bereich kennen, in dem λ sich aufhalten muss.
- Die Qualität von $\hat{\lambda}^{aic}$ hängt von der Qualität des Schätzers $\hat{\sigma}_n^2$ und von der Qualität der Modellannahme (in Hinsicht auf den Approximationsfehler) bzw. die daraus resultierende Gl. (1.13) ab.

Wir werden das AIC nur einmal in Kap. 2 beim Ridge-Schätzer verwenden. Abwandlungen des AIC, welche für Bayes'sche Modelle im maschinellen Lernen entwickelt wurden, sind zum Beispiel BIC *(Bayes Information Criterion)* oder DIC *(Deviance Information Criterion)*.

1.6 Vorverarbeitung der Daten in der Praxis

In Anwendungsproblemen können sich die einzelnen Komponenten Daten X_i um einige Größenordnungen unterscheiden. Auch wenn dies für die mathematische Theorie irrelevant ist, so kann dies bei der Berechnung der Algorithmen auf dem Computer problematisch sein. In einigen Algorithmen werden beispielsweise Rechenoperationen wie Exponentialfunktionen oder Polynomfunktionen hohen Grades auf die Trainingsdaten X_i angewandt, so dass zu große bzw. zu kleine Werte der X_i möglicherweise zu Fehlern in der Gleitpunktarithmetik oder zu arithmetischen Überläufen der verwendeten Variablen führen.

Aus diesem Grund empfiehlt es sich, die Daten vor der Anwendung von Algorithmen zu standardisieren und eventuell um deterministische Größen zu erweitern. Formal drücken wir dies aus mittels einer Transformation

$$T : \mathcal{X}^n \to \tilde{\mathcal{X}}^n,$$

welche die ursprünglichen Trainingsdaten $(X_{ij})_{j=1,\ldots,d} = X_i \in \mathcal{X} \subset \mathbb{R}^d$ auf neue Trainingsdaten $(\tilde{X}_{ij})_{j=1,\ldots,d} = \tilde{X}_i = T(X_i) \in \tilde{\mathcal{X}} \subset \mathbb{R}^{\tilde{d}}$ abbildet. Einige typische Transformationen sind:

Beispiel 1.29 (Beispiele für vorverarbeitende Transformationen)

(i) Zentrierung und Standardisierung (erzeugt Trainingsdaten $\tilde{X}_i$ mit gleichen Größenordnungen):

$$\tilde{X}_{ij} := \frac{X_{ij}^{\circ}}{\sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(X_{kj}^{\circ}\right)^2}}, \qquad X_{ij}^{\circ} := X_{ij} - \frac{1}{n}\sum_{k=1}^{n} X_{kj}$$

Nach dieser Transformation gilt $\frac{1}{n}\sum_{i=1}^{n}\tilde{X}_{ij} = 0$ und $\frac{1}{n}\sum_{i=1}^{n}\tilde{X}_{ij}^2 = 1$.

(ii) Zu restriktive Annahmen einzelner Modelle können erweitert werden, indem die Trainingsdaten X_i modifiziert und in einen größeren Raum eingebettet, die Algorithmen der Modelle dann auf $(\tilde{X}_i, Y_i)$ anstatt (X_i, Y_i) angewandt werden. Zwei Beispiele für T sind:

$$\tilde{X}_i := \left(1, (X_i)^T\right)^T \quad \text{(Hinzufügen eines Mittelwerts)},$$

$$\tilde{X}_i := \left(1, X_{i1}, \ldots, X_{id}, X_{i1}^2, \ldots, X_{id}^2\right) \quad \text{(Quadratische Funktionen der Komponenten)}$$

Formal bedeutet eine solche Modellerweiterung, dass die zuvor getroffene Annahme $f^* \in \mathcal{F}$ modifiziert wird zu $f^*(x) = \tilde{f}^*(T(x))$ mit $\tilde{f}^* \in \mathcal{F}$. Ein durch $(\tilde{X}_i, Y_i)$ erhaltener Algorithmus $\tilde{f}_n : \tilde{\mathcal{X}} \to \mathcal{Y}$ liefert daher mittels $\hat{f}_n(x) := \tilde{f}_n(T(x))$ einen Algorithmus für die ursprünglichen Beobachtungen $x \in \mathcal{X}$.

Außer in Kap. 2 werden wir die Standardisierung von Trainingsdaten bei den in diesem Buch vorgestellten Algorithmen nicht mehr kommentieren.

Achtung: Insbesondere in Hinsicht auf (ii) gilt:

- Bei Anwendung eines Algorithmus auf die transformierten Daten $(\tilde{X}_i, Y_i)_{i=1,\ldots,n}$ muss die Modellannahme auf diese und nicht mehr für die ursprünglichen Daten $(X_i, Y_i)_{i=1,\ldots,n}$ zutreffen!
- Da viele Algorithmen bei hohen Dimensionen von $\tilde{X}_i$ schlechter funktionieren, ist diese Technik mit Bedacht einzusetzen. In Kap. 4 werden wir eine Erweiterung kennenlernen.

1.7 Übungen

1. Weisen Sie nach, dass die in Lemma 1.3 angegebenen Funktionen f^* das zugehörige Risiko $R(f)$ minimieren.

Lineare Algorithmen für Regressionsprobleme

2

Inhaltsverzeichnis

In diesem Kapitel betrachten wir Regressionsprobleme, d. h., es gilt $\mathcal{Y} = \mathbb{R}$. Wir bewerten das Risiko von Entscheidungsregeln mit der quadratischen Verlustfunktion $L(y, s) = (y - s)^2$. Aus Lemma 1.3 folgt, dass die Bayes-Regel die Form

$$f^*(x) = \mathbb{E}[Y_1 | X_1 = x]$$

besitzt. In diesem Kapitel nehmen wir sehr häufig Bezug auf die Komponenten der Beobachtungen X_i. Um Verwirrungen mit der Notation zu vermeiden, nutzen wir daher nicht (X, Y) als einen Prototyp für ein Trainingsdatum, sondern (X_1, Y_1).

In Abschn. 2.1 führen wir die Modellannahme ein, diskutieren die Bedeutung der einzelnen Bestandteile und leiten daraus einige notwendige Annahmen ab. Außerdem erläutern wir Gründe für eine Standardisierung der Trainingsdaten. Wir leiten einen einfachen Ausdruck für das Excess Bayes Risk einer beliebigen Entscheidungsfunktion her und stellen zwei Beispiele vor, die wir in den folgenden Abschnitten stets als Illustration nutzen werden. Die häufig bei dieser Modellannahme verwendeten Algorithmen diskutieren wir erst ab Abschn. 2.2.

S. Richter, *Statistisches und maschinelles Lernen*,
https://doi.org/10.1007/978-3-662-59354-7_2

2.1 Die Modellannahme

Das Modell der linearen Regression setzt voraus, dass die Bayes-Regel $f^*(x)$ linear in dem Komponenten von x ist:

Modellannahme 2.1 (Lineare Regression) Mit Parametern $\beta^* = (\beta_0^*, \beta_1^*, \ldots, \beta_d^*) \in \mathbb{R}^{d+1}$ gilt:

$$f^*(x) = \beta_0^* + \sum_{j=1}^{d} \beta_j^* x_j, \qquad x \in \mathcal{X}$$

In der Terminologie von Kap. 1 ist die Modellannahme dazu äquivalent, dass

$$f^* \in \mathcal{F} = \{f : \mathbb{R}^d \to \mathbb{R} : f(x) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j \text{ mit } \beta = (\beta_0, \beta_1, \ldots, \beta_d)^T \in \mathbb{R}^{d+1}\}$$

gilt.

Sind i. i. d. Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$ gegeben und definieren wir $\varepsilon_i := Y_i - f^*(X_i)$, so gilt

$$Y_i = \beta_0^* + \sum_{j=1}^{d} \beta_j^* X_{ij} + \varepsilon_i, \quad i = 1, \ldots, n, \tag{2.1}$$

und die ε_i erfüllen $\mathbb{E}[\varepsilon_i | X_i] = 0$ $(i = 1, \ldots, n)$, da $f^*(x) = \mathbb{E}[Y_1 | X_1 = x]$. Das bedeutet, die Beobachtungen Y_i entstehen aus einer Linearkombination der X_i mit uns unbekannten Koeffizienten β_j^* $(j = 0, \ldots, d)$ und einem uns unbekannten, additiven Term ε_i. Wir können ε_i als „Messfehler" interpretieren, weil der Erwartungswert dieser Zufallsvariable null ist: $\mathbb{E}\varepsilon_i = \mathbb{E}[\mathbb{E}[\varepsilon_i | X_i]] = 0$. Im Gegensatz zu $\beta_1^*, \ldots, \beta_d^*$ nimmt β_0^* eine Sonderrolle ein und drückt den Durchschnittswert der Y_i aus, welcher nicht allein durch $X_{i1}, \ldots, X_{id}$ beschrieben werden kann. In der englischsprachigen Literatur heißt β_0^* *intercept.*

2.1.1 Standardisierung und Vermeidung von Überparametrisierung

Interpretation

Hier wollen wir uns mit der Interpretation der Parameter β_j^* im Modell beschäftigen. Da ein Algorithmus $\hat{f}_n$ zur Schätzung von f^* mittels Trainingsdaten ebenfalls in $\mathcal{F}$ liegen soll (d. h. $\hat{f}_n \in \mathcal{F}$), hat dieser notwendigerweise die Struktur

$$\hat{f}_n(x) := \hat{\beta}_0 + \sum_{j=1}^{d} \hat{\beta}_j x_j,$$

und man benötigt Schätzungen $\hat{\beta}_j$ für β_j^*. In der Praxis geben diese wichtige Informationen über Stärke des Einflusses einzelner Komponenten von X_1 auf Y_1 und erlauben *Vergleiche* der Einflussstärken zwischen verschiedenen Komponenten. Wir wollen dies zunächst theoretisch analysieren und dann die wesentlichen Schlussfolgerungen auf das praktische Vorgehen übertragen.

Für die folgenden Resultate nutzen wir die *Kovarianz* zweier Zufallsvariablen Z_1, Z_2. Ist $\mathbb{E}\left[Z_1^2\right] < \infty$ und $\mathbb{E}\left[Z_2^2\right] < \infty$, so ist diese definiert durch

$$\mathrm{Cov}(Z_1, Z_2) := \mathbb{E}\left[(Z_1 - \mathbb{E}Z_1)(Z_2 - \mathbb{E}Z_2)\right].$$

Gilt $\mathrm{Cov}(Z_1, Z_2) = 0$, so nennen wir Z_1, Z_2 *unkorreliert*. Dies ist beispielsweise der Fall, wenn Z_1, Z_2 stochastisch unabhängig sind, d. h. eine Beobachtung von Z_1 keine Informationen über den Wert von Z_2 bereitstellt. Im Allgemeinen ist die Kovarianz ein Maß für die lineare Abhängigkeit zweier Zufallsvariablen. Der folgende Satz rechtfertig diese Bedeutung:

Satz 2.2 Es sei $\mathrm{Var}(Z_1) \neq 0 \neq \mathrm{Var}(Z_2)$. Dann heißt

$$\rho(Z_1, Z_2) := \frac{\mathrm{Cov}(Z_1, Z_2)}{\sqrt{\mathrm{Var}(Z_1)\mathrm{Var}(Z_2)}} \tag{2.2}$$

Korrelation von Z_1 und Z_2, und es gilt stets $\rho(Z_1, Z_2) \in [0, 1]$. Es ist $\rho(Z_1, Z_2) = \pm 1$ genau dann, wenn $a, b \in \mathbb{R}$ existieren mit $Z_1 = aZ_2 + b$.

Das bedeutet, die Kovarianz nimmt ihren maximal möglichen Wert an, falls ein exakt linearer Zusammenhang zwischen Z_1, Z_2 besteht.

Im Modell 2.1 können folgende Gleichungen hergeleitet werden:

Lemma 2.3 Für $j = 1, \ldots, d$ gilt

$$\beta_j^* = \frac{\mathrm{Cov}(X_{1j}, Y_1 - \sum_{k\in\{1,\ldots,d\}\setminus\{j\}} \beta_k^* X_{1k})}{\mathrm{Var}(X_{1j})}. \tag{2.3}$$

Ist zusätzlich $\mathrm{Cov}(X_{1j}, X_{1k}) = 0$ für $j, k \in \{1, \ldots, d\}$, $j \neq k$, so vereinfacht sich obige Gleichung zu

$$\beta_j^* = \frac{\mathrm{Cov}(X_{1j}, Y_1)}{\mathrm{Var}(X_{1j})}. \tag{2.4}$$

Beweis Aus Gl. (2.1), der Linearität der Kovarianz und $\mathrm{Cov}(X_{1j}, \varepsilon_1) = \mathbb{E}[X_{1j}\varepsilon_1] - \mathbb{E}X_{1j}\mathbb{E}\varepsilon_1 = \mathbb{E}[X_{1j}\mathbb{E}[\varepsilon_1|X_1]] = 0$ folgt

$$\mathrm{Cov}(X_{1j}, Y_1) = \sum_{k=1}^{d} \beta_k^* \mathrm{Cov}(X_{1j}, X_{1k}).$$

Gl. (2.3) und (2.4) folgen nun durch Umformen.

Gl. (2.4) besitzt eine besonders einfache Interpretation: Sind die einzelnen Komponenten von X_1 unkorreliert, so ist β_j^* umso größer, je stärker X_{1j} von Y_1 abhängt, d. h. je größer der Einfluss einer Veränderung von X_{1j} auf die Veränderung von Y_1 ist.

In Anwendungsproblemen sind die Komponenten von X_1 üblicherweise nicht unkorreliert. Zur Verdeutlichung von Gl. (2.3) betrachten wir das folgende Beispiel:

Beispiel 2.4 Es sei $\mathcal{X} = \mathbb{R}^2$ und $X_1 = (X_{11}, X_{12})^T$. Wir wollen den Zusammenhang zwischen Blutdruck Y_1, Körpergröße X_{11} und Körpergewicht X_{12} von Menschen mittels einer linearen Regression beschreiben. In diesem Fall sind X_{11}, X_{12} nicht unkorreliert, denn bei zufällig ausgewählten Personen wird man bei höherem X_{11} im Durchschnitt auch ein höheres X_{12} erwarten. Anschaulich bedeutet das, dass einige Informationen von X_{11} über Y auch in X_{12} enthalten sind. Gl. (2.3) sagt aus, dass β_1^* misst, wie stark die Abhängigkeit von Y von den Informationen in X_{11} ist, die nicht bereits in X_{12} enthalten sind.

Allgemein gesprochen bedeutet Gl. (2.3), dass β_j^* den um die anderen Komponenten von X_1 bereinigten linearen Einfluss von X_{1j} auf Y_1 ausdrückt.

Diese Interpretation trifft nicht zu, falls eine der Komponenten von X_1 exakt durch eine Linearkombination der anderen Komponenten darstellbar ist. Um die Erläuterung möglichst einfach zu halten, betrachten wir folgenden Spezialfall:

Beispiel 2.5 Es sei $\mathcal{X} = \mathbb{R}^2$ und $X_1 = (X_{11}, X_{12})^T$, und es gelte $X_{11} = X_{12}$. Modellieren wir trotzdem

$$Y_{i1} = \beta_0^* + \beta_1^* X_{i1} + \beta_2^* X_{i2} + \varepsilon_i, \quad i = 1, \ldots, n,$$

so gilt offensichtlich für jeden weiteren Parameter $\alpha^* = \left(\alpha_0^*, \alpha_1^*, \alpha_2^*\right)^T \in \mathbb{R}^3$ mit $\alpha_1^* + \alpha_2^* = \beta_1^* + \beta_2^*$ ebenfalls:

$$Y_{i1} = \alpha_0^* + \alpha_1^* X_{i1} + \alpha_2^* X_{i2} + \varepsilon_i, \quad i = 1, \ldots, n,$$

d. h., in dem Modell kann $\beta^* = \left(\beta_0^*, \beta_1^*, \beta_2^*\right)^T$ nicht eindeutig bestimmt werden.

Eine deterministische lineare Abhängigkeit zwischen den Komponenten von X_1 führt also dazu, dass zu viele Parameter zur Beschreibung des Zusammenhangs zwischen Y_1 und X_1 eingeführt werden. Dies ist nicht wünschenswert, da dann eine sinnvolle Interpretation von β^* nicht möglich ist. In der Praxis kann dies vermieden werden, indem die betreffenden Komponenten von X_1 entfernt werden und zur Beschreibung von Y_1 nur der entsprechend reduzierte Vektor genutzt wird. In der theoretischen Beschreibung des Modells werden wir dieses Szenario im Folgenden ausschließen.

Definition 2.6 (Überparametrisierung)
Das Modell 2.1 heißt *überparametrisiert* und die Parameter $\beta_0^*, \beta_1^*, \ldots, \beta_d^*$ *nicht identifizierbar,* falls mindestens ein $j \in \{1, \ldots, d\}$ existiert, so dass

$$X_{1j} = \sum_{k \in \{1,\ldots,d\} \setminus \{j\}} \lambda_k X_{1k} \quad \text{mit geeigneten } \lambda_k \in \mathbb{R}. \qquad \blacklozenge$$

Eine kompaktere Beschreibung der Überparametrisierung ist mittels der Kovarianzmatrix von X_1 möglich:

Definition 2.7
Die symmetrische Matrix $\Sigma \in \mathbb{R}^{d \times d}$ mit den Einträgen

$$\Sigma_{jk} := \mathrm{Cov}(X_{1j}, X_{1k}), \qquad j, k \in \{1, \ldots, d\}$$

heißt *Kovarianzmatrix* von X_1. $\blacklozenge$

Man kann folgenden Satz zeigen:

Satz 2.8 Die Matrix Σ ist positiv semidefinit. Das Modell 2.1 ist genau dann *nicht* überparametrisiert, wenn Σ positiv definit ist.

Beweis Ist $v = (v_1, \ldots, v_d)^T \in \mathbb{R}^d$ beliebig, so gilt $v^T \Sigma v = \sum_{j,k=1}^d v_j \,\mathrm{Cov}(X_{1j}, X_{1k}) v_k$ $= \mathbb{E}\left[\left(\sum_{j=1}^d v_j (X_{1j} - \mathbb{E}X_{1j})\right)^2\right] \geq 0$. Gibt es mindestens eine Linearkombination $v \in \mathbb{R}^d$ mit $\sum_{j=1}^d v_j X_{1j} = 0$, so folgt $v^T \Sigma v = 0$.

Im Folgenden gehen wir stets davon aus, dass Σ positiv definit ist.

Normierung

Um die Größe der β_j^* $(j = 1, \ldots, d)$ untereinander vergleichen und daraus Schlüsse über die Stärke des Einflusses von X_{1j} auf Y_1 ziehen zu können, ist eine Standardisierung der Trainingsdaten notwendig. Eine übliche Forderung ist:

Definition 2.9 (Normierung)
Der Vektor X_1 heißt *normiert,* falls für alle $j = 1, \ldots, d$ gilt: $\mathrm{Var}(X_{1j}) = 1$. $\blacklozenge$

Ist X_1 normiert und gilt zusätzlich $\mathrm{Cov}(X_{1j}, X_{1k}) = 0$ für $j, k \in \{1, \ldots, d\}$, $j \neq k$, so ist laut Lemma 2.3:

$$\beta_j^* = \underbrace{\rho(X_{1j}, Y_1)}_{\in[-1,1]} \cdot \sqrt{\mathrm{Var}(Y_1)}, \tag{2.5}$$

d. h., alle β_j^*, $j = 1, \ldots, d$ sind von der gleichen Größenordnung, geben die (um $\mathrm{Var}(Y_1)$ skalierte) Korrelation von X_{1j} mit Y_1 an und erlauben somit einen tatsächlichen Vergleich der Einflüsse der verschiedenen Komponenten von X_{1j} auf Y_1. Im Falle der Korreliertheit gibt es eine ähnliche Darstellung

$$\frac{\beta_j^*}{\sqrt{\left(\beta_j^*\right)^2 + \mathrm{Var}(\varepsilon_1)}} = \rho\left(X_{1j}, Y_1 - \sum_{k \in \{1,\ldots,d\}\setminus\{j\}} \beta_k^* X_{1k}\right).$$

Die Annahme der Standardisierung an den Vektor X_1 ist in der Praxis nicht erfüllt und muss durch eine Manipulation der Daten geschehen, vgl. Bemerkung 2.13.

Zentrierung

Im Modell 2.1 gilt zunächst

$$\mathbb{E}Y_1 = \beta_0^* + \sum_{j=1}^{d} \beta_j^* \mathbb{E}X_{1j}.$$

Eine anschaulichere Bedeutung von β_0^* ist nur gewährleistet, wenn $\mathbb{E}X_{1j} = 0$ ($j = 1, \ldots, d$) gilt, denn dann stellt β_0^* den Erwartungswert der Beobachtungen Y_i dar. Auch beim Beweisen theoretischer Resultate wirkt diese Bedingung vereinfachend, weil dann $\Sigma_{jk} = \mathrm{Cov}(X_{1j}, X_{1k}) = \mathbb{E}[X_{1j}X_{1k}]$ gilt.

Definition 2.10 (Zentrierung)
Der Vektor X_1 heißt *zentriert,* falls für alle $j = 1, \ldots, d$ gilt: $\mathbb{E}X_{1j} = 0$. ♦

Standardisierung, Durchführung in der Praxis

Definition 2.11 (Standardisierung)
Ein Vektor X_1 heißt *standardisiert,* falls er normiert und zentriert ist. ♦

Erfüllt (X_1, Y_1) die Modellannahme 2.1, ist aber nicht standardisiert, so kann man durch eine einfache Modifikation von X_1 eine Zufallsvariable $\tilde{X}_1$ erhalten, welche standardisiert ist:

Lemma 2.12 Erfüllt (X_1, Y_1) die Modellannahme 2.1, so erfüllt auch $(\tilde{X}_1, Y_1)$ mit $\tilde{X}_1 := (\tilde{X}_{11}, \ldots, \tilde{X}_{1d})^T$ und

$$\tilde{X}_{1j} := \frac{X_{1j} - \mathbb{E}X_{1j}}{\sqrt{\mathrm{Var}(X_{1j})}}$$

die Modellannahme 2.1 mit neuen Parametern $\tilde{\beta}_0^* := \beta_0^* + \sum_{j=1}^d \beta_j^* \mathbb{E}X_{1j}$, $\tilde{\beta}_j^* := \beta_j^* \cdot \sqrt{\mathrm{Var}(X_{1j})}$ $(j = 1, \ldots, d)$, und $\tilde{X}_1$ ist standardisiert.

Beweis Es ist $Y_1 = \beta_0^* + \sum_{j=1}^d \beta_j^* X_{1j} + \varepsilon_1 = \left(\beta_0^* + \sum_{j=1}^d \beta_j^* \mathbb{E}X_{1j}\right) + \sum_{j=1}^d \left(\beta_j^* \sqrt{\mathrm{Var}(X_{1j})}\right) \cdot \tilde{X}_{1j}$.

In der Praxis sind $\mathrm{Var}(X_{1j})$ und $\mathbb{E}X_{1j}$ nicht bekannt. Stattdessen verwendet man die auf den erhaltenen Trainingsdaten basierenden empirischen Entsprechungen

$$\hat{M}_j = \frac{1}{n}\sum_{i=1}^n X_{ij}, \qquad \hat{S}_j := \sqrt{\frac{1}{n}\sum_{i=1}^n \left(X_{ij} - \hat{M}_j\right)^2}, \tag{2.6}$$

für welche nach dem starken Gesetz der großen Zahlen gilt: $\hat{M}_j \to \mathbb{E}X_{1j}, \hat{S}_j \to \sqrt{\mathrm{Var}(X_{1j})}$. Wir fassen die bisherigen Beobachtungen zusammen in folgender Anleitung:

Bemerkung 2.13 (Herstellung der Standardisierung und Interpretation) Gegeben seien Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$:

1. Zentriere und standardisiere die X_i: Für $j = 1, \ldots, d$, setze $\hat{M}_j, \hat{S}_j$ wie in Gl. (2.6) und setze
$$\tilde{X}_{ij} := \frac{X_{ij} - \hat{M}_j}{\hat{S}_j}, \quad i = 1, \ldots, n$$
sowie $\tilde{X}_i := (\tilde{X}_{i1}, \ldots, \tilde{X}_{id})^T$.
2. Erhalte einen Algorithmus $\tilde{f}_n(\tilde{x}) = \tilde{\beta}_0 + \sum_{j=1}^d \tilde{\beta}_j \tilde{x}_j$ basierend auf den Trainingsdaten $(\tilde{X}_i, Y_i)$. Neue Beobachtungen x müssen zunächst mittels $\tilde{x}_j = \frac{x_j - \hat{M}_j}{\hat{S}_j}$ transformiert werden.
3. Interpretiere den Einfluss der verschiedenen Komponenten von X_1 auf Y_1 mittels der $\tilde{\beta}_j$ vergleichbarer Größe.
4. Erhalte einen Algorithmus $\hat{f}_n(x) = \hat{\beta}_0 + \sum_{j=1}^d \hat{\beta}_j x_j$ für nichttransformierte Beobachtungen x durch

$$\hat{\beta}_0 := \tilde{\beta}_0^* - \sum_{j=1}^d \hat{\beta}_j \hat{M}_j, \qquad \hat{\beta}_j := \frac{\tilde{\beta}_j}{\hat{S}_j}, \quad (j = 1, \ldots, d).$$

Prinzipiell sind die ab Abschn. 2.2 vorgestellten Algorithmen auch auf nichtstandardisierte Trainingsdaten anwendbar. Wie wir sehen werden, basieren einige Algorithmen allerdings direkt auf der Bewertung der Größen der β_j^*. Werden solche Algorithmen auf standardisierte Beobachtungen angewandt, liegt das Augenmerk dann stärker auf einem Vergleich der „Einflüsse" der verschiedenen X_{1j} auf Y_1, und die erhaltenen Schätzungen $\tilde{\beta}_j$ sind einfacher zu interpretieren.

2.1.2 Einordnung der Modellannahme, Beispiele

Die Modellannahme 2.1 setzt einen linearen Zusammenhang zwischen Y_1 und den Komponenten X_1 voraus und ist daher nur für relativ spezielle Problemstellungen zutreffend. Im Gegensatz zu vielen anderen Modellen erlaubt ein Algorithmus hier jedoch auch direkt eine *Interpretation* in Hinsicht auf die Auswirkungen einzelner Komponenten von X_1 auf Y_1, vgl. Gl. (2.5). Aufgrund der Einfachheit des Modells gibt es eine gut ausgearbeitete und theoretisch untermauerte Theorie der Bestimmung „wichtiger" Komponenten von X_1 (sogenannte Modellwahl, engl. *model selection* oder *feature extraction*) auch für Probleme mit sehr hoher Dimension d, vgl. zum Beispiel die theoretischen Aussagen über Lasso-Schätzer in Abschn. 2.4. Ein Anwendungsbeispiel für die komplexeren Algorithmen der linearen Regression ist die Identifizierung von für eine Erkrankung verantwortlichen Genen. Hier entspricht die „Stärke" der Ausprägung für jedes einzelne Gen einer Komponente von X_1 und Y_1 einem zuvor bestimmten Erkrankungsrisiko der Person. In diesem Fall ist d (Anzahl der untersuchten Gene) wesentlich größer als n (Anzahl der Testpersonen). Als einfaches Modell dieses Problems betrachten wir das folgende Beispiel:

Beispiel 2.14 (Hochdimensionale Trainingsdaten) Die Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$ seien gegeben durch folgendes Modell:

$$Y_i = \beta_0^* + \sum_{j=1}^{d} \beta_j^* X_{ij} + \varepsilon_i, \quad i = 1, \ldots, n,$$

wobei $X_{ij} \sim U[0, 1]$ unabhängig und gleichverteilt auf $[0, 1]$ $(i = 1, \ldots, n, j = 1, \ldots, d)$ und die Messfehler $\varepsilon_i \sim N(0, \sigma^2)$ unabhängig normalverteilt mit $\sigma > 0$ sind.

Im Folgenden betrachten wir das Modell mit Dimension $d = 10$, es gelte

$$\beta_j^* := \begin{cases} 1, & j = 1, \\ -1, & j = 3, \\ 0{,}2, & j = 5, \\ 0, & j \notin \{1, 3, 5\}, \end{cases}$$

d. h., nur drei Komponenten von X_1 gehen tatsächlich in den Wert von Y_1 ein, und die zu schätzende Bayes-Regel ist $f : \mathbb{R}^d \to \mathbb{R},\ f(x) = x_1 - x_3 + 0{,}2x_5$.

Erwartet man komplexere Zusammenhänge zwischen X_1 und Y_1, so ist das Modell linearer Regression nicht angemessen. Soll beispielsweise die Anzahl der Menschen auf einem Graustufenbild bestimmt werden, so wählt man Y_1 als die Anzahl der Menschen und X_1 als das Graustufenbild, repräsentiert durch die Graustufenwerte der spaltenweise ausgelesenen Pixel. In diesem Fall erwartet man nicht, dass einzelne Komponenten von X_1 (d. h. einzelne Pixel auf dem Bild) besondere Auswirkungen auf Y_1 haben, sondern ein komplexes Zusammenspiel vieler Pixel. Im Prinzip ist es möglich, auch komplexere funktionale Zusammenhänge mit Hilfe der linearen Regression nachzubilden, vgl. Beispiel 1.29(ii): Dafür definiert man neue Beobachtungen $\tilde{X}_i := h(X_i)$, $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ und nimmt an, dass $\tilde{X}_i$ die Modellannahme 2.1 erfüllt. Im einfachen Fall $X_i \in \mathbb{R}$, $h(x) = (x, x^2, x^3, \ldots, x^{10})$ erhalten wir folgendes Modell, um die Bayes-Regel $f^* : \mathbb{R} \to \mathbb{R}$ durch Polynome vom Grad 10 zu beschreiben:

Beispiel 2.15 (Funktionaler Zusammenhang) Die Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$ seien gegeben durch folgendes Modell:

$$Y_i = \beta_0^* + \sum_{j=1}^{d} \beta_j^* X_i^j + \varepsilon_i, \quad i = 1, \ldots, n,$$

wobei $X_i \sim U[0, 1]$ unabhängig und gleichverteilt auf $[0, 1]$ verteilt und die Messfehler $\varepsilon_i \sim N(0, \sigma^2)$ unabhängig normalverteilt mit $\sigma = 0{,}3$ sind. Es sei $d = 10$ und

$$\beta_j^* := \begin{cases} 1, & j = 1, \\ -1, & j = 3, \\ 0{,}2, & j = 5, \\ 0, & j \notin \{1, 3, 5\}, \end{cases}$$

d. h., die zu schätzende Bayes-Regel ist $f^* : \mathbb{R} \to \mathbb{R}$, $f^*(x) = x - x^3 + 0{,}2x^5$.

Tatsächlich ist solch eine Erweiterung aber mit einem großen Anstieg der Dimension der Beobachtungen $\tilde{X}_i$ verbunden und wirft die Frage nach der Auswahl geeigneter Funktionen h auf; detaillierte Bemerkungen dazu finden sich in Abschn. 4.2.

Wir werden sehen (vgl. Satz 2.36), dass eine hohe Korrelation zwischen den Komponenten von X_1 (d. h., falls Σ weit von einer Diagonalmatrix entfernt ist) im Allgemeinen zu schlechteren Konvergenzraten der hier vorgestellten Algorithmen führt. Während in Beispiel 2.14 die Komponenten von X_1 als unabhängig modelliert werden (was für Genausprägungen möglicherweise annähernd zutrifft), so sind die Komponenten von $\tilde{X}_1$, erhalten durch

Modellerweiterungen (wie in Beispiel 2.14), sehr stark abhängig und liefern möglicherweise sogar nicht invertierbare Σ.

Das Beispiel 2.14 hat den Vorteil, dass die Ergebnisse der Algorithmen leicht grafisch veranschaulicht werden können. Wir werden es daher vor allem zur Illustration und als Negativbeispiel für vorliegende Abhängigkeiten zwischen den Komponenten von X_1 und deren Auswirkungen auf die verschiedenen Algorithmen nutzen.

2.1.3 Vereinfachung des Modells und Bayes-Risiko

Bemerkung 2.16 Aus didaktischen Gründen werden wir die theoretische Behandlung der Schätzer für den *intercept* β_0^* nicht durchführen, da dessen Berücksichtigung im Allgemeinen zu unübersichtlicheren Beweisen führt. Für die theoretischen Resultate gehen wir daher stets davon aus, dass $\beta_0^* = 0$. Das bedeutet, dass

$$f^* \in \mathcal{F}_0 := \left\{ f : \mathbb{R}^d \to \mathbb{R} : f(x) = \sum_{j=1}^{d} \beta_j x_j \text{ mit } \beta = (\beta_1, \ldots, \beta_d)^T \in \mathbb{R}^d \right\}$$

gilt und die Verfahren entsprechend für die Ermittlung von Komponenten $\beta^* = (\beta_1^*, \ldots, \beta_d^*)^T \in \mathbb{R}^d$ vereinfacht worden sind.

Im Modell 2.1 sind das Bayes-Risiko und die Differenz des Risikos einer beliebigen Entscheidungsregel zum Bayes-Risiko explizit berechenbar.

Lemma 2.17 (Bayes-Risiko) Es gelte die Modellannahme 2.1 mit $f^* \in \mathcal{F}_0$. Dann gilt:

1. $R(f^*) = \sigma^2 := \mathrm{Var}(\varepsilon_1)$.
2. Ist $f \in \mathcal{F}_0$ eine beliebige Entscheidungsregel mit $f(x) = \sum_{j=1}^{d} \beta_j x_j$ mit $\beta = (\beta_1, \ldots, \beta_d)^T \in \mathbb{R}^d$, so gilt $R(f) - R(f^*) = \|\Sigma^{1/2}(\beta - \beta^*)\|_2^2$.

Beweis

(i) Wegen $\mathbb{E}\varepsilon_1 = 0$ gilt

$$R(f^*) = \mathbb{E}[(Y_1 - f^*(X_1))^2] = \mathbb{E}[\varepsilon_1^2] = \mathrm{Var}(\varepsilon_1).$$

(ii) Wegen $\mathbb{E}[\varepsilon_1 | X_1] = 0$ gilt

$$\begin{aligned} R(f) - R(f^*) &= \mathbb{E}[(X_1^T \beta - Y)^2] - \mathbb{E}[\varepsilon_1^2] = \mathbb{E}[(X_1^T(\beta - \beta^*) + \varepsilon_1)^2] - \mathbb{E}[\varepsilon_1^2] \\ &= \mathbb{E}[(X_1^T(\beta - \beta^*))^2] = (\beta - \beta^*)^T \mathbb{E}[X_1 X_1^T](\beta - \beta^*) = \|\Sigma^{1/2}(\beta - \beta^*)\|_2^2. \end{aligned}$$

2.2 Der Kleinste-Quadrate-Schätzer

2.2.1 Definition und explizite Darstellung

Zur Bestimmung eines Algorithmus nutzen wir den allgemeinen Ansatz aus Bemerkung 1.14, d. h., wir minimieren

$$\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f(X_i))$$

über alle $f \in \mathcal{F}$. Da jedes $f \in \mathcal{F}$ durch ein $\beta = (\beta_0, \beta_1, \ldots, \beta_d)^T \in \mathbb{R}^{d+1}$ beschrieben wird, kann die Minimierung auf diese Parameter verlagert werden:

Definition 2.18 (Kleinste-Quadrate-Schätzer)
Sei

$$\hat{R}_n(\beta) := \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \beta_0 - \sum_{j=1}^{d}\beta_j X_{ij}\right)^2.$$

Der Schätzer

$$\hat{\beta}^{KQ} :\in \operatorname*{arg\,min}_{\beta\in\mathbb{R}^{d+1}} \hat{R}_n(\beta)$$

heißt *Kleinste-Quadrate-Schätzer (KQ-Schätzer)* für β^*. Der zugehörige Algorithmus ist

$$\hat{f}_n^{KQ}(x) = \hat{\beta}_0^{KQ} + \sum_{j=1}^{d}\hat{\beta}_j^{KQ} x_j.$$

♦

Wir formulieren nun eine explizite Darstellung des Algorithmus $\hat{f}_n^{KQ}$. Zur kompakten Formulierung führen wir zunächst einige neue Größen ein:

Lemma 2.19 (Kompakte Form des KQ-Schätzers) Die durch

$$\mathbb{X} := \begin{pmatrix} 1 & X_1^T \\ \vdots & \vdots \\ 1 & X_n^T \end{pmatrix} \in \mathbb{R}^{n\times(d+1)}, \qquad \mathbb{Y} := \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix} \in \mathbb{R}^n, \qquad \mathbb{e} := \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix} \in \mathbb{R}^n$$

definierten Größen heißen *Designmatrix* $\mathbb{X}$, *Regressionsvektor* $\mathbb{Y}$ und *Residuenvektor* $\mathbb{e}$. In diesem Fall ist Gl. (2.1) äquivalent zu

$$\mathbb{Y} = \mathbb{X}\beta^* + \mathbb{e}.$$

Für den KQ-Schätzer gelten folgende Aussagen:

- Es gilt die *Normalengleichung* $(\mathbb{X}^T\mathbb{X})\hat{\beta}^{KQ} = \mathbb{X}^T\mathbb{Y}$.
- Hat $\mathbb{X}$ vollen Spaltenrang, so ist

$$\hat{\beta}^{KQ} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}. \tag{2.7}$$

Beweis Es ist $\hat{R}_n(\beta) = (\mathbb{Y}-\mathbb{X}^T\beta)^T(\mathbb{Y}-\mathbb{X}^T\beta)$. Die einzige Extremstelle ist gegeben durch

$$0 = \partial_\beta \hat{R}_n(\hat{\beta}^{KQ}) = 2\mathbb{X}^T\mathbb{Y} - 2\mathbb{X}^T\mathbb{X}\hat{\beta}^{KQ},$$

dies liefert die Normalengleichung. Die restlichen Aussagen folgen durch Umformen.

Bemerkungen

- Sei $\tilde{X}_i := (1, X_i^T)^T$, $i = 1, \ldots, n$. Ist X_1 stetig verteilt, Σ positiv definit und $d+1 \leq n$, so sind alle $(d+1)$-elementigen Teilmengen aus $\{\tilde{X}_1, \ldots, \tilde{X}_n\}$ $\mathbb{P}$-f. s. linear unabhängig. In diesem Falle ist $\mathbb{X}^T\mathbb{X} = \sum_{i=1}^n \tilde{X}_i\tilde{X}_i^T$ $\mathbb{P}$-f. s. positiv definit und der Schätzer $\hat{\beta}^{KQ}$ durch Gl. (2.7) definiert.
- Im Falle $d+1 > n$ liegen mindestens genauso viele Parameter $\beta_0, \beta_1, \ldots, \beta_d$ wie durch $\hat{R}_n(\beta) = 0$ verursachte Zwangsbedingungen vor, so dass in diesem Fall stets $\hat{R}_n(\hat{\beta}^{KQ}) = 0$ erreicht werden kann und $\hat{f}_n^{KQ}$ überangepasst an die Trainingsdaten ist. Außerdem sind in diesem Falle die Vektoren $\tilde{X}_1, \ldots, \tilde{X}_n$ linear abhängig, so dass $\mathbb{X}^T\mathbb{X}$ nicht invertierbar und eine Darstellung von $\hat{\beta}^{KQ}$ durch Gl. (2.7) nicht möglich ist.

Wir schließen, dass nur im Falle $d+1 \leq n$ ein niedriges Excess Bayes Risk für $\hat{f}_n^{KQ}$ erwartet werden kann. Eine entsprechende theoretische Aussage wird durch Satz 2.36 geliefert.

Zur Visualisierung zeigen wir zunächst das Verhalten von $\hat{f}_n^{KQ}$ für Trainingsdaten der Dimensionen $d \in \{1, 2\}$:

Beispiel 2.20 (Niedrigdimensionale lineare Zusammenhänge) Gegeben sind $n = 100$ i. i. d. Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$, die sich gemäß der folgenden linearen Regressionsmodelle entwickeln: Es sei jeweils $\varepsilon_i \sim N(0, 1)$, $X_{ij} \sim U[0, 1]$ seien gleichverteilt auf $[0, 1]$, und Y_i entsteht durch

(i) $Y_i = \beta_0^* + \beta_1^* X_{i1} + \varepsilon_i$ mit $X_i = X_{i1} \in \mathbb{R}$, $\beta^* = (-1, 3)^T$
(ii) $Y_i = \beta_0^* + \beta_1^* X_{i1} + \beta_2^* X_{i2} + \varepsilon_i$ mit $X_i = (X_{i1}, X_{i2})^T \in \mathbb{R}^2$, $\beta^* = (-1, 3, 4)^T$.

In Abb. 2.1 sind die erhaltenen Entscheidungsregeln $\hat{f}_n^{KQ}$ dargestellt, die einer Gerade in (i) bzw. einer Hyperebene in (ii) entsprechen. In diesen niedrigdimensionalen Beispielen liefert $\hat{f}_n^{KQ}$ auch zahlenmäßig gut mit β^* übereinstimmende Schätzungen: In (i) erhalten wir: $\hat{\beta}^{KQ} = (-1{,}18, 3{,}31)^T$, und in (ii): $\hat{\beta}^{KQ} = (-1{,}06, 3{,}27, 3{,}89)^T$.

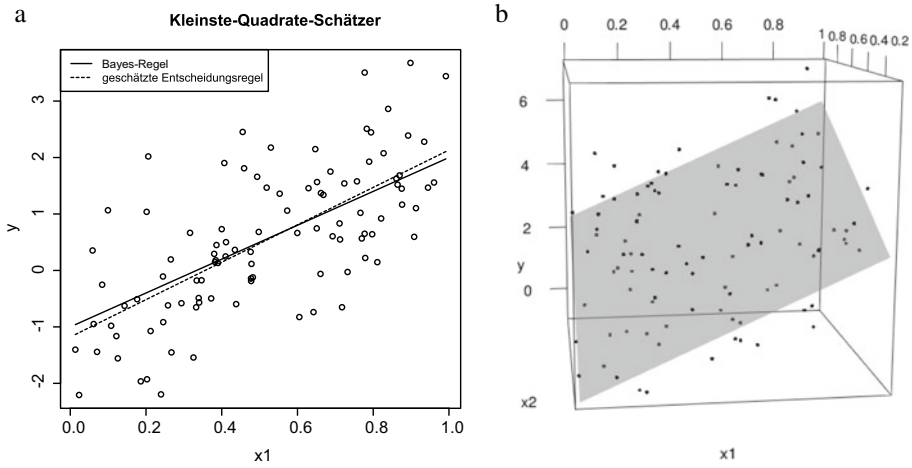


Abb. 2.1 Anwendung von $\hat{f}_n^{KQ}$ auf Trainingsdaten aus Beispiel 2.20. **a** Beispiel 2.20(i). **b** Beispiel 2.20(ii). Der Algorithmus entspricht jeweils einer Gerade bzw. Hyperebene, welche geeignet durch die Trainingsdaten gelegt wird

Das folgende Beispiel illustriert das Verhalten von $\hat{f}_n^{KQ}$, falls die Modellannahme verletzt ist:

Beispiel 2.21 (Nichtlineare Zusammenhänge) Gegeben sind $n = 100$ i. i. d. Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$ gemäß folgendem Modell: $\varepsilon_i \sim N(0, 0{,}3^2)$, und $X_i \sim U[0, 1]$ seien gleichverteilt auf $[0, 1]$ mit

$$Y_i = -1 + X_i^3 + \varepsilon_i, \quad i = 1, \ldots, n.$$

In Abb. 2.2 ist $\hat{f}_n^{KQ}$ basierend auf (X_i, Y_i), $i = 1, \ldots, n$ dargestellt; man erhält $\hat{\beta}^{KQ} = (-1{,}24, 0{,}93)^T$. Aufgrund der falschen Modellannahme (hier ist $f^*(x) = -1 + x^3$, d. h. $f^* \notin \mathcal{F}$ mit $d = 1$), ist $\hat{f}_n^{KQ}$ systematisch von f^* verschieden. Der Algorithmus versucht stattdessen, die bestmögliche Annäherung an f^* innerhalb von $\mathcal{F}$ zu finden, d. h. mittels Entscheidungsregeln der Form $f(x) = \beta_0^* + \beta_1^* x$.

Beispiel 2.22 Wir betrachten jeweils $n = 100$ Trainingsdaten aus Beispiel 2.14 ($\hat{f}_n^{KQ,1}$) und 2.15 ($\hat{f}_n^{KQ,2}$) mit $d = 10$, $\sigma = 0{,}3$. Wir erhalten die Schätzer

$$\begin{aligned}\hat{\beta}^{KQ,1} &= (0{,}07, 1{,}07, -0{,}08, -1{,}11, -0{,}03, 0{,}08, -0{,}03, 0{,}07, -0{,}02, -0{,}06, 0{,}10)^T,\\ \hat{\beta}^{KQ,2} &= (-2{,}7 \cdot 10^{-1}, 5{,}1 \cdot 10^{-1}, 1{,}9 \cdot 10^2, -2{,}7 \cdot 10^3, 1{,}8 \cdot 10^4, -6{,}3 \cdot 10^4,\\ &\qquad 1{,}4 \cdot 10^5, -1{,}9 \cdot 10^5, 1{,}6 \cdot 10^5, -7{,}7 \cdot 10^4, 1{,}5 \cdot 10^4)^T,\end{aligned}$$

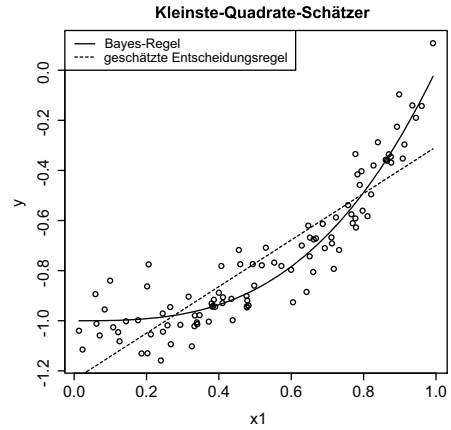


Abb. 2.2 Anwendung von $\hat{f}_n^{KQ}$ auf Trainingsdaten aus Beispiel 2.21, welche nicht der Modellannahme genügen

sowie Testfehler (ermittelt durch neu erzeugte Beobachtungen des Modells)

$$R\left(\hat{f}_n^{KQ,1}\right) \approx 0{,}0974, \quad R\left(\hat{f}_n^{KQ,2}\right) \approx 0{,}0952.$$

Im Vergleich zum Bayes-Risiko $R(f^*) = \sigma^2 = 0{,}09$ verhalten sich die Risiken $R\left(\hat{f}_n^{KQ,i}\right)$, $i = 1, 2$ also in beiden Beispielen sehr gut.

Eine grafische Veranschaulichung für $\hat{f}_n^{KQ,2}$ ist in Abb. 2.3 zu sehen. Offensichtlich gilt $\hat{f}_n^{KQ,2} \approx f^*$, allerdings verändert $\hat{f}_n^{KQ,2}$ sein Verhalten häufiger. Dies ist Ausdruck der Überanpassung, die durch das bereits relativ große Verhältnis von $\frac{d}{n} \approx 0{,}1$ eingetreten ist. Die Schätzung $\hat{\beta}^{KQ,2}$ von $\beta^* = (0, 1, 0, -1, 0, 0{,}2, 0, 0, 0, 0, 0)^T$ ist jedoch sehr schlecht. Grund hierfür ist die hohe Korrelation der Komponenten X_i^j, $j = 1, \ldots, d$ untereinander. Im Detail diskutieren wir dieses Verhalten nach Satz 2.36.

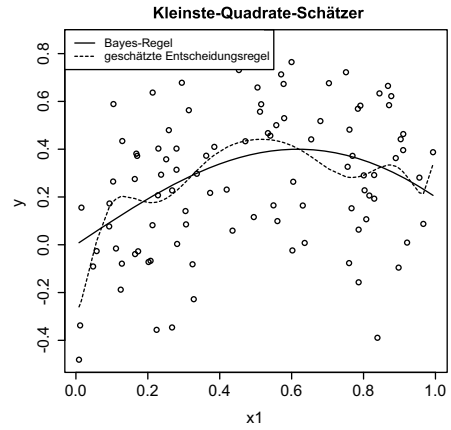


Abb. 2.3 Anwendung von $\hat{f}_n^{KQ}$ auf $n = 100$ Trainingsdaten aus Beispiel 2.15

2.2.2 Theoretische Resultate

Bemerkung 2.16 gilt entsprechend für diesen Abschnitt. Insbesondere ist hier $\hat{\beta}^{KQ} = \arg\min_{\beta\in\mathbb{R}^d} \hat{R}_n((0,\beta^T)^T) = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$ (falls $d \leq n$) mit modifiziertem $\mathbb{X} = (X_1^T,\ldots,X_n^T)^T \in \mathbb{R}^{n\times d}$.

Wir betrachten hier ein Resultat aus [18, Theorem 11, angewandt mit $t = \log(d)$] über die Konvergenzrate des Excess Bayes Risks:

Satz 2.23 Es gelte Modellannahme 2.1 mit $\beta_0^* = 0$, und X_1 sei zentriert. Es gelte $\varepsilon_1 \sim N(0,\sigma^2)$ und $\frac{1}{\sqrt{d}}\|\Sigma^{-1/2}X_1\|_2 \leq \rho_0$ f. s. mit einer Konstanten $\rho_0 > 0$ unabhängig von d. Dann gilt: Es gibt eine Konstante $c = c(\sigma^2,\rho_0^2)$, so dass für $d \geq \log(n)$ und $\frac{n}{d\log(d)} \to \infty$ gilt:

$$\limsup_{n\to\infty} \mathbb{P}\left(R\left(\hat{f}_n^{KQ}\right) - R(f^*) \geq c\cdot\frac{d}{n}\right) = 0 \tag{2.8}$$

Beweis Wir geben nur eine Motivation. Wir schreiben zur Abkürzung $\hat{\beta} = \hat{\beta}^{KQ}$. Mit Lemma 2.19 folgt:

$$\hat{\beta} - \beta^* = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y} - (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{X}\beta^* = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{e}$$

Sei $\tilde{T}_n := (X_i)_{i=1,\ldots,n}$. Dann gilt mit dem „Trick“ $a = \mathrm{Sp}(a)$ für $a \in \mathbb{R}$ und den Eigenschaften der Spur:

$$\begin{aligned}
\mathbb{E}\left[R\left(\hat{f}_n^{KQ}\right)|\tilde{T}_n\right] - R(f^*) = \quad & \mathbb{E}\left[\|\Sigma^{1/2}(\hat{\beta}-\beta^*)\|_2^2|\tilde{T}_n\right] \\
= \quad & \mathbb{E}\left[\mathbb{e}^T\mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\Sigma(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{e}|\tilde{T}_n\right] \\
= \quad & \mathrm{Sp}\left(\mathbb{E}[\mathbb{e}\mathbb{e}^T]\mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\Sigma(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\right) \\
\overset{\mathbb{E}[\mathbb{e}\mathbb{e}^T]=\sigma^2\cdot I_{n\times n}}{=} \quad & \frac{\sigma^2}{n}\cdot\mathrm{Sp}\left(\Sigma\left(\frac{1}{n}\mathbb{X}^T\mathbb{X}\right)^{-1}\right)
\end{aligned}$$

Wegen $\mathbb{E}\left[\frac{1}{n}\mathbb{X}^T\mathbb{X}\right] = \mathbb{E}\left[X_1X_1^T\right] = \Sigma$ erwarten wir $\Sigma\left(\frac{1}{n}\mathbb{X}^T\mathbb{X}\right)^{-1} \approx I_{d\times d}$ und damit $\mathrm{Sp}\left(\Sigma\left(\frac{1}{n}\mathbb{X}^T\mathbb{X}\right)^{-1}\right) \approx d$. Die exakte mathematische Behandlung des Faktors ist jedoch schwieriger und wird hier übersprungen. Grob gesprochen definiert man „günstige Ereignisse“ $A_n := \left\{\frac{1}{n}\mathbb{X}^T\mathbb{X}$ ist „nahe“ an $\Sigma\right\}$ und erhält mit der Markov-Ungleichung für beliebiges $\gamma > 0$:

$$\begin{aligned}\mathbb{P}\left(R\left(\hat{f}_n^{KQ}\right) - R(f^*) > \gamma\right) &\leq \mathbb{E}\left[\mathbb{P}\left(R\left(\hat{f}_n^{KQ}\right) - R(f^*) > \gamma | \tilde{T}_n\right) \mathbb{1}_{A_n}\right] + \mathbb{P}\left(A_n^c\right) \\ &\leq \mathbb{E}\left[\frac{\mathbb{E}\left[R\left(\hat{f}_n^{KQ}\right) - R(f^*) | \tilde{T}_n\right]}{\gamma} \mathbb{1}_{A_n}\right] + \mathbb{P}\left(A_n^c\right) \\ &\leq \frac{\sigma^2}{n}\mathbb{E}\left[\mathrm{Sp}\left(\Sigma\left(\frac{1}{n}\mathbb{X}^T\mathbb{X}\right)^{-1}\right)\mathbb{1}_{A_n}\right] + \mathbb{P}\left(A_n^c\right)\end{aligned}$$

Bemerkungen

- Die Kernaussage von Satz 2.36 ist, dass $\hat{f}_n^{KQ}$ mit Konvergenzrate $\frac{d}{n}$ lernt, d. h., nur für $d \ll n$ ist eine Konvergenz von $R(\hat{f}_n^{KQ})$ gegen das Bayes-Risiko zu erwarten. Damit liefert Satz 2.36 die Bestätigung für die nach Lemma 2.19 anschaulich hergeleitete Vermutung. Die Bedingung $d \geq \log(n)$ ist nicht notwendig, sondern wird nur für die Angabe einer möglichst einfachen Konvergenzrate genutzt.
- Die Annahme normalverteilter Fehler $\varepsilon_i \sim N(0, \sigma^2)$ kann durch allgemeinere Bedingungen ersetzt werden, genauso wie die Forderung $\frac{1}{\sqrt{d}}\|\Sigma^{-1/2}X_1\|_2 \leq \rho_0$, vgl. [18]. Die Konstante ρ_0 misst die Streuung von X_1, welche nicht durch eine Standardisierung mit der Varianz von X_1 neutralisiert werden kann. Je stärker die Verteilung von X_1 von einer Normalverteilung abweicht, desto stärker hängt ρ_0 noch von Σ ab. Je wahrscheinlicher große Werte für X_1 sind, desto größer ist ρ_0.
- Die Konstante c ist in etwa von der Form $c \approx \sigma^2\rho_0^2$. Das Excess Bayes Risk von $\hat{f}_n^{KQ}$ wächst also mit der Varianz der Fehler ε_1 und mit der Streuung von X_1.
- Wie aus dem Beweis ersichtlich ist, gilt $\mathbb{E}\left[\left(\hat{\beta}^{KQ} - \beta^*\right)^2 | \tilde{T}_n\right] = \frac{\sigma^2}{n}\mathrm{Sp}\left(\left(\frac{1}{n}\mathbb{X}^T\mathbb{X}\right)^{-1}\right) \approx \frac{\sigma^2}{n}\mathrm{Sp}(\Sigma^{-1})$. Im Gegensatz zum Excess Bayes Risk

$$\mathbb{E}\left[R(\hat{f}_n^{KQ}) | \tilde{T}_n\right] - R(f^*) = \frac{\sigma^2}{n} \cdot \mathrm{Sp}\left(\Sigma\left(\frac{1}{n}\mathbb{X}^T\mathbb{X}\right)^{-1}\right) \approx \frac{\sigma^2}{n}d$$

  hängt die Konvergenzrate von den Parametern $\hat{\beta}^{KQ} - \beta^*$ *zusätzlich* noch von der Beschaffenheit von Σ ab. Gerade bei stark korrelierten X_1 ist Σ nur „sehr knapp" invertierbar, und das Inverse Σ^{-1} enthält sehr große Werte auf der Diagonalen, was zu einem hohen Faktor $\mathrm{Sp}(\Sigma^{-1})$ führt. Dies erklärt die schlechte Schätzung von $\hat{\beta}^{KQ,2}$ in Beispiel 2.22.

Im Folgenden wollen wir eine Abwandlung von $\hat{f}_n^{KQ}$ diskutieren, die bei höheren Dimensionen d von X_i einen niedrigeren Generalisierungsfehler *und* bessere Schätzungen $\hat{\beta}^{KQ}$ liefert. Zur Motivation lernen wir zunächst eine weitere Eigenschaft von $\hat{f}_n^{KQ}$ kennen. Mit $\tilde{T}_n := (X_i)_{i=1,\ldots,n}$ gilt

$$\mathbb{E}\left[\hat{\beta}^{KQ} | \tilde{T}_n\right] = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T(\mathbb{X}\beta^* + \mathbb{E}[\mathbb{e}|\tilde{T}_n]) = \beta^*,$$

und daher mit beliebigem $x \in \mathcal{X}$:

$$\mathbb{E}\left[\hat{f}_n^{KQ}(x)|\tilde{T}_n\right] = x^T \mathbb{E}\left[\hat{\beta}^{KQ}|\tilde{T}_n\right] = x^T \beta^* = f^*(x)$$

Dies bedeutet, es gilt stets $\text{Bias}(\hat{f}_n^{KQ}(x)) = 0$. Weiter ist

$$\hat{f}_n^{KQ}(x) = x^T \hat{\beta}^{KQ} = \underbrace{\left[x^T (\mathbb{X}^T \mathbb{X})^{-1} \mathbb{X}^T\right]}_{=:a^T} \mathbb{Y},$$

d. h., $\hat{f}_n^{KQ}(x)$ ist linear in den Beobachtungen $Y_1, \ldots, Y_n$. Man kann folgenden Satz zeigen:

Satz 2.24 (Gauß-Markov-Theorem) Sei $d \leq n$ und $x \in \mathbb{R}^d$ beliebig sowie $\psi = x^T \beta$. Der Schätzer $\hat{\psi} = x^T \hat{\beta}^{KQ}$ hat folgende Eigenschaften:

(i) Es gibt $a \in \mathbb{R}^d$ mit $\hat{\psi} = a^T \mathbb{Y}$, und es gilt $\mathbb{E}[\hat{\psi}|\tilde{T}_n] = \psi$.
(ii) Für alle Schätzer der Form $\tilde{\psi} = \tilde{a}^T \mathbb{Y}$ mit $\mathbb{E}[\tilde{\psi}|\tilde{T}_n] = \psi$ gilt: $\text{Var}(\hat{\psi}|\tilde{T}_n) \leq \text{Var}(\tilde{\psi}|\tilde{T}_n)$.

Das bedeutet: Der Algorithmus $\hat{f}_n^{KQ}(x)$ ist unter allen Algorithmen ohne Bias (bedingt auf $\tilde{T}_n$) der Schätzer mit der geringsten Varianz $\text{Var}(\hat{f}_n^{KQ}(x)|\tilde{T}_n)$. Man kann also nicht hoffen, Algorithmen mit geringerem Excess Bayes Risk zu finden, die *keinen* Bias haben. Motiviert durch die Bias-Varianz-Zerlegung des Excess Bayes Risks (vgl. Lemma 1.9) untersuchen wir im Folgenden Algorithmen mit einem *von null verschiedenen* Bias und hoffen auf eine große Ersparnis bei der Varianz und damit auf ein geringes Excess Bayes Risk. Anschaulich bedeutet das Forcieren eines Bias, dass wir bei der Konstruktion des Algorithmus während der Schätzung von β Nebenbedingungen in Form von Bestrafungstermen vorgeben, vgl. Bemerkung 1.16.

2.3 Ridge-Schätzer

Die anschauliche Motivation für den im Folgenden eingeführten Ridge-Schätzer und alle weiteren Schätzer ist die Vorstellung, dass nur wenige Komponenten von X_1 tatsächlich einen messbaren Einfluss auf Y_1 haben. Wir wollen also einen Teilvektor von X_1 finden, der Y_1 fast genauso gut erklärt wie das gesamte X_1. In Hinsicht auf die Modellannahme bedeutet das:

Annahme: Nur wenige β_j^* sind tatsächlich signifikant von 0 verschieden. (2.9)

Sind diese Parameter identifiziert, können sie besser auf Basis der Trainingsdaten geschätzt werden. Ein Extrembeispiel für die Identifizierung der geeigneten Parameter ist das Prüfen aller Möglichkeiten:

Beispiel 2.25 (Subset selection) Für ein $\beta = (\beta_0, \beta_1, \ldots, \beta_d)$ sei $S(\beta) := \{j \in \{1, \ldots, d\} : \beta_j \neq 0\}$ die Menge der Indizes, bei welchen β von null verschieden ist. Für jedes $k = 1, \ldots, d$ suche den KQ-Schätzer

$$\hat{\beta}^{(k)} = \operatorname*{arg\,min}_{\beta \in \mathbb{R}^{d+1}, \#S(\beta)=k} \hat{R}_n(\beta), \tag{2.10}$$

der nur k der Komponenten von β nicht auf 0 setzt (außer β_0, das im Allgemeinen immer benötigt wird). Da mit immer mehr Parametern das Risiko immer stärker verringert werden kann, ist $k \mapsto \hat{R}_n(\hat{\beta}^{(k)})$ jedoch monoton fallend und eine geeignete Wahl von k und dem finalen Schätzer $\hat{\beta}^{(k)}$ nicht anhand von $\hat{R}_n(\hat{\beta}^{(k)})$ möglich. Stattdessen nutzt man beispielsweise ein AIC-Kriterium (vgl. Gl. (1.14)) der Form

$$\hat{\beta}^{subset} := \hat{\beta}^{(\hat{k})}, \qquad \hat{k} := \operatorname*{arg\,min}_{k=1,\ldots d} \left[\hat{R}_n(\hat{\beta}^{(k)}) + 2 \cdot \frac{k}{n} \cdot \sigma^2 \right],$$

welches die Anzahl der benötigten Parameter durch einen additiven Term berücksichtigt. Auch wenn dieser Algorithmus in der Theorie gute Eigenschaften hat (je nachdem wie gut die Wahl $\hat{k}$ ist), so ist die schnelle Berechenbarkeit in der Praxis nicht gewährleistet: Für jede Teilmenge $S \subset \{1, \ldots, d\}$ mit $\#S = k$ muss in Gl. (2.10) ein Schätzer berechnet werden, um $\hat{\beta}^{(k)}$ ermitteln zu können. Um alle Terme $\hat{\beta}^{(0)}, \ldots, \hat{\beta}^{(d)}$ zu erhalten, ist daher die Berechnung von insgesamt 2^d Schätzern nötig. Dies ist nur für kleine $d \leq 20$ ohne Weiteres möglich.

Bei der Entwicklung von Algorithmen für Gl. (2.9) muss daher stets auch deren Berechenbarkeit in praktischen Anwendungen im Vordergrund stehen.

Beim Ridge-Schätzer verfolgt man nicht das Ziel, die Komponenten von β direkt auf null zu setzen. Stattdessen sollen die Komponenten von β verkleinert werden, wofür ein Bestrafungsterm $J(\beta)$ für deren Größe eingeführt wird (sogenannter *shrinkage penalty*). Die Hoffnung ist, dass durch eine solche Bestrafung vor allem diejenigen β_j verkleinert werden, welche zu den am wenigsten für Y_1 relevanten Komponenten von X_1 gehören. Ausgeschlossen von dieser Bestrafung wird der *intercept* β_0, welcher eine allgemeine Eigenschaft von Y_1 charakterisiert und laut Modell keine direkte Verbindung mit X_1 besitzt. Für den Ridge-Schätzer wählt man $J^R(\beta) = \|(\beta_1, \ldots, \beta_d)^T\|_2^2$, d. h., wir bestrafen die $\|\cdot\|_2$-Norm von β ohne den *intercept* β_0 (da dieser nicht mit den Komponenten von X_1 verbunden ist).

Definition 2.26 (Ridge-Schätzer)
Sei $\lambda \geq 0$ und für $\beta = (\beta_0, \ldots, \beta_d)^T \in \mathbb{R}^{d+1}$ definiere

$$J^R(\beta) = \sum_{j=1}^{d} \beta_j^2.$$

Definiere den *Ridge-Schätzer*

$$\hat{\beta}_\lambda^R := \underset{\beta \in \mathbb{R}^{d+1}}{\arg\min} \left\{ \hat{R}_n(\beta) + \lambda \cdot J^R(\beta) \right\}$$

Der zugehörige Algorithmus ist

$$\hat{f}_{n,\lambda}^R(x) = \hat{\beta}_{\lambda,0}^R + \sum_{j=1}^{d} \hat{\beta}_{\lambda,j}^R x_j, \qquad x = (x_1, \ldots, x_d)^T \in \mathbb{R}^d. \quad \blacklozenge$$

λ ist ein *Bestrafungsparameter* und kontrolliert die Stärke der Verkleinerung der Koeffizienten: Je größer λ, desto kleiner müssen die Koeffizienten $\beta_1, \ldots, \beta_d$ einer Lösung sein. Bevorzugt werden durch diese Methode also $\hat{\beta}_\lambda^R$, bei welchen eine große Zahl von Koeffizienten nahe oder gleich null ist.

Aufgrund der einfachen Form von $J^R(\beta)$ ist durch Bestimmung der Extremstelle von $\beta \mapsto \hat{R}_n(\beta) + \lambda \cdot J(\beta)$ eine explizite Darstellung von $\hat{\beta}_\lambda^R$ möglich:

Lemma 2.27 (Explizite Darstellung des Ridge-Schätzers) Sei $\lambda > 0$. Dann gilt mit $E := \mathrm{diag}(0, 1, 1, \ldots, 1) \in \mathbb{R}^{(d+1)\times(d+1)}$:

$$\hat{\beta}_\lambda^R = (\mathbb{X}^T \mathbb{X} + \lambda n E)^{-1} \mathbb{X}^T \mathbb{Y} \tag{2.11}$$

Bemerkungen

- Im Gegensatz zu $\hat{\beta}^{KQ}$ existiert im Falle $\lambda > 0$ immer ein eindeutiger Ridge-Schätzer $\hat{\beta}_\lambda^R$. Da $\mathbb{X}^T \mathbb{X}$ stets positiv semidefinit ist, wird sie durch Addition von $\lambda n E$ invertierbar (selbst wenn $d > n$). Im Fall $\lambda = 0$ gilt $\hat{\beta}^{KQ} = \hat{\beta}_\lambda^R$.
- Im Spezialfall $\frac{1}{n}\mathbb{X}^T \mathbb{X} = I_{(d+1)\times(d+1)}$ und $d \leq n$ kann man zeigen, dass $\hat{\beta}_{\lambda,j}^R = \frac{1}{1+\lambda} \hat{\beta}_j^{KQ}$ $(j = 1, \ldots, d)$, d. h., $\hat{\beta}_\lambda^R$ entsteht durch gleichmäßige Skalierung der Einträge von $\hat{\beta}^{KQ}$ Richtung 0. Für allgemeines $\mathbb{X}$ werden die Einträge von $\hat{\beta}_\lambda^R$ jedoch *verschieden stark* in Richtung 0 skaliert.

2.3.1 Wahl des Bestrafungsparameters

Ein in der Praxis verwendbarer Algorithmus entsteht aus $\hat{f}^R_{n,\lambda}$ erst, wenn auch eine Wahl $\hat{\lambda}$ von λ basierend auf den Daten vorgegeben wird. Der finale Algorithmus ist dann $\hat{f}^R_{n,\hat{\lambda}}$. Ein Standardverfahren ist die Wahl $\hat{\lambda} = \hat{\lambda}^{cv}$ mittels Cross Validation, vgl. Definition 1.22.

Eine alternative Wahl von λ kann mittels des AIC-Kriteriums erfolgen, vgl. Definition 1.27. Wir berechnen dafür die Größe $\mathrm{df}(\lambda)$ unter der Annahme, dass $X_1, \ldots, X_n$ deterministisch sind, d. h., alle Ausdrücke werden als bedingte Kovarianzen gegeben $X_1, \ldots, X_n$ aufgefasst:

Lemma 2.28 (Effektive Anzahl der Parameter bei Ridge-Schätzung) Für $\mathrm{df}(\lambda) := \frac{1}{\sigma^2}\sum_{i=1}^n \mathrm{Cov}(Y_i, \hat{f}^R_{n,\lambda}(X_i))$ gilt:

$$\mathrm{df}(\lambda) = 1 + \sum_{j=1}^{d} \frac{s_j}{s_j + \lambda},$$

wobei $s_0 \geq s_1 \geq \ldots \geq s_d \geq 0$ die Eigenwerte von $\hat{\Sigma} := \frac{1}{n}\mathbb{X}^T\mathbb{X}$ sind.

Bemerkung: Ist $\lambda = 0$, gilt $\mathrm{df}(\hat{f}^{ridge}_{n,\lambda}) = d$. Für $\lambda \to \infty$ gilt $\mathrm{df}(\hat{f}^{ridge}_{n,\lambda}) \to 0$.

Beweis Es gilt $\sum_{i=1}^n \mathrm{Cov}(Y_i, \hat{f}^R_{n,\lambda}(X_i)) = \mathrm{Sp}(\mathrm{Cov}(\mathbb{Y}, (\hat{f}^R_{n,\lambda}(X_i))_{i=1,\ldots,n}))$, wobei $\mathrm{Cov}(Z_1, Z_2) := \mathrm{Cov}(Z_{1j}, Z_{1k}))_{j,k=1,\ldots,n}$ für n-dimensionale Zufallsvektoren Z_1, Z_2. Mit $\mathrm{Cov}(\mathbb{e}, \mathbb{e}) = \sigma^2 I_{n\times n}$ folgt

$$\begin{aligned}
\mathrm{Sp}\left(\mathrm{Cov}\left(\mathbb{Y}, \left(\hat{f}^R_{n,\lambda}(X_i)\right)_{i=1,\ldots,n}\right)\right) &= \mathrm{Sp}\left(\mathrm{Cov}(\mathbb{X}(\mathbb{X}^T\mathbb{X} + \lambda n E)^{-1}\mathbb{X}^T\mathbb{Y}, \mathbb{Y})\right) \\
&= \sigma^2 \mathrm{Sp}\left(\mathbb{X}(\mathbb{X}^T\mathbb{X} + \lambda n E)^{-1}\mathbb{X}^T\right) \\
&= \sigma^2 \mathrm{Sp}\left(\mathbb{X}^T\mathbb{X}(\mathbb{X}^T\mathbb{X} + \lambda n E)^{-1}\right).
\end{aligned}$$

Da $\hat{\Sigma}$ symmetrisch positiv semidefinit ist, gibt es eine Darstellung $\hat{\Sigma} = V D V^T$ mit orthogonaler Matrix $V \in \mathbb{R}^{(d+1)\times(d+1)}$ und Diagonalmatrix $D = \mathrm{diag}(s_0, \ldots, s_d) \in \mathbb{R}^{(d+1)\times(d+1)}$ mit $s_0 \geq s_1 \geq \ldots \geq s_d \geq 0$. Dann ist

$$\mathbb{X}^T\mathbb{X} + \lambda n E = nV(D + \lambda E)V^T$$

und weiter

$$\mathrm{Sp}\left(\mathrm{Cov}\left(\mathbb{Y}, \left(\hat{f}^R_{n,\lambda}(X_i)\right)_{i=1,\ldots,n}\right)\right) = \sigma^2 \mathrm{Sp}\left(D(D + \lambda E)^{-1}\right) = \sigma^2\left[1 + \sum_{j=1}^{d} \frac{s_j}{s_j + \lambda}\right].$$

Einsetzen in Definition 1.27 liefert folgende Definition:

Definition 2.29 (AIC-Kriterium für Ridge-Schätzer)
Seien $s_0 \geq s_1 \geq \ldots \geq s_d \geq 0$ die Eigenwerte von $\frac{1}{n}\mathbb{X}^T\mathbb{X}$. Dann ist $\hat{\lambda}^{aic} = \arg\min_{\lambda \geq 0} \mathrm{AIC}_n(\lambda)$, wobei

$$\mathrm{AIC}_n(\lambda) = \hat{R}_n(\hat{f}^R_{n,\lambda}) + \frac{2\hat{\sigma}^2}{n} \cdot \left[1 + \sum_{j=1}^{d} \frac{s_j^2}{s_j^2 + \lambda}\right]$$

und $\hat{\sigma}^2 := \hat{R}_n(\hat{f}^R_{n,\lambda_{vor}})$ mit einem $\lambda_{vor} \geq 0$. ♦

Die Wahl von $\lambda_{vor} \geq 0$ hat einen wesentlich geringeren Einfluss auf den Algorithmus $\hat{f}^R_{n,\hat{\lambda}^{aic}}$ als die direkte Wahl von λ in $\hat{f}^R_{n,\lambda}$, vgl. die Bemerkungen nach Definition 1.27.

2.3.2 Anwendung auf Beispiele

Um das Verhalten von $\hat{\beta}^R_\lambda$ für verschiedene $\lambda \geq 0$ zu zeigen, können die Profillinien des Schätzers grafisch dargestellt werden:

Definition 2.30 (Profillinien)
Die Abbildung $[0, \infty) \mapsto \mathbb{R}^{d+1}$, $\lambda \mapsto \hat{\beta}^R_\lambda$ heißt *Profillinie* des Ridge-Schätzers. ♦

Wir diskutieren die gleichen Daten wie aus Beispiel 2.22:

Beispiel 2.31 Wir betrachten jeweils $n = 100$ Trainingsdaten aus Beispiel 2.14 ($\hat{f}^{R,1}_{n,\lambda}$) und 2.15 ($\hat{f}^{R,2}_{n,\lambda}$) mit $d = 10$, $\sigma = 0{,}3$. Die Profillinien von $\hat{f}^{R,i}_{n,\lambda}$ zusammen mit den mit 5-fold Cross Validation oder AIC-Validierung ausgewählten Bestrafungsparametern λ sind in Abb. 2.4 ($i = 1$) und Abb. 2.5 ($i = 2$) dargestellt. Für $\lambda \to 0$ nähern sich die Ergebnisse wieder dem KQ-Schätzer an.

Die mit 5-fold Cross Validation ermittelten Schätzer lauten

$$\hat{\beta}^{R,1}_{\hat{\lambda}^{cv}} = (0{,}07,\ 1{,}00,\ -0{,}08,\ -1{,}05,\ -0{,}02,\ 0{,}07,\ -0{,}03,\ 0{,}08,\ -0{,}02,\ -0{,}04,\ 0{,}09)^T,$$
$$\hat{\beta}^{R,2}_{\hat{\lambda}^{cv}} = (-0{,}01,\ 1{,}04,\ -0{,}12,\ -0{,}39,\ -0{,}41,\ -0{,}32,\ -0{,}16,\ 0{,}00,\ 0{,}14,\ 0{,}25,\ 0{,}31)^T,$$

für die Testfehler erhalten wir

$$R(\hat{f}^{R,1}_{n,\hat{\lambda}^{cv}}) \approx 0{,}0963, \quad R(\hat{f}^{R,2}_{n,\hat{\lambda}^{cv}}) \approx 0{,}0909.$$

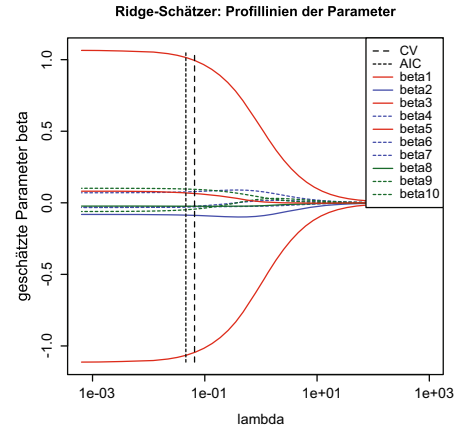


Abb. 2.4 Profillinien von $\hat{\beta}^R_\lambda$ erhalten aus $n = 100$ Trainingsdaten aus Beispiel 2.14. Als senkrechte Linien sind $\hat{\lambda}^{cv}$, $\hat{\lambda}^{aic}$ gewählt durch Cross Validation und AIC-Validierung dargestellt

Im Vergleich zu den Ergebnissen des KQ-Schätzers (vgl. Beispiel 2.22) konnten geringere Risiken erreicht werden. Bemerkenswert ist die starke Veränderung von $\hat{\beta}^{R,2}_{\hat{\lambda}^{cv}}$ im Vergleich zu $\hat{\beta}^{KQ,2}$, die nun eine Interpretation der Einflüsse der verschiedenen Komponenten von X_1 erlaubt.

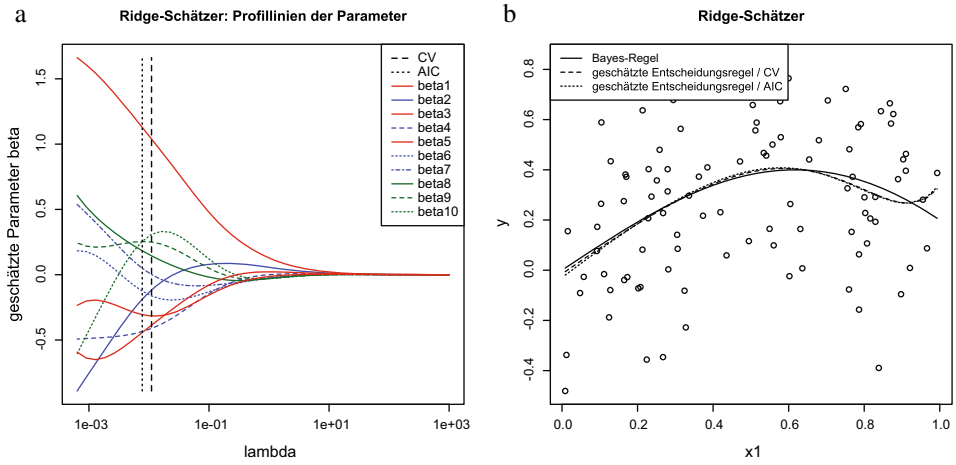


Abb. 2.5 Anwendung des Ridge-Schätzers auf $n = 100$ Trainingsdaten aus Beispiel 2.15. **a** Profillinien von $\hat{\beta}^R_\lambda$. Als senkrechte Linien sind $\hat{\lambda}^{cv}$, $\hat{\lambda}^{aic}$ gewählt durch Cross Validation und AIC-Validierung dargestellt. **b** Darstellung von $\hat{f}^R_{n,\lambda}$

2.3.3 Theoretische Resultate

Die ursprüngliche Motivation hinter Gl. (2.9) ist eine Verkleinerung des Raums $\mathcal{F}$ der möglichen Entscheidungsregeln während der Durchführung des Algorithmus. Wir werden gleich sehen, dass auch der Ridge-Algorithmus eine solche systematische Verkleinerung $\hat{\mathcal{F}}$ des Raums $\mathcal{F}$ vornimmt. Um dies formalisieren zu können, benötigen wir einige Resultate aus der Theorie der konvexen Optimierung.

Exkurs: Konvexe Optimierung

Die hier zitierten Resultate werden wir später auch in Abschn. 4.4 nutzen. Die Abkürzung NB steht hierbei stets für „Nebenbedingung".

Definition 2.32
Seien $g : \mathbb{R}^k \to \mathbb{R}$ und $G : \mathbb{R}^k \to \mathbb{R}^l$ konvexe und stetige Funktionen. Dann heißt

$$\inf_{\theta \in \mathbb{R}^k} g(\theta) \quad \text{unter NB} \quad G(\theta) \leq 0 \tag{2.12}$$

konvexes Optimierungsproblem und bedeutet:

$$\text{Suche } \hat{\theta} \in \mathbb{R}^k \text{ mit } G(\hat{\theta}) \leq 0, \text{ so dass } \inf_{\theta \in \mathbb{R}^k, G(\theta) \leq 0} g(\theta) = g(\hat{\theta}).$$

Hierbei ist $a \leq 0$ für einen Vektor $a \in \mathbb{R}^l$ komponentenweise zu verstehen, d. h. $a_1 \leq 0, \ldots, a_l \leq 0$.

Die Funktion

$$\ell : \mathbb{R}^k \times \mathbb{R}^l_{\geq 0}, \quad \ell(\theta, p) := g(\theta) + \langle p, G(\theta) \rangle$$

heißt zu Gl. (2.12) gehörige *Lagrange-Funktion,* und $p = (p_1, \ldots, p_l)^T$ heißen *Lagrange-Multiplikatoren.* Das Optimierungsproblem aus Gl. (2.12) besitzt die Darstellung

$$\inf_{\theta \in \mathbb{R}^k} \sup_{p \in \mathbb{R}^l_{\geq 0}} \ell(\theta, p), \tag{2.13}$$

d. h., $(\hat{\theta}, \hat{p}) \in \mathbb{R}^k \times \mathbb{R}^l_{\geq 0}$ erfüllt $\ell(\hat{\theta}, \hat{p}) = \inf_{\theta \in \mathbb{R}^k} \sup_{p \in \mathbb{R}^l_{\geq 0}} \ell(\theta, p)$ und heißt *primales Problem.* ♦

Der folgende Satz stellt eine Beziehung von Lösungen des obigen Optimierungsproblems zu einem Gleichungssystem her. Mathematisch folgt dies aus der Theorie der sogenannten starken Dualität (engl. *strong duality*) von konvexen Optimierungsproblemen, vgl. [10, Abschn. 5.2.3].

Satz 2.33 (Optimalitätsbedingungen) Sei ein konvexes Optimierungsproblem gegeben durch Gl. (2.12). Dann gilt:

(i) Gibt es $\hat{\theta} \in \mathbb{R}^k$, $\hat{p} \in \mathbb{R}^l_{\geq 0}$, so dass die *Optimalitätsbedingungen*

$$\hat{\theta} \in \arg\min_{\theta \in \mathbb{R}^k} \ell(\theta, \hat{p}), \qquad G(\hat{\theta}) \leq 0, \qquad \hat{p} \geq 0, \qquad \langle \hat{p}, G(\hat{\theta}) \rangle = 0, \tag{2.14}$$

gelten, so ist $\hat{\theta}$ eine Lösung von 2.12.

(ii) Ist $\hat{\theta}$ eine Lösung von 2.12 und gibt es $\theta \in \mathbb{R}^k$ mit $G(\theta) < 0$ (sog. *Slater-Bedingung*), so gibt es $\hat{p} \in \mathbb{R}^l_{\geq 0}$, so dass 2.14 gilt.

Ist $\theta \mapsto \ell(\theta, \hat{p})$ differenzierbar, so ist $\hat{\theta} \in \arg\min_{\theta \in \mathbb{R}^k} \ell(\theta, \hat{p}) \iff \partial_\theta \ell(\hat{\theta}, \hat{p}) = 0$, und die Optimalitätsbedingungen in Gl. (2.14) heißen *Karush-Kuhn-Tucker-Bedingungen.*

Gilt (i) oder (ii), so ist das Optimierungsproblem aus Gl. (2.12) äquivalent zu einem anderen Optimierungsproblem (dem sogenannten dualen Problem). Diesen Zusammenhang bezeichnet man als starke Dualität.

Satz 2.34 Sind die Voraussetzungen von (i) oder (ii) aus Satz 2.33 erfüllt, so ist $(\hat{\theta}, \hat{p})$ auch Lösung des sogenannten *dualen Problems*

$$\sup_{p \in \mathbb{R}^l_{\geq 0}} \inf_{\theta \in \mathbb{R}^k} \ell(\theta, p). \tag{2.15}$$

Anwendung auf den Ridge-Schätzer

Das ursprüngliche Optimierungsproblem für den Ridge-Schätzer $\hat{\beta}^R_\lambda$ kann wie folgt umformuliert werden:

Lemma 2.35 Für jedes $\lambda > 0$ existiert ein $\hat{t}(\lambda) > 0$ (abhängig von den Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$), so dass $\hat{\beta}^R_\lambda$ Lösung ist von

$$\inf_{\beta \in \mathbb{R}^{d+1}} \hat{R}_n(\beta) \quad unter\ NB \quad \sum_{j=1}^{d} \beta_j^2 \leq \hat{t}(\lambda). \tag{2.16}$$

Beweis Definiere $\hat{t}(\lambda) := \sum_{j=1}^d (\hat{\beta}^R_{\lambda,j})^2$. Wir wenden Satz 2.33(i) mit $G(\beta) = \sum_{j=1}^d \beta_j^2 - \hat{t}(\lambda)$ und $g(\beta) = \hat{R}_n(\beta)$ an. Hier ist $\ell(\beta, \lambda) = g(\beta) + \lambda \cdot G(\beta)$. Aufgrund der Definition von

$\hat{\beta}^R_{\lambda,j}$ gilt $\partial_\beta \ell(\hat{\beta}^R_\lambda, \lambda) = 0$. Man sieht schnell, dass $(\hat{\beta}^R_\lambda, \lambda)$ auch alle anderen Bedingungen in Gl. (2.14) erfüllt. Satz 2.33(i) liefert, dass $\hat{\beta}^R_\lambda$ Lösung von 2.16 ist.

Bemerkungen

- Die englische Bezeichnung *ridge* heißt übersetzt „Wall“ und kommt daher, dass die Bestimmung des Ridge-Schätzers gemäß Lemma 2.35 als Optimierungsproblem verstanden werden kann, bei welchem ein maximaler Wert für die $\|\cdot\|_2$-Norm von β vorgegeben wird. Ist $\mathbb{X}^T\mathbb{X}$ invertierbar und daher auch der KQ-Schätzer eindeutig bestimmt, so kann t unabhängig von λ den Wert $t_{max} = \sum_{j=1}^d \left(\hat{\beta}^{KQ}_j\right)^2$ nicht überschreiten.

Lemma 2.35 liefert folgende Interpretation für den Ridge-Schätzer: Sobald ein λ gewählt wurde und die Trainingsdaten bekannt sind, ist $\hat{t}(\lambda)$ in Gl. (2.16) festgelegt und der Algorithmus $\hat{f}^R_{n,\lambda}$ sucht nur noch innerhalb der Menge

$$\hat{\mathcal{F}}(\lambda) := \{f = [x \mapsto \beta_0 + \sum_{j=1}^d \beta_j x_j] \in \mathcal{F} : \sum_{j=1}^d \beta_j^2 \leq \hat{t}(\lambda)\}.$$

Eventuell gilt dann jedoch $f^* \in \mathcal{F} \backslash \hat{\mathcal{F}}(\lambda)$, womit ein systematischer Fehler bei der Schätzung, d. h. ein Bias, entsteht. Das Vorgehen, dass $\hat{f}^R_{n,\lambda}$ *während der Durchführung* die Menge der möglichen Entscheidungsfunktionen reduziert, bezeichnet man als *Modellwahl* (engl. *model selection*).

Konvergenzrate des Ridge-Algorithmus

Bemerkung 2.16 gilt entsprechend für diesen Abschnitt. Insbesondere ist hier $\hat{\beta}^R_\lambda = \arg\min_{\beta\in\mathbb{R}^d} \{\hat{R}_n((0, \beta^T)^T) + \lambda \cdot J^R(\beta)\} = (\mathbb{X}^T\mathbb{X} + \lambda n I_{d\times d})^{-1}\mathbb{X}^T\mathbb{Y}$ mit modifiziertem $\mathbb{X} = (X_1^T, \ldots, X_n^T)^T \in \mathbb{R}^{n\times d}$.

Wir zeigen hier ein leicht vereinfachtes Resultat aus [18, Theorem 16] über die Konvergenzrate des Excess Bayes Risks des Ridge-Schätzers. Zur Formulierung benötigen wir noch folgende Bezeichnungen: Sei $\Sigma = V D V^T$ eine Spektralzerlegung von Σ, d. h., $V \in \mathbb{R}^{d\times d}$ ist orthogonal und $D = \text{diag}(s_1, \ldots, s_d)$ ist die Diagonalmatrix der Eigenwerte von Σ. Sei $d_{k,\lambda} := \sum_{j=1}^d \left(\frac{s_j}{s_j+\lambda}\right)^k$ für $k = 1, 2$.

Satz 2.36 Es gelte Modellannahme 2.1 mit $\beta_0^* = 0$, und X_1 sei zentriert. Es gelte $\varepsilon_i \sim N(0, \sigma^2)$. Für jedes $\lambda \geq 0$ gebe es $\rho_\lambda \geq 1$ mit $\frac{\|(\Sigma+\lambda I_{d\times d})^{-1/2} X_1\|_2}{\sqrt{d_{1,\lambda}}} \leq \rho_\lambda$ f. s. Dann gilt für $d \geq \log(n)$ und alle $\lambda = \lambda_n > 0$ mit $\frac{n}{\rho_\lambda^2 d \log(d)} \to \infty$:

$$\limsup_{c\to\infty}\limsup_{n\to\infty}\mathbb{P}\left(R(\hat{f}_n^{KQ})-R(f^*)\geq c\cdot\gamma_n(\lambda)\right)=0, \tag{2.17}$$

wobei $\gamma_n(\lambda)=\left(\frac{\sigma^2}{n}d_{2,\lambda}+\lambda^2\sum_{j=1}^{d}\frac{s_j\cdot(V^T\beta^*)_j^2}{(s_j+\lambda)^2}\right)$.

Beweis Wir geben erneut nur eine Motivation. Zur Abkürzung schreibe $\hat{\beta}=\hat{\beta}_\lambda^R$, $\hat{\Sigma}:=\frac{1}{n}\mathbb{X}^T\mathbb{X}$ und $\tilde{T}_n:=(X_i)_{i=1,\ldots,n}$. Dann gilt

$$\mathbb{E}[\hat{\beta}-\beta^*|\tilde{T}_n]=-\lambda(\hat{\Sigma}+\lambda I_{d\times d})^{-1}\beta^*,$$

und somit lautet der quadrierte Bias

$$\left\|\mathbb{E}\left[\Sigma^{1/2}(\hat{\beta}-\beta^*)|\tilde{T}_n\right]\right\|_2^2=\lambda^2\left\|\Sigma^{1/2}\left(\hat{\Sigma}+\lambda I_{d\times d}\right)^{-1}\beta^*\right\|_2^2.$$

Weiter gilt $\hat{\beta}-\mathbb{E}\left[\hat{\beta}|\tilde{T}_n\right]=\frac{1}{n}(\hat{\Sigma}+\lambda I_{d\times d})^{-1}\mathbb{X}^T\mathbb{e}$, daher lautet die Varianz

$$\mathbb{E}\left[\left\|\Sigma^{1/2}\left(\hat{\beta}-\mathbb{E}[\hat{\beta}|\tilde{T}_n]\right)\right\|_2^2|\tilde{T}_n\right]=\frac{\sigma^2}{n}\mathrm{Sp}\left(\Sigma(\hat{\Sigma}+\lambda I_{d\times d})^{-1}\hat{\Sigma}\left(\hat{\Sigma}+\lambda I_{d\times d}\right)^{-1}\right).$$

Gilt $\hat{\Sigma}\approx\Sigma$ und nutzen wir die Zerlegung $\Sigma=VDV^T$, so ist

$$\begin{aligned}\left\|\mathbb{E}\left[\Sigma^{1/2}(\hat{\beta}-\beta^*)|\tilde{T}_n\right]\right\|_2^2&\approx\lambda^2\mathrm{Sp}\left((\beta^*)^TV(D+\lambda I_{d\times d})^{-1}D(D+\lambda I_{d\times d})^{-1}V^T\beta^*\right)\\&=\lambda^2\sum_{j=1}^{d}\frac{s_j\left(V^T\beta^*\right)_j^2}{\left(s_j+\lambda\right)^2}\end{aligned}$$

und mit demselben Vorgehen

$$\mathbb{E}\left[\left\|\Sigma^{1/2}(\hat{\beta}-\mathbb{E}[\hat{\beta}|\tilde{T}_n])\right\|_2^2|\tilde{T}_n\right]\approx\frac{\sigma^2}{n}\sum_{j=1}^{d}\frac{s_j^2}{(s_j+\lambda)^2}=\frac{\sigma^2}{n}d_{2,\lambda}.$$

Insgesamt folgt für das auf $\tilde{T}_n$ bedingte Excess Bayes Risk aufgrund der Bias-Varianz-Zerlegung:

$$\mathbb{E}\left[\left\|\Sigma^{1/2}(\hat{\beta}-\beta^*)\right\|_2^2|\tilde{T}_n\right]\approx\lambda^2\sum_{j=1}^{d}\frac{s_j(V^T\beta^*)_j^2}{(s_j+\lambda)^2}+\frac{\sigma^2}{n}d_{2,\lambda}=\gamma_n$$

Bemerkungen

- Die Bemerkungen zu den Bedingungen nach Satz 2.36 gelten entsprechend.
- Aus dem Beweis ist ersichtlich: Der Bias des Algorithmus wächst mit λ, die Varianz fällt mit λ. Im Spezialfall $\Sigma = I_{d\times d}$ gilt $s_1 = \ldots = s_d = 1$ und damit

$$\gamma_n(\lambda) = \frac{1}{(1+\lambda)^2}\left[\frac{\sigma^2 d}{n} + \lambda^2 \|V^T\beta^*\|_2^2\right].$$

  Das Minimum wird erreicht für $\lambda^* = \frac{\sigma^2 d}{n\|V^T\beta^*\|_2^2}$, in diesem Fall gilt

$$\gamma_n(\lambda^*) = \frac{\sigma^2 d}{n} \cdot \frac{\|V^T\beta^*\|_2^2}{\frac{\sigma^2 d}{n} + \|V^T\beta^*\|_2^2}.$$

  Falls also $\lambda \approx \lambda^*$ gilt, kann wegen $\frac{\|V^T\beta^*\|_2^2}{\frac{\sigma^2 d}{n}+\|V^T\beta^*\|_2^2} \leq 1$ eine bessere Konvergenzrate als beim KQ-Schätzer erreicht werden. Für $\Sigma \neq I_{d\times d}$ können ähnliche Resultate gezeigt werden.
- Falls die Komponenten innerhalb des Vektors X_1 stark korreliert sind, sind einige Eigenwerte s_j von Σ sehr klein. Aus dem Beweis ist ersichtlich, dass die Varianz der Schätzung von $\hat{\beta}_\lambda^R$ folgende Form besitzt:

$$\mathbb{E}\left[\|\hat{\beta} - \mathbb{E}[\hat{\beta}|\tilde{T}_n]\|_2^2 \big| \tilde{T}_n\right] \approx \frac{\sigma^2}{n}\sum_{j=1}^{d}\frac{s_j}{(s_j+\lambda)^2}$$

  Während beim KQ-Schätzer $\lambda = 0$ galt und sehr kleine s_j zu sehr hohen Varianzen von $\hat{\beta}^{KQ,2}$ geführt haben, werden diese nun durch $\lambda > 0$ neutralisiert. Dies erklärt die starke Verbesserung der Schätzung von $\hat{\beta}^{R,2}$ im Vergleich zu $\hat{\beta}^{KQ,2}$ in Beispiel 2.31.

2.4 Lasso-Schätzer

Wie beim Ridge-Schätzer ist unsere Annahme, dass tatsächlich nur wenige $\beta_j^* \neq 0$ sind. Durch eine Änderung des Bestrafungsterms J^R werden wir einen neuen Schätzer definieren, der Lösungen liefert, die dieser Annahme stärker gerecht werden.

Aus theoretischer Sicht wäre es am besten, direkt die Anzahl der Nicht-Null-Parameter zu bestrafen, d. h. $J^0(\beta) = \|\beta\|_0 = \#\{j = 1, \ldots, d : \beta_j \neq 0\}$ zu wählen. Für dieses J sind jedoch keine effizienten Berechnungsmethoden zur Bestimmung des Minimierers bekannt, so dass die praktische Umsetzbarkeit nicht gegeben ist.

Wir werden im Folgenden sehen, dass die Wahl $J^L(\beta) = \sum_{j=1}^d |\beta_j|$ (welche der $\|\cdot\|_1$-Norm von $(\beta_1, \ldots, \beta_d)$ entspricht) jedoch ähnliche Eigenschaften wie J^0 aufweist.

Definition 2.37 (Lasso-Schätzer)
Sei $\lambda \geq 0$ und für $\beta = (\beta_0, \beta_1, \ldots, \beta_d)^T \in \mathbb{R}^{d+1}$ definiere

$$J^L(\beta) = \sum_{j=1}^{d} |\beta_j|.$$

Der *Lasso-Schätzer* ist

$$\hat{\beta}_\lambda^L :\in \arg\min_{\beta \in \mathbb{R}^{d+1}} \left\{ \hat{R}_n(\beta) + \lambda \cdot J^L(\beta) \right\}$$

Der zugehörige Algorithmus ist

$$\hat{f}_{n,\lambda}^L(x) = \hat{\beta}_{\lambda,0}^L + \sum_{j=1}^{d} \hat{\beta}_{\lambda,j}^L x_j. \qquad \blacklozenge$$

Es gibt keine explizite Formel für $\hat{\beta}_\lambda^L$ in den Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$. Jedoch gibt es stets ein globales Minimum der Zielfunktion $\beta \mapsto S(\beta) := \hat{R}_n(\beta) + \lambda \cdot J^L(\beta)$, da S konvex ist und $\lim_{\|\beta\|_2 \to \infty} S(\beta) = \infty$ gilt.

Der Name „Lasso“ ist eine Abkürzung für *Least absolute shrinkage and selection operator* und weist darauf hin, dass $\hat{\beta}_\lambda^L$ tatsächlich eine *Auswahl* aus allen möglichen Komponenten von β trifft. Wir zeigen dies kurz an einem sehr einfachen Spezialfall:

Beispiel 2.38 (Verhalten des Lasso-Schätzers) Es sei $d = n$ und $X_{ij} = \mathbb{1}_{\{i=j\}}$ $(i, j = 1, \ldots, n)$. Zur Vereinfachung reduzieren wir das Modell auf eines ohne *intercept,* d. h., wir nehmen $\beta_0^* = 0$ an. Dann ist

$$Y_i = \beta_i^* + \varepsilon_i, \quad i = 1, \ldots, n.$$

Wegen des bekannten $\beta_0^* = 0$ betrachten wir nur die Minimierung über $\beta = (\beta_1, \ldots, \beta_d) \in \mathbb{R}^d$ mit

$$\hat{\beta}_\lambda^L :\in \arg\min_{\beta \in \mathbb{R}^d} \left\{ \hat{R}_n((0, \beta^T)^T) + \lambda \cdot J^L(\beta) \right\}. \tag{2.18}$$

Dann kann $\hat{\beta}_\lambda^L$ durch eine komponentenweise Minimierung erhalten werden:

$$\hat{\beta}_\lambda^L = \arg\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{n} \{ \frac{1}{n} (Y_i - \beta_i)^2 + \lambda |\beta_i| \} = \left(\arg\min_{\beta_j \in \mathbb{R}} \{ \frac{1}{n} (Y_j - \beta_j)^2 + \lambda |\beta_j| \} \right)_{j=1,\ldots,d},$$

Das Minimum der Funktion $a \mapsto \frac{1}{n}(Y_j - a)^2 + \lambda|a|$ ist leicht explizit zu ermitteln (vgl. Lemma 2.43 unten):

$$\hat{\beta}_{\lambda,j}^{lasso} = \begin{cases} Y_j - \frac{\lambda}{2}n, & Y_j > \frac{\lambda}{2}n, \\ Y_j + \frac{\lambda}{2}n, & Y_j < -\frac{\lambda}{2}n, \\ 0, & |Y_j| \leq \frac{\lambda}{2}n \end{cases}$$

Das bedeutet, falls $|Y_j| \leq \frac{\lambda}{2}n$ gilt, wird $\hat{\beta}_{\lambda,j}^{lasso}$ *exakt* null. Beim Ridge-Schätzer und KQ-Schätzer würde man dagegen erhalten:

$$\hat{\beta}_{\lambda,j}^{R} = \frac{Y_j}{1 + n\lambda}, \qquad \hat{\beta}_j^{KQ} = Y_j,$$

d. h., dort entstehen die Schätzer durch Skalisierung oder direkte Übernahme der Beobachtungen Y_j und werden nicht auf null gesetzt.

Für allgemeine Modelle lässt sich ein ähnliches Verhalten von $\hat{\beta}_\lambda^L$ nicht unmittelbar herleiten, wir werden aber im Folgenden eine grafische Motivation bereitstellen. Wie beim Ridge-Schätzer lässt sich das Optimierungsproblem umformulieren:

Lemma 2.39 Für jedes $\lambda > 0$ existiert ein $\hat{t}(\lambda) > 0$, so dass $\hat{\beta}_\lambda^L$ Lösung ist von

$$\inf_{\beta \in \mathbb{R}^d} \hat{R}_n(\beta) \quad unter\ NB \quad \sum_{j=1}^{d} |\beta_j| \leq \hat{t}(\lambda).$$

Beweis Wähle $\hat{t}(\lambda) := \sum_{j=1}^{d} |\hat{\beta}_{\lambda,j}^L|$ und gehe weiter vor wie im Beweis von Satz 2.35.

Unter Nutzung des entsprechenden Resultats für den Ridge-Schätzer aus Lemma 2.35 erhalten wir für $d = 2$ folgende grafische Interpretationen:

Bemerkung 2.40 (Vergleich Lasso- und Ridge-Schätzer) Sei $d = 2$, und wie in Beispiel 2.38 sei $\beta_0^* = 0$ und $\beta = (\beta_1, \beta_2)^T \in \mathbb{R}^2$; der Schätzer $\hat{\beta}_\lambda^L$ werde gemäß Gl. (2.18) ermittelt.

Sind Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$ gegeben und ein $\lambda > 0$ ausgewählt, so erhält man durch Lemma 2.35 und 2.39 Werte $\hat{t}^R, \hat{t}^L > 0$, so dass $\hat{\beta}_\lambda^R, \hat{\beta}_\lambda^L$ als Minimierer von

$$g(\beta) := \hat{R}_n((0, \beta^T)^T) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \beta_1 X_{i1} - \beta_2 X_{i2})^2$$

unter den Nebenbedingungen $\sum_{j=1}^{d} \beta_j^2 \leq \hat{t}^R$ bzw. $\sum_{j=1}^{d} |\beta_j| \leq \hat{t}^L$ interpretiert werden können.

Die Abbildung $\beta \mapsto g(\beta)$ ist ein Paraboloid, die Höhenlinien $H_c = \{\beta \in \mathbb{R}^2 : \hat{R}_n(\beta) = c\}$ für festes $c \geq 0$ entsprechen Ellipsen. Je größer c, desto größer werden die Ellipsen.

Die Minimierung von $\beta \mapsto g(\beta)$ über $\sum_{j=1}^{d} \beta_j^2 \leq \hat{t}^R$ erfolgt anschaulich, indem man ausgehend vom Zentrum der Ellipse (dem KQ-Schätzer $\hat{\beta}^{KQ} = \arg\min_{\beta \in \mathbb{R}^2} g(\beta)$) die Ellipsen schrittweise so weit vergrößert (d. h. den Wert von $g(\beta)$ erhöht), bis das erste Mal $\sum_{j=1}^{d} \beta_j^2 \leq \hat{t}^R$ erfüllt ist. Der Schnittpunkt der Ellipse mit der Menge $\{\beta \in \mathbb{R}^d : \sum_{j=1}^{d} \beta_j^2 \leq \hat{t}^R\}$ entspricht $\hat{\beta}_\lambda^R$. Dieses Vorgehen ist in Abb. 2.6 illustriert. Beim Ridge-Schätzer entspricht die Nebenbedingung $\sum_{j=1}^{d} \beta_j^2 \leq \hat{t}^R$ einem Kreis; im Allgemeinen liegt $\hat{\beta}_\lambda^R$ mitten im Koordinatensystem, und keine Komponente ist null. Beim Lasso-Schätzer hingegen beschreibt die Nebenbedingung $\sum_{j=1}^{d} |\beta_j| \leq \hat{t}^L$ ein gedrehtes Quadrat; falls

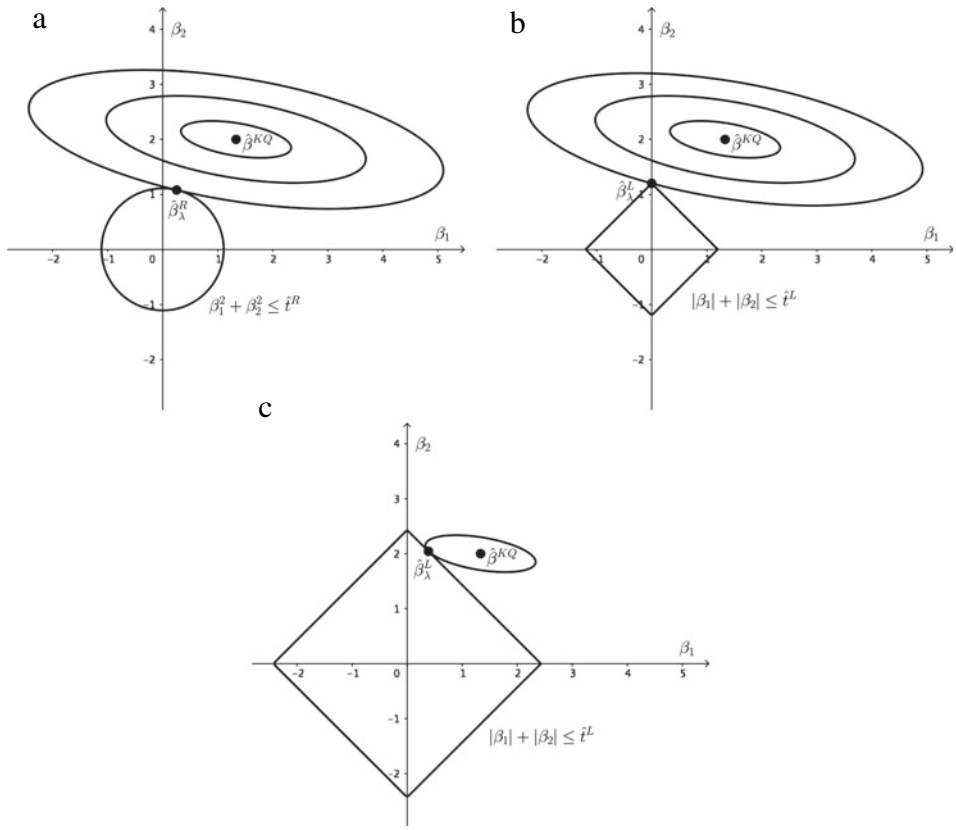


Abb. 2.6 Darstellung der Höhenlinien von $g(\beta)$ aus Beispiel 2.40 zur Veranschaulichung des Minimierungsvorgangs bei Ridge- und Lasso-Schätzer. **a** Verhalten des Ridge-Schätzers, und **b** Verhalten Lasso-Schätzers bei gleichem λ. **c** Verhalten des Lasso-Schätzers bei zu kleinem λ. Dann sind beide Komponenten von $\hat{\beta}_\lambda^L$ nicht null

$\hat{t}^L$ groß genug ist (d. h. λ klein genug), liegt $\hat{\beta}^L_\lambda$ automatisch auf einer Ecke der Menge $\{\beta \in \mathbb{R}^d : \sum_{j=1}^d |\beta_j| \leq \hat{t}^L\}$, und somit ist mindestens eine Komponente von $\hat{\beta}^L_\lambda$ exakt null.

Wir zeigen nun das Verhalten des Lasso-Algorithmus anhand der Daten aus Beispiel 2.22:

Beispiel 2.41 Wir betrachten jeweils $n = 100$ Trainingsdaten aus Beispiel 2.14 ($\hat{f}^{L,1}_{n,\lambda}$) und 2.15 ($\hat{f}^{L,2}_{n,\lambda}$) mit $d = 10$, $\sigma = 0{,}3$. Die Profillinien von $\hat{f}^{L,i}_{n,\lambda}$ zusammen mit den mit 5-fold Cross Validation ausgewählten Bestrafungsparametern λ sind in Abb. 2.7a ($i = 1$) und Abb. 2.8 ($i = 2$) dargestellt. Für $\lambda \to 0$ nähern sich die Ergebnisse wieder dem KQ-Schätzer an.

Die mit 5-fold Cross Validation ermittelten Schätzer lauten

$$\hat{\beta}^{L,1}_{\hat{\lambda}^{cv}} = (0{,}11, 0{,}94, 0, -1{,}04, 0, 0, 0, 0, 0, 0)^T,$$
$$\hat{\beta}^{L,2}_{\hat{\lambda}^{cv}} = (-0{,}01, 1{,}04, 0, 1{,}00, 0, 0, 0, 0, 0, 0{,}24)^T.$$

Man sieht, dass die beiden wichtigsten Nicht-Null-Komponenten $\beta^*_1 = 1$, $\beta^*_3 = -1$ richtig erkannt und geschätzt werden, aber die Komponente $\beta^*_5 = 0{,}2$ aufgrund des Rauschens in den Trainingsdaten nicht erkannt wird. Liegen mehr Trainingsdaten vor, so werden alle Nicht-Null-Komponenten erkannt (für das Beispiel 2.14 ist dies in Abb. 2.7b) mit $n = 500$ Trainingsdaten illustriert). Für die Testfehler erhalten wir

$$R(\hat{f}^{L,1}_{n,\hat{\lambda}^{cv}}) \approx 0{,}0961, \quad R(\hat{f}^{L,2}_{n,\hat{\lambda}^{cv}}) \approx 0{,}0906.$$

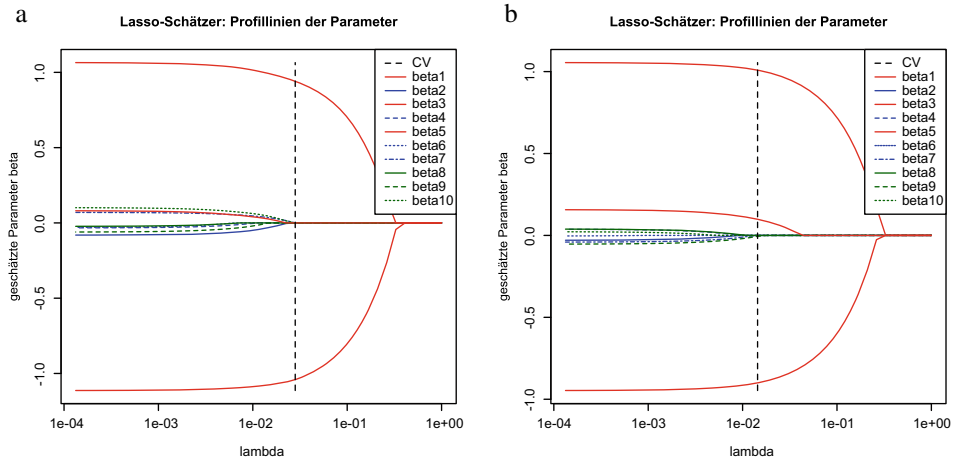


Abb. 2.7 Profillinien von $\hat{\beta}^L_\lambda$ erhalten aus **a** $n = 100$ und **b** $n = 500$ Trainingsdaten aus Beispiel 2.14. Als senkrechte Linie ist $\hat{\lambda}^{cv}$ gewählt durch Cross Validation dargestellt

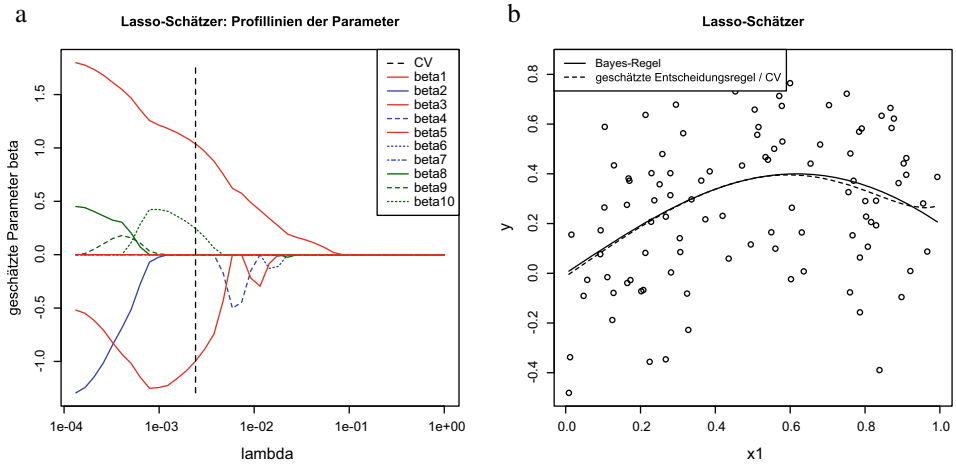


Abb. 2.8 Anwendung des Lasso-Schätzers auf $n = 100$ Trainingsdaten aus Beispiel 2.15. **a** Profillinien von $\hat{\beta}^L_\lambda$. Als senkrechte Linie ist $\hat{\lambda}^{cv}$ gewählt durch Cross Validation dargestellt. **b** Darstellung von $\hat{f}^L_{n,\lambda}$

Im Vergleich zu den Ergebnissen des Ridge-Schätzers (vgl. Beispiel 2.31) konnten die Risiken weiter verringert werden.

2.4.1 Berechnung von Lasso-Schätzern

In diesem Abschnitt geben wir ein Verfahren an, mit welchem der Lasso-Schätzer $\hat{\beta}^L_\lambda$, d. h. ein Minimierer der Zielfunktion

$$\beta \mapsto \hat{R}_n(\beta) + \lambda \cdot J^L(\beta), \tag{2.19}$$

in der Praxis bestimmt werden kann.

Eine in der Praxis häufig genutzte Methode ist der genannte LARS-Algorithmus (Abkürzung für *Least Angle Regression*), der geometrische Eigenschaften der Lösung ausnutzt. Da er sehr stark auf Gl. (2.19) zugeschnitten ist und nur wenige Verallgemeinerungen erlaubt, werden wir diesen hier nicht vorstellen, sondern leiten ein Lösungsverfahren aus einer allgemeineren Technik ab. Diese nutzt die Konvexität des Summanden $\hat{R}_n(\beta)$ sowie die einfache, additive Struktur von $J^L(\beta) = \sum_{j=1}^d |\beta_j|$ aus.

Koordinatenweiser Abstieg

Zur Formulierung nutzen wir folgende allgemeine Methode (vgl. [32]) des koordinatenweisen Abstiegs (engl. *coordinate descent*). Die Idee hinter dem Verfahren ist, dass die unten

definierte Funktion q global eine sehr stark einem Paraboloid ähnelnde Struktur hat. Man kann sich daher dem globalen Minimum im „Zickzackkurs“ iterativ nähern, indem man jeweils nur bzgl. einer Koordinate minimiert.

Satz 2.42 Sei

$$q : \mathbb{R}^k \to \mathbb{R}, \quad q(z) := \phi(z) + \sum_{j=1}^{k} \phi_j(z_j),$$

wobei $\phi : \mathbb{R}^k \to \mathbb{R}$ eine konvexe, differenzierbare Funktion und $\phi_j : \mathbb{R} \to \mathbb{R}$ konvexe Funktionen sind. q habe beschränkte Niveaumengen, d. h., für jedes $c > 0$ sei $\{z \in \mathbb{R}^k : q(z) = c\}$ beschränkt. Für jedes $j \in \{1, \ldots, k\}$ und fest gewählte $a_i \in \mathbb{R}$, $i \neq j$ sei $z_j \mapsto q(a_1, \ldots, a_{j-1}, z_j, a_{j+1}, \ldots, a_k)$ strikt konvex.

Sei $z^{(0)} \in \mathbb{R}^k$ beliebig und setze $m = 0$. Wiederhole für $j = 1, \ldots, k, 1, \ldots, k, \ldots$:

(i) Optimiere nur bzgl. z_j, d. h., setze

$$z_j^{(m+1)} := \arg\min_{z_j \in \mathbb{R}} q(z_1^{(m)}, \ldots, z_{j-1}^{(m)}, z_j, z_{j+1}^{(m)}, \ldots, z_k^{(m)})$$

und $z^{(m+1)} := (z_1^{(m)}, \ldots, z_{j-1}^{(m)}, z_j^{(m+1)}, z_{j+1}^{(m)}, \ldots, z_d^{(m)})^T$. Erhöhe m um 1.

Dann gilt: $z^{(m)} \to z^{(\infty)}$ $(m \to \infty)$ und $z^{(\infty)}$ ist ein globaler Minimierer von q.

Zur übersichtlicheren Behandlung von β_0 sei $(\tilde{X}_{i0}, \ldots, \tilde{X}_{id})^T = \tilde{X}_i := (1, X_i^T)^T$, $i = 1, \ldots, n$. Für das Lasso-Optimierungsproblem wählen wir dann mit $\beta = (\beta_0, \ldots, \beta_d)^T \in \mathbb{R}^{d+1}$:

$$\phi(\beta) := \hat{R}_n(\beta) = \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - \sum_{j=0}^{d} \beta_j \tilde{X}_{ij} \right)^2, \qquad \phi_j(\beta_j) = \begin{cases} |\beta_j|, & j = 1, \ldots, d \\ 0, & j = 0 \end{cases}$$

Damit ergibt sich für $j = 1, \ldots, d$:

$$\begin{aligned} q(\beta) &= \frac{1}{n} \sum_{i=1}^{n} (Y_i - \sum_{l=0, l \neq j}^{d} \tilde{X}_{il}\beta_l - \tilde{X}_{ij}\beta_j)^2 + \lambda|\beta_j| + \sum_{l=0, l \neq j}^{d} |\beta_l| \\ &= -2 \underbrace{\left[\frac{1}{n} \sum_{i=1}^{n} (Y_i - \sum_{l=0, l \neq j}^{d} \tilde{X}_{il}\beta_l) \tilde{X}_{ij} \right]}_{=:a} \beta_j + \underbrace{\left[\frac{1}{n} \sum_{i=1}^{n} \tilde{X}_{ij}^2 \right]}_{=:b} \beta_j^2 + \lambda|\beta_j| + \text{const.} \end{aligned} \tag{2.20}$$

Die Funktion $\beta_j \mapsto q(\beta)$ besitzt ein eindeutiges Minimum gemäß folgendem Lemma:

Lemma 2.43 Sei $a \in \mathbb{R}$ und $\lambda, b > 0$. Die Funktion $g : \mathbb{R} \to \mathbb{R}$, $g(z) = -2az + bz^2 + \lambda|z|$ ist streng konvex und besitzt ein eindeutiges Minimum bei

$$z_{min} = \begin{cases} \frac{a-\frac{\lambda}{2}}{b}, & a > \frac{\lambda}{2} \\ \frac{a+\frac{\lambda}{2}}{b}, & a < -\frac{\lambda}{2} \\ 0, & |a| \leq \frac{\lambda}{2} \end{cases} = \operatorname{sign}(a)\frac{1}{b}(|a| - \frac{\lambda}{2})^+,$$

wobei $w^+ := \max\{0, w\}$ den Positivteil einer Zahl $w \in \mathbb{R}$ bezeichnet.

Im Falle $j = 0$ erhält man das gleiche Ergebnis wie in Gl. (2.20) ohne den Term $\lambda|\beta_j|$ und damit den Minimierer $\frac{a}{b}$. Anwendung von Satz 2.42 auf das Lasso-Problem liefert insgesamt folgendes Berechnungsverfahren:

Bemerkung 2.44 (Berechnung des Lasso-Schätzers) Sei $\gamma > 0$. Setze $\beta^{(0)} := (0, \ldots, 0)^T \in \mathbb{R}^{d+1}$ und $m = 0$.

Wiederhole über $j = 0, \ldots, d, 0, \ldots, d, \ldots$ bis zur Konvergenz (d. h. bis $\max_{j=0,\ldots,d} |\beta_j^{(m)} - \beta_j^{(m-1)}| \leq \gamma$)

1. $r_{ij} := Y_i - \sum_{l=0, l\neq j}^{d} \tilde{X}_{il}\beta_l^{(m)}$, $i = 1, \ldots, n$,
2. $\tilde{\beta}_j := \frac{1}{n}\sum_{i=1}^{n} \tilde{X}_{ij} r_{ij}$ ($= a$ in obiger Notation),
3. aktualisiere $\beta_j^{(m)}$:

$$\beta_j^{(m+1)} := \begin{cases} \frac{1}{\frac{1}{n}\sum_{i=1}^{n} \tilde{X}_{ij}^2} \cdot \operatorname{sign}(\tilde{\beta}_j)(|\tilde{\beta}_j| - \frac{\lambda}{2})^+, & j = 1, \ldots, d \\ \tilde{\beta}_j, & j = 0 \end{cases}$$

   und $\beta^{(m+1)} := (\beta_1^{(m)}, \ldots, \beta_{j-1}^{(m)}, \beta_j^{(m+1)}, \beta_{j+1}^{(m)}, \ldots, \beta_d^{(m)})^T$,
4. erhöhe m um 1.

Dann ist $\hat{\beta}_\lambda^L \approx \beta^{(m)}$.

$\gamma > 0$ wird als Toleranzparameter bezeichnet und steuert, wann die Iteration abgebrochen wird. γ sollte relativ klein gewählt werden, z. B. $\gamma = 10^{-5}$, damit $\beta^{(m)}$ sicher nahe am globalen Minimum $\hat{\beta}_\lambda^L$ liegt. Je kleiner λ, desto größer sind im Allgemeinen die Werte von $\hat{\beta}_\lambda^L$ und umso mehr Iterationen sind bis zum Abbruch notwendig.

2.4.2 Theoretische Resultate

Bemerkung 2.16 gilt entsprechend für diesen Abschnitt. Insbesondere ist hier $\hat{\beta}_\lambda^L = \arg\min_{\beta\in\mathbb{R}^d}\{\hat{R}_n((0,\beta^T)^T) + \lambda \cdot J^L(\beta)\}$ und $\mathbb{X} = (X_1^T, \ldots, X_n^T)^T \in \mathbb{R}^{n\times d}$.

Die Qualität des Lasso-Schätzers ist vor allem in den Fällen besser als die des Ridge- oder KQ-Schätzers, wenn tatsächlich nur wenige $\beta_j^* \neq 0$ sind, d. h. Gl. (2.9) erfüllt ist. Dies kann mittels der Menge

$$S^* := S(\beta^*) := \{j \in \{1,\ldots,d\} : \beta_j^* \neq 0\}$$

der Nicht-Null-Komponenten von β^* formalisiert werden. Für das nachstehende theoretische Resultat gehen wir davon aus, dass $\#S^* \ll n$, aber $d \gg n$ gilt. Vor der Darstellung des theoretischen Resultats motivieren wir die dafür benötigten Bedingungen.

Wie wir in Lemma 2.19 gesehen haben, folgen die Trainingsdaten der Gleichung $\mathbb{Y} = \mathbb{X}\beta^* + \mathbb{e}$. Beim KQ-Schätzer erhält man einen Schätzer für β^* durch Multiplikation von $\frac{1}{n}\mathbb{X}^T$ mit beiden Seiten, denn dann gilt $\frac{1}{n}\mathbb{X}^T\mathbb{Y} = \frac{1}{n}\mathbb{X}^T\mathbb{X}\beta^* + \frac{1}{n}\mathbb{X}^T\mathbb{e}$ und daher $\beta^* \approx (\frac{1}{n}\mathbb{X}^T\mathbb{X})^{-1}\frac{1}{n}\mathbb{X}^T\mathbb{Y}$.

Damit dieses Argument funktioniert, muss $\hat{\Sigma} := \frac{1}{n}\mathbb{X}^T\mathbb{X} \in \mathbb{R}^{d\times d}$ invertierbar und damit der minimale Eigenwert, ausgedrückt durch den Rayleigh-Quotienten

$$\lambda_{min}(\hat{\Sigma}) := \min_{v\in\mathbb{R}^d} \frac{v^T\hat{\Sigma}v}{\|v\|_2^2},$$

positiv sein. Es ist aber $\mathrm{Rang}(\hat{\Sigma}) \leq n$ und wegen $d \gg n$ sicher $\lambda_{min}(\hat{\Sigma}) = 0$. Der Ausweg aus diesem Dilemma ist die Information $\#S^* \ll n$. Wir sehen gleich, dass auch eine schwächere Annahme an $\hat{\Sigma}$ ausreicht: Ist nämlich y und ein „Gleichungssystem“

$$S(\beta) = S^*, \quad y = \hat{\Sigma}\beta$$

für $\beta \in \mathbb{R}^d$ gegeben, so brauchen wir nicht die Invertierbarkeit von $\hat{\Sigma}$, sondern $\hat{\Sigma}$ muss (als lineare Abbildung aufgefasst) nur auf der Menge

$$\tilde{C}^* := \{\beta \in \mathbb{R}^d : S(\beta) = S^*\}$$

invertierbar sein. Nutzen wir für einen Vektor $v \in \mathbb{R}^d$ und eine Indexmenge $S \subset \{1,\ldots,d\}$ die Notation $v_S = (v_j \mathbb{1}_{\{j\in S\}})_{j=1,\ldots,d}$, so können wir auch schreiben: $\tilde{C}^* = \{\beta \in \mathbb{R}^d : \|\beta_{(S^*)^c}\|_1 = 0\}$. Mathematisch kann Invertierbarkeit einer linearen Abbildung auf einer Teilmenge mittels

$$\min_{v\in\tilde{C}^*} \frac{v^T\hat{\Sigma}v}{\|v\|_2^2} = \min_{v\in\tilde{C}^*} \frac{v^T\hat{\Sigma}v}{\|v_{S^*}\|_2^2} > 0 \tag{2.21}$$

gefordert werden, dies entspricht anschaulich dem kleinsten „Eigenwert“ der auf $\tilde{C}^*$ eingeschränkten linearen Abbildung $v \mapsto \hat{\Sigma}v$. In einem theoretischen Resultat über das Excess

Bayes Risk dürfen wir jedoch keine Bedingungen an die konkrete Form der erhaltenen Trainingsdaten stellen.

Stattdessen stellen wir Forderungen an die durch $\hat{\Sigma}$ geschätzte Größe Σ. Es genügt jedoch nicht, die Forderung in Gl. (2.21) stattdessen einfach an Σ zu stellen, da die Eigenschaft in einem Beweis wegen $\Sigma \neq \hat{\Sigma}$ nicht ohne Weiteres auf $\hat{\Sigma}$ übertragen werden kann. Wir verallgemeinern daher C^* so, dass nicht exakt $\|\beta_{(S^*)^c}\|_1 = 0$ gelten muss, sondern nur $\|\beta_{(S^*)^c}\|_1 \leq 3\|\beta_{S^*}\|_1$. Wir definieren:

$$C^* := \{\beta \in \mathbb{R}^d : \|\beta_{(S^*)^c}\|_1 \leq 3\|\beta_{S^*}\|_1\}$$

und fordern die sogenannte *Restricted Eigenvalue Property (REP):*

$$\Lambda^*_{min}(\Sigma) := \min_{v \in C^*} \frac{\left\|\Sigma^{1/2} v\right\|_2^2}{\|v_{S^*}\|_2^2} = \min_{v \in C^*} \frac{v^T \Sigma v}{\|v_{S^*}\|_2^2} > 0. \tag{2.22}$$

Wir rekapitulieren nun ein Standardresultat aus [12, Kap. 6] für die Eigenschaften von $\hat{\beta}^L_\lambda$, erweitern es aber auf ein Resultat für das Excess Bayes Risk. Für diese Erweiterung ist noch eine schwächere Annahme im Stile von Gl. (2.22) über die maximalen „Eigenwerte“ von Σ auf den Vektoren mit höchstens $\#S^*$ von null verschiedenen Komponenten nötig, die wir mittels

$$\Lambda^*_{max}(\Sigma) := \max_{\|v\|_2 = 1, \#\{j \in \{1,\ldots,d\} : v_j \neq 0\} \leq \#S^*} \|\Sigma^{1/2} v\|_2^2$$

formulieren.

Satz 2.45 Es gelte Modellannahme 2.1 mit $\beta^*_0 = 0$, und X_1 sei zentriert. Es seien $\varepsilon_i \sim N(0, \sigma^2)$, $X_i \sim N(0, \Sigma)$ i. i. d. mit $\Sigma_{jj} = 1$ für alle $j = 1, \ldots, d$. Die Größen $\Lambda^*_{min}(\Sigma) > 0$, $\Lambda^*_{max}(\Sigma) < \infty$ seien unabhängig von d. Gilt $d = d_n \to \infty$, $\frac{\log(d)\#S^*}{n} \to 0$ und ist $\lambda = \lambda_n \geq 12\sigma\sqrt{\frac{\log(d)}{n}}$ beliebig, so folgt

$$\limsup_{n \to \infty} \mathbb{P}\left(R(\hat{f}^L_{n,\lambda}) - R(f^*) \geq 8\lambda^2 \frac{\#S^*}{\Lambda^*_{min}(\Sigma)}\right) = 0.$$

Beweis Zur Abkürzung schreiben wir $\hat{\beta} = \hat{\beta}^L_\lambda$, $S = S^*$, $s := \#S^*$, $\Lambda^* = \Lambda^*_{min}(\Sigma)$ und $\tilde{T}_n := (X_i)_{i=1,\ldots,n}$. Wir zeigen zunächst eine Aussage unter den Annahmen, dass ein Ereignis aus dem Schnitt der beiden Mengen $A := \{\frac{2}{n} \max_{j=1,\ldots,d} |\mathbb{e}^T \mathbb{X}_{\cdot j}| \leq \frac{\lambda}{2}\}$ (wobei $\mathbb{X}_{\cdot j} := (X_{1j}, \ldots, X_{nj})^T$) und $B := \{\Lambda^*(\hat{\Sigma}, 3) \geq \frac{\Lambda^*}{2}\}$ eingetreten ist. Dann gelten nacheinander die folgenden Aussagen:

1. Es gilt:
$$\frac{1}{n}\left\|\mathbb{X}(\hat{\beta}-\beta^*)\right\|_2^2+\lambda\|\hat{\beta}\|_1 \le \frac{2}{n}\mathbb{e}^T\mathbb{X}(\hat{\beta}-\beta^*)+\lambda\|\beta^*\|_1$$
Beweis: $\hat{\beta}$ ist Minimierer von $\beta \mapsto \hat{R}_n((0,\beta^T)^T)+J^L(\beta)=\frac{1}{n}\|\mathbb{Y}-\mathbb{X}\beta\|_2^2+\lambda J^L(\beta)$, daher gilt
$$\frac{1}{n}\left\|\underbrace{\mathbb{Y}-\mathbb{X}\hat{\beta}}_{=:a}\right\|_2^2+\lambda\|\hat{\beta}\|_1 \le \frac{1}{n}\left\|\underbrace{\mathbb{Y}-\mathbb{X}\beta^*}_{=\mathbb{e}}\right\|_2^2+\lambda\|\beta^*\|_1.$$
Mit $\|a\|_2^2-\|\mathbb{e}\|_2^2=\|a-\mathbb{e}\|_2^2+2\mathbb{e}^T(a-\mathbb{e})$ folgt die Behauptung.
2. Es gilt $\frac{2}{n}|\mathbb{e}^T\mathbb{X}(\hat{\beta}-\beta^*)| \le \frac{2}{n}\max_{j=1,\dots,d}|\mathbb{e}^T\mathbb{X}_{\cdot j}|\cdot\|\hat{\beta}-\beta^*\|_1$.
3. Es gilt (auf A): $\frac{2}{n}\|\mathbb{X}(\hat{\beta}-\beta^*)\|_2^2 \le \lambda\{3\|\hat{\beta}_S-\beta_S^*\|_1-\|\hat{\beta}_{S^c}\|_1\}$.
Beweis: Es ist
$$\begin{aligned}\frac{2}{n}\|\mathbb{X}(\hat{\beta}-\beta^*)\|_2^2 &\overset{(1.),\,(2.),\,\text{auf } A}{\le} \lambda\left\{\underbrace{\|\hat{\beta}-\beta^*\|_1}_{=\|\hat{\beta}_S-\beta_S^*\|_1+\|\hat{\beta}_{S^c}\|_1}+2\underbrace{\|\beta^*\|_1}_{=\|\beta_S^*\|_1}-2\underbrace{\|\hat{\beta}\|_1}_{=\|\hat{\beta}_S\|_1+\|\hat{\beta}_{S^c}\|_1}\right\}\\ &= \lambda\left\{\|\hat{\beta}_S-\beta_S^*\|_1-\|\hat{\beta}_{S^c}\|_1+2\underbrace{(\|\beta_S^*\|_1-\|\hat{\beta}_S\|_1)}_{\le\|\hat{\beta}_S-\beta_S^*\|_1}\right\}\\ &\le \lambda\left\{3\|\hat{\beta}_S-\beta_S^*\|_1-\|\hat{\beta}_{S^c}\|_1\right\}.\end{aligned}$$
4. Auf A gilt: $3\|\hat{\beta}_S-\beta_S^*\|_1 \ge \|\hat{\beta}_{S^c}\|_1$, d. h. $\hat{\beta}-\beta^* \in C^*$.
Beweis: Wegen $0 \le \frac{2}{n}\|\mathbb{X}(\hat{\beta}-\beta^*)\|_2^2$ folgt dies direkt aus (3).
5. Auf $A\cap B$ gilt:
$$\left\|\hat{\beta}_S-\beta_S^*\right\|_1^2 \le s\cdot\left\|\hat{\beta}_S-\beta_S^*\right\|_2^2 \le \frac{1}{n}\left\|\mathbb{X}(\hat{\beta}-\beta^*)\right\|_2^2\frac{2s}{\Lambda^*}$$
6. Auf $A\cap B$ gilt:
$$\frac{1}{n}\left\|\mathbb{X}(\hat{\beta}-\beta^*)\right\|_2^2+\lambda\left\|\hat{\beta}-\beta^*\right\|_1 \le 8\lambda^2\frac{s}{\Lambda^*}$$
Beweis: Es gilt

$$\begin{aligned}\frac{2}{n}\|\mathbb{X}(\hat{\beta}-\beta^*)\|_2^2+\lambda \underbrace{\|\hat{\beta}-\beta^*\|_1}_{=\|\hat{\beta}_S-\beta_S^*\|_1+\|\hat{\beta}_{S^c}\|_1} &\overset{(4)}{\leq} 4\lambda\|\hat{\beta}_S-\beta_S^*\|_1\\ &\overset{(5)}{\leq} \frac{1}{\sqrt{n}}\|\mathbb{X}(\hat{\beta}-\beta^*)\|_2\cdot 4\lambda\sqrt{\frac{2s}{\Lambda^*}}\\ &\overset{4ab\leq a^2+4b^2}{\leq} \frac{1}{n}\|\mathbb{X}(\hat{\beta}-\beta^*)\|_2^2+8\lambda^2\frac{s}{\Lambda^*}.\end{aligned}$$

Abziehen von $\frac{1}{n}\|\mathbb{X}(\hat{\beta}-\beta^*)\|_2^2$ auf beiden Seiten liefert die Behauptung.

In [44, Theorem 1.6] wird gezeigt, dass es von n, d unabhängige Konstanten $c', \bar{c} > 0$ mit folgender Eigenschaft gibt: Für alle

$$n > c' \max\{\Lambda^*(\Lambda^*_{max}(\Sigma))^{1/2} s\log(5ed/s), 9\log(d)\} \quad und \quad s \leq \frac{d}{2}$$

erfüllt das Ereignis

$$C := \{\forall \delta \in C^*(3) : \frac{1}{2}\|\Sigma^{1/2}\delta\|_2 \leq \|\hat{\Sigma}^{1/2}\delta\|_2 \leq \frac{3}{2}\|\Sigma^{1/2}\delta\|_2\}$$

die Eigenschaft $\mathbb{P}(C^c) \leq \exp(-\bar{c}n)$. Offensichtlich ist $C \subset B$. Wir schließen:

7. Auf der Menge $A \cap C$ gilt wegen (4) und (6):

$$\frac{1}{2}\|\Sigma^{1/2}(\hat{\beta}-\beta^*)\|_2^2 \leq 8\lambda^2\frac{s}{\Lambda^*}$$

Es ist $\|\mathbb{X}_{\cdot j}\|_2^2 = \sum_{i=1}^n X_{ij}^2 = n\hat{\Sigma}_{jj}$. Sei $V_j := \frac{e^T\mathbb{X}_{\cdot j}}{\sqrt{n}}$, dann folgt $(V_j|\tilde{T}_n) \sim N(0, \hat{\Sigma}_{jj}\sigma^2)$. Ist Φ die Verteilungsfunktion der Standardnormalverteilung, so gilt für alle $x \in \mathbb{R} : 1-\Phi(x) \leq e^{-x^2/2}$, daher

$$\begin{aligned}\mathbb{P}(A^c|\tilde{T}_n) &\leq d\max_{j=1,\ldots,d}\mathbb{P}(|V_j| > \frac{\lambda}{4\sqrt{n}}) \leq 2d\max_{j=1,\ldots,d}\left[1-\Phi\left(\frac{\lambda\sqrt{n}}{4\hat{\Sigma}_{jj}^{1/2}\sigma}\right)\right]\\ &\leq 2d\max_{j=1,\ldots,d}\exp\left(-\frac{1}{2}\left(\frac{\lambda\sqrt{n}}{4\hat{\Sigma}_{jj}^{1/2}\sigma}\right)^2\right).\end{aligned}$$

Da $X_{ij} \sim N(0,1)$, $i = 1,\ldots,n$ unabhängig sind, ist $\sum_{i=1}^n (X_{ij}^2 - 1)$ $\chi^2(n)$-verteilt (Chi-Quadrat-Verteilung mit n Freiheitsgraden). Die Chernoff-Ungleichung liefert für deren Verteilungsfunktion $\Phi_{\chi^2(n)}$ die Eigenschaft $1-\Phi_{\chi^2(n)}(nz) \leq (ze^{1-z})^{n/2}$ für beliebiges $0 < z < 1$. Daher erfüllt die Menge $D := \{\forall j \in \{1,\ldots,d\} : \hat{\Sigma}_{jj} \leq \frac{3}{2}\Sigma_{jj} = \frac{3}{2}\}$:

$$\mathbb{P}(D^c) \le d \max_{j=1,\dots,d} \mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n}(X_{ij}^2 - 1)\right| > \frac{1}{2}\right) \le 2d\left(\frac{1}{2}e^{1/2}\right)^{n/2}$$

Insgesamt folgt mit (7):

$$\begin{aligned}
&\mathbb{P}\left(R(\hat{f}_n^L) - R(f^*) > 8\lambda^2 \frac{s}{\Lambda^*}\right) \\
&= \mathbb{E}\left[\mathbb{P}(\|\Sigma^{1/2}(\hat{\beta} - \beta^*)\|_2^2 > 8\lambda^2 \frac{s}{\Lambda^*} | \tilde{T}_n)\mathbb{1}_{C\cap D}\right] + \mathbb{P}(C^c) + \mathbb{P}(D^c) \\
&\le \mathbb{E}[\mathbb{P}(A^c|\tilde{T}_n)\mathbb{1}_D] + \mathbb{P}(C^c) + \mathbb{P}(D^c) \\
&\le 2d \exp\left(-\frac{1}{2}\left(\frac{\lambda\sqrt{n}}{6\sigma}\right)^2\right) + \exp(-\bar{c}n) + 2d(\frac{1}{2}e^{1/2})^{n/2}
\end{aligned}$$

und daher die Behauptung.

Bemerkungen

- Mit
$$\lambda = \lambda^* := 12\sigma\sqrt{\frac{\log(d)}{n}} \tag{2.23}$$
erhalten wir aus Satz 2.45, dass das Excess Bayes Risk $R(\hat{f}_{n,\lambda}^L) - R(f^*)$ mit Rate
$$\frac{\sigma^2}{\Lambda^*_{min}(\Sigma)} \cdot \frac{\log(d)\#S^*}{n} \tag{2.24}$$
gegen null konvergiert. Diese Rate kann also auch in der Praxis durch geeignete Wahl eines Schätzers $\hat{\lambda}$ erreicht werden. Wie durch den Beweis ersichtlich ist (vgl. Schritt (7)), erhält man für $\|\hat{\beta}_{\lambda^*}^L - \beta^*\|_1$ eine ähnliche Rate $\frac{\sigma}{\Lambda^*_{min}(\Sigma)} \cdot \sqrt{\frac{\log(d)}{n}}\#S^*$.
- Gl. (2.24) liefert, dass die Rate im Wesentlichen nur von der Anzahl der „effektiv vorhandenen" Parameter $\#S^*$ (d. h. die Anzahl der Nicht-Null-Komponenten von β^*) beeinflusst wird, auch wenn das Modell mit d Parametern formuliert wird. Man muss nur mit einem Faktor $\log(d)$ dafür bezahlen, die Indizes j der Nicht-Null-Komponenten von β^* nicht zu kennen. Ist $\#S^*$ sehr klein und d sehr groß, so kann $\hat{f}_{n,\lambda^*}^L$ im Vergleich zum Ridge-Schätzer ein wesentlich niedrigeres Risiko liefern.
- Wie beim Ridge-Schätzer wird die Rate von der Varianz des Rauschens σ^2 beeinflusst. Aufgrund der oben verwendeten Beweisführung, bei welcher das Excess Bayes Risk aus dem Verhalten von $\hat{\beta}_\lambda^L$ hergeleitet wurde, gibt es im Gegensatz zum Ridge-Schätzer außerdem einen Faktor $\frac{1}{\Lambda^*_{min}(\Sigma)}$, durch den die Abhängigkeitsstruktur der Komponenten X_1 sehr stark in die Rate in Gl. (2.24) eingeht. Unter leicht abgewandelten Voraussetzungen kann man zeigen, dass der Faktor durch einen nicht so stark von der Struktur Σ abhängigen Term ersetzt werden kann (vgl. [43]), wobei aber auch die Rate zu $\sqrt{\frac{\log(d)}{n}}$ anstelle von $\frac{\log(d)}{n}$ verschlechtert wird.

Während beim Excess Bayes Risk von $\hat{f}^L_{n,\lambda}$ unter geeigneten Annahmen also auf den Faktor $\frac{1}{\Lambda^*_{min}(\Sigma)}$ in der oberen Schranke verzichtet werden kann, ist dies für den Schätzer $\hat{\beta}^L_\lambda$ von β^* nicht der Fall. Das bedeutet, falls zwischen zwei Komponenten $X_{1j}, X_{1j'}$ von X_1 mit $j, j' \in S^*$ starke Korrelationen bestehen, so sind die Schätzungen $\hat{\beta}^L_{\lambda,j}$, $\hat{\beta}^L_{\lambda,j'}$ eventuell sehr schlecht. Dazu betrachten wir folgenden Extremfall: Ist $X_{1j} = X_{1j'}$, so genügt es, X_{1j} zur Beschreibung von Y_1 zu benutzen. Auch wenn zum Beispiel $\beta^*_j = \beta^*_{j'} = 0{,}5$ gilt, so wird der Lasso-Schätzer eher $\hat{\beta}^L_{\lambda,j} \approx 1$, $\hat{\beta}^L_{\lambda,j'} \approx 0$ liefern, damit $\hat{\beta}^L_\lambda$ möglichst wenige Nicht-Null-Komponenten besitzt. Beim Ridge-Schätzer hingegen wäre $\hat{\beta}^R_{\lambda,j} \approx 0{,}5$, $\hat{\beta}^R_{\lambda,j'} \approx 0{,}5$, da dann $0{,}5^2 + 0{,}5^2 = 0{,}5 \leq 1 = 1^2 + 0^2$ und somit der Bestrafungsterm J^R einen geringeren Wert liefert. Eine Korrektur des Verhaltens des Lasso-Schätzers bei stark korrelierten Komponenten von X_1 kann daher durch eine Kombination von Ridge- und Lasso-Schätzer erreicht werden:

Definition 2.46 (Elastisches Netz)
Sei $\lambda > 0$, $\alpha \in (0,1)$. Für $\beta = (\beta_0, \ldots, \beta_d) \in \mathbb{R}^{d+1}$ sei $J^{EN}(\beta) = \sum_{j=1}^d \{\alpha|\beta_j| + (1-\alpha)\beta_j^2\}$. Der Schätzer

$$\hat{\beta}^{EN}_{\lambda,\alpha} := \arg\min_{\beta \in \mathbb{R}^{d+1}} \left\{ \hat{R}_n(\beta) + \lambda J^{EN}(\beta) \right\}$$

heißt *Elastisches-Netz-Schätzer* (engl. *elastic net*). ♦

Der Name leitet sich aus der Form der durch J^{EN} erzeugten Nebenbedingung ab, die Bestrafungen zwischen der $\|\cdot\|_1$- und der $\|\cdot\|_2$-Norm für β vorsieht. Es gibt mannigfaltige Modifikationen des Lasso-Schätzers und darauf aufbauende statistische Methoden, um die Nicht-Null-Komponenten von β^* mit noch geringerem Fehler zu ermitteln (vgl. [12] für einen Überblick).

2.5 Übungen

1. Begründen Sie, warum $\hat{\Sigma} = \frac{1}{n}\mathbb{X}^T\mathbb{X}$ mit $\mathbb{X}$ auf Lemma 2.19 stets positiv semidefinit ist. Zeigen Sie, dass $\hat{\Sigma}$ sogar positiv definit ist, wenn $d+1 \leq n$ gilt und $\tilde{X}_1, \ldots, \tilde{X}_{d+1}$ linear unabhängig sind.
2. Leiten Sie die explizite Formel für den Ridge-Schätzer in Gl. (2.11) aus der Definition her.
3. Implementieren Sie den KQ-, Ridge- und Lasso-Algorithmus und wenden Sie diese auf Trainingsdaten aus den Beispielen 2.14 und 2.15 für verschiedene $d \geq 5$ an.

Allgemeines zu Klassifikationsproblemen

3

Inhaltsverzeichnis

Liegt ein Klassifikationsproblem mit $K \in \mathbb{N}$, $K \geq 2$ Klassen vor, so sind Trainingsdaten $(X_i, Y_i) \overset{\text{iid}}{\sim} \mathbb{P}^{(X,Y)}$, $i = 1, \ldots, n$ gegeben mit $X_i \in \mathcal{X} = \mathbb{R}^d$ und $Y_i \in \mathcal{Y} = \{1, \ldots, K\}$. Zur Bewertung von Entscheidungsregeln nutzen wir die 0–1-Verlustfunktion $L(y, s) = \mathbb{1}_{\{y \neq s\}}$. In diesem Fall ist jede Abbildung $f^* : \mathcal{X} \to \mathcal{Y}$ mit

$$f^*(x) \in \underset{k \in \{1,\ldots,K\}}{\arg\max}\ \mathbb{P}(Y = k | X = x), \quad x \in \mathcal{X} \tag{3.1}$$

Bayes-Regel (vgl. Bemerkung 1.3). Die Bayes-Regel f^* ist auf den Punkten $x \in \mathcal{X}$ nicht eindeutig festgelegt, bei welchen mehr als ein $k \in \{1, \ldots, K\}$ Maximierer in Gl. (3.1) ist. Alle Punkte dieser Art sind gegeben durch die Menge

$$U^* := \bigcup_{k_1, k_2 \in \{1,\ldots,K\}, k_1 \neq k_2} U^*_{k_1,k_2},$$

wobei

$$U^*_{k_1,k_2} := \{x \in \mathcal{X} : k_1, k_2 \in \underset{k \in \{1,\ldots,K\}}{\arg\max}\ \mathbb{P}(Y = k | X = x)\}.$$

Die Beschreibung einer Theorie, welche auch die Punkte $x \in U^*$ mit einschließt, ist aufwendig; wir werden daher in diesem Buch eine Vereinfachung vornehmen. Im Folgenden nehmen wir stets an, dass

$$\mathbb{P}(X \in U^*) = 0, \tag{3.2}$$

S. Richter, *Statistisches und maschinelles Lernen*,
https://doi.org/10.1007/978-3-662-59354-7_3

d. h., wir gehen davon aus, dass für jede Beobachtung (fast sicher) eine eindeutige „beste Klasse“ existiert und die Bayes-Regel f^* wie im Regressionsfall $\mathbb{P}^X$-f.s. eindeutig bestimmt ist. Gl. (3.2) wird in den besprochenen Modellen entweder implizit vorausgesetzt (durch konkrete Vorgabe der Verteilung von X gegeben Y wie zum Beispiel in Modellannahme 4.1) oder für theoretische Aussagen explizit gefordert (wie zum Beispiel in Satz 4.48, Gl. (4.54)). Ist X stetig verteilt, so entspricht Gl. (3.2) der Annahme, dass U^* eine Lebesgue-Nullmenge ist. Anschaulich kann man sich U^* dann als eine höchstens $(d-1)$-dimensionale Untermannigfaltigkeit von $\mathbb{R}^d$ vorstellen; ist $\mathcal{X} = \mathbb{R}^2$, so entspricht U^* also einer Menge von Kurven im Raum $\mathbb{R}^2$.

3.1 Entscheidungsregionen, -ränder und Diskriminantenfunktionen

3.1.1 Entscheidungsregionen und Entscheidungsränder

Bei Klassifikationsproblemen induziert jede Entscheidungsregel f eine Aufteilung des Raumes $\mathcal{X}$ in Bereiche (sogenannte *Entscheidungsregionen*), welche alle Punkte $x \in \mathcal{X}$ beinhalten, die f der gleichen Klasse zuordnet. Wir formalisieren dies in der folgenden Definition. Es bezeichne ∂A den Rand einer Menge $A \subset \mathcal{X}$ (d. h. relativ zu $\mathcal{X}$).

Definition 3.1
Sei $f : \mathcal{X} \to \mathcal{Y}$ eine beliebige Entscheidungsregel und f^* eine Bayes-Regel. Für $k \in \{1, \ldots, K\}$ nennen wir

$$\Omega_k := \Omega_k(f) := \{x \in \mathcal{X} : f(x) = k\} \tag{3.3}$$

von f induzierte *Entscheidungsregionen* (engl. *decision regions*) und $\partial\Omega_k$ *Entscheidungsränder* (engl. *decision boundaries*) für die Klasse k. Im Falle $f = f^*$ nennen wir $\Omega_k^* := \Omega_k(f^*)$ bzw. $\partial\Omega_k^*$ *optimale Entscheidungsregionen* bzw. *optimale Entscheidungsränder.* ♦

Ein Beispiel für mögliche optimale Entscheidungsregionen und die zugehörigen Entscheidungsränder im Falle $\mathcal{X} = [0, 1]^2$ und $K = 2$ ist in Abb. 3.1 zu sehen.

Für den Bayes-Fehler kann in diesem Zusammenhang folgende Aussage formuliert werden.

Lemma 3.2 Der Bayes-Fehler ist

$$R(f^*) = \sum_{k=1}^{K} \mathbb{P}(X \notin \Omega_k^* | Y = k)\mathbb{P}(Y = k).$$

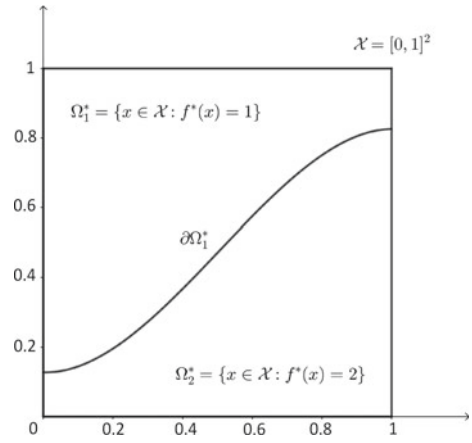


Abb. 3.1 Beispiel für mögliche optimale Entscheidungsregionen und die zugehörigen Entscheidungsränder

Beweis Es gilt

$$R(f^*) = \mathbb{P}(Y \neq f^*(X)) = \sum_{k=1}^{K} \mathbb{P}(k \neq f^*(X)|Y = k)\mathbb{P}(Y = k)$$
$$= \sum_{k=1}^{K} \mathbb{P}(X \notin \Omega_k^*|Y = k)\mathbb{P}(Y = k),$$

wobei die zweite Gleichheit aus dem Satz der totalen Wahrscheinlichkeit folgt.

Es gibt eine natürliche Beziehung zwischen den Entscheidungsrändern und den Punkten aus U^*, auf welchen f^* nicht eindeutig festgelegt ist. Der Entscheidungsrand $\partial\Omega_{k_1}^* \cap \partial\Omega_{k_2}^*$ enthält die Punkte x, bei welchen die Bayes-Regel f^* von einem Funktionswert k_1 in einen anderen k_2 *übergeht.* Falls die Abbildung $x \mapsto \mathbb{P}(Y = k|X = x)$ stetig ist (dies wird in den meisten Modellannahmen implizit angenommen), kann ein solcher Übergang kann nur stattfinden, wenn $x \in U_{k_1,k_2}^*$ liegt, d. h., wenn es bei x mehr als einen Maximierer in Gl. (3.1) gibt. In diesem Sinne gilt anschaulich

$$„\partial\Omega_{k_1}^* \cap \partial\Omega_{k_2}^* = U_{k_1,k_2}^*“. \tag{3.4}$$

In pathologischen Sonderfällen gilt diese Gleichheit nicht strikt mathematisch, in den meisten praktischen Situationen ist dies allerdings nicht von Belang.

Gl. (3.4) in Verbindung mit der Stetigkeitsannahme an $x \mapsto \mathbb{P}(Y = k|X = x)$ bedeutet, dass es für einen Algorithmus $\hat{f}_n$ gerade für Argumente $x \in \mathcal{X}$ in der Nähe der optimalen Entscheidungsränder sehr schwierig ist, die Bayes-Regel f^* korrekt zu schätzen.

3.1.2 Formulierung der Bayes-Regel mittels Diskriminantenfunktionen

Ist $h : [0, 1] \times \mathcal{X} \to \mathbb{R}$ eine beliebige Funktion, die strikt monoton wachsend in ihrer ersten Komponente ist, so ist Gl. (3.1) äquivalent zu

$$f^*(x) \in \underset{k\in\{1,\dots,K\}}{\arg\max}\ \delta_k^*(x), \qquad \delta_k^*(x) = h(\mathbb{P}(Y = k|X = x), x). \tag{3.5}$$

Die Darstellung wird häufig als Hilfsmittel genutzt, um Modellannahmen an f^* leichter formulieren oder Entscheidungsregeln f leichter definieren zu können. Erlauben Funktionen $\delta_k^*, k = 1, \dots, K$ die obige Darstellung, so spricht man von optimalen Diskriminantenfunktionen.

Definition 3.3
Sei $f : \mathcal{X} \to \mathcal{Y}$ eine beliebige Entscheidungsregel, Ω_k ihre Entscheidungsregionen und f^* die Bayes-Regel. Die Abbildungen $\delta_k : \mathcal{X} \to \mathbb{R}$ $(k = 1, \dots, K)$ heißen *Diskriminantenfunktionen* für f (engl. *discriminant functions*), falls

$$\Omega_k = \left\{x \in \mathcal{X} : \delta_k(x) = \max_{j\in\{1,\dots,K\}} \delta_j(x)\right\}$$

bzw. äquivalent dazu

$$f(x) = \underset{k\in\{1,\dots,K\}}{\arg\max}\ \delta_k(x), \quad k = 1, \dots, K. \tag{3.6}$$

Ist $f = f^*$, so heißen $\delta_k^* = \delta_k$ *optimale Diskriminantenfunktionen.* ♦

Wir geben nun ein weiteres wichtiges (von $\delta_k^*(x) = \mathbb{P}(Y = y|X = x)$ verschiedenes) Beispiel für optimale Diskriminantenfunktionen.

Lemma 3.4 Besitzt $\mathbb{P}^{X,Y}$ eine Wahrscheinlichkeitsdichte $g_{X,Y}$ bzgl. des Produkts des Lebesguemaßes auf $\mathbb{R}^d$ mit dem Zählmaß auf $\{1, \dots, K\}$ und ist $\pi_k := \mathbb{P}(Y = k)$ sowie $g_k(x) := \frac{g_{X,Y}(x,k)}{\pi_k}$ die bedingte Dichte von X gegeben $Y = k$, so gilt: Die Funktionen

$$\delta_k^*(x) = g_k(x) \cdot \pi_k, \quad k = 1, \dots, K$$

sind optimale Diskriminantenfunktionen.

Beweis Es bezeichne g_X die Dichte von X bzgl. des Lebesgue-Maßes auf $\mathbb{R}^d$. Aufgrund des Satzes von Bayes gilt für die optimale Diskriminantenfunktion $\mathbb{P}(Y = k|X = x)$:

$$\mathbb{P}(Y = k|X = x) = \frac{g_{X,Y}(x, k)}{g_X(x)} = \frac{g_{X|Y=k}(x) \cdot \mathbb{P}(Y = k)}{g_X(x)} = \frac{g_k(x)\pi_k}{g_X(x)}$$

Multiplikation mit $g_X(x)$ ist eine monotone Funktion in k, daher folgt die Behauptung.

Damit haben wir gesehen: Modellannahmen an f^* können formuliert werden durch

- Annahmen an $\mathbb{P}(Y = k|X = x)$ aufgrund der Darstellung in Gl. (3.1),
- (grafische) Annahmen an die Entscheidungsregionen Ω_k^*,
- Annahmen an die bedingten Dichten g_k aufgrund von Lemma 3.4.

3.2 Formulierung von Algorithmen und Berechnung des Excess Bayes Risks

Sind Trainingsdaten $(X_i, Y_i), i = 1, \ldots, n$ gegeben, so verlagert sich die Bestimmung eines Algorithmus $\hat{f}_n$ für f^* auf das Finden von Schätzern $\hat{\delta}_{n,k}$ für optimale Diskriminantenfunktionen δ_k^*. Sind solche Schätzer verfügbar, so ist gemäß Gl. (3.6) durch

$$\hat{f}_n(x) :\in \underset{k\in\{1,\ldots,K\}}{\arg\max}\ \hat{\delta}_{n,k}(x), \quad x \in \mathcal{X} \tag{3.7}$$

ein Algorithmus gegeben. Wir werden in den Abschn. 4.1, 4.2 und Kap. 5 einige Beispiele für $\hat{\delta}_{n,k}$ und deren Ermittlung sehen.

Um das Excess Bayes Risk von $\hat{f}_n$ zu berechnen, wird dann eine Möglichkeit benötigt, Aussagen über die Qualität von $\hat{\delta}_{n,k}$ in Hinsicht auf die Schätzung von δ_k^* auf $R(\hat{f}_n) - R(f^*)$ zu übertragen. In diesem Buch werden wir das Excess Bayes Risk von Klassifikationsalgorithmen der Übersichtlichkeit halber nur für den Fall von $K = 2$ Klassen diskutieren (auch wenn diese für mehr als zwei Klassen anwendbar sind); es ist naheliegend, dass sich zumindest die Konvergenzrate für mehr als zwei Klassen nicht substanziell ändert.

Möglichkeit 1
Im Falle von $K = 2$ Klassen hat f^* die vereinfachte Darstellung

$$f^*(x) = \underset{k\in\{1,2\}}{\arg\max}\ \eta(x) = \begin{cases} 1, & \eta(x) \geq \frac{1}{2}, \\ 2, & \eta(x) < \frac{1}{2} \end{cases} \quad \text{mit} \quad \eta(x) := \mathbb{P}(Y = k|X = x). \tag{3.8}$$

Falls (z. B. durch monotone Transformationen von $\hat{\delta}_{n,k}$) diese Darstellung auch für $\hat{f}_n$ erreicht werden kann, d. h.

$$\hat{f}_n(x) = \begin{cases} 1, & \hat{\eta}(x) \geq \frac{1}{2}, \\ 2, & \hat{\eta}(x) < \frac{1}{2}, \end{cases}$$

mit einer geeigneten Funktion $\hat{\eta}$ abhängig von den Trainingsdaten, und ist $\hat{\eta}$ ein Schätzer für η, so kann mittels des folgenden Lemmas eine Aussage über $R(\hat{f}_n) - R(f^*)$ aus einer Aussage über $\mathbb{E}|\hat{\eta}(X) - \eta(X)|$ abgeleitet werden:

Lemma 3.5 Sei $\mathcal{Y} = \{1, 2\}$, und sei $L(y, s) = \mathbb{1}_{\{y \neq s\}}$ die 0–1-Verlustfunktion. Sei $\eta(x) := \mathbb{P}(Y = 1|X = x)$. Ist $f : \mathcal{X} \to \mathcal{Y}$ eine Entscheidungsregel der Form

$$f(x) = \begin{cases} 1, & m(x) \geq \frac{1}{2} \\ 2, & m(x) < \frac{1}{2} \end{cases}$$

mit einem geeigneten $m : \mathcal{X} \to [0, 1]$, so gilt

$$R(f) - R(f^*) \leq 2\mathbb{E}|m(X) - \eta(X)|. \tag{3.9}$$

Beweis Es gilt

$$\begin{aligned} R(f) - R(f^*) = \mathbb{P}(f(X) \neq Y) - \mathbb{P}(f^*(X) \neq Y) = \mathbb{E}\big[\mathbb{P}(f(X) \neq Y|X) \\ - \mathbb{P}(f^*(X) \neq Y|X)\big]. \end{aligned} \tag{3.10}$$

Wegen $Y \in \mathcal{Y} = \{1, 2\}$ folgt

$$\begin{aligned} &\mathbb{P}(f(X) \neq Y|X) - \mathbb{P}(f^*(X) \neq Y|X) \\ &= \sum_{k=1,2} \big\{\mathbb{P}(f(X) \neq k, Y = k|X) - \mathbb{P}(f^*(X) \neq k, Y = k|X)\big\} \\ &= \sum_{k=1,2} \big\{\mathbb{1}_{\{f(X)\neq k\}} - \mathbb{1}_{\{f^*(X)\neq k\}}\big\} \cdot \mathbb{P}(Y = k|X) \\ &= \big\{\mathbb{1}_{\{f(X)\neq 1\}} - \mathbb{1}_{\{f^*(X)\neq 1\}}\big\} \cdot \eta(X) + \big\{\mathbb{1}_{\{f(X)\neq 2\}} - \mathbb{1}_{\{f^*(X)\neq 2\}}\big\} \cdot (1 - \eta(X)). \end{aligned}$$

Da $f(X) \in \{1, 2\}$, gilt $\mathbb{1}_{\{f(X)\neq 1\}} = f(X) - 1$ und $\mathbb{1}_{\{f(X)\neq 2\}} = 2 - f(X)$. Damit erhalten wir:

$$\begin{aligned} &\mathbb{P}(f(X) \neq Y|X) - \mathbb{P}(f^*(X) \neq Y|X) \\ &\quad = \{f(X) - f^*(X)\}\eta(X) + \{f^*(X) - f(X)\}(1 - \eta(X)) \\ &\quad = 2\Big(\eta(X) - \frac{1}{2}\Big) \cdot \{f(X) - f^*(X)\} \end{aligned}$$

Für $x \in \mathcal{X}$ gilt $f(x) - f^*(x) \in \{-1, 0, 1\}$. Es gilt außerdem

$$f(x) - f^*(x) \leq 0 \iff f^*(x) = 2 \iff \eta(x) < \frac{1}{2} \iff \eta(x) - \frac{1}{2} < 0.$$

Daher folgt

$$\mathbb{P}(f(X) \neq Y|X) - \mathbb{P}(f^*(X) \neq Y|X) = 2\Big|\eta(X) - \frac{1}{2}\Big| \cdot \mathbb{1}_{\{f(X)\neq f^*(X)\}}. \tag{3.11}$$

Im Falle $f(X) \neq f^*(X)$ können zwei Möglichkeiten auftreten:

- $f(X) = 1, f^*(X) = 2 \Rightarrow \eta(X) < \frac{1}{2} \leq m(X) \Rightarrow \left|\eta(X) - \frac{1}{2}\right| \leq \left|\eta(X) - m(X)\right|$
- $f(X) = 2, f^*(X) = 1 \Rightarrow \eta(X) \geq \frac{1}{2} > m(X) \Rightarrow \left|\eta(X) - \frac{1}{2}\right| \leq \left|\eta(X) - m(X)\right|$

Daher folgt aus Gl. (3.11):

$$\mathbb{P}(f(X) \neq Y|X) - \mathbb{P}(f^*(X) \neq Y|X) \leq 2\left|\eta(X) - m(X)\right|$$

Einsetzen in Gl. (3.10) liefert die Behauptung.

Bemerkung Auf die rechte Seite von Gl. (3.9) kann die Cauchy-Schwarz-Ungleichung ($\mathbb{E}|Z| \leq \mathbb{E}[Z^2]^{1/2}$ für reellwertige Zufallsvariablen Z) angewandt werden, um Resultate für den *mean squared error* von $\hat{\eta}$ nutzen zu können.

Möglichkeit 2
Wir haben in Lemma 3.4 gesehen, dass f^* unter geeigneten Annahmen auch die Darstellung

$$f^*(x) = \underset{k\in\{1,2\}}{\arg\max}\ \eta_k(x), \quad \text{mit} \quad \eta_k(x) := \pi_k g_k(x) \tag{3.12}$$

besitzt. *Falls* (z. B. durch monotone Transformationen von $\hat{\delta}_{n,k}$) diese Darstellung auch für $\hat{f}_n$ erreicht werden kann, d. h.

$$\hat{f}_n(x) = \underset{k\in\{1,2\}}{\arg\max}\ \hat{\eta}_k(x),$$

mit Schätzern $\hat{\eta}_k$ für η_k, so gibt es auch hier eine Aussage über das Risiko von $\hat{f}_n$ in Abhängigkeit von Aussagen über $\hat{\eta}_k$:

Lemma 3.6 Sei $\mathcal{Y} = \{1, 2\}$, und sei $L(y, s) = \mathbb{1}_{\{y\neq s\}}$ die 0–1-Verlustfunktion. Ist $f(x) = \arg\max_{k\in\{1,2\}} m_k(x)$ mit Abbildungen $m_k : \mathcal{X} \to [0, \infty)$ $(k \in \{1, 2\})$, so folgt

$$R(f) - R(f^*) \leq 2 \sum_{k=1,2} \mathbb{E}\left|\frac{m_k(X)}{g(X)} - \frac{\eta_k(X)}{g(X)}\right|, \tag{3.13}$$

wobei $\eta_k(x) := \pi_k g_k(x)$ und g die Lebesgue-Dichte von X bezeichnet.

Beweis Definiere $m(x) := \frac{m_1(x)}{m_1(x)+m_2(x)}$. Dann gilt für alle $x \in \mathcal{X}$:

$$m(x) \geq \frac{1}{2} \quad \Longleftrightarrow \quad m_1(x) \geq m_2(x) \quad \Longleftrightarrow \quad f(x) = 1,$$

d. h., f hat die Struktur aus Lemma 3.5. Nach dem Satz von Bayes und dem Satz von der totalen Wahrscheinlichkeit gilt $\eta(x) := \mathbb{P}(Y = 1|X = x) = \frac{\eta_1(x)}{\eta_1(x)+\eta_2(x)}$ und $g(x) = \eta_1(x) + \eta_2(x)$. Daher ist

$$m - \eta = \frac{1}{g}\Big[\frac{m_2}{m_1 + m_2} \cdot (m_1 - \eta_1) + \frac{m_1}{m_1 + m_2}(\eta_2 - m_2)\Big].$$

Es folgt

$$|m - \eta| \leq \Big|\frac{m_1 - \eta_1}{g}\Big| + \Big|\frac{m_2 - \eta_2}{g}\Big|.$$

Anwendung von Lemma 3.5 liefert die Behauptung.

Bemerkung Auf die rechte Seite von Gl. (3.13) kann die Cauchy-Schwarz-Ungleichung angewandt werden, um Resultate über den mean squared error von $\hat{\eta}_k$ nutzen zu können.

3.3 Bestimmung von Algorithmen durch Lösen von Optimierungsproblemen

Hier untersuchen wir, wie der Standardansatz aus Bemerkung 1.14 zur Ermittlung von Algorithmen $\hat{f}_n$ genutzt werden kann. Wir erklären dies im Falle von $K = 2$ Klassen und Klassenmenge $\mathcal{Y} = \{-1, +1\}$. Viele komplexere Algorithmen wie beispielsweise die Support Vector Machine (vgl. Abschn. 4.4) oder das Boosting (vgl. Abschn. 6.5) können als Anwendung dieses Ansatzes aufgefasst werden. Die Einschränkung auf zwei Klassen bedeutet in der Praxis keine Einschränkung; in Abschn. 3.4 wird gezeigt, wie aus Algorithmen für Zwei-Klassen-Probleme Algorithmen für Mehr-Klassen-Probleme gewonnen werden können.

In seiner Grundformulierung lautet der Ansatz aus Bemerkung 1.14:

$$\hat{f}_n \in \arg\min_{f \in \mathcal{F}} \hat{R}_n(f), \qquad \hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{Y_i \neq f(X_i)\}}, \tag{3.14}$$

wobei $\mathcal{F} \subset \{f : \mathcal{X} \to \mathcal{Y} \text{ messbar}\}$ eine geeignete Funktionenklasse ist. Häufig ist dabei die Funktionenklasse grafisch motiviert, d. h., man erwartet, dass $f \in \mathcal{F}$ äquivalent zu der Annahme ist, dass die zu f gehörigen Entscheidungsränder die Form

$$\partial\Omega_1 = \{x \in \mathcal{X} : \delta(x) = 0\} \tag{3.15}$$

mit $\delta \in \tilde{\mathcal{F}} \subset \{\delta : \mathcal{X} \to \mathbb{R} \text{ messbar}\}$ besitzen.

Beispiel 3.7 Sollen die Entscheidungsränder eine lineare Form besitzen, so muss

$$\tilde{\mathcal{F}} = \{\delta : \mathcal{X} \to \mathbb{R} \mid \text{Es gibt } \beta \in \mathbb{R}^d, \beta_0 \in \mathbb{R}, \, so\, dass\, für\, alle\, x \in \mathcal{X} : \delta(x) = \beta^T x + \beta_0\}$$

gewählt werden.

Wegen Gl. (3.15) gilt

$$f(x) = \text{sign}(\delta(x)), \qquad \text{sign}(z) := \begin{cases} 1, & z \geq 0, \\ -1, & z < 0, \end{cases} \tag{3.16}$$

d. h., die Funktion δ „erweitert" den Wertebereich von f sinnvoll auf den gesamten Raum $\mathbb{R}$. Andererseits besitzt δ dann auch automatisch eine Interpretation als Diskriminantenfunktion von f, denn mit der Setzung $\delta_1(x) := \delta(x)$, $\delta_{-1}(x) := 0$ gilt:

$$f(x) = \underset{k \in \{-1,+1\}}{\arg\max} \; \delta_k(x)$$

Auch für sehr kleine Klassen $\mathcal{F}$ ist eine Berechnung des Minimierers aus Gl. (3.14) auf dem Computer oft nicht effizient möglich. Der Grund ist die fehlende Konvexität der 0–1-Verlustfunktion L und der diskrete Wertebereich von f. Im Folgenden zeigen wir, wie das Optimierungsproblem näherungsweise gelöst werden kann, indem wir die einzelnen Bestandteile schrittweise durch ähnliche Größen ersetzen.

- Schritt 1: Der diskrete Wertebereich von f kann vermieden werden, indem wir f durch das zugehörige δ ersetzen. Eine Approximation von Gl. (3.14) kann dann erhalten werden durch

  $$\hat{f}_n(x) = \text{sign}(\hat{\delta}_n(x)), \quad x \in \mathcal{X},$$

  wobei

  $$\hat{\delta}_n \in \underset{\delta \in \tilde{\mathcal{F}}}{\arg\min} \; \hat{R}_n(\delta), \qquad \hat{R}_n(\delta) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{Y_i \neq \delta(X_i)\}}. \tag{3.17}$$

- Schritt 2: Durch die oben getätigte Änderung ist die Verlustfunktion $L(y, s) = \mathbb{1}_{\{y \neq s\}}$ kein adäquates Maß für die Messung der Ungleichheit zwischen $Y_i \in \mathcal{Y}$ und $\delta(X_i) \in \mathbb{R}$ mehr. Stattdessen drückt $-Y_i \cdot \delta(X_i)$ nun mit seinem Vorzeichen aus, ob Y_i richtig klassifiziert wurde. Die abgewandelte Verlustfunktion $L_0(y, s) = \mathbb{1}_{\{-ys \geq 0\}}$ erfüllt

  $$L(y, s) = \mathbb{1}_{\{y \neq s\}} = \mathbb{1}_{\{-ys \geq 0\}} = L_0(y, s), \quad y, s \in \mathcal{Y},$$

  und stellt damit eine Erweiterung von L dar, die auch für Werte $s \in \mathbb{R}$ eine korrekte Bestrafung vornimmt. Das Optimierungsproblem in Gl. (3.17) wandelt sich zu

  $$\hat{\delta}_n \in \underset{\delta \in \tilde{\mathcal{F}}}{\arg\min} \; \hat{R}_n^0(\delta), \qquad \hat{R}_n^0(\delta) = \frac{1}{n} \sum_{i=1}^{n} L_0(Y_i, \delta(X_i)). \tag{3.18}$$

- Schritt 3: Da L_0 nicht stetig ist, sind auch Optimierungsprobleme der Form Gl. (3.18) noch schwer lösbar. Wir werden sehen, dass viele anschaulich motivierte und über die Zeit entstandene Algorithmen anstelle von L_0 eine *Approximation* $\tilde{L}$ von L_0 als Verlustfunktion verwenden, die *stetig* und *konvex* in $s \in \mathbb{R}$ ist. Die Konvexität von $\tilde{L}$ garantiert hierbei die Existenz eindeutiger Minimierer, und die Theorie konvexer Optimierungsprobleme liefert effiziente Berechnungsmöglichkeiten für $\hat{\delta}_n$ in Gl. (3.18).
  Im Kontext von Gl. (3.16) und der Interpretation von δ als optimale Diskriminantenfunktion entsprechen große (positive oder negative) Werte von $\delta(x)$ anschaulich der Vorstellung, dass die für $f(x)$ getroffene Entscheidung sehr „sicher" ist. Ist diese Entscheidung falsch, so sollte sie sehr stark bestraft werden. Die bereits oben identifizierten Terme $-Y_i\delta(X_i)$ geben somit ein sinnvolles Maß an, wie „schlecht" eine Entscheidung mittels δ war, denn: $-Y_i\delta(X_i)$ ist groß, falls sich δ bei X_i mit hoher Sicherheit für die falsche Klasse entschieden hat, und klein, falls sich δ bei X_i mit hoher Sicherheit für die korrekte Klasse entschieden hat. Für $\tilde{L}$ ergibt sich damit auf natürliche Weise die Form

$$\tilde{L}(y, s) = \phi(-y \cdot s),$$

  wobei $\phi : \mathbb{R} \to [0, \infty)$ eine nichtnegative, monoton wachsende und konvexe Funktion ist, und das Optimierungsproblem in Gl. (3.18) wird wie in Bemerkung 3.8 angenähert.

Bemerkung 3.8 (Standardvorgehen zur Ermittlung von $\hat{f}_n$) Ist $\tilde{L} : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}$, $\tilde{L}(y, s) = \phi(-ys)$ mit einer nichtnegativen monoton wachsenden, konvexen Funktion $\phi : \mathbb{R} \to [0, \infty)$ und $\mathcal{Y} = \{-1, +1\}$, so wähle

$$\hat{\delta}_n \in \arg\min_{\delta \in \tilde{\mathcal{F}}} \tilde{R}_n(\delta), \qquad \tilde{R}_n(\delta) = \frac{1}{n}\sum_{i=1}^{n} \tilde{L}(Y_i, \delta(X_i)) \tag{3.19}$$

und setze $\hat{f}_n(x) = \operatorname{sign}(\hat{\delta}_n(x))$, $x \in \mathcal{X}$.

In diesem Buch werden wir zum Beispiel folgende Funktionen ϕ betrachten:

$$\phi_{\text{hinge}}(z) = (1 - z)^+, \quad \phi_{\exp}(z) = e^z, \quad \phi_{\log}(z) = \log(1 + e^z)$$

Nicht jede Wahl von ϕ in Bemerkung 3.8 führt jedoch zu einem sinnvollen Algorithmus $\hat{f}_n$. Durch die Formulierung in Gl. (3.19) dürfen wir erwarten, dass $\hat{\delta}_n \approx \delta^*$, wobei

$$\delta^* \in \arg\min_{\delta \in \tilde{F}} \tilde{R}(\delta), \qquad \tilde{R}(\delta) := \mathbb{E}\tilde{L}(Y, \delta(X)).$$

Damit folgt $\hat{f}_n = \operatorname{sign}(\hat{\delta}_n) \approx \operatorname{sign}(\delta^*)$. Die Konvergenz $\hat{f}_n \to f^*$ $(n \to \infty)$ kann dann nur gelten, wenn $f^* = \operatorname{sign}(\delta^*)$ gilt.

Definition 3.9 (Kalibrierungsbedingung (K1))
Eine Verlustfunktion $\tilde{L}(y, s) = \phi(-ys)$ erfüllt die *Kalibrierungsbedingung,* wenn

$$\delta^* :\in \operatorname*{arg\,min}_{\delta:\mathcal{X}\to\mathbb{R}\text{ messbar}} \tilde{R}(\delta), \qquad \tilde{R}(\delta) := \mathbb{E}\tilde{L}(Y, \delta(X))$$

erfüllt:

$$f^*(x) = \operatorname{sign}(\delta^*(x)), \quad x \in \mathcal{X} \tag{3.20}$$

♦

Das folgende Resultat erleichtert die Verifikation von (K1), indem eine explizite Formel für δ^* in Abhängigkeit der Größe $\eta(x) = \mathbb{P}(Y = 1|X = x)$ vorgegeben wird.

Lemma 3.10 Sei $\phi : \mathbb{R} \to [0, \infty)$ nichtnegativ, monoton nichtfallend und messbar. Sei

$$\Phi_\eta(z) := \phi(-z)\eta + \phi(z)(1 - \eta)$$

und $A := \{\eta \in [0, 1] : z \mapsto \Phi_\eta(z)$ hat einen eindeutigen globalen Minimierer$\}$. Jede Funktion $\delta^* \in \arg\min_{\delta:\mathcal{X}\to\mathbb{R}\text{ messbar}} \tilde{R}(\delta)$ ist dann auf $\{x : \eta(x) \in A\}$ $\mathbb{P}^X$-f.s. eindeutig bestimmt und gegeben durch

$$\delta^*(x) = \operatorname*{arg\,min}_{z\in\mathbb{R}} \Phi_{\eta(x)}(z), \qquad x \in \mathcal{X}.$$

Beweis Es gilt

$$\begin{aligned}
\tilde{R}(\delta) &= \mathbb{E}\phi(-Y\delta(X)) = \mathbb{E}[\mathbb{E}[\phi(-Y\delta(X))|X]] \\
&= \mathbb{E}\big[\phi(-\delta(X))\mathbb{P}(Y = 1|X) + \phi(\delta(X))\mathbb{P}(Y = -1|X)\big] \\
&= \int \Big\{\phi(-\delta(x))\underbrace{\mathbb{P}(Y = 1|X = x)}_{=\eta(x)} + \phi(\delta(x))\underbrace{\mathbb{P}(Y = -1|X = x)}_{=1-\eta(x)}\Big\}\, d\mathbb{P}^X(x) \\
&= \int_{\{x:\eta(x)\in A\}} \Phi_{\eta(x)}(\delta(x))\, d\mathbb{P}^X + \int_{\{x:\eta(x)\in A^c\}} \Phi_{\eta(x)}(\delta(x))\, d\mathbb{P}^X
\end{aligned}$$

Da die Werte von δ^* auf $\{x : \eta(x) \in A\}$ und $\{x : \eta(x) \in A^c\}$ unabhängig voneinander gewählt werden können, dürfen die Integrale getrennt minimiert werden. Das erste Integral wird $\mathbb{P}^X$-f.s. eindeutig durch die angegebene Funktion δ^* minimiert, da der Integrand punktweise minimiert wird.

Oft ist $A = (0, 1)$, d. h., es müssen formal die Punkte x mit $\eta(x) = 1$ und $\eta(x) = 0$ ausgeschlossen werden. Ist beispielsweise ϕ stetig differenzierbar, so erfüllt δ^* auf $\{x : \eta(x) \in A\}$ $\mathbb{P}^X$-f.s. die Gleichung $\frac{\phi'(\delta^*(x))}{\phi'(-\delta^*(x))} = \frac{\eta(x)}{1-\eta(x)}$.

Es muss dann jeweils anhand der konkreten Wahl von ϕ geprüft werden, ob (K1) erfüllt ist. Wir werden dies erst tun, wenn ein konkretes ϕ vorliegt. Für ϕ_{hinge} diskutieren wir dies in Lemma 4.46, für $\phi_{\exp}$ und $\phi_{\log}$ in Lemma 6.42.

Berechnung des Excess Bayes Risks
Für den Minimierer $\hat{\delta}_n$ von Gl. (3.19) erhält man mittels statistischer Methoden typischerweise eine Formel für den Ausdruck $\tilde{R}(\hat{\delta}_n) - \tilde{R}(\delta^*)$. Damit eine Rate für das Excess Bayes Risk $R(\hat{f}_n) - R(f^*)$ von $\hat{f}_n$ daraus ermittelt werden kann, benötigen wir:

(K2) *Risikoübertragungsformel:* Es gibt eine Ungleichung, die für jedes messbare $\delta : \mathcal{X} \to \mathbb{R}$ und das zugehörige $f = \operatorname{sign}(\delta) : \mathcal{X} \to \mathcal{Y}$ die Größe $R(f) - R(f^*)$ nach oben durch $\tilde{R}(\delta) - \tilde{R}(\delta^*)$ abschätzt.

Wir zeigen nun einige Sätze, welche uns für verschiedene ϕ eine solche Formel liefern. Wir nutzen dafür eine durch [8] inspirierte Darstellung. In [41] wird folgendes Lemma gezeigt:

Lemma 3.11 Ist $\phi : \mathbb{R} \to [0, \infty)$ eine nichtnegative, monoton wachsende und konvexe Funktion und gibt es Konstanten $c \geq 0$, $s \geq 1$ so dass für alle $\eta \in [0, 1]$ gilt:

$$\left|\frac{1}{2} - \eta\right|^s \leq c^s (1 - H(\eta)), \qquad H(\eta) := \min_{z \in \mathbb{R}} \Phi_\eta(z) \tag{3.21}$$

($\Phi_\eta(z)$ aus Lemma 3.10), so gilt für jedes messbare $\delta : \mathcal{X} \to \mathbb{R}$ und $f := \operatorname{sign}(\delta)$:

$$R(f) - R(f^*) \leq 2c(\tilde{R}(\delta) - \tilde{R}(\delta^*))^{1/s}$$

Der Beweis nutzt die Darstellung aus Gl. (3.11) und verwendet dann die Voraussetzung Gl. (3.21). Man kann zeigen (vgl. [8, nach Lemma 4]), dass $\phi_{\exp}$ und $\phi_{\log}$ die Voraussetzung Gl. (3.21) mit $s = 2$ und $c = \sqrt{2}$ erfüllen und ϕ_{hinge} dies mit $s = 1$ und $c = \frac{1}{2}$ (vgl. Lemma 4.46) erfüllt.

Durch Analyse der Konvergenzraten in der Praxis sowie theoretische Aussagen über die bestmöglichen Konvergenzraten wurde entdeckt, dass die Abschätzungen aus Lemma 3.11 in vielen (aber nicht allen) Fällen schlechte obere Schranken liefern. Diese Fälle, in welchen die Abschätzung verbessert werden kann, werden durch folgende Bedingung charakterisiert (vgl. [3, 33]):

Definition 3.12 (Low-noise condition)
Gibt es $\alpha \in [0, 1]$, $\beta > 0$, so dass für alle messbaren $f : \mathcal{X} \to \mathcal{Y}$ gilt:

$$\mathbb{P}(f(X) \neq f^*(X)) \leq \beta (R(f) - R(f^*))^\alpha, \tag{3.22}$$

so erfüllt die Verteilung von (X, Y) die *low-noise condition.* ♦

Offensichtlich erfüllt jede Verteilung die *low-noise condition* mit $\alpha = 0$, $\beta = 1$. In [3, Lemma 5] wird eine anschaulich leichter verständliche Charakterisierung gezeigt:

Lemma 3.13 Die Bedingung Gl. (3.22) mit $\alpha \in [0, 1)$, $\beta > 0$ ist äquivalent zu: Es gibt eine Konstante $c' > 0$, so dass für alle $\varepsilon > 0$ gilt:

$$\mathbb{P}\Big(0 < \big|\eta(X) - \frac{1}{2}\big| \leq \varepsilon\Big) \leq c' \varepsilon^{\frac{\alpha}{1-\alpha}}$$

Im Falle $\alpha = 1$ ist Gl. (3.22) äquivalent zu

$$\mathbb{P}\Big(0 < \big|\eta(X) - \frac{1}{2}\big| \leq \frac{1}{2\beta}\Big) = 0.$$

Die *low-noise condition* charakterisiert also das Verhalten der Wahrscheinlichkeit $\eta(x) = \mathbb{P}(Y = 1|X = x)$. Ist für ein $x \in \mathcal{X}$ der Wert $\eta(x)$ nahe bei $\frac{1}{2}$, so ist die Auswahl der korrekten Klasse schwierig. Die Wahrscheinlichkeit der Menge $\{0 < \big|\eta(X) - \frac{1}{2}\big| \leq \varepsilon\}$ misst, wie häufig solche „schwierigen Entscheidungen" im gegebenen Problem betrachtet werden müssen. Im Extremfall $\alpha = 1$ hat $\eta(x)$ stets einen positiven Abstand von mindestens $\frac{1}{2\beta}$ zu $\frac{1}{2}$.

Gilt Gl. (3.22), so erhalten wir eine bessere obere Schranke für $R(f) - R(f^*)$ (vgl. [3, Theorem 3]):

Lemma 3.14 Ist ϕ wie in Lemma 3.11 mit Konstanten $c > 0$, $s \geq 1$ und gibt es Konstanten $\alpha \in [0, 1]$, $\beta > 0$, so dass Gl. (3.22) erfüllt ist, so gilt für jedes messbare $\delta : \mathcal{X} \to \mathbb{R}$ und $f := \operatorname{sign}(\delta)$:

$$R(f) - R(f^*) \leq \Big(\frac{2^s c}{\beta^{1-s}} (\tilde{R}(\delta) - \tilde{R}(\delta^*))\Big)^{\frac{1}{s-s\alpha+\alpha}}$$

Während für $\phi = \phi_{\text{hinge}}$ mit $s = 1$ keine Verbesserung möglich ist, verbessert sich der Exponent für $\phi \in \{\phi_{\exp}, \phi_{\log}\}$ von $\frac{1}{2}$ (mit $\alpha = 0$) zu 1 (mit $\alpha = 1$). Es zeigt sich, dass die Annahme der *low-noise condition* nötig sein kann, um schnellere Konvergenzraten des Excess Bayes Risks nachweisen zu können.

Mit Lemma 3.14 kann jedoch keine Verbesserung der Konvergenzrate des Excess Bayes Risks in Hinsicht auf die *Beziehung* zwischen Dimension d der X_i und der Anzahl n der Trainingsdaten erreicht werden, da dieser Zusammenhang bereits komplett in $\tilde{R}(\hat{\delta}_n) - \tilde{R}(\delta^*)$ enthalten ist und durch Potenzieren nicht verändert wird.

3.4 Reduktion von Mehr-Klassen-Problemen

Grundsätzlich können Probleme mit $K > 2$ auch mit Klassifizierern von Zwei-Klassen-Problemen gelöst werden. Hier gehen wir davon aus, dass diese für Probleme mit Klassen $\mathcal{Y} = \{-1, +1\}$ konstruiert wurden. Wir stellen zwei populäre Strategien vor.

Bemerkung 3.15 (Reduktion von Mehr-Klassen-Problemen) Gegeben seien Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$ mit $Y_i \in \mathcal{Y} = \{1, \ldots, K\}$. Dann können Mehr-Klassen-Probleme wie folgt mittels Klassifizierern für Zwei-Klassen Probleme gelöst werden:

(i) ‚One-vs.-rest'-Strategie: Vergleiche für ein festes $x \in \mathcal{X}$ jeweils die Trainingsdaten einer Klasse κ mit den restlichen Daten aller anderen Klassen. Mittels der Diskriminantenfunktion an der Stelle x erhalten wir ein Maß, wie wahrscheinlich die Klasse κ für x im Gegensatz zu allen anderen Klassen ist. Formal: Für $\kappa = 1, \ldots, K$, definiere

$$\tilde{Y}_i^{(\kappa)} := \begin{cases} 1, & Y_i = \kappa, \\ -1, & Y_i \neq \kappa. \end{cases}$$

Bestimme einen Klassifizierer $\hat{f}_n^{(\kappa)}$ bzw. Diskriminantenfunktionen $\hat{\delta}_k^{(\kappa)}$ $(k = -1, 1)$, basierend auf $(X_i, \tilde{Y}_i^{(\kappa)})$, $i = 1, \ldots, n$. Definiere damit den Klassifizierer

$$\hat{f}_n(x) := \underset{\kappa \in \{1,\ldots,K\}}{\arg\max}\ \hat{\delta}_1^{(\kappa)}(x).$$

Für ein $x \in \mathcal{X}$ wird also die Klasse κ ausgewählt, welche beim *gleichzeitigen* Vergleich mit allen anderen möglichen Klassen die höchste Diskriminantenfunktion $\hat{\delta}_1^{(\kappa)}(x)$ besitzt.

(ii) ‚One-vs.-one'-Strategie: Für $\{\kappa, k\} \subset \{1, \ldots, K\}$ $(\kappa \neq k)$ bestimme einen Klassifizierer $\hat{f}_n^{\{\kappa,k\}}$ basierend auf (X_i, Y_i), $i \in \{s \in \{1, \ldots, n\} : Y_s \in \{\kappa, k\}\}$. Setze

$$\hat{f}_n(x) = \underset{\kappa \in \{1,\ldots,K\}}{\arg\max}\ \#\{k \in \{1, \ldots, K\} : \hat{f}_n^{\{\kappa,k\}}(x) = \kappa\}. \tag{3.23}$$

Diese Technik nennt man auch *majority vote* – für ein $x \in \mathcal{X}$ wird die Klasse κ ausgewählt, welche sich durch Vergleich mittels der Klassifizierer $\hat{f}_n^{\kappa,k}(x)$ am häufigsten gegen jede andere Klasse k durchsetzt. Dieser Ansatz kann noch verfeinert werden, wenn man statt der reinen Anzahl der Entscheidungen für Klasse κ in Gl. (3.23) mit Diskriminantenfunktionen arbeitet.

Falls K relativ groß ist, hat die One-vs.-one-Strategie einen wichtigen Vorteil in der Praxis: Für die Ermittlung der Klassifizierer $\hat{f}_n^{\{\kappa,k\}}$ wird jeweils nur ein Teil der Trainingsdaten benutzt. Falls die Berechnung eines Klassifizierers basierend auf n Trainingsdaten mehr als $c \cdot n$ Rechenschritte benötigt ($c > 0$ eine Konstante), spart man durch diese Strategie Rechenzeit.

Lineare Methoden für Klassifizierungsprobleme und SVMs

4

Inhaltsverzeichnis

Im Folgenden betrachten wir Klassifizierungsprobleme mit K Klassen, d. h., es liegen Trainingsdaten $(X_i, Y_i) \overset{\text{iid}}{\sim} \mathbb{P}^{(X,Y)}$, $i = 1, \ldots, n$ vor mit $X_i \in \mathcal{X} = \mathbb{R}^d$ und $Y_i \in \mathcal{Y} = \{1, \ldots, K\}$. Zur Bewertung von Entscheidungsregeln nutzen wir die 0-1-Verlustfunktion $L(y, s) = \mathbb{1}_{\{y \neq s\}}$. In diesem Fall erfüllt die Bayes-Regel (vgl. Bemerkung 1.3):

$$f^*(x) \in \underset{k \in \{1,\ldots,K\}}{\arg\max}\ \mathbb{P}(Y = k | X = x), \quad x \in \mathcal{X}$$

4.1 Lineare und quadratische Diskriminantenanalyse (LDA)

Bei der Diskriminantenanalyse nehmen wir an, dass die Beobachtungen X aus jeder Klasse $Y = k$ normalverteilt sind mit verschiedenen Mittelwerten und Kovarianzmatrizen. Wie wir sehen werden, führt dies zu optimalen Diskriminantenfunktionen, die lineare oder quadratische Funktionen in x sind, was den Namen der Methode erklärt.

S. Richter, *Statistisches und maschinelles Lernen*,
https://doi.org/10.1007/978-3-662-59354-7_4

4.1.1 Modellannahme und Algorithmus

Modellannahme 4.1 (Diskriminantenanalyse) Es seien $\mu_k \in \mathbb{R}^d$, $\Sigma_k \in \mathbb{R}^{d\times d}$ symmetrisch positiv definit ($k = 1, \ldots, K$). Es gelte

$$(X|Y = k) \sim N(\mu_k, \Sigma_k) \quad (k = 1, \ldots, K),$$

d. h., für die bedingte Dichte g_k von X gegeben $Y = k$ gilt:

$$g_k(x) = \frac{1}{(2\pi)^{d/2}\det(\Sigma_k)^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right)$$

Es sei $\pi_k := \mathbb{P}(Y = k)$. Damit können wir eine spezielle Wahl der optimalen Diskriminantenfunktionen in diesem Modell ausdrücken:

Lemma 4.2

(i) Die Abbildungen

$$\delta_k^*(x) := -\frac{1}{2}\log\det(\Sigma_k) - \frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k + \log\pi_k$$

sind optimale Diskriminantenfunktionen.

(ii) Gilt $\Sigma_k = \Sigma \in \mathbb{R}^{d\times d}$ für alle $k = 1, \ldots, K$, so sind

$$\delta_k^*(x) := x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log\pi_k$$

optimale Diskriminantenfunktionen.

Beweis

(i) In obigem Modell gilt:

$$\log[g_k(x)\pi_k] = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log\det(\Sigma_k) - \frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k + \log\pi_k$$

Da $\log(\cdot) + \frac{d}{2}\log(2\pi)$ eine monoton wachsende Funktion ist, folgt die erste Behauptung mit Lemma 3.4.

(ii) In diesem Fall kann $\log(\cdot) + \frac{d}{2}\log(2\pi) + \frac{1}{2}\log\det(\Sigma) + \frac{1}{2}x^T\Sigma^{-1}x$ als monotone Transformation gewählt werden, da Σ nicht mehr von der Klasse k abhängt.

Weil die Diskriminantenfunktionen in diesem Modell notwendig linear bzw. quadratisch sind, heißt dieses Verfahren auch lineare (bzw. quadratische) Diskriminanzanalyse. Wegen Definition 3.3 bedeutet das, dass die optimalen Entscheidungsränder

$$\Omega_k^* = \{x \in \mathcal{X} : \delta_k^*(x) = \max_{j \in \{1,\dots,K\}} \delta_j^*(x)\}$$

in diesem Modell die Klassen mit linearen (im Fall (i) oben) bzw. quadratischen (Fall (ii) oben) Funktionen trennen.

Die Schätzung der optimalen Diskriminantenfunktionen erfolgt durch Schätzung der damit verknüpften Parameter (Mittelwerte und Kovarianzmatrizen) aus den Trainingsdaten. Die Ergebnisse von Lemma 4.2 können dann direkt in die Praxis übertragen werden, indem man μ_k, Σ_k und π_k durch Schätzer ersetzt:

Definition 4.3 (Lineare und quadratische Diskriminanzanalyse [LDA/QDA])
Seien $n_k = \#\{i : Y_i = k\}$ Anzahl der Beobachtungen der Klasse $k \in \{1, \dots, K\}$. Definiere

$$\hat{\pi}_k = \frac{n_k}{n}, \qquad \hat{\mu}_k = \frac{1}{n_k} \sum_{i:Y_i=k} X_i, \qquad \hat{\Sigma}_k = \frac{1}{n_k} \sum_{i:Y_i=k} (X_i - \hat{\mu}_k)(X_i - \hat{\mu}_k)^T.$$

Sei

$$\hat{\delta}_k^{QDA}(x) = -\frac{1}{2} x^T \hat{\Sigma}_k^{-1} x + x^T \hat{\Sigma}_k^{-1} \hat{\mu}_k + \left[\log \hat{\pi}_k - \frac{1}{2} \hat{\mu}_k^T \hat{\Sigma}_k^{-1} \hat{\mu}_k - \frac{1}{2} \log \det(\hat{\Sigma}_k) \right].$$

Der *QDA-Klassifizierer* ist definiert durch $\hat{f}_n^{QDA}(x) = \arg\max_{k \in \{1,\dots,K\}} \hat{\delta}_k^{QDA}(x)$.

Gilt zusätzlich $\Sigma_k = \Sigma \in \mathbb{R}^{d \times d}$ für alle $k = 1, \dots, K$, so definiere

$$\hat{\delta}_k^{LDA}(x) = x^T \hat{\Sigma}^{-1} \hat{\mu}_k - \frac{1}{2} \hat{\mu}_k^T \hat{\Sigma}^{-1} \hat{\mu}_k + \log \hat{\pi}_k, \qquad \hat{\Sigma} := \frac{1}{n} \sum_{k=1}^{K} n_k \hat{\Sigma}_k.$$

Der *LDA-Klassifizierer* ist definiert durch $\hat{f}_n^{LDA}(x) = \arg\max_{k \in \{1,\dots,K\}} \hat{\delta}_k^{LDA}(x)$. ♦

Zur Veranschaulichung der Unterschiede zwischen LDA und QDA betrachten wir folgendes Beispiel:

Beispiel 4.4 Wir betrachten $K = 3$ und die folgenden zwei Fälle:

a) LDA-Modell mit $\mu_1 = (2, 0)^T$, $\mu_2 = (0, 2)^T$, $\mu_3 = (2, 2)^T$ und $\Sigma = 0{,}2 \cdot I_{2\times 2}$,
b) QDA-Modell mit $\mu_1 = (0, 1)^T$, $\mu_2 = (0, -1)^T$, $\mu_3 = (3, 0)^T$ und $\Sigma_1 = \Sigma_2 = diag(2, 0{,}2)$, $\Sigma_3 = diag(0{,}2, 2)$.

Im Folgenden betrachten wir jeweils $n = 150$ Trainingsdaten, wobei $n_1 = n_2 = n_3 = 50$ Trainingsdaten pro Klasse vorliegen. In Abb. 4.1 und 4.2 sind jeweils typische Realisierungen der Fälle (a), (b) zu sehen sowie die Anwendung von LDA und QDA. Man kann sehen, dass eine korrekte Spezifikation (LDA bei (a), QDA bei (b)) jeweils zu einer besseren Schätzung der Entscheidungsregionen führt.

Die Verteilungsvoraussetzung der Diskriminantenanalyse (d. h. $(X|Y = k)$ ist normalverteilt) ist eine starke Einschränkung an die Verwendbarkeit der Methode in der Praxis. Sind beispielsweise Realisierungen *einer* Klasse typischerweise sehr ‚groß' und sehr ‚klein', aber

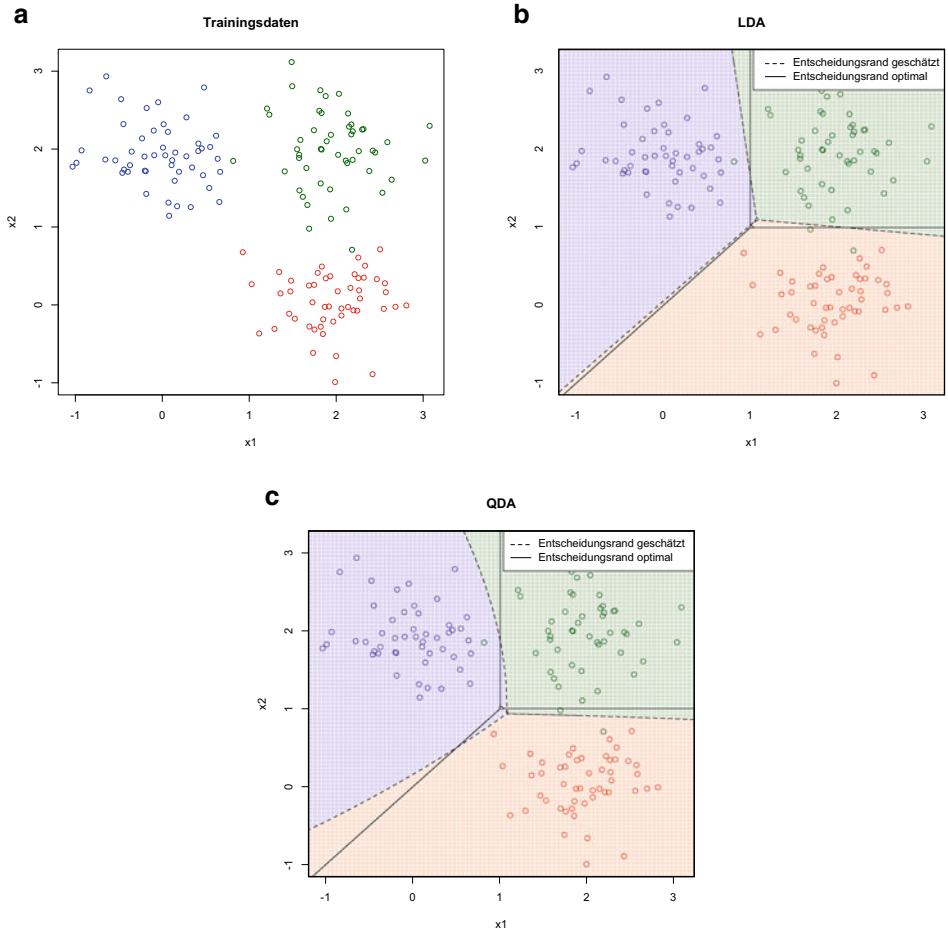


Abb. 4.1 Darstellung der Klassifizierer $\hat{f}_n^{LDA}$ und $\hat{f}_n^{QDA}$ der LDA/QDA angewandt auf die Daten aus Beispiel 4.4(a). **a** Trainingsdaten. **b**, **c** Anwendung von LDA/QDA; die farblichen Schattierungen entsprechen den geschätzten Entscheidungsregionen der jeweiligen Klassen

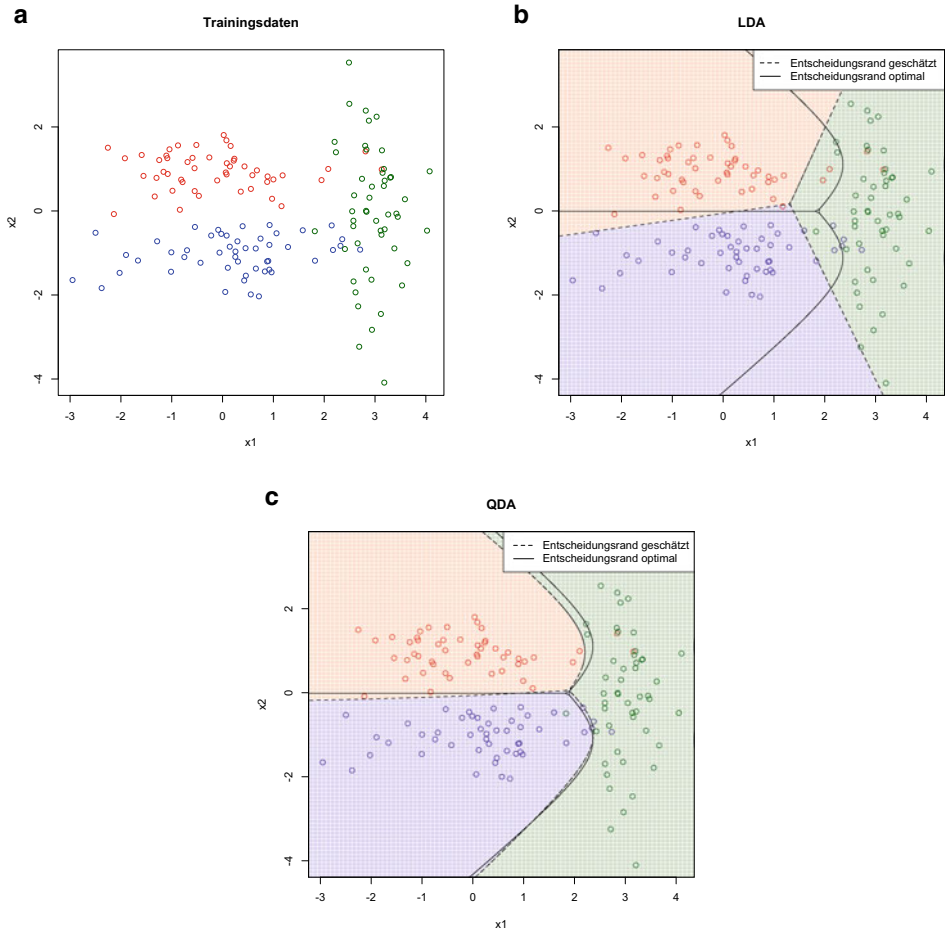


Abb. 4.2 Darstellung der Klassifizierer $\hat{f}_n^{LDA}$ und $\hat{f}_n^{QDA}$ der LDA/QDA angewandt auf die Daten aus Beispiel 4.4(b). **a** Trainingsdaten. **b**, **c** Anwendung von LDA/QDA; die farblichen Schattierungen entsprechen den geschätzten Entscheidungsregionen der jeweiligen Klassen

nicht ‚mittelgroß', so kann dies nicht durch eine Normalverteilung beschrieben werden. Wir betrachten dazu folgendes Beispiel:

Beispiel 4.5 Sei $K = 2$. Es seien Y_i i. i. d. mit $\mathbb{P}(Y_1 = 1) = \mathbb{P}(Y_1 = 2) = \frac{1}{2}$ und

$$(X_i|Y_i = 1) \sim \frac{1}{2}N(\mu_1, \Sigma) + \frac{1}{2}N(\mu_3, \Sigma), \qquad (X_i|Y_i = 2) \sim \frac{1}{2}N(\mu_2, \Sigma) + \frac{1}{2}N(\mu_4, \Sigma),$$

wobei $\Sigma = \frac{1}{4} I_{2\times 2}$ und

$$\mu_1 = \begin{pmatrix} -4 \\ 0 \end{pmatrix}, \quad \mu_2 = \begin{pmatrix} -2 \\ 0 \end{pmatrix}, \quad \mu_3 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \mu_2 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}.$$

In Abb. 4.3 wurden $n = 240$ Trainingsdaten (X_i, Y_i) basierend auf obigem Modell erzeugt und $\hat{f}_n^{LDA}$, $\hat{f}_n^{QDA}$ angewandt. Man sieht, dass beide Klassifizierer an der korrekten Zuordnung der Klassen scheitern.

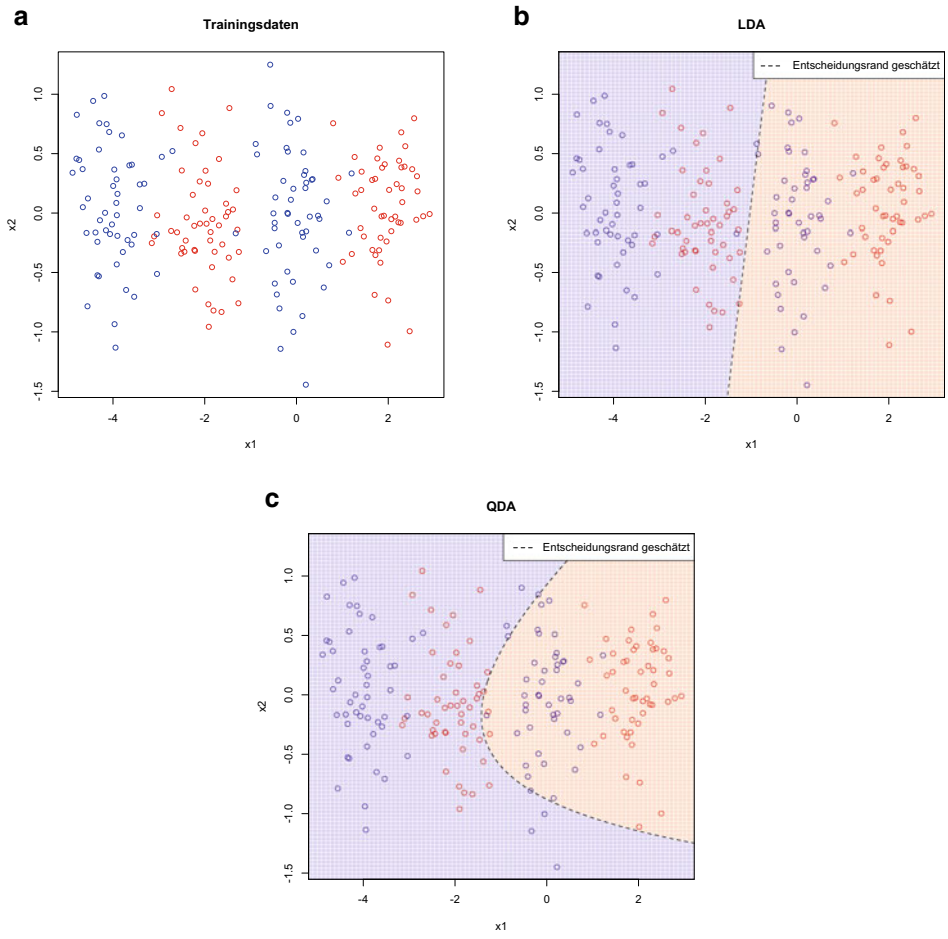


Abb. 4.3 Darstellung der Klassifizierer $\hat{f}_n^{LDA}$ und $\hat{f}_n^{QDA}$ der LDA/QDA angewandt auf Trainingsdaten aus Beispiel 4.5. **a** Trainingsdaten. **b**, **c** Anwendung von LDA/QDA; die farblichen Schattierungen entsprechen den geschätzten Entscheidungsregionen der jeweiligen Klassen

4.1.2 Theoretische Resultate

Aus Gründen der Übersichtlichkeit beschränken wir uns hier auf Resultate für $K = 2$ Klassen. Im Gegensatz zu vielen anderen Verfahren gibt es bei der LDA aufgrund der einfachen Modellannahmen geschlossene Ausdrücke für den Bayes-Fehler gemäß folgendem Lemma:

Lemma 4.6 Sei $K = 2$ und gelte $\Sigma = \Sigma_1 = \Sigma_2$. Sei $T := \log \frac{\pi_2}{\pi_1}$ und $\Delta := (\mu_2 - \mu_1)^T \Sigma^{-1}(\mu_2 - \mu_1)$ (sog. *Mahalanobis-Distanz*). Dann gilt für den Bayes-Fehler:

$$R(f^*) = \pi_1 \Phi\left(\frac{T - \Delta/2}{\sqrt{\Delta}}\right) + \pi_2 \left(1 - \Phi\left(\frac{T + \Delta/2}{\sqrt{\Delta}}\right)\right),$$

wobei $\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) \mathrm{d}z$ die Verteilungsfunktion der Standardnormalverteilung ist.

Beweis Sei

$$\Lambda(x) := \delta_1^*(x) - \delta_2^*(x) = x^T \Sigma^{-1}(\mu_1 - \mu_2) - \frac{1}{2}\mu_1^T \Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^T \Sigma^{-1}\mu_2 - T.$$

Für die optimalen Entscheidungsregionen gilt: $\Omega_2^* = \{x \in \mathbb{R}^d : \delta_2^*(x) \geq \delta_1^*(x)\} = \{x \in \mathbb{R}^d : \Lambda(x) \leq 0\}$, $\Omega_0 = \{x \in \mathbb{R}^d : \Lambda(x) \geq 0\}$. (Insbesondere:

$$\partial\Omega_2 = \{x \in \mathbb{R}^d : \Lambda(x) = 0\} = \{\langle \Sigma^{-1}(\mu_1 - \mu_2), x - \frac{1}{2}(\mu_1 + \mu_2)\rangle = T\}$$

Der optimale Entscheidungsrand ist also eine Hyperebene mit Normalenvektor $\Sigma^{-1}(\mu_1 - \mu_2)$)

Lemma 3.2 impliziert

$$\begin{aligned} R(f^*) &= \pi_1 \mathbb{P}(X \in \Omega_2 | Y = 1) + \pi_2 \mathbb{P}(X \in \Omega_1 | Y = 2) \\ &= \pi_1 \mathbb{P}(\Lambda(X) \leq 0 | Y = 1) + \pi_2 \mathbb{P}(\Lambda(X) \geq 0 | Y = 2). \end{aligned}$$

Nun ist

$$[\Lambda(X)|Y = 1] \sim N\left(\frac{1}{2}\Delta - T, \Delta\right), \qquad [\Lambda(X)|Y = 2] \sim N\left(-\frac{1}{2}\Delta - T, \Delta\right).$$

Daher gilt mit $Z \sim N(0, 1)$:

$$R(f^*) = \pi_1 \cdot \mathbb{P}\left(Z \leq -\frac{\frac{1}{2}\Delta + T}{\sqrt{\Delta}}\right) + \pi_2 \cdot \mathbb{P}\left(Z \geq \frac{\frac{1}{2}\Delta + T}{\sqrt{\Delta}}\right),$$

dies liefert die Behauptung.

Es bezeichne $\|A\|_2$ die Frobenius-Norm einer Matrix A, und $\lambda_{min}(A), \lambda_{max}(A)$ ihren kleinsten bzw. größten Eigenwert. Da $\hat{\pi}_k$, $\hat{\mu}_k$, $\hat{\Sigma}_k$ auf Mittelwerten von i. i. d. Zufallsvariablen basieren, gibt es eine Konstante $c > 0$ unabhängig von d, n mit der Eigenschaft

$$\mathbb{E}[(\hat{\pi}_k - \pi_k)^2] \leq \frac{c}{n}, \qquad \mathbb{E}[\|\hat{\mu}_k - \mu_k\|_2^2] \leq c \cdot \frac{d}{n}, \qquad \mathbb{E}[\|\hat{\Sigma}_k - \Sigma_k\|_2^2] \leq c \cdot \frac{d^2}{n}. \tag{4.1}$$

Mittels Lemma 3.5 kann die schlechteste Rate $\frac{d^2}{n}$ auf $\hat{f}_n^{QDA}$ übertragen werden:

Satz 4.7 Es gelte die Modellannahme aus Definition 4.1 und $\pi_k \in (0, 1)$, $k = 1, 2$. Außerdem gebe es Konstanten $c_u, c_o > 0$ unabhängig von d, so dass $c_o \geq \lambda_{max}(\Sigma) \geq \lambda_{min}(\Sigma) \geq c_u$ gilt. Falls $d = d_n$ mit $\frac{d^2}{n} \to 0$, so ist

$$\lim_{c \to \infty} \limsup_{n \to \infty} \mathbb{P}(R(\hat{f}_n^{QDA}) - R(f^*) \geq c \cdot d n^{-1/2}) = 0,$$

d. h., $\hat{f}_n^{QDA}$ lernt mit Konvergenzrate $dn^{-1/2}$.

Beweis Definiere $\eta(x) := \mathbb{P}(Y = 1|X = x) = \frac{\pi_1 g_1(x)}{\pi_1 g_1(x) + \pi_2 g_2(x)}$ und $\hat{d}_k(x) := \frac{\hat{\pi}_k \hat{g}_k(x)}{\hat{\pi}_1 \hat{g}_1(x) + \hat{\pi}_2 \hat{g}_2(x)}$, wobei

$$\hat{g}_k(x) = \exp\left[-\frac{d}{2}\log(2\pi) - \frac{1}{2}\log\det(\hat{\Sigma}_k) - \frac{1}{2}(x - \hat{\mu}_k)\hat{\Sigma}_k^{-1}(x - \hat{\mu}_k)\right]. \tag{4.2}$$

Dann gilt

$$\hat{f}_n^{QDA}(x) = \arg\max_{k=1,2} \hat{d}_k(x) = \begin{cases} 1, & \hat{d}_1(x) \geq \frac{1}{2} \\ 2, & \hat{d}_1(x) < \frac{1}{2} \end{cases},$$

denn $\hat{\delta}_k = \log(\hat{d}_k) + \log(\hat{\pi}_1 \hat{g}_1 + \hat{\pi}_2 \hat{g}_2) + \frac{d}{2}\log(2\pi)$ ist eine monotone Transformation von $\hat{d}_k$. Seien $T_n := (X_i, Y_i)_{i=1,\ldots,n}$ die Trainingsdaten. Mit Lemma 3.5(i) folgt

$$R(\hat{f}_n^{QDA}) - R(f^*) \leq 2\mathbb{E}[|\hat{d}_1(X) - \eta(X)|\,|T_n].$$

Es gilt $\hat{d}_1(X) = S\left(\log\left(\frac{\hat{\pi}_2}{\hat{\pi}_1} \cdot \frac{\hat{g}_2(X)}{\hat{g}_1(X)}\right)\right)$ mit $S : \mathbb{R} \to \mathbb{R}$, $S(x) = \frac{1}{1+e^x}$. S ist Lipschitz-stetig mit Lipschitz-Konstante $\frac{1}{4}$, daher folgt

$$\begin{aligned} &R(\hat{f}_n^{QDA}) - R(f^*) \\ &\leq \frac{1}{2}\sum_{k=1,2} |\log(\hat{\pi}_k) - \log(\pi_k)| + \frac{1}{2}\sum_{k=1,2} \mathbb{E}[|\log(\hat{g}_k(X)) - \log(g_k(X))|\,|T_n]. \end{aligned} \tag{4.3}$$

Für den restlichen Beweis sind sehr viele Resultate aus der Matrixalgebra nötig, deren ausführliche Diskussion wir hier überspringen. Stattdessen geben wir nur einen Überblick über die erhaltenen Zwischenresultate. Es gibt Konstanten $c_1, c_2 > 0$ abhängig von

$\lambda_{min}(\Sigma_1)$, $\lambda_{min}(\Sigma_2)$, so dass für jede symmetrische positiv semidefinite Matrix $\Sigma \in \mathbb{R}^{d\times d}$ und $k \in \{1, 2\}$ gilt:

$$\|\Sigma - \Sigma_k\|_2 \leq c_1 \quad \Rightarrow \quad \lambda_{min}(\Sigma) \geq c_2$$

Außerdem gibt es eine Konstante $c_3 > 0$, so dass für jedes $a \in [0, 1]$ und $k \in \{1, 2\}$ gilt:

$$|a - \pi_k| \leq \frac{\pi_k}{2} \quad \Rightarrow \quad |\log(a) - \log(\pi_k)| \leq c_3|a - \pi_k|$$

Definiere das Ereignis

$$A := \bigcap_{k\in\{1,2\}} \{\|\Sigma - \Sigma_k\|_2 \leq c_1\} \cap \bigcap_{k\in\{1,2\}} \{|\hat{\pi}_k - \pi_k| \leq \frac{\pi_k}{2}\}.$$

Wegen $(X|Y = k) \sim N(\mu_k, \Sigma_k)$ gibt es eine Konstante c_4 abhängig von $\lambda_{min}(\Sigma_k)$, $\lambda_{max}(\Sigma_k)$, $k = 1, 2$, so dass auf A gilt:

$$\begin{aligned}\mathbb{E}[|\log(\hat{g}_k(X)) - \log(g_k(X))|\,|T_n] &\leq \mathbb{E}[|\log(\hat{g}_k(X)) - \log(g_k(X))|^2\,|T_n]^{1/2} \\ &\leq c_4 \left\{\|\hat{\Sigma}_k - \Sigma_k\|_2 + \|\hat{\mu}_k - \mu_k\|_2\right\}\end{aligned}$$

und damit, eingesetzt in Gl. (4.3),

$$\begin{aligned}&R(\hat{f}_n^{QDA}) - R(f^*) \\ &\leq \frac{c_3}{2} \sum_{k=1,2} |\hat{\pi}_k - \pi_k| + \frac{c_4}{2} \sum_{k=1,2} \left\{\|\hat{\Sigma}_k - \Sigma_k\|_2 + \|\hat{\mu}_k - \mu_k\|_2\right\}. \qquad (4.4)\end{aligned}$$

Wegen

$$\mathbb{P}\left(R(\hat{f}_n^{QDA}) - R(f^*) \geq cdn^{-1/2}\right) \leq \mathbb{P}\left(R(\hat{f}_n^{QDA}) - R(f^*) \geq cdn^{-1/2}, A\right) + \mathbb{P}(A^c),$$

der Markov-Ungleichung und den Abschätzungen aus Gl. (4.4) und (4.1) folgt die Behauptung.

Entsprechend Satz 4.7 können gute Eigenschaften von $\hat{f}_n^{QDA}$ (und analog auch für $\hat{f}_n^{LDA}$) nur im Falle $d^2 \ll n$ erwartet werden. Für hochdimensionale Daten mit $d^2 \gg n$ ist die Diskriminantenanalyse in der oben präsentierten Form nicht geeignet, um die Entscheidungsränder korrekt zu schätzen.

4.2 Logistische Regression

Bei der linearen Diskriminanzanalyse haben wir gesehen, dass die optimalen Entscheidungsränder durch lineare Funktionen gegeben waren. Allerdings entstand dieses Verhalten durch eine konkrete Verteilungsannahme an die Beobachtungen in den Klassen, d. h. an $X|Y = k$.

Wir zeigen hier zunächst, dass auch andere Modelle lineare optimale Entscheidungsränder erlauben. Fordern wir, dass die Entscheidungsränder (stückweise) lineare Funktionen sind, so erwarten wir, dass $\beta^{*(k)} \in \mathbb{R}^d$, $\beta_0^{*(k)} \in \mathbb{R}$ $(k = 1, \ldots, K)$ existieren mit

$$\Omega_k^* = \left\{ x \in \mathcal{X} : \left(\beta_0^{*(k)} + x^T \beta^{*(k)} \right) = \underset{j \in \{1,\ldots,K\}}{\arg\max} \left(\beta_0^{*(j)} + x^T \beta^{*(j)} \right) \right\}. \tag{4.5}$$

Aufgrund Definition 3.3 kann die Annahme in Gl. (4.5) auch wie folgt formuliert werden: Die Verteilung (X, Y) muss optimale Diskriminantenfunktionen δ_k^*, $k = 1, \ldots, K$ besitzen mit

$$\delta_k^*(x) = \beta_0^{*(k)} + x^T \beta^{*(k)}, \quad x \in \mathcal{X}, \quad k \in \{1, \ldots, K\}.$$

Andererseits wissen wir, dass jede Funktion $h : [0, 1] \times \mathcal{X} \to \mathbb{R}$, die streng monoton wachsend in ihrer ersten Komponente ist, durch

$$\delta_k^*(x) = h\left(\mathbb{P}(Y = k|X = x), x\right), \quad x \in \mathcal{X}, \quad k \in \{1, \ldots, K\}$$

eine optimale Diskriminantenfunktion definiert. Das bedeutet, solange die Gleichung

$$h\left(\mathbb{P}(Y = k|X = x), x\right) \overset{!}{=} \beta_0^{*(k)} + x^T \beta^{*(k)}, \quad x \in \mathcal{X}, \quad k \in \{1, \ldots, K\} \tag{4.6}$$

erfüllt ist, besitzt das Modell lineare optimale Entscheidungsränder.

Damit durch Gl. (4.6) ein sinnvoller Ansatz beschrieben wird, muss die Funktion h den gesamten Bereich der reellen Zahlen erreichen können. Bei der *logistischen Regression* wird

$$h(p, x) := \log\left(\frac{p}{\mathbb{P}(Y = K|X = x)} \right)$$

gewählt, d. h., die Klassenwahrscheinlichkeiten werden entsprechend $\mathbb{P}(Y = K|X = x)$ reskaliert und dann logarithmiert. Für die Klasse $k = K$ ist damit Gl. (4.6) automatisch erfüllt, wenn $\beta_0^{*(K)} = 0$ und $\beta^{*(K)} = 0$ gesetzt wird. Der Übersichtlichkeit halber gehen wir außerdem im Folgenden von $\beta_0^{(k)} = 0$ $(k = 1, \ldots, K - 1)$ aus.

Modellannahme 4.8 (Logistische Regression) Für $k = 1, \ldots, K - 1$ gebe es $\beta^{*(k)} \in \mathbb{R}^d$ mit

$$\log\left(\frac{\mathbb{P}(Y = k|X = x)}{\mathbb{P}(Y = K|X = x)} \right) = x^T \beta^{*(k)}, \qquad x \in \mathbb{R}^d. \tag{4.7}$$

Dann sind $\tilde{\delta}_k^*(x) = x^T \beta^{*(k)}$ $(k = 1, \ldots, K - 1)$, $\tilde{\delta}_K^*(x) \equiv 0$ optimale Diskriminantenfunktionen.

Aus der Modellannahme folgt auch eine explizite Darstellung von $\mathbb{P}(Y = k|X = x)$:

Lemma 4.9 Äquivalent zur Modellannahme Gl. (4.7) ist

$$\mathbb{P}(Y = k|X = x) = \frac{\exp(x^T\beta^{*(k)})}{1 + \sum_{j=1}^{K-1}\exp(x^T\beta^{*(j)})}, \quad k = 1, \ldots, K-1,$$
$$\mathbb{P}(Y = K|X = x) = \frac{1}{1 + \sum_{j=1}^{K-1}\exp(x^T\beta^{*(j)})}.$$

Beweis Wende $\exp(\cdot)$ auf beiden Seiten von Gl. (4.7) an und nutze $\sum_{k=1}^{K}\mathbb{P}(Y = k|X = x) = 1$.

Ermittlung von Schätzern für $\delta_k^*(x)$ Wegen $\delta_k^*(x) = x^T\beta^{*(k)}$ verlagert sich die Suche von Schätzern für $\delta_k^*(x)$ auf die Suche geeigneter Schätzer für $\beta^{*(k)}$, $k = 1, \ldots, K-1$. Ein oft verwendeter Ansatz zur Schätzung dieser Parameter ist die *Maximum-Likelihood-Methode*. Zur Durchführung benötigen wir zunächst eine Darstellung der gemeinsamen Dichte $f(x_1, y_1, \ldots, x_n, y_n)$ von (X_i, Y_i), $i = 1, \ldots, n$ in Abhängigkeit von den zu schätzenden Parametern und den Beobachtungen. Ist $f_{X,Y}$ die Dichte von (X_1, Y_1) bzgl. eines geeigneten dominierenden Maßes (vgl. Bedingungen in Lemma 3.4) und $g(x)$ die entsprechende Marginaldichte von X_1, so folgt mit dem Satz von Bayes:

$$f_{X,Y}(x, y) = \mathbb{P}(Y = y|X = x) \cdot g(x)$$

Da (X_i, Y_i) i. i. d. sind, folgt:

$$\begin{aligned}\log f(x_1, y_1, \ldots, x_n, y_n) &= \sum_{i=1}^{n}\log f_{X,Y}(x_i, y_i)\\ &= \sum_{i=1}^{n}\log\mathbb{P}(Y = y_i|X = x_i) + \sum_{i=1}^{n}\log g(x_i)\end{aligned}$$

Aufgrund der Darstellungen in Lemma 4.9 gilt:

$$\begin{aligned}\log\mathbb{P}(Y = y|X = x) &= \sum_{k=1}^{K-1}\mathbb{1}_{\{y=k\}}\left\{x^T\beta^{*(k)} - \log\left(1 + \sum_{j=1}^{K-1}\exp(x^T\beta^{*(j)})\right)\right\}\\ &\quad -\mathbb{1}_{\{y=K\}}\log\left(1 + \sum_{k=1}^{K-1}\exp(x^T\beta^{*(k)})\right)\\ &= \left\{\sum_{k=1}^{K-1}\mathbb{1}_{\{y=k\}}\cdot x^T\beta^{*(k)}\right\} - \log\left(1 + \sum_{k=1}^{K-1}\exp(x^T\beta^{*(k)})\right)\end{aligned}$$

Setzen wir die Beobachtungen X_i, Y_i ein und ersetzen wir $\beta^{*(k)}$ durch beliebige $\beta^{(k)}$, so erhalten wir:

$$\begin{aligned}&\log f(X_1, Y_1, \ldots, X_n, Y_n)\\ &= \sum_{i=1}^{n}\left[\left\{\sum_{k=0}^{K-2} \mathbb{1}_{\{Y_i=k\}} \cdot X_i^T \beta^{(k)}\right\} - \log\left(1 + \sum_{k=0}^{K-2} \exp(X_i^T \beta^{(k)})\right)\right] + \sum_{i=1}^{n} \log g(X_i)\end{aligned}$$

Bei der Maximum-Likelihood-Methode wählen wir die Parameter $\beta^{(1)}, \ldots, \beta^{(K-1)} \in \mathbb{R}^d$ aus, für welche die Beobachtungen X_i, Y_i am wahrscheinlichsten gewesen wären. Die ‚Wahrscheinlichkeit' messen wir hierbei mittels der Dichte $f(X_1, Y_1, \ldots, X_n, Y_n)$. Durch die monotone Transformation $\frac{1}{n} \log f(X_1, Y_1, \ldots, X_n, Y_n)$ und durch Entfernen des additiven Terms $\sum_{i=1}^{n} \log g(X_i)$ (welcher nicht von den Parametern abhängt) werden die lokalen Maxima von $\frac{1}{n} \log f(X_1, Y_1, \ldots, X_n, Y_n)$ in den Parametern $\beta^{(1)}, \ldots, \beta^{(K-1)}$ nicht verändert. Wir definieren also

$$L_n(\theta) = \frac{1}{n} \sum_{i=1}^{n}\left[\left\{\sum_{k=1}^{K-1} \mathbb{1}_{\{Y_i=k\}} \cdot X_i^T \beta^{(k)}\right\} - \log\left(1 + \sum_{k=1}^{K-1} \exp(X_i^T \beta^{(k)})\right)\right], \tag{4.8}$$

die sogenannte *Likelihood-Funktion,* und maximieren diese Größe in

$$\theta = ((\beta^{(1)})^T, \ldots, (\beta^{(K-1)})^T)^T \in \mathbb{R}^{(K-1)d}.$$

Definition 4.10 (Klassifizierer: Logistische Regression)
Sei

$$\hat{\theta} = ((\hat{\beta}^{(1)})^T, \ldots, (\hat{\beta}^{(K-1)})^T)^T \in \operatorname*{arg\,max}_{\theta \in \mathbb{R}^{(K-1)d}} L_n(\theta).$$

Sei

$$\hat{\delta}_k^{LR}(x) = \begin{cases} x^T \hat{\beta}^{(k)}, & k = 1, \ldots, K-1, \\ 0, & k = K \end{cases}.$$

Der *Logistische-Regression-Klassifizierer* ist $\hat{f}_n^{LR}(x) = \arg\max_{k \in \{1,\ldots,K\}} \hat{\delta}_k^{LR}(x)$. ♦

Bemerkungen

- Da $(z_1, \ldots, z_{K-1}) \mapsto -\log(1 + \sum_{k=1}^{K-1} e^{z_k})$ eine konkave Funktion ist, ist $L(\theta)$ konkav in θ und besitzt daher ein globales Maximum. Es gibt allerdings keine geschlossene Darstellung des Maximierers. Da $\theta \mapsto L(\theta)$ differenzierbar ist, kann ein globaler Maximierer zum Beispiel mit einem Newton-Verfahren ermittelt werden.
- Mit Sätzen über Maximum-Likelihood-Schätzer kann man zeigen, dass das Excess Bayes Risk von $\hat{f}_n^{LR}$ Konvergenzrate $\left(\frac{d}{n}\right)^{1/2}$ besitzt, vgl. das theoretische Resultat in Abschn. 4.2.1. Nur falls $d \ll n$ gilt, kann daher eine gute Qualität von $\hat{f}_n^{LR}$ erwartet werden.

- Aufgrund der Form der Diskriminantenfunktionen liefert die logistische Regression in der oben beschriebenen Variante wie die lineare Diskriminantenanalyse lineare Entscheidungsränder.

Für den Spezialfall zweier Klassen existiert eine einfachere Form von $L_n(\theta)$:

Bemerkung 4.11 (Alternative Schreibweise im Fall $K = 2$) Mit den geänderten Klassenbezeichnungen $\mathcal{Y} = \{-1, 1\}$ und $\theta = \beta^{(1)}$ kann $L_n(\theta)$ im Fall $K = 2$ wie folgt umgeschrieben werden (beachte: $Y_i \in \{-1, 1\}$ und daher $\mathbb{1}_{\{Y_i=1\}} = \frac{1}{2}(1 + Y_i)$):

$$L_n(\theta) = \sum_{i=1}^{n} \left[\frac{1}{2}(1 + Y_i) \cdot X_i^T \beta^{(0)} - \log(1 + \exp(X_i^T \beta^{(0)})) \right] \tag{4.9}$$

Damit kann die verwendete Maximum-Likelihood-Methode auch als Anwendung des Ansatzes aus Bemerkung 3.8 mit

$$\tilde{\mathcal{F}} = \{\delta : \mathcal{X} \to \mathbb{R} | \ Es\ gibt\ \beta^{(0)} \in \mathbb{R}^d\ so\ dass\ für\ alle\ x \in \mathcal{X} : \delta(x) = x^T \beta^{(0)}\}$$

und

$$\tilde{L}(y, s) := \frac{1}{2}(1 + y) \cdot s - \log(1 + \exp(s))$$

interpretiert werden, wobei $\tilde{L}$ den sogenannten *Kreuzentropieverlust* bezeichnet. Das bedeutet, $\hat{f}_n^{LR}$ lässt sich im Sinne von Abschn. 3.3 als näherungsweiser Minimierer eines empirischen Risikos interpretieren.

Beispiel 4.12 Wir betrachten die Modelle aus Beispiel 4.4 bzw. Beispiel 4.5. Anstelle der Trainingsdaten (X_i, Y_i) nutzen wir die erweiterten Trainingsdaten $\tilde{X}_i := (1, X_i^T)^T \in \mathbb{R}^3$ und entsprechend $\hat{\delta}_k(x) = (1, x)^T \hat{\beta}^{(k)}$ $(k = 1, \ldots, K - 1)$, um auch vom Ursprung verschobene Entscheidungsränder zulassen zu können. Formal ändern wir damit die Modellannahme, auf deren Basis wir logistische Regression durchführen, vgl. Gl. (4.10).

In Abb. 4.4 sind die zu $\hat{f}_n^{LR}$ gehörigen Entscheidungsregionen eingetragen. Durch Vergleich der Abb. 4.4 mit den Resultaten der LDA in Abb. 4.1 und 4.2 kann man sehen, dass die logistische Regression sehr stark versucht, Fehlklassifizierungen der Trainingsdaten zu vermeiden. Der Grund ist, dass sie im Gegensatz zur LDA die Beobachtungen X_i nicht als Realisierung einer Verteilung sieht und daher nicht entsprechenden Zwangsbedingungen durch die Form der Verteilung unterworfen ist. Die logistische Regression liefert weiterhin lineare Entscheidungsränder und ist daher in der bisher vorgestellten Form nicht in der Lage, kompliziertere Entscheidungsregionen korrekt zu schätzen (vgl. Abb. 4.4c).

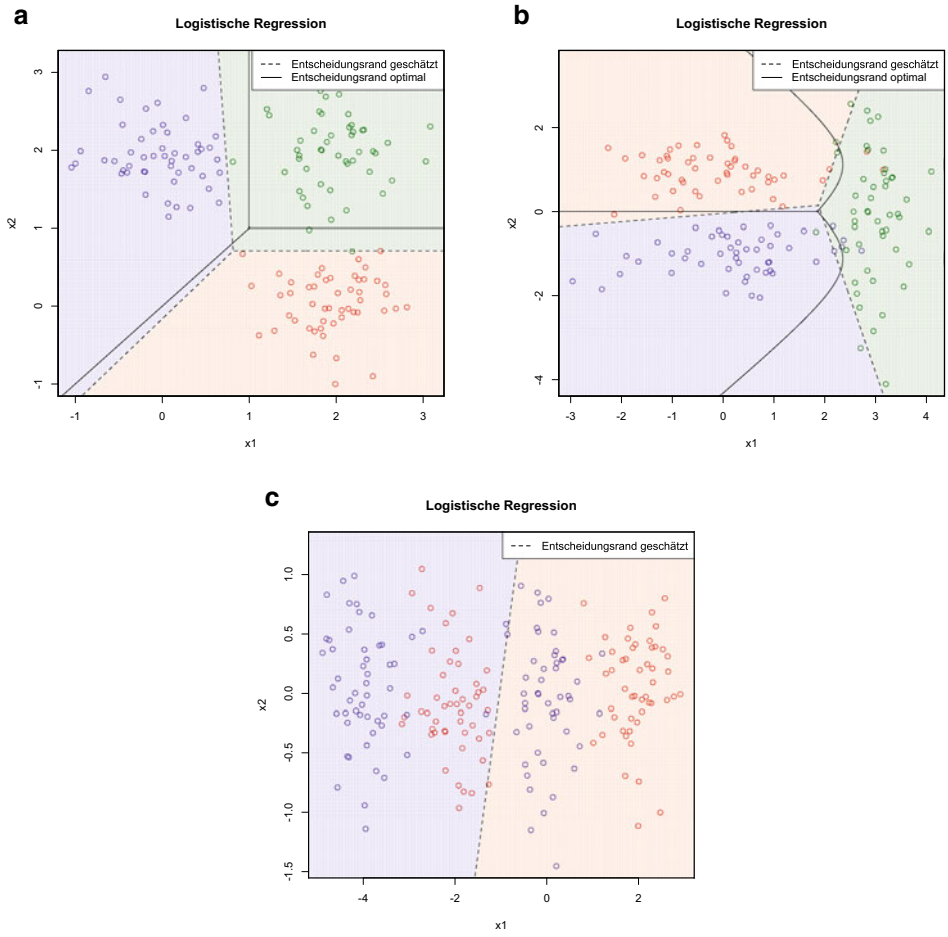


Abb. 4.4 Darstellung des Klassifiziers $\hat{f}_n^{LR}$ ermittelt durch logistische Regression. **a** angewandt auf die Daten aus Beispiel 4.4(a); **b** auf die Daten aus Beispiel 4.4(b); **c** auf die Daten aus Beispiel 4.5. Die farblichen Schattierungen entsprechen den geschätzten Entscheidungsregionen der jeweiligen Klassen

Wir haben die Methode der logistischen Regression mit dem Ziel hergeleitet, lineare optimale Entscheidungsregionen und Entscheidungsränder zu erhalten. Im Fall $K = 2$ beispielsweise gilt wie eingangs motiviert

$$\Omega_1^* = \{x \in \mathcal{X} : \delta_1^*(x) \geq \delta_2^*(x)\} = \{x \in \mathcal{X} : x^T \beta^{(1)} \geq 0\},$$

d. h., die Grenze zwischen den Entscheidungsregionen Ω_1^* und Ω_2^* *muss* eine lineare Funktion sein. Im Unterschied zur linearen Diskriminantenanalyse wird jedoch keine Verteilungsannahme an X gegeben Y gestellt, wodurch logistische Regression zum Beispiel bei

groben Verstößen gegen die Normalverteilungsannahme (zum Beispiel durch einige Ausreißer) wesentlich stabiler arbeitet. Falls die Modellannahme der LDA erfüllt ist, besitzen die durch LDA geschätzten Entscheidungsränder eine geringere Varianz als die logistische Regression. Auch in diesem Falle kann man aber zeigen, dass die geschätzten Entscheidungsränder, erhalten durch logistische Regression, nicht wesentlich schlechter als die der LDA sein können. In diesem Sinne ist die logistische Regression oft der LDA vorzuziehen.

Für viele Anwendungen ist die Annahme linearer Entscheidungsränder, d. h., dass sich die Klassen mit einer linearen Funktion trennen lassen, jedoch zu stark. Es gibt aber einen einfachen mathematischen Trick, um kompliziertere Formen von Entscheidungsregionen zu erreichen: Ersetze die ursprünglich erhaltenen Beobachtungen X_i durch $\tilde{X}_i := h(X_i)$, wobei $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ eine geeignete Einbettung in einen höherdimensionalen Raum mit $\tilde{d} > d$ ist (vgl. Beispiel 4.14). Wende dann die logistische Regression Definition 4.10 auf die transformierten Beobachtungen $(\tilde{X}_i, Y_i)$, $i = 1, \ldots, n$ an. Formal entspricht dies der Änderung unserer Modellannahme 4.8 zu

$$\log\left(\frac{\mathbb{P}(Y = k|X = x)}{\mathbb{P}(Y = K|X = x)}\right) = h(x)^T \tilde{\beta}^{*(k)}, \quad x \in \mathbb{R}^d, \tag{4.10}$$

mit $\tilde{\beta}^{*(k)} \in \mathbb{R}^{\tilde{d}}$, $k = 1, \ldots, K-1$. Wir erhalten folgende Definition:

Definition 4.13 (Klassifizierer: Logistische Regression nichtlinear)
Sei $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ eine Abbildung und $\tilde{X}_i = h(X_i)$. Sei $\tilde{L}_n(\theta)$ die Funktion, welche aus $L_n(\theta)$ in Gl. (4.8) durch Ersetzen von $\tilde{X}_i$ durch X_i hervorgeht. Sei

$$\tilde{\theta} = ((\tilde{\beta}^{(1)})^T, \ldots, (\tilde{\beta}^{(K-1)})^T)^T \in \underset{\theta \in \mathbb{R}^{(K-1)d}}{\arg\max}\, \tilde{L}_n(\theta).$$

Setze

$$\hat{\delta}_k^{LR,nichtlin}(x) = \begin{cases} h(x)^T \tilde{\beta}^{(k)}, & k = 1, \ldots, K-1, \\ 0, & k = K \end{cases}.$$

Der *Logistische-Regression-Klassifizierer* ist

$$\hat{f}_n^{LR,nichtlin}(x) = \underset{k \in \{1,\ldots,K\}}{\arg\max}\, \hat{\delta}_k^{LR,nichtlin}(x). \qquad \blacklozenge$$

Da eine sinnvolle Auswahl der Funktion h ohne Kenntnis der wahren Entscheidungsränder nur sehr beschränkt möglich ist, muss man in der Praxis einen wesentlich größeren Raum $\mathbb{R}^{\tilde{d}}$ nutzen als eigentlich nötig.

Beispiel 4.14 Wir betrachten die gleichen Modelle wie in Beispiel 4.12 mit

$$h(x_1, x_2) = (1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3)^T \tag{4.11}$$

und den erweiterten Beobachtungen $\tilde{X}_i = h(X_i)$. Die Entscheidungsränder entstehen demzufolge durch Kombination kubischer Gleichungen der Form

$$\alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_1^2 + \alpha_4 x_2^2 + \alpha_5 x_1^3 + \alpha_6 x_2^3 = 0.$$

In Abb. 4.5 sind die zu $\hat{f}_n^{LR,nichtlin}$ gehörigen Entscheidungsregionen eingetragen (verkleinert im Gegensatz zu Abb. 4.4 für einen besseren Überblick der geschätzten Entscheidungsränder). Man erkennt, dass $\hat{f}_n^{LR,nichtlin}$ wesentlich besser als $\hat{f}_n^{LR}$ in der Lage ist,

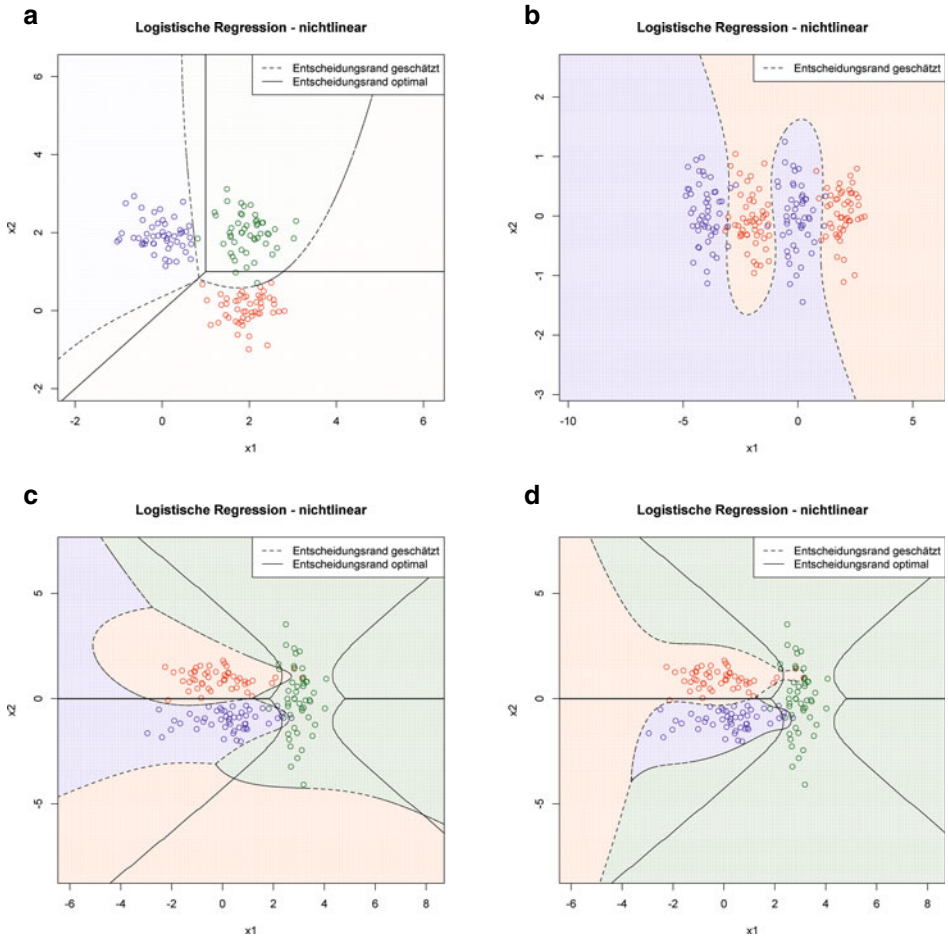


Abb. 4.5 Darstellung des Klassifiziers $\hat{f}_n^{LR,nichtlin}$ ermittelt durch logistische Regression angewandt auf die modifizierten Trainingsdaten aus Beispiel 4.4(a) (vgl. a), aus Beispiel 4.5 (vgl. b) mit h aus (4.11) und die Daten aus Beispiel 4.4(b) für zwei h gewählt durch Gl. (4.11) (vgl. c) und Gl. (4.12) (vgl. d). Die farblichen Schattierungen entsprechen den geschätzten Entscheidungsregionen der jeweiligen Klassen

die gegebenen Trainingsdaten zu trennen, allerdings machen sich an einigen Stellen bereits Überanpassungen durch starke Abweichungen vom optimalen Entscheidungsrand bemerkbar. Extrem wird dieses Verhalten bei Nutzung der Funktion

$$h(x_1, x_2) = (1, x_1, x_2, x_1^2, x_2^2, \ldots, x_1^7, x_2^7)^T, \tag{4.12}$$

vgl. Abb. 4.5d.

Analog zu Lasso-Schätzern (vgl. Definition 2.37) kann auch eine mittels $\|\cdot\|_1$-Norm bestrafte logistische Regression definiert werden. Dies ist sinnvoll, wenn man erwartet, dass nur wenige Komponenten von X_0 (bzw. $\tilde{X}_0$, falls die nichtlineare Regression genutzt wird) wirklich relevant für die Beschreibung von Y_0 sind.

Definition 4.15 (Klassifizierer: Logistische Regression mit $\|\cdot\|_1$-Regularisierung) Sei $\lambda \geq 0$. Sei $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ eine Abbildung und $\tilde{X}_i = h(X_i)$. Sei

$$\hat{\theta} = ((\hat{\beta}^{(0)})^T, \ldots, (\hat{\beta}^{(K-2)})^T)^T \in \operatorname*{arg\,max}_{\theta \in \mathbb{R}^{(K-1)d}} \left\{ \tilde{L}_n(\theta) - \lambda J(\theta) \right\}, \qquad J(\theta) = \sum_{k=1}^{K-1} \|\beta^{(k)}\|_1.$$

Setze nun $\hat{\delta}_j^{LR,nichtlin,L_1}$ und $\hat{f}_n^{LR,nichtlin,L_1}$ wie in Definition 4.13. ♦

Bemerkung Auch $\theta \mapsto L_n(\theta) - \lambda J(\theta)$ ist konkav, besitzt also ein globales Maximum. Für die Maximierung dieser gibt es stabile numerische Lösungsverfahren im Stile von Satz 2.42. In der Praxis wählt man λ zum Beispiel durch Cross Validation.

Beispiel 4.16 Wir betrachten dieselbe Situation wie in Beispiel 4.14 mit h gewählt durch Gl. (4.12) und bestimmen den Klassifizierer $\hat{f}_n^{LR,nichtlin,L_1}$ aus Definition 4.15 mit $\lambda = 0{,}01$.

In Abb. 4.6 sind die zu $\hat{f}_n^{LR,nichtlin,L_1}$ gehörigen Entscheidungsregionen eingetragen. Im Gegensatz zu $\hat{f}_n^{LR,nichtlin}$ (vgl. Abb. 4.5) werden nun die zur Beschreibung der Entscheidungsränder unwichtigen Komponenten von $\tilde{X}_i = h(X_i)$ erkannt und die entsprechenden Komponenten von $\beta^{(k)}$, $k = 1, \ldots, K-1$ auf 0 gesetzt.

Die Verwendung der logistischen Regression aus Definition 4.15 verlangt nach einer möglichst zum Problem passenden Wahl von h. Auch wenn die $\|\cdot\|_1$-Penalisierung grundsätzlich die Anwendung auf hochdimensionale Trainingsdaten ermöglicht, so ist für zufriedenstellende Resultate eine möglichst reichhaltige Klasse von Funktionen $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ zu wählen. Möchte man zum Beispiel quadratische Polynome zur Beschreibung der Entscheidungsränder zulassen, so muss $h(x)$ die Komponenten x_j^2 und $x_i x_j$ $(i, j = 1, \ldots, d)$ beinhalten, so dass $\tilde{d}$ von der Größenordnung d^2 ist. Der benötigte *Speicherplatz* für die erweiterten Trainingsdaten $\tilde{X}_i$ ist dann oft zu groß, so dass man eine sinnvolle Auswahl der Komponenten h treffen, d. h. Vorwissen in die Konstruktion von h einfließen lassen muss.

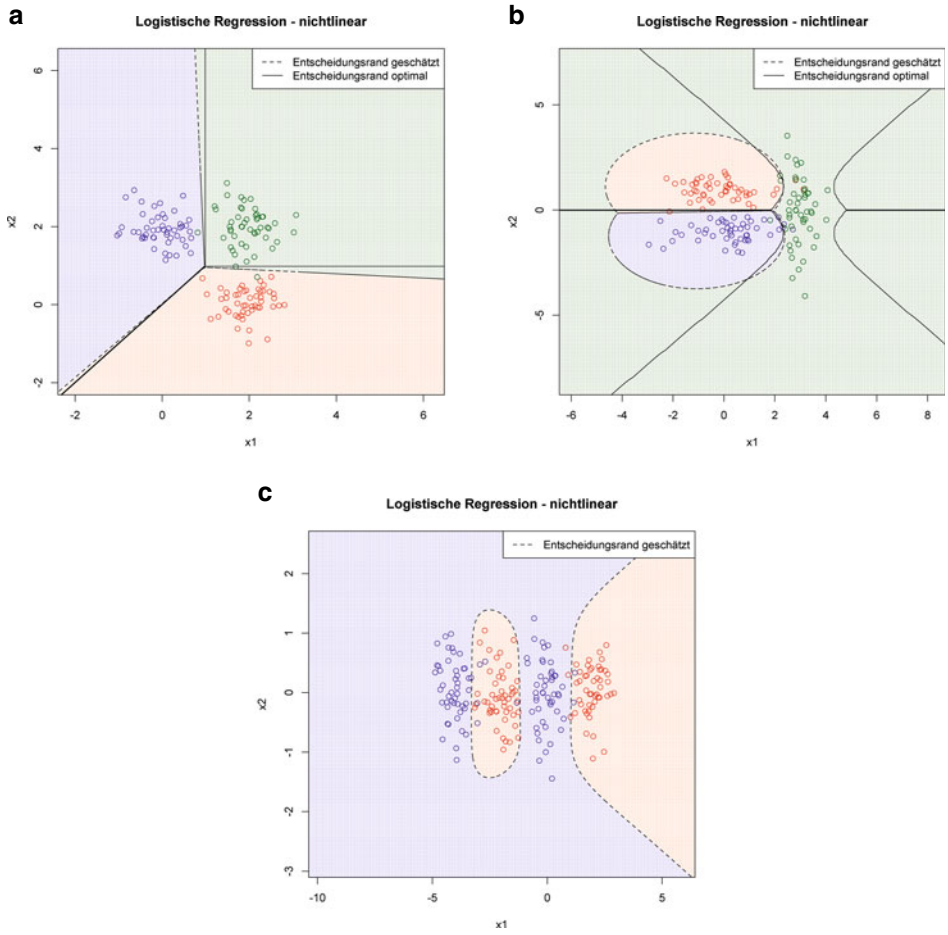


Abb. 4.6 Darstellung des Klassifiziers $\hat{f}_n^{LR,nichtlin,L_1}$ ermittelt durch logistische Regression mit $\lambda = 0{,}01$ und h gewählt durch Gl. (4.12) angewandt auf die Daten aus Beispiel 4.4(a) (vgl. a), (b) (vgl. b) und Beispiel 4.5 (vgl. c). Die farblichen Schattierungen entsprechen den geschätzten Entscheidungsregionen der jeweiligen Klassen

In Abschn. 4.4 stellen wir ein weiteres Verfahren zur Ermittlung von Klassifizierern vor, bei welchem eine Erweiterung $\tilde{X}_i = h(X_i)$ der ursprünglichen Trainingsdaten ohne die Nutzung weiteren Speicherplatzes möglich ist und sogar keine explizite Wahl der Funktionen h getroffen werden muss.

4.2.1 Theoretische Resultate

Wir betrachten den Spezialfall von $K = 2$ Klassen. Es sei $\hat{\theta} = \hat{\beta}^{(1)}$ der Minimierer von $L_n(\theta)$. Für das Excess Bayes Risk von $\hat{f}_n^{LR}$ gilt dann folgendes Lemma:

Lemma 4.17 Existiert $\Sigma = \mathbb{E}[XX^T]$, so gilt

$$R(\hat{f}_n^{LR}) - R(f^*) \leq \frac{1}{2} \left\| \Sigma^{1/2}(\hat{\beta}^{(1)} - \beta^{*(1)}) \right\|_2 .$$

Beweis Definiere $\eta(x) := \mathbb{P}(Y = 1|X = x) = S(x^T\beta^{*(1)})$ und $\hat{d}_1(x) = S(x^T\hat{\beta}^{(1)})$, $\hat{d}_2(x) = S(0) = \frac{1}{2}$ mit $S : \mathbb{R} \to \mathbb{R}$, $S(x) := \frac{e^x}{1+e^x}$. Da S monoton wachsend ist, gilt

$$\hat{f}_n^{LR}(x) = \arg\max_{k=1,2} \hat{d}_k(x) = \begin{cases} 1, & \hat{d}_1(x) \geq \frac{1}{2}, \\ 2, & \hat{d}_1(x) < \frac{1}{2}. \end{cases}$$

Die Funktion S ist Lipschitz-stetig mit Lipschitz-Konstante $\frac{1}{4}$. Sei $T_n := (X_i, Y_i)_{i=1,\ldots,n}$. Mit Lemma 3.5(i) folgt:

$$\begin{aligned} R(\hat{f}_n^{LR}) - R(f^*) &\leq 2\mathbb{E}[|S(X^T\beta^{*(1)}) - S(X^T\hat{\beta}^{(1)})|\,|T_n] \\ &\leq 2\mathbb{E}[|X^T(\beta^{*(1)} - \hat{\beta}^{(1)})|\,|T_n] \leq \frac{1}{2}\mathbb{E}\left[(X^T(\beta^{*(1)} - \hat{\beta}^{(1)}))^2|T_n\right]^{1/2} \\ &= \frac{1}{2} \left\| \Sigma^{1/2}(\hat{\beta}^{(1)} - \beta^{*(1)}) \right\|_2 \end{aligned}$$

Lemma 4.17 bedeutet, dass sich die Konvergenzrate der Parameter $\hat{\beta}^{(1)} - \beta^{*(1)}$ in oben dargestellter Weise direkt auf die Konvergenzrate des Excess Bayes Risks von $\hat{f}_n^{LR}$ überträgt. Mit Hilfe der Maximum-Likelihood-Theorie (vgl. zum Beispiel [20]) kann man zeigen, dass für festes $d \in \mathbb{N}$ gilt:

$$n\|I(\beta^{*(1)})^{1/2}(\hat{\beta}^{(1)} - \beta^{*(1)})\|_2^2 \approx \|Z\|_2^2 = \sum_{j=1}^{d} Z_j^2,$$

wobei $Z \sim N(0, I_{d\times d})$ gilt und $I(\beta^{*(1)})$ die sogenannte Fisher-Informationsmatrix des Problems ist. Da sich der Ausdruck $Z := \frac{1}{d}\sum_{j=1}^{d} Z_j^2$ weitestgehend unabhängig von d entwickelt, gilt näherungsweise

$$\|I(\beta^{*(1)})^{1/2}(\hat{\beta}^{(1)} - \beta^{*(1)})\|_2^2 \approx \frac{d}{n} \cdot Z,$$

und wir erwarten gemäß Lemma 4.17, dass approximativ

$$\begin{aligned} R(\hat{f}_n^{LR}) - R(f^*) &\leq \lambda_{max}(\Sigma^{1/2}I(\beta^{*(1)})^{-1/2}) \cdot \|I(\beta^{*(1)})^{1/2}(\hat{\beta}^{(1)} - \beta^{*(1)})\|_2 \\ &\leq \lambda_{max}(\Sigma^{1/2}I(\beta^{*(1)})^{-1/2})\sqrt{Z}\left(\frac{d}{n}\right)^{1/2} \end{aligned}$$

gilt (wir nutzen oben die Notation $\lambda_{max}(A)$ für den maximalen Eigenwert einer Matrix A). Solange der maximale Eigenwert von $\Sigma^{1/2} I(\beta^{*(1)})$ in d beschränkt bleibt, erwarten wir eine Konvergenzrate von $\left(\frac{d}{n}\right)^{1/2}$ für das Excess Bayes Risk von $\hat{f}_n^{LR}$.

Für $\hat{f}_n^{LR,nichtlin,L_1}$ können ähnliche Konvergenzraten des Excess Bayes Risk wie für den Lasso-Schätzer in Abschn. 2.4 gezeigt werden.

4.3 Separierende Hyperebenen

In diesem Abschnitt konzentrieren wir uns auf den Fall $K = 2$, d. h. der Klassifizierung in zwei Klassen (sogenannte *binäre Klassifikation*). Dies geschieht vor allem aus didaktischen Gründen; die hier vorgestellten Algorithmen können prinzipiell auch für mehr als zwei Klassen formuliert werden, die Notation wird dann aber ausladender. Es gibt auch elementare Möglichkeiten, Mehr-Klassen-Probleme auf Klassifizierer mit Zwei-Klassen-Problemen zurückzuführen, siehe Abschn. 3.4.

Anstelle von $\mathcal{Y} = \{1, 2\}$ bezeichnen wir die Klassen nun mit $\mathcal{Y} = \{-1, +1\}$.

4.3.1 Motivation

Im Gegensatz zu den bisher in diesem Kapitel vorgestellten Methoden leiten wir die nächste Methode nicht basierend auf einer Modellannahme an die Verteilung $\mathbb{P}^{(X_1,Y_1)}$ her, sondern ermitteln einen Klassifizierer auf Basis der grafischen Anschauung der Trainingsdaten. Die dazu passende Theorie und die entsprechende zugrunde liegende Modellannahme werden danach in Abschn. 4.6 hergeleitet.

Als Motivation für diese Technik dient der folgende naive Klassifizierer für zwei Klassen $\mathcal{Y} = \{-1, +1\}$ mittels linearer Regression:

Beispiel 4.18 (Klassifizierung mit linearer Regression) Gegeben Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$ mit $Y_i \in \mathcal{Y} = \{-1, +1\}$, führe eine lineare Regression auf Basis des Modells

$$Y_i = \beta_0 + X_i^T \beta + \varepsilon_i, \quad i = 1, \ldots, n, \tag{4.13}$$

mit Parametern $\beta_0 \in \mathbb{R}$, $\beta \in \mathbb{R}^d$ durch (vgl. Vorwort). Wir erhalten Schätzer $\hat{\beta}_0$, $\hat{\beta}$ sowie die Hyperebene

$$\hat{\delta}_n(x) = \hat{\beta}_0 + x^T \hat{\beta}.$$

Als Klassifizierer verwenden wir

$$\hat{f}_n^{linreg}(x) := \text{sign}(\hat{\delta}_n(x)), \qquad \text{sign}(y) := \begin{cases} 1, & y \geq 0, \\ -1, & y < 0. \end{cases}$$

In Abb. 4.7 ist dieses Verfahren anschaulich dargestellt: Die Klassen Y_i werden als reelle Zahlen betrachtet und mittels einer Hyperebene an die Daten X_i angepasst. Für ein $x \in \mathcal{X}$ wird überprüft, ob die angepasste Hyperebene bei x einen positiven oder negativen Wert besitzt und entsprechend die Klasse $+1$ oder -1 zugeordnet.

Werden die erhaltenen Entscheidungsränder auf anschaulicher Basis bewertet, so hat der Ansatz aus dem obigen Beispiel 4.18 zwei nicht zufriedenstellende Eigenschaften:

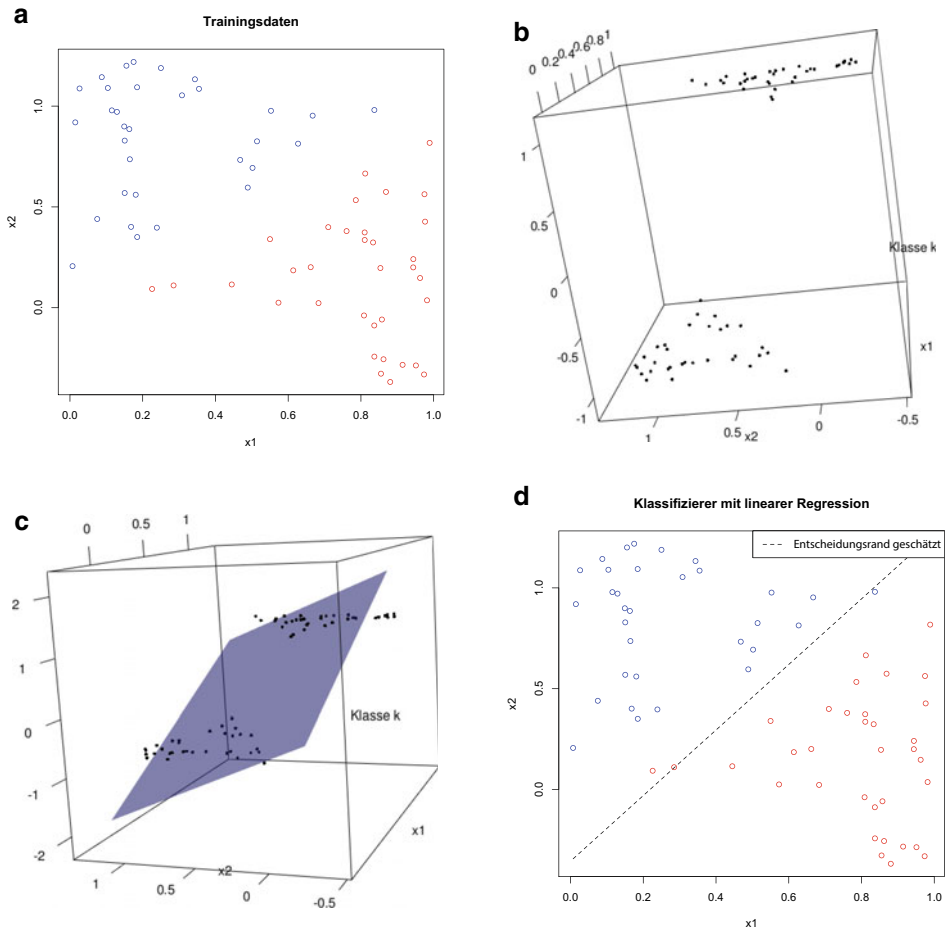


Abb. 4.7 Darstellung des Ansatzes aus Beispiel 4.18. **a** Trainingsdaten. **b** Interpretation der Trainingsdaten als Daten des linearen Modells (4.13), **c** Darstellung der Regressionsebene $\hat{\delta}_n(x)$, **d** Entscheidungsrand des Klassifizierers $\hat{f}_n^{linreg}$

(N1) Selbst wenn die Trainingsdaten X_i mit einer Hyperebene vollständig trennbar wären (vgl. Abb. 4.8), so liefert $\hat{f}_n^{linreg}$ nicht notwendig eine solche Lösung.
(N2) Der Ansatz funktioniert nicht gut, wenn zwei Klassen *kollinear* zueinander sind und daher nicht mit einer Hyperebene getrennt werden können, wie zum Beispiel in Abb. 4.3.

Im Folgenden wollen wir den „naiven" Ansatz aus Beispiel 4.18 so modifizieren, dass die Eigenschaften (N1) und (N2) korrigiert werden. Wir geben hier eine kurze Diskussion, warum die Korrektur von (N1) wünschenswert ist:

Bemerkung 4.19 (Diskussion von [N1]) Bei der grafischen Bewertung der Trainingsdaten geht man davon aus, dass neue Realisierungen in der Nähe der bereits beobachteten Daten liegen. Kann ein Klassifizierer die beiden Klassen also sehr deutlich mit einer Hyperebene trennen, so werden nur sehr wenige neue Beobachtungen (nämlich nur diejenigen, die entgegen den Trainingsdaten nun auf der falschen Seite der Hyperebene liegen) falsch klassifiziert. Außerdem sollten Beobachtungen, welche sehr weit von Beobachtungen anderer Klassen entfernt sind, aus grafischer Sicht keinen starken Einfluss auf den geschätzten Entscheidungsrand ausüben. Umgekehrt: Der Entscheidungsrand sollte vor allem durch die Beobachtungen bestimmt sein, welche sehr nah an Beobachtungen anderer Klassen liegen. Diese Bedingungen ergeben sich aus einer rein grafischen Betrachtung der Trainingsdaten. In Beispiel 4.18 jedoch, wo die Schätzung des Entscheidungsrandes auf der Modellannahme aus Gl. (4.13) beruht, tragen naturgemäß alle Beobachtungen in gleichem Maße zum geschätzten Entscheidungsrand bei – oberste Priorität ist schließlich keine grafische Trennung, sondern eine möglichst genaue Anpassung an das Modell.

4.3.2 Die optimale separierende Hyperebene

In diesem Abschnitt gehen wir davon aus, dass die zu den beiden verschiedenen Klassen gehörigen Trainingsdaten $\{X_i : Y_i = -1\}$ und $\{X_i : Y_i = +1\}$ durch eine Hyperebene trennbar, d. h. *linear separierbar* sind. Solche Hyperebenen nennen wir *separierende Hyperebenen*. In diesem Kontext wollen wir einen Klassifizierer herleiten, welcher diese Mengen mit einer möglichst ‚guten' Hyperebene vollständig trennt. Unter einer Hyperebene verstehen wir dabei gemäß folgender Definition:

Definition 4.20
Eine *(affine) Hyperebene* in $\mathbb{R}^d$ mit Normalenvektor $\beta \in \mathbb{R}^d$ und Stützvektor $x_0 \in \mathbb{R}^d$ ist eine Menge

$$\delta := \{x \in \mathbb{R}^d : \beta^T (x - x_0) = 0\} \overset{\beta_0 := -\beta^T x_0}{=} \{x \in \mathbb{R}^d : \beta_0 + \beta^T x = 0\} \subset \mathbb{R}^d. \qquad \blacklozenge$$

Wir geben einige elementare Eigenschaften von Hyperebenen wieder.

Lemma 4.21 Sei δ eine Hyperebene in $\mathbb{R}^d$ mit Normalenvektor β und Stützvektor x_0.

(i) Für alle $x_1, x_2 \in \delta$ gilt $\beta^T(x_1 - x_2) = 0$. Auch der Vektor $\beta^* = \frac{\beta}{\|\beta\|_2}$ ist normal zu δ bzgl. des euklidischen Skalarprodukts.
(ii) Die Distanz (mit Vorzeichen) für jedes $x \in \mathbb{R}^d$ zu δ ist gegeben durch

$$(\beta^*)^T(x - x_0) = \frac{1}{\|\beta\|_2}(\beta^T x + \beta_0).$$

Wie in Beispiel 4.18 soll der Klassifizierer dann die Gestalt

$$\hat{f}_n(x) = \text{sign}(\hat{\delta}_n(x)), \qquad \hat{\delta}_n(x) = \beta^T x + \beta_0$$

mit geeigneten β, β_0 haben. Ist $\hat{\delta}_n(x)$ eine separierende Hyperebene, so gilt

$$Y_i = 1 \quad \Leftrightarrow \quad \hat{f}_n(X_i) = 1 \quad \Leftrightarrow \quad \hat{\delta}_n(X_i) = \beta^T X_i + \beta_0 > 0. \tag{4.14}$$

In Abb. 4.8 ist eine Menge von Trainingsdaten mit mehreren verschiedenen trennenden Hyperebenen zu sehen. Durch die Forderung in Gl. (4.14) ist $\hat{f}_n(x)$ also noch nicht eindeutig bestimmt. Zur genaueren Charakterisierung von β, β_0 nutzt man folgenden grafischen Ansatz: Wir suchen diejenige Hyperebene, welche die beiden Punktmengen so trennt, dass auf beiden Seiten der Hyperebene noch möglichst viel freier Platz zu den Punktmengen ist. Im Fall $d = 2$ entspricht dies anschaulich der Suche einer möglichst breiten ‚Straße', die zwischen den Punktmengen verläuft. Die Mitte der Straße ist die gesuchte trennende Hyperebene (vgl. Abb. 4.8b).

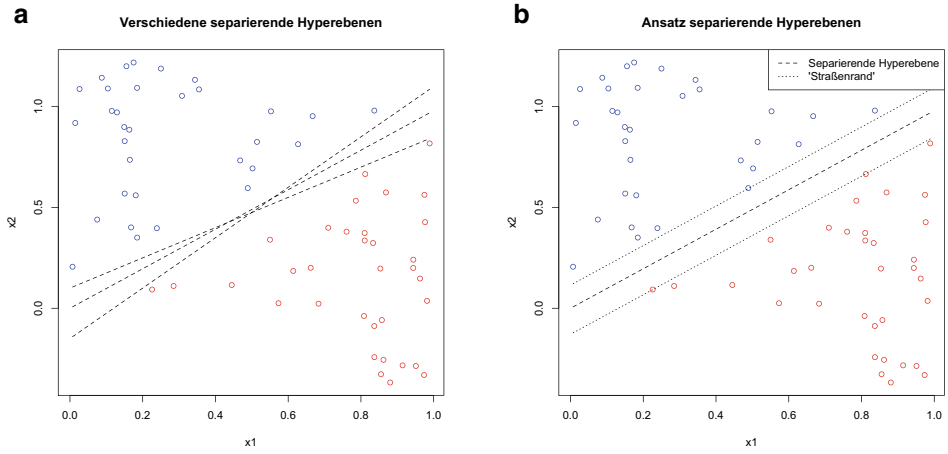


Abb. 4.8 **a** Vollständig mit einer Hyperebene trennbare Trainingsdaten mit mehreren verschiedenen trennenden Hyperebenen. **b** Ansatz der separierenden Hyperebenen

Die Idee hinter diesem Ansatz folgt der Diskussion in Bemerkung 4.19: Weitere Punkte der Klasse ‚-1' liegen in der Nähe der bereits beobachteten Punkte mit Klasse ‚-1'. Je größer der Abstand aller Punkte $\{X_i : Y_i = -1\}$ zur trennenden Hyperebene, desto wahrscheinlicher ist es, dass auch die neuen Punkte auf der richtigen Seite der Hyperebene liegen, also richtig klassifiziert werden.

Wir formalisieren nun den obigen Ansatz. Beachte, dass gilt:

- Falls $Y_i = +1$ korrekt klassifiziert, so gilt $X_i^T \beta + \beta_0 > 0$.
- Falls $Y_i = -1$ korrekt klassifiziert, so gilt $X_i^T \beta + \beta_0 < 0$.

In beiden Fällen ist also bei korrekter Klassifikation $Y_i \cdot (X_i^T \beta + \beta_0) \geq 0$.

Herleitung Schritt 1 Für jedes $i \in \{1, \ldots, n\}$ sei $m_i := Y_i \cdot \frac{1}{\|\beta\|_2}(X_i^T \beta + \beta_0)$ der Abstand von X_i zur Hyperebene $\hat{\delta}_n(x) = x^T \beta + \beta_0$. Sei

$$M := \inf_{i=1,\ldots,n} m_i$$

der minimale Abstand zur Hyperebene. Hierbei gilt: Ist $M < 0$, so werden Punkte durch $\hat{\delta}_n$ falsch klassifiziert. Gefunden werden soll eine Hyperebene $\hat{\delta}_n$, sodass M möglichst groß und positiv ist:

$$\begin{aligned} \max_{\beta,\beta_0} M \quad \text{unter NB} \quad \forall i = 1, \ldots, n : \quad & Y_i \cdot \frac{1}{\|\beta\|_2}(X_i^T \beta + \beta_0) = m_i \geq M \geq 0, \\ \iff & Y_i (X_i^T \beta + \beta_0) \geq M \|\beta\|_2 \geq 0 \end{aligned} \tag{4.15}$$

Schritt 2 (β, β_0) ist im hergeleiteten Optimierungsproblem nicht eindeutig bestimmt. Jedes $(c \cdot \beta, c \cdot \beta_0)$ mit $c > 0$ ist auch Lösung. Wir können diese Lösungen ausschließen, indem wir für c eine Normierung vorgeben. Sei c so gewählt, dass

$$\|\beta\|_2 = 1/M. \tag{4.16}$$

Eingesetzt in Gl. (4.15) ergibt sich:

Definition 4.22 (Optimale separierende Hyperebene)
Sind $\hat{\beta}, \hat{\beta}_0$ Lösungen von

$$\min_{\beta \in \mathbb{R}^d, \beta_0 \in \mathbb{R}} \frac{1}{2}\|\beta\|_2^2 \quad \text{unter NB} \quad \forall i = 1, \ldots, n : \quad Y_i (X_i^T \beta + \beta_0) \geq 1, \tag{4.17}$$

so heißt $\hat{\delta}_n^{OSH}(x) = \hat{\beta}^T x + \hat{\beta}_0$ *optimale separierende Hyperebene,* und

$$\hat{f}_n^{OSH}(x) = \operatorname{sign}(\hat{\delta}_n^{OSH}(x)), \qquad \operatorname{sign}(z) = \begin{cases} 1, & z \geq 0, \\ -1, & z < 0 \end{cases}$$

ist der zugehörige Klassifizierer. ♦

Bemerkung 4.23

- Dies ist ein konvexes Optimierungsproblem im Sinne von Gl. (2.12) mit

$$G(\beta, \beta_0) := \left(1 - Y_i(X_i^T \beta + \beta_0)\right)_{i=1,\ldots,n}, \qquad g(\beta, \beta_0) := \frac{1}{2}\|\beta\|_2^2.$$

- Falls die Trainingsdaten nicht durch eine Hyperebene trennbar sind, so besitzt das Optimierungsproblem ab der Bedingung $M \geq 0$ bzw. ab der Substitution in Gl. (4.16) keine Lösung mehr. In diesem Sinne ist der Klassifizierer nicht geeignet, wenn sich die Beobachtungen verschiedener Klassen „überlappen".
- $\hat{\delta}_n^{OSH}$ hat auf beiden Seiten den maximal möglichen Abstand zu den Trainingsdaten. Durch die Substitution in Gl. (4.16) entsprechen nun Beobachtungen $i \in \{1, \ldots, n\}$ mit $Y_i(X_i^T \beta + \beta_0) = 1$ den Punkten, welche diesen maximal möglichen Abstand bestimmen. Diese Punkte liegen also genau auf der in Abb. 4.8 dargestellten gepunkteten Linie, beschrieben durch $\hat{\delta}_n(x) = \pm 1$, und werden *Supportvektoren* genannt. Ein Supportvektor „stützt" die beiden Geraden $\hat{\delta}_n(x) = \pm 1$ („Straßenränder"), denn aufgrund seiner Existenz kann der Abstand von $\hat{\delta}_n^{OSH}$ zu den Trainingsdaten nicht vergrößert werden. Für $d = 2$ muss es offensichtlich mindestens drei Supportvektoren geben.

Durch eine Analyse des Optimierungsproblems mittels des Satzes 2.33 erhalten wir mehr Einsicht in die Lösungen und deren Eigenschaften.

Lemma 4.24 Falls die Punktmengen $\{X_i : Y_i = -1\}$ und $\{X_i : Y_i = +1\}$ durch eine Hyperebene trennbar sind, ist die Slater-Bedingung für das Optimierungsproblem Gl. (4.17) erfüllt.

Beweis Aufgrund der Voraussetzung gibt es $\beta \in \mathbb{R}^d$, $\beta_0 \in \mathbb{R}$ mit der Eigenschaft $Y_i(X_i^T \beta + \beta_0) > 0$ für alle $i = 1, \ldots, n$. Dasselbe erhalten wir für $\beta' := c\beta, \beta_0' := c\beta_0$ mit beliebigem $c > 0$. Für $c > 0$ groß genug folgt für alle $i = 1, \ldots, n$: $G(\beta', \beta_0')_i < 0$.

Sei nun

$$\ell : \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}, \quad \ell(\beta, \beta_0, p) := g(\beta, \beta_0) + p^T G(\beta, \beta_0)$$

die zum Optimierungsproblem gehörige Lagrange-Funktion (vgl. Definition 2.32). Nach Satz 2.33 erfüllt eine Lösung $(\hat{\beta}, \hat{\beta}_0)$ von (4.17) die Optimalitätsbedingungen, d. h., es gibt Lagrange-Multiplikatoren $\hat{p} = (\hat{p}_1, \ldots, \hat{p}_n)^T \in \mathbb{R}^n_{\geq 0}$ mit

$$0 \geq G(\hat{\beta}, \beta_0), \tag{4.18}$$

$$0 = \langle \hat{p}, G(\hat{\theta}) \rangle, \tag{4.19}$$

$$0 = \nabla_\beta \ell(\hat{\beta}, \hat{\beta}_0, \hat{p}) = \hat{\beta} - \sum_{i=1}^{n} \hat{p}_i Y_i X_i^T \quad \Longleftrightarrow \quad \hat{\beta} = \sum_{i=1}^{n} \hat{p}_i Y_i \cdot X_i, \tag{4.20}$$

$$0 = \nabla_{\beta_0} \ell(\hat{\beta}, \hat{\beta}_0, \hat{p}) = -\sum_{i=1}^{n} \hat{p}_i Y_i \quad \Longleftrightarrow \quad 0 = \sum_{i=1}^{n} \hat{p}_i Y_i. \tag{4.21}$$

Ausgeschrieben lautet die Optimalitätsbedingung Gl. (4.19):

$$\sum_{i=1}^{n} \hat{p}_i \left[1 - Y_i (X_i^T \hat{\beta} + \hat{\beta}_0) \right] = 0$$

Wegen $\hat{p}_i \geq 0$, $1 - Y_i(X_i^T \hat{\beta} + \hat{\beta}_0) \leq 0$ sind alle Summanden kleiner gleich null. Da die Summe null sein soll, müssen alle Summanden null sein. Es folgt für alle $i \in \{1, \ldots, n\}$:

$$\hat{p}_i = 0 \quad \text{oder} \quad 1 - Y_i(X_i^T \hat{\beta} + \hat{\beta}_0) = 0$$

Die Lagrange-Multiplikatoren mit $\hat{p}_i \neq 0$ geben also genau die Supportvektoren an, vgl. Bemerkung nach Definition 4.22.

Gl. (4.20) bedeutet: Der Normalenvektor $\hat{\beta}$ der optimalen separierenden Hyperebene $\hat{\delta}_n^{OSH}$ ergibt sich als Linearkombination der Supportvektoren. Diese Feststellung ist später von enormem theoretischem Interesse, da sie eine einfache Beziehung zwischen den Lagrange-Multiplikatoren und der Lösung $\hat{\beta}$ erlaubt.

Zum Abschluss vergleichen wir kurz die Klassifizierer der bisher betrachteten Methoden:

Bemerkung 4.25 (Vergleich mit LDA und logistischer Regression)

- Der Klassifizierer $\hat{f}_n^{OSH}$ ist rein grafisch und ohne Modellannahme motiviert. Im Gegensatz zu logistischer Regression und LDA haben hier die Beobachtungen, welche nahe an Beobachtungen anderer Klassen liegen, einen stärkeren Einfluss auf $\hat{f}_n^{OSH}$. Daher ist $\hat{f}_n^{OSH}$ auch robuster gegenüber Ausreißern in den Trainingsdaten.
- Bei LDA zählen alle Punkte zur Konstruktion des Klassifizierers gleich stark, auch diejenigen, die weit weg vom optimalen Entscheidungsrand liegen. LDA liefert eventuell keine separierende Hyperebene, selbst wenn es eine gibt. Falls die Trainingsdaten die Modellannahme der LDA erfüllen, ist der LDA-Klassifizierer besser als $\hat{f}_n^{OSH}$, da $\hat{f}_n^{OSH}$ sich in seiner Form zu stark an den verrauschten Daten am Entscheidungsrand beeinflussen lässt.

- Der Klassifizierer der logistischen Regression $\hat{f}_n^{LR}$ findet eine separierende Hyperebene, falls es eine gibt, denn dann wird die Optimierungsfunktion in Gl. (4.9) maximiert.

Im Folgenden verallgemeinern wir $\hat{f}_n^{OSH}$, so dass dieser auch für Trainingsdaten, welche nicht linear separierbar sind, ein Ergebnis liefert.

4.4 Support Vector Machines (SVM)

Support Vector Machines bilden eine Verallgemeinerung von $\hat{f}_n^{OSH}$. Sie liefern auch im Falle nicht linear separierbarer Trainingsdaten einen Klassifizierer, wobei die Grundidee von $\hat{f}_n^{OSH}$ und insbesondere dessen grafische Motivation aufrechterhalten wird.

Während eine optimale separierende Hyperebene so positioniert werden musste, dass auf der einen Seite die Trainingsdaten $\{X_i : Y_i = -1\}$ und auf der anderen Seite die Trainingsdaten $\{X_i : Y_i = +1\}$ lagen, so erlauben wir nun, dass sich einige Trainingsdaten auf der falschen Seite befinden dürfen. Dazu führen wir für jede Beobachtung X_i einen Toleranzparameter ξ_i ein, welcher den Grad der Falschklassifizierung auf einer Skala von 0 bis ∞ misst. Indem wir die Summe der ξ_i minimieren, vermeiden wir zu viele falsch klassifizierte Trainingsdaten.

Formal sei $\xi = (\xi_1, \ldots, \xi_n)^T \in \mathbb{R}^n$. Wir nennen ξ_i die zu X_i gehörige *Schlupfvariable* (engl. *slack variable*). Wir ändern die Nebenbedingung der optimalen separierenden Hyperebene in Gl. (4.22) zu

$$Y_i(X_i^T \beta + \beta_0) \geq 1 - \xi_i$$

unter den zusätzlichen Bedingungen $\xi_i \geq 0, \sum_{i=1}^n \xi_i \leq D$. Hierbei ist $D > 0$ eine Schranke, die proportional zur Anzahl und Stärke der erlaubten Falschklassifizierungen ist. $(1 - \xi_i)$ signalisiert, ob und mit wie viel Abstand der Punkt X_i auf der falschen Seite der Hyperebene liegt. Wir wollen also immer noch eine möglichst breite ‚Straße' zwischen den Trainingsdaten finden, geben aber mittels D eine maximale ‚Menge' an Trainingsdaten an, welche in die Straße hineinragen oder sogar auf der falschen Seite liegen dürfen.

Damit erhalten wir die folgende neue Formulierung:

$$\min_{\beta,\beta_0,\xi} \frac{1}{2}\|\beta\|_2^2 \qquad \text{unter NB} \qquad \forall i = 1, \ldots, n: \quad Y_i(X_i^T \beta + \beta_0) \geq 1 - \xi_i,$$

$$\xi_i \geq 0,$$

$$\sum_{i=1}^n \xi_i \leq D$$

Wie bei der Äquivalenz von Ridge- oder Lasso-Schätzern in Kap. 2 kann die Nebenbedingung $\sum_{i=1}^n \xi_i \leq D$ mittels Satz 2.33 durch einen additiven Term in der Optimierungsfunktion ersetzt werden. Wir erhalten folgende Definition:

Definition 4.26 (SVM-Klassifizierer)
Sei $C \geq 0$. Seien $\hat{\beta}_C, \hat{\beta}_{0,C}, \hat{\xi}$ Lösungen von

$$\min_{\beta \in \mathbb{R}^k, \beta_0 \in \mathbb{R}, \xi \in \mathbb{R}^n} \frac{1}{2}\|\beta\|_2^2 + C \sum_{i=1}^n \xi_i \quad \text{unter NB} \quad \forall i = 1, \ldots, n: \quad Y_i(X_i^T \beta + \beta_0) \geq 1 - \xi_i,$$
$$\xi_i \geq 0, \tag{4.22}$$

setze $\hat{\delta}_{n,C}^{SVM}(x) = \hat{\beta}_C^T x + \hat{\beta}_{0,C}$. Dann heißt

$$\hat{f}_{n,C}^{SVM}(x) = \text{sign}(\hat{\delta}_{n,C}^{SVM}(x))$$

SVM-Klassifizierer. ♦

Bemerkung 4.27

- Die Formulierung der Nebenbedingung an ξ_i in Definition 4.26 ist praktischer als die Bedingung $\sum_{i=1}^n \xi_i \leq D$. Letztere gibt nämlich eine maximale Schranke an $\sum_{i=1}^n \xi_i$ vor, die in Anwendungen evtl. zu Beginn schwierig festzulegen ist. Dahingegen gibt $C \geq 0$ im Penalisierungsterm $C \sum_{i=1}^n \xi_i$ anschaulich nur die „Stärke" an, mit welcher $\sum_{i=1}^n \xi_i$ minimiert werden soll.
- Die Interpretation der Werte von $\hat{\xi}_i$ ist wie folgt:
  - Ist $\hat{\xi}_i > 1$, so wird X_i durch $\hat{f}_{n,C}^{SVM}$ falsch klassifiziert.
  - Ist $\hat{\xi}_i \in (0, 1]$, so wird X_i durch $\hat{f}_{n,C}^{SVM}$ korrekt klassifiziert, liegt aber innerhalb der Menge $\{x : \hat{\delta}_{n,C}^{SVM}(x) \in [-1, 1]\}$, d. h. im Inneren der gefundenen breitestmöglichen „Straße".
  - Ist $\hat{\xi}_i = 0$, so wird X_i durch $\hat{f}_{n,C}^{SVM}$ korrekt klassifiziert, und X_i liegt auch nicht innerhalb der Menge $\{x : \hat{\delta}_{n,C}^{SVM}(x) \in [-1, 1]\}$.

Definition 4.26 liefert ein quadratisches konvexes Programm im Sinne von Satz 2.33, für dessen Lösung schnelle numerische Verfahren existieren, vgl. Abschn. 4.5. Zur Berechnung wird allerdings häufig das zugehörige *duale Problem* genutzt, dessen Formulierung wir nachfolgend herleiten. Diese Formulierung wird uns im Folgenden auch helfen, die Methode auf nichtlineare Entscheidungsränder zu verallgemeinern.

Beispiel 4.28 Wir betrachten die folgenden Modelle (vgl. Beispiele 4.4, 4.5): Es sei jeweils Y_i i. i. d. mit $\mathbb{P}(Y_1 = -1) = \mathbb{P}(Y_1 = +1) = \frac{1}{2}$ und

a) $(X_i|Y_i = -1) \sim N((0,2)^T, 0{,}2I_{2\times 2})$, $(X_i|Y_i = -1) \sim N((2,0)^T, 0{,}2I_{2\times 2})$,
b) $(X_i|Y_i = -1) \sim N((0,0)^T, \text{diag}(2,0{,}2))$, $(X_i|Y_i = -1) \sim N((2{,}5,0)^T, \text{diag}(0{,}2,2))$,
c)

$$(X_i|Y_i = 1) \sim \frac{1}{2}N((-4,0)^T, \frac{1}{4}I_{2\times 2}) + \frac{1}{2}N((0,0)^T, \frac{1}{4}I_{2\times 2}),$$
$$(X_i|Y_i = 2) \sim \frac{1}{2}N((-2,0)^T, \frac{1}{4}I_{2\times 2}) + \frac{1}{2}N((2,0)^T, \frac{1}{4}I_{2\times 2}).$$

Wir beobachten jeweils $n = 120$ Trainingsdaten (X_i, Y_i). Die geschätzten Entscheidungsregionen mittels $\hat{f}_{n,C}^{SVM}$ für $C \in \{0{,}1, 100\}$ sind in Abb. 4.9 dargestellt.

In Abb. 4.9a, b liegt der Fall linear separierbarer Daten vor. Eine starke Bestrafung ($C = 100$) der Schlupfvariablen ξ_i führt zu einer möglichst engen ‚Straße' um die Hyperebene $\hat{\delta}_{n,C}^{SVM}$ ohne Fehlklassifizierungen und damit zu dem gleichen Ergebnis wie beim Algorithmus der optimal separierenden Hyperebene (vgl. Definition 4.22). Ist $C = 0{,}1$, so gibt es mehr Supportvektoren (d. h. $\xi_i \neq 0$) und für die Berechnung des Normalenvektors von $\hat{\delta}_{n,C}^{SVM}$ werden gemäß Gl. (4.23) mehr Datenpunkte einbezogen. Dies führt im Allgemeinen zu einer stabileren Schätzung von $\hat{\delta}_{n,C}^{SVM}$, welche nicht so stark von den verrauschten Trainingsdaten am Rang der beiden Punktmengen $\{X_i : Y_i = +1\}$ und $\{X_i : Y_i = -1\}$ abhängt. In Abb. 4.9c, d wird die SVM auf nicht linear separierbaren Trainingsdaten des Modells (b) angewandt. Auch hier führt $C = 100$ zu einer kleineren Menge an Supportvektoren als $C = 0{,}1$ und zu einem geringeren Abstand von $\{x : \hat{\delta}_{n,C}^{SVM}(x) = 1\}$ zu $\{x : \hat{\delta}_{n,C}^{SVM}(x) = -1\}$. In Abb. 4.9e, f wird die SVM auf die Trainingsdaten des Modells (c) angewandt. Hier ist aufgrund der Struktur der Daten keine sinnvolle Trennung mit einer Hyperebene möglich; für jedes $C > 0$ erzeugt die Trennung mittels $\hat{\delta}_{n,C}^{SVM}(x)$ in etwa eine gleich große Menge an Supportvektoren und falsch klassifizierten Trainingsdaten.

4.4.1 Duale Formulierung der SVM

Die im Folgenden hergeleitete Methode ist eine Umformulierung des Optimierungsproblems der SVM mit Parametern $\beta \in \mathbb{R}^d$, $\beta_0 \in \mathbb{R}$ und $\xi = (\xi_1, \ldots, x_n)^T \in \mathbb{R}^n$ in dessen duales Optimierungsproblem mit Parametern $\alpha \in \mathbb{R}^n$. Im Falle hochdimensionaler Daten ($d \gg n$) wird also trotzdem nur eine Optimierung über n Parameter durchgeführt. Dies ist interessant in Hinsicht auf die Erfahrungen mit $\hat{f}_n^{LR,nichtlin}$ aus Definition 4.13. Dort wurde die ursprüngliche Methode auf die erweiterten Daten $\tilde{X}_i \in \mathbb{R}^{\tilde{d}}$ angewandt, wobei $\tilde{d} \gg d$ gelten konnte.

Ausgehend von der Formulierung in Definition 4.26 und den Bezeichnungen in Satz 2.33, setze für $\theta = (\beta, \beta_0, \xi)$:

$$f(\theta) = \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^{n}\xi_i, \quad G(\theta) = \begin{pmatrix} \left(1 - \xi_i - Y_i(X_i^T\beta + \beta_0)\right)_{i=1,\ldots,n} \\ -\xi \end{pmatrix}.$$

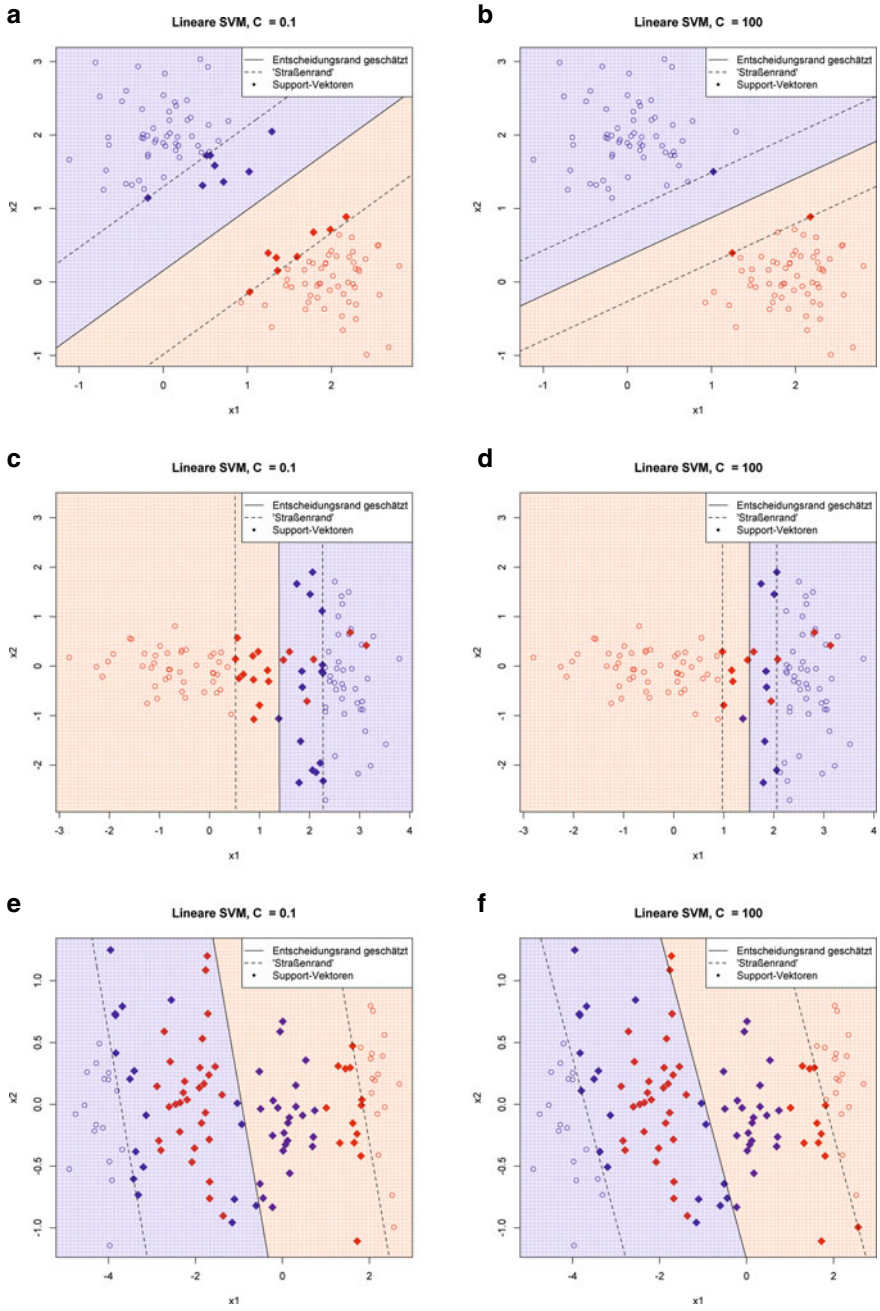


Abb. 4.9 Darstellung der Klassifizierer $\hat{f}_{n,C}^{SVM}$ angewandt auf die Daten aus Beispiel 4.28 mit dem zugehörigen „Straßenrand“ $\{\hat{\delta}_{n,C}^{SVM}(x) = \pm 1\}$. **a**, **c**, **e** $C = 0{,}1$, **b**, **d**, **f** $C = 100$. **a**, **b** Modell (a). **c**, **d** Modell (b). **e**, **f** Modell (c). Die farblichen Schattierungen entsprechen den geschätzten Entscheidungsregionen der jeweiligen Klassen

Wir nennen die zu G gehörigen dualen Parameter $\alpha = (\alpha_1, \ldots, \alpha_n)^T$, $\gamma = (\gamma_1, \ldots, \gamma_n)^T \in \mathbb{R}^n_{\geq 0}$ und definieren die Lagrange-Funktion:

$$\ell(\theta, \alpha, \gamma) = f(\theta) + \sum_{i=1}^{n} \alpha_i \left(1 - \xi_i - Y_i(X_i^T \beta + \beta_0)\right) - \sum_{i=1}^{n} \gamma_i \xi_i$$

Für $\xi_i > 0$ groß genug ($i = 1, \ldots, n$) ist offensichtlich $G(\theta)_i < 0$. Daher ist die Slater-Bedingung erfüllt, und eine Lösung $(\hat{\beta}, \hat{\beta}_0, \hat{\xi})$ von Gl. (4.22) erfüllt die Optimalitätsbedingungen aus Satz 2.33, d. h., es gibt $\hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n)^T$, $\hat{\gamma} = (\hat{\gamma}_1, \ldots, \hat{\gamma}_n)^T \in \mathbb{R}^n_{\geq 0}$ mit

$$0 = \nabla_\beta \ell(\hat{\theta}, \hat{\alpha}, \hat{\gamma}) \iff \hat{\beta} = \sum_{i=1}^{n} \hat{\alpha}_i Y_i \cdot X_i, \tag{4.23}$$

$$0 = \nabla_{\beta_0} \ell(\hat{\theta}, \hat{\alpha}, \hat{\gamma}) \iff \sum_{i=1}^{n} \hat{\alpha}_i Y_i = 0, \tag{4.24}$$

$$0 = \nabla_\xi \ell(\hat{\theta}, \hat{\alpha}, \hat{\gamma}) \iff \hat{\alpha}_i = C - \hat{\gamma}_i, \tag{4.25}$$

$$\hat{\alpha}_i \geq 0, \quad \hat{\gamma}_i \geq 0, \tag{4.26}$$

$$0 = \langle (\hat{\alpha}^T, \hat{\gamma}^T)^T, G(\hat{\theta}) \rangle \iff 0 = \hat{\alpha}_i \left(1 - \hat{\xi}_i - Y_i(X_i^T \hat{\beta} + \hat{\beta}_0)\right),$$

$$0 = \hat{\gamma}_i \hat{\xi}_i, \tag{4.27}$$

$$0 \geq G(\hat{\theta}) \iff \hat{\xi}_i \geq 0, \quad 1 - \hat{\xi}_i - Y_i(X_i^T \hat{\beta} + \hat{\beta}_0) \leq 0. \tag{4.28}$$

Unser Ziel ist die Ermittlung des dualen Problems Gl. (2.15). Das ist ein Optimierungsproblem in den Parametern α, γ und nicht mehr in den Parametern β, β_0, ξ. Die Darstellung Gl. (2.15) gibt vor, wie das duale Problem erhalten werden kann: Wir setzen die Optimalitätsbedingungen Gl. (4.23), (4.24), (4.25) in ℓ ein:

$$\begin{aligned}
\ell(\hat{\theta}, \hat{\alpha}, \hat{\gamma}) &= \frac{1}{2} \left\| \sum_{i=1}^{n} \hat{\alpha}_i Y_i X_i \right\|_2^2 + \underbrace{\sum_{i=1}^{n} \xi_i (\hat{\lambda} - \hat{\alpha}_i - \hat{\gamma}_i)}_{=0} + \hat{\beta}_0 \underbrace{\sum_{i=1}^{n} \hat{\alpha}_i Y_i}_{=0} \\
&\quad + \sum_{i=1}^{n} \hat{\alpha}_i - \sum_{i=1}^{n} \hat{\alpha}_i Y_i X_i^T \left(\sum_{i'=1}^{n} \hat{\alpha}_{i'} Y_{i'} X_{i'} \right) \\
&= \sum_{i=1}^{n} \hat{\alpha}_i - \frac{1}{2} \sum_{i,i'=1}^{n} \hat{\alpha}_i \hat{\alpha}_{i'} Y_i Y_{i'} X_i^T X_{i'} \\
&= -\frac{1}{2} \hat{\alpha}^T Q \hat{\alpha} + \mathbb{1}^T \hat{\alpha}
\end{aligned}$$

mit den Abkürzungen

- $\mathbb{1} = (1, \ldots, 1)^T \in \mathbb{R}^n$ ein Vektor bestehend aus n Einsen,
- $\mathbb{Y} = (Y_1, \ldots, Y_n)^T$,
- $Q = (Q_{ij})_{i,j=1,\ldots,n}$ eine Matrix mit

$$Q_{ij} := Y_i Y_j X_i^T X_j, \qquad i, j = 1, \ldots, n.$$

Die Optimalitätsbedingungen liefern Nebenbedingungen an die Extremstellen:

$$(5.24) \iff \sum_{i=1}^{n} \hat{\alpha}_i Y_i = 0, \qquad (5.25), (5.26) \Rightarrow \forall i = 1, \ldots, n: \quad 0 \le \hat{\alpha}_i \le C$$

Die restlichen Bedingungen aus Gl. (4.23), (4.27) und (4.28) stellen Verbindungen zum ursprünglichen (primalen) Optimierungsproblem her und liefern keine Nebenbedingungen für das duale Problem. Weder Nebenbedingungen noch die Optimierungsfunktion enthalten noch die Lagrange-Multiplikatoren $\hat{\gamma}$, diese konnten also vollständig entfernt werden. Wir erhalten folgende Definition:

Definition 4.29 (SVM-Klassifizierer, duale Version)
Sei $C \ge 0$. Sei $\hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n)^T \in \mathbb{R}^n$ Lösung von

$$\min_{\alpha \in \mathbb{R}^n} \left\{ \frac{1}{2} \alpha^T Q \alpha - \mathbb{1}^T \alpha \right\} \qquad \text{unter NB} \qquad \mathbb{Y}^T \alpha = 0, \quad \forall i = 1, \ldots, n: 0 \le \alpha_i \le C.$$

Dann gilt $\hat{\beta}_C = \sum_{i=1}^n \hat{\alpha}_i Y_i \cdot X_i$. Ist $i' \in \{1, \ldots, n\}$ mit $0 < \hat{\alpha}_{i'} < C$, so gilt $\hat{\beta}_{0,C} = Y_{i'} - X_{i'}^T \hat{\beta}_C$.

Der zugehörige Klassifizierer $\hat{f}_{n,C}^{SVM}$ ist in Definition 4.26 gegeben. ♦

Bemerkung 4.30

- Das duale Problem ist ursprünglich ein Maximierungsproblem; durch Multiplikation der Zielfunktion ℓ mit (-1) wurde es ein Minimierungsproblem.
- Das duale Problem hat zunächst keine anschauliche Bedeutung. Dessen Lösungen $\hat{\alpha}$ liefern aber mittels der Optimalitätsbedingungen die Optimierer $\hat{\beta}_C$, $\hat{\beta}_{0,C}$ des ursprünglichen Problems: Die Formel für $\hat{\beta}_C$ folgt aus Gl. (4.23); ist $0 < \hat{\alpha}_{i'} < C$, so folgen mit Gl. (4.25), (4.27) die Aussagen $\hat{\gamma}_i \ne 0, \hat{\xi}_i = 0$ und dann mit Gl. (4.27) $1 = Y_i(X_i^T \hat{\beta}_C + \hat{\beta}_{0,C})$.
- Die Werte der $\hat{\alpha}_i$ haben folgende Interpretation:
  - Gilt $0 < \hat{\alpha}_i < C$, so folgt wie oben $\hat{\xi}_i = 0$ und $1 = Y_i(X_i^T \hat{\beta}_C + \hat{\beta}_{0,C})$. Das bedeutet, X_i erfüllt die Nebenbedingung der optimalen separierenden Hyperebene, wird korrekt durch $\hat{f}_{n,C}^{SVM}$ klassifiziert und liegt auf dem „Straßenrand“, welcher durch die Gleichungen $\hat{\delta}_{n,C}(x) = \pm 1$ beschrieben wird. Diese Vektoren nennen wir auch hier *Support-Vektoren.*

- Gilt $\hat{\alpha}_i = 0$, so wird X_i korrekt klassifiziert und liegt außerhalb der Menge $\{x \in \mathbb{R}^d : \hat{\delta}_{n,C}(x) \in [-1, 1]\}$.
- Gilt $\hat{\alpha}_i = C$, so gilt wegen (4.25), (4.27): $\hat{\xi}_i \neq 0$ und $Y_i(X_i^T \hat{\beta} + \hat{\beta}_0) = 1 - \hat{\xi}_i$, d. h., X_i liegt innerhalb der Menge $\{x \in \mathbb{R}^d : \hat{\delta}_{n,C}(x) \in [-1, 1]\}$ oder wird (im Falle $\hat{\xi}_i > 1$) von $\hat{f}_{n,C}^{SVM}$ falsch klassifiziert. Auch in diesem Fall nennen wir X_i einen *Support-Vektor,* da er mittels Gl. (4.23) zur Änderung von $\hat{\beta}$ beigetragen hat.

Bis jetzt erlaubt die Formulierung der SVM nur lineare Entscheidungsränder. Ähnlich wie bei der logistischen Regression können wir durch Transformation der ursprünglichen Beobachtungen X_i nichtlineare Unterscheidungsränder erhalten. Dies ist Gegenstand von Abschn. 4.4.2.

4.4.2 Verallgemeinerte SVM und der Kern-Trick

Der SVM-Klassifizierer aus Definition 4.26 teilt den Beobachtungsraum $\mathcal{X} = \mathbb{R}^d$ entlang des Entscheidungsrands

$$\partial\hat{\Omega}_1 = \{x \in \mathbb{R}^d : \hat{\delta}_{n,C}(x) = \hat{\beta}_C^T x + \hat{\beta}_{0,C} = 0\}$$

in zwei Klassen auf. Falls der optimale Entscheidungsrand jedoch *nicht* die Form $\partial\Omega_1^* = \{x \in \mathbb{R}^d : \beta_0 + \beta^T x = 0\}$ mit $\beta_0 \in \mathbb{R}$, $\beta \in \mathbb{R}^d$ besitzt, wird $\partial\hat{\Omega}_1$ auch für große Beobachtungszahlen n keine gute Schätzung für $\partial\Omega_1$ liefern.

Eine Verallgemeinerung des Verfahrens kann wie folgt erreicht werden: Wir ersetzen die ursprünglichen Beobachtungen X_i durch neue Beobachtungen $\tilde{X}_i = h(X_i)$ mit einer geeigneten Funktion $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ und wenden dann den SVM-Klassifizierer $\hat{f}_{n,C}^{SVM}$ auf die Trainingsdaten $(\tilde{X}_i, Y_i)$ an. Wir erhalten dann eine Aufteilung des Raums $\mathbb{R}^{\tilde{d}}$ entlang des Entscheidungsrands

$$\partial\tilde{\Omega}_1^* = \{\tilde{x} \in \mathbb{R}^{\tilde{d}} : \tilde{\beta}_C^T \tilde{x} + \tilde{\beta}_{0,C} = 0\}$$

bzw. im ursprünglichen Raum $\mathbb{R}^d$ entlang

$$\partial\tilde{\Omega}_1^{*nichtlin} = \{x \in \mathbb{R}^d : \tilde{\beta}_C^T h(x) + \tilde{\beta}_{0,C} = 0\}. \tag{4.29}$$

Üblicherweise ist die Dimension $\tilde{d}$ der erweiterten Beobachtungen wesentlich größer als d. Motivation für dieses Vorgehen ist die Hoffnung, dass bei geeigneter Wahl der Funktion h eine lineare Trennung der Punktmengen $\{\tilde{X}_i : Y_i = +1\}$ und $\{\tilde{X}_i : Y_i = -1\}$ im höherdimensionalen Raum $\mathbb{R}^{\tilde{d}}$ möglich ist, obwohl es für die Mengen $\{X_i : Y_i = +1\}$ und $\{X_i : Y_i = -1\}$ in $\mathbb{R}^d$ nicht möglich war. Ein Beispiel ist in Abb. 4.10 gegeben: Dort sind die ursprünglichen Trainingsdaten nicht linear trennbar, doch durch die Erweiterung auf $\tilde{X}_i = h(X_i)$ mit

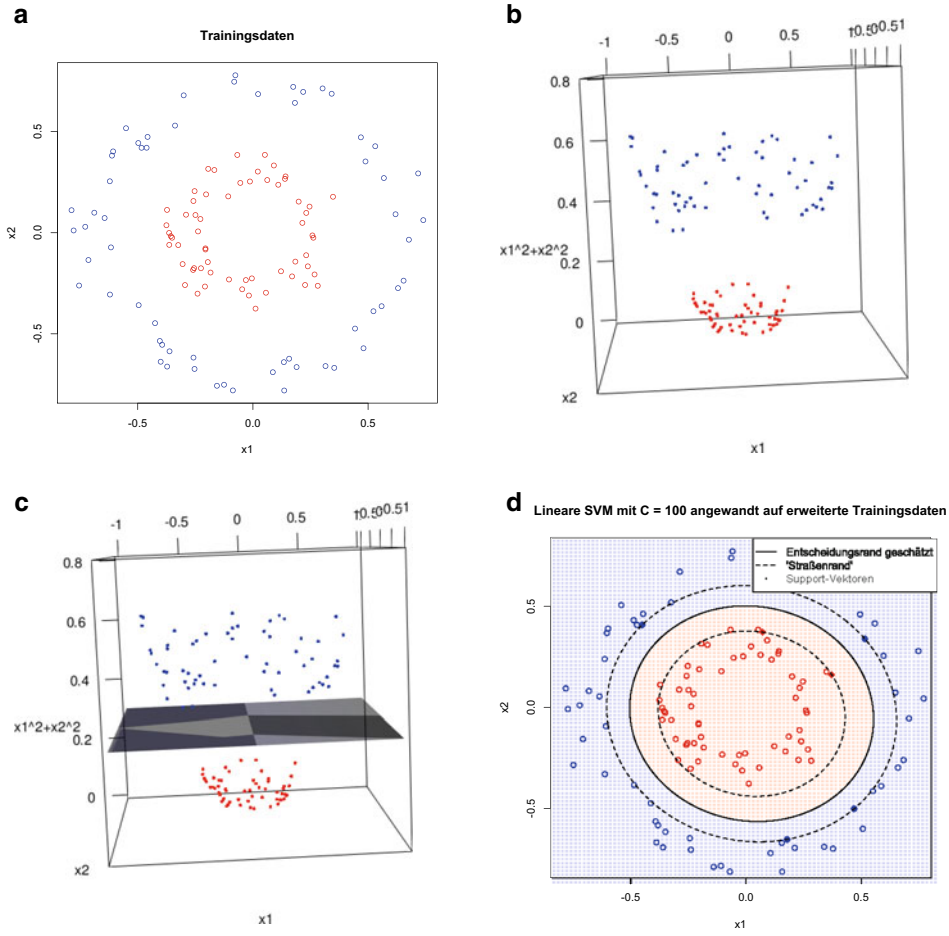


Abb. 4.10 Motivation für die nichtlineare SVM. **a** nicht linear trennbare Trainingsdaten in $\mathbb{R}^2$, **b** Einbettung der Trainingsdaten in einen höherdimensionalen Raum $\mathbb{R}^3$ mittels Gl. (4.30), **c** separierende Hyperebene in $\mathbb{R}^3$, ermittelt durch den SVM-Algorithmus mit $C = 100$, **d** zugehöriger Entscheidungsrand $\partial\hat{\tilde{\Omega}}_1$ aus Gl. (4.31) und Supportvektoren im ursprünglichen Raum $\mathbb{R}^2$ der Trainingsdaten

$$h : \mathbb{R}^2 \to \mathbb{R}^3, h(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2)^T \tag{4.30}$$

ist eine lineare Trennung möglich. Die gefundene separierende Hyperebene im höherdimensionalen Raum $\mathbb{R}^3$ (und die damit ermittelten Parameter $\hat{\tilde{\beta}}_C, \hat{\tilde{\beta}}_{0,C}$) führt zu einem nichtlinearen Entscheidungsrand im ursprünglichen Raum $\mathbb{R}^2$ durch

$$\partial\hat{\tilde{\Omega}}_1 = \{x \in \mathbb{R}^d : \hat{\tilde{\beta}}_C^T h(x) + \hat{\tilde{\beta}}_{0,C} = 0\}. \tag{4.31}$$

Die Nichtlinearität des Entscheidungsrands wird dabei allein durch die gewählte Funktion h erzeugt. Es erscheint sinnvoll, $\tilde{d}$ möglichst groß zu wählen, um genügend interessante Funktionen $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ zuzulassen. Ist zum Beispiel $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ mit

$$h(x) = (x_1, \ldots, x_d, x_1^2, x_1x_2, x_1x_3, \ldots, x_1x_d, x_2^2, x_2x_3, \ldots, x_2x_d, \ldots, x_d^2) \tag{4.32}$$

und $\tilde{d} = d + \sum_{i=1}^{d} i = \frac{d(d+3)}{2}$, so erlaubt (4.29) nun eine Aufteilung des Raums mittels Ellipsoiden anstatt zuvor nur mit Geraden.

Wie bereits bei der logistischen Regression bemerkt, ist der Einbettungsansatz in der Praxis mit folgenden Problemen verbunden:

(P1) Welche Funktion h wählt man? Gerade bei hochdimensionalen Trainingsdaten ist eine gute Wahl von h nicht offensichtlich.

(P2) Damit genug neue Funktionen aus den ursprünglichen X_i gebildet werden können, muss $\tilde{d} \gg d$ sein, oft sogar $\tilde{d} \gg d^2$ (vgl. Gl. (4.32)). Wird der Ansatz ohne Weiteres angewandt, so müssen die neuen Trainingsdaten $h(X_i)$ berechnet und abgespeichert werden, was für große d bereits lange Rechenzeiten und einen hohen Speicherplatzbedarf verursachen kann.

Wie sich gleich herausstellt, erlaubt die duale Formulierung der SVM im Gegensatz zur logistischen Regression eine Lösung des Problems (P2). Haben wir eine Funktion $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ ausgewählt und definieren wir die Matrix $\tilde{Q} = (\tilde{Q}_{ij})_{i,j=1,\ldots,n}$ durch

$$\tilde{Q}_{ij} = y_i y_j h(X_i)^T h(X_j), \tag{4.33}$$

so liefert eine Anwendung der SVM auf die eingebetteten Trainingsdaten $\tilde{X}_i = h(X_i)$ folgenden Algorithmus:

Definition 4.31 (Nichtlinearer SVM-Klassifizierer, duale Version)
Sei $C \geq 0$. Sei $\hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n)^T$ Lösung von

$$\min_{\alpha \in \mathbb{R}^n} \left\{ \frac{1}{2} \alpha^T \tilde{Q} \alpha - \mathbb{1}^T \alpha \right\} \qquad \text{unter NB} \qquad \mathbb{Y}^T \alpha = 0, \quad \forall i = 1, \ldots, n : 0 \leq \alpha_i \leq C.$$

Der zugehörige Klassifizierer lautet

$$\hat{f}_{n,C}(x) = \hat{f}_{n,C}^{SVM,nl}(x) = \text{sign}(\hat{\delta}_{n,C}^{SVM,nl}(x)),$$

wobei $\hat{\beta}_C^{nl} = \sum_{i=1}^{n} \hat{\alpha}_i Y_i \cdot h(X_i)$, und mit einem $i \in \{1, \ldots, n\}$ mit $0 < \hat{\alpha}_i < C$: $\hat{\beta}_{0,C}^{nl} := Y_i - X_i^T \hat{\beta}_C^{nl}$, und

$$\hat{\delta}_{n,C}^{SVM,nl}(x) = (\hat{\beta}_C^{nl})^T h(x) + \hat{\beta}_{0,C}^{nl} = \sum_{i=1}^{n} \hat{\alpha}_i Y_i \cdot h(X_i)^T h(x) + \hat{\beta}_{0,C}^{nl}. \tag{4.34}$$

♦

Bemerkungen

a) Die Dimension des Featureraums durch die Betrachtung von $\tilde{X}_i \in \mathbb{R}^{\tilde{d}}$ anstelle von $X_i \in \mathbb{R}^d$ ist wesentlich vergrößert (oft gilt $\tilde{d} \gg d$ und damit $\tilde{d} \gg n$). Trotzdem hängt das Optimierungsproblem nur von n Variablen ab, skaliert also zunächst nicht mit $\tilde{d}$.

b) Angenommen, die Komponenten von h sind genügend ‚vielfältig', so dass der optimale Entscheidungsrand sehr ähnlich zu

$$\partial\Omega_1 = \{x \in \mathbb{R}^d : \beta_0 + \beta^T h(x) = 0\}$$

mit geeigneten $\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^{\tilde{d}}$ ist. Aus der Interpretation von C als Bestrafung für Falschklassifizierungen folgt: Wählt man C groß genug, so sind nur wenige $\hat{\alpha}_i \neq 0$, daher ist der Lösungsvektor $\hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n)^T$ in Definition 4.31 dünn besetzt. Dies schützt die SVM auch bei sehr großen $\tilde{d}$ vor einer zu starken Anpassung an die Trainingsdaten.

c) Die Beobachtungen X_i, $i = 1, \ldots, n$ bzw. die neuen Beobachtungen $\tilde{X}_i = h(X_i)$ gehen in das Optimierungsproblem nur über die *Skalarprodukte* $h(X_i)^T h(X_j)$ bzw. $h(X_i)^T h(x)$ ein (vgl. Gl. (4.33) und (4.34)). Das bedeutet, eine Berechnung der neuen Beobachtungen $\tilde{X}_i = h(X_i)$ ist für die Anwendung der SVM mittels Definition 4.31 gar nicht nötig, man muss nur die Werte der Terme $h(X_i)^T h(X_j)$ bzw. $h(X_i)^T h(x)$ kennen.

Aus Bemerkung (c) folgt: Für die Anwendung der nichtlinearen SVM genügt es, die Werte der Terme $K(x, x') := h(x)^T h(x')$ für $x, x' \in \mathbb{R}^d$ festzulegen, eine explizite Definition der Funktion h ist nicht nötig. K berechnet also anschaulich die euklidischen Skalarprodukte der transformierten Beobachtungen $\tilde{X}_i = h(X_i)$ im Raum $\mathbb{R}^{\tilde{d}}$.

Im Folgenden werden wir sehen, dass eine explizite Darstellung von K der Form $K(x, x') = h(x)^T h(x')$ für theoretische Schlussfolgerungen ersetzt werden kann durch allgemeinere Eigenschaften von K. Diese wollen wir im Folgenden für einen allgemeinen metrischen Raum $(\mathcal{X}, D)$ festhalten:

Definition 4.32

Eine Funktion $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ heißt *Kernfunktion.*

(i) K heißt *symmetrisch,* falls für alle $x, x' \in \mathcal{X}$ gilt:

$$K(x, x') = K(x', x)$$

(ii) K heißt *positiv semidefinit,* falls für alle $n \in \mathbb{N}$ und für alle $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d$ gilt: Die Matrix

$$\left(K(x^{(i)}, x^{(j)})\right)_{i,j=1,\ldots,n}$$

ist positiv semidefinit.

(iii) K heißt *stetig,* falls K als Funktion $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ stetig ist.

K heißt *Mercer-Kern,* falls K eine Kernfunktion mit den Eigenschaften (i), (ii), (iii) ist. ♦

Ist nun $K : \mathbb{R}^d \to \mathbb{R}^d \to \mathbb{R}$ eine beliebige Kernfunktion und $Q^K = (Q^K_{ij})_{i,j=1,\ldots,n}$ definiert durch

$$Q^K_{ij} = Y_i Y_j K(X_i, X_j),$$

so können wir folgenden SVM-Klassifizierer definieren:

Definition 4.33 (Nichtlinearer SVM-Klassifizierer mit Kern K, duale Version)
Sei $C \geq 0$. Sei $\hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n)^T$ Lösung von

$$\min_{\alpha \in \mathbb{R}^n} \left\{ \frac{1}{2} \alpha^T Q^K \alpha - \mathbb{1}^T \alpha \right\} \qquad \text{unter NB} \qquad \mathbb{Y}^T \alpha = 0, \quad \forall i = 1, \ldots, n : 0 \leq \alpha_i \leq C.$$

Der zugehörige Klassifizierer lautet

$$\hat{f}_{n,C}(x) = \hat{f}^{SVM,K}_{n,C}(x) = \text{sign}(\hat{\delta}^{SVM,K}_{n,C}(x)),$$

wobei

$$\hat{\delta}^{SVM,K}_{n,C}(x) = \sum_{i=1}^{n} \hat{\alpha}_i Y_i \cdot K(X_i, x) + \hat{\beta}_{0,C},$$

und $\hat{\beta}_{0,C} = Y_i - \sum_{j=1}^{n} \hat{\alpha}_j Y_j \cdot K(X_j, X_i)$ mit einem $i \in \{1, \ldots, n\}$ mit $0 < \hat{\alpha}_i < C$. ♦

Die Frage nach der geeigneten Wahl einer Einbettung $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ verlagert sich damit zu der Frage, welche Kerne K für die Anwendung der SVM gewählt werden sollen. Die ursprüngliche lineare SVM kann erhalten werden mittels des *linearen Kerns*

$$K(x, x') = x^T x'.$$

Um auch nichtlineare Entscheidungsränder zuzulassen, müssen kompliziertere Kerne K gewählt werden. Die folgenden Kernfunktionen werden häufig in der Praxis gewählt (diese sind auch automatisch eine Antwort auf das Problem [P1] oben):

Beispiel 4.34 (Wahl von K in der Praxis)

(i) *Linearer Kern*

$$K^{lin}(x, x') = x^T x'$$

(ii) *Polynomkern* vom Grad $p \in \mathbb{N}$:

$$K_p^{poly}(x, x') = (1 + x^T x')^p$$

(iii) *Gauß-Kern* (oder *RBF-Kern*, engl. *radial basis function*) mit Abstieg $\gamma > 0$:

$$K_\gamma^{gauss}(x, x') = \exp(-\gamma \cdot \|x - x'\|_2^2)$$

(iii) *Sigmoidkern* mit $\kappa_1, \kappa_2 \in \mathbb{R}, \kappa_1 > 0$:

$$K_{\kappa_1,\kappa_2}^{sigm}(x, x') = \tanh(\kappa_1 x^T x' + \kappa_2)$$

Durch den Übergang zu Kernen in der Nutzung der SVM verlieren wir die Anschauung, welche Einbettungen h der ursprünglichen Beobachtungen betrachtet werden und wie die zugehörigen Entscheidungsränder $\partial\hat{\Omega}_1$ aussehen. Im Folgenden wollen wir rekonstruieren, welchen Funktionen h die jeweiligen Kerne entsprechen. Wir geben eine Antwort für (ii) in Beispiel 4.35 unten. Für (iii), (iv) ist die konkrete Beantwortung schwieriger und wird zunächst theoretisch in Satz 4.36 behandelt.

Beispiel 4.35 Ist $d = p = 2$, so gilt

$$K_p^{poly}(x, x') = h(x)^T h(x')$$

mit $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$, $\tilde{d} = 6$ und

$$h(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)^T,$$

denn:

$$\begin{aligned} K_p(x, x') &= (1 + x^T x')^2 = (1 + x_1x_1' + x_2x_2')^2 \\ &= 1 + 2x_1x_1' + 2x_2x_2' + (x_1x_1')^2 + (x_2x_2')^2 + 2x_1x_2x_1'x_2' \\ &= h(x)^T h(x') \end{aligned}$$

Der Polynomkern zur Ordnung $p = 2$ ermöglicht also mit h Ellipsen als Entscheidungsränder.

Ist K ein Mercer-Kern (vgl. Definition 4.32), so liefert der folgende Satz von Mercer (vgl. [40, Satz VI.4.2]) zumindest die theoretische Existenz geeigneter Funktionen h. Im Folgenden sei für $\mathcal{X} \subset \mathbb{R}^d$

$$L^2(\mathcal{X}) := \{h : \mathcal{X} \to \mathbb{R} \text{ messbar} : \int_{\mathcal{X}} h(x)^2 \, \mathrm{d}x < \infty\} \tag{4.35}$$

der Hilbert-Raum, der quadratintegrierbaren, messbaren Funktionen $h : \mathcal{X} \to \mathbb{R}$ mit Skalarprodukt $\langle h_1, h_2 \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} h_1(x) h_2(x) \, \mathrm{d}x$ und Norm $\|h\|_{L^2(\mathcal{X})} := \sqrt{\langle h, h \rangle_{L^2(\mathcal{X})}}$. Eine Funktion $h \in L^2(\mathcal{X})$ heißt *normiert,* falls $\|h\|_{L^2(\mathcal{X})} = 1$. Eine Familie von Funktionen $h_k \in L^2(\mathcal{X})$, $k \in \mathbb{N}$ heißt *Orthonormalbasis,* falls alle h_k normiert sind und $\int_{\mathcal{X}} h_k(x) h_l(x) \, \mathrm{d}x = 0$ für $k \neq l$. Für $\mathcal{Z} \subset \mathbb{R}^d$ sei

$$C(\mathcal{Z}) := \{f : \mathcal{Z} \to \mathbb{R} \text{ stetig}\}$$

der Raum der stetigen Funktionen $f : \mathcal{Z} \to \mathbb{R}$. Der Raum der quadratintegrierbaren Folgen sei

$$\ell^2 := \{(a_k)_{k \in \mathbb{N}} | \text{ Für alle } k \in \mathbb{N} \text{ ist } a_k \in \mathbb{R} \text{ und } \sum_{k=1}^{\infty} a_k^2 < \infty\}. \tag{4.36}$$

Satz 4.36 (Satz von Mercer) Ist $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ ein Mercer-Kern und ist $\mathcal{X} \subset \mathbb{R}^d$ beschränkt, so gibt es eine Orthonormalbasis $h_k^\circ \in L^2(\mathcal{X})$ mit stetigen h_k° und $\lambda_k \geq 0$ ($k \in \mathbb{N}$), so dass

$$K(x, x') = \sum_{k=1}^{\infty} \lambda_k h_k^\circ(x) h_k^\circ(x') \qquad x, x' \in \mathcal{X}.$$

Die Konvergenz ist absolut und gleichmäßig in $C(\mathcal{X} \times \mathcal{X})$. Die Werte λ_k, h_k° entsprechen den Eigenwerten und zugehörigen normierten Eigenfunktionen des Operators

$$T_K : L^2(\mathcal{X}) \to L^2(\mathcal{X}), \quad (T_K g)(x) = \int_{\mathcal{X}} K(x, x') g(x') \, \mathrm{d}x'$$

(d. h. $T_K h_k^\circ = \lambda_k h_k^\circ$, $k \in \mathbb{N}$). Die Funktion

$$h(\cdot) := (\sqrt{\lambda_k} \cdot h_k^\circ(\cdot))_{k \in \mathbb{N}} : \mathcal{X} \to \ell^2$$

ist wohldefiniert, stetig und erfüllt

$$K(x, x') = h(x)^T h(x'), \qquad x, x' \in \mathcal{X}.$$

Die im Satz von Mercer definierte Funktion h entspricht genau der eingeführten Einbettungsfunktion. Vor allem in der Anwendung wird diese auch als *feature map* bezeichnet. Die Kenntnis der Funktionen h bzw. h_k° sind bei einem Kern von Interesse, um eine adäquate Wahl hinsichtlich der Struktur der Entscheidungsränder und Entscheidungsregionen treffen zu können.

Für den Gauß-Kern (vgl. Beispiel 4.34(iii)) ergibt sich folgendes Resultat:

Beispiel 4.37 Wegen $\|x - x'\|_2^2 = \|x\|_2^2 + \|x'\|_2^2 - 2x^T x'$ und $\exp(z) = \sum_{j=0}^{\infty} \frac{z^j}{j!}$ gilt:

$$\begin{aligned}
&K_\gamma^{gauss}(x, x') \\
&= \exp(-\gamma \|x - x'\|_2^2) \\
&= \sum_{j=0}^{\infty} \frac{(2\gamma x^T x')^j}{j!} \exp(-\gamma \|x\|_2^2) \exp(-\gamma \|x'\|_2^2) \\
&= \sum_{j=0}^{\infty} (2\gamma)^j \sum_{\sum_{i=1}^d n_i = j} \exp(-\gamma \|x\|_2^2) \frac{x_1^{n_1} \cdot \ldots \cdot x_d^{n_d}}{\sqrt{n_1! \cdot \ldots \cdot n_d!}} \exp(-\gamma \|x'\|_2^2) \frac{(x_1')^{n_1} \cdot \ldots \cdot (x_d')^{n_d}}{\sqrt{n_1! \cdot \ldots \cdot n_d!}}
\end{aligned}$$

Zur Vereinfachung betrachten wir nur den Fall $d = 1$. Dann gilt:

$$\begin{aligned}
K_\gamma^{gauss}(x, x') &= \sum_{j=0}^{\infty} \frac{(2\gamma)^j}{j!} x^j \exp(-\gamma x^2) \cdot (x')^j \exp(-\gamma (x')^2) \\
&= \sum_{j=0}^{\infty} h_j(x) h_j(x')
\end{aligned}$$

mit $h_j(x) = \left(\frac{(2\gamma)^j}{j!}\right)^{1/2} x^j \exp(-\gamma x^2)$. Bei h_j handelt es sich also im Wesentlichen um die Monombasis, jedoch werden die einzelnen Polynome durch den Exponentialterm $\exp(-\gamma x^2)$ für große x abgeschwächt. Anschaulich werden dadurch starke Oszillationen für große x vermieden, welche bei Anpassungen mit Polynomen zu hohen Grades üblicherweise auftreten.

Allerdings bilden die Funktionen $(h_j)_{j \in \mathbb{N}}$ noch keine Orthonormalbasis und entsprechen damit formal nicht den Polynomen h_j° aus dem Satz von Mercer. Die Funktionen h_j° haben die Gestalt

$$h_j^\circ(x) = c_j \exp(-\tilde{\gamma} x^2) \cdot H_{j-1}(a_j x),$$

wobei $a_j, c_j, \tilde{\gamma} \in \mathbb{R}$ geeignete Konstanten und H_j die Hermite-Polynome sind.

Durch die voranstehenden Ergebnisse haben wir nun eine grobe Vorstellung, welchen Funktionen h die jeweiligen K entsprechen. Die Struktur des zur SVM gehörigen Entscheidungsrands kann jedoch auch direkt am Kern K abgelesen werden. In Definition 4.33 wurde festgelegt:

$$\hat{f}_{n,C}^{SVM,K}(x) = \text{sign}(\hat{\delta}_{n,C}^{SVM,K}(x))$$

mit

$$\hat{\delta}_{n,C}^{SVM,K}(x) = \sum_{i=1}^{n} \hat{\alpha}_i Y_i \cdot K(X_i, x) + \hat{\beta}_{0,C} \tag{4.37}$$

$\hat{\delta}_{n,C}^{SVM,K}(x)$ besteht also aus Linearkombinationen der Terme $K(X_i, x)$ mit $\hat{\alpha}_i \neq 0$. Das bedeutet, dass der Entscheidungsrand $\{x \in \mathbb{R}^d : \hat{\delta}_{n,C}^{SVM,K}(x) = 0\}$ wie folgt entsteht: Um

jeden Supportvektor X_i wird ein „Hügel“ in Form der Kernfunktion $x \mapsto K(x, X_i)$ gelegt, wobei die Höhe durch $\hat{\alpha}_i$ beeinflusst wird. Der Entscheidungsrand ist dann die $-\hat{\beta}_{0,C}$-Niveaulinie der aufaddierten Hügel. In Abb. 4.11a, b sind die Niveaulinien von

$$x \mapsto K_2^{poly}(x, X_1), \qquad x \mapsto \frac{1}{2} K_2^{poly}(x, X_1) - \frac{9}{5} K_2^{poly}(x, X_2) \tag{4.38}$$

auf Basis von zwei Trainingspunkten $X_1 = (0{,}7, 0{,}6)^T$, $X_2 = (0{,}2, 0{,}3)^T$ dargestellt; in Abb. 4.11c, d sind die Niveaulinien von

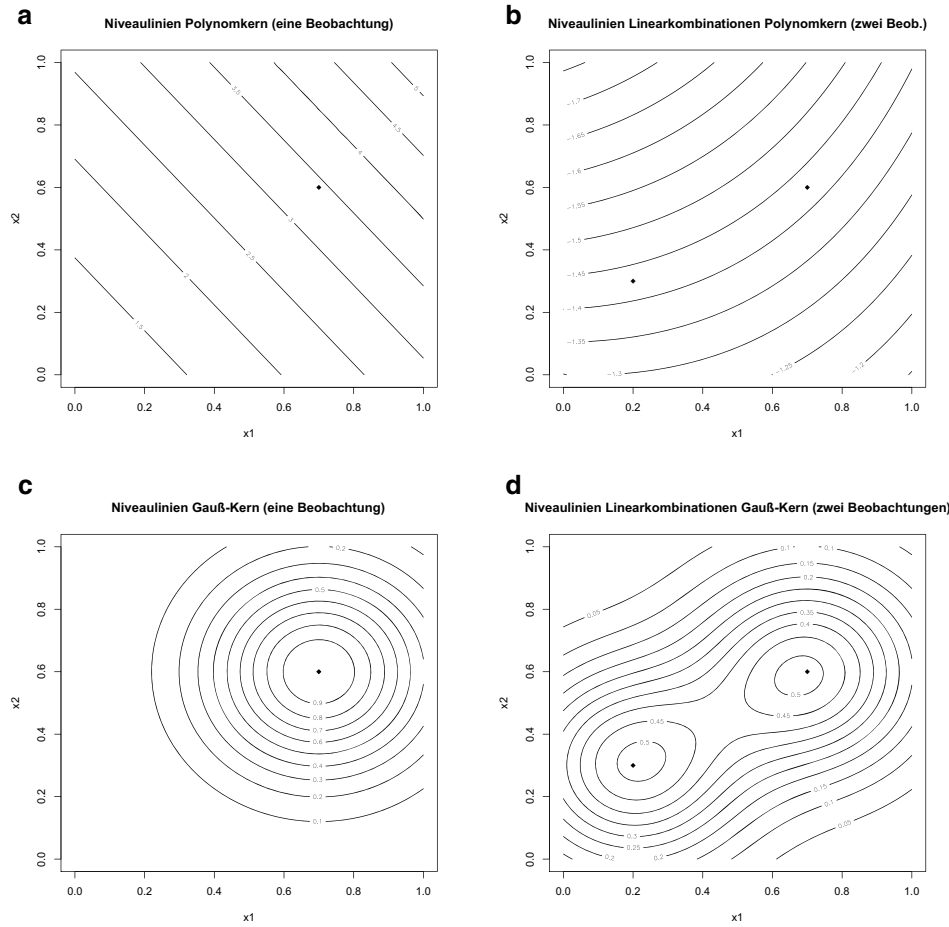


Abb. 4.11 **a**, **b** Niveaulinien von Linearkombinationen des Polynomkerns (vgl. Gl. (4.38)), **c**, **d** Niveaulinien von Linearkombinationen des Gauß-Kerns (unten, vgl. Gl. (4.39))

$$x \mapsto K_{10}^{gauss}(x, X_1), \qquad x \mapsto \frac{1}{2}K_2^{poly}(x, X_1) + \frac{1}{2}K_2^{poly}(x, X_2) \tag{4.39}$$

zu sehen.

Wir zeigen nun ein Beispiel für die geschätzten Entscheidungsränder und den Klassifizierer $\hat{f}_{n,C}^{SVM,K}$ unter Nutzung der nichtlinearen SVM mit Kern K.

Beispiel 4.38 (Anwendung der nichtlinearen SVM) Wir wenden die nichtlineare SVM auf die nicht linear separierbaren Trainingsdaten aus Beispiel 4.28(b), (c) an.

In Abb. 4.12 sind die Resultate für Beispiel 4.28(b) für den Polynomkern mit $p = 2$ und $C \in \{1, 10\}$ zu sehen. Für größeres C ergeben sich weniger Support-Vektoren und eine geringe Breite der „Straße", ansonsten erhält man aber relativ ähnliche Entscheidungsränder. In Abb. 4.13 wurde der SVM-Klassifizierer jeweils mit $C = 1$ und dem Polynomkern mit $p \in \{2, 4\}$ angewandt. Während die Menge der Entscheidungsränder mit dem Polynomkern $p = 2$ noch zu klein ist, um eine sinnvolle Trennung zu erlauben, ermöglicht $p = 4$ bereits eine fast perfekte Trennung der gegebenen Trainingsdaten.

In Abb. 4.14 und 4.15 sind die Resultate für Beispiel 4.28(b) für den Gauß-Kern mit $C = 1$ und $\gamma \in \{0{,}02, 1, 10\}$ zu sehen. Wie in Abb. 4.11 motiviert, entstehen die Entscheidungsränder als Niveaulinien der Kernfunktionen, welche auf jeden Support-Vektor als Hügel gelegt werden. $\gamma = 10$ erzeugt daher jeweils detailliertere Entscheidungsränder als $\gamma = 1$.

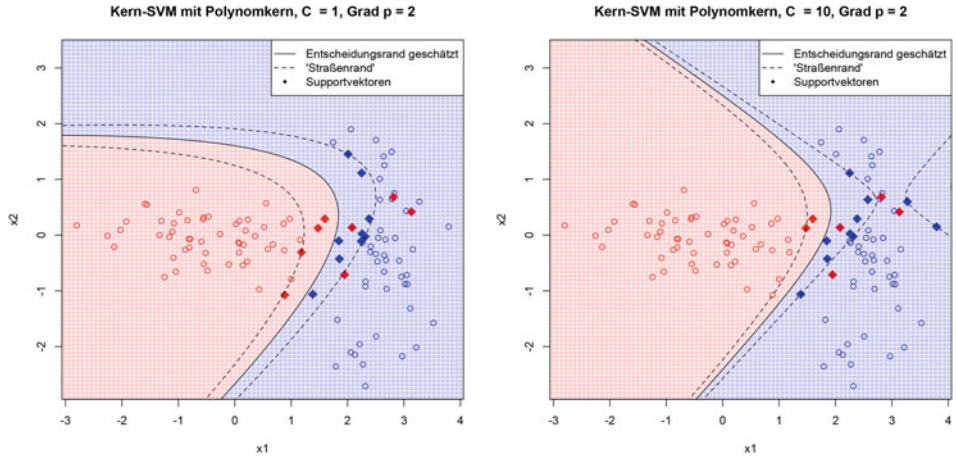


Abb. 4.12 Darstellung der geschätzten Entscheidungsränder des nichtlinearen SVM-Klassifizierers $\hat{f}_{n,C}^{SVM,K_p^{poly}}$ mit Polynomkern auf Basis der Trainingsdaten aus Beispiel 4.28(b) mit $p = 2$ und $C \in \{1, 10\}$. Zusätzlich ist jeweils der „Straßenrand" $\{x \in \mathcal{X} : \hat{\delta}_{n,C}^{SVM,K_p^{poly}}(x) = \pm 1\}$ eingezeichnet

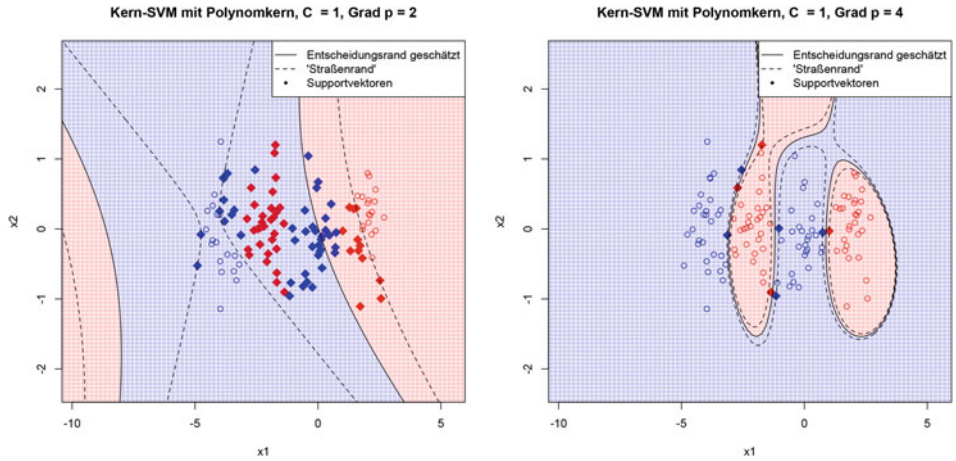


Abb. 4.13 Darstellung der geschätzten Entscheidungsränder des nichtlinearen SVM-Klassifizierers $\hat{f}_{n,C}^{SVM,K_p^{poly}}$ mit Polynomkern auf Basis der Trainingsdaten aus Beispiel 4.28(c) mit $p \in \{2, 4\}$ und $C = 1$. Zusätzlich ist jeweils der „Straßenrand" $\{x \in \mathcal{X} : \hat{\delta}_{n,C}^{SVM,K_p^{poly}}(x) = \pm 1\}$ eingezeichnet

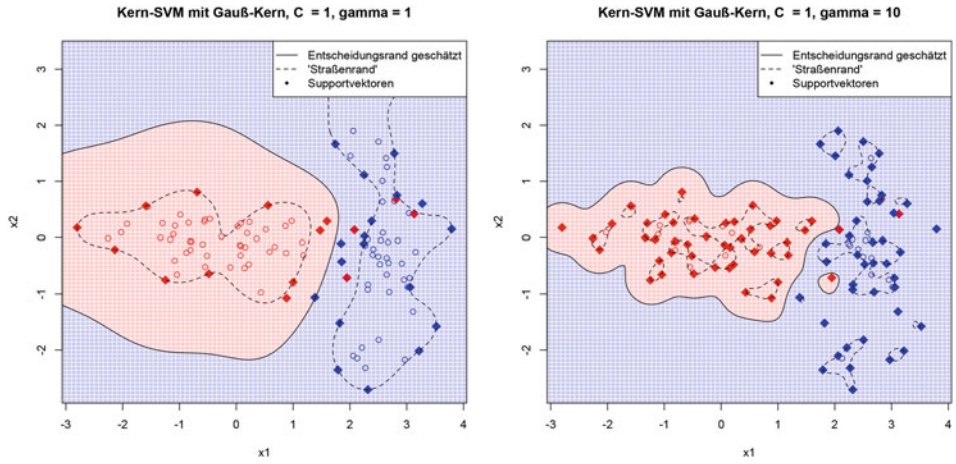


Abb. 4.14 Darstellung der geschätzten Entscheidungsränder des nichtlinearen SVM-Klassifizierers $\hat{f}_{n,C}^{SVM,K_\gamma^{gauss}}$ mit Gauß-Kern auf Basis der Trainingsdaten aus Beispiel 4.28(b) mit $C = 1$ und $\gamma \in \{1, 10\}$. Zusätzlich ist jeweils der „Straßenrand" $\{x \in \mathcal{X} : \hat{\delta}_{n,C}^{SVM,K_\gamma^{gauss}}(x) = \pm 1\}$ eingezeichnet

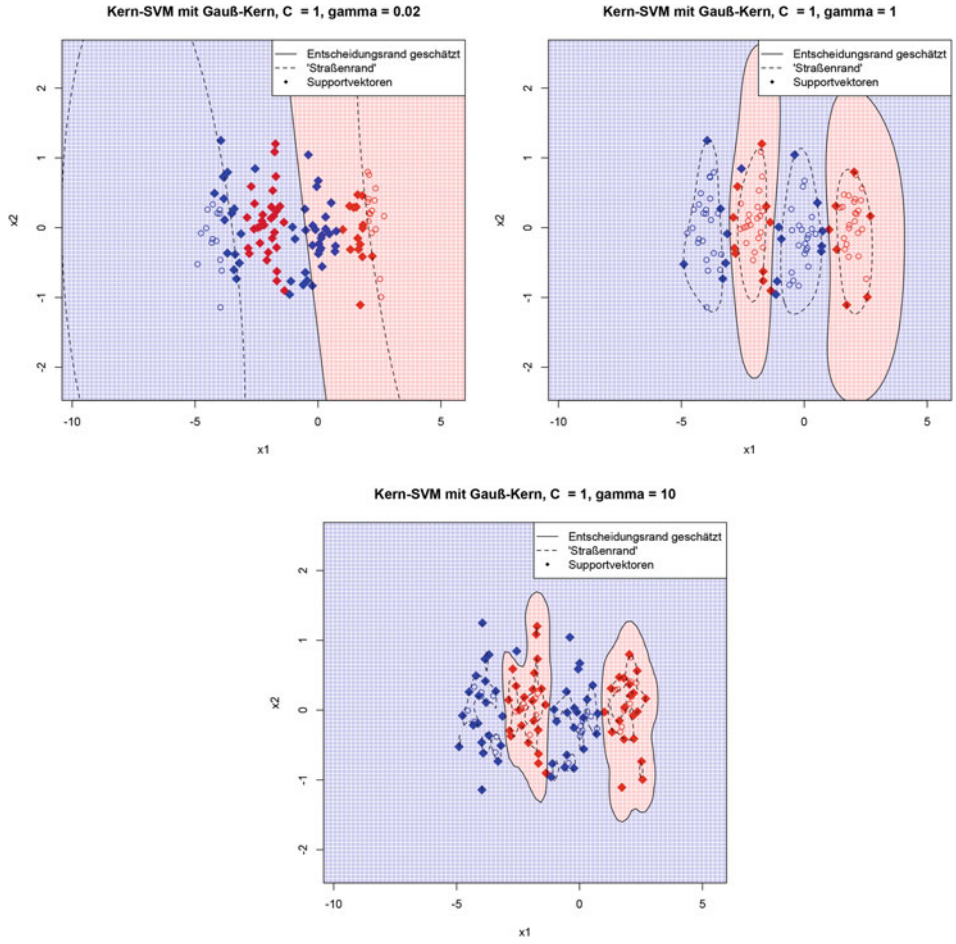


Abb. 4.15 Darstellung der geschätzten Entscheidungsränder des nichtlinearen SVM-Klassifizierers $\hat{f}_{n,C}^{SVM,K_\gamma^{gauss}}$ mit Gauß-Kern auf Basis der Trainingsdaten aus Beispiel 4.28(c) mit $C = 1$ und $\gamma \in \{0{,}02, 1, 10\}$. Zusätzlich ist jeweils der „Straßenrand" $\{x \in \mathcal{X} : \hat{\delta}_{n,C}^{SVM,K_\gamma^{gauss}}(x) = \pm 1\}$ eingezeichnet

Besonders in den Abbildungen zum Gauß-Kern ist ersichtlich, dass der SVM-Klassifizierer sehr stark auf die vorhandenen Trainingsdaten fokussiert ist, besonders wenn γ sehr groß gewählt wird. Entsprechend ist der Klassifizierer auch nicht in Lage, Strukturen „außerhalb" der beobachteten Trainingsdaten sinnvoll abzubilden. Im Gegensatz zum Polynomkern mit niedrigem Grad p ist der Gauß-Kern in der Lage, auch komplexe Strukturen zu erkennen, ein Beispiel ist in Abb. 4.16 gegeben.

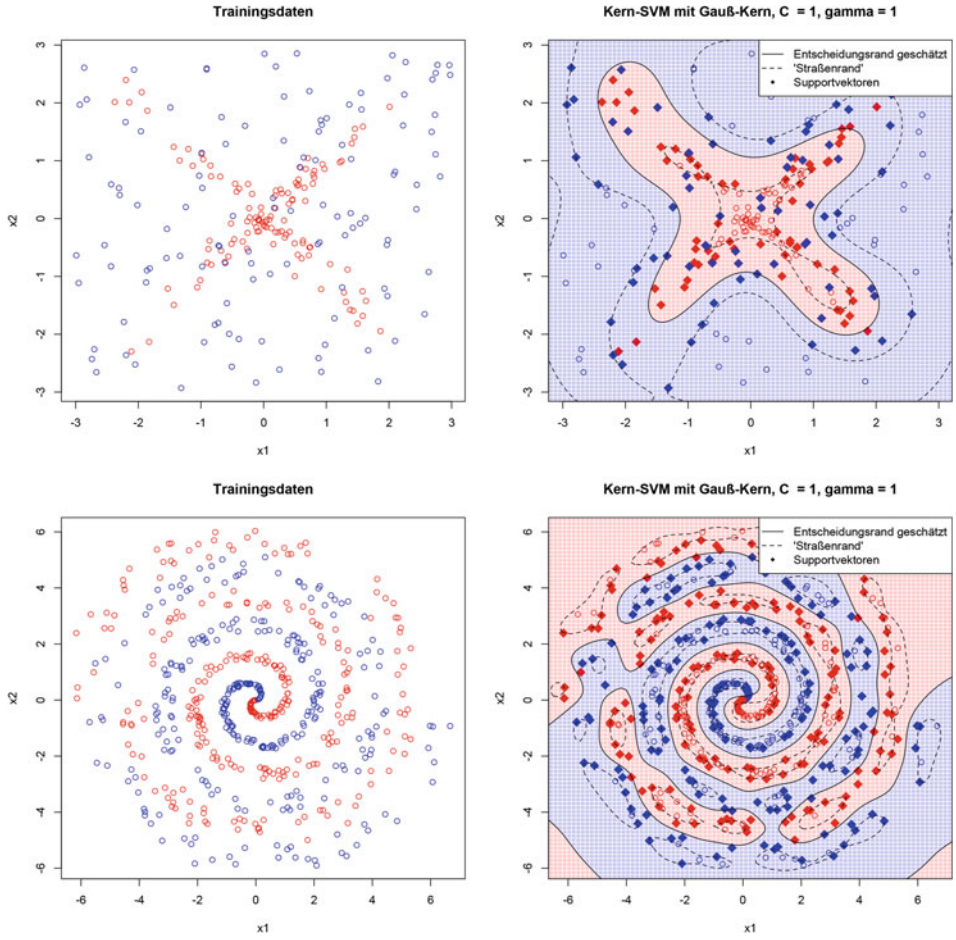


Abb. 4.16 Darstellung der geschätzten Entscheidungsränder des nichtlinearen SVM-Klassifizierers $\hat{f}_{n,C}^{SVM,K_\gamma^{gauss}}$ mit Gauß-Kern auf Basis der jeweils links dargestellten Trainingsdaten

Die Menge aller Funktionen, welche durch Linearkombinationen aus Gl. (4.37) erzeugt werden können, bilden einen Funktionenraum abhängig vom Kern K, welcher im Folgenden eingeführt wird. Dieser Raum ist vor allem für die theoretische Betrachtung der SVM relevant, welche in Abschn. 4.6 besprochen wird. Ein Beweis der hier formulierten Eigenschaften findet sich in [40].

Satz 4.39 (Reproducing Kernel Hilbert Space, RKHS) Sei $\mathcal{X} \subset \mathbb{R}^d$ beschränkt und $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ ein Mercer-Kern. Dann gibt einen eindeutig bestimmten Hilbertraum $\mathcal{H}_K$, genannt *Reproducing Kernel Hilbert Space,* welcher der Abschluss des Raumes

$$\left\{ g : \mathcal{X} \to \mathbb{R}, g(x) = \sum_{i=1}^{n} c_i \cdot K(x^{(i)}, x) : x^{(1)}, \ldots, x^{(n)} \in \mathcal{X}, c_1, \ldots, c_n \in \mathbb{R}, n \in \mathbb{N} \right\}$$

in $\{h : \mathcal{X} \to \mathbb{R} \text{ messbar}\}$ bzgl. des Skalarprodukts

$$\langle g, \tilde{g} \rangle_K := \sum_{i=1}^{n} \sum_{j=1}^{m} a_i b_j K(x^{(i)}, x^{(j)}),$$

mit $g(\cdot) = \sum_{i=1}^{n} a_i K(x^{(i)}, \cdot)$, $\tilde{g}(\cdot) = \sum_{j=1}^{m} b_j K(x^{(j)}, \cdot)$, ist. Die zugehörige Norm ist $\|g\|_K := \sqrt{\langle g, g \rangle_K}$.

$\mathcal{H}_K$ hat die *Reproduktionseigenschaft,* jedes Element $g \in \mathcal{H}_K$ kann erhalten werden durch Skalarprodukte mit K:

$$g(x) = \langle g, K(x, \cdot) \rangle_K, \qquad x \in \mathcal{X},$$

und besitzt eine Darstellung (h die Funktion aus Satz 4.36)

$$g(x) = \sum_{j=1}^{\infty} g_j h_j(x) = G^T h(x), \qquad x \in \mathcal{X}$$

mit geeigneten Koeffizienten $g_j \in \mathbb{R}$ ($j \in \mathbb{N}$) und $G := (g_j)_{j \in \mathbb{N}}$. Es gilt die Isometrie

$$\|g\|_K^2 = \sum_{j=1}^{\infty} g_j^2.$$

Bei Anwendung der nichtlinearen SVM mit Kern K ist $\hat{\delta}_{n,C}^{SVM,K} - \hat{\beta}_{0,C}$ ein Element von $\mathcal{H}_K$, d. h., es gilt stets $\hat{\delta}_{n,C}^{SVM,K} - \hat{\beta}_{0,C} \in \mathcal{H}_K$.

4.5 Berechnung von SVM

Das Optimierungsproblem, welches für eine nichtlineare SVM mit Kern K in Definition 4.33 gelöst werden muss, ist ein quadratisches Programm und konvex. Es besitzt daher mindestens ein globales Minimum $\hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n)^T \in \mathbb{R}^n$. Ist die Matrix Q^K sogar positiv definit, so ist das Minimum eindeutig. Zur Lösung des Optimierungsproblems nutzt man ähnlich wie beim Lasso-Schätzer eine iterative Prozedur, welche das ursprüngliche Problem durch koordinatenweise Betrachtung in einfache Teilprobleme zerlegt. In [15, Theorem 4] wird gezeigt, dass das hier vorgestellte Verfahren gegen ein globales Minimum konvergiert. Zur Beschreibung orientieren wir uns an [14] und schreiben kurz $Q = Q^K$. Die zu minimierende Funktion ist dann

$$Z(\alpha) = \frac{1}{2} \alpha^T Q \alpha - \mathbb{1}^T \alpha.$$

Gestartet wird das Verfahren mit dem Vektor $\alpha^0 = (0, \ldots, 0)^T \in \mathbb{R}^n$ (dieser erfüllt die Nebenbedingungen). Im $(k+1)$-ten Iterationsschritt wird wie folgt verfahren:

(S1) Wähle zwei Indizes $\{i, j\} \subset \{1, \ldots, n\}$ mit $i < j$, genannt *working set* und minimiere

$$(\alpha_i, \alpha_j) \mapsto Z(\alpha_1^k, \ldots, \alpha_{i-1}^k, \alpha_i, \alpha_{i+1}^k, \ldots, \alpha_{j-1}^k, \alpha_j, \alpha_{j+1}^k, \ldots, \alpha_n^k) \tag{4.40}$$

nur bezüglich (α_i, α_j) und ohne Beachtung eines Teils der Nebenbedingungen. Die Minimierer bezeichnen wir mit $(\tilde{\alpha}_i, \tilde{\alpha}_j)$. Die Auswahl von $\{i, j\}$ erfolgt dabei so, dass bei der Optimierung ein möglichst großer Schritt in Richtung des globalen Minimums absolviert wird.

(S2) Für $m \in \{1, \ldots, n\} \backslash \{i, j\}$ setze $\alpha_m^{k+1} := \alpha_m^k$. Wähle geeignete Modifikationen $(\alpha_i^{k+1}, \alpha_j^{k+1})$ von $(\tilde{\alpha}_i, \tilde{\alpha}_j)$, so dass $\alpha^{k+1} = (\alpha_1^{k+1}, \ldots, \alpha_n^{k+1})^T$ wieder alle Nebenbedingungen erfüllt.

Bemerkungen

- Für die Wahl der Indizes $\{i, j\}$ in (S1) gibt es eine Vielzahl von Vorschlägen in der Literatur, vgl. [15]. Man nennt diese Auswahl auch *working set selection.*
- Ein Vorteil von (S1) ist, dass die Matrix $Q = (K(X_i, X_j))_{i,j=1,\ldots,n}$ nicht vollständig berechnet und abgespeichert werden muss. Außerdem erwartet man bei genügend großem C in den Nebenbedingungen, dass nur wenige $\hat{\alpha}_i$ der Lösung von null verschieden sind und daher nicht zu viele Iterationen nötig sind.

Im Folgenden erläutern wir die Schritte (S1) und (S2) detaillierter. Zunächst untersuchen wir das auf zwei Parameter (α_i, α_j) reduzierte Optimierungsproblem der SVM und leiten daraus die nötige Theorie für den Schritt (S1) ab. Für einen Vektor $v = (v_i)_{i=1,\ldots,n}$ und eine Menge $N \subset \{1, \ldots, n\}$ nutzen wir die Bezeichnung $v_N := (v_j)_{j \in N} \in \mathbb{R}^{\#N}$. Wir schreiben const.$(w)$ für Terme, welche nur von w und festen Konstanten abhängen.

Lemma 4.40 (Auf zwei Indizes reduziertes Optimierungsproblem) Seien $i, j \in \{1, \ldots, n\}$ und $N := \{1, \ldots, n\} \backslash \{i, j\}$. Dann gilt für $\alpha \in \mathbb{R}^n$:

$$Z(\alpha) = \frac{1}{2} \begin{pmatrix} \alpha_i & \alpha_j \end{pmatrix} \begin{pmatrix} Q_{ii} & Q_{ij} \\ Q_{ji} & Q_{jj} \end{pmatrix} \begin{pmatrix} \alpha_i \\ \alpha_j \end{pmatrix} + (Q_{i,N}\alpha_N - 1)\alpha_i + (Q_{j,N}\alpha_N - 1)\alpha_j + const.(\alpha_N),$$

und die Nebenbedingungen lauten

$$Y_i\alpha_i + Y_j\alpha_j = -\mathbb{Y}_N^T\alpha_N, \tag{4.41}$$

$$0 \le \alpha_i \le C, 0 \le \alpha_j \le C. \tag{4.42}$$

Vom k-ten Iterationsschritt sei $\alpha^k \in \mathbb{R}^n$ gegeben. Im nächsten Schritt ermitteln wir die Minimierer der Zielfunktion für beliebig vorgegebene i, j unter einem Teil der Nebenbedingungen. Dies entspricht der theoretischen Grundlage für (S1):

Lemma 4.41 (Minimierer von Z unter einem Teil der Nebenbedingungen) Sei $\tilde{\alpha} \in \mathbb{R}^n$ mit $\tilde{\alpha}_N = \alpha_N^k$ und

$$\begin{aligned} \tilde{\alpha}_i &= \alpha_i^k + d_i = \alpha_i^k + Y_i \frac{b_{ij}}{a_{ij}}, \\ \tilde{\alpha}_j &= \alpha_j^k + d_j = \alpha_j^k - Y_j \frac{b_{ij}}{a_{ij}}, \end{aligned} \tag{4.43}$$

wobei

$$\begin{aligned} a_{ij} &:= K(X_i, X_i) + K(X_j, X_j) - 2K(X_i, X_j), \\ b_{ij} &:= -Y_i \nabla_i f(\alpha^k) + Y_j \nabla_j f(\alpha^k). \end{aligned} \tag{4.44}$$

Dann ist $\tilde{\alpha}$ ein Minimierer der Zielfunktion (4.40) unter der Nebenbedingung (4.41), und es gilt

$$Z(\tilde{\alpha}) = -\frac{1}{2}\frac{b_{ij}^2}{a_{ij}} + const.(\alpha_N^k). \tag{4.45}$$

Beweis Sei zunächst $\tilde{\alpha} \in \mathbb{R}^n$ beliebig mit $\tilde{\alpha}_N = \alpha_N^k$. Sei $d_l := \tilde{\alpha}_l - \alpha_l^k$ $(l = i, j)$. Da $\mathbb{Y}^T \alpha^k = 0$ aus dem vorherigen Optimierungsschritt erfüllt ist, gilt

$$Y_i \tilde{\alpha}_i^k + Y_j \tilde{\alpha}_j^k = -\mathbb{Y}_N^T \alpha_N^k.$$

Die Gültigkeit der Nebenbedingung Gl. (4.41) für $\tilde{\alpha}$ ist daher äquivalent zu

$$Y_i d_i + Y_j d_j = 0 \quad \Longleftrightarrow \quad Y_i d_i = -Y_j d_j. \tag{4.46}$$

Weiter ist

$$\nabla_i Z(\tilde{\alpha}) = Q_{ii}\tilde{\alpha}_i + Q_{ij}\tilde{\alpha}_j + (Q_{i,N}\alpha_N^k - 1). \tag{4.47}$$

Einsetzen von (4.46) und (4.47) in Z liefert:

$$\begin{aligned} Z(\tilde{\alpha}) &= \frac{1}{2}\begin{pmatrix} d_i & d_j \end{pmatrix}\begin{pmatrix} Q_{ii} & Q_{ij} \\ Q_{ji} & Q_{jj} \end{pmatrix}\begin{pmatrix} d_i \\ d_j \end{pmatrix} + \begin{pmatrix} \nabla_i Z(\alpha^k) & \nabla_j Z(\alpha^k) \end{pmatrix}\begin{pmatrix} d_i \\ d_j \end{pmatrix} + const.(\alpha_N^k) \\ &= \frac{1}{2} a_{ij} d_j^2 + b_{ij} Y_j d_j + const.(\alpha_N^k) \end{aligned}$$

$Z(\tilde{\alpha})$ ist eine quadratische Funktion in d_j und wird minimal für $d_j = -\frac{Y_j b_{ij}}{a_{ij}}$. Nutzung von Gl. (4.46) liefert die Behauptungen aus Gl. (4.43) und (4.45).

Die Minimierer aus Gl. (4.43) erfüllen nicht notwendig die zweite Nebenbedingung $0 \leq \tilde{\alpha}_i \leq C, 0 \leq \tilde{\alpha}_j \leq C$. Mittels einer einfachen Heuristik werden diese unter Beibehaltung der ersten Nebenbedingung zurück in das Rechteck $[0, C]^2$ verschoben. Dies entspricht der Grundlage für den Schritt (S2):

Bemerkung 4.42 Ist $Y_i \neq Y_j$ (z. B. $Y_i = -1$, $Y_j = 1$), so ist die erste Nebenbedingung gemäß Gl. (4.46) äquivalent zu

$$\tilde{\alpha}_i - \tilde{\alpha}_j = \alpha_i^k - \alpha_j^k,$$

d. h., $(\tilde{\alpha}_i, \tilde{\alpha}_j)$ erfüllt die erste Nebenbedingung für alle Werte auf einer Geraden. Wegen $0 \leq \alpha_i^k, \alpha_j^k \leq C$ verläuft diese Gerade durch das Rechteck $[0, C]^2$. Der Punkt $(\tilde{\alpha}_i, \tilde{\alpha}_j)$ wird nun so lange auf der Geraden verschoben, bis er das erste Mal in das Rechteck eintritt; dies liefert den finalen Vektor $\tilde{\alpha}$ des Iterationsschritts. Für $Y_i = Y_j$ verfahre analog.

4.5.1 Stoppkriterium

Da eine Konvergenz des Iterationsverfahrens gegen einen globalen Minimierer $\hat{\alpha}$ möglicherweise erst nach langer Zeit eintritt, benötigen wir noch einen Indikator, wann sich α^k nah genug an $\hat{\alpha}$ befindet. Anschaulich überprüfen wir dies anhand der Größe des Gradienten von $Z(\alpha^k)$, welcher die Nebenbedingungen geeignet berücksichtigt. Das folgende Lemma formalisiert diese Idee:

Lemma 4.43 Sei $\hat{\alpha} \in [0, C]^n$ mit $\mathbb{Y}^T \hat{\alpha} = 0$. Dann ist $\hat{\alpha}$ genau dann eine Lösung des Optimierungsproblems aus Definition 4.33, falls

$$m(\hat{\alpha}) \leq M(\hat{\alpha}) \quad bzw. \quad m(\hat{\alpha}) - M(\hat{\alpha}) \leq 0, \tag{4.48}$$

wobei

$$m(\hat{\alpha}) = \max_{i \in I_{up}(\hat{\alpha})} -Y_i \nabla_i Z(\hat{\alpha}), \qquad M(\hat{\alpha}) = \min_{i \in I_{low}(\hat{\alpha})} -Y_i \nabla_i Z(\hat{\alpha})$$

und

$$I_{up}(\alpha) = \{t \in \{1, \ldots, n\} : \alpha_t < C, Y_t = 1 \; oder \; \alpha_t > 0, Y_t = -1\},$$
$$I_{low}(\alpha) = \{t \in \{1, \ldots, n\} : \alpha_t < C, Y_t = -1 \; oder \; \alpha_t > 0, Y_t = 1\}.$$

Beweis „⇒“: Sei $\hat{\alpha}$ eine Lösung des Optimierungsproblems der nichtlinearen SVM mit Kern K. Sei $K(x, x') = h(x)^T h(x')$ (vgl. Satz 4.36) und definiere $\hat{\lambda}_i := Y_i(h(X_i)^T \hat{\beta} + \hat{\beta}_0) + \hat{\xi}_i - 1$. Aus den Optimalitätsbedingungen Gl. (4.25), (4.27), (4.28) (mit $h(X_i)$ statt X_i) folgt:

$$\hat{\lambda}_i \geq 0, \quad \hat{\lambda}_i \hat{\alpha}_i = 0, \quad \hat{\xi}_i (C - \hat{\alpha}_i) = 0, \quad \hat{\xi}_i \geq 0 \quad (i = 1, \ldots, n) \tag{4.49}$$

Es ist $\nabla Z(\hat{\alpha}) = Q\hat{\alpha} - \mathbb{1}$ und daher mit der Optimalitätsbedingung Gl. (4.23):

$$\begin{aligned}\nabla_i Z(\hat{\alpha}) &= \sum_{j=1}^{n} Q_{ij}\hat{\alpha}_j - 1 = Y_i h(X_i)^T \sum_{j=1}^{n} Y_j \hat{\alpha}_j h(X_j) - 1 \\ &= Y_i h(X_i)^T \hat{\beta} - 1 = \hat{\lambda}_i - \hat{\xi}_i - \hat{\beta}_0 Y_i\end{aligned}$$

Mit Gl. (4.49) folgt

$$\nabla_i Z(\hat{\alpha}) + \hat{\beta}_0 Y_i = \hat{\lambda}_i - \hat{\xi}_i \begin{cases} \geq 0, & \hat{\alpha}_i < C \\ \leq 0, & \hat{\alpha}_i > 0. \end{cases}$$

Durch Multiplikation mit Y_i und unter Nutzung von $Y_i^2 = 1$ erhalten wir die äquivalente Aussage

$$Y_i \nabla_i Z(\hat{\alpha}) + \hat{\beta}_0 \begin{cases} \geq 0, & \hat{\alpha}_i < C, Y_i = +1 \; oder \; \hat{\alpha}_i > 0, Y_i = -1, \\ \leq 0, & \hat{\alpha}_i > 0, Y_i = +1 \; oder \; \hat{\alpha}_i < C, Y_i = -1. \end{cases}$$

Dies ist äquivalent zu

$$m(\hat{\alpha}) \leq \hat{\beta}_0 \leq M(\hat{\alpha}).$$

„⇐“: Sei nun umgekehrt $\hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n)^T \in [0, C]^n$ so, dass

$$m(\hat{\alpha}) \leq M(\hat{\alpha}). \tag{4.50}$$

Im Folgenden definieren wir die fehlenden Parameter des primalen bzw. dualen Optimierungsproblems geeignet und zeigen, dass diese die Optimalitätsbedingungen erfüllen. Aufgrund von Satz 2.33 ist $\hat{\alpha}$ dann eine Lösung des Optimierungsproblems aus Definition 4.33.

Definiere $\hat{\beta} := \sum_{j=1}^{n} h(X_j)^T \hat{\alpha}_j Y_j$ (damit wird Gl. (4.23) erfüllt). Wegen Gl. (4.50) gibt es $\hat{\beta}_0 \in \mathbb{R}$ mit

$$m(\hat{\alpha}) \leq \hat{\beta}_0 \leq M(\hat{\alpha}),$$

d. h.

$$\begin{aligned} &für\ alle\ i \in I_{up}(\hat{\alpha}): \; -Y_i \nabla_i Z(\hat{\alpha}) \leq \hat{\beta}_0, \\ &für\ alle\ i \in I_{low}(\hat{\alpha}): \; -Y_i \nabla_i Z(\hat{\alpha}) \geq \hat{\beta}_0, \end{aligned}$$

bzw. unter Nutzung von $\nabla_i Z(\hat{\alpha}) = Y_i h(X_i)^T \hat{\beta} - 1$:

$$\begin{aligned} &für\ alle\ i \in I_{up}(\hat{\alpha}): \; Y_i \leq \hat{\beta}_0 + h(X_i)^T \hat{\beta}, \\ &für\ alle\ i \in I_{low}(\hat{\alpha}): \; Y_i \geq \hat{\beta}_0 + h(X_i)^T \hat{\beta}. \end{aligned}$$

Definiere weiter

$$\hat{\xi}_i := \begin{cases} 1 - Y_i(h(X_i)^T\hat{\beta} + \hat{\beta}_0), & \hat{\alpha}_i = C, \\ 0, & \hat{\alpha}_i < C, \end{cases},$$

$$\hat{\lambda}_i := \begin{cases} Y_i(h(X_i)^T\hat{\beta} + \hat{\beta}_0) + \hat{\xi}_i - 1, & \hat{\alpha}_i > 0, \\ 0, & \hat{\alpha}_i = 0 \end{cases} \qquad (i = 1, \ldots, n).$$

Dann gilt für alle $i = 1, \ldots, n$: $\hat{\xi}_i(C - \hat{\alpha}_i) = 0$ und $\hat{\lambda}_i\hat{\alpha}_i = 0$.

Sei nun $i \in \{1, \ldots, n\}$ fest gewählt. Ist $\hat{\alpha}_i < C$, so ist $\hat{\xi}_i = 0$; ist hingegen $\hat{\alpha}_i = C > 0$, so gilt entweder $Y_i = 1$ (und damit $i \in I_{low}(\hat{\alpha})$, d. h. $1 = Y_i \geq \hat{\beta}_0 + h(X_i)^T\hat{\beta}$, d. h. $\hat{\xi}_i \geq 0$), oder $Y_i = -1$ (und damit $i \in I_{up}(\hat{\alpha})$, d. h. $-1 = Y_i \leq \hat{\beta}_0 + h(X_i)^T\hat{\beta}$, d. h. $\hat{\xi}_i \geq 0$). In jedem Fall folgt $\hat{\xi}_i \geq 0$. Analog folgt für $\hat{\lambda}_i \geq 0$ entlang dieser Fallunterscheidungen. Damit sind auch die Optimalitätsbedingungen Gl. (4.25), (4.27), (4.28) erfüllt.

4.5.2 Das komplette Verfahren

Mit den eben formulierten Lemmata können wir den SVM-Algorithmus berechnen. Ziel ist es, die Abbruchbedingung (4.48) zu erreichen. Da diese aufgrund numerischer Gegebenheiten und evtl. langsamer Konvergenz nicht exakt möglich ist, geben wir stattdessen eine Toleranzgrenze $\varepsilon > 0$ (z. B. $\varepsilon = 10^{-6}$) vor. Gilt

$$m(\alpha) - M(\alpha) \leq \varepsilon,$$

wird das Iterationsverfahren abgebrochen. Die gesamte Berechnungsroutine für $\hat{f}_{n,C}^{SVM,K}$ lautet nun gemäß folgender Definition:

Definition 4.44 (Verfahren zur Bestimmung des SVM-Klassifizierers)
Sei $\alpha^0 = (0, \ldots, 0)^T \in \mathbb{R}^n$. Für $k = 0, 1, 2, 3, \ldots$, wiederhole:

(i) Falls $m(\alpha^k) - M(\alpha^k) \leq \varepsilon$, Abbruch.
(ii) *Working set selection:* Bestimme zwei Indizes i, j wie folgt:
- Sei $i \in \arg\max_{t \in I_{up}(\alpha^k)}\{-Y_t\nabla_t Z(\alpha^k)\}$.
- Sei $j \in \arg\min_{t \in I_{low}(\alpha^k) \text{ und } -Y_t\nabla_t Z(\alpha^k) < -Y_i\nabla_i Z(\alpha^k)} \left\{-\frac{b_{it}^2}{a_{it}}\right\}$ (vgl. (4.44)).

Hierbei ist $\nabla_i Z(\alpha^k) = \sum_{j=1}^n Q_{ij}\alpha_j^k - 1$.

(iii) *Bestimmung der Minimierer von Z ohne die Nebenbedingung* $0 \le \tilde{\alpha}_i, \tilde{\alpha}_j \le C$, vgl. Lemma 4.41: Sei

$$\tilde{\alpha}_i = \alpha_i^k + Y_i \frac{b_{ij}}{a_{ij}}, \qquad \tilde{\alpha}_j = \alpha_j^k - Y_j \frac{b_{ij}}{a_{ij}}.$$

(iv) *Anpassung an die Nebenbedingungen* $0 \le \tilde{\alpha}_i, \tilde{\alpha}_j \le C$, vgl. Bemerkung 4.42:
Falls $Y_i \neq Y_j$:
- Ist $\tilde{\alpha}_j > C$, so setze $\tilde{\alpha}_j = C$ und $\tilde{\alpha}_i = C - (\alpha_i^k - \alpha_j^k)$.
- Ist $\tilde{\alpha}_j < 0$, so setze $\tilde{\alpha}_j = 0$ und $\tilde{\alpha}_i = \alpha_i^k - \alpha_j^k$.

Analoges Vorgehen in den Fällen $\tilde{\alpha}_i > C, \tilde{\alpha}_i < 0$ und mit $Y_i = Y_j$.

(v) Setze $\alpha_i^{k+1} = \tilde{\alpha}_i, \alpha_j^{k+1} = \tilde{\alpha}_j$ und $\alpha_l^{k+1} = \alpha_l^k$ für $l \in \{1, \ldots, n\} \backslash \{i, j\}$.

Dann ist α^k eine Näherung für den Minimierer $\hat{\alpha}$ aus Definition 4.33. ♦

Bemerkung Schritt (ii) *working set selection* ist eine ‚gierige' Auswahlstrategie. Es wird das i ausgewählt, welches zum maximalen Wert von $m(\alpha^k)$ geführt hat. Der Index j wird so gewählt, dass der Funktionswert von Z durch Optimierung entlang $\tilde{\alpha}_i, \tilde{\alpha}_j$ den kleinstmöglichen Wert im aktuellen Iterationsschritt erreicht, siehe Gl. (4.45). Das j wird dabei allerdings nur aus den Indizes gewählt, welche dafür sorgen, dass $M(\alpha^k) < m(\alpha^k)$ gilt, d. h. eine echte Verkleinerung von M gegenüber m eintritt.

4.6 Theoretische Resultate zur SVM

Um den SVM-Algorithmus theoretisch behandeln zu können, benötigen wir neben der ursprünglichen und der zugehörigen dualen Formulierung eine dritte Formulierung des Optimierungsproblems. Bisher gilt für den Klassifizierer aus Definition 4.33:

$$\hat{f}_{n,C}^{SVM,K} = \text{sign}(\hat{\delta}_{n,C}^{SVM,K}),$$

wobei $\hat{\delta}_{n,C}^{SVM,K}$ eine *lineare Funktion* der Minimierer $\hat{\alpha}_i$ ist. Unser Ziel ist es, das Problem so umzuformulieren, dass $\hat{\delta}_{n,C}^{SVM,K}$ *direkt* ein Minimierer ist und nicht aus einer Funktion der Minimierer entsteht. Idealerweise erhalten wir eine Struktur wie in Abschn. 3.3, Bemerkung 3.8, und können dann Eigenschaften von $\hat{\delta}_{n,C}^{SVM,K}$ direkt auf $\hat{f}_{n,C}^{SVM,K}$ übertragen.

4.6.1 Dritte Formulierung des Optimierungsproblems

Sei $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ ein Mercer-Kern und $\mathcal{X} \subset \mathbb{R}^d$ kompakt. Sei $h : \mathcal{X} \to \mathbb{R}^\infty$ aus Satz 4.36 die Funktion mit $K(x, x') = h(x)^T h(x')$. Wir leiten die neue Formulierung aus dem primalen Problem der Definition 4.26 her, wobei X_i durch $h(X_i)$ ersetzt wird und wir $G = (g_j)_{j \in \mathbb{N}} \in \ell^2$ anstelle von β verwenden, vgl. Gl. (4.36). Eine Minimierung über alle

möglichen Koeffizienten $G \in \mathbb{R}^\infty$ ist nicht sinnvoll, da für $G \notin \ell^2$ gilt: $\|G\|_2^2 = \sum_{j=1}^\infty g_j^2 = \infty$.

- *Schritt 1: Adaption von Definition 4.26.* Seien $\hat{G}_C, \hat{\beta}_{0,C}, \hat{\xi}$ Minimierer von

$$\min_{G \in \ell^2, \beta_0 \in \mathbb{R}, \xi \in \mathbb{R}^n} \frac{1}{2}\|G\|_2^2 + C \sum_{i=1}^n \xi_i$$
$$\text{unter NB} \quad \forall i = 1, \ldots, n: \quad Y_i(G^T h(X_i) + \beta_0) \geq 1 - \xi_i, \quad \xi_i \geq 0,$$

  dann hat der SVM-Klassifizierer die Form

$$\hat{f}_{n,C}^{SVM,K}(x) = \text{sign}(\hat{\delta}_{n,C}^{SVM,K}(x)), \qquad \hat{\delta}_{n,C}^{SVM,K}(x) = \hat{G}_C^T h(x) + \hat{\beta}_{0,C}.$$

- *Schritt 2: Übergang zur Minimierung über $\mathcal{H}_K$.* Da jede Funktion $g \in \mathcal{H}_K$ eine Darstellung $g = G^T h$ besitzt, minimieren wir mit $G \in \ell^2$ über alle Funktionen $g \in \mathcal{H}_K$. Die Norm $\|G\|_2^2 = \sum_{j=1}^\infty g_j^2 = \|g\|_K^2$ geht wegen der Isometrieaussage in Satz 4.39 in die Norm von g in $\mathcal{H}_K$ über. Seien $\hat{g}, \hat{\beta}_{0,C}, \hat{\xi}$ Minimierer von

$$\min_{g \in \mathcal{H}_K, \beta_0 \in \mathbb{R}, \xi \in \mathbb{R}^n} \frac{1}{2}\|g\|_K^2 + C \sum_{i=1}^n \xi_i$$
$$\text{unter NB} \quad \forall i = 1, \ldots, n: \quad Y_i(g(X_i) + \beta_0) \geq 1 - \xi_i, \quad \xi_i \geq 0. \tag{4.51}$$

  dann hat der SVM-Klassifizierer die Form

$$\hat{f}_{n,C}^{SVM,K}(x) = \text{sign}(\hat{\delta}_{n,C}^{SVM,K}(x)), \qquad \hat{\delta}_{n,C}^{SVM,K}(x) = \hat{g}(x) + \hat{\beta}_{0,C}$$

- *Schritt 3: Elimination von ξ.* Für jedes feste $g \in \mathcal{H}_K$ und $\beta_0 \in \mathbb{R}$ minimiert $\hat{\xi}(g, \beta_0) = (\hat{\xi}_i(g, \beta_0))_{i=1,\ldots,n}$ mit

$$\hat{\xi}_i(g, \beta_0) = \begin{cases} 0, & Y_i(g(X_i) + \beta_0) \geq 1, \\ 1 - Y_i(g(X_i) + \beta_0), & Y_i(g(X_i) + \beta_0) < 1 \end{cases} = (1 - Y_i(g(X_i) + \beta_0))^+$$

  den Ausdruck Gl. (4.51) in $\xi \in \mathbb{R}^n$. Einsetzen dieses Resultats in Gl. (4.51) und Multiplikation mit $\frac{1}{nC} > 0$ sowie die Substitution $\lambda := \frac{1}{2nC}$ liefert das folgende reduzierte Problem:

$$\min_{g \in \mathcal{H}_K, \beta_0 \in \mathbb{R}} \left\{ \frac{1}{n} \sum_{i=1}^n (1 - Y_i(g(X_i) + \beta_0))^+ + \lambda \cdot \|g\|_K^2 \right\}$$

Damit können wir das Optimierungsproblem der SVM in der in Kap. 1 eingeführten Schreibweise als Minimierer eines empirischen Risikos mit zusätzlichem Bestrafungsterm schreiben, *und* der Minimierer entspricht direkt $\hat{\delta}_{n,C}^{SVM,K}$.

Lemma 4.45 (SVM-Klassifizierer, Formulierung mit Bestrafungsterm) Sei $\lambda > 0$ und C so gewählt, dass $\lambda = \frac{1}{2nC}$. Sei

$$\mathcal{H}_K^b := \{\delta(x) = \delta_0(x) + b : \delta_0 \in \mathcal{H}_K, b \in \mathbb{R}\}.$$

Definiere $\tilde{L}(y, s) := (1 - y \cdot s)^+$, die sogenannte *Hinge-Verlustfunktion.* Sei

$$J(\delta) := \|\delta_0\|_K^2.$$

Dann ist

$$\hat{\delta}_{n,C}^{SVM,K} :\in \operatorname*{arg\,min}_{\delta \in \mathcal{H}_K^b} \left\{ \frac{1}{n} \sum_{i=1}^{n} \tilde{L}(Y_i, \delta(X_i)) + \lambda \cdot J(\delta) \right\} \tag{4.52}$$

sowie $\hat{f}_{n,C}^{SVM,K}(x) = \operatorname{sign}(\hat{\delta}_{n,C}^{SVM,K}(x))$.

Bemerkung Laut der Darstellung in Definition 4.33 gilt: $\hat{\delta}_{n,C}^{SVM,K} \in \mathcal{C}$ mit

$$\mathcal{C} := \{\delta : \mathbb{R}^d \to \mathbb{R} \mid \delta(x) = \sum_{i=1}^{n} a_i K(X_i, x) + b \text{ mit } a_1, \ldots, a_n \in \mathbb{R}, b \in \mathbb{R}\} \subset \mathcal{H}_K^b.$$

Das ist bemerkenswert, weil von $\hat{\delta}_{n,C}^{K} \in \mathcal{H}_K^b$ zunächst nur die Form

$$\hat{f}_n(x) = \sum_{i=1}^{\infty} a_i K(x_i, x) + b$$

mit geeigneten $x_i \in \mathbb{R}^d$, $b, a_i \in \mathbb{R}$ ($i \in \mathbb{N}$) erwartet wird, d. h. eine Summe mit unendlich vielen Summanden. Wir geben einen kurzen heuristischen Beweis, wie auch ohne die vorherige Formulierung aus Definition 4.33 anhand von Gl. (4.52) eingesehen werden kann, dass sogar $\hat{\delta}_{n,C}^{SVM,K} \in \mathcal{C}$ gilt: Sei $\delta(x) = b + \rho_1(x) + \rho_2(x)$, wobei $\rho_1(x) = \sum_{i=1}^{n} a_i K(X_i, x)$ und $\rho_2 \in \mathcal{C}^\perp := \{\rho \in \mathcal{H}_K : \forall g \in \mathcal{C} : \langle \rho, g \rangle_K = 0\}$ ($\mathcal{C}^\perp$ ist das *orthogonale Komplement* von $\mathcal{C}$ in $\mathcal{H}_K$ bzgl. des Skalarprodukts $\langle \cdot, \cdot \rangle_K$). Dann gilt wegen der Reproduktionseigenschaft des Kerns (vgl. Satz 4.39) für $i = 1, \ldots, n$:

$$\rho_2(X_i) = \langle \rho_2, K(X_i, \cdot) \rangle_K \overset{K(X_i,\cdot) \in \mathcal{C}}{=} 0$$

Es folgt

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\tilde{L}(Y_i,\delta(X_i))+\lambda\cdot J(\delta)\\
=\;&\frac{1}{n}\sum_{i=1}^{n}\tilde{L}(Y_i,b+\rho_1(X_i))+\lambda\cdot(J(\rho_1)+J(\rho_2))\\
\geq\;&\frac{1}{n}\sum_{i=1}^{n}\tilde{L}(Y_i,b+\rho_1(X_i))+\lambda\cdot J(\rho_1),
\end{aligned}$$

d. h., jedes δ mit $\rho_2 \not\equiv 0$ ist mindestens genauso groß wie das entsprechende Gegenstück mit $\rho_2 \equiv 0$. Daher hat ein Minimierer von Gl. (4.52) die Form $\delta(x) = b + \rho_1(x)$.

4.6.2 Die Modellannahme

Mit der Interpretation aus Bemerkung 3.8 schätzen wir mit $\hat{\delta}_{n,C}^{SVM,K}$ also eine optimale *Diskriminantenfunktion* δ^*. Damit $\text{sign}(\hat{\delta}_{n,C}^{SVM,K}) = \hat{f}_{n,C}^{SVM,K} \to f^*$ gelten kann, muss $\hat{\delta}_{n,C}^{SVM,K} \to \delta^*$ gelten und die Kalibrierungsbedingung (K1)

$$f^* = \text{sign}(\delta^*)$$

erfüllt sein. Damit das Excess Bayes Risk von $\hat{\delta}_{n,C}^{SVM,K}$ bzgl. $\tilde{L}$ auf das Excess Bayes Risk von $\hat{f}_{n,C}^{SVM,L}$ bzgl. der 0-1-Verlustfunktion übertragen werden kann, benötigen wir eine Risikoübertragungsformel (K2) (vgl. Abschn. 3.3). Das folgende Lemma liefert beide Resultate:

Lemma 4.46 Sei $\tilde{L}$ die Hinge-Verlustfunktion und $\eta(x) := \mathbb{P}(Y = 1|X = x)$. Dann gelten folgende Aussagen:

(i) (K1) Der Minimierer

$$\delta^* \in \arg\min_{\delta:\mathcal{X}\to\mathbb{R}} \tilde{R}(\delta), \qquad \tilde{R}(\delta) := \mathbb{E}\tilde{L}(Y,\delta(X))$$

erfüllt $\delta^* = f^*$ und insbesondere $f^* = \text{sign}(\delta^*)$ $\mathbb{P}^X$-f. s. auf der Menge $\{x \in \mathbb{R}^d : \eta(x) \notin \{0, \frac{1}{2}, 1\}\}$.

(ii) (K2) Für jedes $\delta : \mathcal{X} \to \mathbb{R}$ mit $f = \text{sign}(\delta)$ gilt:

$$R(f) - R(f^*) \leq \tilde{R}(\delta) - \tilde{R}(\delta^*)$$

Beweis

(i) Die Hinge-Verlustfunktion erfüllt $\tilde{L}(y,s) = \phi(-ys)$ mit $\phi(z) = (1+z)^+$. Wir wenden Lemma 3.10 an und erhalten:

$$\delta^*(x) \in \arg\min_{z\in\mathbb{R}} \Phi_{\eta(x)}(z),$$

$\mathbb{P}^X$-f. s. auf $\{x \in \mathbb{R}^d : \eta(x) \notin \{0, \frac{1}{2}, 1\}\}$, wobei

$$\Phi_\eta(z) = \phi(-z)\eta + \phi(z)(1-\eta) = (1-z)^+\eta + (1+z)^+(1-\eta).$$

Es ist leicht zu sehen, dass für $\eta \in (\frac{1}{2}, 1)$ dieser Ausdruck durch $z = 1$ und für $\eta \in (0, \frac{1}{2})$ durch $z = -1$ (eindeutig) minimiert wird. Daraus ergibt sich auf $\{x \in \mathbb{R}^d : \eta(x) \notin \{0, \frac{1}{2}, 1\}$:

$$\arg\min_{z\in\mathbb{R}} \Phi_{\eta(x)}(z) = \begin{cases} 1, & \eta(x) \in (\frac{1}{2}, 1), \\ -1, & \eta(x) \in (0, \frac{1}{2}) \end{cases} = f^*(x), \quad x \in \mathcal{X}$$

(ii) Wir haben gesehen, dass

$$H(\eta) = \min_{z\in\mathbb{R}} \Phi_\eta(z) = \begin{cases} 2(1-\eta), & \eta \geq \frac{1}{2} \\ 2\eta, & \eta < \frac{1}{2} \end{cases}.$$

Es folgt $1 - H(\eta) = 2|\frac{1}{2} - \eta|$. Damit ist die Voraussetzung von Lemma 3.11 mit $c = \frac{1}{2}$ und $s = 1$ erfüllt, und wir erhalten die Behauptung.
(Für den Hinge-Verlust ist (ii) auch leicht elementar einzusehen).

Mit diesen Ergebnissen liegt die Basis vor, um eine Modellannahme zu formulieren. Der Raum $\mathcal{H}_K^b$ (in welchem unsere Schätzer $\hat{\delta}_{n,C}^{SVM,K}$ liegen) enthält bei Verwendung der von uns vorgestellten und in der Praxis verwendeten Kerne (Gauß-Kern, Polynomkern) nur stetige Funktionen. Dies steht konträr zu der Feststellung aus Lemma 4.46, dass $\delta^* = f^*$, d. h. $\delta^*(x) \in \mathcal{Y} = \{-1, +1\}$, $x \in \mathcal{X}$ eine unstetige Funktion ist. Daher können wir nicht als Modellannahme stellen, dass $\delta^* \in \mathcal{H}_K^b$. Wir müssen hoffen, dass δ^* sich genügend gut durch Funktionen aus $\mathcal{H}_K^b$ approximieren lässt, d. h., eine Folge $\delta_m^* \in \mathcal{H}_K^b$, $m \in \mathbb{N}$, existiert mit $\delta^* = \lim_{m\to\infty} \delta_m^*$, wobei die Konvergenz hier nicht bzgl. $\|\cdot\|_K$ von $\mathcal{H}_K^b$, sondern bzgl. der Norm eines größeren Raumes gemeint ist, der $\mathcal{H}_K^b$ enthält. Eine passende Wahl für diesen größeren Raum ist $L^1(\mathcal{X})$ mit Norm $\|h\|_{L^1(\mathcal{X})} := \int_{\mathcal{X}} |h(x)|\, \mathrm{d}x$. Die Modellannahme lautet dann:

Modellannahme 4.47 (SVM mit Kern K) Es gilt $\delta^* \in L^1(\mathcal{X})$, und es gibt eine Folge $(\delta_m^*)_{m\in\mathbb{N}} \in \mathcal{H}_K^b$ mit $\|\delta^* - \delta_m^*\|_{L^1(\mathcal{X})} \to 0$ $(m \to \infty)$.

Die Modellannahme stellt damit eine Bedingung an die Reichhaltigkeit der Funktionen in $\mathcal{H}_K^b$. Nur wenn sich δ^* gut durch Funktionen aus $\mathcal{H}_K^b$ approximieren lässt, ist die Modellannahme erfüllt.

Im Allgemeinen können wir nicht hoffen, dass $\hat{\delta}_{n,C}^{SVM,K}$ eine gute Approximation von δ^* darstellt. Der SVM-Algorithmus forciert, dass die Darstellung

$$\hat{\delta}_{n,C}^{SVM,K}(x) = \sum_{i=1}^{n} \hat{\alpha}_i Y_i \cdot K(X_i, x) + \hat{\beta}_{0,C} \tag{4.53}$$

nur wenige Summanden enthält (nur wenige $\hat{\alpha}_i \neq 0$) und damit $\hat{\delta}_{n,C}^{SVM,K}$ eine sehr glatte, differenzierbare Funktion ist. Zur Darstellung einer unstetigen Funktion ist jedoch die Nutzung unendlich vieler Summanden notwendig. Eine gute Näherung durch $\hat{\delta}_{n,C}^{SVM,K}$ ist daher erst für sehr viele, dicht liegende Trainingsdaten zu erwarten, bei denen auch viele Summanden in Gl. (4.53) mittels $\hat{\alpha}_i \neq 0$ aktiviert werden.

4.6.3 Theoretische Resultate für das Excess Bayes Risk

Eine schlechte Approximation von δ^* durch $\hat{\delta}_{n,C}^{SVM,K}$ muss jedoch nicht zwangsläufig ein großes Excess Bayes Risk nach sich ziehen. Entscheidend ist vor allem, dass die Vorzeichen der Funktionen δ^* und $\hat{\delta}_{n,C}^{SVM,K}$ übereinstimmen und entsprechend $\tilde{R}(\hat{\delta}_{n,C}^{SVM,K}) - \tilde{R}(\delta^*)$ klein ist. Um dies quantifizieren zu können, ist es notwendig, dass ein Resultat in die Konvergenzrate auch den Approximationsfehler

$$\inf_{\delta \in \mathcal{H}_K^b} \tilde{R}(\delta) - \tilde{R}(\delta^*)$$

einbezieht. Das folgende Ergebnis ist entnommen aus [7, Theorem 3.1] (unter Nutzung von $\phi(x) = 2x^2$, $M = 1$ und $\delta = n^{-1}$ in der dortigen Notation). Zur Vereinfachung wird davon ausgegangen, dass δ^* bereits durch $\mathcal{H}_K$ ausreichend beschrieben werden kann, d. h., anstelle von $\mathcal{H}_K^b$ wird in Gl. (4.52) über $\mathcal{H}_K$ minimiert und die Modellannahme entsprechend mit $\mathcal{H}_K$ anstatt $\mathcal{H}_K^b$ formuliert. Das Resultat kann für beliebige Verteilungen von X formuliert werden; zur Vereinfachung der Notation beschränken wir uns hier auf gleichverteilte X (ansonsten sind die Werte λ_j verschieden von den Werten im Satz von Mercer 4.36).

Satz 4.48 Es sei $\mathcal{X} = [0,1]^d$ und $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ ein Mercer-Kern mit $\sup_{x \in \mathcal{X}} K(x,x) \leq 1$. Es sei X gleichverteilt auf $[0,1]^d$. Es gebe Konstanten $\eta_0, \eta_1 > 0$ mit

$$Für\ alle\ x \in \mathcal{X}: \quad \left|\eta(x) - \frac{1}{2}\right| \geq \eta_0, \qquad \min\{\eta(x), 1-\eta(x)\} \geq \eta_1. \tag{4.54}$$

Seien $\lambda_1 \geq \lambda_2 \geq \ldots$ die Werte aus dem Satz von Mercer 4.36. Sei $s_d(n) := \frac{1}{\sqrt{n}} \inf_{a \in \mathbb{N}} \left\{ \frac{a}{\sqrt{n}} + \eta_1 \sqrt{\sum_{j>a} \lambda_j} \right\}$. Dann gibt es eine Konstante $c > 0$ unabhängig von d, n, so dass für alle

$$\lambda \geq \frac{c}{\eta_1} \left\{ s_d(n) + \frac{\log(n \log(n))}{n} \right\} \tag{4.55}$$

und $d = d_n$ gilt:

$$\lim_{n \to \infty} \mathbb{P}\left(\tilde{R}(\hat{\delta}_{n,C}^{SVM,K}) - \tilde{R}(\delta^*) \geq A_d(n) + \gamma_d(n) \right) = 0, \tag{4.56}$$

wobei

$$\gamma_d(n) := 2\lambda \cdot \left[16 + c \frac{\eta_1}{\eta_0} \right], \quad A_d(n) := 2 \inf_{\delta \in \mathcal{H}_K} \left[\tilde{R}(\delta) - \tilde{R}(\delta^*) + 4\lambda \|\delta\|_K^2 \right].$$

Bemerkung 4.49

- Wegen Lemma 4.46 gilt

$$R(\hat{f}_{n,C}^{SVM,K}) - R(f^*) \leq \tilde{R}(\hat{\delta}_{n,C}^{SVM,K}) - \tilde{R}(\delta^*),$$

  und das Resultat kann unmittelbar auf das Excess Bayes Risk des SVM-Klassifizierers $\hat{f}_{n,C}^{SVM,K}$ (bzgl. 0-1-Verlust) übertragen werden.
- Die Bedingung in Gl. (4.54) ist die *low-noise condition* aus Definition 3.12 mit $\alpha = 1$. Sie fordert, dass $\eta(x) = \mathbb{P}(Y = 1 | X = x)$ wegbeschränkt von 0, $\frac{1}{2}$, 1 ist. Anschaulich bedeutet diese Bedingung, dass es keine „schwierigen" ($\eta(x) = \frac{1}{2}$) und keine „eindeutigen" ($\eta(x) \in \{0, 1\}$) Entscheidungen für eine Klasse gibt.
- Die Abhängigkeit von der Dimension d der Trainingsdaten geht durch die Eigenwerte λ_j und damit über die Funktion $s_d(n)$ in λ ein. Durch die Bedingung in Gl. (4.55) werden zu kleine Werte von λ (d. h. zu große Werte von C) ausgeschlossen.
- Der Summand $A_d(n)$ der Rate quantifiziert den Approximationsfehler, der durch die Modellannahme verursacht wird (wir nehmen an: $\delta^* \in L^1(\mathcal{X})$, aber nutzen nur $\hat{\delta}_{n,C}^{SVM,K} \in \mathcal{H}_K$ zur Schätzung). Auch für viele andere in diesem Buch vorgestellte Algorithmen existieren solche Resultate mit expliziter Einarbeitung des Approximationsfehlers. Da aber im Gegensatz zur SVM bei den anderen Algorithmen eine Modellannahme formuliert werden kann, für welche der Approximationsfehler als null angenommen werden kann, lassen wir dessen Beschreibung dort der Übersichtlichkeit halber weg.
- Ist $\delta_m \in \mathcal{H}_K, m \in \mathbb{N}$ eine Folge mit $\|\delta_m - \delta^*\|_1 \to 0$ (vgl. Modellannahme 4.47), so gilt $\tilde{R}(\delta_m) - \tilde{R}(\delta^*) \leq \|\delta_m - \delta^*\|_1 \to 0$, da X gleichverteilt auf $[0, 1]^d$. In diesem Fall gilt für den Approximationsfehler:

$$A_d(n) \leq 2 \left[\|\delta_m - \delta^*\|_1 + 4\lambda \cdot \|\delta_m\|_K^2 \right]$$

Da $\delta^* \notin \mathcal{H}_K$ nicht stetig ist, gilt allerdings $\|\delta_m\|_K^2 \to \infty$ $(m \to \infty)$. Je nach gewähltem Kern K sind somit genauere Informationen über die Rate $\|\delta_m - \delta^*\|_1$ und $\|\delta_m\|_K^2$ nötig, um eine Bedingung an $\lambda = \lambda_n$ stellen zu können und eine Folge $m = m(n)$ zu finden, so dass $A_d(n) \to 0$. Diese zusätzliche Bedingung an λ, die im Allgemeinen dazu führt, dass λ noch größer als in Gl. (4.56) gewählt werden muss, verlangsamt die Rate $A_d(n)+\gamma_d(n)$.

Wir zeigen nun, welche Raten $\gamma_d(n)$ für Gauß- und Polynomkern erreicht werden können. Wie gerade bemerkt, ist die tatsächlich erreichte Rate des Excess Bayes Risks von $\hat{f}_{n,C}^{SVM,K}$ jedoch langsamer und hängt davon ab, wie gut δ^* durch Funktionen aus $\mathcal{H}_K$ approximiert werden kann.

Beispiel 4.50 (Anwendung auf den Polynomkern) Falls $K(x, x') = K_p^{poly}(x, x') = (1+x^T x')^p$ der Polynomkern mit $p \in \mathbb{N}$, so gibt es $c(p, d) \in \mathbb{N}$ mit $\lambda_j = 0$ für $j \geq c(p, d)$. Grob approximiert ist $c(p, d) \leq C \cdot 2^p d^p$ mit einer Konstanten C unabhängig von p, d, vgl. Lemma 4.35 für den Spezialfall $p = d = 2$.

Es folgt $\gamma_j = 0$ für $j \geq c(p, d)$. Daher ist $s_d(n) \leq \frac{c(p,d)}{n}$. Wählen wir λ entsprechend Gl. (4.55) mit „=“, so erhalten wir unter der Annahme $c(p, d) \geq \log(n \log(n))$:

$$\gamma_d(n) \leq 4\left[16 + c\frac{\eta_1}{\eta_0}\right] \cdot \frac{c(p, d)}{n}$$

Je größer p, desto schlechter wird das Verhältnis von d zu n, bei welchem noch Konvergenz gegen null erwartet werden kann. Gleichzeitig sinkt mit p aber auch der Approximationsfehler $A_d(n)$, da der Raum $\mathcal{H}_{K_p^{poly}}$ größer wird.

Beispiel 4.51 (Anwendung auf den Gaußkern) Falls $K(x, x') = K_\gamma^{gauss}(x, x') = \exp(-\gamma\|x - x'\|_2^2)$, so gilt für $j \in \mathbb{N}$ näherungsweise $\lambda_j \leq c_1(c_2 j)^d \rho(\gamma)^j$ mit Konstanten $c_1, c_2 > 0$ unabhängig von d und $\rho \in (0, 1)$, vgl. [38, Example 1].

Eine grobe Abschätzung liefert $\sum_{j>a} \lambda_j \leq c_3 \frac{d!}{(1-\rho)^d}\rho^a$ mit einer Konstanten $c_3 > 0$ unabhängig von d, γ. Wählen wir a so, dass

$$\frac{1}{\sqrt{n}} \stackrel{!}{=} c_3 \frac{d!}{(1-\rho)^d}\rho^a,$$

folgt $s_d(n) \leq c_4 \frac{d \log(d)}{n}$ mit einer Konstanten $c_4 > 0$ unabhängig von d. Wählen wir λ entsprechend Gl. (4.55) mit „=“, so erhalten wir unter der Annahme $d \log(d) \geq \log(n \log(n))$:

$$\gamma_d(n) \leq 4\left[16 + c\frac{\eta_1}{\eta_0}\right] \leq \frac{d \log(d)}{n}$$

Für Gauß-Kerne wurde in [31] eine alternative Modellannahme in Form einer sogenannten *geometric noise condition* an die Größe $|\eta(x) - \frac{1}{2}|$ formuliert, unter welcher der

Approximationsfehler explizit berechnet werden kann. Unter der dort getroffenen Annahme gilt mit Konstanten $\alpha_{geo}, c_{geo} > 0$ unabhängig von d (vgl. [31, Theorem 2.7]):

$$A_d(n) \leq \lambda \cdot \left(\frac{81\gamma}{\pi}\right)^{d/2} + 8c_{geo}(2d)^{\alpha_{geo}d/2} \cdot \gamma^{-\alpha_{geo}d/2}$$

Wie man sieht, ist auch die Wahl des Abstiegsparameters $\gamma > 0$ im Gauß-Kern K_γ^{gauss} elementar, um einen guten Approximationsfehler zu erhalten. Der Ausdruck $A_d(n)$ wird klein für $\gamma = \gamma(\lambda) = \lambda^{-\frac{1}{(\alpha_{geo}+1)d}}$. In [31, Theorem 2.8] wird gezeigt, dass eine sinnvolle Wahl von γ durch $n^{\frac{c(\alpha_{geo})}{d}}$ gegeben ist, wobei $c(\alpha_{geo})$ eine Konstante abhängig von α_{geo} ist. In der Praxis sollte γ also mit der Anzahl der Trainingsdaten n wachsen (was für einen steileren Abstieg sorgt), aber mit der Dimension d fallen.

4.7 Übungen

1. Implementieren Sie die Algorithmen $\hat{f}_n^{LDA}$ und $\hat{f}_n^{QDA}$ aus Definition 4.3 in R und wenden Sie diese auf die Modelle aus Beispiel 4.4 und 4.5 an.
2. Zeigen Sie die Aussagen aus Lemma 4.21.
3. Implementieren Sie den SVM-Klassifizierer $\hat{f}_{n,C}^{SVM,K}$ mit frei wählbarem Bestrafungsparameter C und Kern K mittels Definition 4.44 und wenden Sie ihn für verschiedene C, K auf die Daten aus Beispiel 4.28 an.
4. Erweitern Sie den implementierten SVM-Klassifizierer unter Nutzung der Methoden aus Bemerkung 3.15, um ihn auf die Drei-Klassen-Probleme aus Beispiel 4.4 anzuwenden.

Nichtparametrische Methoden und der naive Bayes-Klassifizierer

5

Inhaltsverzeichnis

In diesem Kapitel betrachten wir einige elementare Algorithmen aus der Nichtparametrik. In diesem Teilgebiet der Statistik werden die Modellannahmen $f^* \in \mathcal{F}$ so allgemein wie möglich gehalten. Im Gegensatz zu den vorherigen Kapiteln nehmen wir nicht mehr an, dass $\mathcal{F}$

- nur durch endlich viele Parameter beschrieben wird (vgl. lineare Modelle für Regression, Kap. 2) oder
- durch abzählbar viele Parameter beschrieben wird und sich jedes $f \in \mathcal{F}$ bzw. damit verknüpfte Diskriminantenfunktionen als unendliche Linearkombination von fest vorgegebenen Basisfunktionen darstellen lässt (vgl. Reproducing Kernel Hilbert Spaces, Kap. 4, dort war die Basisfunktion durch die zum Kern gehörige Orthonormalbasis gegeben) und somit die Form des Entscheidungsrands bereits durch die Modellannahme bereits grob vorgegeben ist.

Stattdessen wollen wir nur allgemeine strukturelle Annahmen an f^* oder die Diskriminantenfunktionen stellen. In diesem Kapitel machen wir dies durch die Annahme, dass bei Regressionsproblemen f^* stetig differenzierbar ist.

Die nichtparametrische Statistik stellt auch Theorien für allgemeinere Strukturannahmen zur Verfügung. So können ebenso Algorithmen und zugehörige Aussagen für den Generalisierungsfehler für monotone oder Hölder-stetige Funktionen angegeben werden. Für solche Resultate verweisen wir zum Beispiel auf [34].

S. Richter, *Statistisches und maschinelles Lernen*,
https://doi.org/10.1007/978-3-662-59354-7_5

5.1 Nichtparametrische Algorithmen für Regressionsprobleme

In diesem Abschnitt betrachten wir Regressionsprobleme mit quadratischer Verlustfunktion.

5.1.1 Herleitung und Motivation

Sei $C > 0$. Wir definieren die Menge der stetig differenzierbaren Funktionen mit durch C beschränkter Ableitung,

$$\mathcal{F} := \{f : \mathcal{X} \to \mathbb{R} | f \text{ ist stetig differenzierbar und } \sup_{x \in \mathcal{X}} \sup_{j=1,...,d} |\partial_{x_j} f(x)| \leq C\}.$$

Die Modellannahme an f^* lautet dann:

Modellannahme 5.1 (Nichtparametrische Regression – Version 1) Es gibt $C > 0$ mit $f^* \in \mathcal{F}_C$.

Die Beschränkung der Ableitung durch C ist für theoretische Resultate notwendig.

Für i. i. d. Trainingsdaten (X_i, Y_i), $i = 1, ..., n$ erwarten wir bei einem Regressionsproblem folgende Struktur: Definieren wir $\varepsilon_i := Y_i - f^*(X_i)$, so sind auch ε_i, $i = 1, ..., n$ i. i. d. mit $\mathbb{E}[\varepsilon_i | X_i] = 0$, $i = 1, ..., n$, und es gilt

$$Y_i = f^*(X_i) + \varepsilon_i, \qquad i = 1, ..., n. \tag{5.1}$$

Wir erwarten also, dass Y_i durch eine von ε_i verrauschte Anwendung von f^* aus X_i entsteht. Zur Illustration betrachten wir folgendes Beispiel für den Spezialfall eindimensionaler X_i:

Beispiel 5.2 (Eindimensionales Regressionsproblem) Die Daten X_i, $i = 1, ..., n$ seien gleichverteilt auf $\mathcal{X} = [0, 1]$, und es gelte

$$Y_i = f^*(X_i) + \varepsilon_i, \qquad i = 1, ..., n,$$

wobei $\varepsilon_i \sim N(0, \sigma^2)$, $\sigma = 0{,}5$ und $f^*(x) = \sin(2\pi x)$.

Ein Beispiel für $n = 50$ gezogene Trainingsdaten ist in Abb. 5.1a dargestellt.

Im Folgenden betrachten wir die quadratische Verlustfunktion $L(y, s) = (y - s)^2$. Eine direkte Verwendung des Standardansatzes aus Abschn. 1.3, die Minimierung des empirischen Risikos

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f(X_i))$$

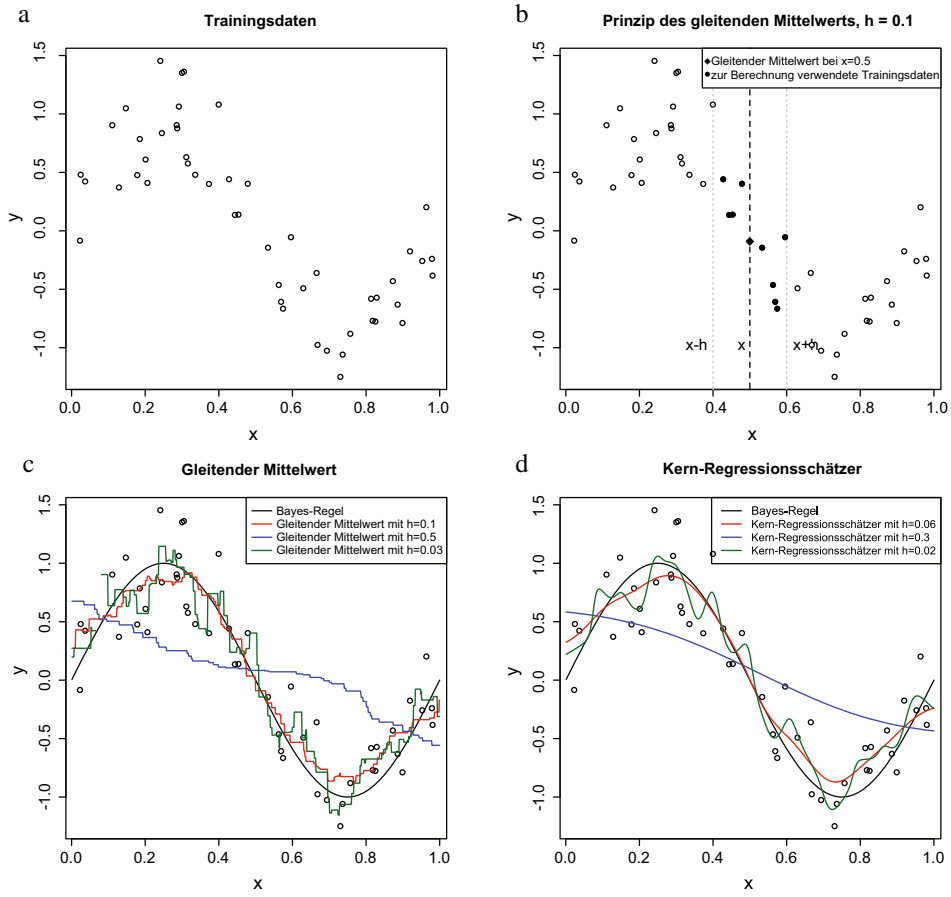


Abb. 5.1 **a** $n = 50$ Trainingsdaten aus Beispiel 5.2, **b** Darstellung der Berechnung des gleitenden Mittelwerts an einer Stelle $x = 0{,}5$, **c** Darstellung des gleitenden Mittelwerts für drei verschiedene Bandbreiten $h \in \{0{,}03, 0{,}1, 0{,}5\}$, **d** Darstellung des Kern-Regressionsschätzers mit $h \in \{0{,}02, 0{,}06, 0{,}3\}$

über alle $f \in \mathcal{F}_C$, ist nicht sinnvoll, da es (für C groß genug) offensichtlich Minimierer $\hat{f}$ mit der Eigenschaft $\hat{f}(X_i) = Y_i$, $i = 1, ..., n$ und damit $\hat{R}_n(\hat{f}) = 0$ gibt.

Stattdessen verwenden wir zunächst einen grafischen Ansatz: Da f^* als differenzierbare Funktion stetig ist, gilt:

$$|x - X_i| \text{ „klein“} \quad \Rightarrow \quad |f^*(x) - f^*(X_i)| \text{ „klein“} \quad \overset{Y_i \approx f^*(X_i)}{\Rightarrow} \quad |f^*(x) - Y_i| \text{ „klein“} \tag{5.2}$$

Das bedeutet, für ein fest vorgegebenes $x \in \mathcal{X}$ können wir $f^*(x)$ wie folgt annähern: Wir suchen eine Beobachtung X_i in der Nähe von x und nutzen dann $f^*(x) \approx Y_i$. Aufgrund

des zusätzlichen zufälligen Fehlers ε_i in Gl. (5.1) ist Y_i jedoch im Allgemeinen noch keine gute Schätzung für $f^*(x)$.

Zur Entfernung des Einflusses der ε_i suchen wir daher nicht nur *eine* Beobachtung X_i in der Nähe von x, sondern *alle* Beobachtungen, die nicht weiter als eine fest vorgegebene Distanz $h > 0$ von x entfernt liegen, und schätzen $f^*(x)$ durch den Mittelwert aller zugehörigen Beobachtungen Y_i. Dieses Prinzip nennt man gleitenden Mittelwert; in Abb. 5.1b ist das Vorgehen für die Trainingsdaten aus Beispiel 5.2 verdeutlicht. Eine Formalisierung dieses Ansatzes ergibt folgende Definition:

Definition 5.3
Sei $h > 0$. Dann heißt

$$\hat{f}_{n,h}^{gm}(x) := \frac{\sum_{i\in\{1,\dots,n\}:|X_i-x|\le h} Y_i}{\#\{i \in \{1,\dots,n\} : |X_i - x| \le h\}}, \qquad x \in \mathbb{R},$$

Gleitender-Mittelwert-Algorithmus (oder *Histogramm-Schätzer*) mit *Bandbreite* h. ♦

In Abb. 5.1c ist $\hat{f}_{n,h}^{gm}$ angewandt auf die Trainingsdaten aus Beispiel 5.2 mit verschiedenen Bandbreiten h zu sehen. Offensichtlich ist eine gute Wahl von h entscheidend für ein niedriges Risiko $R(\hat{f}_{n,h}^{gm})$:

Bemerkung 5.4 (Bandbreitenwahl)

- Bei zu großen Bandbreiten h zieht $\hat{f}_{n,h}^{gm}(x)$ zu viele Beobachtungen zur Schätzung von $f^*(x)$ heran und wird zu „flach", d. h., $\hat{f}_{n,h}^{gm}$ liefert eine systematisch falsche Schätzung von f^*.
- Bei zu kleinen Bandbreiten h hingegen nutzt $\hat{f}_{n,h}^{gm}(x)$ zu wenige Beobachtungen in der Nähe von x zur Schätzung von $f^*(x)$. In diesem Fall wird die grobe Form von f^* zwar erkannt, aber die durch $\hat{f}_{n,h}^{gm}$ repräsentierte Funktion ist zu gezackt, d. h., $\hat{f}_{n,h}^{gm}$ hängt zu stark von den Trainingsdaten ab, hat also eine zu hohe Varianz. Im Extremfall ist $\hat{f}_{n,h}^{gm}$ an einigen Stellen gar nicht definiert, da der Nenner (und der Zähler) null sind.

Im später folgenden Satz 5.9 wird dieses Verhalten theoretisch untermauert. Die Bandbreite h kann als Tuningparameter des Algorithmus aufgefasst werden, eine sinnvolle Auswahl wird dann beispielsweise durch Cross Validation gewährleistet.

5.1.2 Kern-Regressionsschätzer

An Abb. 5.1c ist ersichtlich, dass $\hat{f}_{n,h}^{gm}$ keine stetigen Funktionen liefert. Der Grund ist, dass $\hat{f}_{n,h}^{gm}(x)$ bei langsamer Verschiebung des Werts x seinen Wert abrupt ändert, sobald ein neues Trainingsdatum in den Einzugsbereich $[x - h, x + h]$ von x eintritt. Aufgrund der Modellannahme $f^* \in \mathcal{F}_C$ ist es jedoch wünschenswert, dass ein Algorithmus zur Schätzung

von f^* dieselben Glattheitseigenschaften wie f^* besitzt. Im Folgenden modifizieren wir den gleitenden Mittelwert zu einer glatten Funktion. Setzen wir

$$W(u) := \frac{1}{2}\mathbb{1}_{[-1,1]}(u) := \begin{cases} 1, & u \in [-1,1] \\ 0, & u \in \mathbb{R}\backslash[-1,1] \end{cases}, \tag{5.3}$$

so erhalten wir die alternative Schreibweise

$$\hat{f}_{n,h}^{gm}(x) := \frac{\sum_{i=1}^{n} W\left(\frac{X_i - x}{h}\right) Y_i}{\sum_{i=1}^{n} W\left(\frac{X_i - x}{h}\right)}, \quad x \in \mathbb{R}, \tag{5.4}$$

von $\hat{f}_{n,h}^{gm}$. Dies erlaubt die Interpretation von $\hat{f}_{n,h}^{gm}(x)$ als gewichtetes Mittel *aller* Y_i, wobei die Gewichte der einzelnen Beobachtungen Y_i durch $K\left(\frac{X_i - x}{h}\right)$ gegeben sind. Durch W in Gl. (5.3) wird jede Beobachtung entweder mit Gewicht 1 oder 0 in die Berechnung einbezogen. Durch Rekapitulation des ursprünglichen Ansatzes (5.2) ist klar, dass die Qualität von $\hat{f}_{n,h}^{gm}(x)$ verbessert werden kann, wenn Beobachtungen Y_i mit kleinem Abstand $|X_i - x|$ stärker gewichtet werden als solche mit größerem $|X_i - x|$.

Formal können wir dies durch eine Änderung der Funktion W erreichen:

- Ist W stetig, so erhält automatisch auch der Algorithmus in Gl. (5.4) diese wünschenswerte Eigenschaft.
- Anschaulich sollte $W(0)$ groß sein und $W(u) \to 0$ für $u \to \pm\infty$ gelten, diese Bedingungen werden für theoretische Resultate aber nicht benötigt.
- Da W in Gl. (5.4) in Zähler und Nenner in gleicher Form vorkommt, kann ohne Beschränkung der Allgemeinheit eine Normierung von W verlangt werden. Damit W in Zähler und Nenner getrennt einer „Gewichtung“ entspricht, verlangen wir die Normierung $\int W(u)\,\mathrm{d}u = 1$.

Eine Erweiterung des Prinzips auf d-dimensionale Trainingsdaten erreichen wir durch Erweiterung der Funktion W auf den Definitionsbereich $\mathbb{R}^d$:

Definition 5.5
Eine stetige Funktion $W : \mathbb{R}^d \to [0, \infty)$ heißt *Kernfunktion*, falls $\int_{\mathbb{R}^d} W(u)\,\mathrm{d}u = 1$. ♦

Beispiele für d-dimensionale Kernfunktionen können aus eindimensionalen Kernfunktionen $\tilde{W}$ mittels

$$W(u) := \prod_{j=1}^{d} \tilde{W}(u_j), \qquad u = (u_1, \ldots, u_d)^T \in \mathbb{R}^d$$

gewonnen werden:

Beispiel 5.6 (Kernfunktionen)

- $\tilde{W} : \mathbb{R} \to \mathbb{R}$, $\tilde{W}(u) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{u^2}{2})$ heißt eindimensionaler *Gauß-Kern,*

$$W : \mathbb{R}^d \to \mathbb{R}, \quad W(u) = \frac{1}{(2\pi)^{d/2}} \exp(-\frac{\|u\|_2^2}{2})$$

  heißt d-dimensionaler Gauß-Kern.
- $\tilde{W}(x) = \frac{1}{2} \mathbb{1}_{[-1,1]}(u)$ heißt eindimensionaler *Rechteck-Kern,*

$$W : \mathbb{R}^d \to \mathbb{R}, \quad W(u) = \frac{1}{2^d} \mathbb{1}_{\{\|u\|_1 \leq 1\}}$$

  heißt d-dimensionaler Rechteck-Kern.

Für $d = 1$ sind die beiden Kerne in Abb. 5.2 dargestellt.

Nach diesen Vorüberlegungen können wir eine Erweiterung der Idee des gleitenden Mittelwerts auf d-dimensionale Daten $X_i \in \mathcal{X} \subset \mathbb{R}^d$ wie folgt definieren:

Definition 5.7 (Kern-Regressionsschätzer)
Sei $h > 0$. Sei $W : \mathbb{R}^d \to \mathbb{R}$ eine Kernfunktion. Der Algorithmus

$$\hat{f}_{n,h}^W(x) = \frac{\frac{1}{nh^d} \sum_{i=1}^n W\left(\frac{X_i - x}{h}\right) Y_i}{\frac{1}{nh^d} \sum_{i=1}^n W\left(\frac{X_i - x}{h}\right)}$$

heißt *Kern-Regressionsschätzer* oder *Nadaraya-Watson-Schätzer* mit Kern W und Bandbreite h. ♦

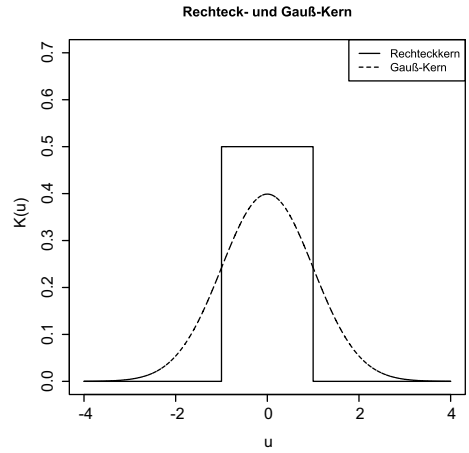


Abb. 5.2 Darstellung des Rechteck- und des Gauß-Kerns in einer Dimension

Die Aussagen für die Wahl der Bandbreite h aus Satz 5.4 gelten entsprechend, in der Praxis erfolgt eine Wahl zum Beispiel durch Cross Validation. In Abb. 5.1 ist $\hat{f}^W_{n,h}$ mit dem eindimensionalen Gauß-Kern und drei verschiedenen Bandbreiten dargestellt.

5.1.3 Theoretische Resultate

Im Folgenden beweisen wir ein theoretisches Resultat für das Excess Bayes Risk von $\hat{f}^W_{n,h}$. Für den Beweis müssen auch Bedingungen an die wahre Dichte g der Beobachtungen X gestellt werden, die wir mit in die Modellannahme aufnehmen:

Modellannahme 5.8 (Nichtparametrische Regression – Version 2) Es sei $\mathcal{X} \subset \mathbb{R}^d$ kompakt. Es gebe $C > 0$ mit $f^* \in \mathcal{F}_C$ und $g \in \mathcal{F}_C$. Es gebe Konstanten $c_{g,min}, c_{g,max}$, $c_{f*,max} > 0$ unabhängig von d mit $c_{g,max} \geq \sup_{x\in\mathcal{X}} g(x) \geq \inf_{x\in\mathcal{X}} g(x) \geq c_{g,min}$ und $\sup_{x\in\mathcal{X}} |f^*(x)| \leq c_{f*,max}$.

Aufgrund der Quotientenstruktur von $\hat{f}^W_{n,h}$ ist eine Berechnung des Generalisierungsfehlers schwierig. Wir geben unter der Annahme kompakter $\mathcal{X}$ und beschränkter Fehler ε_i eine obere Schranke für $\mathbb{E}R(\hat{f}^W_{n,h})$ an. Auch wenn es sich nur um eine obere Schranke handelt, so ist deren Rate in n, h und d optimal (vgl. [34]).

Satz 5.9 (Excess Bayes Risk von Kern-Regressionschätzern) Es gelte die Modellannahme 5.8. Weiter gelten folgende Aussagen:

- Es gebe eine Konstante $c_{\varepsilon,max} > 0$ mit $|\varepsilon_1| \leq c_{\varepsilon,max}$ f. s.
- W sei eine Kernfunktion mit $\int_{\mathbb{R}^d} W(u)^2 \, \mathrm{d}u < \infty$ und $\int_{\mathbb{R}^d} |u| W(u) \, \mathrm{d}u < \infty$.

Dann gibt es eine von x, n, h, d unabhängige Konstante $c > 0$, so dass für alle $x \in \mathcal{X}$:

$$\mathrm{MSE}\left(\hat{f}^W_{n,h}(x)\right) \leq c \cdot \left[\frac{1}{nh^d} + h^2\right], \qquad \mathbb{E}R\left(\hat{f}^W_{n,h}\right) - R(f^*) \leq c \cdot \left[\frac{1}{nh^d} + h^2\right] \tag{5.5}$$

Die oberen Schranken erreichen für $h = h^* := n^{-\frac{1}{d+2}}$ die optimale Konvergenzrate in n; in diesem Falle gilt

$$\mathbb{E}R(\hat{f}^W_{n,h^*}) - R(f^*) \leq 2c \cdot n^{-\frac{2}{d+2}}. \tag{5.6}$$

Beweis Wir schreiben $\hat{f}^W_{n,h} = \frac{\hat{m}_{n,h}}{\hat{g}_{n,h}}$ mit $\hat{m}_{n,h}(x) = \frac{1}{nh^d} \sum_{i=1}^n W\left(\frac{X_i - x}{h}\right) Y_i$, $\hat{g}_{n,h}(x) = \frac{1}{nh^d} \sum_{i=1}^n W\left(\frac{X_i - x}{h}\right)$. Dann gilt

$$\hat{f}^W_{n,h} - f^* = \frac{\hat{m}_{n,h} - f^* \cdot \hat{g}_{n,h}}{g} + \frac{g - \hat{g}_{n,h}}{g} \cdot \left(\hat{f}^W_{n,h} - f^*\right).$$

Es gilt $|Y_i| \leq c_{\varepsilon,max} + c_{f^*,max}$ f. s. und wegen $W(\cdot) \geq 0$ weiter: $|\hat{f}^W_{n,h}| \leq c_{\varepsilon,max} + c_{f^*,max}$ f. s.

Mit der Regel $(a+b)^2 \leq 2a^2 + 2b^2$ erhalten wir:

$$\begin{aligned}&\left(\hat{f}^W_{n,h}(x) - f^*(x)\right)^2 \\ &\leq \frac{2}{c^2_{g,min}} \left(\hat{m}_{n,h}(x) - f^*(x)\cdot \hat{g}_{n,h}(x)\right)^2 + \frac{2(c_{\varepsilon,max} + 2c_{f^*,max})}{c^2_{g,min}} \left(\hat{g}_{n,h}(x) - g(x)\right)^2\end{aligned} \tag{5.7}$$

Es gilt

$$\mathbb{E}\left[\left(\hat{g}_{n,h}(x) - g(x)\right)^2\right] = \mathrm{Var}(\hat{g}_{n,h}(x)) + |\mathbb{E}\hat{g}_{n,h}(x) - g(x)|^2, \tag{5.8}$$

und da die X_i i. i. d. sind, folgt

$$\begin{aligned}\mathbb{E}\hat{g}_{n,h}(x) - g(x) &= \frac{1}{h^d}\mathbb{E}W(\frac{X_1 - x}{h}) - g(x) = \frac{1}{h^d}\int_{\mathbb{R}^d} W(\frac{z-x}{h}) g(z)\,\mathrm{d}z - g(x) \\ &= \int_{\mathbb{R}^d} W(u)(g(x+uh) - g(x))\,\mathrm{d}u,\end{aligned}$$

wobei im letzten Schritt $u := \frac{z-x}{h}$ und $\int_{\mathbb{R}^d} W(u)\,\mathrm{d}u = 1$ genutzt wurde. Mit $g \in \mathcal{F}_C$ schließen $|\mathbb{E}\hat{g}_{n,h}(x) - g(x)| \leq Ch \cdot \int_{\mathbb{R}^d} W(u)|u|\,\mathrm{d}u$. Da die X_i i. i. d. sind, gilt

$$\begin{aligned}\mathrm{Var}(\hat{g}_{n,h}(x)) &= \frac{1}{n^2h^{2d}} \sum_{i=1}^n \mathrm{Var}\left(W\left(\frac{X_i - x}{h}\right)\right) \leq \frac{1}{nh^{2d}}\mathbb{E}\left[W\left(\frac{X_1 - x}{h}\right)^2\right] \\ &\leq \frac{1}{nh^d}\int_{\mathbb{R}^d} W(u)^2 g(x+uh)\,\mathrm{d}u \leq \frac{c_{g,max}}{nh^d}\int_{\mathbb{R}^d} W(u)^2\,\mathrm{d}u.\end{aligned}$$

Einsetzen dieser Resultate in Gl. (5.8) liefert $\mathbb{E}\left[(\hat{g}_{n,h}(x) - g(x))^2\right] \leq c(\frac{1}{nh^d} + h^2)$. Da $|\varepsilon_i| \leq c_{\varepsilon,max}$ f. s., (X_i, ε_i) i. i. d. und die Struktur

$$\hat{m}_{n,h}(x) - f^*(x)\cdot\hat{g}_{n,h}(x) = \frac{1}{nh^d}\sum_{i=1}^n W\left(\frac{X_i - x}{h}\right)\varepsilon_i$$

ähnlich der von $\hat{g}_{n,h}(x) - g(x)$ ist, folgen ähnliche Aussagen für den ersten Summanden in (5.7).

Die Aussage für das Excess Bayes Risk folgt direkt aus Lemma 1.12. Minimieren der Funktion $h \mapsto \frac{1}{nh^d} + h^2$ liefert $h^* = (\frac{d}{2})^{\frac{1}{d+2}} \cdot n^{-\frac{1}{d+2}}$ und $\frac{1}{n(h^*)^d} + (h^*)^2 = (\frac{d}{2})^{\frac{2}{d+2}}(1 + \frac{2}{d}) \cdot n^{-\frac{2}{d+2}}$. Die im Satz angegebene Wahl von h erreicht dieselbe Konvergenzrate in n.

Bemerkung 5.10

- Die Wahl von W hat keinen Einfluss auf die Raten von n, h, d in den oberen Schranken in Gl. (5.5), sondern nur auf die im Resultat auftretenden Konstanten.
- Wie aus dem Beweis von Satz 5.9 ersichtlich, entspricht der Term $\frac{1}{nh^d}$ der Varianz, der Summand h^2 dem Bias von $\hat{f}^W_{n,h}$. Während für kleine Bandbreiten h eine hohe Varianz auftritt, erhält man für große h einen hohen systematischen Fehler durch den Bias. Die im Satz angegebene optimale Wahl von $h^* = n^{-\frac{1}{d+2}}$ ist nur von theoretischem Interesse und in der Praxis nutzlos, da mit h^* nur eine (schlechte) *obere Schranke* des Excess Bayes Risks minimiert wird, die nur die in n erreichbare Konvergenzrate korrekt abbildet. Das h, welches das Excess Bayes Risk selbst minimiert, hängt von f^* und g ab und ist in Anwendungsproblemen nicht verfügbar.
- *Fluch der Dimension* (engl. *curse of dimension*): An Gl. (5.6) ist ersichtlich, dass bei hoher Dimension d der Eingangsdaten selbst bei optimaler Bandbreite sehr viele Trainingsdaten nötig sind, um ein kleines Excess Bayes Risk zu erhalten. Genauer muss $d \ll \log(n)$ gelten, damit gemäß der oberen Schranke überhaupt $\mathbb{E}R(\hat{f}^W_{n,h^*}) - R(f^*) \to 0\,(n \to \infty)$ erwartet werden kann.
  Der Grund liegt in der sehr allgemeinen Strukturvoraussetzung, die nur Differenzierbarkeit für f^* fordert und damit keine globalen Eigenschaften von f^* festlegt. Daher können Beobachtungen X_i fern von einem Punkt $x \in \mathcal{X}$ keine hilfreichen Informationen zur Bestimmung von $f^*(x)$ beisteuern. Der Algorithmus $\hat{f}^W_{n,h}$ arbeitet dementsprechend auch nur „lokal", d. h., nur Beobachtungen in der Nähe von x haben einen starken Einfluss auf die Schätzung von $f^*(x)$. Damit also $\hat{f}^W_{n,h} \approx f^*$ überhaupt erwartet werden kann, muss eine große Dichte an Beobachtungen im ganzen Raum $\mathcal{X}$ vorliegen, bzw. in der Nähe von *jedem* $x \in \mathcal{X}$ müssen sich einige Beobachtungen X_i befinden.
  Als Beispiel betrachten wir $\mathcal{X} = [0,1]^d$. Wir wollen erreichen, dass jeder Punkt $x \in \mathcal{X}$ höchstens einen Abstand von 0,1 zu einem Trainingsdatum X_i besitzt. Im Falle $d = 1$ wird dafür $n \geq 5$ benötigt (im Falle $n = 5$ müsste $\{X_i, i = 1, ..., 5\}$ genau der Menge $\{0,1, 0,3, 0,5, 0,7, 0,9\}$ entsprechen). Im Falle $d = 2$ sind bereits $n \geq 5^2$ Daten nötig und allgemein $n \geq 5^d$. Damit ergibt sich allgemein die Forderung $d \leq \frac{\log(n)}{\log(5)}$. Damit für die Berechnung von $\hat{f}^W_{n,h}(x)$ mehrere relevante Trainingsdaten zur Verfügung stehen, muss daher $d \ll \log(n)$ gelten, was genau der oben aus der Theorie hergeleiteten Bedingung entspricht.
- Der Fluch der Dimension ist letztlich eine Konsequenz aus der allgemeinen Modellannahme, d. h. der großen Komplexität von $\mathcal{F}_C$. Damit ist zwar der Approximationsfehler von $\hat{f}^W_{n,h}$ gering, aber der Schätzfehler sehr hoch. Man kann zeigen (vgl. [34]), dass *kein* Algorithmus eine bessere Konvergenzrate des Excess Bayes Risks als die oben stehende haben kann.
- Sind f^*, g nicht nur einmal, sondern k-mal differenzierbar ($k \in \mathbb{N}$) mit der durch eine Konstante $C > 0$ beschränkten k-ten Ableitung, so kann mittels Taylor-Entwicklungen und weiterer Annahmen an den Kern W gezeigt werden: $\mathbb{E}R(\hat{f}^W_{n,h}) - R(f^*) \leq c \cdot \left[\frac{1}{nh^d} + h^{2k}\right]$. In diesem Fall ergibt sich mit $h = h^* = n^{-\frac{1}{2k+d}}$ für das Excess Bayes Risk die

obere Schranke $\mathbb{E}R(\hat{f}_{n,h}^W) - R(f^*) \leq cn^{-\frac{2k}{2k+d}}$. Der Fluch der Dimension kann also durch die Annahme, dass f^*, g genügend oft differenzierbar sind (z. B. $2k \approx d$), und die Verwendung eines geeigneten Kerns W verhindert werden. Tatsächlich wird in der Praxis aber selten $k > 2$ angenommen. Ab $k = 3$ fordert das theoretische Resultat, dass der Kern K auch negative Werte annehmen muss, was bei zu geringen Datenmengen zu Instabilitäten der Schätzung führt. Diese können nur vermieden werden, wenn tatsächlich die k-te Ableitung von f^* durch eine Konstante C beschränkt ist, was de facto aber nur Polynome niedrigen Grades erfüllen.

5.2 Nichtparametrische Algorithmen für Klassifikationsprobleme

Wir betrachten hier Klassifikationsprobleme mit K Klassen, d. h. $\mathcal{Y} = \{1, ..., K\}$. Zur Bestimmung des Risikos nutzen wir die 0-1-Verlustfunktion. In diesem Fall lautet die Bayes-Regel $f^*(x) = \arg\max_{k\in\{1,...,K\}} \mathbb{P}(Y = k|X = x)$, vgl. Lemma 1.3.

Zur Bestimmung von geeigneten Klassifizierern nutzen wir die Form der optimalen Diskriminantenfunktionen in Lemma 3.4: Es gebe eine Wahrscheinlichkeitsdichte g von X, und es gebe bedingte Dichten $g_k := f_{X|Y=k}$ von X gegeben $Y = k$ bzgl. des Lebesgue-Maßes auf $\mathbb{R}^d$. Weiter sei $\pi_k = \mathbb{P}(Y = k)$. Wir bestimmen zunächst Schätzer für die optimalen Diskriminantenfunktionen $\tilde{\delta}_k^*(x) = g_k(x)\pi_k$ und dann für

$$f^*(x) = \underset{k\in\{1,...,K\}}{\arg\max}\, \tilde{\delta}_k^*(x). \tag{5.9}$$

Die Schätzung von π_k ist durch

$$\hat{\pi}_k := \frac{\#\{i \in \{1, ..., n\} : Y_i = k\}}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{Y_i=k\}}$$

möglich. Im Folgenden motivieren wir, wie die Wahrscheinlichkeitsdichten g_k geschätzt werden können. Wir motivieren dies für die Dichte g von X und übertragen die Resultate dann auf $g_k = f_{X|Y=k}$.

5.2.1 Herleitung des Kern-Dichteschätzers

Im Beweis von Satz 5.9 wurde bereits gesehen, dass $\hat{g}_{n,h}^W(x) := \frac{1}{nh^d}\sum_{i=1}^n W\left(\frac{X_i-x}{h}\right)$ eine Schätzung von g mit guten theoretischen Eigenschaften liefert. Wir halten folgende Definition fest:

Definition 5.11 (Kern-Dichteschätzer)
Sei $h > 0$ und $W : \mathbb{R}^d \to \mathbb{R}$ eine Kernfunktion. Dann heißt

$$\hat{g}^W_{n,h}(x) := \frac{1}{nh^d} \sum_{i=1}^{n} W\left(\frac{X_i - x}{h}\right), \qquad x \in \mathcal{X}$$

der *Kern-Dichteschätzer* mit Bandbreite h und Kern K für die Dichte g von X. ♦

Anschaulich legt $\hat{g}^W_{n,h}$ um jedes Trainingsdatum X_i einen Hügel in Form der Kernfunktion W und wertet durch $\hat{g}^W_{n,h}(x)$ aus, wie viele Beobachtungen in der Nähe von $x \in \mathcal{X}$ erhalten wurden. Da die Dichte $g(x)$ angibt, wie wahrscheinlich Beobachtungen in der Nähe von x sind, liefert $\hat{g}^K_{n,h}(x)$ damit auch grafisch eine naheliegende Schätzung. In Abb. 5.3 ist dies für eindimensionale Beobachtungen ($d = 1$) illustriert.

Eine theoretische Herleitung von $\hat{g}^W_{n,h}$ ist für eindimensionales $\mathcal{X} = \mathbb{R}$ wie folgt möglich: Ist g die Dichte von X bzgl. des Lebesgue-Maßes und $G(x) = \mathbb{P}(X_1 \leq x)$ deren Verteilungsfunktion, so gilt

$$g(x) = G'(x) = \lim_{h \to 0} \frac{G(x+h) - G(x-h)}{2h} = \lim_{h \to 0} \frac{\mathbb{P}(x-h < X_1 \leq x+h)}{2h}.$$

Ist h fest gewählt, so gilt wegen des starken Gesetzes der großen Zahlen:

$$\frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{(x-h,x+h]}(X_i) \to \mathbb{E}\mathbb{1}_{(x-h,x+h]}(X_1) = \mathbb{P}(x-h < X_1 \leq x+h)$$

Vertauschung der beiden Grenzprozesse liefert den Ansatz

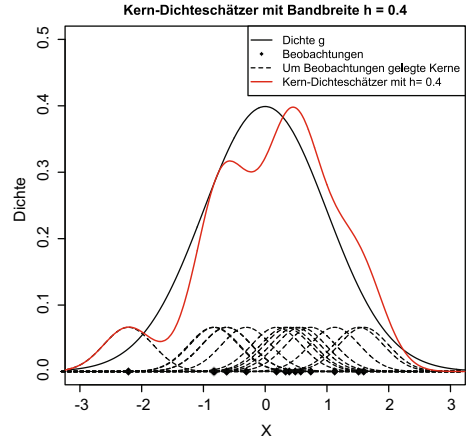


Abb. 5.3 Veranschaulichung des Vorgehens beim Kern-Dichteschätzer

$$\hat{g}_{n,h}(x) = \frac{1}{nh} \sum_{i=1}^{n} \frac{1}{2} \mathbb{1}_{(x-h,x+h]}(X_i).$$

Das bedeutet, $\hat{g}_{n,h}(x)$ zählt, wie viele Beobachtungen X_i sich in der Nähe von x befinden, wobei jede Beobachtung X_i mit $|X_i - x| \leq h$ als „nahe“ an x gilt. Das Resultat wird durch $2nh$ geteilt, damit $\hat{g}_{n,h}$ die Normierungsbedingung einer Wahrscheinlichkeitsdichte $\int g(x)\, \mathrm{d}x = 1$ erfüllt; dabei entspricht der Faktor n der Anzahl der verwendeten Beobachtungen, und der Faktor $2h$ nimmt Bezug darauf, dass jede Beobachtung in Bezug auf die gesamte x-Achse $2h$-mal gezählt wird. Schreiben wir nun wieder

$$\hat{g}_{n,h}(x) = \frac{1}{nh} \sum_{i=1}^{n} W\left(\frac{X_i - x}{h}\right)$$

mit dem Rechteck-Kern $W(x) = \frac{1}{2}\mathbb{1}_{[-1,1]}(x)$ und ersetzen dann W durch beliebige Kerne, erhalten wir den oben definierten Kern-Dichteschätzer $\hat{g}^W_{n,h}$.

Die Aussagen über die Qualität von $\hat{g}^W_{n,h}$ in Abhängigkeit von h lassen sich von den Aussagen über $\hat{f}^W_{n,h}$ aus Satz 5.9 übertragen. Beispielsweise gilt unter geeigneten Annahmen an g die Abschätzung $\mathbb{E}[(\hat{g}^W_{n,h}(x) - g(x))^2] \leq c(\frac{1}{nh^d} + h^2)$ mit einer Konstante $c > 0$ für alle $x \in \mathcal{X}$.

5.2.2 Klassifikation mit Kern-Dichteschätzern

Die bedingte Dichte $g_k = f_{X|Y=k}$ drückt aus, welche Wahrscheinlichkeitsdichte X für Beobachtungen mit $Y = k$ besitzt. Wir nutzen dafür die oben definierten Kern-Dichteschätzer, indem wir nur Beobachtungen X_i mit $Y_i = k$ zur Berechnung nutzen. Ersetzen wir die Größen in Gl. (5.9) durch ihre Schätzer, erhalten wir folgende Definition:

Definition 5.12 (Kern-Dichte-Klassifizierer)
Sei $h > 0$, und $W : \mathbb{R}^d \to [0, \infty)$ eine Kernfunktion. Definiere

$$\hat{\pi}_k = \frac{n_k}{n}, \qquad n_k = \#\{i \in \{1, ..., n\} : Y_i = k\}, \qquad k = 1, ..., K.$$

und

$$\hat{g}^W_{k,n,h}(x) = \frac{1}{n_k h^d} \sum_{i \in \{1,...,n\}: Y_i = k} W\left(\frac{X_i - x}{h}\right).$$

Setze $\hat{\hat{\delta}}_k(x) = \hat{\pi}_k \cdot \hat{g}^W_{k,n,h}(x)$, dann heißt

$$\hat{f}^{K,klass}_{n,h}(x) := \underset{k=1,...,K}{\arg\max}\, \hat{\hat{\delta}}_k(x)$$

der *Kern-Dichte-Klassifizierer* für f^* mit Bandbreite h und Kern W. ♦

Wie beim Kern-Regressionsschätzer wird h in der Praxis zum Beispiel mit Cross Validation gewählt. Da $\hat{f}_{n,h}^{K,klass}$ maßgeblich auf den Kern-Dichteschätzern $\hat{g}_{k,n,h}^{W}$ basiert, gelten Bemerkungen über den Fluch der Dimension entsprechend.

Zur Visualisierung betrachten wir das folgende Beispiel:

Beispiel 5.13 (Zweidimensionales Klassifikationsproblem) Wir betrachten $K = 2$ Klassen und $\mathcal{Y} = \{1, 2\}$. Es seien X_{i1}, X_{i2} i. i. d. gleichverteilt auf $[0,1]$, $X_i = (X_{i1}, X_{i2})^T$. Unabhängig davon seien $\varepsilon_i \sim N(0, \sigma^2)$ i. i. d. mit $\sigma > 0$. Es sei $\kappa : \mathbb{R}^2 \to \mathbb{R}$ und

$$Y_i = \begin{cases} 1, & \kappa(X_i) - \varepsilon_i \leq 0 \\ 2, & \kappa(X_i) - \varepsilon_i > 0 \end{cases} \in \mathcal{Y} = \{1, 2\}.$$

Entsprechend ist der Bayes-Klassifizierer des Problems gegeben durch

$$f^*(x) = \begin{cases} 1, & \kappa(x) \leq 0 \\ 2, & \kappa(x) > 0 \end{cases}.$$

Wir betrachten jeweils $n = 300$ Trainingsdaten der folgenden Spezialfälle (hierbei ist $x = (x_1, x_2)^T \in [0, 1]^2$):

(i) $\kappa = \kappa_1$ mit $\kappa_1(x) = x_2 - 0{,}5 - 0{,}3 \sin(2\pi x_1)$ und $\sigma = 0{,}2$,
(ii) $\kappa = \kappa_2$ mit $\kappa_2(x) = (x_1 - \frac{1}{2}) \cdot (x_2 - \frac{1}{2})$ und $\sigma = 0{,}05$

Die Trainingsdaten und die zugehörigen Ergebnisse von $\hat{f}_{n,h}^{W,klass}$ mit zweidimensionalem Gauß-Kern für (i), (ii) sind in Abb. 5.4 dargestellt. In Abb. 5.4a–d kann man sehen, wie sich $\hat{f}_{n,h}^{W,klass}$ bei Anwendung auf die Trainingsdaten von (i) verhält. Der Algorithmus wurde angewandt mit dem Gauß-Kern und drei verschiedenen Bandbreiten. Ist h nahe dem optimalen Wert für h, so wird der Entscheidungsrand gut geschätzt (vgl. Abb. 5.4b). Ist h zu groß, wird die Grobstruktur des optimalen Entscheidungsrands nicht mehr korrekt getroffen, d. h., es gibt einen hohen Bias (vgl. Abb. 5.4c). Ein zu kleines h führt zu einer Überanpassung an die Trainingsdaten und zu hoher Varianz des geschätzten Entscheidungsrands (vgl. Abb. 5.4d). Für (ii) ist $\hat{f}_{n,h}^{W,klass}$ nur für eine gute Wahl von h dargestellt (vgl. Abb. 5.4f).

Zur genaueren Analyse der Qualität von $\hat{f}_{n,h}^{W,klass}$ in Termen des Generalisierungsfehlers ermitteln wir zunächst eine alternative Darstellung. Der in Definition 5.12 verwendete Ansatz nutzte Lemma 3.4 mit den „vereinfachten" optimalen Diskriminantenfunktionen $\tilde{\delta}_k^*(x) = \pi_k \cdot g_k(x)$. Alternativ können jedoch auch direkt die optimalen Diskriminantenfunktionen

$$\delta_k^*(x) := \mathbb{P}(Y = k | X = x), \quad k = 1, ..., K,$$

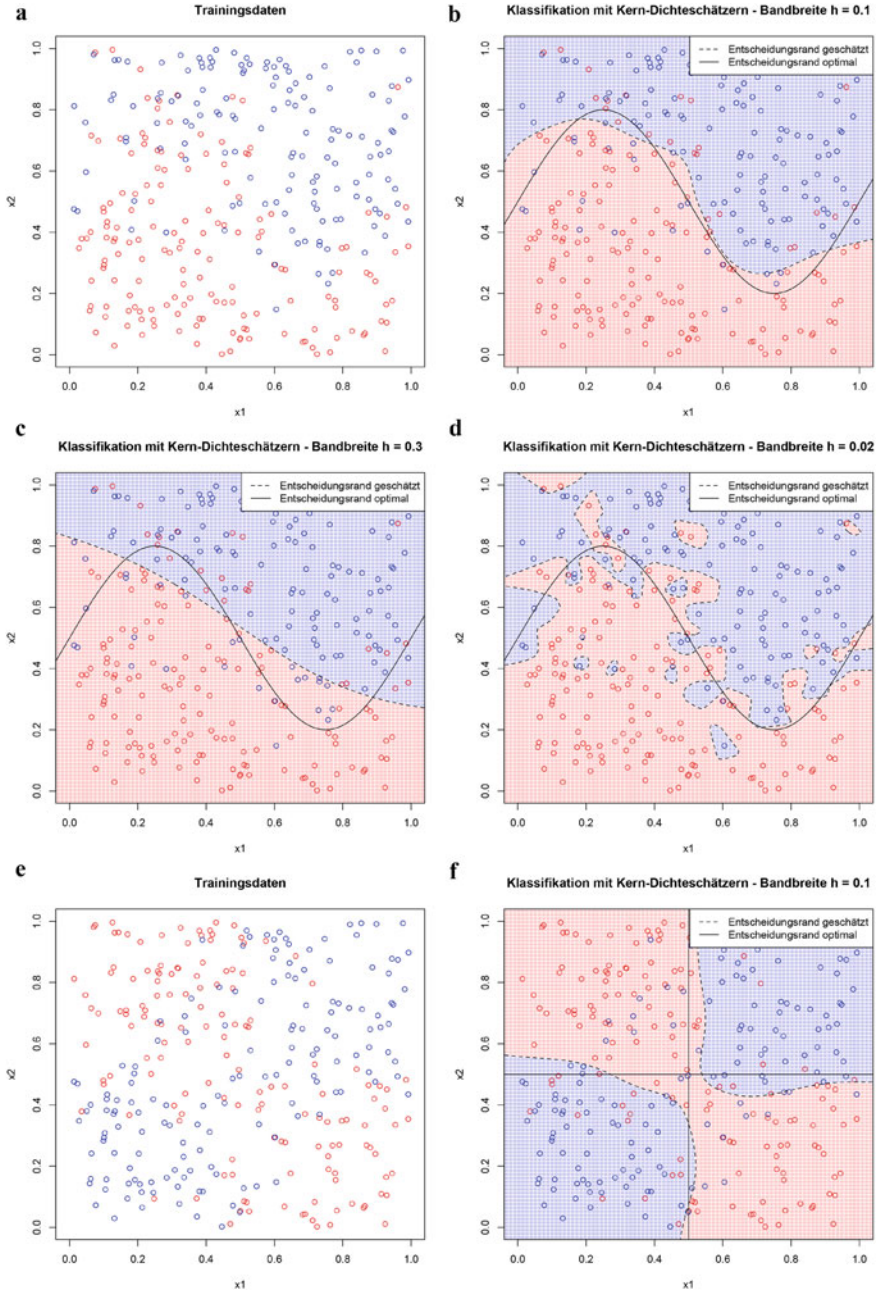


Abb. 5.4 Anwendung von $\hat{f}_{n,h}^{W,klass}$ auf jeweils $n = 300$ Trainingsdaten aus Beispiel 5.13(i), (ii) unter Nutzung des zweidimensionalen Gauß-Kerns. **a** Trainingsdaten im Fall (i). **b**, **c**, **d** $\hat{f}_{n,h}^{W,klass}$ für Bandbreiten $h \in \{0{,}02, 0{,}1, 0{,}3\}$. e): Trainingsdaten im Fall (ii). **f** $\hat{f}_{n,h}^{W,klass}$ für $h = 0{,}1$

verwendet werden. Der Satz von Bayes und der Satz von der totalen Wahrscheinlichkeit liefern:

$$\delta_k^*(x) = \frac{\pi_k g_k(x)}{g(x)} = \frac{\pi_k g_k(x)}{\sum_{k=1}^K \pi_k g_k(x)}$$

Setzen wir die oben ermittelten Schätzer für π_k und g_k ein, erhalten wir wegen $\sum_{k=1}^K \mathbb{1}_{\{Y_i=k\}} = 1$:

$$\hat{\delta}_k(x) := \frac{\hat{\pi}_k \cdot \hat{g}_{k,n,h}^W(x)}{\sum_{l=1}^K \pi_l \cdot \hat{g}_{l,n,h}^W(x)} = \frac{\frac{1}{nh^d} \sum_{i=1}^n W\left(\frac{X_i - x}{h}\right) \mathbb{1}_{\{Y_i=k\}}}{\frac{1}{nh^d} \sum_{i=1}^n W\left(\frac{X_i - x}{h}\right)}$$

Da $\hat{\tilde{\delta}}_k(x)$ aus $\hat{\delta}_k(x)$ durch Multiplikation des Nenners (d. h. einer von k unabhängigen Größe) hervorgeht, erhalten wir die folgende alternative Darstellung von $\hat{f}_{n,h}^{W,klass}$:

Lemma 5.14 Es gilt:

$$\hat{f}_{n,h}^{W,klass}(x) = \underset{k\in\{1,\ldots,K\}}{\arg\max}\, \hat{\delta}_k(x), \qquad x \in \mathcal{X},$$

wobei

$$\hat{\delta}_k(x) := \frac{\sum_{i=1}^n W\left(\frac{X_i - x}{h}\right) \mathbb{1}_{\{Y_i=k\}}}{\sum_{i=1}^n W\left(\frac{X_i - x}{h}\right)}$$

Im Gegensatz zur Formulierung in Definition 5.12 hat $\hat{\delta}_k$ direkt die Form eines Kern-Regressionsschätzers, und Eigenschaften können aus Satz 5.9 abgeleitet werden.

Wir geben im Folgenden theoretische Resultate für das Excess Bayes Risk von $\hat{f}_{n,h}^{W,klass}(x)$ im Falle zweier Klassen ($K = 2$) mit $\mathcal{Y} = \{1, 2\}$ an. Die Bayes-Regel bzgl. der 0-1-Verlustfunktion ist dann wegen $\sum_{k=1}^2 \mathbb{P}(Y = k|X = x) = 1$ von der Form

$$f^*(x) = \underset{k\in\{1,2\}}{\arg\max}\, \mathbb{P}(Y = k|X = x) = \begin{cases} 1, & \delta_1^*(x) \geq \frac{1}{2} \\ 2, & \delta_1^*(x) < \frac{1}{2} \end{cases}, \quad x \in \mathcal{X}.$$

Analog gilt $\sum_{k=1}^2 \hat{\delta}_k(x) = 1$ und damit

$$\hat{f}_{n,h}^{W,class}(x) = \begin{cases} 1, & \hat{\delta}_1(x) \geq \frac{1}{2} \\ 2, & \hat{\delta}_1(x) < \frac{1}{2} \end{cases}, \quad x \in \mathcal{X}. \tag{5.10}$$

Mit Hilfe des Lemmas 3.5 und den Aussagen für den Kern-Regressionsschätzer (Satz 5.9) können die Eigenschaften von $\hat{\delta}_1$ auf $\hat{f}_{n,h}^{W,klass}$ übertragen werden. Zur Formalisierung stellen wir folgende allgemeine Modellannahme an f^* bzw. die damit verknüpfte optimale Diskriminantenfunktion und die Dichte g von X:

Modellannahme 5.15 (Nichtparametrische Klassifikation) Es sei $\mathcal{X}$ kompakt. Es gebe $C > 0$, so dass $\delta_1^*(x) = \mathbb{P}(Y = 1|X = x)$ erfüllt: $\delta_1^* \in \mathcal{F}_C$. Es gelte $g \in \mathcal{F}_C$ und $\inf_{x\in\mathcal{X}} g(x) > 0$.

Wir erhalten folgende Aussage über das Excess Bayes Risk von $\hat{f}_{n,h}^{W,klass}$:

Satz 5.16 (Excess Bayes Risk von Kern-Dichte-Klassifizierern) Es sei $K = 2$ und $\mathcal{Y} = \{1, 2\}$. Es gelte die Modellannahme 5.15. Weiter sei W eine Kernfunktion mit $\int_{\mathbb{R}^d} W(u)^2 \, \mathrm{d}u < \infty$ und $\int_{\mathbb{R}^d} |u| W(u) \, \mathrm{d}u < \infty$.

Dann gibt es eine von n, h, d unabhängige Konstante $c > 0$, so dass für alle $x \in \mathcal{X}$ gilt:

$$\mathbb{E}R\left(\hat{f}_{n,h}^{W,klass}\right) - R(f^*) \leq c \cdot \left[\frac{1}{nh^d} + h^2\right]^{1/2}$$

Beweis Wegen Gl. (5.10) und Lemma 3.5 gilt:

$$\mathbb{E}R(\hat{f}_{n,h}^{W,klass}) - R(f^*) \leq 2\mathbb{E}\left[(\hat{\delta}_1(X) - \delta_1^*(X))^2\right]^{1/2} = 2\mathbb{E}\left[\mathbb{E}\left[(\hat{\delta}_1(X) - \delta_1^*(X))^2|X\right]\right]^{1/2} \tag{5.11}$$

Mit der Definition $Z := \mathbb{1}_{\{Y=1\}}$ gilt für alle $x \in \mathcal{X}$: $\mathbb{E}[Z|X = x] = \mathbb{P}(Y = 1|X = x) = \delta_1^*(x)$, d. h., die Trainingsdaten (X_i, Z_i) mit $Z_i := \mathbb{1}_{\{Y_i=1\}}$ folgen einem Regressionsmodell mit Bayes-Regel δ_1^*. Offensichtlich gilt für $\varepsilon_i = Z_i - \delta_1^*(X_i)$: $|\varepsilon_i| \leq 2$. Der innere bedingte Erwartungswert auf der rechten Seite von Gl. (5.11) entspricht damit dem *mean squared error* eines Kern-Regressionsschätzers. Laut Satz 5.9 gibt es ein $c > 0$, so dass für alle $x \in \mathcal{X}$ gilt:

$$\mathbb{E}\left[(\hat{\delta}_1(x) - \delta_1^*(x))^2\right] \leq c \cdot \left[\frac{1}{nh^d} + h^2\right]$$

Es folgt die Behauptung.

Aufgrund der gleichen Form des Excess Bayes Risk treffen alle Bemerkungen zum Kern-Regressionsschätzer auch auf Kern-Dichte-Klassifizierer zu, insbesondere leidet auch $\hat{f}_{n,h}^{W,klass}$ unter dem „Fluch der Dimension".

5.2.3 Der naive Bayes-Klassifizierer

Neben den hier kennengelernten Kern-Regressionsschätzern und Kern-Dichte-Klassifizierern gibt es noch viele weitere Ansätze und Methoden der nichtparametrischen Statistik zur Bestimmung von f^* unter der Modellannahme $f^* \in \mathcal{F}_C$ bzw. $\delta_1^* \in \mathcal{F}_C$. Eine weitere prominente Methode zur Schätzung von f^* in Regressionsproblemen ist beispielsweise der Reihen-Projektionsschätzer (engl. *orthogonal series estimator*), welcher f^* als Element des Hilbertraums der quadratintegrierbaren Funktionen auffasst, f^* als

unendliche Linearkombination einer geeigneten Orthonormalbasis darstellt und dann versucht, die Linearfaktoren zu schätzen. Die bei d Dimensionen entstehende optimale Konvergenzrate des Excess Bayes Risk von $n^{-\frac{1}{d+2}}$ ist jedoch keine Eigenheit von Kernschätzern, sondern tritt auch bei Reihen-Projektionsschätzern auf. Man kann sogar zeigen (vgl. [34, Minimax-Theorie]), dass unter bestimmten Annahmen keine schnellere Konvergenzrate des Excess Bayes Risk unter der Modellannahme $f^* \in \mathcal{F}_C$ möglich ist.

Um mit nichtparametrischen Methoden wie den Kernschätzern also bessere Raten als $n^{-\frac{1}{d+2}}$ für das Excess Bayes Risk zu erreichen, muss die Modellannahme modifiziert werden. Wir diskutieren hier ein Beispiel für die Klassifikation anhand des *naiven Bayes-Klassifizierers*. Dieser Algorithmus geht davon aus, dass die *Komponenten* $X_{i1}, ..., X_{id}$ der Daten X_i in jeder Klasse $Y_i = k$ stochastisch unabhängig sind. Sind $g_{kj} := f_{X_j|Y=k}$ die bedingten Dichten von X_j gegeben $Y = k$, so fordern wir formal gemäß folgender Definition:

Modellannahme 5.17 (Bayes-Klassifizierer) Es sei $\mathcal{X}$ kompakt. Für $k = 1, ..., K$ gelte

$$g_k(x) = \prod_{j=1}^{d} g_{kj}(x_j), \quad x = (x_1, ..., x_d)^T \in \mathbb{R}^d,$$

und es gebe $C > 0$ mit $g_{k1}, ..., g_{kd} \in \mathcal{F}_C$. Es gelte $\pi_k \in (0, 1)$, und es gebe $c_{g,min} > 0$ mit $\inf_{x \in \mathcal{X}} g_{kj}(x) \geq c_{g,min}$ $(k = 1, ..., K, j = 1, ..., d)$.

Dies bedeutet *nicht,* dass die Komponenten von X unabhängig sind. Zum besseren Verständnis dieses Unterschieds und der Modellannahme geben wir ein anschauliches Beispiel mit $K = 2$ Klassen:

Beispiel 5.18 Es seien $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2\}$. Anschaulich interpretieren wir die Trainingsdaten (X_i, Y_i), $i = 1, ..., n$ als Messresultate von der Lautstärke in Dezibel (Komponente X_{i1}) und der Flughöhe in Metern (Komponente X_{i2}) von Silvesterraketen. Es gebe zwei verschiedene Marken von Raketen, welche durch $Y_i = 1$ und $Y_i = 2$ angegeben werden, es gelte $\mathbb{P}(Y_1 = 1) = \mathbb{P}(Y_1 = 2) = \frac{1}{2}$.

(i) Verhalten sich alle Raketen einer Marke gleichartig, so ist es naheliegend, dass die Lautstärke und die Flughöhe von Raketen einer Marke unabhängig sind, d. h., formal sind die Komponenten X_{i1}, X_{i2} unabhängig gegeben Y_i. Eine beispielhafte Modellierung ist durch

$$(X_1|Y_1 = 1) \sim N\left(\begin{pmatrix}60\\40\end{pmatrix}, \begin{pmatrix}4 & 0\\0 & 3\end{pmatrix}\right), \qquad (X_1|Y_1 = 2) \sim N\left(\begin{pmatrix}80\\50\end{pmatrix}, \begin{pmatrix}6 & 0\\0 & 4\end{pmatrix}\right)$$

gegeben, eine Darstellung von $n = 100$ Trainingsdaten ist in Abb. 5.5a zu sehen. Damit entspricht diese Modellierung der Modellannahme 5.17. Ist die Klasse Y_i jedoch unbekannt, so sind X_{i1}, X_{i2} nicht unabhängig: Eine Beobachtung X_i mit $X_{i1} = 82$ impliziert mit hoher Wahrscheinlichkeit $Y_i = 2$ und damit $X_{i2} \approx 50$; ein Wert $X_{i1} \approx 40$ ist dagegen unwahrscheinlich.

(ii) Für ein zweites Beispiel nehmen wir nun an, dass Marke 1 gemischt aus zwei Raketenarten besteht, als beispielhafte Modellierung betrachten wir

$$(X_1|Y_1 = 1) \sim \frac{1}{2}N\left(\begin{pmatrix}60\\40\end{pmatrix}, \begin{pmatrix}4&0\\0&3\end{pmatrix}\right) + \frac{1}{2}N\left(\begin{pmatrix}80\\50\end{pmatrix}, \begin{pmatrix}4&0\\0&3\end{pmatrix}\right),$$
$$(X_2|Y_2 = 2) \sim N\left(\begin{pmatrix}60\\50\end{pmatrix}, \begin{pmatrix}4&0\\0&6\end{pmatrix}\right).$$

In diesem Fall ist die Modellannahme 5.17 verletzt, denn X_i in der Klasse $Y_i = 1$ verhält sich so wie in (i) alle X_i gemeinsam. Ein Beispiel für $n = 100$ Trainingsdaten ist in Abb. 5.5b zu sehen.

Die Modellannahme 5.17 trifft also anschaulich zu, wenn alle gemessenen Eigenschaften der Objekte einer Klasse unabhängig voneinander sind.

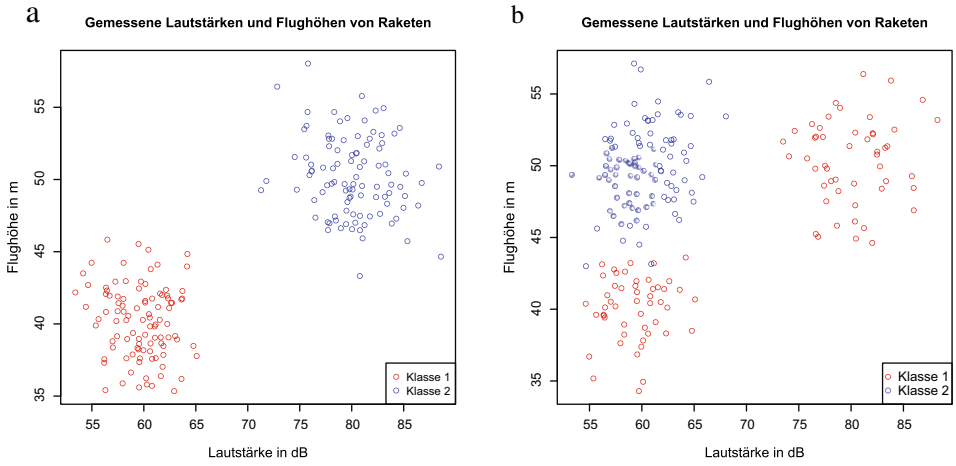


Abb. 5.5 Verdeutlichung der Modellannahme 5.17 vom naiven Bayes-Klassifizierer mittels der Trainingsdaten aus Beispiel 5.18(i), (ii). **a** Darstellung der Trainingsdaten aus (i), die Modellannahme ist erfüllt. **b** Die Modellannahme ist nicht erfüllt

Durch eine alternative Formulierung der Modellannahme am Ende des Kapitel (vgl. Bemerkung 5.21) werden wir die Modellannahme noch einmal aus einer grafischen Perspektive formulieren.

Der naive Bayes-Klassifizierer hat seinen Namen von der Tatsache, dass die Modellannahme 5.17 eine Annahme an die Priori-Verteilung $g_k = f_{X|Y=k}$ stellt und diese „naiv" ist in dem Sinne, dass die Komponenten der X_i gegeben die Klasse $Y_i = k$ unabhängig sein sollen. Beim historisch ersten Ansatz dieser Art wurde zusätzlich verlangt, dass die g_k Normalverteilungen folgen, womit eine gewisse Nähe des Algorithmus zur Linearen Diskriminanzanalyse vorliegt. Wir werden die g_k jedoch wie zuvor mit Kern-Dichteschätzern ermitteln:

Definition 5.19 (Naiver Bayes-Klassifizierer)
Sei $h > 0$. Sei $W : \mathbb{R} \to \mathbb{R}$ eine Kernfunktion. Definiere

$$\hat{\pi}_k = \frac{n_k}{n}, \qquad n_k = \#\{i : Y_i = k\}, \qquad k = 1, ..., K.$$

Sei

$$\hat{g}_{kj,n,h}(x_j) = \frac{1}{n_k h} \sum_{i:Y_i=k} W\left(\frac{X_{ij} - x_j}{h}\right)$$

und $\hat{g}_{k,n,h}(x) = \prod_{j=1}^{d} \hat{g}_{kj,n,h}(x_j)$. Sei $\hat{\tilde{\delta}}_k(x) = \hat{\pi}_k \cdot \hat{g}_{k,n,h}(x)$. Dann heißt

$$\hat{f}_{n,h}^{W,naiv}(x) := \underset{k\in\{1,...,K\}}{\arg\max}\ \hat{\tilde{\delta}}_k(x)$$

naiver Bayes-Klassifizierer. ♦

In Abb. 5.6 ist das Verhalten von $\hat{f}_{n,h}^{W,naiv}$ mit jeweils durch Cross Validation gewähltem h für die Trainingsdaten aus Beispiel 5.13(i), (ii) dargestellt. Während in (i) die Modellannahme erfüllt ist und eine adäquate Schätzung des optimalen Entscheidungsrands erfolgt, liefert $\hat{f}_{n,h}^{W,naiv}$ in (ii) systematisch falsche Resultate. In (ii) ist die Modellannahme nicht erfüllt, vgl. dazu das Beispiel 5.18(ii) aus wahrscheinlichkeitstheoretischer Sicht oder Lemma 5.21 aus grafischer Sicht.

Da die einzelnen $\hat{g}_{kj,n,h}$ nur noch die eindimensionalen Wahrscheinlichkeitsdichten g_{kj} mit den eindimensionalen Beobachtungen X_{ij} schätzen, ist deren Konvergenzrate in n unabhängig von d. Wir erwarten daher auch für das Excess Bayes Risk von $\hat{f}_{n,h}^{W,naiv}$ eine Konvergenzrate, die wesentlich weniger stark von d abhängt. Wir geben hier ein einfaches theoretisches Resultat im Falle von $K = 2$ Klassen.

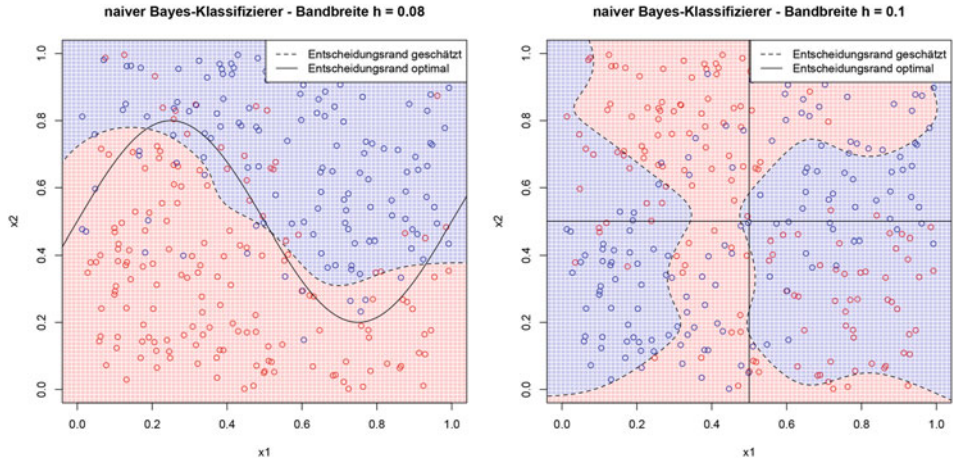


Abb. 5.6 Anwendung von $\hat{f}_{n,h}^{W,naiv}$ auf jeweils $n = 300$ Trainingsdaten aus Beispiel 5.13(i), (ii) unter Nutzung des zweidimensionalen Gauß-Kerns mit durch Cross Validation gewählter Bandbreite h

Satz 5.20 Es sei $\mathcal{X} = [0,1]^d \subset \mathbb{R}^d$ kompakt, $K = 2$ und $\mathcal{Y} = \{1,2\}$. Es gelte die Modellannahme Definition 5.17. W sei eine Kernfunktion mit $\int_{\mathbb{R}} W(u)^2 \, \mathrm{d}u < \infty$ und $\int_{\mathbb{R}} |u| W(u) \, \mathrm{d}u < \infty$. Dann gilt für alle $d = d_n$ und $h = h_n \to 0$ mit $\frac{nh}{\log(n)} \to \infty$:

$$\limsup_{c\to\infty} \limsup_{n\to\infty} \mathbb{P}\left(R(\hat{f}_{n,h}^{W,naiv}) - R(f^*) \geq c \cdot \gamma_n(h) \right),$$

wobei $\gamma_n(h) = n^{-1/2} + (\frac{d}{nh})^{1/2} + dh$. Ist insbesondere $h = h^* = (dn)^{-1/3}$, so gilt

$$\gamma_n(h^*) \leq 2d^{2/3} n^{-1/3}.$$

Beweis Wir nutzen die Abkürzungen $\hat{g}_{kj} := \hat{g}_{kj,n,h}$ und $\tilde{T}_n := (X_i)_{i=1,\ldots,n}$. Mit Lemma 3.6 erhalten wir

$$R\left(\hat{f}_{n,h}^{W,naiv}\right) - R(f^*) \leq 2 \sum_{k=1,2} \mathbb{E}\left[\left| \frac{\hat{\pi}_k \hat{g}_k(X)}{g(X)} - \frac{\pi_k g_k(X)}{g(X)} \right| \middle| \tilde{T}_n \right]. \tag{5.12}$$

Die Abbildung $\hat{g}_k(x)$ ist eine Dichte in x bzgl. des Lebesgue-Maßes, denn $\int \hat{g}_k(x) \, \mathrm{d}x = \prod_{j=1}^d \int \hat{g}_{kj}(x_j) \, \mathrm{d}x_j$ und

$$\begin{aligned} \int \hat{g}_{kj}(x_j) \, \mathrm{d}x_j &= \int \hat{g}_{kj}(x_j) \, \mathrm{d}x_j = \frac{1}{nh\hat{\pi}_k} \sum_{i=1}^n \mathbb{1}_{\{Y_i = k\}} \int K\left(\frac{X_{ij} - x_j}{h} \right) \mathrm{d}x_j \\ &= \frac{1}{n\hat{\pi}_k} \sum_{i=1}^n \mathbb{1}_{\{Y_i = k\}} = 1. \end{aligned}$$

Damit ist $\mathbb{E}[\frac{\hat{g}_k(X)}{g(X)}] = \int \hat{g}_k(x)\,\mathrm{d}x = 1$ und

$$\mathbb{E}\left[\left|\frac{\hat{\pi}_k\hat{g}_k(X)}{g(X)} - \frac{\pi_k g_k(X)}{g(X)}\right| \middle| \tilde{T}_n\right] \leq |\hat{\pi}_k - \pi_k| + \pi_k \cdot \mathbb{E}\left[\left|\frac{\hat{g}_k(X)}{g(X)} - \frac{g_k(X)}{g(X)}\right| \middle| \tilde{T}_n\right]. \quad (5.13)$$

Sei $\mathbb{Y} = (Y_i)_{i=1,\ldots,n}$. Wir erhalten die Bias-Varianz-Zerlegung

$$\begin{aligned}
&\mathbb{E}\left[\left|\frac{\hat{g}_k(X)}{g(X)} - \frac{g_k(X)}{g(X)}\right| \middle| \tilde{T}_n\right] \\
&= \int |\hat{g}_k(x) - g_k(x)|\,\mathrm{d}x \\
&\leq \int |\hat{g}_k(x) - \mathbb{E}[\hat{g}_k(x)|\mathbb{Y}]|\,\mathrm{d}x + \int |\mathbb{E}[\hat{g}_k(x)|\mathbb{Y}] - g_k(x)|\,\mathrm{d}x =: V_n + B_n.
\end{aligned} \quad (5.14)$$

Wir erhalten aus Gl. (5.12), (5.13) und (5.14):

$$\begin{aligned}
&\mathbb{P}\left(R(\hat{f}_{n,h}^{W,naiv}) - R(f^*) \geq c\gamma_n\right) \\
&\leq \sum_{k=1,2} \left\{\mathbb{P}(|\hat{\pi}_k - \pi_k| \geq \frac{c}{4}n^{-1/2}) + \mathbb{P}(B_n \geq \frac{c}{4\pi_k}dh) + \mathbb{P}(V_n \geq \frac{c}{4\pi_k}(\frac{d}{nh})^{1/2})\right\}
\end{aligned} \quad (5.15)$$

Anwendung der Tschebyscheff-Ungleichung auf den ersten Summanden und $\mathbb{E}[(\hat{\pi}_k - \pi_k)^2] \leq \frac{\pi_k(1-\pi_k)}{n}$ liefert $\limsup_{c,n\to\infty} \mathbb{P}(|\hat{\pi}_k - \pi_k| \geq \frac{c}{4}n^{-1/2}) = 0$. Wegen

$$\begin{aligned}
B_n &\leq \sum_{j=1}^{d} \int \left(\prod_{j'<j} g_{kj'}(x_{j'})\right) \cdot |\mathbb{E}[\hat{g}_{kj}(x_j)|\mathbb{Y}] - g_{kj}(x_j)| \cdot \left(\prod_{j'>j} \mathbb{E}[\hat{g}_{kj',n,h}(x_{j'})|\mathbb{Y}]\right) \mathrm{d}x \\
&= \sum_{j=1}^{d} \int |\mathbb{E}[\hat{g}_{kj}(x_j)|\mathbb{Y}] - g_{kj}(x_j)|\,\mathrm{d}x_j \overset{g_{kj}\in\mathcal{F}_C}{\leq} Cdh \int |u|K(u)\,\mathrm{d}u
\end{aligned}$$

gilt $\limsup_{n\to\infty} \mathbb{P}(B_n \geq \frac{c}{8\pi_k}dh) = 0$ für c groß genug.

Der Rest des Beweises beschäftigt sich mit dem Varianzterm V_n. Die Schwierigkeit ist, anders als beim Biasterm, nur den Faktor $\sqrt{d}$ anstatt d zu erhalten. V_n ist die Totalvariation zwischen den Dichten $\hat{g}_k$ und $\mathbb{E}[\hat{g}_k|\mathbb{Y}]$ und kann durch deren Hellinger-Abstand abgeschätzt werden (vgl. [34, Definition 2.3 und Lemma 2.3]). Wir erhalten mit $\hat{g}_k(x) = \prod_{j=1}^{d} \hat{g}_{kj}(x_j)$ und den Eigenschaften des Hellinger-Abstands:

$$
\begin{aligned}
V_n^2 &= \left(\int \left| \hat{g}_k(x) - \mathbb{E}[\hat{g}_k(x)|\mathbb{Y}] \right| \, \mathrm{d}x \right)^2 \leq \int \left| \sqrt{\hat{g}_k(x)} - \sqrt{\mathbb{E}[\hat{g}_k(x)|\mathbb{Y}]} \right|^2 \, \mathrm{d}x \\
&= 2 \left(1 - \prod_{j=1}^{d} \left(1 - \frac{1}{2} \int \left| \sqrt{\hat{g}_{kj}(z)} - \sqrt{\mathbb{E}[\hat{g}_{kj}(z)|\mathbb{Y}]} \right|^2 \, \mathrm{d}z \right) \right) \\
&\leq \sum_{j=1}^{d} \int \left| \sqrt{\hat{g}_{kj}(z)} - \sqrt{\mathbb{E}[\hat{g}_{kj}(z)|\mathbb{Y}]} \right|^2 \, \mathrm{d}z, \qquad (5.16)
\end{aligned}
$$

wobei die letzte Ungleichung aus $|\prod_{j=1}^d a_j - \prod_{j=1}^d b_j| \leq \sum_{j=1}^d |a_j - b_j|$ für $|a_j|, |b_j| \leq 1$ folgt. Im Folgenden zeigen wir mit Hilfe obiger Abschätzung $\mathbb{E}[V_n^2 \mathbb{1}_{A_n}] \leq c' \cdot \frac{d}{nh}$, wobei $A_n := \{\hat{\pi}_k \geq \frac{\pi_k}{2}\}$. Wegen $\limsup_{n\to\infty} \mathbb{P}(A_n)^c \leq \limsup_{n\to\infty} \mathbb{P}(|\hat{\pi}_k - \pi_k| \geq \frac{\pi_k}{2}) = 0$ und der Markov-Ungleichung folgt dann $\limsup_{c\to\infty} \limsup_{n\to\infty} \mathbb{P}(V_n \geq \frac{c}{4\pi_k}(\frac{d}{nh})^{1/2}) = 0$ und damit durch Gl. (5.15) die Behauptung des Satzes.

Zum Ausschließen unerwünschter Ereignisse definieren wir $A_n(z) := \{|\hat{g}_{kj}(z) - \mathbb{E}[\hat{g}_{kj}(z)|\mathbb{Y}]| \leq \frac{c_{g,min}}{4}, |\mathbb{E}[\hat{g}_{kj}(z)|\mathbb{Y}] - g_{kj}(z)| \leq \frac{c_{g,min}}{4}\}$. Mit Gl. (5.16), der Lipschitz-Stetigkeit der Wurzel auf $[\frac{c_{g,min}}{2}, \infty)$ und $|\sqrt{a} - \sqrt{b}|^4 \leq a^2 + b^2 \leq 3((a-b)^2 + b^2)$ folgt

$$
\begin{aligned}
\mathbb{E}[V_n^2 \mathbb{1}_{A_n}] &\leq d \cdot \mathbb{E}\Big[\int \mathbb{E}\Big[\left| \sqrt{\hat{g}_{k1}(z)} - \sqrt{\mathbb{E}[\hat{g}_{k1}(z)|\mathbb{Y}]} \right|^2 \Big| \mathbb{Y} \Big] \, \mathrm{d}z \cdot \mathbb{1}_{A_n} \Big] \\
&\leq \frac{d}{2c_{g,min}} \cdot \mathbb{E}\Big[\int \mathbb{E}[(\hat{g}_{k1}(z) - \mathbb{E}[\hat{g}_{k1}(z)|\mathbb{Y}])^2 | \mathbb{Y}] \, \mathrm{d}z \cdot \mathbb{1}_{A_n} \Big] \\
&\quad + 3d \cdot \mathbb{E}\Big[\int Z_n \mathbb{P}(A_n(z)^c|\mathbb{Y})^{1/2} \, \mathrm{d}z \cdot \mathbb{1}_{A_n} \Big] \qquad (5.17)
\end{aligned}
$$

mit $Z_n(z) := \big(\mathbb{E}[(\hat{g}_{k1}(z) - \mathbb{E}[\hat{g}_{k1}(z)|\mathbb{Y}])^2|\mathbb{Y}] + \mathbb{E}[\hat{g}_{k1}(z)|\mathbb{Y}]^2\big)^{1/2}$. Auf A_n gilt $\mathbb{E}[(\hat{g}_{k1}(z) - \mathbb{E}[\hat{g}_{k1}(z)|\mathbb{Y}])^2|\mathbb{Y}] \leq \frac{2}{\pi_k nh} \int W(u)^2 g_{k1}(z+uh) \, \mathrm{d}u$ (vgl. Beweis zu Satz 5.9) und mittels der Bernstein-Ungleichung:

$$
\begin{aligned}
\mathbb{P}(A_n(z)^c|\mathbb{Y}) &\overset{h\ klein}{=} \mathbb{P}\left(|\hat{g}_{kj}(z) - \mathbb{E}[\hat{g}_{kj}(z)|\mathbb{Y}]| \leq \tfrac{c_{g,min}}{4} \right) \\
&\leq 2 \exp\left(- \frac{\frac{1}{2}(\frac{c_{g,min}}{4})^2 nh}{\int W(u)^2 g_{k1}(z+uh) \, \mathrm{d}u + \frac{1}{3} \frac{c_{g,min} c_{K,max}}{4nh}} \right) \leq 2n^{-2}
\end{aligned}
$$

für $\frac{nh}{\log(n)}$ groß genug. Die Terme $z \mapsto \int W(u)^2 g_{k1}(z+uh) \, \mathrm{d}u$ und $z \mapsto Z_n(z)$ sind integrierbar in z. Mit Gl. (5.17) erhalten wir $\mathbb{E}[V_n^2 \mathbb{1}_{A_n}] \leq c' \cdot \frac{d}{nh}$ mit einer Konstanten $c' > 0$.

Damit besitzt der naive Bayes-Klassifizierer mit $d^{2/3} n^{-1/3}$ eine wesentlich schnellere Konvergenzrate als normale Kern-Dichte-Klassifizierer. Solange $d \ll n^{1/2}$ gilt und die Modellannahme erfüllt ist, konvergiert $\hat{f}_{n,h}^{W,naiv}$ für immer mehr Trainingsdaten n gegen das Bayes-Risiko.

Zum besseren grafischen Verständnis der Modellannahme des naiven Bayes-Klassifizierers zeigen wir eine Folgerung für die optimalen Diskriminantenfunktionen:

Lemma 5.21 Gilt die Modellannahme aus Definition 5.17, so haben die optimalen Diskriminantenfunktionen $\delta_k^\circ(x) := \log\left(\frac{\mathbb{P}(Y=k|X=x)}{\mathbb{P}(Y=K|X=x)}\right)$, $k = 1, ..., K$ die Form

$$\delta_k^\circ(x) = \begin{cases} \beta_k + \sum_{j=1}^d h_{kj}(x_j), & k = 1, ..., K-1 \\ 0, & k = K \end{cases}, \qquad x = (x_1, ..., x_d)^T \in \mathbb{R}^d \tag{5.18}$$

mit $\beta_k \in \mathbb{R}$ und $h_{kj} \in \mathcal{F}_{\tilde{C}}$ $(j = 1, ..., d, k = 1, ..., K)$ mit geeignetem $\tilde{C} > 0$. Speziell im Fall $K = 2$ ergibt sich für den optimalen Entscheidungsrand:

$$\partial\Omega_1^* = \partial\Omega_2^* = \{x \in \mathcal{X} : \delta_1^\circ(x) = \delta_2^\circ(x)\} = \{x \in \mathcal{X} : \beta_1 + \sum_{j=1}^d h_{1j}(x_j) = 0\}$$

Beweis Mittels des Satzes von Bayes gilt $\mathbb{P}(Y = k|X = x) = \frac{\pi_k g_k(x)}{g(x)}$. Einsetzen in δ_k° liefert für $k = 1, ..., K-1$:

$$\delta_k^\circ(x) = \log\left(\frac{\pi_k g_k(x)}{\pi_K g_K(x)}\right) = \log\left(\frac{\pi_k}{\pi_K}\right) + \sum_{j=1}^d \log\left(\frac{g_{kj}(x_j)}{g_{Kj}(x_j)}\right)$$

und damit die gewünschte Struktur.

Bemerkungen

- Ist beispielsweise h_d bijektiv mit differenzierbarer Umkehrfunktion, so fordern wir im Fall $K = 2$, dass der optimale Entscheidungsrand $\partial\Omega_1^* = \{x \in \mathcal{X} : x_d = h_d^{-1}(-\beta_1 - \sum_{j=1}^{d-1} h_{1j}(x_j))\}$ Graph einer differenzierbaren Funktion in $(x_1, ..., x_{d-1})$ ist, bei welcher die einzelnen Komponenten von x nur auf sehr eingeschränkte Weise miteinander interagieren. Im Fall $d = 2$ muss sich der Entscheidungsrand notwendig als Graph einer differenzierbaren Funktion darstellen lassen, damit sind keine „Sprünge“ oder Löcher des Entscheidungsrands möglich, wie dies beispielsweise in Abb. 5.6b der Fall ist.
- Im Vergleich zu logistischer Regression, wo eine lineare Struktur

$$\log\left(\frac{\mathbb{P}(Y = k|X = x)}{\mathbb{P}(Y = K-1|X = x)}\right) = \sum_{j=1}^d \beta_j^{(k)} x_j$$

vorausgesetzt wurde, lässt die Modellannahme des naiven Bayes-Klassifizierers eine beliebige funktionale Abhängigkeit von x_j in den einzelnen Summanden zu. Diese Verallgemeinerung nennt man *additives Modell* und ist auch in den folgenden Modellen ein wichtiger Spezialfall.

Ist die Modellannahme des naiven Bayes-Klassifizierers nicht zu stark verletzt (d. h., die optimalen Diskriminantenfunktionen lassen sich zumindest gut durch Gl. (5.18) approximieren), so liefert $\hat{f}_{n,h}^{W,naiv}$ weiterhin gute Ergebnisse. Der Grund ist, dass durch die Verletzung der Modellannahme zwar ein höherer Approximationsfehler und damit verbunden ein höherer Bias auftritt, aber die Schätzung aufgrund des ‚relativ' kleinen Raums möglicher Diskriminantenfunktionen eine kleine Varianz besitzt. Diese Eigenschaft macht den naiven Bayes-Klassifizierer auch heute noch attraktiv in praktischen Anwendungen.

Regressions- und Klassifikationsbäume; Bagging, Boosting und Random Forests

6

Inhaltsverzeichnis

In diesem Kapitel behandeln wir eine weitere Klasse von nichtparametrischen Algorithmen, die sogenannten *Bäume.* Durch ihre schnelle Berechenbarkeit und die darauf aufbauenden Erweiterungen sind Bäume in der Praxis sehr beliebt. Die einfache Struktur erlaubt sowohl die Anwendung auf Regressions- als auch Klassifikationsprobleme. Da außer Differenzierbarkeitsannahmen keine Voraussetzungen an die Bayes-Regel f^* (bzw. eine zugehörige optimale Diskriminantenfunktion) gestellt werden, unterliegen zumindest die zunächst eingeführten Basisalgorithmen weiterhin dem Fluch der Dimension. Wir zeigen dies mathematisch rigoros am Spezialfall der dyadischen Bäume.

Die in diesem Kapitel eingeführten Techniken des Bagging, Boosting und der Random Forests verbessern die Basisalgorithmen so, dass bei Vorliegen bestimmter Strukturen von f^* bessere Konvergenzraten nachgewiesen werden können.

6.1 Binäre Bäume

In diesem Kapitel motivieren wir zunächst die Verbindung zwischen Bäumen und Entscheidungsregeln. Wir beschränken uns hier und im Weiteren auf sogenannte *binäre Bäume*.

S. Richter, *Statistisches und maschinelles Lernen,*
https://doi.org/10.1007/978-3-662-59354-7_6

Die Grundidee zum Finden einer geeigneten Entscheidungsregel basierend auf gegebenen Trainingsdaten $(X_i, Y_i), i = 1, \ldots, n$ ist hier wie folgt: Teile die Daten X_i (bzw. den Raum $\mathcal{X} \subset \mathbb{R}^d$) in zwei Hälften entlang einer Koordinate $j \in \{1, \ldots, d\}$ auf, so dass die Mengen $\{Y_i : X_{ij} < s\}$ und $\{Y_i : X_{ij} \geq s\}$ möglichst gut durch Konstanten c_1, c_2 approximiert werden können. Auf den beiden neu erhaltenen Teilmengen wird die Prozedur wiederholt. Stoppt man das Vorgehen nach einer gewissen Anzahl an Aufteilungen, erhält man eine Zerlegung des Raumes, bei welcher jedem einzelnen Teilraum ein Wert zugewiesen ist, der die dort enthaltenen Trainingsdaten Y_i möglichst gut approximiert. In Abb. 6.1 ist dieses Prinzip für einige Trainingsdaten eines Regressionsproblems für die erste Aufteilung und die darauf folgenden zwei weiteren Aufteilungen dargestellt.

Die sukzessive Aufteilung des Raumes $\mathcal{X}$ entlang jeweils einer Koordinate kann mit einer Baumstruktur formalisiert werden, bei welcher die erste Aufteilung der Wurzel des Baumes entspricht, von welcher dann zwei Knoten ausgehen (die entstandenen Teilräume), usw. Zur Formalisierung rekapitulieren wir zunächst die Definition eines binären Baums als Spezialfall eines Graphen. Wir setzen hier die Definition eines gerichteten Graphen und die Notation ‚$v_1 \to v_2$' für eine Kante zwischen den Knoten v_1 und v_2 voraus.

Definition 6.1 (Binärer Baum T)
Ein *binärer Baum* $T = (V_T, E_T)$ ist ein gerichteter, azyklischer (d. h., es gibt keine Menge $v_1, \ldots, v_n \in V_T$ mit $v_1 \to v_2 \to \cdots \to v_n \to v_1$) Graph mit folgenden Eigenschaften:

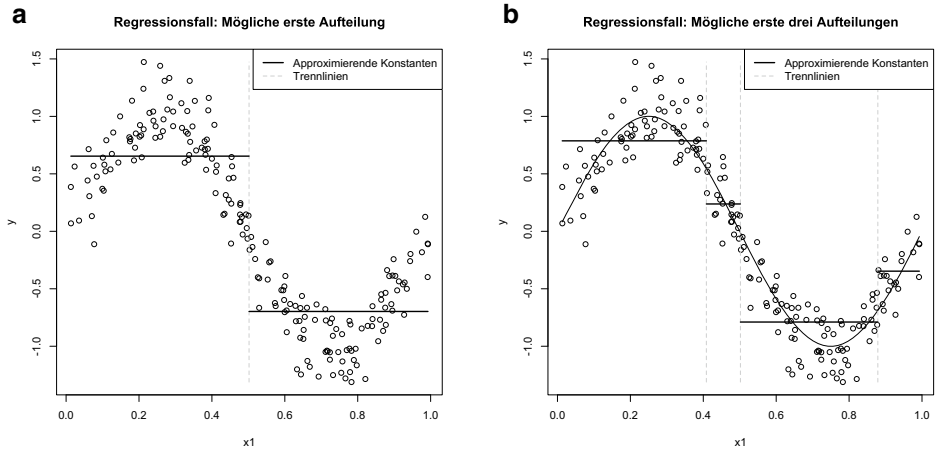


Abb. 6.1 **a** Aufteilung des Raumes $\mathcal{X} = [0, 1]$ in zwei Hälften (markiert durch die Trennlinie bei $s \approx 0{,}5$), so dass in beiden Teilmengen die Daten möglichst gut durch eine Konstante approximiert werden. **b** ein mögliches Resultat nach zwei weiteren Aufteilungen der entstandenen Teilräume $[0, 0{,}5]$ und $[0{,}5, 1]$

- Von jedem Knoten gehen entweder keine Kanten *(Blatt)* oder genau zwei Kanten aus *(innerer Knoten)*.
- Auf jeden Knoten außer einem (der sog. *Wurzel*) zeigt jeweils genau eine Kante.

Die Menge der Blätter bezeichnen wir mit B_T, die Anzahl der Blätter sei $\#T := \#B_T$. ♦

Indem jedem Knoten eine Koordinate $j \in \{1, \ldots, d\}$ und eine Zahl $s \in \mathbb{R}$ und jedem Blatt ein Wert $y \in \mathcal{Y}$ zugewiesen wird, können die Algorithmen aus obiger Grundidee adäquat durch folgende Struktur ausgedrückt werden:

Definition 6.2 (Regressions-/Klassifikationsbaum T)
Sei T ein Baum. Jedem inneren Knoten $v \in V_T \setminus B_T$ seien Werte

- $j(v) \in \{1, \ldots, d\}$ *(split index)* und
- $s(v) \in \mathbb{R}$ *(split point),*

jedem Blatt $v \in B_T$ sei ein Wert $y(v) \in \mathcal{Y}$ zugeordnet. In diesem Fall heißt T *CART* (engl. *Classification And Regression Tree*). T heißt *Regressionsbaum,* falls $\mathcal{Y} = \mathbb{R}$, und *Klassifikationsbaum,* falls $\mathcal{Y} = \{1, \ldots, K\}$. ♦

Der Begriff CART wurde maßgeblich geprägt durch [11]. Wir geben nun ein grafisches Beispiel für CARTs. Die Prozedur zur Erzeugung der Bäume aus den Daten wird in Abschn. 6.3 dargestellt.

Beispiel 6.3

- *Regression:* Es sei $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \mathbb{R}$. Gegeben seien $n = 200$ i. i. d. Trainingsdaten des Modells
$$Y = \sin(2\pi X) + \varepsilon, \qquad \varepsilon \sim N(0, \sigma^2), \qquad X \sim U[0, 1], \tag{6.1}$$
d. h., X ist gleichverteilt auf $[0, 1]$ und unabhängig von ε. Es sei $\sigma = 0{,}2$. In Abb. 6.2 ist ein möglicher Regressionsbaum (abgebrochen nach $7 = 1 + 2 + 4$ Aufteilungen) und dessen grafische Darstellung basierend auf den Trainingsdaten zu sehen.
- *Klassifikation:* Es sei $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2\}$. Es seien X_{i1}, X_{i2} i. i. d. gleichverteilt auf $[0, 1]$, $X_i = (X_{i1}, X_{i2})^T$. Unabhängig davon seien $\varepsilon_i \sim N(0, \sigma^2)$ i. i. d. mit $\sigma > 0$. Es sei
$$Y_i = \begin{cases} 1, & \kappa(X_i) - \varepsilon_i \leq 0 \\ 2, & \kappa(X_i) - \varepsilon_i > 0 \end{cases}, \tag{6.2}$$

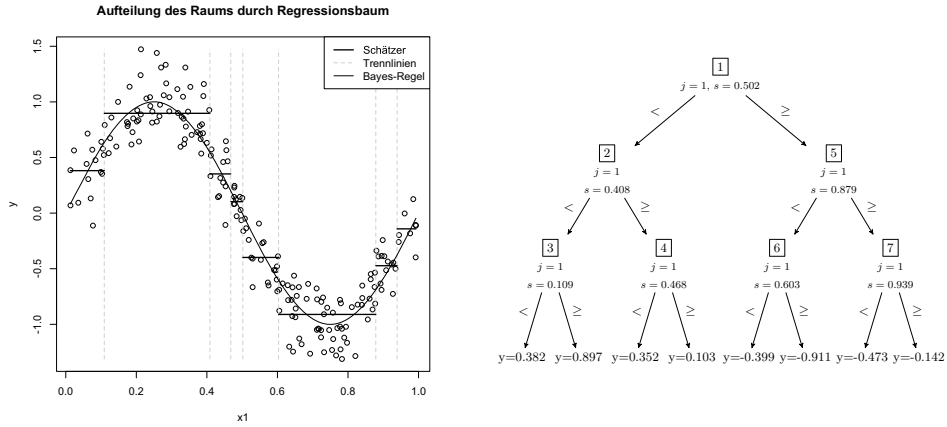


Abb. 6.2 Beispiel: Ein Regressionsbaum basierend auf Trainingsdaten des Modells Gl. (6.1)

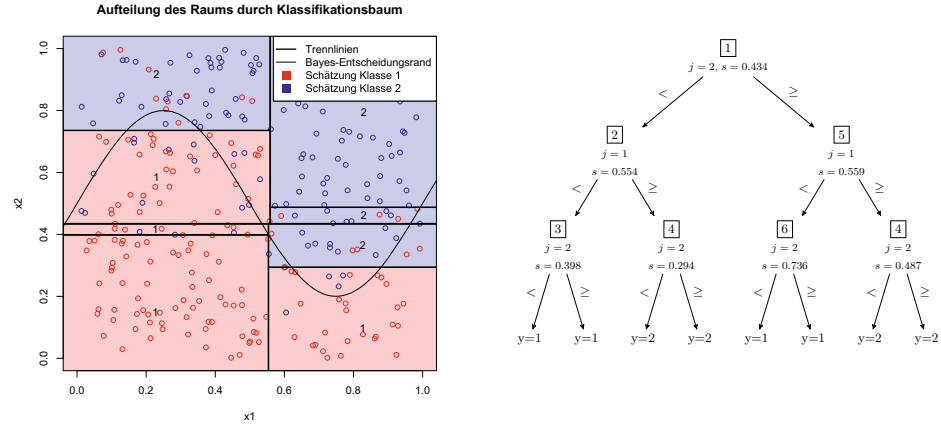


Abb. 6.3 Beispiel: Ein Klassifikationsbaum basierend auf Trainingsdaten des Modells Gl. (6.2)

vgl. Beispiel 5.13. Wir wählen $\sigma = 0{,}2$, $\kappa(x) = x_2 - 0{,}5 - 0{,}3\sin(2\pi x_1)$ für $x = (x_1, x_2)^T \in \mathbb{R}^2$. In Abb. 6.3 ist ein möglicher Klassifikationsbaum (abgebrochen nach sieben Aufteilungen) und dessen grafische Darstellung basierend auf $n = 300$ Beobachtungen zu sehen.

Wir überzeugen uns nun, dass durch obige Definition ein CART T durch seine Blätter $v \in B_T$ tatsächlich eine Partition des Raumes $\mathcal{X}$ induziert. Daraus folgt, dass jeder CART durch die Zuordnung der Werte $y(v)$ auf den Blättern $v \in B_T$ einer Entscheidungsregel $\mathcal{X} \to \mathcal{Y}$ entspricht.

Bemerkung 6.4 (und Definition) Sei T ein CART.

- *Partition des Raumes:* Sei $v_0 \in V_T$ die Wurzel und $A(v_0) := \mathcal{X}$.
  Für jeden inneren Knoten $v \in V_T \setminus B_T$ seien $v_1, v_2 \in V_T$ die Knoten mit $v \to v_1, v \to v_2$.
  Setze

$$A(v_1) := \{x \in A(v) : x_{j(v)} < s(v)\}, \qquad A(v_2) := \{x \in A(v) : x_{j(v)} \geq s(v)\}.$$

  Ist $B_T = \{w_1, \ldots, w_{\#T}\}$, so sind die Mengen $A_m := A(w_m)$, $m = 1, \ldots, \#T$ disjunkt und

$$\bigcup_{m=1}^{\#T} A_i = \mathcal{X},$$

  d. h., sie bilden eine Partition von $\mathcal{X}$. Die Menge $\pi(T) := \{A_1, \ldots, A_{\#T}\}$ heißt von T *erzeugte Partition* des Raums $\mathcal{X}$.
- *Jeder CART induziert eine Entscheidungsregel:* Sei

$$f_T(x) := \sum_{m=1}^{\#T} y(w_m) \cdot \mathbb{1}_{A_m}(x).$$

  Dann ist $f_T : \mathcal{X} \to \mathcal{Y}$ wohldefiniert (da die A_m disjunkt sind, wird jeweils nur ein Summand ausgewählt, und es treten oben de facto gar keine Summen auf) und heißt der von T induzierte *CART-Algorithmus.*

Durch diese Verbindung kann die Suche nach geeigneten Entscheidungsregeln auf die Suche nach geeigneten Bäumen verlagert werden. Um die Größe von Bäumen (d. h. die Anzahl der Aufteilungen des Raums) zu steuern, definieren wir:

Definition 6.5
Sei T ein Baum, v_0 seine Wurzel und $v \in V_T$.

- Seien $v_0, \ldots, v_{n-1} \in V_T$ mit $v_0 \to v_1 \to \cdots \to v_{n-1} \to v$. Wir nennen $t(v) := n$ die *Tiefe des Knotens* v,
- $t(T) := \max_{v \in B_T} t(v)$ die *Tiefe des Baums* T. ♦

In Abschn. 6.2 studieren wir zunächst die theoretischen Eigenschaften von auf Bäumen basierenden Entscheidungsregeln anhand einer Teilmenge aller möglichen CARTs.

6.2 Dyadische Bäume – theoretische Resultate

Die Menge aller möglichen CARTs T mit vorgegebener Tiefe $t_{\max} \in \mathbb{N}$ können durch die Menge

$$\mathcal{T}(t_{\max}) = \{T \text{ ist CART mit Tiefe } t(T) \leq t_{\max}\}$$

beschrieben werden. Entgegen der eingangs motivierten sukzessiven Aufteilung des Raumes durch einen schrittweisen Aufbau eines Baumes ist es daher auch möglich, die Suche nach einem geeigneten Baum direkt als Optimierungsproblem über alle möglichen $T \in \mathcal{T}(t_{\max})$ zu begreifen. Folgen wir dem Standardansatz aus Bemerkung 1.14, sollte ein Algorithmus mit möglichst geringem Risiko bestimmt werden können, indem

$$\hat{T}_n := \operatorname*{arg\,min}_{T \in \mathcal{T}(t_{\max})} \hat{R}_n(f_T), \qquad \hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f(X_i)) \tag{6.3}$$

und dann $\hat{f}_n := f_{\hat{T}_n}$ als der vom Baum induzierte Algorithmus gewählt wird. Der Ansatz ist sowohl praktisch als auch theoretisch nicht geeignet: Einerseits gibt es keine effizienten Berechnungsmethoden, um $\hat{T}_n$ aus der großen Menge aller möglichen CARTs auszuwählen. Andererseits ist ohnehin zu erwarten, dass bei zu großem $t_{\max}$ eine Lösung von Gl. (6.3) überangepasst an die Trainingsdaten ist, d. h. $\hat{R}_n(f_{\hat{T}_n}) = 0$ gilt. Es muss daher entweder die Menge der möglichen Bäume verringert oder ein Bestrafungsterm eingeführt werden.

In diesem Abschnitt reduzieren wir die Menge der CARTs zunächst auf dyadische Bäume, die eine Aufteilung immer nur genau in der Mitte der aktuell betrachteten Menge erlauben. Um dies sinnvoll definieren zu können, sei im Folgenden

$$\mathcal{X} = \prod_{j=1}^{d} [a_j, b_j] \tag{6.4}$$

mit geeigneten $a_j, b_j \in \mathbb{R}$ $(j = 1, \ldots, d)$. Indem a_j, b_j groß genug gewählt werden, ist diese Annahme in der Praxis nicht problematisch.

Definition 6.6
Ein CART T heißt *dyadisch,* falls für jedes $v \in V_T \setminus B_T$ gilt: Mit $A(v) = \prod_{j=1}^{d} [a_j(v), b_j(v)]$ gilt $s(v) := \frac{a_{j(v)}(v) + b_{j(v)}(v)}{2}$. ♦

Indem wir auch die mögliche Tiefe des Baums einschränken, entsteht eine *endliche* Menge von möglichen Bäumen. In den im Folgenden besprochenen theoretischen Resultaten wird häufig auch angenommen, dass in keiner Koordinate „besonders oft“ geteilt werden muss, was zu der folgenden Menge führt:

Definition 6.7
Sei $A \in \mathbb{N}$ und

$$\mathcal{T}_A := \{T \text{ ist dyadischer CART mit } t(T) \leq A \cdot d \\ \text{und jede Koordinate wird höchstens } A\text{-mal geteilt}\}. \qquad \blacklozenge$$

Anschaulich gewährleistet eine Wahl von $A \approx \frac{\log_2(n)}{d}$, dass bei einem Baum mit $t(T) = A \cdot d$ jedes Blatt genau einen Punkt aufnehmen kann, wenn die Punkte X_i gleichmäßig in einem quadratischen Gitter im Raum verteilt sind. Ist d groß, bedeutet dies eine große Einschränkung an die Bäume, da bei zufälligen Beobachtungen X_i erhalten aus einer Wahrscheinlichkeitsverteilung evtl. mehr Teilungen (d. h. mehr als $\frac{\log_2(n)}{d}$) in einer bestimmten Koordinate gebraucht werden. Daher ist man versucht, A eher größer zu wählen, z. B. $A \approx \log_2(n)$.

Durch die Endlichkeit der Menge $\mathcal{T}_A$ ist es prinzipiell möglich, einen Minimierer von $\hat{R}_n(f_T)$ über $\mathcal{T}_A$ zu bestimmen. Man kann zeigen (vgl. zum Beispiel [9]), dass es dafür effiziente Berechnungsmethoden gibt. In Hinsicht auf praktische Anwendungen ist die Reduktion auf dyadische Bäume natürlich relativ willkürlich und wird nur selten verwendet. Im Gegensatz zum allgemeinen Minimierungsproblem in Gl. (6.3) oder den später eingeführten heuristischen Methoden sind jedoch für Minimierer über $\mathcal{T}_A$ einfach zu erfassende theoretische Resultate verfügbar. Deren Diskussion sowie die Einführung der Technik des „Prunings“ ist das Hauptanliegen dieses Abschnitt. Wir werden sehen, dass ohne tiefergehende strukturelle Annahmen an f^* die Konvergenzrate des Excess Bayes Risks der Algorithmen ebenfalls am „Fluch der Dimension“ leidet (vgl. Bemerkung 5.10), d. h., für große d ist die Konvergenz für $n \to \infty$ sehr langsam.

6.2.1 Der Standardansatz

Gegeben seien Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$ eines Klassifikationsproblems und $L(y, s) = \mathbb{1}_{\{y \neq s\}}$ der 0-1-Verlust. Da ein Minimierer $\hat{T}_n \in \arg\min_{T \in \mathcal{T}_A} \hat{R}_n(f_T)$ mit großer Wahrscheinlichkeit ein „voll ausgewachsener“ Baum ist (d. h. $t(T) = A \cdot d$) und an die Daten überangepasst ist, führen wir einen Bestrafungsterm $J(T)$ für die Größe des Baumes T ein. Dieser soll gewährleisten, dass der Minimierer nur die tatsächlich vorliegenden Strukturen aus den Trainingsdaten in die Baumstruktur übernimmt. Für die Wahl J gibt es verschiedene Möglichkeiten:

Bemerkung 6.8 (Bestrafungsterme) Eine Auswahl möglicher Bestrafungsterme ist:

$$J^{(1)}(T) = \#T, \qquad J^{(2)}(T) = \sqrt{\#T}, \qquad J_n^{(3)}(T) = \sum_{B \in \pi(T)} \sqrt{\frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{X_i \in B\}}}$$

Während $J^{(1)}$, $J^{(2)}$ die Anzahl der Blätter in verschiedenen Abstufungen bestrafen, misst $J_n^{(3)}$, wie gut der Baum die Verteilung der Punkte im Raum mit seiner Struktur berücksichtigt und ob er mehr Aufteilungen dort vornimmt, wo viele Daten sind (dies führt zu einer guten „räumlichen Anpassung“ an die zugrunde liegende Verteilung, engl. *spatial adaptivity*). Auch $J_n^{(3)}(T)$ ist monoton wachsend in $\#T$: Ist $\#T = 1$ und $B = \mathcal{X}$ das einzige Blatt, so gilt $J_n^{(3)}(T) = 1$; ist jedoch $\#T = n$ und $\pi(T) = \{B_1, \ldots, B_n\}$ so, dass $X_i \in B_i$, $i = 1, \ldots, n$ (jedes Blatt enthält nur ein Trainingsdatum), dann gilt $J_n^{(3)}(T) = \sqrt{n}$.

Für die Verwendung eines Bestrafungsterms J in der Praxis ist maßgeblich, ob entsprechend effiziente Berechnungsverfahren existieren, um beispielsweise auch Cross Validation o. Ä. für den Bestrafungsparameter λ durchführen zu können. Auch wenn für einige Arten von Bestrafungstermen wie beispielsweise $J_n^{(3)}$ in der Theorie leicht bessere Konvergenzraten gezeigt werden konnten (vgl. Bemerkungen nach Satz 6.11) und auch anschauliche Argumente für diese Wahl existieren, so wird in der Praxis trotzdem $J^{(1)}$, $J^{(2)}$ bevorzugt (z. B. aufgrund von Eigenschaften wie in Satz 6.19).

Wir geben nun ein theoretisches Resultat aus [28] wieder, bei welchem $J(T) = \sqrt{\#T}$ gewählt wird. Dazu definieren wir:

Definition 6.9
Sei $A \in \mathbb{N}$, $\lambda > 0$. Für einen CART T definiere $J(T) = \sqrt{\#T}$. Sei

$$\hat{T}_{n,\lambda,A} :\in \operatorname*{arg\,min}_{T \in \mathcal{T}_A} \left\{ \hat{R}_n(f_T) + \lambda \cdot J(T) \right\}.$$

Der zugehörige Algorithmus $\hat{f}^{dC}_{n,\lambda,A} := f_{\hat{T}_{n,\lambda,A}}$ heißt *dyadischer CART-Algorithmus* mit Bestrafungsterm J. ♦

Im Folgenden sei der Einfachheit halber $K = 2$ und $\mathcal{X} = [0,1]^d$. Da wir bereits Gl. (6.4) angenommen haben, kann Letzteres durch Reskalieren erreicht werden. Damit $\hat{f}^{dC}_{n,\lambda,A}$ die Bayes-Regel f^* gut genug approximieren kann, muss eine Modellannahme an die zugehörige optimale Diskriminantenfunktion $\delta_1^*(x) = \mathbb{P}(Y = 1 | X = x)$ gestellt werden. Wir fordern Folgendes:

- Die Verteilung von X darf nicht zu stark an einem bestimmten Ort im Raum konzentriert sein, vgl. (i) unten, denn dann muss $\hat{T}_{n,\lambda,A}$ eine große Tiefe haben um diesen Ort mit dyadischen Aufteilungen erreichen und die Daten separieren zu können.
- Der optimale Entscheidungsrand ist glatt genug, um durch genügend grobe stückweise konstante Funktionen (wie sie bei $\hat{f}^{dC}_{n,\lambda,A}$ entstehen) approximiert werden zu können, vgl. (ii) unten.

Zur formalen Beschreibung sei g eine Lebesgue-Dichte von $\mathbb{P}^X$.

Modellannahme 6.10 (Dyadische Bäume) Es sei $\mathcal{X} = [0,1]^d$. Es gebe Konstanten $c_1, c_2 > 0$ unabhängig von d, so dass folgende Aussagen gelten:

(i) $\sup_{x \in \mathcal{X}} g(x) \leq c_1$.
(ii) Für jedes $k \in \mathbb{N}$ und $m = 2^k$ gilt: Die Funktion δ_1^* geht durch höchstens $c_2 m^{d-1}$ der m^d Würfel, in die $\mathcal{X}$ zerfällt.

Ist δ_1^* stetig, so bedeutet die Bedingung (ii) anschaulich, dass der durch die Funktion δ_1^* erzeugte optimale Entscheidungsrand $\partial\Omega_1^* = \{x \in \mathcal{X} : \delta_1^*(x) = \frac{1}{2}\}$ nicht zu „dicht“ im Raum $\mathcal{X}$ liegt, d. h. die Funktion δ_1^* nicht flächendeckend auf $\mathcal{X}$ zwischen Werten kleiner und größer als $\frac{1}{2}$ wechselt. Dies ist zum Beispiel erfüllt, wenn $\partial\Omega_1^* = \{(x_1, \ldots, x_{d-1}, \phi(x_1, \ldots, x_{d-1}) : x_1, \ldots, x_{d-1} \in [0,1]\}$ mit einer Lipschitz-stetigen Funktion $\phi : [0,1]^{d-1} \to [0,1]$ gilt, d. h. sich eine Koordinate als Lipschitz-stetige Funktion der anderen darstellen lässt. In diesem Sinne entspricht (ii) in etwa der Forderung an δ_1^* in Modellannahme 5.15.

Wir erhalten aus [28, Theorem 3] folgenden Satz:

Satz 6.11 (Excess Bayes Risk für den dyadischen CART-Algorithmus) Es gelte die Modellannahme 6.10. Es sei $A = \lfloor \frac{\log_2(n)}{d+1} \rfloor$. Dann gibt es eine Konstante $c > 0$ unabhängig von d, n, so dass für $\lambda = \sqrt{\frac{32 \log(en)}{n}}$ gilt:

$$\mathbb{E} R(\hat{f}_{n,\lambda,A}^{dC}) - R(f^*) \leq c \cdot \left(\frac{\log(n)}{n} \right)^{\frac{1}{d+1}}$$

Bemerkungen

- Auch wenn oben ein konkretes λ vorgegeben ist, so würde man in der Praxis trotzdem λ durch Cross Validation wählen, damit die Wahl von λ besser auf die Trainingsdaten zugeschnitten ist.
- Erfüllt die Verteilung von (X, Y) die *low-noise condition* (vgl. Definition 3.12), so kann gezeigt werden, dass die optimale Rate eines Klassifizierers durch $n^{-1/d}$ gegeben ist. Unter der Annahme, dass die *low-noise condition* in etwa dieselben Modelle zulässt wie die Modellannahme 6.10 (ii), ist die Konvergenzrate von $\hat{f}_n$ höchstens ein wenig schlechter als die optimale Rate. In diesem Sinne unterliegt $\hat{f}_{n,\lambda,A}^{dC}$ zwar dem Fluch der Dimension, aber unter den gegebenen, nicht besonders informativen Modellannahmen ist auch kein besseres Verhalten möglich.
- Die Berechnung von $\hat{f}_{n,\lambda,A}^{dC}$ kann je nach Größe von d sehr aufwendig sein, da A in der Praxis häufig größer als $\frac{\log_2(n)}{d}$ gewählt werden muss, um eine gleichmäßige Aufteilung der Trainingsdaten mit $\hat{f}_{n,\lambda,A}^{dC}$ zu erreichen.
- Durch Wahl komplizierterer Bestrafungsterme J (vgl. [29, 30]) kann die optimale Rate $n^{-1/d}$ erreicht werden.

6.2.2 Zurückschneiden von Bäumen (Pruning)

Bei der Berechnung von $\hat{f}^{dC}_{n,\lambda,A}$ werden viele irrelevante Bäume $T \in \mathcal{T}_A$ überprüft, bei welchen Bereiche mit wenigen Trainingsdaten unnötig oft aufgeteilt werden. Ein alternativer Ansatz lautet wie folgt: Wir geben einen sehr großen, an die Trainingsdaten überangepassten Baum $T_0 \in \mathcal{T}_A$ vor, dessen Aufteilungsstruktur sich jedoch ebenfalls bereits an den Trainingsdaten orientiert. Das bedeutet, wir erwarten, dass T_0 den Raum $\mathcal{X}$ besonders dort detailliert aufteilt, wo viele Trainingsdaten liegen, und dafür Bereiche ohne Trainingsdaten gar nicht aufteilt. Anstelle einer Suche über alle möglichen $T \in \mathcal{T}_A$ suchen wir nun nur unter allen Bäumen, die eine „gröbere“ Aufteilung des Raumes als T_0 vornehmen. Mathematisch kann dies mittels zurückgeschnittener Teilbäume ausgedrückt werden:

Definition 6.12 (Teilbaum, zurückgeschnittener Teilbaum)
Sei T ein Baum. Ein Baum $T' \subset T$ (d. h. $V_{T'} \subset V_T$ und $E_{T'} \subset E_T$) heißt *Teilbaum* (engl. *subtree*) von T. Ist $v_0 \in V_T$ die Wurzel von T und $v_0 \in V_{T'}$ (d. h., v_0 ist auch Wurzel von T'), so heißt T' *zurückgeschnittener Teilbaum* (engl. *pruned subtree*) von T, und wir schreiben $T' \prec T$. ♦

Ein zurückgeschnittener Teilbaum entsteht aus T dadurch, dass man wiederholt alle von einem Knoten ausgehenden weiteren Knoten entfernt. Anschaulich entspricht dies dem „Abschneiden“ des ausgewählten Knotens, vgl. Abb. 6.4.

Basierend auf der obigen Motivation erhalten wir den folgenden Algorithmus:

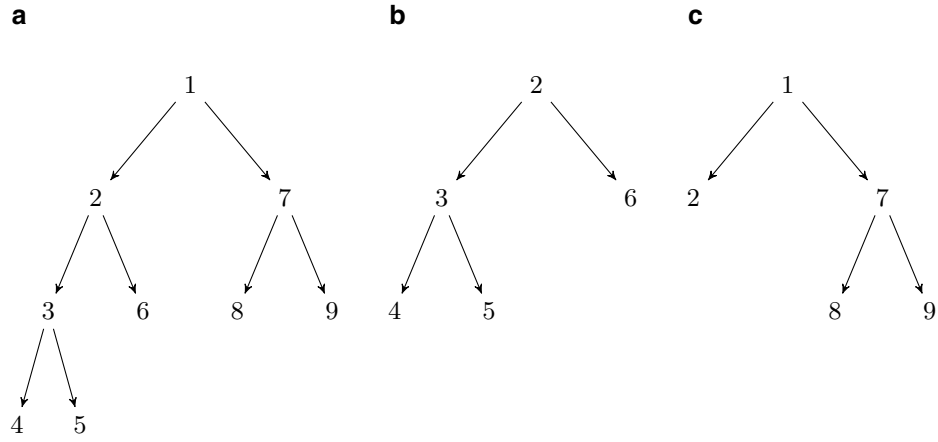


Abb. 6.4 **a** ursprünglicher Baum T. **b** Ein Teilbaum T' von T, aber kein zurückgeschnittener Teilbaum: T' enthält nicht den Wurzelknoten $1 \in V_T$. **c** Ein zurückgeschnittener Teilbaum von T, entfernt wurden alle Kinder des Knotens $2 \in V_T$

Definition 6.13 (Dyadischer CART-Algorithmus mit Pruning)
Sei $A \in \mathbb{N}$ und $\lambda > 0$. Sei $J(T) = \sqrt{\#T}$ und

$$\hat{T}^{pC}_{n,\lambda,A} :\in \operatorname*{arg\,min}_{T \prec T_0} \left\{ \hat{R}_n(f_T) + \lambda \cdot J(T) \right\}. \tag{6.5}$$

Der zugehörige Algorithmus $\hat{f}^{pC}_{n,\lambda,A} := f_{\hat{T}^{pC}_{n,\lambda}}$ heißt *dyadischer CART-Algorithmus* mit Pruning. ♦

Unter der Annahme, dass T_0 groß genug ist, folgt das Konvergenzresultat für $\hat{f}^{pC}_{n,\lambda,A}$ direkt aus demjenigen für $\hat{f}^{dC}_{n,\lambda,A}$ (vgl. [29, Section 2]):

Satz 6.14 Es gelte die Modellannahme 6.10. Es sei $A = \lfloor \frac{\log_2(n)}{d+1} \rfloor$. Es sei $T_0 \in \mathcal{T}_A$, so dass jedes Blatt $v \in B_T$ Tiefe $t(v) = A \cdot d$ besitzt. Dann gibt es eine Konstante $c > 0$ unabhängig von d, n, so dass für $\lambda = \sqrt{\frac{32 \log(en)}{n}}$ gilt:

$$\mathbb{E} R(\hat{f}^{pC}_{n,\lambda,A}) - R(f^*) \leq C \cdot \left(\frac{\log(n)}{n} \right)^{\frac{1}{d+1}}$$

6.3 Effiziente Erzeugung von CARTs

Ein Optimierungsproblem über dyadische Bäume kann zwar explizit gelöst werden, jedoch dauert die Suche bei hohen Dimensionen d oder bei hoher Anzahl n an Trainingsdaten sehr lange. Abgesehen davon ist die dyadische Struktur zu starr; in vielen Fällen können schneller sinnvolle Partitionen des Raumes erreicht werden, wenn man die Teilungen nicht jeweils genau in der Mitte durchführen muss.

Auch für allgemeine CARTs ohne die Beschränkung auf dyadische Bäume ist ohne weitere Annahmen an f^* keine bessere Konvergenzrate des Excess Bayes Risks als $n^{-1/d}$ zu erwarten. Wir werden allerdings später erweiterte Algorithmen kennenlernen, die unter speziellen Annahmen an f^* bessere Raten erreichen können. Diese Algorithmen basieren auf der Erzeugung sehr vieler verschiedener Bäume aus den Trainingsdaten, stellen aber oft keine hohen Anforderungen an die *Qualität* der einzelnen Bäume. Dieser Anforderung folgend werden wir nun ein Verfahren vorstellen, das in der Lage ist, sehr schnell einen CART bereitzustellen, der nur eine näherungsweise Lösung von $\min_{T \text{ CART}} \{\hat{R}_n(f_T) + \lambda \cdot J(T)\}$ bestimmt.

In einem ersten Schritt erzeugen wir dazu aus den Trainingsdaten einen sehr großen Baum $\hat{T}^g_n$ mit vorgegebener Tiefe bzw. vorgegebener minimaler Anzahl von Trainingsdaten in einem Blatt, der bereits *näherungsweise* das empirische Risiko minimiert. Einige erweiterte Algorithmen wie das Boosting (vgl. Abschn. 6.5) verlangen nur nach Bäumen geringer Tiefe, so dass dort direkt $\hat{T}^g_n$ verwendet werden kann. Soll der resultierende Baum jedoch

ein möglichst geringes Risiko haben, so kann $\hat{T}_n^g$ sehr groß erzeugt werden und in einem zweiten Schritt als T_0 im Optimierungsproblem aus Gl. (6.5) dienen.

6.3.1 Gierige Verfahren

Als Motivation für die Konstruktion von $\hat{T}_n^g$ dient folgende Beobachtung: Haben wir einen beliebigen CART T gegeben und sind die Blätter $\{v_m : m = 1, \ldots, \#T\} = B_T$ und die zugehörige Partition $\{A_m = A(v_m) : m = 1, \ldots, \#T\}$ bereits festgelegt, so ist die Bestimmung der zugehörigen Werte $y_m := y(v_m)$ leicht: Es gilt nämlich

$$f_T(x) = \sum_{m=1}^{\#T} y_m \mathbb{1}_{A_m}(x),$$

und daher (vgl. Übung 1):

(*) für Regression mit $L(y, s) = (y - s)^2$:

$$\hat{R}_n(f_T) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - f_T(X_i))^2 = \sum_{m=1}^{\#T}\frac{1}{n}\sum_{i:X_i\in A_m}(Y_i - y_m)^2 \tag{6.6}$$

wird minimal für $y_m = \frac{1}{n}\sum_{i:X_i\in A_m} Y_i$, $m = 1, \ldots, \#T$.

(**) für Klassifikation mit $L(y, s) = \mathbb{1}_{\{y\neq s\}}$:

$$\hat{R}_n(f_T) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{\{Y_i\neq f_T(X_i)\}} = \sum_{m=1}^{\#T}\frac{1}{n}\sum_{i:X_i\in A_m}\mathbb{1}_{\{Y_i\neq y_m\}} \tag{6.7}$$

wird minimal für $y_m = \arg\max_{k\in\{0,\ldots,K-1\}} \#\{i : X_i \in A_m, Y_i = k\}, m = 1, \ldots, \#T$.

Gibt man eine Partition vor, so existieren also explizite Formeln für die Werte y_m, $m = 1, \ldots, \#T$. Dies bedeutet, dass ein Lösungsverfahren, welches

$$\min_{T \text{ CART}} \hat{R}_n(f_T) \tag{6.8}$$

bestimmen soll, nur über die Partitionen optimieren muss. Aufgrund der großen Menge an Partitionen ist dies in der Praxis jedoch immer noch nicht effizient berechenbar. Man versucht daher, eine näherungsweise Lösung zu ermitteln. Dazu minimiert man nicht über alle möglichen Bäume, sondern baut (wie am Anfang von Abschn. 6.1 motiviert) einen CART T schrittweise auf. Dies geschieht wie folgt: Ist $v \in B_T$ ein Blatt von T, so entsteht durch Anfügen zweier Knoten v_1, v_2 an v ein neuer Baum T_{neu}. Zur korrekten Beschreibung von T_{neu} als CART muss festgelegt werden, entlang welcher Koordinate $j(v) \in \{1, \ldots, d\}$ und an welchem Punkt $s(v) \in \mathbb{R}$ die Aufteilung von $A(v)$ erfolgt ($A(v)$ ist der bisher

dem Knoten v zugeordnete Raum). Weiter müssen die Werte $c_1 := y(v_1) \in \mathcal{Y}$ und $c_2 := y(v_2) \in \mathcal{Y}$ gesetzt werden, welche der Baum T_{neu} auf den neu entstandenen Räumen $A_1(j,s) = A(v) \cap \{x \in \mathcal{X} : x_j < s\}$ und $A_2(j,s) = A(v) \cap \{x \in \mathcal{X} : x_j \geq s\}$ annimmt. Es werden diejenigen Werte s, j, c_1, c_2 ausgewählt, welche das empirische Risiko (im Vergleich zum bisherigen Risiko $\hat{R}_n(f_T)$)

$$\hat{R}_n(f_{T_{neu}}) = \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f_{T_{neu}}(X_i))$$

am stärksten verringern. Die Prozedur wird gestoppt, wenn in jedem Blatt nur noch wenige Datenpunkte enthalten sind (z. B. jeweils nur einer). Solch ein Verfahren, welches das Minimierungsproblem in Schritte zerteilt und nur für jeden einzelnen Schritt die jeweils optimale Lösung bestimmt, nennt man *gieriges Verfahren* (engl. *greedy algorithm*). Im Allgemeinen wird dadurch keine optimale Lösung des gesamten Problems Gl. (6.8) erreicht, aber in vielen Fällen zumindest eine gute Näherung. Im Regressionsfall erhalten wir die folgende Definition:

Definition 6.15 (Gieriges Verfahren CART/Regressionsfall)
Sei $T^{(0)} = (v_0, \emptyset)$ und $k = 1$.

(i) Für jedes Blatt $v \in B_{T^{(k-1)}}$ mit $\#\{i : X_i \in A(v)\} > 1$ führe durch (falls $k = 1$ und daher $v = v_0$, setze $A(v) = \mathcal{X}$):

1. Setze $A_1(j,s) := A(v) \cap \{x : x_j < s\}$ und $A_2(j,s) := A(v) \cap \{x : x_j \geq s\}$.
2. Sei $T_{neu}(j,s,c_1,c_2)$ der Baum, welcher durch Anfügen zweier Knoten v_1, v_2 an den Knoten v in $T^{(k-1)}$ mit $s(v) = s \in \mathbb{R}$, $j(v) = j \in \{1,\ldots,d\}$ und $c_1 = c_1(v_1)$, $c_2 = c_2(v_2)$ entsteht.
3. Löse

$$\begin{aligned} &\min_{j,s,c_1,c_2} \hat{R}_n(f_{T_{neu}}(j,s,c_1,c_2)) \\ &\quad = \min_{j,s,c_1,c_2} \Bigg\{ \sum_{i:X_i \in \mathcal{X} \setminus A(v)} (Y_i - f_{T^{(k-1)}}(X_i))^2 \\ &\qquad + \sum_{i:X_i \in A_1(j,s)} (Y_i - c_1)^2 + \sum_{i:X_i \in A_2(j,s)} (Y_i - c_2)^2 \Bigg\} \qquad (6.9) \\ &\overset{(*)}{=} \min_{j,s} \left\{ \sum_{i:X_i \in A_1(j,s)} (Y_i - \hat{c}_1(j,s))^2 + \sum_{i:X_i \in A_2(j,s)} (Y_i - \hat{c}_2(j,s))^2 \right\}, \qquad (6.10) \end{aligned}$$

   mit $\hat{c}_1(j,s) = \frac{1}{n} \sum_{i:X_i \in A_1(j,s)} Y_i$, $\hat{c}_2(j,s) = \frac{1}{n} \sum_{i:X_i \in A_2(j,s)} Y_i$; erhalte $\hat{j}, \hat{s}$.
4. Setze $T^{(k)} = T_{neu}(\hat{j}, \hat{s}, \hat{c}_1(\hat{j},\hat{s}), \hat{c}_2(\hat{j},\hat{s}))$ und erhöhe k um 1.

Wird (i) nicht mehr ausgeführt (alle Blätter enthalten nur noch ein Trainingsdatum), so gebe den Baum $\hat{T}_n^g = T^{(k)}$ zurück. Wir nennen $\hat{T}_n^g$ den durch ein *gieriges Verfahren ermittelten CART.* ♦

Bemerkungen

- In (3) haben wir den zuvor ermittelten Sachverhalt (*) genutzt. Dies erlaubte es, c_1, c_2 aus dem Optimierungsproblem zu entfernen. Der Summand $\sum_{i:X_i \in \mathcal{X} \setminus A(v)} (Y_i - f_{T^{(k-1)}}(X_i))^2$ hingegen ist unabhängig von j, s, c_1, c_2 und kann ignoriert werden.
- Das Minimierungsproblem in (3) ist einfach lösbar, denn $j \in \{1, \ldots, d\}$ läuft nur über eine endliche Menge von Werten, und der *split point* $s \in \mathbb{R}$ kann nur an n Stellen sinnvoll gesetzt werden (zwischen den Koordinaten der Trainingsdaten $\{X_{ij} : i = 1, \ldots, n\}$). Die Optimierung über s wird mit zunehmender Baumgröße einfacher, da immer weniger Trainingsdaten überhaupt im gerade betrachteten Knoten $A(v)$ liegen.
- Obiger gieriger Algorithmus liefert im Allgemeinen gute Näherungen für die Lösung von Gl. (6.8), die aber sehr variabel sein können: Schon kleine Änderungen an den Trainingsdaten können die gesamte Struktur des Baums verändern. Begründet ist dies im schrittweisen Aufbau des Baumes.

Im Klassifikationsfall mit $\mathcal{Y} = \{1, \ldots, K\}$ kann die Konstruktion genauso durchgeführt werden wie in Definition 6.15, allerdings ändert sich Schritt (3) aufgrund der abweichenden Verlustfunktion $L(y, s) = \mathbb{1}_{\{y \neq s\}}$:

Definition 6.16 (Gieriges Verfahren CART/Klassifikationsfall)
Erstelle $\hat{T}_n^g$ wie in Definition 6.15, aber ersetze (3) durch:

$$
\begin{aligned}
&\min_{j,s,c_1,c_2} \hat{R}_n(f_{T_{neu}}(j, s, c_1, c_2)) \\
&= \min_{j,s,c_1,c_2} \left\{ \text{const.} + \sum_{i:X_i \in A_1(j,s)} \mathbb{1}_{\{Y_i \neq c_1\}} + \sum_{i:X_i \in A_2(j,s)} \mathbb{1}_{\{Y_i \neq c_2\}} \right\} \\
&\overset{(**)}{=} \min_{j,s} \left\{ \sum_{i:X_i \in A_1(j,s)} \mathbb{1}_{\{Y_i \neq \hat{c}_1(j,s)\}} + \sum_{i:X_i \in A_2(j,s)} \mathbb{1}_{\{Y_i \neq \hat{c}_2(j,s)\}} \right\} \\
&= \min_{j,s} \left\{ \#A_1(j,s) \cdot [1 - \hat{p}_{\hat{c}_1(j,s)}(A_1(j,s))] + \#A_2(j,s) \cdot [1 - \hat{p}_{\hat{c}_2(j,s)}(A_2(j,s))] \right\} \\
&= \min_{j,s} \left\{ \#A_1 \cdot [1 - \hat{p}_{\hat{c}_1}(A_1)] + \#A_2 \cdot [1 - \hat{p}_{\hat{c}_2}(A_2)] \right\}, \qquad (6.11)
\end{aligned}
$$

(im letzten Schritt wurde zur Übersicht die Abhängigkeit von j, s unterdrückt), wobei

$$
\hat{p}_k(A) := \frac{\#\{i : X_i \in A, Y_i = k\}}{\#A} \qquad (6.12)
$$

und

$$\hat{c}_1(j,s) = \operatorname*{arg\,max}_{k\in\{1,\dots,K\}} \hat{p}_k(A_1(j,s)), \tag{6.13}$$

$$\hat{c}_2(j,s) = \operatorname*{arg\,max}_{k\in\{1,\dots,K\}} \hat{p}_k(A_2(j,s)). \tag{6.14}$$

♦

Bemerkung 6.17 In der Praxis wird anstatt Gl. (6.11) jedoch häufig ein anderes Risiko minimiert, welches durch eine auf Klassifizierungsprobleme verallgemeinerte quadratische Verlustfunktion entsteht und folgendes Optimierungsproblem liefert:

$$\min_{j,s} \left\{ \#A_1 \cdot \sum_{k=1}^{K} \hat{p}_k(A_1)[1-\hat{p}_k(A_1)] + \#A_2 \cdot \sum_{k=1}^{K} \hat{p}_k(A_2)[1-\hat{p}_k(A_2)] \right\}$$

Im Gegensatz zu $[1-\hat{p}_{\hat{c}_1}(A_1)]$ verwendet man also $\sum_{k=1}^{K} \hat{p}_k(A_1)[1-\hat{p}_k(A_1)]$, den sogenannten *Gini-Index*. Tatsächlich geht die gewählte Klasse $\hat{c}_1(j,s)$ dann gar nicht mehr in das Optimierungsproblem ein; mittels der Optimierer $\hat{j}, \hat{s}$ und Gl. (6.13), (6.14) wird sie aber trotzdem für die Erstellung des Baumes benutzt.

Der Gini-Index misst auf eine andere Weise, wie gut die für A_1 gewählte Klasse sich an die Daten in A_1 anpasst. Statt nur die ausgewählte Klasse zu berücksichtigen, wird gemessen, wie „rein" die Menge A_1 ist und wie stark die einzelnen Klassen noch in A_1 vorkommen. Es wird bevorzugt, wenn A_1 fast nur aus einer Klasse besteht, denn: Der Gini-Index $G(p_1,\dots,p_K) = \sum_{k=1}^{K} p_k \cdot (1-p_k)$ ist groß, falls alle p_k in etwa gleich groß sind, und $G(p_1,\dots,p_K)$ ist klein, falls z. B. $p_1 = 1$ und $p_2 = \cdots = p_K = 0$ gilt (vgl. Übung 2).

In Abb. 6.5 ist ein Beispiel für einen nahezu „voll ausgewachsenen" Baum erzeugt durch $\hat{T}_n^g$ aus Definition 6.15 angewandt auf Beobachtungen aus Gl. (6.1) zu sehen. Nahezu „voll ausgewachsen" bedeutet hier, dass das Verfahren durchgeführt wurde, bis in jedem Blatt nur noch zwei Trainingsdaten enthalten waren *oder* eine Tiefe von 10 erreicht wurde. In Abb. 6.6 ist $\hat{T}_n^g$ aus Definition 6.16 für das Klassifikationsproblem aus Gl. (6.2) dargestellt.

6.3.2 Zurückschneiden

Die oben eingeführten Verfahren liefern voll ausgewachsene, an die Trainingsdaten überangepasste CARTs $\hat{T}_n^g$. Da der Aufbau des Baums in Definition 6.15 bzw. 6.16 erst beendet wird, wenn jedes Blatt nur noch ein Trainingsdatum (X_i, Y_i) enthält oder eine maximal vorgegebene, datenunabhängige Tiefe überschritten wird, wird fast jedem X_i genau die Beobachtung Y_i zugewiesen, aber kaum die wahre Struktur der Bayes-Regel gelernt.

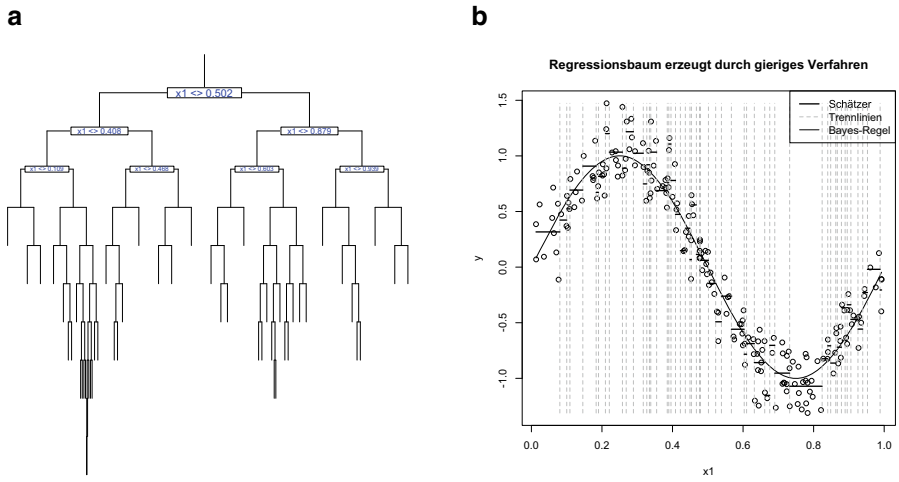


Abb. 6.5 **a** CART $\hat{T}_n^g$ erzeugt durch den gierigen Algorithmus. **b** Die Linie „Schätzer" gibt die zum Baum gehörige Entscheidungsregel $f_{\hat{T}_n^g}$ an. Zum Vergleich ist die Bayes-Regel eingezeichnet

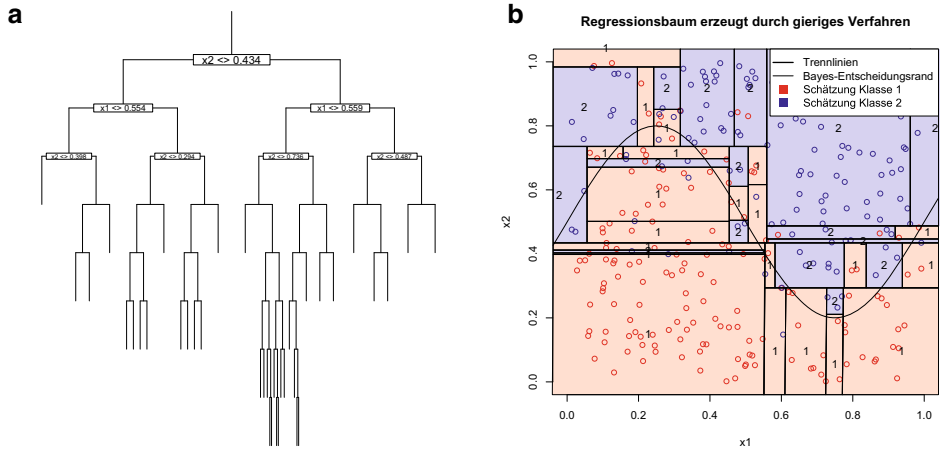


Abb. 6.6 **a** CART $\hat{T}_n^g$ erzeugt durch den gierigen Algorithmus. **b** Die farbigen Hinterlegungen geben die zum Baum gehörige Entscheidungsregel $f_{\hat{T}_n^g}$ an. Zum Vergleich ist der optimale Entscheidungsrand eingezeichnet

Der Ansatz, den gierigen Algorithmus bereits frühzeitig zu stoppen, d. h. bevor $\hat{T}_n^g$ zu groß wird, ist oft nicht sinnvoll: Das einzige naheliegende Maß für die Entscheidung, einen Stopp durchzuführen, ist der Betrag, um welchen das empirische Risiko (oder ein Validierungsfehler) durch das Hinzufügen eines neuen Knotens abgesenkt werden konnte. Da nach einer schlechten Aufteilung aber eventuell eine sehr gute möglich ist, liefert solch ein Vorgehen systematisch zu grobe Strukturen.

Stattdessen versucht man, auf Basis des voll ausgewachsenen Baums $\hat{T}_n^g$ einen geeigneten Teilbaum zu finden, indem man das Optimierungsproblem aus Gl. (6.5) mit $T_0 = \hat{T}_n^g$ durchführt. Dieses Vorgehen heißt *cost-complexity pruning*.

Definition 6.18
Sei $\hat{T}_n^g$ der Baum aus Definition 6.15 bzw. 6.16. Sei $\lambda > 0$. Dann heißt

$$\hat{T}_{n,\lambda}^{g,p} :\in \operatorname*{arg\,min}_{T \prec \hat{T}_n^g} \left\{ \hat{R}_n(f_T) + \lambda \cdot \#T \right\}$$

CART ermittelt durch gieriges Verfahren und Zurückschneiden. Der zugehörige Algorithmus wird mit $\hat{f}_{n,\lambda}^{g,p} := f_{\hat{T}_{n,\lambda}^{g,p}}$ bezeichnet. ♦

Der folgende Satz (vgl. [11] oder [27]) erlaubt eine effiziente Berechnung des obigen Minimums *und* ermöglicht gleichzeitig eine effiziente Bestimmung von λ zum Beispiel mittels Cross Validation. Die Aussage gilt für eine große Klasse an Verlustfunktionen L (die durch $\hat{R}_n(f_T)$ in das Optimierungsproblem eingehen), insbesondere die für uns relevanten $L(y,s) = (y-s)^2$ und $L(y,s) = \mathbb{1}_{\{y \neq s\}}$:

Satz 6.19 Sei $T^{(0)}$ ein CART. Wiederhole für $p = 1, 2, 3, \ldots, P$, bis $T^{(P)}$ nur noch aus der Wurzel besteht:

$$T^{(p)} :\in \operatorname*{arg\,min}_{T \prec T^{(p-1)}} \frac{\hat{R}_n(f_T) - \hat{R}_n(f_{T^{(p-1)}})}{\#T^{(p-1)} - \#T} \tag{6.15}$$

Dann gilt für jedes $\lambda > 0$: Der CART

$$T_\lambda \in \operatorname*{arg\,min}_{T \prec T^{(0)}} \left\{ \hat{R}_n(f_T) + \lambda \cdot \#T \right\}$$

erfüllt: $T_\lambda \in \{T^{(0)}, \ldots, T^{(P)}\}$.

Der Iterationsschritt in Gl. (6.15) heißt *weakest link pruning:* Es wird immer der Teilbaum gebildet, welcher den geringsten Pro-Knoten-Anstieg von $\hat{R}_n(f_T)$ im Vergleich zum Risiko $\hat{R}_n(f_{T^{(p-1)}})$ des vorherigen Baums in der Sequenz besitzt. In Abb. 6.7 sind ausgewählte Vertreter der Sequenz $T^{(0)}, \ldots, T^{(P)}$ des CART $T^{(0)} = \hat{T}_n^g$ aus Abb. 6.5 zu sehen, in Abb. 6.8 entsprechend für den CART $T^{(0)} = \hat{T}_n^g$ aus Abb. 6.6 im Falle der Klassifikation.

Das zentrale Ergebnis von Satz 6.19 ist die Tatsache, dass sich die Sequenz $T^{(0)}, T^{(1)}, \ldots, T^{(P)}$ für verschiedene λ nicht ändert. Damit erhalten wir folgende Methode zur Berechnung von $\{\hat{T}_{n,\lambda}^{g,p} : \lambda > 0\}$:

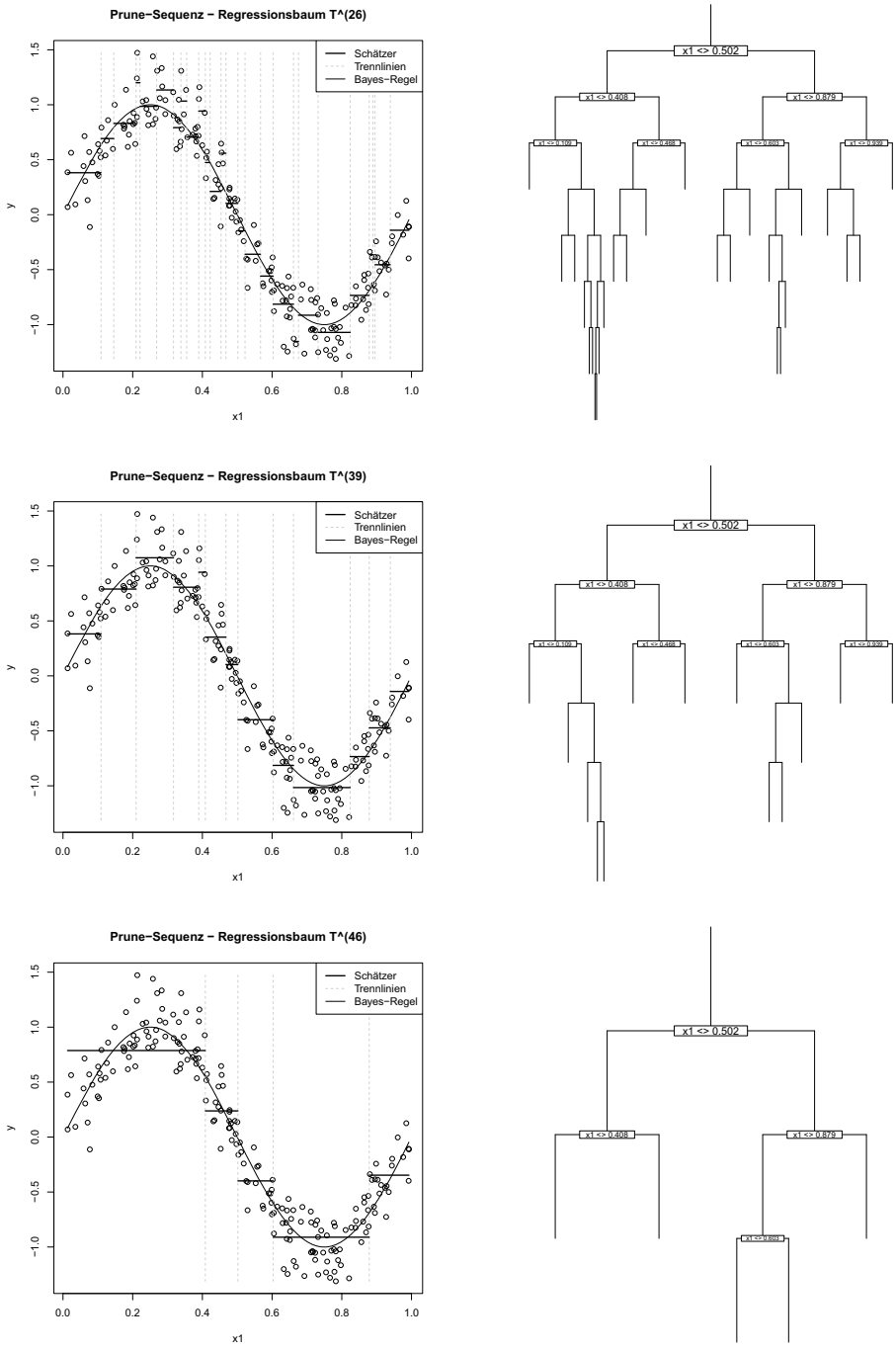


Abb. 6.7 *Weakest link pruning:* Ausgewählte Vertreter der Sequenz $T^{(0)}, \ldots, T^{(P)}$ des CART $T^{(0)} = \hat{T}_n^g$ aus Abb. 6.5

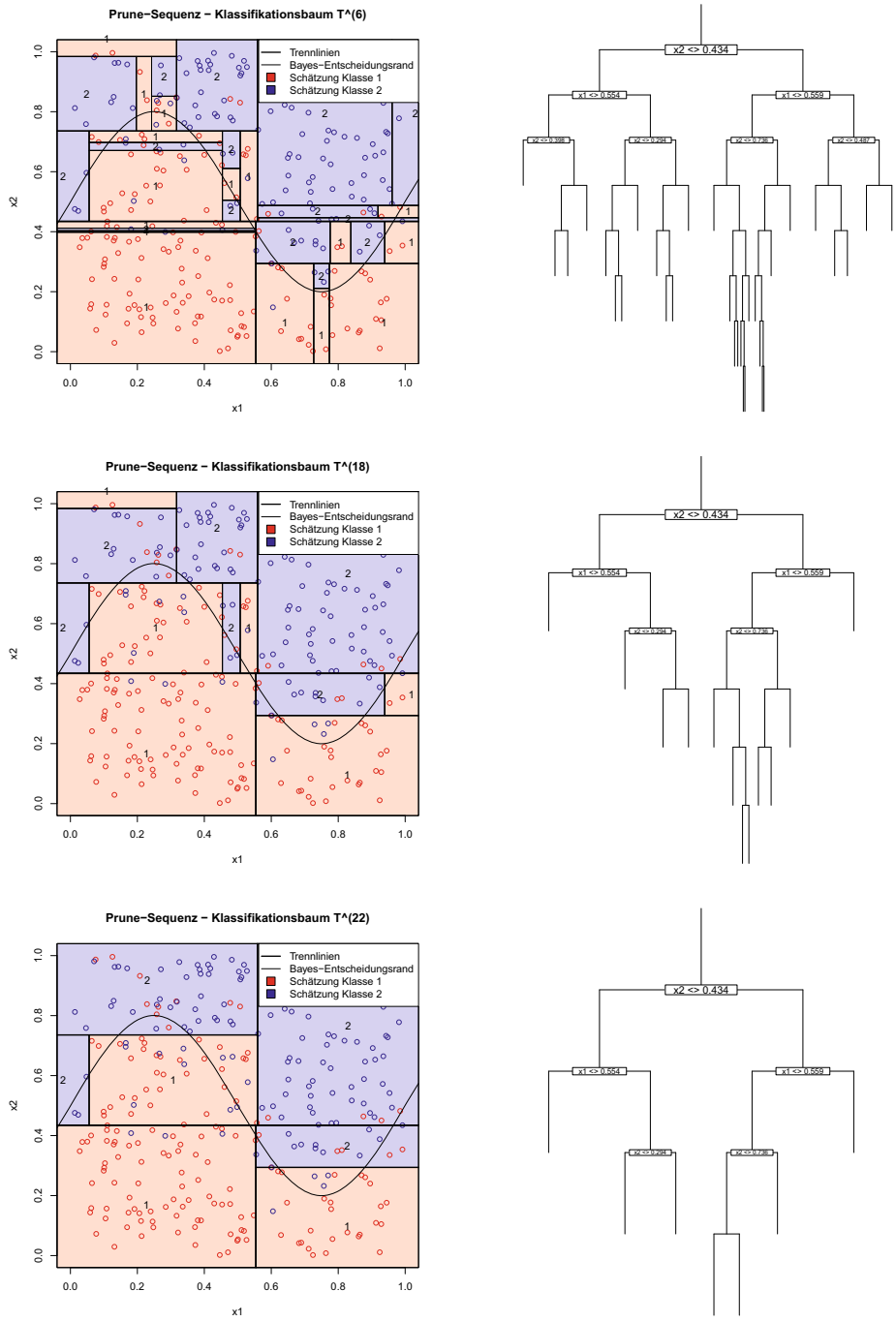


Abb. 6.8 *Weakest link pruning:* Ausgewählte Vertreter der Sequenz $T^{(0)}, \ldots, T^{(P)}$ des CART $T^{(0)} = \hat{T}_n^g$ aus Abb. 6.6

Lemma 6.20 Bestimme die Sequenz $\hat{T}_n^g = T^{(0)}, T^{(1)}, \ldots, T^{(P)}$ gemäß Gl. (6.15). Für $\lambda > 0$ bestimme

$$\hat{p}(\lambda) :\in \operatorname*{arg\,min}_{p\in\{0,\ldots,P\}} \left\{ \hat{R}_n(f_{T^{(p)}}) + \lambda \cdot \#T^{(p)} \right\}.$$

Dann ist $\hat{T}_{n,\lambda}^{g,p} = T^{(\hat{p}(\lambda))}$.

Der Parameter $\lambda > 0$ wird in der Praxis zum Beispiel mit Cross Validation (vgl. Definition 1.22) bestimmt. Da die Kombination von Cross Validation mit den Ergebnissen aus Lemma 6.20 etwas komplexer ist, formulieren wir die korrekte Durchführung aus:

Bemerkung 6.21 Sei $\Lambda \subset (0, \infty)$ eine (endliche) Menge. Sei $\{1, \ldots, n\} = \bigcup_{m=1}^{M} I_m$ eine Partition der Beobachtungen in M Indexmengen.

(i) Für $m = 1, \ldots, M$:
   1. Ermittle $\hat{T}_n^g(m)$ basierend auf den Beobachtungen $\{1, \ldots, n\} \setminus I_m$ und daraus die Sequenz $\{T^{(0)}(m), \ldots, T^{(P_m)}(m)\}$ aus Satz 6.19.
   2. Für $\lambda \in \Lambda$ bestimme $\hat{T}_{n,\lambda}^{g,p}(m)$ mittels Lemma 6.20, d. h.

$$\hat{p}(\lambda, m) :\in \operatorname*{arg\,min}_{p\in\{0,\ldots,P\}} \left\{ \frac{1}{n} \sum_{i\in\{1,\ldots,n\}\setminus I_m} L(Y_i, f_{T^{(p)}(m)}) + \lambda \cdot \#T^{(p)}(m) \right\}.$$

   und $\hat{T}_{n,\lambda}^{g,p}(m) := T^{(\hat{p}(\lambda,m))}$.

(ii) Berechne für $\lambda \in \Lambda$:

$$\mathrm{CV}_n(\lambda) := \frac{1}{n} \sum_{m=1}^{M} \sum_{i\in I_m} L(Y_i, f_{\hat{T}_{n,\lambda}^{g,p}(m)}(X_i)),$$

   und ermittle $\hat{\lambda}_n := \arg\min_{\lambda\in\Lambda} \mathrm{CV}_n(\lambda)$.

(iii) Ermittle $\hat{T}_n^g$ basierend auf den Beobachtungen $\{1, \ldots, n\}$ und darauf basierend die Sequenz $\{T^{(0)}, \ldots, T^{(P)}\}$ aus Satz 6.19. Berechne dann $\hat{T}_{n,\hat{\lambda}_n}^{g,p}$ mittels Lemma 6.20.

In Abb. 6.7 entspricht $\hat{T}_{n,\hat{\lambda}_n}^{g,p} = T^{(39)}$, in Abb. 6.8 entspricht $\hat{T}_{n,\hat{\lambda}_n}^{g,p} = T^{(18)}$.

Bemerkung 6.22 Einige Vorteile von CARTs, welche zu deren Beliebtheit beigetragen haben:

- Aus Trainingsdaten erhaltene CARTs sind leicht zu interpretieren: Die Baumstruktur gibt eine Reihe von Fragen an x vor (anhand des *split index* und des *split points*), bei deren sukzessiver Beantwortung am Ende ein Resultat (der Wert $f_T(x)$) ausgegeben wird. Da der Baum mit den ersten Aufteilungen die Grobstruktur der Bayes-Regel nachbildet

und mit größerer Tiefe immer detailgetreuer wird, nimmt mit jeder weiteren Frage die Bedeutung der Antwort ab.
- Bäume sind einfach zu visualisieren.

Einige Nachteile von CARTs, mit deren Beseitigung wir uns in den Abschn. 6.4 und 6.5 auseinandersetzen wollen, sind folgende:

- Die Struktur aus Daten bestimmten CARTs ist sehr variabel: Schon kleine Änderungen in den Trainingsdaten können zu völlig anderen Splits und Baumstrukturen T führen, so dass auch die Entscheidungsregel f_T eine hohe Varianz besitzt. Dies läuft der oben vorgetragenen Intuition zuwider, dass die erste Entscheidung „die wichtigste" sein sollte und zeigt, dass die Bedeutung der erzeugten Struktur von CARTs basierend auf Trainingsdaten nicht überschätzt werden sollte.
- Die entstehenden Bayes-Regeln sind nicht stetig, obwohl das bei Modellannahmen häufig von f^* bzw. den Bayes-Entscheidungsrändern erwartet wird. Mit Hilfe der Technik des Baggings (vgl. Abschn. 6.4) können Bäume so modifiziert werden, dass die ersten beiden Nachteile aufgehoben werden.
- Bäume haben Probleme, additive Strukturen korrekt wiederzugeben: Betrachte folgendes Modell (Regression) mit $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \mathbb{R}$, und $s_1, s_2 \in \mathbb{R}$, so dass

$$Y = \mathbb{1}_{\{X_1 < s_1\}} + \mathbb{1}_{\{X_2 < s_2\}} + \varepsilon, \tag{6.16}$$

wobei ε ein zufälliger Beobachtungsfehler sei (dieses Beispiel ist entnommen aus [17]). Um die korrekte Struktur zu ermitteln, muss zunächst ein Split entlang der Koordinate X_1 bei s_1 durchgeführt werden und danach in *beiden* entstandenen Bereichen ein Split entlang der Koordinate X_2 bei s_2. Auf Basis der verrauschten Trainingsdaten wird t_2 jedoch in beiden Bereichen unterschiedlich geschätzt ($\hat{s}_2^L \neq \hat{s}_2^R$) und damit eine Struktur der Form

$$\begin{aligned} f_{\hat{T}}(x) = \hat{c}_1 \mathbb{1}_{\{X_1 < \hat{s}_1, X_2 < \hat{s}_2^L\}} + \hat{c}_2 \mathbb{1}_{\{X_1 < \hat{s}_1, X_2 \geq \hat{s}_2^L\}} \\ + \hat{c}_3 \mathbb{1}_{\{X_1 \geq \hat{s}_1, X_2 < \hat{s}_2^R\}} + \hat{c}_4 \mathbb{1}_{\{X_1 \geq \hat{s}_1, X_2 \geq \hat{s}_2^R\}} \end{aligned}$$

erhalten, d. h., die einfache Struktur von Gl. (6.16) wird nicht wiedergegeben. Es gibt komplexere Algorithmen wie MARS (vgl. [17]), die solche Strukturen direkt erfassen können. Wir lernen in Abschn. 6.5 die Technik des Boostings kennen, welche ebenfalls in der Lage ist, gute Algorithmen für Modelle der Form Gl. (6.16) zu liefern.

Wir haben in diesem Abschnitt nur *binäre Bäume* betrachtet, d. h., in jedem Aufteilungsschritt werden aus einer Menge zwei neue Mengen durch einen senkrechten Schnitt erzeugt (sogenannter *binärer Split*). Es ist aus mathematischer Sicht nicht notwendig, Aufteilungen

in mehr Mengen pro Schritt zuzulassen, da dasselbe Resultat durch sukzessive Ausführung von binären Splits erhalten werden kann. In der Praxis werden Techniken, welche in einem Schritt in mehr als zwei Mengen aufteilen, nur bei sehr speziellen Problemstellungen verwendet.

6.4 Bagging

Bagging ist eine Abkürzung für „Bootstrap Aggregating“. Der englische Begriff *bootstrap* bedeutet übersetzt „Schnürsenkel“ – die Methode soll die Ähnlichkeit mit dem Vorgehen ausdrücken, sich selbst an den eigenen Schnürsenkeln hochzuziehen und sich somit aus eigener Kraft und ohne fremde Hilfe auf wundersame Weise hochzuarbeiten. Übertragen auf unsere Situation bedeutet dies, dass die Technik es ermöglicht, ohne weitere Daten oder Wissen von außen einen *bereits vorhandenen* Algorithmus $\hat{f}_n$ zu verbessern. Das Grundkonzept sieht vor, aus den bereits beobachteten Trainingsdaten neue Trainingsdaten (sogenannte *Bootstrap-Stichproben*) zu generieren (bzw. anzusammeln, daher *aggregating*), auf welche dann erneut der Algorithmus $\hat{f}_n$ angewandt werden kann. Man erhält so Ergebnisse des Algorithmus resultierend aus immer neuen Trainingsdaten; durch Kombination der Ergebnisse wird ein stabilerer, neuer Algorithmus erzeugt.

Die Technik ist nicht auf die Anwendung auf Bäume beschränkt, sondern kann auf beliebige Algorithmen angewandt werden. Allerdings zeigt sich in der Praxis, dass besonders bei Bäumen eine starke Verbesserung erreicht werden kann. Wir rekapitulieren zunächst die elementare Boostrap-Technik aus der Sicht der Statistik und wenden diese dann auf CARTs an.

Definition 6.23 (Bootstrap für i. i. d. Beobachtungen – Regression)
Sei $B \in \mathbb{N}$ (Anzahl der Bootstrap-Stichproben).

Gegeben seien i. i. d. reellwertige Zufallsvariablen $Z_i \sim \mathbb{P}^Z, i = 1, \ldots, n$. Die Verteilung $\mathbb{P}^Z$ hänge von einem Parameter $\theta \in \mathbb{R}$ ab, und $\hat{\theta}_n = \hat{\theta}_n(Z_1, \ldots, Z_n)$ sei ein Schätzer für θ basierend auf den Beobachtungen $Z_1, \ldots, Z_n$.

Für $b = 1, \ldots, B$:

- Ziehe zufällig gleichverteilt, mit Zurücklegen $Z_1^{*b}, \ldots, Z_n^{*b}$ aus $\{Z_1, \ldots, Z_n\}$.
- Berechne $\hat{\theta}_n^{*b} = \hat{\theta}_n(Z_1^{*b}, \ldots, Z_n^{*b})$.

Der *Bootstrap-Schätzer* für θ lautet

$$\hat{\theta}_n^{boot} = \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}_n^{*b}.$$

♦

Wir erhalten also in jeder Wiederholung $b \in \{1, \ldots, n\}$ neue Beobachtungen $Z_1^{*b}, \ldots, Z_n^{*b}$ durch simples zufälliges Ziehen aus den ursprünglichen Beobachtungen $Z_1, \ldots, Z_n$. Da das Ziehen *mit Zurücklegen* erfolgt, können einige Z_i in den neuen Beobachtungen doppelt vorkommen. Erst dadurch wird gewährleistet, dass $\{Z_1^{*b}, \ldots, Z_n^{*b}\}$ tatsächlich von $\{Z_1, \ldots, Z_n\}$ verschieden ist.

Der Ausdruck $\hat{\theta}_n^{boot}$ schätzt für großes $B \in \mathbb{N}$ den Term

$$\theta_n := \mathbb{E}[\hat{\theta}_n^{*1} | Z_1, \ldots, Z_n],$$

denn nach dem (bedingten) starken Gesetz der großen Zahlen gilt $\mathbb{P}$-f.s.:

$$\hat{\theta}_n^{boot} = \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}_n^{*b} \to \mathbb{E}[\hat{\theta}_n^{*1} | Z_1, \ldots, Z_n] = \theta_n \qquad (B \to \infty)$$

Die exakte Berechnung von θ_n ist zwar möglich, aber sehr rechenintensiv, da über alle möglichen Kombinationen $(Z_1^*, \ldots, Z_n^*)$ mit $Z_i^* \in \{Z_1, \ldots, Z_n\}$ gewichtet mit deren Wahrscheinlichkeit gemittelt werden müsste. Deswegen nutzt man stattdessen obige Approximation durch $\hat{\theta}_n^{boot}$. Wir zeigen nun, dass $\hat{\theta}_n^{boot}$ im Allgemeinen bessere Eigenschaften als der ursprüngliche Schätzer $\hat{\theta}_n$ besitzt.

Um diesen Vergleich mathematisch durchführen zu können, nutzen wir zusätzlich die Interpretation $\hat{\theta}_n \overset{d}{\approx} \hat{\theta}_n^{*1}$, d. h., wir nehmen an, dass $\hat{\theta}_n$ im Wesentlichen die gleiche Verteilung wie $\hat{\theta}_n^{*1}$ besitzt (beide Schätzer basieren auf einem Satz von Trainingsdaten). Damit erhalten wir

$$\hat{\theta}_n^{boot} - \hat{\theta}_n \quad \overset{d}{\approx} \quad \theta_n - \hat{\theta}_n^{*1}$$

und leiten nun eine exakte mathematische Aussage für die rechte Seite her:

Proposition 6.24 Es gilt $\mathrm{Var}(\theta_n) \leq \mathrm{Var}(\hat{\theta}_n^{*1})$, aber $\mathbb{E}\theta_n = \mathbb{E}\hat{\theta}_n^{*1}$. Das heißt, θ_n hat den gleichen Bias, aber eine kleinere Varianz als $\hat{\theta}_n^{*1}$.

Beweis Die Aussage $\mathbb{E}\theta_n = \mathbb{E}\hat{\theta}_n^{*1}$ ist aufgrund der Turmeigenschaft für bedingte Erwartungswerte klar. Weiter gilt:

$$\begin{aligned}
\mathrm{Var}(\hat{\theta}_n^{*1}) &= \mathbb{E}[(\hat{\theta}_n^{*1} - \mathbb{E}\hat{\theta}_n^{*1})^2] \\
&\overset{\mathbb{E}\hat{\theta}_n^{*1} = \mathbb{E}\theta_n}{=} \mathbb{E}\left[\mathbb{E}[(\hat{\theta}_n^{*1} - \theta_n + \theta_n - \mathbb{E}\theta_n)^2 |\, Z_1, \ldots, Z_n]\right] \\
&= \mathbb{E}\left[\mathbb{E}[(\hat{\theta}_n^{*1} - \theta_n)^2 |\, Z_1, \ldots, Z_n]\right] + \mathbb{E}[(\theta_n - \mathbb{E}\theta_n)^2] \\
&\overset{\theta_n = \mathbb{E}[\hat{\theta}_n^{*1} | Z_1, \ldots, Z_n]}{=} \mathbb{E}\mathbb{E}\mathrm{Var}(\hat{\theta}_n^{*1} | Z_1, \ldots, Z_n) + \mathrm{Var}(\theta_n) \qquad (6.17)
\end{aligned}$$

Die Varianz von θ_n wird konkret geringer um den Term

$$\mathbb{E}\mathrm{Var}(\hat{\theta}_n^{*1} | Z_1, \ldots, Z_n),$$

d. h. um den Erwartungswert der bedingten Varianz von $\hat{\theta}_n^{*1}$ gegeben $Z_1, \ldots, Z_n$. Dieser Term misst, wie stark der ursprüngliche Schätzer $\hat{\hat{\theta}}_n$ variiert, wenn er auf Daten angewandt wird, die zufällig aus den Trainingsdaten gezogen werden. Diese Messung imitiert damit, wie variabel $\hat{\theta}_n$ auf viele verschiedene Ausprägungen von jeweils n Trainingsdaten reagiert. Übertragen auf $\hat{\theta}_n^{boot}$ bedeutet das: $\hat{\theta}_n^{boot}$ stabilisiert $\hat{\theta}_n$, indem durch Mittelung des Verhaltens von $\hat{\theta}_n$ über viele verschiedene (durch zufälliges Ziehen imitierte) Trainingsdaten die Abhängigkeit von deren ursprünglicher Form verringert wird.

6.4.1 Anwendung auf Regressionsbäume

Ist $\lambda > 0$ fest gewählt, so können wir für jedes $x \in \mathcal{X}$ die Bootstrap-Technik aus Definition 6.23 anwenden mit:

- $Z_i = (X_i, Y_i)$,
- $\theta = f^*(x)$ (Bayes-Regel) und
- $\hat{\theta}_n = \hat{f}_n(x, X_1, Y_1, \ldots, X_n, Y_n)$ ein CART-Algorithmus für Regression, z. B. $\hat{f}_n(x) = f_{\hat{T}_{n,\lambda}^{g,p}}(x)$.

Dadurch erhalten wir den sogenannten Bagging-Algorithmus für Regressionsbäume:

Definition 6.25 (Bagging für Regressionsbäume)
Sei $B \in \mathbb{N}$ (Anzahl der Bootstrap-Samples), $x \in \mathcal{X}$.
Gegeben seien $(X_i, Y_i) \overset{\text{iid}}{\sim} \mathbb{P}^{(X,Y)}$, $i = 1, \ldots, n$. Für $b = 1, \ldots, B$:

- Ziehe zufällig gleichverteilt, mit Zurücklegen $(X_1^{*b}, Y_1^{*b}), \ldots, (X_n^{*b}, Y_n^{*b})$ aus $((X_1, Y_1), \ldots, (X_n, Y_n))$.
- Berechne $\hat{f}_n^{*b}(x) = \hat{f}_n(x, X_1^{*b}, Y_1^{*b} \ldots, X_n^{*b}, Y_n^{*b})$.

Der *Bagging-Algorithmus* für Regression lautet

$$\hat{f}_n^{bagg}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_n^{*b}(x). \qquad \blacklozenge$$

Bemerkung Der Parameter λ kann auch individuell für jede Bootstrap-Stichprobe bestimmt werden; dafür sollte jedoch nicht Cross Validation genutzt werden. Der Grund ist, dass die

Stichproben $(X_1^{*b}, Y_1^{*b}), \ldots, (X_n^{*b}, Y_n^{*b})$ oft einige Trainingsdaten doppelt oder sogar mehrfach enthalten und damit Daten gleichzeitig zum Schätzen und zum Validieren verwendet werden, was das Prinzip von Cross Validation konterkariert.

In Abb. 6.9 sind zwei verschiedene $\hat{f}_n^{*b}$ basierend auf Beobachtungen des Modells aus Gl. (6.1) dargestellt sowie das Resultat des zugehörigen Bagging-Algorithmus $\hat{f}_n^{bagg}$ für $B = 1000$. Den Parameter $\lambda = \hat{\lambda}_n$ haben wir zu Beginn einmal basierend auf den ursprünglichen Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$ mittels Bemerkung 6.21 bestimmt.

Die Entscheidungsregel $\hat{f}_n(x)$ basierend auf einem einzigen Baum ist stark abhängig von seiner Struktur der Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$, d. h., $\hat{f}_n$ besitzt eine große

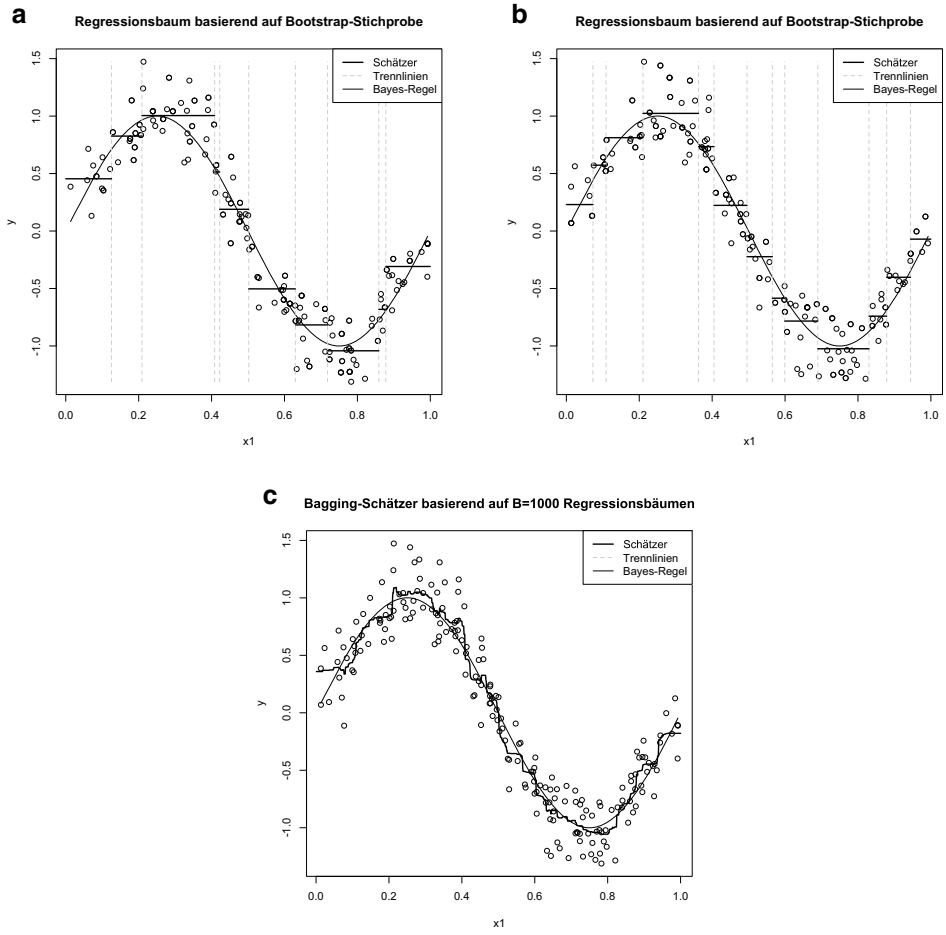


Abb. 6.9 Bagging für Regression: **a**, **b** sind die Resultate eines CART-Algorithmus $\hat{f}_n^{*b}$ für Regression angewandt auf zwei Bootstrap-Stichproben. **c** ist das Resultat des kompletten Bagging-Algorithmus $\hat{f}_n^{bagg}$

Varianz. Durch Bagging kann die Varianz wesentlich reduziert werden. Anschaulich äußert sich das darin, dass die Form von $\hat{f}_n^{bagg}$ wesentlich *glatter* ist als die von $\hat{f}_n$. Allerdings verliert man durch Bagging auch die Interpretierbarkeit der Baumstruktur, denn $\hat{f}_n^{bagg}$ ist nun ein Mittelwert vieler verschiedener Bäume.

6.4.2 Anwendung auf Klassifikationsbäume

Sei nun $\hat{f}_n(x) = \hat{f}_n(x, X_1, Y_1, \ldots, X_n, Y_n)$ ein CART-Algorithmus für Klassifikation mit $\mathcal{Y} = \{1, \ldots, K\}$, z. B. $\hat{f}_n(x) = f_{\hat{T}_{n,\lambda}^{g,p}}(x)$ mit fest gewähltem $\lambda > 0$. Im Gegensatz zur Regression ist hier keine sinnvolle Mittelwertbildung möglich. Stattdessen betrachten wir die Ergebnisse $\hat{f}_n^{*b}(x), b = 1, \ldots, B$ auf den Bootstrap-Stichproben als verschiedene „Meinungen", welche Klasse $k \in \mathcal{Y}$ die richtige für $x \in \mathcal{X}$ ist. Motiviert durch die Anschauung definieren wir den Bagging-Algorithmus als die Klasse, welche bei den B Stichproben am häufigsten ausgewählt wurde (sog. *majority vote*):

Definition 6.26 (Bagging für Klassifikationsbäume per majority vote)
Sei die Notation wie in Definition 6.25. Der *Bagging-Algorithmus* für Klassifikation lautet

$$\hat{f}_n^{bagg}(x) = \underset{k \in \{1,\ldots,K\}}{\arg\max} \#\{b \in \{1, \ldots, B\} : \hat{f}_n^{*b}(x) = k\}. \qquad \blacklozenge$$

Bei der 0-1-Verlustfunktion $L(y, s) = \mathbb{1}_{\{y \neq s\}}$ gilt keine Ungleichung der Form Gl. (6.17), d. h., bei obigem Algorithmus kann man nicht davon ausgehen, dass $\hat{f}_n^{bagg}$ ein geringeres Risiko als der ursprüngliche Algorithmus $\hat{f}_n$ besitzt. Es gibt Spezialfälle, in denen der Algorithmus durch Bagging schlechter werden kann. Man kann aber in einigen Fällen zeigen (zum Beispiel für Baumstümpfe, d. h. Bäume mit Tiefe 1), dass Bagging trotzdem zu einer Reduktion des Risikos führt (vgl. [13, Proposition 2.1, Corollary 2.1]).

Die Entwicklung eines Bagging-Algorithmus, welcher ähnlich wie im Regressionsfall eine Garantie für ein geringeres Risiko gibt, darf daher nicht direkt auf den Resultaten $\hat{f}_n^{*b}(x)$ und der 0-1-Verlustfunktion basieren. Ein typischer Ansatz in der Praxis ist es, den Bagging-Algorithmus nicht direkt auf $\hat{f}_n$, sondern stattdessen auf andere mit $\hat{f}_n(x)$ in Verbindung stehende, stetige Werte anzuwenden, für die eine Ungleichung der Form Gl. (6.17) wieder gilt (mit der quadratischen Verlustfunktion) und für die eine Mittelwertbildung Sinn hat. Im Folgenden sei $\hat{f}_n$ ein Algorithmus, der in der Lage ist, anstatt nur eine Entscheidung für die Klasse $y \in \mathcal{Y}$ von $x \in \mathcal{X}$ auch noch eine Schätzung für die Wahrscheinlichkeit $p_k^*(x) := \mathbb{P}(Y = k | X = x)$ anzugeben. Dann ist es naheliegend, Definition 6.23 auf die stetigen $\theta = p_k^*(x)$ anzuwenden. Wurde ein Klassifikationsbaum durch Trainingsdaten erzeugt, so existiert eine natürliche Schätzung für $p_k^*(x)$:

Definition 6.27
Sei T ein Klassifikationsbaum und (X_i, Y_i), $i = 1, \ldots, n$ Beobachtungen. Sei $\pi(T) = \{A_1, \ldots, A_{\#T}\}$ die Partition des Raumes durch T. Für $A \subset \mathcal{X}$ sei wie in Gl. (6.12) definiert:

$$\hat{p}_k(A) := \frac{\#\{i : X_i \in A, Y_i = k\}}{\#A}$$

Dann sind

$$\hat{p}_{T,k}(x) := \sum_{m=1}^{\#T} \hat{p}_k(A_m) \cdot \mathbb{1}_{A_m}(x), \qquad k = 1, \ldots, K$$

die vom Baum T zugeordneten *Klassenwahrscheinlichkeiten* von x bzgl. der Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$. ♦

Einsetzen in die Definition 6.23 mit $Z_i = (X_i, Y_i)$, $\theta = p_k^*(x) = \mathbb{P}(Y = k|X = x)$ und $\hat{\theta}_n = \hat{p}_{\hat{T}_n,k}(x)$ $(k = 1, \ldots, K)$, wobei $\hat{T}_n(X_1, Y_1, \ldots, X_n, Y_n)$ ein CART-Algorithmus für Klassifiation (z. B. $\hat{T}_n = \hat{T}_{n,\lambda}^{g,p}$), liefert folgende Definition:

Definition 6.28 (Bagging für Klassifikationsbäume per Klassen-WS)
Sei $B \in \mathbb{N}$ (Anzahl der Bootstrap-Stichproben), $x \in \mathcal{X}$. Gegeben seien $(X_i, Y_i) \overset{\text{iid}}{\sim} \mathbb{P}^{(X,Y)}$, $i = 1, \ldots, n$. Für $b = 1, \ldots, B$:

- Ziehe zufällig gleichverteilt, mit Zurücklegen, $(X_1^{*b}, Y_1^{*b}), \ldots, (X_n^{*b}, Y_n^{*b})$ aus $((X_1, Y_1), \ldots, (X_n, Y_n))$.
- Berechne $\hat{p}_k^{*b}(x) = \hat{p}_{\hat{T}_n^{*b},k}(x)$ mit $\hat{T}_n^{*b}(x) = \hat{T}_n(x, X_1^{*b}, Y_1^{*b}, \ldots, X_n^{*b}, Y_n^{*b})$.

Der *Bagging-Algorithmus* mit Klassen-Wahrscheinlichkeiten lautet

$$\hat{f}_n^{bagg}(x) = \underset{k \in \{1,\ldots,K\}}{\arg\max}\, \hat{p}_k^{bagg}(x), \qquad \hat{p}_k^{bagg}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{p}_k^{*b}(x).$$ ♦

Abb. 6.10 zeigt zwei verschiedene $\hat{f}_n^{*b}$ basierend auf Beobachtungen aus dem Modell (6.2) sowie das Resultat des zugehörigen Bagging-Algorithmus $\hat{f}_n^{bagg}$ für $B = 1000$ mit *majority vote* aus Definition 6.26. Der Algorithmus aus Definition 6.28 liefert ein ähnliches Ergebnis.

Wir geben noch zwei abschließende Bemerkungen zu Bagging:

Bemerkung 6.29

- Die Komplexität der Entscheidungsregeln, welche von Bagging durch einen Ausgangsalgorithmus $\hat{f}_n$ erzeugt werden können, hängt wesentlich von der Komplexität der von $\hat{f}_n$ erzeugten Entscheidungsregeln ab. Liefert $\hat{f}_n$ beispielsweise nur Entscheidungsregeln aus einem sehr beschränkten Funktionenraum (zum Beispiel nur sogenannte *Baumstümpfe,* d. h. Bäume mit Tiefe 1), so wird auch $\hat{f}_n^{bagg}$ nur Funktionen aus einem sehr kleinem Funktionenraum liefern können und somit ggf. die Komplexität von f^* nicht darstellen

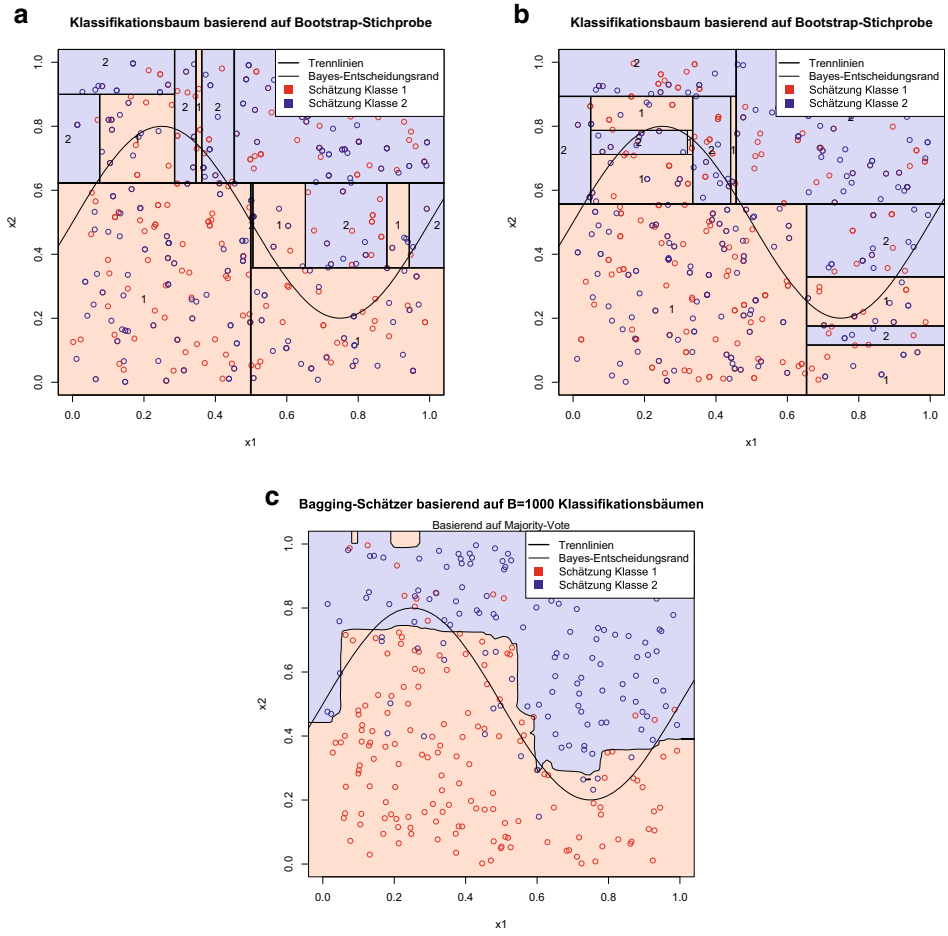


Abb. 6.10 Bagging für Regression: **a**, **b** sind die Resultate eines CART-Algorithmus $\hat{f}_n^{*b}$ für Klassifikation angewandt auf zwei Bootstrap-Stichproben. **c** ist das Resultat des kompletten Bagging-Algorithmus $\hat{f}_n^{bagg}$ mit *majority vote* aus Definition 6.26

können. Die Technik des *majority vote* ist trotz der formal schlechteren Eigenschaften populär und wurde mittels der Random Forests (vgl. Abschn. 6.6) verbessert.

- Ohne konkretere Modellannahmen an f^* unterliegt $\hat{f}_n^{bagg}$ immer noch dem Fluch der Dimension, d. h., das Excess Bayes Risk von $\hat{f}_n^{bagg}$ hat eine Konvergenzrate nicht besser als $n^{-1/d}$. Allerdings wissen wir aufgrund von Proposition 6.24, dass $\hat{f}_n^{bagg}$ für *endliches* n immer einen Algorithmus mit geringerem Risiko liefert (eine bessere *finite sample performance*).

6.5 Boosting

Boosting (verstärken) beschreibt eine Technik, aus einem bereits vorliegenden Algorithmus $\hat{f}_n$ iterativ einen neuen Algorithmus $\hat{f}_n^{boost}$ mit einem geringeren Generalisierungsfehler herzuleiten. In diesem Kontext wird $\hat{f}_n$ Basisalgorithmus genannt, und $\hat{f}_n^{boost}$ Boosting-Algorithmus. Auch wenn der Basisalgorithmus keine besonders komplexen Entscheidungsregeln liefert, d. h. $\hat{f}_n$ nur in einem einfachen Funktionenraum liegt, können für den zugehörigen Boosting-Algorithmus $\hat{f}_n^{boost}$ unter geeigneten Modellannahmen gute Konvergenzraten gezeigt werden. Damit ist Boosting vor allem für Bäume interessant: Mittels eines gierigen Verfahrens kann beispielsweise schnell ein CART mit Tiefe 3 erzeugt werden – damit liegt der zugehörige Algorithmus $\hat{f}_n$ in einem sehr einfachen Funktionenraum. Auch wenn $\hat{f}_n$ aufgrund seiner groben Struktur oft noch keine gute Schätzung für f^* ist, so kann der darauf aufbauende $\hat{f}_n^{boost}$ eine sehr gute Näherung von f^* liefern.

Wir motivieren zunächst das Verfahren. Eine formale Beschreibung des Boosting-Algorithmus erfolgt erst in Definition 6.32. Gegeben Trainingsdaten (X_i, Y_i), $i = 1, \ldots, n$ ist die Grundidee wie folgt: Ausgehend von einer beliebigen Entscheidungsregel $\hat{f}^{(0)}$ wird schrittweise ein Algorithmus $\hat{f}^{(m)}$ aufgebaut ($m \in \mathbb{N}$). Im m-ten Schritt werden durch das bisher erstellte $\hat{f}^{(m)}$ einige Trainingsdaten noch nicht gut erklärt, d. h., $L(Y_i, \hat{f}^{(m)}(X_i))$ ist für einige $i \in \{1, \ldots, n\}$ noch groß. Der Basisalgorithmus $\hat{f}_n$ wird verwendet, um das Verhalten von $\hat{f}^{(m)}$ besonders bei diesen schlecht erklärten Trainingsdaten zu korrigieren und damit einen verbesserten Algorithmus $\hat{f}^{(m+1)}$ zu erzeugen. Während man bei Regressionsproblemen direkt die bisher entstandenen „Fehler“ $Y_i - \hat{f}^{(m)}(X_i)$ in die Erstellung von $\hat{f}_n$ einbezieht, geschieht die Korrektur bei Klassifikation anschaulich durch Anwendung von $\hat{f}_n$ auf modifizierte Trainingsdaten, bei welchen noch nicht gut erklärte Beobachtungen stärker gewichtet und somit bei Anwendung von $\hat{f}_n$ stärker berücksichtigt werden.

Als Grenzwert dieses Vorgehens entsteht $\hat{f}_n^{boost}$.

Wir illustrieren das Vorgehen am Beispiel eines Regressionsproblems mit quadratischer Verlustfunktion und motivieren, warum $\hat{f}_n^{(m)}$ besser arbeiten sollte als $\hat{f}_n$:

Bemerkung 6.30 (Vorgehen und Motivation: Boosting für Regression)

1. Bestimme $\hat{f}_n^{(1)} = \hat{f}_n(X_1, Y_1, \ldots, X_n, Y_n)$.
2. Bestimme die „Fehler“ $\tilde{Y}_i^{(1)} := Y_i - \hat{f}_n^{(1)}(X_i)$.
3. Bestimme $\hat{g}_n^{(2)} = \hat{f}_n(X_1, \tilde{Y}_1^{(1)}, \ldots, X_n, \tilde{Y}_n^{(1)})$, welcher nun versucht, die durch $\hat{f}_n^{(1)}$ gemachten Fehler möglichst gut vorherzusagen.
4. Der Algorithmus $\hat{f}_n^{(2)} := \hat{f}_n^{(1)} + \hat{g}_n^{(2)}$ sollte nun besser arbeiten als $\hat{f}_n^{(1)}$, denn

$$\hat{f}_n^{(2)}(X_i) = (Y_i - \tilde{Y}_i^{(1)}) + (\tilde{Y}_i^{(1)} + \text{Fehler } \hat{g}_n^{(2)}) = Y_i + \text{Fehler } \hat{g}_n^{(2)}.$$

Da $\hat{g}_n^{(2)}$ insgesamt „kleinere“ Werte beschreiben muss als $\hat{f}_n^{(1)}$ (nämlich nur die Fehler aus dem zweiten Schritt), sollte

$$|Y_i - \hat{f}_n^{(2)}(X_i)| < |Y_i - \hat{f}_n^{(1)}(X_i)|, \quad i = 1, \ldots, n$$

und damit $\hat{R}_n(\hat{f}_n^{(2)}) < \hat{R}_n(\hat{f}_n^{(1)})$ gelten.

Das Prinzip wird dann wiederholt, indem $\hat{f}_n$ auf die Daten $(X_i, \tilde{Y}_i^{(2)})$, $i = 1, \ldots, n$ mit den Fehlern $\tilde{Y}_i^{(2)} := Y_i - \hat{f}_n^{(2)}(X_i)$ angewandt wird, um einen Algorithmus $\hat{g}_n^{(3)}$ zu erhalten, usw.

Durch dieses Vorgehen wird ein Algorithmus generiert, der zunächst nur zum Ziel hat, das empirische Risiko $f \mapsto \hat{R}_n(f)$ zu minimieren. Aufgrund der Form von $\hat{f}_n$ ist allerdings auch $\hat{f}_n^{(k)}$ in seiner Form eingeschränkt. Wir werden sehen, dass bei ausreichend „groben" $\hat{f}_n$ und einem rechtzeitigen Abbruch der obigen Iteration eine Überanpassung an die Trainingsdaten verhindert werden kann.

6.5.1 Formale Beschreibung

Im Folgenden nehmen wir an, dass der Basisalgorithmus $\hat{f}_n$ in einer Klasse $\mathcal{C} \subset \{f : \mathcal{X} \to \mathcal{Y}\}$ von einfach strukturierten Entscheidungsfunktionen liege. Beispielsweise bilden sogenannte Baumstümpfe (Bäume mit Tiefe 1) oder allgemeiner CARTs mit Tiefe J eine solche Klasse $\mathcal{C}$:

Beispiel 6.31 Für $J \in \mathbb{N}$ sei

$$\mathcal{C}_J := \{f_T : T \; CART \text{ mit Tiefe } J\}. \tag{6.18}$$

Im Spezialfall $J = 1$ erhalten wir die Klasse der Baumstumpf-Entscheidungsregeln

$$\mathcal{C}_1 = \{y_1 \cdot \mathbb{1}_{\{x_j < s\}} + y_2 \cdot \mathbb{1}_{\{x_j \geq s\}} : y_1, y_2 \in \mathcal{Y}, j \in \{1, \ldots, d\}, s \in \mathbb{R}\}.$$

Das oben im Falle der Regression motivierte Verfahren liefert schrittweise Algorithmen der Form

$$\hat{f}_n^{(M)} = \hat{g}_n^{(1)} + \cdots + \hat{g}_n^{(M)}, \tag{6.19}$$

als deren Limes für $M \to \infty$ der Boosting-Algorithmus $\hat{f}_n^{boost}$ entsteht. Im Falle der Klassifikation ist die Darstellung eines neuen Klassifizierers durch Gl. (6.19) jedoch nicht sinnvoll, da die rechte Seite im Allgemeinen nicht mehr in $\mathcal{Y}$ enthalten ist. Um hier ein ähnlich strukturiertes Verfahren verwenden zu können, nutzen wir den folgenden Trick: Wir betrachten nur noch zwei Klassen $\mathcal{Y} = \{-1, +1\}$ und schätzen statt f^* eine optimale Diskriminantenfunktion $\delta^* : \mathcal{X} \to \mathbb{R}$ mit der Eigenschaft

$$f^*(x) = \text{sign}(\delta^*(x)).$$

Sind $\hat{g}_n^{(m)} : \mathcal{X} \to \mathcal{Y} = \{-1, +1\}$, $m = 1, \ldots, M$ Algorithmen, so gilt $\hat{f}_n^{(M)} \in \{-M, M\}$. Für $x \in \mathcal{X}$ ist beispielsweise $\hat{f}_n^{(M)}(x) = M$, wenn $\hat{g}_n^{(m)}(x) = 1$ für alle $m \in \{1, \ldots, M\}$ gilt. Entsprechend drückt $\hat{f}_n^{(M)}(x)$ mit seiner Größe aus, wie sicher $x \in \mathcal{X}$ der Klasse -1 bzw. $+1$ angehört.

Im oben beschriebenen Vorgehen für Regression (Bemerkung 6.30) nimmt der Einfluss der einzelnen $\hat{g}_n^{(m)}$ auf das Endergebnis $\hat{f}_n^{(M)}$ mit jeder Iteration ab, da die zu beschreibenden Werte $\tilde{Y}_i^{(m)}$ und damit auch die Funktionswerte der Algorithmen $\hat{g}_n^{(m)}$ kleiner werden. Im Klassifikationsfall jedoch liefert aufgrund des Wertebereichs $\mathcal{Y} = \{-1, +1\}$ jeder Summand $\hat{g}_n^{(m)}$ in Gl. (6.19) denselben Beitrag, und der Einfluss auch für hohe m nimmt nicht ab. Um die Konvergenz des Verfahrens und eine Gewichtung der einzelnen Summanden gewährleisten zu können, erweitern wir die Darstellung auf

$$\hat{f}_n^{(M)} = \hat{\beta}_1 \hat{g}_n^{(1)} + \cdots + \hat{\beta}_M \hat{g}_n^{(M)}$$

mit geeigneten Koeffizienten $\hat{\beta}_m \geq 0$. Auch im Regressionsfall kann eine solche Darstellung entstehen, wenn keine quadratische Verlustfunktion genutzt wird, vgl. Definition 6.37.

Bei der mathematischen Beschreibung von $\hat{f}_n^{boost}$ ignorieren wir zunächst das schrittweise Vorgehen zur Ermittlung und formulieren den Limes $\hat{f}_n^{boost}$ nur auf Basis der erhaltenen Summenstruktur.

Definition 6.32 (Boosting-Algorithmus)
Sei $\mathcal{C}$ eine Menge von Entscheidungsfunktionen . Seien $B_m \subset [0, \infty)$ $(m \in \mathbb{N})$ und

$$F = \{x \mapsto \sum_{m=1}^{M} \beta_m \cdot g^{(m)}(x) : M \in \mathbb{N}, \forall m \in \{1, \ldots, M\} : g^{(m)} \in \mathcal{C}, \beta_m \in B_m\}. \quad (6.20)$$

Sei $L : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ eine Verlustfunktion.

- Im Falle $\mathcal{Y} = \mathbb{R}$ (Regression) heißt

$$\hat{f}_n^{boost} :\in \arg\min_{f \in F} \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f(X_i)).$$

- Im Falle $\mathcal{Y} = \{-1, +1\}$ (Klassifikation) heißt

$$\hat{f}_n^{boost} = \text{sign}(\hat{\delta}_n^{boost}), \hat{\delta}_n^{boost} :\in \arg\min_{\delta \in F} \frac{1}{n} \sum_{i=1}^{n} L(Y_i, \delta(X_i))$$

Boosting-Algorithmus basierend auf $\mathcal{C}$. ♦

Man kann zeigen, dass das obige Optimierungsproblem in vielen Fällen näherungsweise durch ein iteratives Bestimmen der Summanden von $\hat{f}_n^{boost}$ bzw. $\hat{\delta}_n^{boost}$ gelöst werden kann (wie eingangs motiviert). Dieses Vorgehen wird *forward stagewise additive modeling* genannt:

Definition 6.33 (Forward stagewise additive modeling)
Sei $M \in \mathbb{N}$. Setze $\hat{f}^{(0)} \equiv 0$. Für $m = 1, 2, 3, \ldots, M$, bestimme

$$(\hat{\beta}_m, \hat{g}^{(m)}) :\in \underset{\beta \in B_m, g \in \mathcal{C}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} L(Y_i, \hat{f}^{(m-1)}(X_i) + \beta \cdot g(X_i)),$$

wobei $\hat{f}^{(m)} := \hat{f}^{(m-1)} + \hat{\beta}_m \cdot \hat{g}^{(m)}$. Dann ist

$$\hat{f}^{(M)}(x) = \sum_{m=1}^{M} \hat{\beta}_m \cdot \hat{g}^{(m)}(x), \quad x \in \mathcal{X}.$$

Im Falle der Klassifikation benutzen wir die Bezeichnung $\hat{\delta}^{(m)}$ anstelle von $\hat{f}^{(m)}$. ♦

In [42, Lemma 4.2/Theorem 4.1] werden Bedingungen formuliert, unter welchen die Approximation

$$\hat{f}^{(M)} \approx \hat{f}_n^{boost} \quad \text{bzw.} \quad \hat{\delta}^{(M)} \approx \hat{\delta}_n^{boost} \tag{6.21}$$

gerechtfertigt ist:

Lemma 6.34 Ist $R(f) = \mathbb{E}L(Y, f(X))$ eine konvexe Funktion in f (d. h., für alle $\alpha \in [0, 1]$, $f_1, f_2 \in F$ gilt $R(\alpha f_1 + (1-\alpha) f_2) \leq \alpha R(f_1) + (1-\alpha) R(f_2)$), und ist

$$\sum_{m=1}^{\infty} \tilde{h}_m = \infty, \qquad \sum_{m=1}^{\infty} \tilde{h}_m^2 < \infty, \qquad \tilde{h}_m := \sup_{\beta \in B_m} \beta, \tag{6.22}$$

so folgt für $x \in \mathcal{X}$: $\hat{f}^{(M)}(x) \to \hat{f}_n^{boost}(x)$ bzw. $\hat{\delta}^{(M)}(x) \to \hat{\delta}_n^{boost}(x)$ $(M \to \infty)$.

Das bedeutet, solange die Größe der Gewichte β_m (gesteuert durch die Menge möglicher Gewichte B_m) für $m \to \infty$ nicht zu schnell und nicht zu langsam absinkt, gilt Gl. (6.21). Durch dieses Resultat ist es gerechtfertigt, das komplizierte Minimierungsproblem von $\hat{f}_n^{boost}$ durch das einfachere, schrittweise Minimierungsproblem vom *forward stagewise additive modeling* zu ersetzen.

Wir nutzen daher $\hat{f}^{(M)}$ bzw. $\hat{\delta}^{(M)}$ zur Bestimmung von expliziten Formeln bzw. Verfahren für Boosting-Algorithmen. Anstelle der exakten Minimierer $(\hat{\beta}_m, \hat{g}^{(m)})$ werden wir jedoch oft nur Näherungen $(\tilde{\beta}_m, \tilde{g}^{(m)})$ nutzen. Tatsächlich erlaubt die Formulierung in [42] derlei Abweichungen, und es ist immer noch möglich, theoretische Resultate nachzuweisen (vgl. Abschn. 6.5.5).

6.5.2 Boosting und Regressionsbäume

Es sei jetzt L die quadratische Verlustfunktion und entsprechend $f^*(x) = \mathbb{E}[Y_1|X_1 = x]$. Ziel dieses Abschnitts ist die Ermittlung expliziter Formeln für einen Boosting-Algorithmus, wenn wir für den Basisalgorithmus einen Regressionsbaum mit Tiefe höchstens $J \in \mathbb{N}$ wählen. Wir ermitteln dazu $\hat{f}_{n,M}^{forward}$ unter Verwendung der quadratischen Verlustfunktion $L(y, s) = (y - s)^2$. Alle möglichen durch den Basisalgorithmus erzeugten Entscheidungsregeln liegen in der Klasse $\mathcal{C}_J$, vgl. Gl. (6.18).

Da $f \in \mathcal{C}_J$ jede reelle Zahl erreichen kann, ist die Klasse F in Gl. (6.20) durch die Gewichte β_m überparametrisiert; durch Weglassen der Gewichte bzw. Setzen von $\beta_m = 1$ ($m \in \mathbb{N}$) können dieselben Funktionen beschrieben werden. Die Formel für $\hat{f}^{(M)}$ reduziert sich daher wieder zu

$$\hat{f}^{(M)}(x) = \sum_{m=1}^{M} \hat{g}^{(m)}(x),$$

mit

$$\begin{aligned}\hat{g}^{(m)} &:\in \arg\min_{g \in \mathcal{C}_J} \frac{1}{n} \sum_{i=1}^{n} L(Y_i, \hat{f}^{(m-1)}(X_i) + g(X_i)) \\ &= \arg\min_{g \in \mathcal{C}_J} \frac{1}{n} \sum_{i=1}^{n} (\tilde{Y}_i^{(m-1)} - g(X_i))^2 = \arg\min_{g \in \mathcal{C}_J} \frac{1}{n} \sum_{i=1}^{n} L(\tilde{Y}_i^{(m-1)}, g(X_i)), \qquad (6.23)\end{aligned}$$

wobei $\tilde{Y}_i^{(m-1)} = Y_i - \hat{f}^{(m-1)}(X_i)$.

Die exakte Lösung dieses Optimierungsproblems ist aufwendig. Wir ermitteln stattdessen eine näherungsweise Lösung mittels eines CART-Algorithmus für Regression, welcher X_i an die Werte $\tilde{Y}_i^{(m-1)}$ anpasst (vgl. Definition 6.15). Zusammengefasst ergibt sich folgendes Verfahren:

Definition 6.35 (Boosting mittels Regressionsbäumen)
Sei $L(y, s) = (y - s)^2$ die quadratische Verlustfunktion. Sei $J \in \mathbb{N}$ und $M \in \mathbb{N}$. Sei $\tilde{f}^{(0)} = \tilde{g}^{(0)} \equiv 0$. Für $m = 1, 2, 3, \ldots, M$, führe durch:

- Setze $\tilde{Y}_i^{(m-1)} := Y_i - \tilde{f}^{(m-1)}(X_i)$.
- Setze $\tilde{g}^{(m)}(x) := f_{\hat{T}_n}(x)$, wobei $\hat{T}_n$ ein Regressionsbaum mit Tiefe J basierend auf den Daten $(X_i, \tilde{Y}_i^{(m-1)})$, $i = 1, \ldots, n$ ist.
- $\tilde{f}^{(m)}(x) = \tilde{f}^{(m-1)}(x) + \tilde{g}^{(m)}(x)$.

Dann heißt $\hat{f}_{n,M}^{boost,\approx}(x) := \tilde{f}^{(M)}(x)$ *genäherter Boosting-Algorithmus* für Regression. ♦

In Abb. 6.11 sind einige $\tilde{f}^{(M)}(x)$ mit $J = 1$ angewandt auf $n = 200$ Beobachtungen des Modells Gl. (6.1) für verschiedene Anzahlen von Iterationen M abgebildet. Zusätzlich

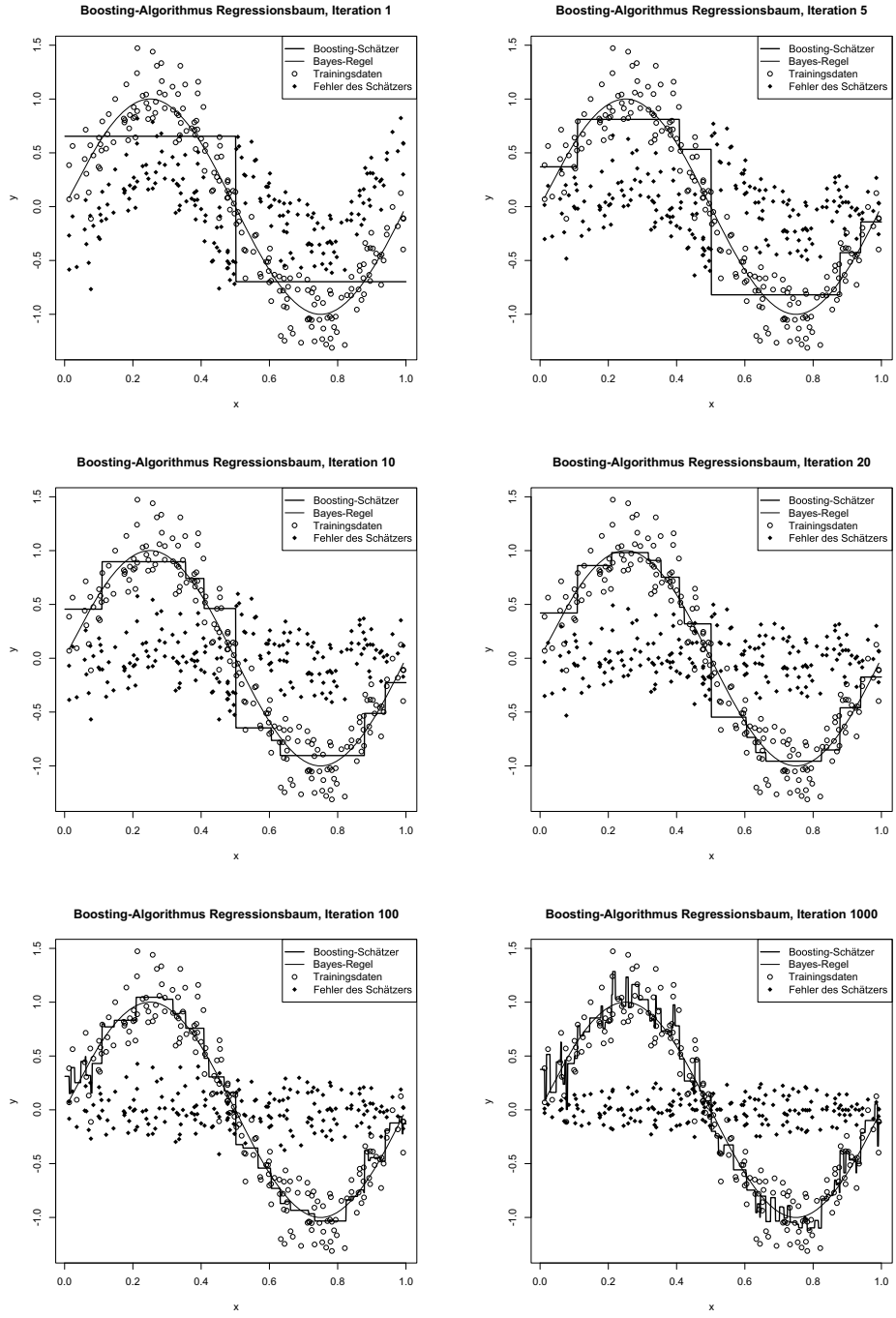


Abb. 6.11 Boosting mit Regressionsbäumen: Darstellung von $\hat{f}_{n,M}^{boost,\approx}$ für verschiedene $M \in \mathbb{N}$. Der Basisalgorithmus entspricht einem Regressionsbaum mit Tiefe $J = 1$

werden die aktuellen Fehler $\tilde{Y}_i^{(M-1)}$ aufgetragen. Man erkennt, dass $\tilde{f}^{(M)}$ für große M (in der Abbildung für $M = 1000$ unten rechts zu sehen) zu Überanpassung neigt. Dies liegt jedoch nicht an der Approximation $\tilde{f}^{(M)} \approx \hat{f}_n^{boost}$, sondern daran, dass auch der Boosting-Algorithmus $\hat{f}_n^{boost}$ selbst überangepasst an die Trainingsdaten ist. Selbst für äußerst einfache Klassen $\mathcal{C}_J$ (d. h. J klein) ist es möglich, ein Element $f \in F$ mit $f(X_i) = Y_i$ $(i = 1, \ldots, n)$ zu finden, wenn die Anzahl der Summanden M nur groß genug gewählt wird, vgl. Abschn. „Modellannahme". Für die spätere theoretische Betrachtung muss daher für sinnvolle Resultate entweder eine Bedingung an die Größe von β_m oder an die Größe von M gestellt werden. In diesem Sinne sind B_m und M als *Hyperparameter* des Verfahrens zu betrachten.

In der Praxis ist eine Begrenzung von M wesentlich einfacher umzusetzen; sie entspricht dem Stoppen des Verfahrens in Definition 6.35 nach einer bestimmten Anzahl von Iterationen (sog. *early stopping*). Diese Anzahl ist wesentlich geringer, als was rein von der Berechnungszeit her möglich wäre.

Fassen wir M als Hyperparameter auf, so können wir das Standardverfahren aus Bemerkung 1.21 zur Auswahl von M nutzen, d. h., wir überwachen den Validierungsfehler während der Boosting-Iterationen. Dazu werden die ursprünglichen Beobachtungen $(X_i, Y_i)_{i=1,\ldots,N}$ aufgeteilt in eine Menge an Trainingsdaten $(X_i, Y_i)_{i=1,\ldots,n}$ $(n < N)$, welche zur Durchführung des Boosting-Algorithmus verwendet wird, und eine Menge $(\bar{X}_i, \bar{Y}_i)_{i=1,\ldots,N-n}$, welche zur Berechnung des Validierungsfehlers verwendet wird mittels

$$\text{empRV}(\tilde{f}^{(M)}) = \frac{1}{N-n} \sum_{i=1}^{N-n} L(\bar{Y}_i, \tilde{f}^{(M)}(\bar{X}_i)). \tag{6.24}$$

Der Validierungsfehler $\text{empRV}(\tilde{f}^{(M)})$ sinkt für wachsendes M zunächst ab, wächst allerdings wieder an, sobald Überanpassung an die Trainingsdaten eintritt. Wir wählen entsprechend $\hat{M} \in \arg\min_{M \in \mathbb{N}} \text{empRV}(\tilde{f}^{(M)})$. In Abb. 6.12 ist der Verlauf von Validierungsfehler und Trainingsfehler

$$\hat{R}_n(\tilde{f}^{(M)}) = \frac{1}{n} \sum_{i=1}^{n} L(Y_i, \tilde{f}^{(M)}(X_i))$$

basierend auf $N = 400$ Beobachtungen zu sehen, wobei wir nur die bereits in Abb. 6.11 dargestellten $n = 200$ Trainingsdaten zur Erstellung des Boosting-Algorithmus und die restlichen $N - n = 200$ zur Berechnung des Validierungsfehlers in Gl. (6.24) genutzt haben. Gemäß Abb. 6.12 erzeugt $M \approx 100$ die niedrigsten Validierungsfehler, die Entscheidungsregel $\hat{f}^{(100)}$ ist in Abb. 6.11 zu sehen.

Prinzipiell ist auch die Anwendung von $\tilde{f}^{(M)}$ mit $J \geq 2$ möglich. Wir werden allerdings in Abschn. 6.5.5 sehen, dass dies bei diesem Modell nicht notwendig ist. Da beim Boosting M der einzige Hyperparameter ist, sollten dessen Werte möglichst zu vielen Entscheidungsregeln $\tilde{f}^{(M)}$ mit niedrigem Risiko führen, aus denen dann mittels des Validierungsfehlers ausgewählt werden kann. Für hohe J liefern jedoch bereits die einzelnen Entscheidungsregeln $\tilde{g}^{(m)}$ gute Beschreibungen der Trainingsdaten, und $\tilde{f}^{(M)}$ ist oft schon mit sehr kleinem M überangepasst. Daher sollte man J in der Praxis nicht zu groß wählen.

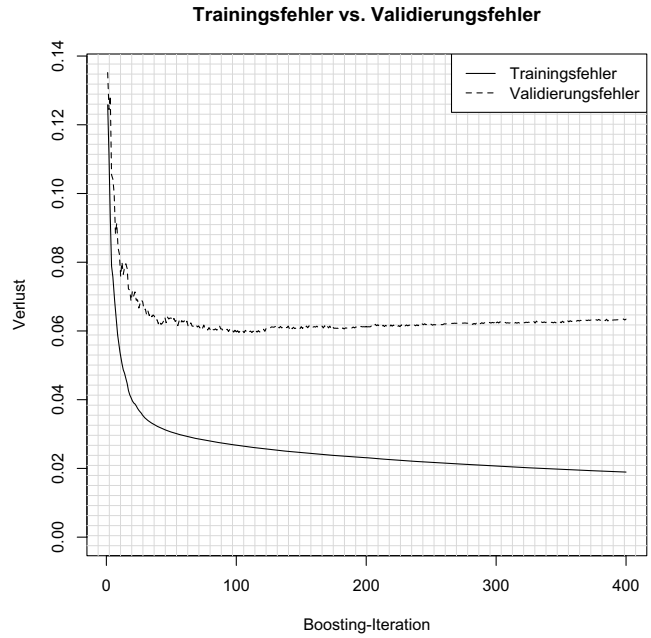


Abb. 6.12 Boosting mit Regressionsbäumen: Überwachung des Risikos von $\hat{f}_{n,M}^{boost,\approx}$ aus Definition 6.35 mittels des Validierungsfehlers

6.5.3 Ausblick: Gradient Boosting mit Regressionsbäumen

Bemerkung 6.36 Je nach Anwendungsproblem werden auch von der quadratischen Verlustfunktion verschiedene Verlustfunktionen L verwendet. Beispiele sind die absolute Verlustfunktion $L(y, s) = |y - s|$ oder der Huber-Verlust

$$L(y,s) = \begin{cases} (y-s)^2, & |y-s| \le \delta, \\ \delta \cdot |y-s|, & |y-s| > \delta \end{cases}$$

mit geeignet gewähltem $\delta > 0$. Diese sorgen für Robustheit des Algorithmus gegenüber Ausreißern, d. h. gegenüber einzelnen Trainingsdaten, welche nicht der Modellannahme entsprechen. Bei Verwendung dieser Verlustfunktionen besitzt Gl. (6.23) keine direkte Interpretation als Regressionsbaum im Sinne von Definition 6.15, der einen quadratischen Verlust minimiert. Für einige Verlustfunktionen sind Modifikationen von Definition 6.15 möglich, indem das Minimierungsproblem in Gl. (6.10) entsprechend für den neuen Verlust gelöst wird. Im Falle von $L(y, s) = |y - s|$ ergibt sich zum Beispiel anstelle des Mittelwerts $\hat{c}_k(j, s) = \sum_{i:X_i \in A_k(j,s)} Y_i$ der Median $\text{med}\{Y_i : X_i \in A_k(j, s)\}$.

Bei komplizierteren, aber zumindest differenzierbaren Verlustfunktionen L nutzt man die Technik des *Gradient Boosting*. Ausgangspunkt dieses Ansatzes ist die Interpretation der Summenstruktur des Boosting-Algorithmus als Iterationen einer Gradientenmethode. Ist das Ziel zunächst die Minimierung von

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f(X_i))$$

über *alle möglichen* Funktionen $f : \mathcal{X} \to \mathcal{Y}$, so unterliegen die verschiedenen Funktionswerte $f(X_i)$, $i = 1, \ldots, n$ keiner Einschränkung. Dann kann die Minimierung in f stattdessen aufgefasst werden als Minimierung in den einzelnen Funktionswerten $(f(X_1), \ldots, f(X_n))^T \in \mathbb{R}^n$, und eine offensichtliche Lösung ist $f(X_i) := Y_i$, $i = 1, \ldots, n$. Diese Feststellung gibt uns jedoch keine Information, wie zu verfahren ist, wenn f Einschränkungen unterliegt. Der Minimierer von $\hat{R}_n(f)$ kann aber auch schrittweise durch eine Gradientenmethode ermittelt werden: Setze $\tilde{f}^{(0)} = 0$ und definiere Funktionen $\tilde{f}^{(m)}$ auf den Werten $X_1, \ldots, X_n$ durch

$$\begin{pmatrix} \tilde{f}^{(m)}(X_1) \\ \vdots \\ \tilde{f}^{(m)}(X_n) \end{pmatrix} = \begin{pmatrix} \tilde{f}^{(m-1)}(X_1) \\ \vdots \\ \tilde{f}^{(m-1)}(X_n) \end{pmatrix} - \beta_m \cdot \partial_{(f(X_1),\ldots,f(X_n))^T} \hat{R}_n(f)\big|_{f=\tilde{f}^{(m-1)}},$$

$$m = 1, 2, \ldots \tag{6.25}$$

mit geeignetem $\beta_m > 0$ (wir sehen gleich, dass diese nicht vorgegeben werden müssen). Setzen wir

$$r_i^{(m-1)} := -\partial_a L(Y_i, a)\big|_{a=\tilde{f}^{(m-1)}(X_i)}, \tag{6.26}$$

so ist Gl. (6.25) äquivalent zu

$$\begin{pmatrix} \tilde{f}^{(m)}(X_1) \\ \vdots \\ \tilde{f}^{(m)}(X_n) \end{pmatrix} = \begin{pmatrix} \tilde{f}^{(m-1)}(X_1) \\ \vdots \\ \tilde{f}^{(m-1)}(X_n) \end{pmatrix} + \beta_m \cdot \begin{pmatrix} r_1^{(m-1)} \\ \vdots \\ r_n^{(m-1)} \end{pmatrix}. \tag{6.27}$$

Definieren wir eine Funktion $\tilde{g}^{(m)}$ auf $X_1, \ldots, X_n$ durch

$$\tilde{g}^{(m)}(X_i) := r_i^{(m-1)}, \quad i = 1, \ldots, n, \tag{6.28}$$

so ist (6.27) äquivalent zu

$$\tilde{f}^{(m)}(X_i) = \tilde{f}^{(m-1)}(X_i) + \beta_m \cdot \tilde{g}^{(m)}(X_i), \quad i = 1, \ldots, n. \tag{6.29}$$

Interpretieren wir die Regel als eine Berechnungsvorschrift für die gesamte Funktion $\tilde{f}^{(m)}$ (d. h. nicht nur auf den Werten $X_1, \ldots, X_n$), so hat Gl. (6.29) genau die Form des *forward stagewise additive modeling* aus Definition 6.33.

Fordern wir nun entsprechend $\tilde{f}^{(m)} \in F$, so muss zwingend $\tilde{g}^{(m)} \in \mathcal{C}$ gelten. Damit Gl. (6.29) dann weiterhin näherungsweise der Anwendung eines Gradientenverfahrens entspricht, muss $\tilde{g}^{(m)} \in \mathcal{C}$ die Bedingung in Gl. (6.28) so gut wie möglich erfüllen. Dies kann beispielsweise erreicht werden durch Lösung von

$$\tilde{g}^{(m)} :\in \arg\min_{g \in \mathcal{C}} \frac{1}{n} \sum_{i=1}^{n} \left(r_i^{(m-1)} - g(X_i) \right)^2 . \tag{6.30}$$

Ist $\tilde{g}^{(m)}$ bestimmt, so kann β_m dem Ansatz aus Definition 6.33 folgend ebenfalls durch ein Minimierungsproblem bestimmt werden, indem Gl. (6.29) validiert wird:

$$\tilde{\beta}_m :\in \arg\min_{\beta \geq 0} \frac{1}{n} \sum_{i=1}^{n} L\left(Y_i, \tilde{f}^{(m-1)}(X_i) + \beta \cdot \tilde{g}^{(m)}(X_i) \right) \tag{6.31}$$

Durch Gl. (6.26), (6.30) und (6.31) ergibt sich dann ein Verfahren, das auch bei komplexeren Verlustfunktionen L eine Näherungslösung für $\hat{f}^{(M)}$ aus dem *forward stagewise additive modeling* liefert. Durch die zweifache Approximation (einerseits durch die Verwendung eines Gradientenverfahrens, andererseits durch die Approximation von Gl. (6.28)) ist jedoch häufig eine Regularisierung der einzelnen erhaltenen Summanden $\tilde{\beta}_m \cdot \tilde{g}^{(m)}$ nötig, um das Verfahren zu stabilisieren. Dies geschieht beispielsweise durch Einführung eines Verkleinerungsfaktors $\nu \in (0, 1]$ (engl. *shrinkage factor/learning rate*), welcher den jeweils neu erhaltenen Summanden abwertet, so dass $\tilde{f}^{(m-1)}$ nur einen „kleinen Schritt" in die durch $\tilde{\beta}_m \cdot \tilde{g}^{(m)}$ vermutete richtige Richtung verschoben wird. Nutzen wir $\mathcal{C} = \mathcal{C}_J$, so führt dies zu folgendem Vorgehen:

Definition 6.37 (Gradienten-Boosting-Algorithmus mit Regressionsbäumen)
Sei L eine beliebige, differenzierbare Verlustfunktion für ein Regressionsproblem. Sei $J \in \mathbb{N}$, $M \in \mathbb{N}$ und $\nu \in (0, 1]$. Sei $\tilde{f}^{(0)} = \tilde{g}^{(0)} \equiv 0$. Für $m = 1, 2, 3, \ldots, M$, führe durch:

1. Setze $r_i^{(m-1)} := -\partial_a L(Y_i, a)\big|_{a=\tilde{f}^{(m-1)}(X_i)}$.
2. Setze $\tilde{g}^{(m)}(x) := f_{\hat{T}_n}(x)$, wobei $\hat{T}_n$ ein Regressionsbaum mit Tiefe J basierend auf den Daten $(X_i, r_i^{(m-1)})$, $i = 1, \ldots, n$ ist.
3. Bestimme $\tilde{\beta}_m :\in \arg\min_{\beta \geq 0} \frac{1}{n} \sum_{i=1}^{n} L\left(Y_i, \tilde{f}^{(m-1)}(X_i) + \beta \cdot \tilde{g}^{(m)}(X_i) \right)$.
4. $\tilde{f}^{(m)}(x) = \tilde{f}^{(m-1)}(x) + \nu \cdot \tilde{\beta}_m \cdot \tilde{g}^{(m)}(x)$.

Dann heißt $\hat{f}_{n,M}^{boost,\approx,grad}(x) := \tilde{f}^{(M)}(x)$ *Gradienten-Boosting-Algorithmus.* ♦

Bemerkungen

- In der Praxis werden häufig relativ kleine Werte für ν, z. B. $\nu \in (0, 0{,}1]$ verwendet.
- Das Optimierungsproblem zur Bestimmung von $\tilde{\beta}_m$ in Schritt (3) ist ein eindimensionales Problem. Lösungen können beispielsweise mit einem Newton-Verfahren ermittelt werden.

6.5.4 Boosting und Klassifikationsbäume

Es sei hier $L(y, s) = \mathbb{1}_{\{y \neq s\}}$ die 0-1-Verlustfunktion. Wie bereits in Abschn. 6.5.1 bemerkt, beschränken wir uns auf Zwei-Klassen-Probleme mit Klassen $\mathcal{Y} = \{-1, +1\}$, d. h. $f^*(x) = \arg\max_{k \in \{-1,+1\}} \mathbb{P}(Y = k | X = x)$, und schätzen statt f^* eine optimale Diskriminantenfunktion $\delta^* : \mathcal{X} \to \mathbb{R}$ mit $\delta^* \in F$ und

$$f^*(x) = \operatorname{sign}(\delta^*(x)), \quad x \in \mathcal{X}, \tag{6.32}$$

mittels

$$\hat{f}_n^{boost} = \operatorname{sign}(\hat{\delta}_n^{boost}(x)), \quad \hat{\delta}_n^{boost} \in \arg\min_{\delta \in F} \frac{1}{n} \sum_{i=1}^{n} \tilde{L}(Y_i, \delta(X_i)),$$

wobei $\tilde{L}$ eine Verlustfunktion ist. Zur praktischen Berechnung kann nicht die 0-1-Verlustfunktion (d. h. $\tilde{L} = L$) genutzt werden, da diese nicht konvex ist und somit weder eine effiziente Minimierung von $\hat{\delta}_n^{boost}$ noch eine Approximation durch das *forward stagewise additive modeling* erlaubt (die Bedingung von Lemma 6.34 ist nicht erfüllt). Stattdessen verwenden wir daher eine Approximation $\tilde{L}$ der 0-1-Verlustfunktion L mit ähnlichen Eigenschaften.

An dieser Stelle verweisen wir auf Abschn. 3.3 und den Standardansatz aus Bemerkung 3.8. Dadurch wird nahegelegt, die natürliche Erweiterung $L_0(y, s) = \mathbb{1}_{\{-ys \geq 0\}}$ von L auf stetige $s \in \mathbb{R}$ durch eine geeignete Verlustfunktion

$$\tilde{L}(y, s) = \phi(-y \cdot s) \tag{6.33}$$

zu ersetzen, wobei ϕ nichtnegativ, konvex und monoton wachsend ist. Boosting kann mit jedem ϕ durchgeführt werden, für welche die Kalibrierungsbedingung gilt, d. h., dass das mit $\hat{\delta}_n^{boost}$ geschätzte δ^* tatsächlich die Gleichung (6.32) erfüllt. Wir zeigen dies für die folgenden beiden Verlustfunktionen in Abschn. 6.5.5:

$$\phi(z) = \phi_1(z) := e^z, \qquad \phi(z) = \phi_2(z) := \log(1 + e^z), \tag{6.34}$$

vgl. Abb. 6.13 für einen Vergleich von L_0 und $\tilde{L}$. $\tilde{L}$ basierend auf ϕ_1 wird auch *exponentielle Verlustfunktion,* $\tilde{L}$ basierend auf ϕ_2 Logit- oder *logistische Verlustfunktion* bezeichnet.

Im Folgenden bestimmen wir $\hat{\delta}_n^{boost}$ aus Definition 6.32 mittels der Approximation $\hat{\delta}^{(M)}$ aus Lemma 6.33 für diese beiden Fälle.

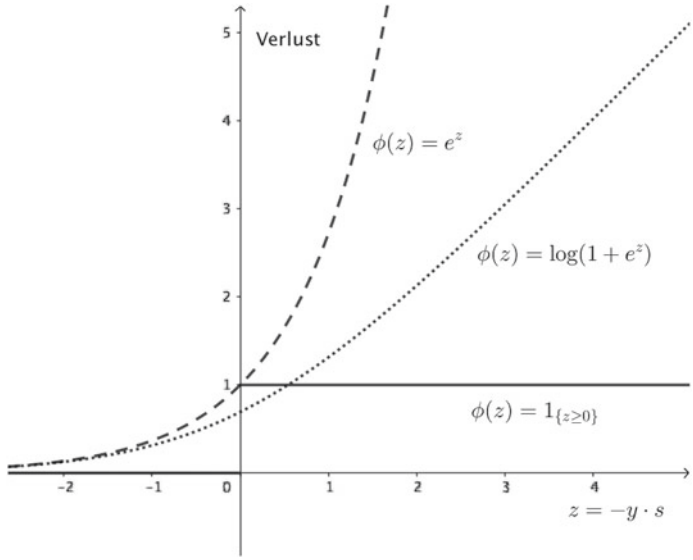


Abb. 6.13 Darstellung verschiedener Funktionen ϕ für $\tilde{L}(y, s) = \phi(y \cdot s)$

Das Verfahren mit $\phi(z) = e^z$
Mit der Wahl $\phi(z) = e^z$ ergeben sich besonders einfache Formeln für $(\hat{\beta}_m, \hat{g}^{(m)})$, die deren nahezu exakte Bestimmung und damit auch eine Interpretation von $\hat{\delta}^{(M)}$ erlauben, was wir hier nachvollziehen:

Im m-ten Schritt sei $\hat{\delta}^{(m-1)}$ gegeben, wir wollen $(\hat{\beta}_m, \hat{g}^{(m)})$ ermitteln. Für $\beta \geq 0$, $g \in \mathcal{C}$ gilt (beachte: $g : \mathcal{X} \to \mathcal{Y} = \{-1, +1\}$):

$$
\begin{aligned}
\Phi(\beta, g) &:= \sum_{i=1}^{n} \tilde{L}(Y_i, \hat{\delta}^{(m-1)}(X_i) + \beta \cdot g(X_i)) \\
&= \sum_{i=1}^{n} \exp\left[-Y_i \cdot (\hat{\delta}^{(m-1)}(X_i) + \beta \cdot g(X_i))\right] \\
&= \sum_{i=1}^{n} w_i^{(m)} \cdot \exp(-\beta Y_i g(X_i)), \qquad w_i^{(m)} := \exp(-Y_i \hat{\delta}^{(m-1)}(X_i)), \\
&\overset{g \in \mathcal{C}}{=} \sum_{i=1}^{n} w_i^{(m)} \left[e^{\beta} \mathbb{1}_{\{Y_i \neq g(X_i)\}} + e^{-\beta} \mathbb{1}_{\{Y_i = g(X_i)\}}\right] \\
&= (e^{\beta} - e^{-\beta}) \cdot \sum_{i=1}^{n} w_i^{(m)} \mathbb{1}_{\{Y_i \neq g(X_i)\}} + e^{-\beta} \cdot \sum_{i=1}^{n} w_i^{(m)}
\end{aligned}
\tag{6.35}
$$

Für jedes $\beta > 0$ gilt

$$\hat{g}^{(m)} := \arg\min_{g \in \mathcal{C}} \Phi(\beta, g) = \arg\min_{g \in \mathcal{C}} \sum_{i=1}^{n} w_i^{(m)} \mathbb{1}_{\{Y_i \neq g(X_i)\}}. \tag{6.36}$$

In diesem Kontext bezeichnet man $w_i^{(m)}$ als *Gewichte,* denn $w_i^{(m)}$ gibt an, wie stark eine Falschklassifizierung bei der Ermittlung von $\hat{g}^{(m)}$ bestraft wurde. Weiter ist

$$\hat{\beta}_m := \arg\min_{\beta \geq 0} \Phi(\beta, \hat{g}^{(m)}) = \frac{1}{2} \log\left(\frac{1 - E(m)}{E(m)}\right), \qquad E(m) := \frac{\sum_{i=1}^{n} w_i^{(m)} \mathbb{1}_{\{Y_i \neq \hat{g}^{(m)}(X_i)\}}}{\sum_{i=1}^{n} w_i^{(m)}}.$$

Mit $\hat{\delta}^{(m)} = \hat{\delta}^{(m-1)} + \hat{\beta}_m \hat{g}^{(m)}$ gilt dann im nächsten Schritt wegen $-Y_i \hat{g}^{(m)}(X_i) = 2 \cdot \mathbb{1}_{\{Y_i \neq \hat{g}^{(m)}(X_i)\}} - 1$:

$$\begin{aligned} w_i^{(m+1)} &= \exp(-Y_i \hat{\delta}^{(m)}(X_i)) = w_i^{(m)} \exp(-Y_i \hat{\beta}_m \hat{g}^{(m)}(X_i)) \\ &= w_i^{(m)} \exp(2\hat{\beta}_m \mathbb{1}_{\{Y_i \neq \hat{g}^{(m)}(X_i)\}}) \exp(-\hat{\beta}_m) \end{aligned} \tag{6.37}$$

Damit gilt also eine Rekursionsgleichung für die Größen $w_i^{(m)}$, $m = 1, \ldots, M$.

Wählen wir $\mathcal{C} = \mathcal{C}_J$, so wird in der Praxis das Optimierungsproblem Gl. (6.36) nicht exakt gelöst, sondern dessen Lösung angenähert. Als Approximation nutzen wir einen Regressionsbaum für Klassifikation, erstellt durch ein gieriges Verfahren. Wir können allerdings nicht direkt die Definition 6.16 nutzen, denn solch ein Baum approximiert einen Minimierer des ungewichteten empirischen Risikos $g \mapsto \hat{R}_n(g) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{Y_i \neq g(X_i)\}}$. Wir modifizieren den Algorithmus aus Definition 6.16 zur Erstellung eines Klassifikationsbaums wie folgt. Seien $w_1, \ldots, w_n \geq 0$ Gewichte und $\hat{R}_n^w(g) := \sum_{i=1}^{n} w_i \mathbb{1}_{\{Y_i \neq g(X_i)\}}$.

Definition 6.38 (Gieriges Verfahren CART/Klassifikation mit Gewichten)
Führe die Konstruktion so durch wie in Definition 6.15, aber ersetze (3) durch

$$\begin{aligned} &\min_{j,s,c_1,c_2} \hat{R}_n^w(f_{T_{neu}}(j, s, c_1, c_2)) \\ &= \min_{j,s,c_1,c_2} \left\{ \text{const.} + \sum_{i: X_i \in A_1(j,s)} w_i \mathbb{1}_{\{Y_i \neq c_1\}} + \sum_{i: X_i \in A_2(j,s)} w_i \mathbb{1}_{\{Y_i \neq c_2\}} \right\} \\ &= \min_{j,s} \left\{ \sum_{i: X_i \in A_1(j,s)} w_i \mathbb{1}_{\{Y_i \neq \hat{c}_1(j,s)\}} + \sum_{i: X_i \in A_2(j,s)} w_i \mathbb{1}_{\{Y_i \neq \hat{c}_2(j,s)\}} \right\} \\ &= \min_{j,s} \left\{ \sum_{i: X_i \in A_1(j,s)} w_i \cdot [1 - \hat{p}^w_{\hat{c}_1(j,s)}(A_1(j,s))] + \sum_{i: X_i \in A_2(j,s)} w_i \cdot [1 - \hat{p}^w_{\hat{c}_2(j,s)}(A_2(j,s))] \right\} \\ &= \min_{j,s} \left\{ \sum_{i: X_i \in A_1} w_i \cdot [1 - \hat{p}^w_{\hat{c}_1}(A_1)] + \sum_{i: X_i \in A_2} w_i \cdot [1 - \hat{p}^w_{\hat{c}_2}(A_2)] \right\}, \end{aligned} \tag{6.38}$$

wobei

$$\hat{p}_k^w(A) := \frac{\sum_{i:Y_i=k, X_i\in A} w_i}{\sum_{i:X_i\in A} w_i}$$

und

$$\hat{c}_1(j,s) = \operatorname*{arg\,max}_{k\in\{1,\dots,K\}} \hat{p}_k^w(A_1(j,s)),$$
$$\hat{c}_2(j,s) = \operatorname*{arg\,max}_{k\in\{1,\dots,K\}} \hat{p}_k^w(A_2(j,s)).$$

Wie in Definition 6.15 wird in der Praxis statt Gl. (6.38) häufig der Gini-Index minimiert:

$$\min_{j,s}\left\{\left(\sum_{i:X_i\in A_1} w_i\right)\cdot\sum_{k=1}^K \hat{p}_k^w(A_1)[1-\hat{p}_k^w(A_1)] + \left(\sum_{i:X_i\in A_2} w_i\right)\cdot\sum_{k=1}^K \hat{p}_k^w(A_2)[1-\hat{p}_k^w(A_2)]\right\}$$

Als Ergebnis des Verfahrens erhalten wir einen CART $\hat{T}_{n,w}^{g,class}$ für Klassifikation. ♦

Mit dieser Vorarbeit können wir den genäherten Boosting-Algorithmus für Klassifikation definieren, der auch unter dem Namen *AdaBoost.M1* bekannt ist (vgl. [16]).

Definition 6.39 (Boosting mittels Klassifikationsbäumen, AdaBoost.M1)
Sei $J \in \mathbb{N}$. Sei $\tilde{\delta}^{(0)} = \tilde{g}^{(0)} = 0$, und $w^{(1)} = \left(w_1^{(1)}, \dots, w_n^{(1)}\right) = \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$, und $x \in \mathcal{X}$ fest. Für $m = 1, 2, 3, \dots$:

1. Sei $\tilde{g}^{(m)}(x) = f_{\hat{T}^{g,class}_{n,w^{(m)}}}(x)$, wobei $\hat{T}^{g,class}_{n,w^{(m)}}$ der Klassifikationsbaum aus Definition 6.38 mit Tiefe J basierend auf den Daten (X_i, Y_i), $i = 1, \dots, n$ ist.
2. $E(m) := \frac{\sum_{i=1}^n w_i^{(m)} \mathbb{1}_{\{Y_i\neq\tilde{g}^{(m)}(X_i)\}}}{\sum_{i=1}^n w_i^{(m)}}$.
3. $\tilde{\beta}_m := \frac{1}{2}\log(\frac{1-E(m)}{E(m)})$.
4. $w_i^{(m+1)} = w_i^{(m)} \cdot \exp\left[2\tilde{\beta}_m \mathbb{1}_{\{Y_i\neq\tilde{g}^{(m)}(X_i)\}}\right]$, $i = 1, \dots, n$.
5. Setze $\tilde{\delta}^{(m)}(x) := \tilde{\delta}^{(m-1)}(x) + \tilde{\beta}_m \tilde{g}^{(m)}(x)$.

Sei $\hat{\delta}_{n,M}^{boost,\approx} := \tilde{\delta}^{(M)}$. Dann heißt $\hat{f}_{n,M}^{boost,\approx}(x) := \text{sign}(\hat{\delta}_{n,M}^{boost,\approx}(x))$ *AdaBoost-Algorithmus* für Klassifikation. ♦

Bemerkungen

- Ist $J = 1$, so entspricht Schritt (1) einer exakten Minimierung über $g \in \mathcal{C}_J$. Je größer J, desto höher ist die Wahrscheinlichkeit, dass $\tilde{g}^{(m)}$ nur eine Näherung des Minimierers ist.
- Die Entwicklung der Gewichte in (4) folgt aus Gl. (6.37). Der Faktor $\exp(-\tilde{\beta}_m)$ in Gl. (6.37) kann weggelassen werden, da er alle Gewichte gleich betrifft und somit das Resultat von Schritt (1) nicht verändert.

- Anschaulich bedeutet (4), dass Trainingsdaten (X_i, Y_i), welche im m-ten Schritt falsch klassifiziert wurden, im nächsten, $(m+1)$-ten Schritt des Verfahrens ein höheres Gewicht erhalten und daher stärker bei der Bestimmung von $\tilde{g}^{(m+1)}$ berücksichtigt werden.
- In der Praxis werden $\tilde{\beta}_m$ oft heuristisch und nicht gemäß (3) gewählt. Man begreift dann den AdaBoost-Algorithmus eher als Durchführung eines Gradientenverfahrens, bei welchem $\tilde{\beta}_m$ die verwendete Schrittweite angibt.

Im Folgenden zeigen wir Resultate für $\hat{f}_{n,M}^{boost,\approx}$ angewandt auf die Daten aus Beispiel 6.3:

Beispiel 6.40 In Abb. 6.11 sind einige $\tilde{f}^{(M)} := \text{sign}(\tilde{\delta}^{(M)}(x))$ angewandt auf $n = 300$ Beobachtungen des Modells Gl. (6.2) für verschiedene Anzahlen von Iterationen M dargestellt. Man erkennt wie zuvor bei Boosting für Regression, dass $\tilde{f}^{(M)}$ für große M (in der Abbildung für $M = 500$ zu sehen) zu Überanpassung neigt.

In Abb. 6.15 ist erneut der Verlauf von Validierungsfehler und Trainingsfehler (gemessen mit der 0-1-Verlustfunktion) basierend auf $N = 600$ Beobachtungen zu sehen, wobei wir nur die bereits in Abb. 6.14 dargestellten $n = 300$ Trainingsdaten zur Erstellung des Boosting-Algorithmus und die restlichen $N - n = 300$ zur Berechnung des Validierungsfehlers genutzt haben. Gemäß Abb. 6.15 erzeugt $M \approx 50$ den niedrigsten Validierungsfehler, die Entscheidungsränder zu $\tilde{f}^{(50)}$ sind in Abb. 6.14 zu sehen.

Das Verfahren mit $\phi(z) = \log(1 + e^z)$

Für die Konstruktion des AdaBoost-Algorithmus wurde die Verlustfunktion $\tilde{L}(y, s) = \phi(-ys)$ mit $\phi(z) = e^z$ verwendet. Für das oben motivierte $\phi(z) = \log(1 + e^z)$ können nicht *unmittelbar* explizite Formeln wie in Definition 6.39 erhalten werden, denn der entsprechende Ausdruck für $\Phi(\beta, g)$ in Gl. (6.35) lautet

$$\Phi(\beta, g) = \sum_{i=1}^{n} \log\left(\frac{1 + w_i^{(m)} e^{\beta}}{1 + w_i^{(m)} e^{-\beta}}\right) \mathbb{1}_{\{Y_i \neq g(X_i)\}} + \sum_{i=1}^{n} \log(1 + w_i^{(m)} e^{-\beta}), \tag{6.39}$$

dessen Minimierung sich nicht mehr getrennt in die Minimierung über $\beta \geq 0$ und die Minimierung über $g \in \mathcal{C}$ separieren lässt. Da man für β im Allgemeinen kleine Werte erwartet (d. h. $\beta \approx$), kann $\Phi(\beta, g)$ mittels einer Taylor-Entwicklung vereinfacht werden: Wegen $\log(1 + a + z) \approx \log(1 + a) + \frac{z}{1+a}$ für $a \geq 0$, $z \in \mathbb{R}$ klein gilt

$$\log(1 + w_i^{(m)} e^{\pm\beta}) \approx \log(1 + w_i^{(m)}) + \frac{1}{1 + w_i^{(m)}} \cdot w_i^{(m)}(e^{\pm\beta} - 1)$$

und daher

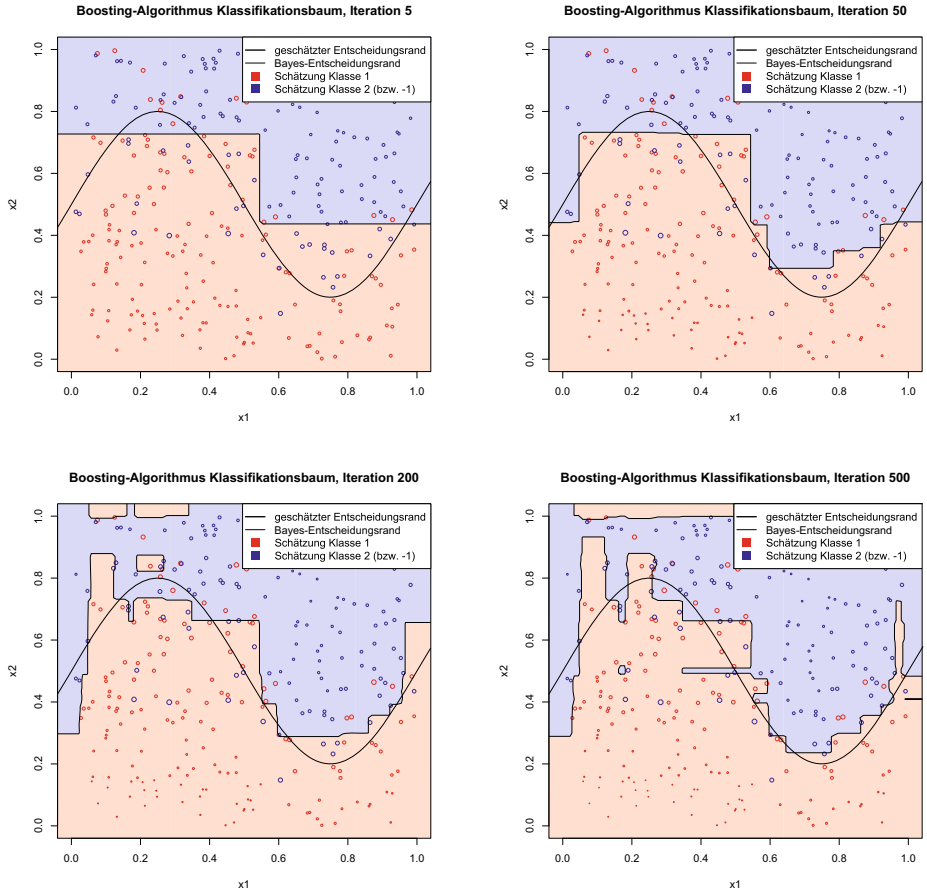


Abb. 6.14 Boosting mit Klassifikationsbäumen: Darstellung der Entscheidungsränder von $\hat{f}_{n,M}^{boost,\approx}$ für verschiedene $M \in \mathbb{N}$. Der Basisalgorithmus entspricht einem Klassifikationsbaum mit Tiefe $J = 1$

$$\Phi(\beta, g) \approx (e^{\beta} - e^{-\beta}) \cdot \sum_{i=1}^{n} \frac{w_i^{(m)}}{1 + w_i^{(m)}} \mathbb{1}_{\{Y_i \neq g(X_i)\}} + e^{-\beta} \cdot \sum_{i=1}^{n} \frac{w_i^{(m)}}{1 + w_i^{(m)}} + \sum_{i=1}^{n} \left\{ \log(1 + w_i^{(m)}) - \frac{w_i^{(m)}}{1 + w_i^{(m)}} \right\}.$$

Durch einen Vergleich mit der Struktur in Gl. (6.35) können wir das Berechnungsverfahren direkt aus dem AdaBoost-Algorithmus abschreiben:

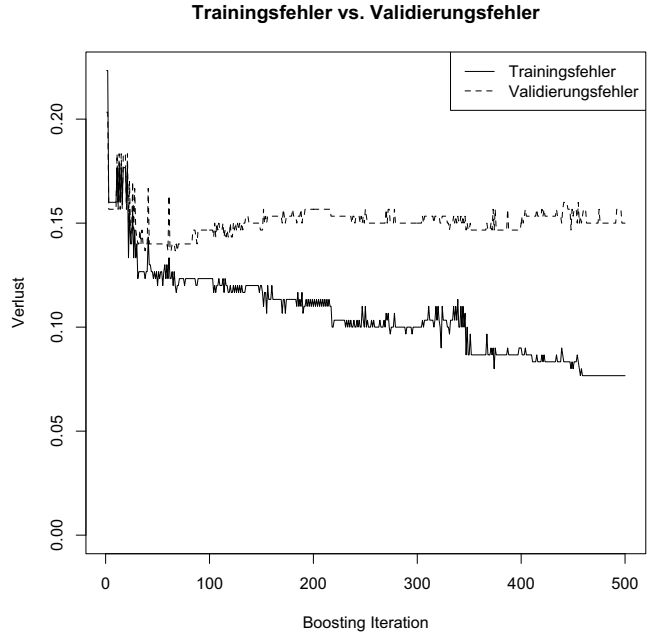


Abb. 6.15 Boosting mit Klassifikationsbäumen: Überwachung des Risikos von $\tilde{f}^{(M)}$ aus Definition 6.39 mittels des Validierungsfehlers

Definition 6.41 (Boosting mittels Klassifikationsbäumen, LogitBoost)
Sei $J \in \mathbb{N}$. Sei $\tilde{\delta}^{(0)} = \tilde{g}^{(0)} = 0$, und $w^{(1)} = \left(w_1^{(1)}, \ldots, w_n^{(1)}\right) = \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$, und $x \in \mathcal{X}$ fest. Für $m = 1, 2, 3, \ldots$:

1. Setze $\bar{w}^{(m)} := (\bar{w}_1^{(m)}, \ldots, \bar{w}_n^{(m)})$ mit $\bar{w}_i^{(m)} := \frac{w_i^{(m)}}{1+w_i^{(m)}}$, $i = 1, \ldots, n$.
2. Sei $\tilde{g}^{(m)}(x) = f_{\hat{T}^{g,class}_{n,\bar{w}^{(m)}}}(x)$, wobei $\hat{T}^{g,class}_{n,\bar{w}^{(m)}}$ der Klassifikationsbaum aus Definition 6.38 mit Tiefe J basierend auf den Daten (X_i, Y_i), $i = 1, \ldots, n$.
3. $E(m) := \frac{\sum_{i=1}^n \bar{w}_i^{(m)} \mathbb{1}_{\{Y_i \neq \tilde{g}^{(m)}(X_i)\}}}{\sum_{i=1}^n \bar{w}_i^{(m)}}$.
4. $\tilde{\beta}_m := \frac{1}{2} \log(\frac{1-E(m)}{E(m)})$.
5. $w_i^{(m+1)} = w_i^{(m)} \cdot \exp\left[2\tilde{\beta}_m \mathbb{1}_{\{Y_i \neq \tilde{g}^{(m)}(X_i)\}}\right] \exp(-\tilde{\beta}_m)$, $i = 1, \ldots, n$.
6. Setze $\tilde{\delta}^{(m)}(x) := \tilde{\delta}^{(m-1)}(x) + \tilde{\beta}_m \tilde{g}^{(m)}(x)$.

Sei $\hat{\delta}^{boost,\approx}_{n,M} := \tilde{\delta}^{(M)}$. Dann heißt $\hat{f}^{boost,\approx}_{n,M}(x) := \text{sign}(\hat{\delta}^{boost,\approx}_{n,M}(x))$ *LogitBoost-Algorithmus* für Klassifikation. ♦

Der einzige Unterschied im Vergleich zum AdaBoost-Algorithmus ist also die Verwendung stabilisierter Gewichte $\bar{w}^{(m)}$ während der Erzeugung eines Klassifikationsbaums. Angewandt auf die Daten aus Beispiel 6.3 ergeben sich ähnliche Ergebnisse wie in Abb. 6.14 und 6.15.

6.5.5 Theoretische Resultate

Wir diskutieren hier theoretische Resultate für Klassifikationsprobleme mit $K = 2$ Klassen und $\mathcal{Y} = \{-1, +1\}$. Für Regressionsprobleme gelten ähnliche Aussagen. Es bezeichne $L(y, s) = \mathbb{1}_{\{y \neq s\}}$ die 0-1-Verlustfunktion und $\tilde{L}(y, s) := \phi(-y \cdot s)$ mit $\phi(z) = e^z$ und $\phi(z) = \log(1 + e^z)$ (vgl. Gl. 6.34).

Wir nutzen nun die in Abschn. 3.3 dargestellte Theorie, um zu zeigen, dass bei Durchführung des Boostings mit $\tilde{L}$ und $\hat{\delta}_n^{boost}$ tatsächlich auch erwartet werden kann, dass das Excess Bayes Risk von $\hat{f}_n^{boost}$ bzgl. der 0-1-Verlustfunktion gegen null konvergiert.

Lemma 6.42 (Kalibrierungsbedingung und Risikoübertragung) Sei $\tilde{L}(y, s) = \phi(-ys)$ und $\eta(x) := \mathbb{P}(Y = 1|X = x)$. Sei

$$\delta^* \in \arg\min_{\delta: \mathcal{X} \to \mathbb{R}} \tilde{R}(\delta), \qquad \tilde{R}(\delta) := \mathbb{E}\tilde{L}(Y, \delta(X)).$$

(i) (K1) Dann gilt:
- falls $\phi(z) = e^z$, so erfüllt

$$\delta^*(x) = \frac{1}{2} \log\left(\frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = -1|X = x)}\right),$$

- falls $\phi(z) = \log(1 + e^z)$, so erfüllt

$$\delta^*(x) = \log\left(\frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = -1|X = x)}\right)$$

$\mathbb{P}^X$-f.s. auf $\{x \in \mathcal{X} : \eta(x) \in \{0, \frac{1}{2}, 1\}\}$. In beiden Fällen gilt insbesondere $\mathbb{P}^X$-f.s. auf $\{x \in \mathcal{X} : \eta(x) \in \{0, \frac{1}{2}, 1\}\}$:

$$f^*(x) = \text{sign}(\delta^*(x)), \quad x \in \mathcal{X}$$

(ii) (K2) In beiden Fällen gilt für jedes $\delta : \mathcal{X} \to \mathbb{R}$ mit $f = \text{sign}(\delta)$:

$$R(f) - R(f^*) \leq 2\sqrt{2}\left(\tilde{R}(\delta) - \tilde{R}(\delta^*)\right)^{1/2} \tag{6.40}$$

Gibt es zusätzlich $\eta_0 > 0$ mit $\mathbb{P}(|\eta(X) - \frac{1}{2}| \leq \frac{1}{2\eta_0}) = 0$, so gilt sogar

$$R(f) - R(f^*) \leq 4\sqrt{2}\eta_0 \cdot \left(\tilde{R}(\delta) - \tilde{R}(\delta^*)\right). \tag{6.41}$$

Beweis

(i) Wir wenden Lemma 3.10 an und erhalten:

$$\delta^*(x) \in \arg\min_{z\in\mathbb{R}} \Phi_{\eta(x)}(z),$$

$\mathbb{P}^X$-f.s. auf $\{x \in \mathbb{R}^d : \eta(x) \notin \{0, \frac{1}{2}, 1\}\}$, wobei

$$\Phi_\eta(z) = \phi(-z)\eta + \phi(z)(1-\eta).$$

Im Falle $\phi(z) = e^z$ ist

$$0 \overset{!}{=} \Phi'_{\eta(x)}(z) = -e^{-z}\eta(x) + e^z(1-\eta(x)) \quad \Longleftrightarrow \quad e^{2z} = \frac{\eta(x)}{1-\eta(x)},$$

im Falle $\phi(z) = \log(1+e^z)$ ist

$$0 \overset{!}{=} \Phi'_{\eta(x)}(z) = -\frac{e^{-z}}{1+e^{-z}}\eta(x) + \frac{e^z}{1+e^z}(1-\eta(x)) \quad \Longleftrightarrow \quad e^z = \frac{\eta(x)}{1-\eta(x)},$$

was jeweils die entsprechende Form von $\delta^*(x)$ liefert.
Insbesondere gilt in beiden Fällen

$$\begin{aligned}\delta^*(x) > 0 &\Longleftrightarrow \mathbb{P}(Y=1|X=x) > \mathbb{P}(Y=-1|X=x)\\ &\Longleftrightarrow f^*(x) = \arg\max_{k\in\{-1,+1\}} \mathbb{P}(Y=k|X=x) = 1.\end{aligned}$$

(ii) Wir wenden Lemma 3.11 an. Man kann zeigen, dass die Voraussetzungen mit $s = \sqrt{2}$, $c = \sqrt{2}$ erfüllt sind. Unter der Zusatzbedingung ist Lemma 3.14 mit $\alpha = 1$ und $\beta = \eta_0$ anwendbar und liefert die Behauptung.

Modellannahme

Wir untersuchen nun, unter welchen Modellannahmen an f^* bzw. den optimalen Entscheidungsrand eine gute Performance des Boosting-Algorithmus zu erwarten ist.

Da

$$\hat{\delta}_{n,M}^{boost,\approx} \approx \hat{\delta}^{(M)} \to \hat{\delta}_n^{boost} \in F \quad (M \to \infty)$$

gilt und für $n \to \infty$ die rechte Seite in $\overline{F}$ liegt (wobei $\overline{F}$ den Abschluss von F in $\{\delta : \mathcal{X} \to \mathbb{R} \text{ messbar}\}$ bzgl. punktweiser Konvergenz bezeichnet), erwarten wir, dass $\hat{\delta}_{n,M}^{boost,\approx}$ jede Funktion in $\overline{F}$ beliebig genau approximieren kann. Daher stellen wir folgende Modellannahme an δ^*:

Definition 6.43 (Vorläufige Modellannahme: Boosting)
Es gelte $\delta^* \in \overline{F}$. ♦

Wegen Lemma 6.42 bedeutet das in beiden Fällen $\phi(z) = e^z$ und $\phi(z) = \log(1 + e^z)$, dass

$$\left[x \mapsto \log\left(\frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = -1|X = x)} \right) \right] \in \overline{F}.$$

Der Faktor $\frac{1}{2}$ ist dabei im Falle $\phi(z) = e^z$ nicht von Bedeutung, da $\overline{F}$ skalierungsinvariant ist.

Es schließt sich die Frage an, welche Menge an Funktionen durch $\overline{F}$ beschrieben wird. Die Struktur von F hängt wesentlich von der Menge $\mathcal{C}$ der gewählten Entscheidungsfunktionen des Boosting-Algorithmus ab (und damit von den gewählten Basisalgorithmen). Im vorherigen Abschnitt haben wir die Basisalgorithmen der Klassifikationsbäume mit Tiefe J betrachtet, deswegen interessiert uns besonders das Aussehen von $\overline{F}$ für $\mathcal{C} = \mathcal{C}_J$. Wir schreiben im Folgenden auch $\mathcal{C}_J = \mathcal{C}_J(d)$, um die Dimension des Ausgangsraums in die Notation aufzunehmen.

Wir zitieren hier einige Ergebnisse aus [8, Section 6]. Parallel dazu geben wir die sogenannte *Vapnik-Chervonenkis-Dimension* (kurz *VC-Dimension*) $\mathrm{VC}(\mathcal{C})$ der Mengen $\mathcal{C}$ an, die später für die Darstellung der Theoreme benötigt wird. Anschaulich misst $\mathrm{VC}(\mathcal{C})$ die „Größe“ einer Menge von Funktionen $\mathcal{C}$ durch die Vielfalt ihrer Graphen. Eine formale Definition findet sich in [35, Kap. 19].

Lemma 6.44 (Gestalt von $\overline{F}$) Sei $\mathcal{X} = \mathbb{R}^d$.

(i) Ist

$$\begin{aligned}\mathcal{C} = \mathcal{C}_1(d) &= \{f_T : \mathcal{X} \to \{-1, +1\} : T \text{ ist Klassifikationsbaum mit Tiefe } 1\} \\ &= \{x \mapsto y_1 \mathbb{1}_{\{x_j < s\}} + y_2 \mathbb{1}_{\{x_j \geq s\}} : y_1, y_2 \in \{-1, +1\}, j \in \{1, \ldots, d\}, s \in \mathbb{R}\},\end{aligned}$$

so gilt

$$\overline{F} = \{\delta : \mathcal{X} \to \mathbb{R} : \delta(x) = \sum_{j=1}^{d} h_j(x_j), h_j : \mathbb{R} \to \mathbb{R} \text{ messbar}\},$$

und $\mathrm{VC}(\mathcal{C}_1(d)) \leq 2 \log_2(2d)$.

(ii) Ist

$$\mathcal{C} = \mathcal{C}_2(d) = \{f_T : \mathcal{X} \to \{-1, +1\} : T \text{ ist Klassifikationsbaum mit Tiefe } 2\},$$

so gilt

$$\overline{F} = \{\delta : \mathcal{X} \to \mathbb{R} : \delta(x) = \sum_{j,k=1}^{d} h_{j,k}(x_j, x_k), h_{j,k} : \mathbb{R}^2 \to \mathbb{R} \text{ messbar}\}.$$

(ii) Ist

$$\mathcal{C} = \mathcal{C}(d) := \{f_T : \mathcal{X} \to \{-1, +1\} : T \text{ ist Klassifikationsbaum mit } d+1 \text{ Blättern}\},$$

so gilt

$$\overline{F} = \{\delta : \mathcal{X} \to \mathbb{R} \text{ messbar}\},$$

und $\mathrm{VC}(\mathcal{C}(d)) \leq d \log_2(2d)$.

Bemerkung In der Situation von (i) gilt zusammen mit der Modellannahme aus Definition 6.45 für die optimale Entscheidungsregion der Klasse $+1$:

$$\Omega_1^* = \{x : f^*(x) = 1\} = \{x : \delta^*(x) > 0\} = \left\{x : \sum_{j=1}^{d} h_j(x_j) > 0\right\} \tag{6.42}$$

mit messbaren Abbildungen $h_j : \mathbb{R} \to \mathbb{R}$. Ist der Basisalgorithmus also ein CART der Tiefe $J = 1$, so müssen die optimalen Entscheidungsregionen notwendigerweise eine Darstellung der Form Gl. (6.42) besitzen. In der Situation (ii), d. h. für CARTs mit Tiefe $J = 2$, ergibt sich entsprechend

$$\Omega_1^* = \{x : f^*(x) = 1\} = \{x : \delta^*(x) > 0\} = \left\{x : \sum_{j,k=1}^{d} h_{j,k}(x_j, x_k) > 0\right\} \tag{6.43}$$

mit messbaren Abbildungen $h_{j,k} : \mathbb{R}^2 \to \mathbb{R}$. Um eine gute Wahl von J im AdaBoost-Algorithmus Definition 6.39 zu treffen, muss also zumindest ungefähr bekannt sein, wie viele Koordinaten bei der Beschreibung des Entscheidungsrandes miteinander interagieren. Bei hochdimensionalen Anwendungsproblemen wird häufig $4 \leq J \leq 8$ gewählt, vgl. [17].

Für die Klasse $\mathcal{C}$ können natürlich auch Entscheidungsfunktionen gewählt werden, die nicht durch Bäume entstanden sind. Dann muss die Bestimmung der $\hat{g}^{(m)}$ bei den in Abschn. 6.5.4 besprochenen Algorithmen entsprechend angepasst werden.

Je vielfältiger die Klasse $\mathcal{C}$, desto größer wird auch die VC-Dimension $VC(\mathcal{C})$. Wir werden im Folgenden sehen, dass sich damit automatisch auch die Konvergenzrate des Excess Bayes Risks von $\hat{f}_n^{boost}$ verschlechtert. Man sollte daher stets eine an das Problem (d. h. an die Struktur von f^* bzw. δ^*) angepasste Wahl von $\mathcal{C}$ treffen und $\mathcal{C}$ nicht zu groß wählen.

Approximationsfehler

Auch wenn durch $\hat{\delta}_{n,M}^{boost,\approx}$ prinzipiell jede Funktion aus $\overline{F}$ beliebig genau approximiert werden kann, so ist $\hat{\delta}_{n,M}^{boost,\approx}$ doch immer nur ein Element von F. In diesem Sinne tritt durch die Modellannahme aus Definition 6.43 ein Approximationsfehler der Form

$$\inf_{\delta \in F} \tilde{R}(\delta) - \tilde{R}(\delta^*)$$

auf, der bei der Herleitung theoretischer Resultate für das Excess Bayes Risk $\tilde{R}(\hat{\delta}_{n,M}^{boost,\approx}) - \tilde{R}(\delta^*)$ berücksichtigt werden müsste. Um dies zu umgehen, nehmen wir im Folgenden an, dass tatsächlich nur $\delta^* \in F$ gilt, womit kein Approximationsfehler vorliegt. In diesem Sinne betrachten wir die Menge $\overline{F}$ nur als Hinweis darauf, welche Funktionen mit Boosting gut approximiert werden können.

Modellannahme 6.45 (Modellannahme: Boosting) Es gelte $\delta^* \in F$.

Excess Bayes Risk

Wir sind an Abschätzungen für das Excess Bayes Risk $R(\hat{f}_{n,M}^{boost,\approx}) - R(f^*)$ der von uns eingeführten Klassifizierer $\hat{f}_{n,M}^{boost,\approx}$ interessiert. In Lemma 6.42(ii) haben wir bereits gesehen, dass dieses auf $\tilde{R}(\hat{\delta}_{n,M}^{boost,\approx}) - \tilde{R}(\delta^*)$ zurückgeführt werden kann.

Da

$$\hat{\delta}_n^{boost} \in \arg\min_{\delta \in F} \tilde{R}_n(\delta), \qquad \tilde{R}_n(\delta) := \frac{1}{n} \sum_{i=1}^{n} \tilde{L}(Y_i, \delta(X_i))$$

ein direkter Minimierer des empirischen Risikos $\tilde{R}_n$ *ohne* Bestrafungsterm und die Klasse F sehr groß ist, neigt $\hat{\delta}_n^{boost}$ zu Überanpassung und wir können von diesem Schätzer keine guten statistischen Eigenschaften erwarten. Tatsächlich nutzen wir in der Praxis jedoch nicht $\hat{\delta}_n^{boost}$, sondern die Approximation $\hat{\delta}_{n,M}^{boost,\approx}$, für die je nach Wahl von M Überanpassung vermieden werden kann.

Wir zeigen hier zwei theoretische Resultate. Zur Formulierung benötigen wir ein Maß für die Komplexität der Funktion δ^*, was durch die Summe der zur Darstellung benötigten Gewichte ausgedrückt werden kann. Wir definieren für $\delta \in F$:

$$\|\delta\|_{1,\mathcal{C}} := \inf \left\{ \sum_{m=1}^{M} \beta_m : \delta = \sum_{m=1}^{M} \beta_m \cdot g_m \in F, M \in \mathbb{N} \right\}$$

Da es mehrere Darstellungen für $\delta \in F$ geben kann, wird das Infimum über alle möglichen Darstellungen gebildet.

Das erste Resultat ist in der Lage, die Auswirkungen der näherungsweisen Minimierungen in $\hat{\delta}_{n,M}^{boost,\approx}$ (die zumindest bei komplexeren Klassen $\mathcal{C}$ auftreten) zu quantifizieren und die stabilisierende Wirkung eines frühen Stoppens (engl. *early stopping*) im Algorithmus zu berücksichtigen. In [42, Theorem 3.2, mit $\beta_m \approx m^{1/4}$ darin, Section 4.3] wird folgender Satz gezeigt:

Satz 6.46 Sei $\mathcal{C} \subset \{f : \mathcal{X} \to \mathcal{Y} \text{ messbar}\}$ eine Klasse von Entscheidungsregeln und $\tilde{L}(y, s) = \phi(-ys)$ mit $\phi(x) = \log(1 + e^x)$. Es gelte die Modellannahme 6.45. Weiter gelten folgende Aussagen:

(i) ϵ_m, $m \in \mathbb{N}$ ist eine Folge positiver reeller Zahlen mit $\epsilon_m \leq \frac{1}{2n}$ und $\sum_{m=1}^{\infty} \epsilon_m < \infty$.
(ii) $B_m \subset [0, \frac{1}{\sqrt{n}}]$ ist so gewählt, dass Gl. (6.22) erfüllt ist.
(iii) Es ist $\tilde{\delta}^{(0)} = 0$, und für $m = 1, 2, 3, \ldots$ sind die Parameter $\tilde{\beta}_m \in B_m$, $\tilde{g}^{(m)} \in \mathcal{C}$ so gewählt, dass

$$\tilde{R}_n(\tilde{\delta}^{(m-1)} + \tilde{\beta}^{(m)} \cdot \tilde{g}^{(m)}) \leq \inf_{\beta \in B_m, g \in \mathcal{C}} \tilde{R}_n(\tilde{\delta}^{(m-1)} + \beta \cdot g) + \varepsilon_m, \tag{6.44}$$

und es gelte $\tilde{\delta}^{(m)} = \tilde{\delta}^{(m-1)} + \tilde{\beta}_m \cdot \tilde{g}^{(m)}$, $m \in \mathbb{N}$.

Dann gilt für $M = \|\delta^*\|_{1,\mathcal{C}}^{1/2} \cdot n^{3/4}$: Es gibt eine Konstante $c > 0$ unabhängig von d, n, so dass

$$\mathbb{E}\tilde{R}(\tilde{\delta}^{(M)}) - \tilde{R}(\delta^*) \leq c\sqrt{\mathrm{VC}(\mathcal{C})} \cdot \|\delta^*\|_{1,\mathcal{C}}^{1/2} \cdot n^{-1/4}. \tag{6.45}$$

Bemerkung 6.47

- In [42] sind auch Resultate für andere Wahlen von ϕ verfügbar, z. B. für $\phi(z) = e^z$. Für $\phi(z) = e^z$ erhält man eine wesentlich schlechtere obere Schranke für $\mathbb{E}\tilde{R}(\tilde{\delta}^{(M)}) - \tilde{R}(\delta^*)$ mit $\log(n)^{-1/2}$ anstelle von $n^{-1/4}$. Mittels der Risikoübertragungsformel aus Lemma 6.42(ii) erhält man aus Gl. (6.45) jeweils ein entsprechendes Resultat für das Excess Bayes Risk $\mathbb{E}R(\hat{f}_{n,M}^{boost,\approx}) - R(f^*)$ von AdaBoost und LogitBoost.
- Je nach Komplexität von $\mathcal{C}$ wächst der Term $\mathrm{VC}(\mathcal{C})$, vgl. Lemma 6.44. Der Term $\|\delta^*\|_{1,\mathcal{C}}$ hingegen fällt mit der Komplexität von $\mathcal{C}$, denn je vielfältiger die Klasse $\mathcal{C}$, desto weniger Gewichte werden in der Darstellung von $\delta^* \in F$ benötigt. Geht man beispielsweise von $\mathcal{C} = \mathcal{C}_1(d)$ aus (damit ist $\mathrm{VC}(\mathcal{C}_1(d)) \leq 2\log_2(2d)$) und dass „pro Dimension" eine feste Anzahl von Elementen aus $\mathcal{C}_1(d)$ zur Darstellung von δ^* benötigt wird, ergibt sich die Abschätzung $\|\delta^*\|_{1,\mathcal{C}} \leq c_1 d$ mit einer Konstanten $c_1 > 0$ unabhängig von d. Dann folgt

$$\mathbb{E}\tilde{R}(\tilde{\delta}^{(M)}) - \tilde{R}(\delta^*) \leq 2cc_1 \cdot \sqrt{\log_2(2d)} d^{1/2} n^{-1/4}.$$

- Bedingungen (i),(ii),(iii) sind formale Bedingungen an die Approximationen $(\tilde{\beta}_m, \tilde{g}^{(m)})$ der exakten Minimierer $(\hat{\beta}_m, \hat{g}^{(m)})$ z. B. in der Konstruktion des AdaBoost- bzw. LogitBoost-Algorithmus. In der Praxis können sie nur schwer überprüft bzw. forciert werden, sondern man muss hoffen, dass sie während des Konstruktionsverfahrens eingehalten werden. Gefordert wird in (i), (iii), dass die mit dem gierigen Verfahren erzeugten Klassifikationsbäume möglichst gute Approximationen des Minimierers sind. (ii) fordert, dass die Gewichte $\tilde{\beta}_m$ über die Zeit nicht zu schnell und nicht zu langsam absinken.
- Satz 6.46 liefert mit $M = \|\delta^*\|_{1,\mathcal{C}}^{1/2} \cdot n^{3/4}$ eine grobe Anzahl an Iterationen, die zum Erhalten der angegebenen Konvergenzrate des Excess Bayes Risks benötigt werden. Abgesehen von dem unbekannten Faktor $\|\delta^*\|_{1,\mathcal{C}}^{1/2}$ gibt es auch keine Aussage, dass dieses M das optimale Risiko erzeugt (es liefert nur eine gute Konvergenzrate in n). In der Praxis muss M daher mittels Überwachung eines Validierungsfehlers gewählt werden.

Weitere Konvergenzresultate mit leicht abgewandelten Bedingungen für Gl. (6.44) finden sich in [4].

Um ein Stoppkriterium theoretisch zu formalisieren, kann anstelle der Anzahl M der Iterationen auch die angesammelte Summe der Gewichte $\|\tilde{\delta}^{(M)}\|_{1,\mathcal{C}}$ verwendet werden. Wir zeigen hier ein Resultat aus [8], bei welchem ein neuer Algorithmus $\hat{\delta}_{n,\lambda}^{boost}$ definiert wird, der durch Minimierung des empirischen Risikos

$$\tilde{R}_n(\delta) = \frac{1}{n}\sum_{i=1}^{n} \tilde{L}(Y_i, \delta(X_i))$$

bei gleichzeitiger Bestrafung von $\|\delta\|_{1,\mathcal{C}}$ (mit einem Hyperparameter $\lambda > 0$) entsteht:

Definition 6.48
Sei $\mathcal{C} \subset \{f : \mathcal{X} \to \mathcal{Y} \text{ messbar}\}$ so, dass $f \in \mathcal{C} \iff -f \in \mathcal{C}$. Weiter sei $B_m = [0, \infty)$, und für $\lambda > 0$:

$$F_\lambda := \{\delta \in F : \|\delta\|_{1,\mathcal{C}} \leq \lambda\}$$

Sei weiter

$$\hat{\delta}_{n,\lambda}^{boost} \in \arg\min_{\delta \in F_\lambda} \tilde{R}_n(\delta)$$

und $\hat{f}_{n,\lambda}^{boost} = \text{sign}(\hat{\delta}_{n,\lambda}^{boost})$. ♦

Bemerkung Der Algorithmus $\hat{\delta}_{n,\lambda}^{boost}$ kann auch als Lösung eines Optimierungsproblems der Form $\arg\min_{\delta \in F} \{\tilde{R}_n(\delta) + \hat{\alpha}(\lambda) \cdot \|\delta\|_{1,\mathcal{C}}\}$ mit geeignetem $\hat{\alpha}(\lambda) > 0$ (abhängig von den Trainingsdaten und λ) aufgefasst werden, vgl. Definition 2.32.

Die Berechnung einer *Näherung* von $\hat{\delta}_{n,\lambda}^{boost}$ in der Praxis ist mittels ähnlicher iterativer Algorithmen wie das oben vorgestellte *forward stagewise additive modeling* möglich, vgl. [8, Remark 3]. Hier gehen wir davon aus, dass

$$\hat{\delta}_{n,\lambda}^{boost} \text{ für geeignetes } \lambda, M \text{ eine sinnvolle Approximation}$$
$$\text{vom AdaBoost- oder LogitBoost-Algorithmus } \hat{\delta}_{n,M}^{boost,\approx} \text{ darstellt.}$$

Der Grund ist, dass auch bei Nutzung von $\hat{\delta}_{n,M}^{boost,\approx}$ die zugehörige Summe der Gewichte immer weiter wächst und durch Abbruch der Iteration M beschränkt wird. Entsprechend interpretieren wir die Resultate für $\hat{\delta}_{n,\lambda}^{boost}$ so, als wären sie prinzipiell auch für $\hat{\delta}_{n,M}^{boost,\approx}$ erreichbar.

In [8, Corollary 11, 12] wird für eine geeignete Wahl von λ das Excess Bayes Risk von $\hat{\delta}_{n,\lambda}^{boost}$ formuliert. Für eine Funktion $h : [0, 1] \to \mathbb{R}$ bezeichne $|h|_{BV}$ die Variation von h. Dann gilt der folgende Satz:

Satz 6.49 Sei $\mathcal{X} = [0,1]^d$, $\mathcal{C} = \mathcal{C}_1(d)$ und $V_d := \mathrm{VC}(\mathcal{C}_1(d)) \leq 2\log_2(2d)$. Es gelte die Modellannahme 6.45, wobei $\delta^*(x) = \sum_{j=1}^d h_j(x_j)$ mit Funktionen $h_j : [0,1] \to \mathbb{R}$ mit $\sum_{j=1}^d |h_j| \leq B \cdot d$, wobei $B > 0$ eine von d unabhängige Konstante ist. Definiere

$$Z(d) := \begin{cases} \sqrt{de^d \log(d)}, & \phi(z) = e^z, \\ \sqrt{d \log(d)}, & \phi(z) = \log(1+e^z). \end{cases}$$

Sei $\lambda_k, k \in \mathbb{N}$ eine Folge von Zahlen in $(1,\infty)$ mit $\sum_{k=1}^\infty \lambda_k^{-1} \leq 1$. Dann gibt es eine Funktion $\zeta : (1,\infty) \to [0,\infty)$ und eine von d, n unabhängige Konstante $c > 0$, so dass

$$\hat{k} := \arg\min_{k \in \mathbb{N}} \left\{ \tilde{R}_n(\hat{\delta}_{n,\lambda_k}) + \zeta(\lambda_k) \right\}$$

erfüllt:

$$\lim_{n\to\infty} \mathbb{P}\left(R(\hat{f}^{boost}_{n,\lambda_{\hat{k}}}) - R(f^*) \geq c \cdot Z(d) \cdot n^{-\frac{1}{4}\frac{V_d+2}{V_d+1}} \right) = 0$$

Gilt die *low-noise condition* mit $\alpha = 1$ (vgl. Definition 3.12), so kann $Z(d) \cdot n^{-\frac{1}{4}\frac{V_d+2}{V_d+1}}$ durch $\left[Z(d) \cdot n^{-\frac{1}{4}\frac{V_d+2}{V_d+1}} \right]^2$ ersetzt werden.

Bemerkung 6.50

- Wäre jedes $h_j : [0,1] \to \mathbb{R}$ Lipschitz-stetig mit Lipschitz-Konstante B, so gilt $\sum_{j=1}^d |h_j| \leq B \cdot d$. Diese Annahme bedeutet also, dass die Funktionswerte von δ^* nicht zu stark variieren. Wir können die Forderung der Lipschitz-Stetigkeit jedoch nicht direkt stellen, da Elemente von F als Linearkombinationen von unstetigen Bäumen nicht stetig sind.
- Im Satz wird eine konkrete Wahl von $\lambda = \lambda_{\hat{k}}$ vorgegeben, diese ist in der Praxis aufgrund der Abhängigkeit der Funktion ζ von unbekannten Konstanten (vgl. [8, Theorem 1] für das genaue Aussehen) nicht verwendbar. In diesem Sinne ist die Aussage eher so zu interpretieren, dass es eine geeignete Wahl von λ gibt, welche die dargestellte Rate des Excess Bayes Risk ermöglicht.
- Die Rate $n^{-\frac{1}{4}\frac{V_d+2}{V_d+1}}$ ist weitestgehend unabhängig von d, es gilt zum Beispiel immer $n^{-\frac{1}{4}\frac{V_d+2}{V_d+1}} \geq n^{-1/2}$. In diesem Sinne ist vor allem der Vorfaktor $Z(d)$ interessant, um das Zusammenspiel zwischen Dimension d und Anzahl der Trainingsdaten n zu untersuchen. Wie bereits in Satz 6.46 erhalten wir ein wesentlich besseres Verhalten für $\phi(z) = \log(1+e^z)$. Solange $d \ll n$, konvergiert das Excess Bayes Risk von $\hat{f}^{boost}_{n,\lambda_{\hat{k}}}$ gegen null.

6.6 Random Forests

6.6.1 Motivation

Random Forests stellen eine Erweiterung des Bagging-Algorithmus dar und sollen seine Leistungsfähigkeit insbesondere bei hochdimensionalen Trainingsdaten verbessern. Wir motivieren *Random Forests* im Kontext von Klassifikationsproblemen mit $K = 2$ Klassen, das Verfahren ist aber auch für eine beliebige Anzahl von Klassen sowie Regressionsprobleme anwendbar. Zunächst analysieren wir den *majority vote*-Ansatz beim Bagging genauer:

Ist $x_0 \in \mathcal{X}$ fest gewählt, so lautet der Bagging-Algorithmus für Klassifikation aus Definition 6.26:

$$\hat{f}_n^{bagg}(x_0) = \underset{k\in\{1,2\}}{\arg\max} \#\{b \in \{1,\dots,B\} : \hat{f}_n^{*b}(x_0) = k\}, \tag{6.46}$$

wobei $b = 1,\dots,B$ Klassifikationsbäume basierend auf Bootstrap-Stichproben von (X_i, Y_i), $i = 1,\dots,n$ sind. Insbesondere gilt:

$$\hat{f}_n^{bagg}(x_0) = 1 \quad \Longleftrightarrow \quad \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{\hat{f}_n^{*b}(x_0)=1\}} > \frac{1}{2}$$

Anschaulich bedeutet dies: Man entscheidet sich bei x_0 für Klasse 1, wenn sich mehr als die Hälfte der $\hat{f}_n^{*b}(x_0)$ für Klasse 1 entscheiden.

Das im Folgenden zitierte *Lemma von der Schwarmintelligenz* trifft eine Aussage über die Qualität von $\hat{f}_n^{bagg}(x_0)$ für $B \to \infty$ unter Annahmen an $f_n^{*b}(x_0)$. Für die kompakte Formulierung benötigen wir die folgenden Größen:

$$a_n := \mathbb{P}(\hat{f}_n^{*b}(x_0) = 1), \qquad \sigma_n^2 := \mathrm{Var}(\mathbb{1}_{\{\hat{f}_n^{*b}(x_0)=1\}})$$

Lemma 6.51 (Schwarmintelligenz) Es gelte $f^*(x_0) = 1$. Weiterhin gelten folgende Aussagen:

(A1) Es gilt $a_n > \frac{1}{2}$. Der Klassifizierer ist also zumindest so gut, dass er sich mit höherer Wahrscheinlichkeit für die korrekte Klasse entscheidet.
(A2) Die $\hat{f}_n^{*b}(x_0)$, $b = 1,\dots,B$ sind unabhängig.

Dann gilt:

$$\mathbb{P}(\hat{f}_n^{bagg}(x_0) = 1) \to 1 \qquad (B \to \infty)$$

Beweis Die $\hat{f}_n^{*b}(x_0)$, $b = 1,\dots,B$ sind identisch verteilt (sie entstehen durch dieselbe Konstruktion aus den Trainingsdaten). Weiter ist

$$\mathbb{E}\left[\frac{1}{B}\sum_{b=1}^{B}\mathbb{1}_{\{\hat{f}_n^{*b}(x_0)=1\}}\right] = \mathbb{E}[\mathbb{1}_{\{\hat{f}_n^{*b}(x_0)=1\}}] = a_n > \frac{1}{2}$$

und

$$\mathrm{Var}\left[\frac{1}{B}\sum_{b=1}^{B}\mathbb{1}_{\{\hat{f}_n^{*b}(x_0)=1\}}\right] = \frac{\mathrm{Var}(\mathbb{1}_{\{\hat{f}_n^{*b}(x_0)=1\}})}{B} = \frac{\sigma_n^2}{B}. \tag{6.47}$$

Mit der Tschebyscheff-Ungleichung folgt:

$$\begin{aligned}\mathbb{P}(\hat{f}_n^{bagg}(x_0)=2) &= \mathbb{P}\left(\frac{1}{B}\sum_{b=1}^{B}\mathbb{1}_{\{\hat{f}_n^{*b}(x_0)=1\}} - a_n < \frac{1}{2} - a_n\right)\\ &\le \mathbb{P}\left(\left|\frac{1}{B}\sum_{b=1}^{B}\mathbb{1}_{\{\hat{f}_n^{*b}(x_0)=1\}} - a_n\right| > a_n - \frac{1}{2}\right)\\ &\le \frac{\sigma_n^2}{B(a_n-\frac{1}{2})^2} \to 0 \qquad (B\to\infty)\end{aligned} \tag{6.48}$$

Das bedeutet: Falls die Annahmen (A1), (A2) zutreffen und genügend Bootstrap-Stichproben ausgewählt werden (d. h. B groß genug), so ist die Wahrscheinlichkeit nahe bei 1, dass sich $\hat{f}_n^{bagg}(x_0)$ korrekt entscheidet. Dieses Phänomen wird in der Literatur als Schwarmintelligenz (engl. *wisdom of crowds*) bezeichnet. Die Annahmen (A1) und (A2) fordern, dass die einzelnen Entscheidungen *unabhängig* voneinander jeweils mit Wahrscheinlichkeit $> \frac{1}{2}$ korrekt getroffen werden. Damit findet eine fortwährende Akkumulation von *verschiedenem* (da unabhängigem) *Wissen* (da mit Wahrscheinlichkeit $> \frac{1}{2}$ korrekte Entscheidung) über die Wahrheit statt, die im Endergebnis zur richtigen Entscheidung führt.

Leider treffen die Annahmen (A1), (A2) beim Bagging-Algorithmus nicht zu: Die Bootstrap-Stichproben $(X_1^{*b}, Y_1^{*b}), \ldots, (X_n^{*b}, Y_n^{*b}), b = 1, \ldots, B$ werden zwar unabhängig aus den Trainingsdaten $(X_1, Y_1), \ldots, (X_n, Y_n)$ gezogen, aber von außen betrachtet (ohne Kenntnis der Trainingdaten) bestehen sie natürlich alle immer aus fast denselben Beobachtungen und sind daher hochgradig abhängig. Entsprechend sind auch die darauf basierenden Klassifizierer $\hat{f}_n^{*b}$ und deren Entscheidungen $\hat{f}_n^{*b}(x_0)$ stochastisch abhängig für $b = 1, \ldots, B$. Bei Anwendung des Bagging-Algorithmus ist also die Annahme (A2) verletzt. Tatsächlich gilt:

(A2') Die Korrelationen $\rho(\hat{f}_n^{*b}(x_0), \hat{f}_n^{*b'}(x_0)) = \rho_n \neq 0$ sind nicht null für alle $b, b' = 1, \ldots, B$.

Dann folgt aber statt Gl. (6.47):

$$\mathrm{Var}\left[\frac{1}{B}\sum_{b=1}^{B}\mathbb{1}_{\{\hat{f}_n^{*b}(x_0)=1\}}\right] = \rho_n \cdot \sigma_n^2 + \frac{1-\rho_n}{B}\cdot\sigma_n^2$$

Da dieser Ausdruck für $B \to \infty$ nicht mehr gegen null geht, gilt das Lemma 6.51 nicht mehr (Beweisschritt Gl. (6.48) funktioniert nicht mehr).

Wollen wir also bessere Ergebnisse mit dem Bagging-Algorithmus aus Gl. (6.46) erhalten, so müssen wir (A2) näherungsweise wiederherstellen. Dies gelingt, indem wir die Konstruktion der Klassifikationsbäume $\hat{f}_n^{*b}$ so modifizieren, dass die Korrelation ρ_n der $\hat{f}_n^{*b}(x_0)$ untereinander verringert wird, ohne dass die Varianz σ_n wesentlich ansteigt.

Konkret wird bei *Random Forests* die geringere Abhängigkeit durch die folgenden beiden Modifikationen erreicht:

(a) Anstatt das normale gierige Verfahren zum Erstellen eines CARTs $\hat{f}_n^{*b}$ zu verwenden, ändern wir den Algorithmus ab. In jedem Schritt darf nicht jede Koordinate $j \in \{1, \ldots, d\}$ für einen Split ausgewählt werden, sondern nur bestimmte Koordinaten $j \in S$, wobei $S \subset \{1, \ldots, d\}$ eine zufällig bestimmte Teilmenge mit $\#S = m < d$ ist. In diesem Sinne ist jeder Baum $\hat{f}_n^{*b}$ gezwungen, nur einen Teil des Wissens über f^* aus den Trainingsdaten zu entnehmen.

(b) Verringere die Anzahl der Elemente einer Bootstrap-Stichprobe auf $A_n < n$, und konstruiere die CARTs $\hat{f}_n^{*b}$ entsprechend auf Basis von nur A_n Beobachtungen. Ist beispielsweise $A_n = \frac{n}{2}$, so können zwei verschiedene Bootstrap-Stichproben aus zwei völlig verschiedenen Teilmengen der Trainingsdaten bestehen; damit sind dann auch die Entscheidungen der aus den Daten konstruierten Bäume unabhängig. In der Statistik spricht man im Falle $A_n < n$ von *Subsampling* anstelle von *Bootstrap*.

6.6.2 Formale Definition

Die Modifikation (a) wird wie folgt umgesetzt:

Definition 6.52 (Gieriger Algorithmus CART für Random Forests)
Sei $0 < m < d, 0 < t \leq n$. Verwende den Algorithmus aus 6.15 bzw. Definition 6.16, aber füge folgenden Schritt ein:

(2.5) Wähle zufällig eine Teilmenge $S \subset \{1, \ldots, d\}$ mit $\#S = m$ aus

und führe die Minimierung in Schritt (3) nur über $j \in S$ durch.
Führe das Verfahren nur durch, bis der Baum insgesamt t Blätter hat. Das Ergebnis bezeichnen wir mit $\hat{T}_{n,m,t}^g = \hat{T}_{n,m,t}^g(X_1, Y_1, \ldots, X_n, Y_n)$. ♦

Die Modifikation (b) wird direkt in die Konstruktion des Algorithmus eingearbeitet:

Definition 6.53 (Random Forest)
Seien (X_i, Y_i), $i = 1, \ldots, n$ Trainingsdaten für ein Regressions- oder Klassifikationsproblem.

- $0 < m < d$ (Anzahl Koordinaten zur Auswahl),
- $0 < A_n \leq n$ (Größe der Bootstrap-Stichproben),
- $0 < t_n \leq A_n$ (Anzahl der Blätter).

Sei $B \in \mathbb{N}$, $x \in \mathcal{X}$. Für $b = 1, \ldots, B$:

- Ziehe zufällig gleichverteilt, mit (falls $A_n < n$: ohne) Zurücklegen $(X_1^{*b}, Y_1^{*b}), \ldots, (X_{A_n}^{*b}, Y_{A_n}^{*b})$ aus $((X_1, Y_1), \ldots, (X_n, Y_n))$.
- Sei $\hat{f}_n^{*b}(x) := f_T(x)$, wobei $T = \hat{T}^g_{A_n,m,t_n}(x, X_1^{*b}, Y_1^{*b} \ldots, X_{A_n}^{*b}, Y_{A_n}^{*b})$ (CART für Regression bzw. Klassifikation).

Der *Random-Forest-Algorithmus* für Regression lautet

$$\hat{f}^{RF}_{n,m,A_n,t_n}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_n^{*b}(x).$$

Der *Random-Forest-Algorithmus* für Klassifikation lautet

$$\hat{f}^{RF}_{n,m,A_n,t_n}(x) = \underset{k \in \{1,\ldots,K\}}{\arg\max} \ \#\{b \in \{1, \ldots, B\} : \hat{f}_n^{*b}(x) = k\}. \qquad \blacklozenge$$

Der Name „Random Forest" leitet sich daraus her, dass sehr viele Bäume basierend auf zufällig gezogenen Bootstrap-Stichproben erzeugt werden müssen, um eine Entscheidungsregel zu erhalten.

In der Praxis (zum Beispiel im R-Paket *randomForest,* vgl. [21]) wählt man bei Klassifikation beispielsweise $1 \leq m \leq \sqrt{d}$, bevorzugt $m = \lfloor \sqrt{d} \rfloor$, und bei Regression $m = \lfloor \frac{d}{3} \rfloor$. Dies sind aber lediglich heuristisch motivierte Werte, eine theoretische Legitimation dafür gibt es (noch) nicht.

Das folgende Resultat (aus [26, Theorem 1]) liefert zunächst für niedrige Dimensionen d der Trainingsdaten unter einer Strukturannahme (sog. additives Modell) eine Aussage über das Excess Bayes Risk von $\hat{f}^{RF}_{n,m,A_n,t_n}$:

Satz 6.54 (Random Forests) Es seien (X_i, Y_i), $i = 1, \ldots, n$ Trainingsdaten eines Regressionsproblems und $L(y, s) = (y - s)^2$ die quadratische Verlustfunktion. Die Bayes-Regel erfülle

$$f^*(x) = \sum_{j=1}^{d} h_j(x_j)$$

mit $h_j : \mathbb{R} \to \mathbb{R}$ stetig. Weiter sei X gleichverteilt auf $\mathcal{X} = [0,1]^d$, und es gelte $\varepsilon := Y - f^*(X) \sim N(0, \sigma^2)$ mit einer Konstante $\sigma > 0$. Sei $d \in \mathbb{N}$ fest vorgegeben. Falls $A_n \to \infty$, $t_n \to \infty$ und $\frac{t_n (\log A_n)^9}{A_n} \to 0$, so gilt:

$$\mathbb{E} R(\hat{f}^{RF}_{n,m,A_n,t_n}) - R(f^*) \to 0 \quad (n \to \infty)$$

Bemerkung Die wesentliche Bedingung ist $\frac{t_n (\log A_n)^9}{A_n} \to 0$, d. h., die Anzahl der Blätter muss geringer sein als die Anzahl A_n der in einer Bootstrap-Stichprobe vorhandenen Beobachtungen.

Auch für hohe Dimensionen d liefern Random Forests oft Entscheidungsregeln mit erstaunlich guten Risiken. Mittlerweile sind auch theoretische Resultate für den hochdimensionalen Kontext verfügbar, vgl. zum Beispiel [37, Theorem 1].

6.7 Übungen

1. Zeigen Sie, dass $y_m = \frac{1}{n} \sum_{X_i \in A_m} Y_i$, $m = 1, \ldots, \#T$ den Ausdruck Gl. (6.6) und $y_m = \arg\max_{k \in \{1,\ldots,K\}} \#\{i : X_i \in A_m, Y_i = k\}$, $m = 1, \ldots, \#T$ den Ausdruck Gl. (6.7) minimiert.
2. Zeigen Sie: Für $(p_1, \ldots, p_K) \in [0,1]^K$ mit $\sum_{k=1}^K p_k = 1$ ist $G(p_1, \ldots, p_K) = \sum_{k=1}^K p_k \cdot (1 - p_k)$ maximal, falls $p_1 = \cdots = p_K = \frac{1}{K}$, und minimal, falls ein $i \in \{1, \ldots, K\}$ existiert mit $p_i = 1$.
3. Implementieren Sie die folgenden Algorithmen und wenden Sie diese auf Trainingsdaten aus Beispiel 6.3 an:
   - das gierige Verfahren zur Ermittlung von Klassifikations/Regressionsbäumen $\hat{T}^g_{n,\lambda}$ aus Definition 6.15 bzw. 6.16, den Algorithmus $\hat{T}^{g,p}_{n,\lambda}$ aus Definition 6.18 sowie die Wahl von λ mittels Cross Validation,
   - den Bagging-Schätzer $\hat{f}^{bagg}_n$ basierend auf Klassifikations- oder Regressionsbäumen (vgl. Definition 6.25 bzw. 6.26),
   - für Regression den Boosting-Algorithmus aus Definition 6.35 und für Klassifikation den AdaBoost- bzw. LogitBoost-Algorithmus aus Definition 6.39 bzw. 6.41.

Neuronale Netzwerke

7

Inhaltsverzeichnis

Neuronale Netzwerke bilden eine flexible Funktionenklasse zur Approximation von stetigen Funktionen. Der Aufbau und die Funktionsauswertung eines neuronalen Netzwerks ist anschaulich der Signalübertragung von Nervenzellen nachempfunden, was den Namen erklärt. Wie Bäume erlauben neuronale Netzwerke sowohl die Behandlung von Regressions- als auch Klassifikationsproblemen.

Die statistische Theorie für die tatsächlich in der Praxis genutzten neuronalen Netzwerke steht noch am Anfang. Wir motivieren zunächst eine allgemeine Definition von neuronalen Netzwerken, das zugehörige Optimierungsproblem und Vorgehensweisen zur näherungsweisen Lösung. Im zweiten Teil nutzen wir die Theorie aus [25] zum Aufzeigen aktueller statistische Resultate.

7.1 Motivation und Definition

In diesem Kapitel sei stets $\mathcal{X} \subset \mathbb{R}^d$. Bei vielen Anwendungsproblemen ist die Dimension d der Beobachtungen (auch: Featurevektor) $X_i \in \mathbb{R}^d$ sehr hoch. Damit Algorithmen des maschinellen Lernens nach annehmbarer Rechenzeit Resultate liefern, werden die Daten X_i oft mittels „Expertenwissen“ vorbearbeitet und zu einem neuen Vektor $\tilde{X}_i \in \mathbb{R}^{\tilde{d}}$ mit geringerer Dimension $\tilde{d} < d$ reduziert. Einige typische Beispiele aus der Klassifizierung sind:

S. Richter, *Statistisches und maschinelles Lernen*,
https://doi.org/10.1007/978-3-662-59354-7_7

- *Musikerkennung:* Nicht die Rohdaten des Musikstücks (Größe z. B. 3 Megabyte, d. h. $d \approx 3 \cdot 10^6$) werden als X_i genutzt, sondern z. B. werden Charakteristiken und Eigenschaften des Frequenzspektrums in einem Vektor $\tilde{X}_i$ mit $\tilde{d} \approx 1000$ zusammengetragen.
- *Bilderkennung:* Nicht die Rohdaten des vollständigen Bildes (z. B. 1000×1000 Pixel, d. h. $d \approx 10^6$) werden als X_i genutzt, sondern zum Beispiel eine komprimierte Version. Alternativ werden für jedes einzelne Pixel Eigenschaften in einem Vektor $\tilde{X}_i$ zusammengetragen (Farbe, Schärfe, Farbe der Nachbarn etc.), und die Berechnung wird pixelweise ausgeführt mit $\tilde{d} \approx 1000$.

Angewandt auf $(\tilde{X}_i, Y_i)_{i=1,\dots,n}$ liefern auch SVMs und Bäume gute Resultate. Dabei wird jedoch implizit vorausgesetzt, dass die Reduktion $\tilde{X}_i$ alle wesentlichen Eigenschaften der Daten X_i zusammengefasst hat, kaum Informationsverlust aufgetreten ist und die Beziehung zwischen Y_i und $\tilde{X}_i$ durch die Komprimierung nicht wesentlich verkompliziert wurde.

Im Gegensatz dazu ermöglichen neuronale Netzwerke die direkte Verarbeitung hochdimensionaler Trainingsdaten. Auch hier möchte man den Informationsgehalt der ursprünglichen Beobachtungen $X_i \in \mathbb{R}^d$ auf niedrigdimensionalere $\tilde{X}_i \in \mathbb{R}^{\tilde{d}}$ komprimieren. Die Reduktion erfolgt hier allerdings auf Basis der Trainingsdaten und nicht unter Nutzung von ‚Expertenwissen'. Auf die reduzierten Daten $(\tilde{X}_i, Y_i)_{i=1,\dots,n}$ wird anschließend eine einfache lineare Regression (bei Regressionsproblemen) bzw. eine logistische Regression (bei Klassifikationsproblemen) angewandt. Die Reduktion von X_i zu $\tilde{X}_i$ entsteht durch $L \in \mathbb{N}$ Schritte gleicher Struktur: In jedem Schritt werden die aktuell vorliegenden Vektoren mittels einer einfachen Transformation modifiziert und gleichzeitig die Dimension der Daten reduziert (oder auch erhöht).

Auf den ersten Blick scheinen lineare Abbildungen eine gute Wahl für diese einfachen Transformationen zu sein. Dann sähen das Vorgehen und die damit verbundene Modellannahme wie folgt aus:

Bemerkung 7.1 (Naiver Ansatz zur Dimensionsreduktion) Seien $p_1, \dots, p_L \in \mathbb{N}$. Für $i = 1, \dots, n$:

1. $X_i^{(1)} := v^{(1)} + W^{(1)} X_i \in \mathbb{R}^{p_1}$ mit geeigneten $v^{(1)} \in \mathbb{R}^{p_1}$, $W^{(1)} \in \mathbb{R}^{p_1 \times d}$.
2. $X_i^{(2)} := v^{(2)} + W^{(2)} X_i^{(1)} \in \mathbb{R}^{p_2}$ mit geeigneten $v^{(2)} \in \mathbb{R}^{p_2}$, $W^{(2)} \in \mathbb{R}^{p_2 \times p_1}$.

■ …

(L) $X_i^{(L)} := v^{(L)} + W^{(L)} X_i^{(L-1)} \in \mathbb{R}^{p_L}$ mit geeigneten $v^{(L)} \in \mathbb{R}^{p_L}$, $W^{(L)} \in \mathbb{R}^{p_L \times p_{L-1}}$.

Annahme: $\left(X_i^{(L)}, Y_i\right)_{i=1,\dots,n}$ folgt dem Modell einer linearen Regression (vgl. Modellannahme 2.1) bzw. einer logistischen Regression (vgl. Modellannahme 4.8).

Durch das Vorgehen in Bemerkung 7.1 erhalten wir jedoch

$$X_i^{(L)} = \tilde{v} + \tilde{W} \cdot X_i \tag{7.1}$$

mit geeigneten $\tilde{v} \in \mathbb{R}^{p_L}$, $\tilde{W} \in \mathbb{R}^{p_L \times d}$. Das bedeutet, das gesamte Modell würde nur einer linearen Regression bzw. einer logistischen Regression von $(X_i, Y_i)_{i=1,\dots,n}$ entsprechen. Komplizierte Zusammenhänge zwischen X_i und Y_i könnten damit nicht beschrieben werden.

Die Darstellung Gl. (7.1) ist möglich, weil Kompositionen linearer Abbildungen wieder lineare Abbildungen sind. Wenn wir jedoch zwischen den einzelnen Schritten in Bemerkung 7.1 eine ‚Störung' der Linearität einbauen, ist diese Schlussfolgerung nicht mehr anwendbar. Eine möglichst einfache Störung kann wie folgt erreicht werden: Sei $\sigma : \mathbb{R} \to \mathbb{R}$ eine nichtlineare Funktion. Ist $x \in \mathbb{R}^d$, so schreiben wir $\sigma(x)$ für die komponentenweise Anwendung $\sigma(x) := (\sigma(x_1), \dots, \sigma(x_d))$. Das Vorgehen ändert sich im Vergleich zu Bemerkung 7.1 wie folgt:

Bemerkung 7.2 (Nichtlinearer Ansatz zur Dimensionsreduktion) Seien $p_1, \dots, p_L \in \mathbb{N}$. Für $i = 1, \dots, n$:

(1) $Z_i^{(1)} := v^{(1)} + W^{(1)} X_i \in \mathbb{R}^{p_1}$ mit geeigneten $v^{(1)} \in \mathbb{R}^{p_1}$, $W^{(1)} \in \mathbb{R}^{p_1 \times d}$.
(1') $X_i^{(1)} := \sigma\left(Z_i^{(1)}\right)$.
(2) $Z_i^{(2)} := v^{(2)} + W^{(2)} X_i^{(1)} \in \mathbb{R}^{p_2}$ mit geeigneten $v^{(2)} \in \mathbb{R}^{p_2}$, $W^{(2)} \in \mathbb{R}^{p_2 \times p_1}$.
(2') $X_i^{(2)} := \sigma\left(Z_i^{(2)}\right)$.

■ …

(L) $Z_i^{(L)} := v^{(L)} + W^{(L)} X_i^{(L-1)} \in \mathbb{R}^{p_L}$ mit geeigneten $v^{(L)} \in \mathbb{R}^{p_L}$, $W^{(L)} \in \mathbb{R}^{p_L \times p_{L-1}}$.
(L') $X_i^{(L)} := \sigma\left(Z_i^{(L)}\right)$.

Annahme: $(X_i^{(L)}, Y_i)_{i=1,\dots,n}$ folgt dem Modell einer linearen Regression (vgl. Modellannahme 2.1) bzw. einer logistischen Regression (vgl. Modellannahme 4.8).

Fassen wir die Schritte (1),(1'),…,(L),(L') aus Bemerkung 7.2 zusammen, so ergibt sich $X_i^{(L)} = \tilde{g}(X_i)$, $i = 1, \dots, n$ mit

$$\tilde{g}(x) = \sigma\left(v^{(L)} + W^{(L)} \cdot \sigma\left(\dots v^{(2)} + W^{(2)} \cdot \sigma\left(v^{(1)} + W^{(1)} \cdot x\right) \dots\right)\right).$$

Wir betrachten nun getrennt die beiden Fälle der linearen und logistischen Regression und leiten jeweils die komplette Modellannahme her:

- Regression: Folgt $(X_i^{(L)}, Y_i)_{i=1,\dots,n}$ dem Modell einer linearen Regression, so gilt mit geeignetem $W^{(L+1)} \in \mathbb{R}^{1\times p_L}$ und i. i. d. ε_i mit $\mathbb{E}\varepsilon_i = 0$:

$$Y_i = W^{(L+1)} \cdot X_i^{(L)} + \varepsilon_i, \quad i = 1, \dots, n,$$

und die Bayes-Regel erfüllt:

$$f^*(x) = \mathbb{E}[Y_1 | X_1 = x] = W^{(L+1)} \cdot \tilde{g}(x)$$

- Klassifikation: Folgt $\left(X_i^{(L)}, Y_i\right)_{i=1,\dots,n}$ dem Modell einer logistischen Regression mit K Klassen, so gilt mit geeignetem $W^{(L+1)} = \left(\left(W_1^{(L+1)}\right)^T, \dots, \left(W_K^{(L+1)}\right)^T\right)^T \in \mathbb{R}^{K\times p_L}$ (vgl. Lemma 4.9):

$$\begin{aligned}\mathbb{P}(Y_1 = k | X_1 = x) &= \mathbb{P}\left(Y_1 = k | X_1^{(L)} = \tilde{g}(x)\right) \\ &= \frac{e^{W_k^{(L+1)}\tilde{g}(x)}}{1 + e^{W_k^{(L+1)}\tilde{g}(x)}} = M_k\left(W^{(L+1)} \cdot \tilde{g}(x)\right), \quad k = 1, \dots, K,\end{aligned}$$

mit $M : \mathbb{R}^K \to \mathbb{R}^K$ definiert durch

$$M(x) = \begin{pmatrix} M_1(x) \\ \vdots \\ M_K(x) \end{pmatrix} = \frac{1}{\sum_{l=1}^K e^{x_l}} \begin{pmatrix} e^{x_1} \\ \vdots \\ e^{x_K} \end{pmatrix}, \qquad x = (x_1, \dots, x_K).$$

Die Bayes-Regel muss dann erfüllen:

$$f^*(x) = \underset{k\in\{1,\dots,K\}}{\arg\max}\ \mathbb{P}(Y_1 = k | X_1 = x) = \underset{k\in\{1,\dots,K\}}{\arg\max}\ M_k\left(W^{(L+1)} \cdot \tilde{g}(x)\right)$$

Wie durch Vergleich mit Lemma 4.9 ersichtlich, wird das Problem durch die Verwendung von $W_K^{(L+1)}$ überparametrisiert. Dies wird hier ignoriert, um stattdessen einfachere Darstellungen zu erhalten.

Schreiben wir kurz $g(x) = W^{(L+1)} \cdot \tilde{g}(x)$, so können wir einen Großteil der Struktur für beide Fälle zusammenfassen:

Definition 7.3
Sei $L \in \mathbb{N}_0$, $p = (p_0, \dots, p_{L+1})^T \in \mathbb{N}^{L+2}$.

Ein *neuronales Netzwerk* mit *Netzwerkarchitektur* (L, p) und *Aktivierungsfunktion* $\sigma : \mathbb{R} \to \mathbb{R}$ ist eine Funktion $g : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}}$ mit

$$g(x) = W^{(L+1)} \cdot \sigma\left(v^{(L)} + W^{(L)} \cdot \sigma\left(\dots v^{(2)} + W^{(2)} \cdot \sigma\left(v^{(1)} + W^{(1)} \cdot x\right)\dots\right)\right), \quad (7.2)$$

wobei $W^{(l)} \in \mathbb{R}^{p_l \times p_{l-1}}, l = 1, \ldots, L+1$ *Gewichtsmatrizen* und $v^{(l)} \in \mathbb{R}^{p_l}$ *Verschiebungsvektoren* heißen. Wir nennen L die Anzahl der *hidden layer*/*Tiefe* des neuronalen Netzwerks und p den *width vector*.

Wir schreiben $\mathcal{F}(L, p)$ für die Menge aller solcher Funktionen g. ♦

Erst bei der Interpretation eines neuronalen Netzwerks ergeben sich Unterschiede für Regressions- und Klassifikationsprobleme:

Definition 7.4
Sei $g \in \mathcal{F}(L, p)$ mit $p_0 = d$.

- g heißt *neuronales Netzwerk für Regression*, falls $p_{L+1} = 1$. Die zu g gehörige Entscheidungsfunktion ist $f = g$.
- g heißt *neuronales Netzwerk für Klassifikation* mit K Klassen, falls $p_{L+1} = K$. Die zu g gehörige Entscheidungsregel ist $f(x) = \arg\max_{k \in \{1,\ldots,K\}} M_k(g(x))$. ♦

Bemerkungen
- In der Praxis verwendet man häufig die sogenannte *ReLU-Funktion* $\sigma(x) = \max\{x, 0\}$ als Aktivierungsfunktion (Abkürzung des englischen Begriffs *rectified linear unit*). Besonders für neuronale Netzwerke mit nicht zu großem L wird auch die *Sigmoidfunktion* $\sigma(x) = \frac{e^x}{1+e^x}$ verwendet. Für große L ist die Sigmoidfunktion in der Praxis nicht geeignet, vgl. die Bemerkung nach Algorithmus 7.19.
- Da keine Einschränkungen an die Gewichtmatrizen $W^{(l)}$ vorgenommen werden, nennt man $g \in \mathcal{F}(L, p)$ auch *fully connected neural network*, vgl. Abb. 7.1 und 7.2.
- Da M eine differenzierbare Approximation des Maximums $\max(x) = \max(x_1, \ldots, x_K)$ für ‚große' x ist, nennt man M auch *Softmax*-Funktion.

7.2 Anschauliche Darstellung neuronaler Netzwerke

Jedes neuronale Netzwerk $g \in \mathcal{F}(L, p)$ besteht aus Funktionen, bei welchen der Eingangswert $x \in \mathbb{R}^d$ L-mal mit einer Gewichtsmatrix W multipliziert und mit einer Aktivierungsfunktion σ verändert wird, vgl. die Motivation in Bemerkung 7.2. In jedem Schritt wird der ursprüngliche Eingabewert $x^{(0)} = x \in \mathbb{R}^d$ aktualisiert mittels

$$x^{(l)} = \sigma\left(v^{(l)} + W^{(l)} \cdot x^{(l-1)}\right), \qquad l = 1, 2, \ldots, L.$$

Im letzten Schritt erhält man den Ausgabewert $y = g(x) = W^{(L+1)} x^{(L)}$.

Die Gewichtsmatrizen geben an, welche Komponenten von $x^{(l-1)}$ den stärksten Einfluss auf die einzelnen Komponenten des nächsten Werts $x^{(l)}$ haben. Daher stellt man die Komponenten der Vektoren $x^{(l)}$, $l = 0, \ldots, L$, als Kreise/Knoten dar, welche entsprechend der Einträge der Gewichtsmatrizen $W^{(l)}$ stark oder schwach verbunden sind. Diese Interpretation liefert auch den Namen der neuronalen Netzwerke: Jeder ‚Kreis'/‚Knoten' entspricht anschaulich einer Nervenzelle (Neuron), die Eingangsinformationen verarbeitet und weitergibt.

Die Aktivierungsfunktion σ und die Verschiebungsvektoren $v^{(l)}$ werden in der Darstellung unterdrückt, man kann nur die Größen L und p ablesen. Abb. 7.1 zeigt eine Visualisierung eines neuronalen Netzwerks für Regression mit $L = 2$ und $p = (1, 3, 3, 1)^T$ zur Modellierung von $f : \mathbb{R} \to \mathbb{R}$. Die konkrete Darstellung von f ist im folgenden Beispiel gegeben:

Beispiel 7.5 (Neuronales Netzwerk für Regression) Wir betrachten den Spezialfall $d = 1$. Zur Modellierung von $f : \mathbb{R} \to \mathbb{R}$ nutzen wir ein neuronales Netzwerk $f = g \in \mathcal{F}(L, p)$ mit $L = 2$ und $p = (1, 3, 3, 1)^T$ (vgl. Abb. 7.1). Dann gilt

$$\begin{aligned} f(x) &= W^{(3)} \cdot \sigma\left(v^{(2)} + W^{(2)} \cdot \sigma(v^{(1)} + W^{(1)} \cdot x)\right) \\ &= \sum_{j=1}^{3} W_{1j}^{(3)} \sigma\left(v_j^{(2)} + \sum_{k=1}^{3} W_{jk}^{(2)} \sigma\left(v_k^{(1)} + \sum_{l=1}^{3} W_{kl}^{(1)} x_l\right)\right) \end{aligned}$$

mit Matrizen $W^{(1)} \in \mathbb{R}^{3\times 1}$, $W^{(2)} \in \mathbb{R}^{3\times 3}$, $W^{(3)} \in \mathbb{R}^{1\times 3}$ und Vektoren $v^{(1)}, v^{(2)} \in \mathbb{R}^3$; insgesamt gibt es 21 Parameter, um die resultierende Funktion f zu beschreiben.

Abb. 7.2 zeige ein neuronales Netzwerk für Klassifikation mit $L = 2$ und $p = (2, 3, 3, 2)^T$ zur Modellierung von $f : \mathbb{R}^2 \to \{1, 2\}$.

Die Menge der Funktionen, welche mit einem neuronalen Netzwerk beschrieben werden können, wächst mit der Anzahl der *hidden layer* $L \in \mathbb{N}$ und deren Dimension. In Abb. 7.3

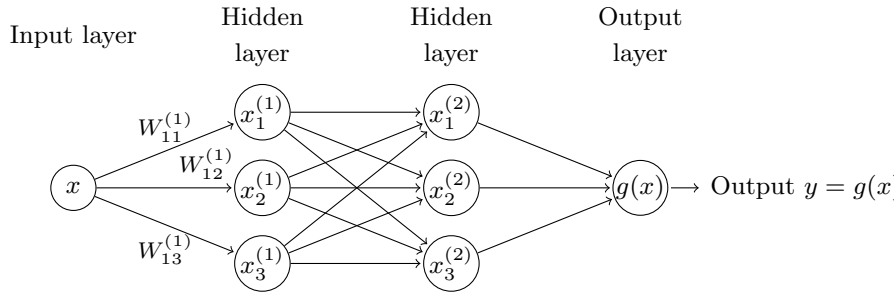


Abb. 7.1 Neuronales Netzwerk für Regression mit zwei *hidden layer*

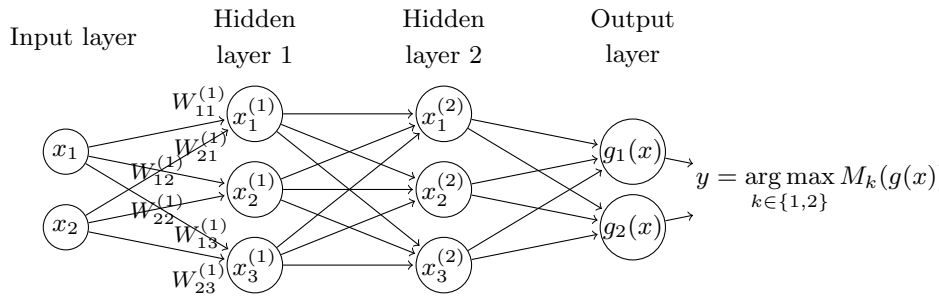


Abb. 7.2 Neuronales Netzwerk für Klassifikation mit $L = 2$ *hidden layer* und $K = 2$ Klassen

sind beispielhaft einige Funktionen $f : \mathbb{R} \to \mathbb{R}$ mit $f \in \mathcal{F}(L, p)$ für verschiedene L, p dargestellt. Durch die Wahl der ReLU-Aktivierungsfunktion $\sigma(x) = \max\{x, 0\}$ ist f in jedem Fall abschnittsweise linear. Während im Falle $L = 0$ nur lineare Funktionen dargestellt werden können, sind für $L = 1$ abhängig von der Dimension $p_1 \in \mathbb{N}$ der *hidden layer* bereits vielfältige Formen möglich.

Wollen wir eine beliebige stetige Funktion $h : \mathbb{R} \to \mathbb{R}$ mit einem neuronalen Netzwerk $f \in \mathcal{F}(L, p)$ für Regression approximieren, so ist formal nur ein *hidden layer* ($L = 1$) notwendig, da h beliebig genau durch Summen von linear transformierten σ-Termen geschrieben werden kann.

In der Praxis (vgl. Abschn. 7.3) verwendet man jedoch auch in diesem Fall mindestens $L = 2$, da dann komplexe Strukturen mit wesentlich kleineren Dimensionen der *hidden layer* und somit einfacher dargestellt werden können.

Für Funktionen $h : \mathbb{R}^d \to \mathbb{R}$ sind je nach Größe von d und Komplexität von h eine größere Anzahl L von Layern nötig, um h gut durch ein Element von $\mathcal{F}(L, p)$ zu approximieren.

7.3 Inferenz neuronaler Netzwerke

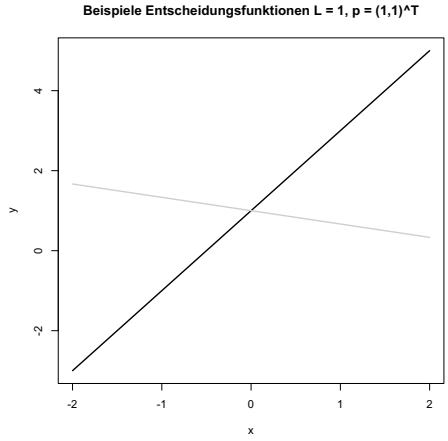
Sind Trainingsdaten $(X_i, Y_i)_{i=1,\dots,n}$ gegeben und wollen wir neuronale Netzwerke mit Netzwerkarchitektur (L, p) zur Ermittlung eines geeigneten Algorithmus nutzen, so entspricht dies folgender Modellannahme (vgl. Herleitung vor Definition 7.3):

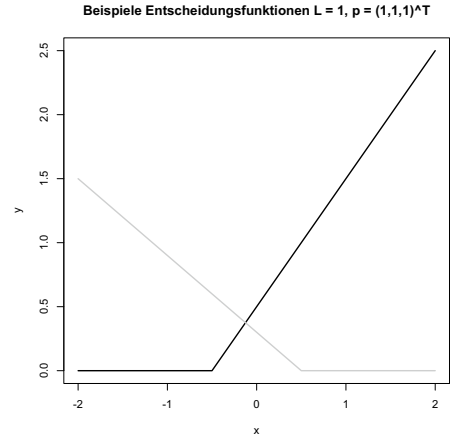
Modellannahme 7.6 (Neuronales Netzwerk)

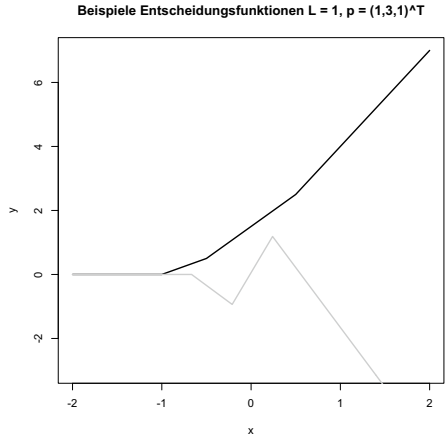
- *Regression:* Die Bayes-Regel erfüllt: $f^* \in \mathcal{F}(L, p)$.
- *Klassifikation:* Es gibt ein neuronales Netzwerk $g^* \in \mathcal{F}(L, p)$ für Klassifikation mit

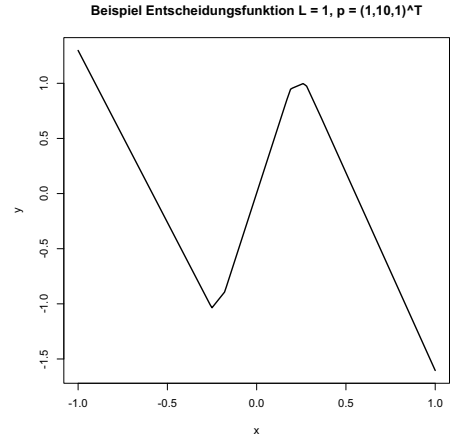
$$\mathbb{P}(Y = k | X = x) = M_k(g^*(x)), \quad k = 1, \dots, K, \quad x \in \mathcal{X}$$

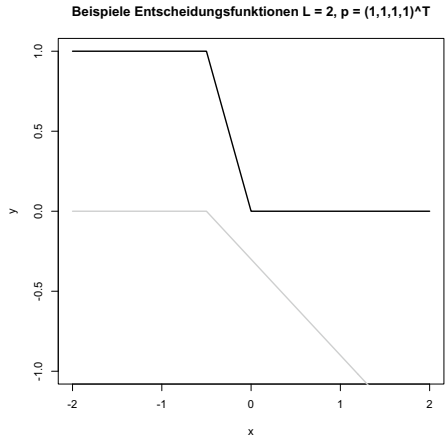
(alternativ: Die Funktionen $(g^*(x))_k$, $k = 1, \dots, K$ sind optimale Diskriminantenfunktionen).











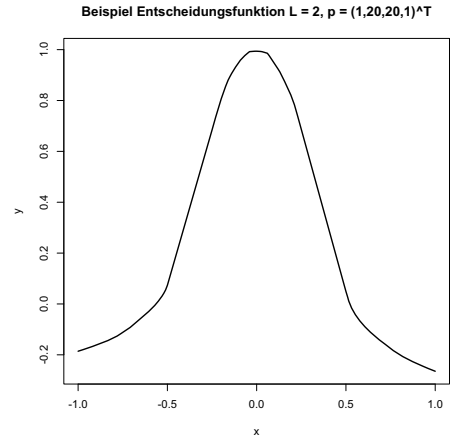


Abb. 7.3 Darstellung von Elementen $f \in \mathcal{F}(L, p)$ mit $p_0 = p_{L+1} = 1$, d. h. $f : \mathbb{R} \to \mathbb{R}$

Im Regressionsfall nutzen wir zur Bestimmung eines Algorithmus den Standardansatz aus Bemerkung 1.14, d. h., wir minimieren das empirische Risiko zusammen mit einem Bestrafungsterm, welcher kleine Werte in den Gewichtsmatrizen bevorzugt. Für $q \geq 1$ sei $\|A\|_q := \left(\sum_{j=1}^{p_1} \sum_{k=1}^{p_2} |A_{jk}|^q\right)^{1/q}$ die q-Norm einer Matrix $A \in \mathbb{R}^{p_1 \times p_2}$:

Definition 7.7 (Neuronaler Netzwerkalgorithmus für Regression)
Sei $L \in \mathbb{N}$, $p \in \mathbb{N}_0^{L+2}$. Für $g \in \mathcal{F}(L, p)$, definiere

$$J(g) = \sum_{l=1}^{L+1} \|W^{(l)}\|_2^2.$$

Sei $\lambda \geq 0$ beliebig. Sei

$$\hat{R}_n(g) := \frac{1}{n} \sum_{i=1}^{n} (Y_i - g(X_i))^2.$$

Dann heißt

$$\hat{f}_{n,\lambda}^{FNN} :\in \operatorname*{arg\,min}_{g \in \mathcal{F}(L,p)} \left\{ \hat{R}_n(g) + \lambda \cdot J(g) \right\} \tag{7.3}$$

neuronaler Netzwerkalgorithmus für Regression (kurz: NN-Algorithmus für Regression). ♦

Bemerkungen

- Mit der Abkürzung „FNN“ signalisieren wir, dass $\hat{f}_{n,\lambda}^{FNN}$ ein *fully connected neural network* ist.
- Wir haben hier eine Version formuliert, welche oft in der Praxis genutzt wird. Die Größe der Verschiebungsvektoren $v^{(l)}$ wird hier nicht bestraft, auch wenn das prinzipiell möglich ist. Aufgrund der großen Parameterzahl ist es naheliegend, statt $\|W^{(l)}\|_2^2$ zum Beispiel die 1-Norm $\|W^{(l)}\|_1$ zu nutzen (vgl. Lasso-Schätzer in Kap. 3), um viele Einträge von $W^{(l)}$ auf null zu setzen. Für hochdimensionale Probleme ist der Rechenaufwand dann jedoch viel größer, weil aufgrund der Nichtkonvexität von $\hat{R}_n(g)$ eine stabile Methode zur Ermittlung von $\hat{f}_{n,\lambda}^{FNN}$ fehlt.

Im Klassifikationsfall nutzen wir den Maximum-Likelihood-Ansatz der logistischen Regression, vgl. Definition 4.10. Aufgrund unserer Annahme erfüllt $g = g^* \in \mathcal{F}(L, p)$ die Gleichung

$$\mathbb{P}(Y = k | X = x) = M_k(g(x)), \quad k = 1, \ldots, K,$$

d. h.

$$\log \mathbb{P}(Y = y | X = x) = \sum_{k=1}^{K} \mathbb{1}_{\{y=k\}} \cdot \log M_k(g(x)) =: c_g(x, y).$$

Zur Bestimmung von g^* auf Basis der Trainingsdaten maximieren wir die empirische Entsprechung

$$\frac{1}{n}\sum_{i=1}^{n} c_g(X_i, Y_i) = \frac{1}{n}\sum_{i=1}^{n}\left\{\sum_{k=1}^{K} \mathbb{1}_{\{Y_i=k\}} \cdot \log M_k(g(X_i))\right\}$$

bzw. minimieren $-\frac{1}{n}\sum_{i=1}^{n} c_g s(X_i, Y_i)$ in Abhängigkeit von $g \in \mathcal{F}(L, p)$:

Definition 7.8 (Neuronaler Netzwerkalgorithmus, Klassifikation)
Sei $L \in \mathbb{N}$, $p \in \mathbb{N}_0^{L+2}$ fest. Sei $\lambda \geq 0$ beliebig. Sei

$$\hat{R}_n(g) := -\sum_{k=1}^{K}\left\{\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{Y_i=k\}} \cdot \log M_k(g(X_i))\right\}.$$

Dann heißt $\hat{f}_{n,\lambda}^{FNN}(x) = \arg\max_{k=1,\ldots,K} M_k\left(\hat{g}_{n,\lambda}^{FNN}(x)\right)$ mit

$$\hat{g}_{n,\lambda}^{FNN} :\in \underset{g\in\mathcal{F}(L,p)}{\arg\min}\left\{\hat{R}_n(g) + \lambda \cdot J(g)\right\}$$

neuronaler Netzwerkalgorithmus für Klassifikation (kurz: NN-Algorithmus für Klassifikation). ♦

Aufgrund der vielen Parameter, von welchen $g \in \mathcal{F}(L, p)$ abhängt, kann der Optimierer $\hat{f}_{n,\lambda}^{FNN}$ aus Definition 7.7 und 7.8 für hochdimensionale Probleme nicht in angemessener Rechenzeit ermittelt werden. Durch die fehlende Konvexität hat die Funktion $g \mapsto \{\hat{R}_n(g) + \lambda \cdot J(g)\}$ neben dem globalen Minimum oft viele lokale Minima. In der Praxis beschränkt man sich auf die Bestimmung eines solchen lokalen Minimums und hofft, dass dieses ähnliche Eigenschaften wie das globale Minimum aufweist.

Zur Bestimmung eines lokalen Minimums werden iterative Gradientenmethoden genutzt. In jedem Schritt werden die Parameter des neuronalen Netzwerks dazu ein Stück in die Richtung des negativen Gradienten verschoben, was zu einer Reduktion des Funktionswerts $\hat{R}_n(g) + \lambda \cdot J(g)$ führt. Um diese übersichtlich darstellen zu können, definieren wir für eine Matrix $A \in \mathbb{R}^{p\times q}$ den Ausdruck

$$\text{vec}(A) = (A_{11}, \ldots, A_{p1}, A_{12}, \ldots, A_{p2}, \ldots, A_{1q}, \ldots, A_{pq})^T \in \mathbb{R}^{pq}, \tag{7.4}$$

die spaltenweise Vektorisierung der Matrix. Entsprechend sei

$$\theta^{(l)} = \begin{pmatrix} \mathrm{vec}(W^{(l)}) \\ v^{(l)} \end{pmatrix}, \qquad \theta = \begin{pmatrix} \theta^{(1)} \\ \vdots \\ \theta^{(L)} \\ \mathrm{vec}(W^{(L+1)}) \end{pmatrix} \tag{7.5}$$

die Vektorisierung des gesamten Parametervektors einer Funktion $g \in \mathcal{F}(L, p)$. Ist θ der zu $g \in \mathcal{F}(L, p)$ gehörige vektorisierte Parametervektor, so schreiben wir auch $g = g_\theta$. Mit diesen Bezeichnungen erhalten wir folgende Definition:

Definition 7.9 (Gradientenmethode zur Bestimmung des NN-Algorithmus)
Sei $(\gamma_t)_{t\in\mathbb{N}}$ eine Folge reeller Zahlen *(learning rate)*. Sei $w > 0$.

Wähle $W_{jk}^{(i)[0]}$ zufällig gleichverteilt auf $[-w, w]$, $v_j^{(i)[0]} := 0$ und $\theta^{[0]}$ entsprechend (7.5).

Für $t = 1, 2, 3, \ldots, T \in \mathbb{N}$, berechne:

$$\theta^{[t]} := \theta^{[t-1]} - \gamma_t \cdot \partial_\theta \left[\hat{R}_n(g_{\theta^{[t-1]}}) + \lambda \cdot J(g_{\theta^{[t-1]}}) \right] \tag{7.6}$$

Setze

- im Regressionsfall: $\hat{f}_{n,\lambda}^{FNN,grad} := g_{\theta^{[T]}}$,
- im Klassifikationsfall: $\hat{f}_{n,\lambda}^{FNN,grad} := \arg\max_{k\in\{1,\ldots,K\}} M_k(g_{\theta^{[T]}})$. ♦

Bemerkung 7.10

- Ein Iterationsschritt der Gradientenmethode (d. h. von $\theta^{[t-1]}$ zu $\theta^{[t]}$) nennt man *Trainingsepoche*. Der Name bezieht sich darauf, dass alle Trainingsdaten einmal verwendet werden, um den Parameter zu aktualisieren. Andere Algorithmen (vgl. Definition 7.18) verwenden nur einen Teil der Trainingsdaten zum Aktualisieren der Parameter.
- Gradientenmethoden garantieren nicht, dass ein globales Minimum (und damit $\hat{f}_{n,\lambda}^{FNN}$) gefunden wird, sondern nur ein lokales Minimum.
- Die *learning rate* γ_t wird üblicherweise recht klein gewählt, zum Beispiel $\gamma_t \approx 10^{-4}$ (wobei dies natürlich abhängig von der Skalierung der Trainingsdaten und des konkreten Problems ist). Um Konvergenz zu gewährleisten, kann man γ_t auch von t abhängen lassen; eine aus der stochastischen Optimierung motivierte Wahl sind γ_t mit $\sum_{t=1}^{\infty} \gamma_t = \infty$ und $\sum_{t=1}^{\infty} \gamma_t^2 < \infty$.
- Bei Anwendung von Gradientenmethoden auf nichtkonvexe Zielfunktionen (wie hier mit $g \mapsto \{\hat{R}_n(g) + \lambda \cdot J(g)\}$ ist eine gute Wahl der Startparameter von entscheidender Bedeutung. Sind die Startparameter sehr weit von den interessanten lokalen Extremstellen entfernt, so ist eine Annäherung mittels Gl. (7.6) nur mit großen Schrittweiten γ_t möglich. Dies führt jedoch oft zu einem „Überspringen“ der lokalen Extremstellen. In der Praxis bedeutet dies, dass $\hat{f}_{n,\lambda}^{FNN,grad}$ auf dem interessanten Bereich, wo die Trainingsdaten

beobachtet wurden, stets nur die Nullfunktion oder eine schlechte Approximation von f^* ist.

Bei der Anwendung der Gradientenmethode aus Definition 7.9 wird daher oft ein etwas größeres Netzwerk (d. h. eine größere Menge an Parametern) gewählt, als eigentlich zur Beschreibung von f^* nötig ist. Die Zielfunktion besitzt dann eine höhere Zahl von (interessanten) lokalen Extremstellen, was es einfacher macht, geeignete Startwerte zu finden.

Eine Initialisierung der Gewichtsmatrizen mit null entspricht sehr ungünstigen Startwerten, da die Ableitungen der Zielfunktion dann null oder sehr nahe null sind (vgl. Algorithmus 7.18) und eine Verschiebung der Parameter zu den Extremstellen wiederum nur mit großer Schrittweite und der damit verbundenen Instabilität möglich ist.

Es ist daher von zentraler Bedeutung, die Startwerte der Gewichtsmatrizen mit einem zufälligen kleinen Rauschen ($w > 0$ klein) festzulegen. Anschaulich liegen die Startwerte dadurch näher bzw. günstiger an lokalen Extremstellen. Durch die zufällige Initialisierung werden allerdings auch einige Komponenten von $\theta^{[0]}$ weiterhin „ungünstig“ gewählt und sind dann im weiteren Verlauf der Gradientenmethode unbrauchbar, da ihre Konvergenz im Vergleich zu anderen, ‚günstig‘ initialisierten Komponenten viel langsamer ist. Dies ist ein weiterer Grund, warum man ein etwas größeres Netzwerk als eigentlich nötig wählen muss.

Durch genauer auf das vorliegende Problem zugeschnittene Modellierungen der neuronalen Netzwerke und fortgeschrittene numerische Algorithmen kann die „Verschwendung“ von Parametern verringert werden.

Um das Verhalten der Gradientenmethode in der Praxis zu diskutieren, betrachten wir die folgenden beiden Beispiele:

Beispiel 7.11 (Regression) Wir betrachten $n = 30$ Trainingsdaten (X_i, Y_i), welche dem Regressionsmodell

$$Y_i = f^*(X_i) + \varepsilon_i, \qquad i = 1, \ldots, n,$$

folgen, wobei $X_i \sim U[-\frac{1}{2}, \frac{1}{2}]$ i. i. d. gleichverteilt auf $[-\frac{1}{2}, \frac{1}{2}]$ und $\varepsilon_i \sim N(0, \tau^2)$ i. i. d. normalverteilt mit Parameter $\tau \in \{0{,}03, 0{,}3\}$ sind.

Zur Schätzung von f^* verwenden wir $\hat{f}_{n,\lambda}^{FNN,grad} \in \mathcal{F}(L, p)$ aus Definition 7.9 mit $\gamma = 0{,}01$. Die Gewichtsmatrizen werden mit $w = 0{,}3$ initialisiert.

Sei zunächst $\tau = 0{,}03$. Zur Schätzung nutzen wir neuronale Netzwerke mit $L = 1$, $p = (1, 10, 1)^T$, $\lambda = 0{,}001$. In Abb. 7.4a)-e) sind einige Zwischenresultate $g_{\theta^{[T]}}$ nach T Trainingsepochen dargestellt. Nach $T = 2000$ Trainingsepochen ist das Gradientenverfahren konvergiert und liefert $\hat{f}_{n,\lambda}^{FNN,grad} = g_{\theta^{[2000]}}$. In Abb. 7.4f) ist ein alternatives Ergebnis bei anderer Initialisierung der Gewichtsmatrizen dargestellt. Die beiden verschiedenen Resultate für $\hat{f}_{n,\lambda}^{FNN,grad}$ in Abb. 7.4e),f) machen deutlich, dass das Funktional in Gl. (7.3) mehrere lokale Minima besitzt.

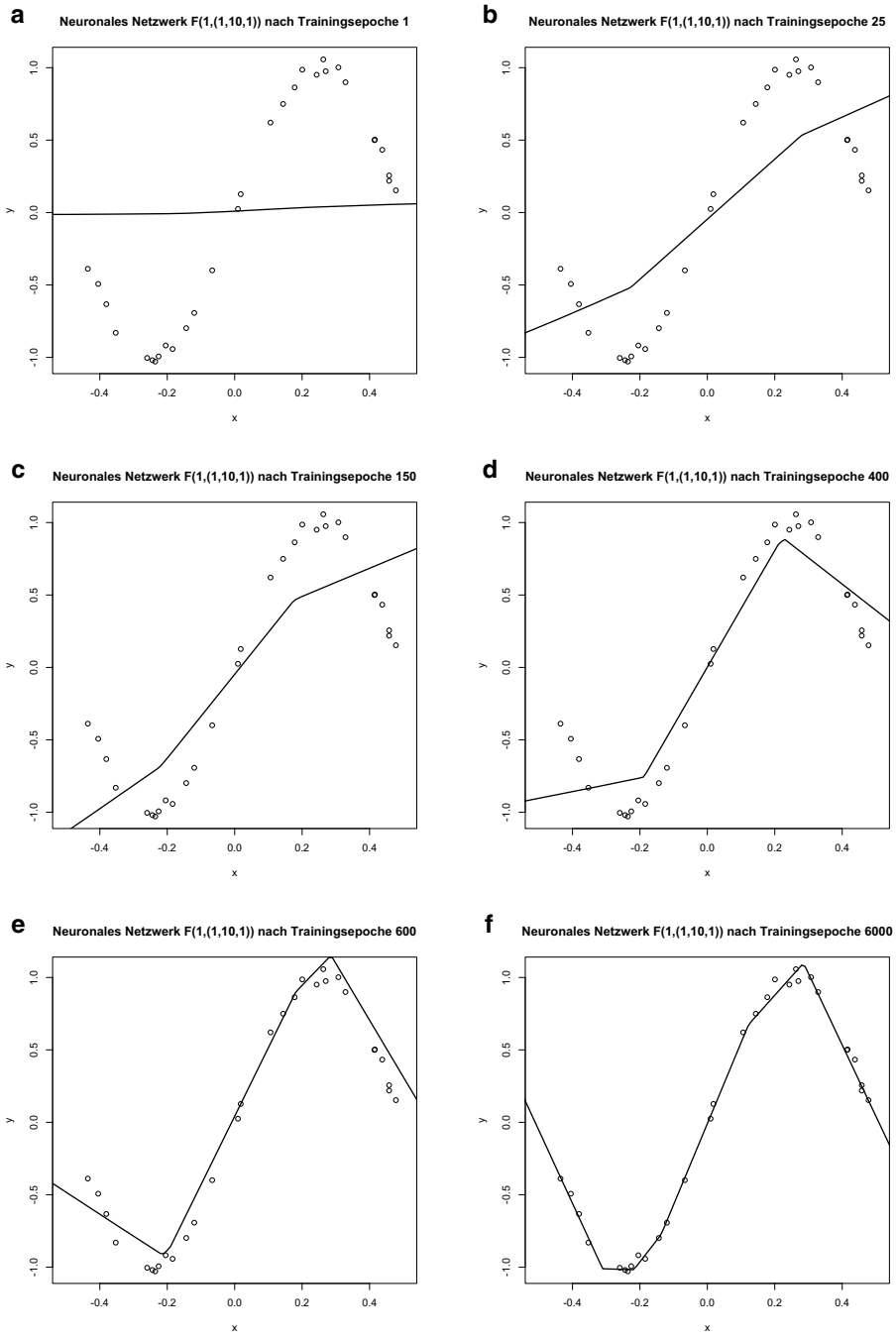


Abb. 7.4 a)-e): Darstellung der Zwischenergebnisse $g_{\theta[T]}$ für $\hat{f}_{n,\lambda}^{FNN,grad} \in \mathcal{F}(1,(1,10,1)^T)$ nach T Trainingsepochen angewandt auf Trainingsdaten aus Beispiel 7.11. f): Ein alternatives Resultat für $\hat{f}_{n,\lambda}^{FNN,grad}$, erhalten durch eine andere Initialisierung der Gewichtsmatrizen

Eine zu große Wahl von λ erzeugt große Abweichungen von $\hat{f}_{n,\lambda}^{FNN,grad}$ zu f^*, während eine zu kleine Wahl von λ zu instabileren Iterationsschritten führt.

Sei nun $\tau = 0{,}3$. Zur Schätzung nutzen wir neuronale Netzwerke mit $L = 2$, $p = (1, 20, 20, 1)^T$ und $\lambda = 0{,}0001$. In Abb. 7.5 sind wieder Zwischenresultate $g_{\theta[T]}$ nach T Trainingsepochen dargestellt. Man sieht, dass die Gradientenmethode hier nach etwa $T = 300$ Trainingsepochen (und trotz des größeren Rauschens der Trainingsdaten) eine gute Approximation von f^* liefert. Aufgrund des geringen Werts von λ konvergiert $g_{\theta[T]}$ für große T hier gegen ein überangepasstes $\hat{f}_{n,\lambda}^{FNN,grad}$. Die in Algorithmus 7.19 vorgestellte Methode *(Stochastic gradient descent)* liefert nach derselben Anzahl von $T = 80.000$ Trainingsepochen ein nicht überangepasstes $\hat{f}_{n,\lambda}^{FNN,sgd} = g_{\tilde{\theta}[T]}$.

Beispiel 7.12 (Klassifikation) Wir betrachten $n = 300$ Trainingsdaten (X_i, Y_i) des folgenden Klassifikationsproblems: Es seien X_{i1}, X_{i2} i. i. d. gleichverteilt auf $[0, 1]$, $X_i = (X_{i1}, X_{i2})^T$. Unabhängig davon seien $\varepsilon_i \sim N(0, 0{,}2^2)$ i. i. d. Für $x = (x_1, x_2)^T \in \mathbb{R}^2$ sei $\delta^*(x) = x_2 - 0{,}5 - 0{,}3\sin(2\pi x_1)$, und

$$Y_i = \begin{cases} 1, & h(X_i) - \varepsilon_i \leq 0 \\ 2, & h(X_i) - \varepsilon_i > 0 \end{cases}.$$

Entsprechend ist der Bayes-Klassifizierer des Problems gegeben durch

$$f^*(x) = \begin{cases} 1, & \delta^*(x) \leq 0 \\ 2, & \delta^*(x) > 0 \end{cases}.$$

Zur Ermittlung von f^* nutzen wir $\hat{f}_{n,\lambda}^{FNN,grad}$ basierend auf neuronalen Netzwerken $\mathcal{F}(L, p)$ mit $L = 2$ und $p = (2, 20, 20, 2)^T$ sowie $\lambda = 0{,}001$ und $\gamma = 0{,}002$. In Abb. 7.6 sind einige Zwischenresultate $g_{\theta[T]}$ nach T Trainingsepochen dargestellt. Nach ca. $T = 3000$ Trainingsepochen ist eine gute Näherung von f^* erreicht; weitere Iterationen führen dann zu einer Überanpassung.

In der Praxis kann die Überanpassung durch neuronale Netzwerke mittels höherer Werte von λ sowie einer dynamischen Wahl von $\gamma = \gamma_t$ verhindert werden; dies ist jedoch von Problem zu Problem verschieden. Eine andere Möglichkeit ist die Überwachung des Validierungsfehlers mittels eines vorher separierten Teils der Trainingsdaten, vgl. Bemerkung 2.21. Gerade bei neuronalen Netzwerken funktioniert letztere Methode relativ gut, da mittels des Gradientenverfahrens schnell eine gute Approximation von f^* erreicht wird, während der Übergang zu einer Überanpassung an die Trainingsdaten sehr viele weitere Iterationen benötigt, vgl. die Abb. 7.5 und 7.6.

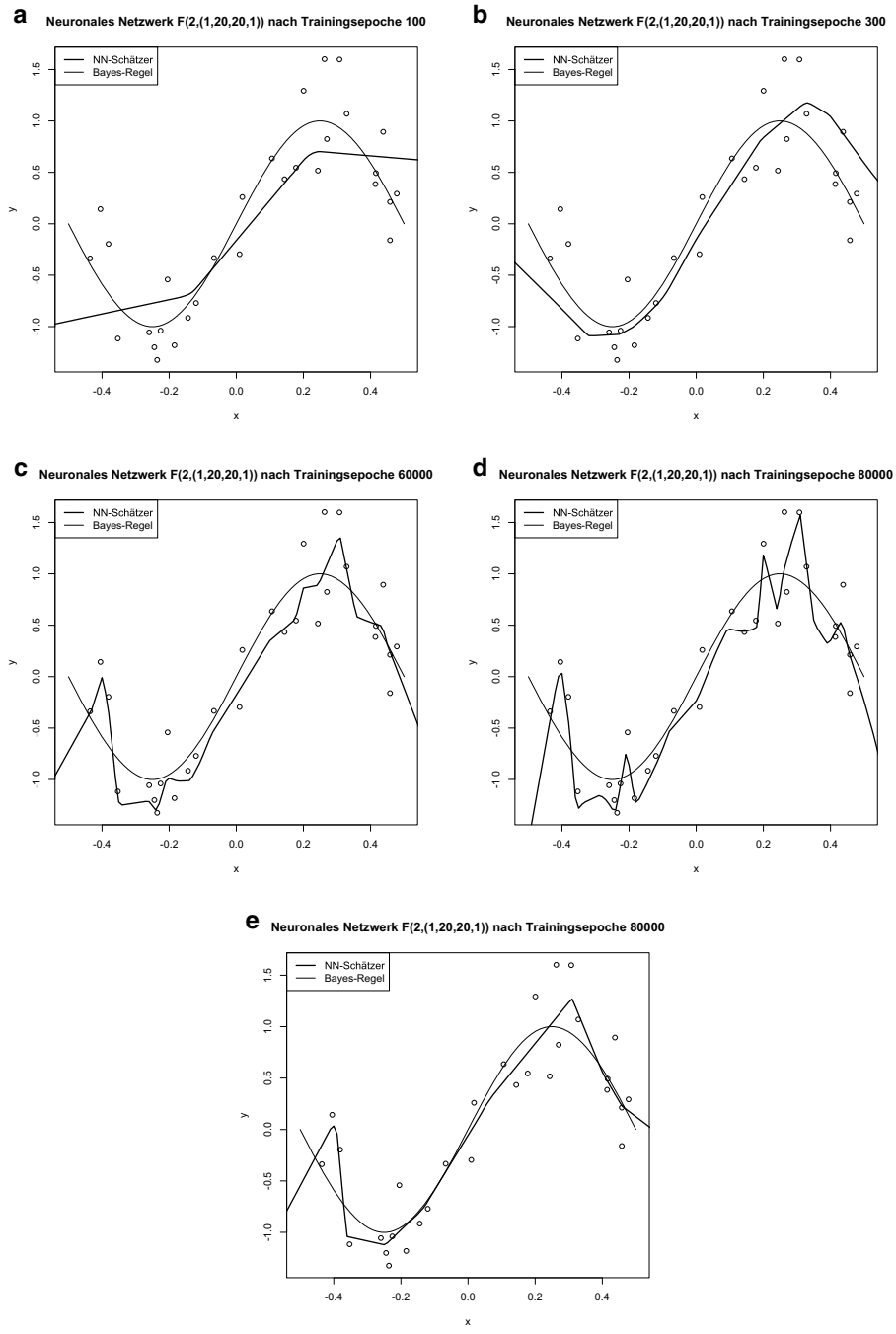


Abb. 7.5 a)-d): Darstellung der Zwischenergebnisse $g_{\theta[T]}$ für $\hat{f}_{n,\lambda}^{FNN,grad} \in \mathcal{F}(1,(1,10,1)^T)$ nach T Trainingsepochen angewandt auf Trainingsdaten aus Beispiel 7.11. e): $\hat{f}_{n,\lambda}^{FNN,sgd} = g_{\tilde{\theta}[T]}$ nach $T = 80.000$ Trainingsepochen

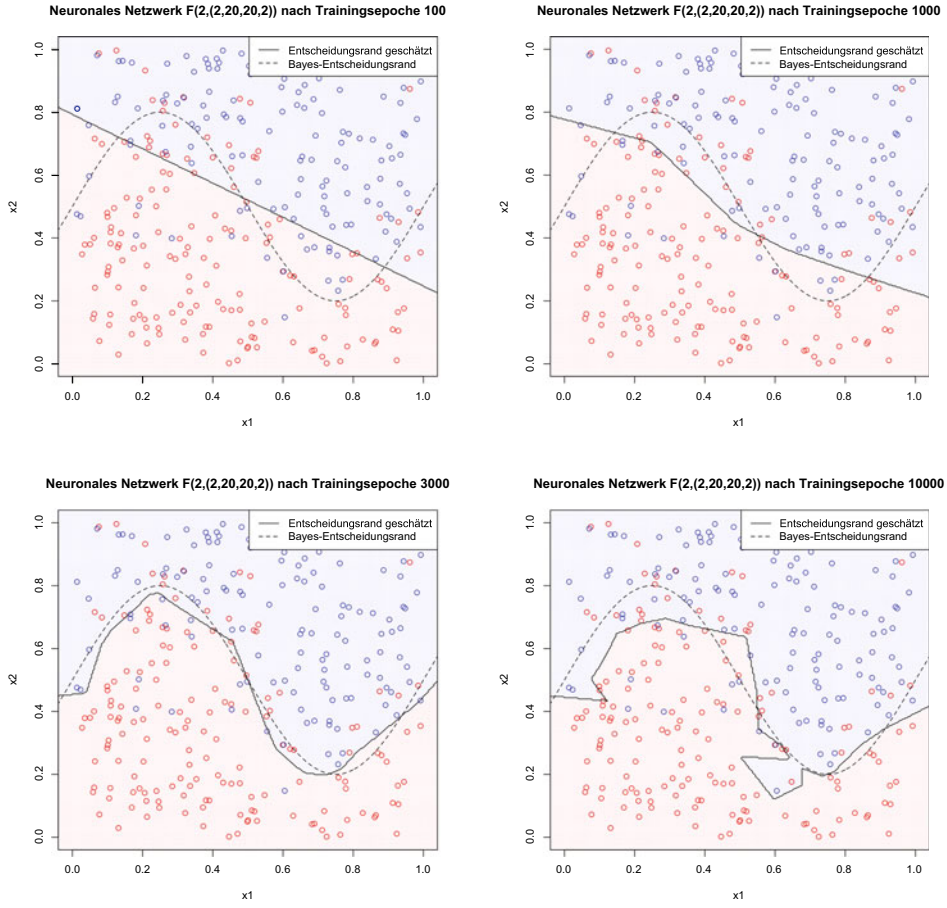


Abb. 7.6 Darstellung der Zwischenergebnisse $g_{\theta[T]}$ für $\hat{f}_{n,\lambda}^{FNN,grad} \in \mathcal{F}(2, (2, 20, 20, 2)^T)$ nach T Trainingsepochen angewandt auf Trainingsdaten aus Beispiel 7.12

Die Berechnung eines Iterationsschritts in Gl. (7.6) der Gradientenmethode ist recht aufwendig. In Abschn. 7.3.1 zeigen wir, wie diese systematisch durchgeführt werden kann.

7.3.1 Forward Propagation und Back Propagation

Aufgrund der rekursiven Struktur von $g \in \mathcal{F}(L, p)$ kann ein Iterationsschritt des Gradientenverfahrens in mehrere Teilschritte zerlegt werden. Basis dafür ist die folgende Zerlegung eines jeden $g \in \mathcal{F}(L, p)$:

Definition 7.13
Für $g \in \mathcal{F}(L, p)$ definiere $x^{(0)}(x) = x$ und

$$x^{(l)}(x) := \sigma\left(z^{(l)}(x)\right), \qquad z^{(l)}(x) := v^{(l)} + W^{(l)} \cdot x^{(l-1)}(x), \qquad l = 1, \ldots, L, \tag{7.7}$$

wobei $\sigma(\cdot)$ komponentenweise angewandt wird.

Dann ist $g(x) = W^{(L+1)} \cdot x^{(L)}(x)$. ♦

Es gilt stets

$$\hat{R}_n(g) = \frac{1}{n} \sum_{i=1}^{n} C\left(Y_i, g(X_i)\right),$$

wobei im Regressionsfall $C(y, s) = (y - s)^2$ und im Klassifikationsfall $C(y, s) = -\left(s_y - \log \sum_{v=1}^{K} e^{s_v}\right)$ ist. Aufgrund der additiven Struktur $\hat{R}_n(g)$ reduziert sich die Bestimmung der Ableitung von $\hat{R}_n(g)$ auf die Bestimmung der Ableitung von $C(Y_i, g(X_i))$, $i = 1, \ldots, n$.

Im Folgenden sei daher $i \in \{1, \ldots, n\}$ fest gewählt. Zur besseren Übersicht schreiben wir kurz $C_i := C(Y_i, g(X_i))$ und nutzen die Abkürzungen $x^{(l)} = x^{(l)}(X_i)$ und $z^{(l)} = z^{(l)}(X_i)$.

Der Einfluss des l-ten Layers auf C_i lässt sich quantifizieren mittels der folgenden Größen:

Definition 7.14
Für $l = 1, \ldots, L$, definiere

$$\delta_j^l := \frac{\partial C_i}{\partial z_j^{(l)}}.$$

Dann heißt $\delta^l := (\delta_j^l)_{j=1,\ldots,p_l}$ der Einfluss des l-ten Layers auf C_i. ♦

Die δ_j^l sind zentral für die Berechnung der Ableitungen von C_i nach den Parametern $v^{(l)}$, $W^{(l)}$. Zunächst leiten wir Zusammenhänge zwischen den $\delta^l, l = 1, \ldots, L$, her. Für Vektoren $a, b \in \mathbb{R}^p$ bezeichne $a \odot b = (a_j b_j)_{j=1,\ldots,p}$ das Hadamard-Produkt von a und b.

Lemma 7.15 (Rekursionsgleichungen für δ^l) Es gilt

(i) für den obersten Layer:

$$\delta^L = \frac{\partial C_i}{\partial x^{(L)}} \odot \sigma'\left(z^{(L)}\right).$$

(ii) für die restlichen Layer ($l = 1, \ldots, L-1$):

$$\delta^l = \left[(W^{(l+1)})^T \delta^{l+1}\right] \odot \sigma'\left(z^{(l)}\right). \tag{7.8}$$

Beweis

(i) Es ist

$$\delta_j^L = \frac{\partial C_i}{\partial z_j^{(L)}} = \frac{\partial C_i}{\partial x_j^{(L)}} \cdot \frac{\partial x_j^{(L)}}{\partial z_j^L} = \frac{\partial C_i}{\partial x_j^{(L)}} \cdot \sigma'(z_j^{(L)}).$$

(ii) Beachte: Jede Komponente von $z^{(l+1)}$ wird von $z_j^{(l)}$ beeinflusst. Daher gilt

$$\delta_j^l = \frac{\partial C_i}{\partial z_j^{(l)}} = \frac{\partial C_i}{\partial x_j^{(l)}} \frac{\partial x_j^{(l)}}{\partial z_j^{(l)}},$$

$$\frac{\partial C_i}{\partial x_j^{(l)}} = \sum_m \frac{\partial C_i}{\partial z_m^{(l+1)}} \cdot \frac{\partial z_m^{(l+1)}}{\partial x_j^{(l)}} = \sum_m \delta_m^{l+1} \cdot W_{mj}^{(l+1)} = (W_{\cdot j})^T \delta^{l+1},$$

d. h.

$$\delta_j^l = \left(W_{\cdot j}^{(l+1)}\right)^T \delta^{l+1} \cdot \sigma'\left(z_j^{(l)}\right).$$

Mittels der δ^l können leicht die Ableitungen der C_i nach $v^{(l)}$, $W^{(l)}$ bestimmt werden:

Lemma 7.16 (Zusammenhang zwischen δ^l und Parameterableitungen) Es gilt für $l = 1, \ldots, L$:

(i) $\frac{\partial C_i}{\partial v^{(l)}} = \delta^l$.
(ii) $\frac{\partial C_i}{\partial W_{jm}^{(l)}} = \delta_j^l \cdot x_m^{(l-1)}$.

Beweis

(i) $\frac{\partial C_i}{\partial v_j^{(l)}} = \frac{\partial C_i}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial v_j^{(l)}} = \delta_j^l \cdot 1 = \delta_j^l$.

(ii) $\frac{\partial C_i}{\partial W_{jm}^{(l)}} = \frac{\partial C_i}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial W_{jm}^{(l)}} = \delta_j^l \cdot x_m^{(l-1)}$.

Um δ^L gemäß Lemma 7.15(i) bestimmen zu können, muss noch $\frac{\partial C_i}{\partial x^{(L)}}$ ermittelt werden. Diese ist bei Regressions- und Klassifikationsproblemen unterschiedlich:

Lemma 7.17 (Einfluss L-ter Layer auf C_i) Es gilt

(i) für Regression:

$$\frac{\partial C_i}{\partial x_j^{(L)}} = -2(Y_i - g(X_i)) \cdot W_{1j}^{(L+1)}$$

und

$$\frac{\partial C_i}{\partial W_{1m}^{(L+1)}} = -2(Y_i - g(X_i)) \cdot x_m^{(L)}.$$

(ii) für Klassifikation:

$$\frac{\partial C_i}{\partial x_j^{(L)}} = -\left(W_{Y_i j}^{(L+1)} - \sum_{k=1}^{K} M_k(g(X_i)) \cdot W_{kj}^{(L+1)} \right)$$

und

$$\frac{\partial C_i}{\partial W_{jm}^{(L+1)}} = -\left\{ \mathbb{1}_{\{Y_i = j\}} - M_j(g(X_i)) \right\} \cdot x_m^{(L)}.$$

Beweis

(i) $\frac{\partial C}{\partial s} = -2(y - s)$. Mit $s = W^{(L+1)} x^{(L)}$ folgt die Behauptung.

(ii) Es ist $\frac{\partial C}{\partial s_k} = \mathbb{1}_{\{y=k\}} - \frac{e^{s_k}}{\sum_{v=1}^{K} e^{s_v}}$. Es folgt mit $s = W^{(L+1)} x^{(L)}$ (und daher $s_k = W_{k\cdot}^{(L+1)} x^{(L)}$ und $\frac{\partial s_k}{\partial x_j^{(L)}} = W_{kj}^{(L+1)}$):

$$\begin{aligned}
\frac{\partial C_i}{\partial x_j^{(L)}} &= \sum_k \frac{\partial C_i}{\partial s_k} \cdot \frac{\partial s_k}{\partial x_j^{(L)}} = -\sum_k \left\{ \mathbb{1}_{\{Y_i = k\}} - \frac{e^{W_{k\cdot}^{(L+1)} x^{(L)}}}{\sum_{v=1}^{K} e^{W_{v\cdot}^{(L+1)} x^{(L)}}} \right\} W_{kj}^{(L+1)} \\
&= -\left(W_{Y_i j}^{(L+1)} - \frac{\sum_{k=1}^{K} e^{W_{k\cdot}^{(L+1)} x^{(L)}} W_{kj}^{(L+1)}}{\sum_{v=1}^{K} e^{W_{v\cdot}^{(L+1)} x^{(L)}}} \right)
\end{aligned}$$

Mit $\frac{\partial s_k}{W_{jm}^{(L+1)}} = \mathbb{1}_{\{j=k\}} x_m^{(L)}$ folgt

$$\begin{aligned}
\frac{\partial C_i}{\partial W_{jm}^{(L+1)}} &= \sum_k \frac{\partial C_i}{\partial s_k} \cdot \frac{\partial s_k}{\partial W_{jm}^{(L+1)}} = -\sum_k \left\{ \mathbb{1}_{\{Y_i = k\}} - \frac{e^{W_{k\cdot}^{(L+1)} x^{(L)}}}{\sum_{v=1}^{K} e^{W_{v\cdot}^{(L+1)} x^{(L)}}} \right\} x_m^{(L)} \mathbb{1}_{\{j=k\}} \\
&= -\left(\mathbb{1}_{\{Y_i = j\}} x_m^{(L)} - \frac{e^{W_{j\cdot}^{(L+1)} x^{(L)}} x_m^{(L)}}{\sum_{v=1}^{K} e^{W_{v\cdot}^{(L+1)} x^{(L)}}} \right).
\end{aligned}$$

Die Ergebnisse der Lemmata 7.15, 7.16 und 7.17 ergeben zusammen einen Algorithmus, welcher *einen Schritt* der Gradientenmethode aus Definition 7.9 (d. h. den Übergang von $\theta^{[r-1]}$ zu $\theta^{[r]}$) durchführt. Dieser Schritt ist unterteilt in zwei weitere Schritte:

- Berechne die Werte $z^{(l)} = z^{(l)}(X_i)$, $x^{(l)} = x^{(l)}(X_i)$ und $g(X_i)$ des neuronalen Netzwerks in allen Layern. Die Trainingsdaten werden also durch das Netzwerk „durchgereicht“, daher wird dieser Schritt *Forward Propagation* genannt.

- Berechne die Ableitungen der Parameter für die jeweiligen Layer des Netzwerks. Aufgrund der Rekursionsstruktur beginnt man vom letzten Layer L und arbeitet sich zum ersten Layer vor. Daher trägt dieser Schritt den Namen *Back Propagation.*

Formal erhalten wir folgenden Algorithmus:

Algorithmus 7.18 (Forward- und Back Propagation/Regression) Für jedes $i = 1, \ldots, n$, führe durch:

1. *(Forward Propagation)* Berechne $z^{(l)} = z^{(l)}(X_i)$, $x^{(l)} = x^{(l)}(X_i)$ und $g(X_i)$ rekursiv gemäß (7.7) für $l = 1, \ldots, L$ mit Parametern $\theta = \theta^{[t-1]}$.
2. *(Back Propagation)*

(2a) Berechne δ^L gemäß Lemma 7.15(i) und Lemma 7.17(i):

$$\delta^L := -2(Y_i - g(X_i)) \cdot (W_{1\cdot}^{(L+1)[t-1]})^T \odot \sigma'(z^{(L)})$$

und

$$\frac{\partial C_i}{\partial W_{1m}^{(L+1)}} = -2(Y_i - g(X_i)) \cdot x_m^{(L)}.$$

(2b) Für $l = 1, \ldots, L$, berechne δ^l gemäß Lemma 7.15(ii):

$$\delta^l := \left[\left(W^{(l+1)[t-1]} \right)^T \delta^{l+1} \right] \odot \sigma'(z^{(l)})$$

und die Ableitungen gemäß Lemma 7.16(i),(ii):

$$\frac{\partial C_i}{\partial v^{(l)}} = \delta^l, \qquad \frac{\partial C_i}{\partial W_{jm}^{(l)}} = \delta_j^l \cdot x_m^{(l-1)}$$

Berechne

$$v^{(l)[t]} := v^{(l)[t-1]} - \gamma_t \cdot \frac{1}{n} \sum_{i=1}^{n} \frac{\partial C_i}{\partial v^{(l)}}, \qquad l = 1, \ldots, L, \tag{7.9}$$

$$W_{jm}^{(l)[t]} := W_{jm}^{(l)[t-1]} - \gamma_t \cdot \left\{ \frac{1}{n} \sum_{i=1}^{n} \frac{\partial C_i}{\partial W_{jm}^{(l)}} + 2\lambda \cdot W_{jm}^{(l)[t-1]} \right\}, \qquad l = 1, \ldots, L+1. \tag{7.10}$$

Die Vektorisierung von $v^{(l)[t]}$, $W^{(l)[t]}$ gemäß Gl. (7.5) entspricht $\theta^{[t]}$ aus Definition 7.9; die zugehörigen Algorithmen bleiben dieselben.

Bemerkungen

- Für Klassifikationsprobleme muss nur Schritt (2a) gemäß Lemma 7.17(ii) abgewandelt werden.
- Bei neuronalen Netzwerken $g \in \mathcal{F}(L, p)$ mit einer großen Anzahl an Layern (d. h. L groß) verändern sich die Parameter der Layer nahe am Input in einem Schritt der Gradientenmethode nur sehr schwach. Dies folgt aus der Rekursionsgleichung (7.8): Die Änderungen δ^l in Layer l entstehen durch Multiplikation von δ^{l+1} mit einer relativ kleinen Zahl ($\sigma'(z^{(l)})$ bzw. $W^{(l+1)}$), d. h., δ^l fällt geometrisch in $L-l$. In der Praxis verwendet man modifizierte Gradientenverfahren, um auch große Ableitungen und Parameteränderungen in den ersten Layern zu erhalten.
- Aufgrund der Rekursionsgleichung (7.8) ist auch die Beschaffenheit der Aktivierungsfunktion σ von wesentlicher Bedeutung für die Konvergenzgeschwindigkeit des Gradientenverfahrens. Ist $\sigma'(x)$ für viele $x \in \mathbb{R}$ klein, so ergeben sich gemäß Gl. (7.8) nur langsame Änderungen der Parameter. Bei der Sigmoidfunktion $\sigma(x) = \frac{e^x}{1+e^x}$ ist beispielsweise $\sigma'(x)$ klein für $|x| \gg 1$. In der Praxis wird daher bei großen Netzwerken häufiger die ReLU-Funktion $\sigma(x) = \max\{x, 0\}$ genutzt.

Neuronale Netzwerke als Modelle zur Approximation von Funktionen wurden schon vor mehreren Jahrzehnten entwickelt. Die Computerarchitekturen sind allerdings erst seit etwa einem Jahrzehnt in der Lage, die nötigen Ressourcen für hochdimensionale Probleme bereitzustellen. Der Aufbau neuronaler Netzwerke ist darauf ausgerichtet, beim Lösen des zugehörigen Optimierungsproblems nur möglichst „einfache“ Operationen zu nutzen, die von Computern schnell berechnet werden können. Die gesamte Modellierung erfolgt stets mit der Motivation, diese auch in der Praxis umsetzen zu können, und ist *nicht* theoretisch motiviert. Eine wesentliche Operation in Algorithmus 7.18 sind Matrixmultiplikationen. Diese können von Grafikkarten schnell durchgeführt werden, ein wesentlicher Teil der Berechnungen kann somit ausgelagert und parallelisiert werden.

7.3.2 Stochastic gradient descent

Je nach Größe des Netzwerks und der Anzahl der Trainingsdaten n kann bereits die Berechnungszeit für einen Schritt des Gradientenverfahrens gemäß Algorithmus 7.18 recht hoch sein. In der Anwendung sind jedoch auch unvollständig gelernte Algorithmen von Bedeutung; sie können zum Beispiel für Prognosen verwendet werden. Aufgrund der Summenstruktur des empirischen Risikos

$$\hat{R}_n(g) = \frac{1}{n} \sum_{i=1}^{n} C_i$$

und der damit verbundenen Summenstruktur der Aktualisierungsgleichungen (7.9) und (7.10) kann die Aktualisierung der Parameter für jedes Trainingsdatum (X_i, Y_i) einzeln ausgeführt werden und liefert schneller Zwischenergebnisse des Anpassungsvorgangs.

Die Reihenfolge der $i \in \{1, \ldots, n\}$ kann dabei auf viele verschiedene, heuristisch motivierte Arten festgelegt werden. Zwei oft genutzte Möglichkeiten sind folgende:

a) Wähle in *jedem* Schritt $i \in \{1, \ldots, n\}$ zufällig.
b) Wähle eine zufällige Reihenfolge $i_1, \ldots, i_n$ von $\{1, \ldots, n\}$. Der Index i durchläuft nacheinander $i_1, \ldots, i_n$.

Bei (b) wird jedes Trainingsdatum nach n Schritten einmal genutzt worden sein, nur die Reihenfolge ist zufällig. Bei (a) gibt es dafür keine Garantie. Der Algorithmus für Methode (b) lautet:

Algorithmus 7.19 (Stochastic gradient descent für neuronale Netzwerke) Sei $\gamma > 0$ (genannt *learning rate*) und $w > 0$.
Sei $W_{jk}^{(i)[0]}$ zufällig gleichverteilt auf $[-w, w]$ und $v_j^{(i)[0]} := 0$, $r = 0$.
Wiederhole für $t = 1, 2, 3, \ldots, T \in \mathbb{N}$:

- Wähle eine zufällige Permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$. Für $i = \pi(1), \pi(2), \ldots, \pi(n)$, führe durch:
  1. Erhöhe r um 1.
  2. Führe Forward- und Back Propagation wie in Definition 7.18, Schritte (1),(2) mit der Beobachtung (X_i, Y_i) durch.
  3. Berechne

$$v^{(l)[t]} := v^{(l)[t-1]} - \frac{\gamma}{n} \cdot \frac{\partial C_i}{\partial v^{(l)}}, \qquad l = 1, \ldots, L,$$

$$W_{jm}^{(l)[t]} := W_{jm}^{(l)[t-1]} - \frac{\gamma}{n} \cdot \left\{ \frac{\partial C_i}{\partial W_{jm}^{(l)}} + 2\lambda \cdot W_{jm}^{(l)[t-1]} \right\}, \qquad l = 1, \ldots, L+1.$$

Sei $\tilde{\theta}^{[T \cdot n]}$ die Vektorisierung von $v^{(l)[T \cdot n]}$, $W^{(l)[T \cdot n]}$ gemäß Gl. (7.5). Dann entspricht $g_{\tilde{\theta}^{[Tn]}}$ dem neuronalen Netzwerk nach T vollständig durchlaufenen Trainingsepochen. Setze

- im Regressionsfall: $\hat{f}_{n,\lambda}^{FNN,sgd} := g_{\tilde{\theta}^{[Tn]}}$,
- im Klassifikationsfall: $\hat{f}_{n,\lambda}^{FNN,sgd} := \arg\max_{k \in \{1,\ldots,K\}} M_k(g_{\tilde{\theta}^{[Tn]}})$.

Da jede Beobachtung im Durchschnitt nur alle n Iterationsschritte in die Aktualisierung der Parameter eingeht und währenddessen $n-1$ andere Beobachtungen die Parameter in andere Richtungen verschieben, können spezifische Eigenschaften viel schwieriger in $\hat{f}_{n,\lambda}^{FNN,sgd}$

ausgeprägt werden. Daher ist der Algorithmus $\hat{f}_{n,\lambda}^{FNN,sgd}$ gegenüber $\hat{f}_{n,\lambda}^{FNN,grad}$ stärker vor Überanpassung an die Trainingsdaten geschützt.

7.4 Theoretische Resultate

Die statistische Theorie für neuronale Netzwerke ist Gegenstand aktueller Forschung, siehe zum Beispiel [2, 5, 19, 25]. In diesem Abschnitt geben wir einige statistische Resultate aus [25] für neuronale Netzwerke im Regressionsfall wieder.

7.4.1 Approximationsqualität neuronaler Netzwerke

Ist L, p fest gewählt, so erfordert eine Analyse von $R(\hat{f}_{n,\lambda}^{FNN})$ eine Aussage darüber, wie klein der Abstand $\sup_{x\in\mathcal{X}} |f^*(x) - f(x)|$ für geeignetes $f \in \mathcal{F}(L, p)$ werden kann, d. h. wie gut die unbekannte Bayes-Regel durch ein neuronales Netzwerk vorgegebener Größe approximiert werden kann. Diese Fragestellung ist auch von praktischem Interesse, denn sie gibt Auskunft über die nötige Größe von Netzwerken in Abhängigkeit der Dimension d der Trainingsdaten.

Die folgenden rein deterministischen Resultate sind entnommen aus [25, Lemma 3-10]. Wir betrachten hier neuronale Netzwerke mit Aktivierungsfunktion $\sigma(x) = \max\{x, 0\}$ und untersuchen, wie viele Layer und welche Dimensionen der Layer hinreichend sind, um eine vorgegebene Funktion $f^* : \mathcal{X} \to \mathbb{R}$ gut zu approximieren. Wir beschränken uns aus Gründen der Übersichtlichkeit auf $\mathcal{X} = [0, 1]^d$. Durch Reskalieren der Beobachtungen können alle hier dargestellten Ergebnisse auf kompakte $\mathcal{X} \subset \mathbb{R}^d$ erweitert werden.

Eine mindestens einmal differenzierbare Funktion $f^* : [0, 1]^d \to \mathbb{R}$ kann wie folgt zerlegt werden:

- f^* besitzt eine Taylor-Entwicklung. Jeder Term der Taylor-Entwicklung besteht aus Produkten $x_{i_1}^{\alpha_1} \cdot \ldots \cdot x_{i_d}^{\alpha_d}$ mit $\alpha_1, \ldots, \alpha_d \in \mathbb{N}_0$.
- Produkte zwischen r Variablen sind mehrmalige Anwendungen von Produkten zweier Variablen.

Der erste Schritt ist daher die Darstellung des Produkts zweier Variablen, d. h. die Funktion

$$\text{mult} : [0, 1]^2 \to [0, 1], \quad \text{mult}(x, y) = x \cdot y$$

mit einem neuronalen Netzwerk. Dieses Problem wird durch die nachstehenden zwei Lemmata beantwortet.

Lemma 7.20 Sei $g : \mathbb{R} \to \mathbb{R}$, $g(x) := x(1-x)$ und

$$T^k : [0,1] \to \mathbb{R}, \quad T^k(x) := \sigma(\frac{x}{2}) - \sigma(x - 2^{1-2k})$$

sowie $R^k : [0,1] \to \mathbb{R}$, $R^k(x) := T^k \circ T^{k-1} \circ \ldots \circ T^1$. Dann gilt

$$\forall x \in [0,1] : \quad |\sum_{k=1}^{m} R^k(x) - g(x)| \leq 2^{-m}.$$

Unter Nutzung der Polarisationsgleichung

$$g\left(\frac{x-y+1}{2}\right) - g\left(\frac{x+y}{2}\right) + \frac{x+y}{2} - \frac{1}{4} = x \cdot y$$

erlaubt dieses Resultat direkt eine Erweiterung auf die Funktion mult:

Lemma 7.21 Es gibt $\text{mult}_m \in \mathcal{F}(m+4, (2,6,6,\ldots,6,1)^T)$ mit

$$\forall x, y \in [0,1] : \quad |\text{mult}_m(x,y) - \text{mult}(x,y)| \leq 2^{-m}.$$

Durch Parallelisierung der Layer von mult_m kann nun auch ein Produkt zwischen r Variablen mit einem neuronalen Netzwerk mit nur geringfügig mehr Layern dargestellt werden:

Lemma 7.22 Es gibt $\text{mult}_m^d \in \mathcal{F}((m+5)\lceil \log_2(d) \rceil, (d, 6d, 6d, \ldots, 6d, 1)^T)$ und

$$\forall x = (x_1, \ldots, x_d)^T \in [0,1]^d : \qquad |\text{mult}_m^d(x) - \prod_{i=1}^{d} x_i| \leq 3^d 2^{-m}.$$

Hierbei ist $\lceil x \rceil = \min\{k \in \mathbb{N} : k \geq x\}$ die Aufrundenfunktion.

Ist $k \in \mathbb{N}$ und f^* k-mal stetig partiell differenzierbar, so lautet das Taylor-Polynom der Ordnung k an einer Stelle $a \in [0,1]^d$

$$P_a^{k-1} f(x) = \sum_{\alpha \in \mathbb{N}_0^d, 0 \leq \|\alpha\|_1 \leq k-1} (\partial^\alpha f)(a) \frac{(x-a)^\alpha}{\alpha!},$$

wobei $\partial^\alpha f^* := \partial_{x_1^{\alpha_1}} \ldots \partial_{x_d^{\alpha_d}}$, $\|\alpha\|_1 = \sum_{j=1}^d \alpha_j$, $\alpha! := \alpha_1! \cdot \ldots \cdot \alpha_d!$ und $v^\alpha := v_1^\alpha \cdot \ldots \cdot v_d^{\alpha_d}$ für $v \in \mathbb{R}^d$. In diesem Kontext wird α auch *Multiindex* genannt.

Ist $\sup_{\alpha\in\mathbb{N}_0^d,\|\alpha\|_1=k}\sup_{x\in[0,1]^d}|\partial^\alpha f^*|\leq K$ mit einer Konstanten $K\in\mathbb{R}$, so gilt nach dem Satz von Taylor für alle $x\in[0,1]^d$:

$$|f^*(x)-P_a^{k-1}f^*(x)|\leq\frac{K}{k!}|x-a|_\infty^k \tag{7.11}$$

Die Abschätzung mittels Gl. (7.11) ist zu grob, wenn nur *ein* Punkt $a\in[0,1]^d$ genutzt wird: Dann würden wir in vielen Fällen kein besseres Resultat als eine obere Schranke der Form $\frac{K}{k!}\cdot\frac{1}{2^k}$ erhalten, egal wie gut die Näherung von $P_a^{k-1}f^*(x)$ durch ein neuronales Netzwerk mittels Lemma 7.22 wäre. Zur besseren Ausnutzung von Gl. (7.11) führen wir daher ein Gitter

$$D(M):=\{a^{(r)}:=\left(\frac{r_j}{M}\right)_{j=1,\ldots,d}:r=(r_1,\ldots,r_d)\in\{0,1,\ldots,M\}^d\}\subset[0,1]^d$$

ein und approximieren f^* je nach Argument x immer mit dem Taylor-Polynom, welches zu dem Gitterpunkt gehört, der am nächsten an x liegt. Dies entspricht der Approximation von f^* durch

$$P^{k-1}f^*(x):=\sum_{a^{(r)}\in D(M)}P_{a^{(r)}}^k f^*(x)\cdot\prod_{j=1}^d\left(1-M\cdot|x_j-a_j^{(r)}|\right)^+,$$

wobei das Produkt $\prod_{j=1}^d\left(1-M\cdot|x_j-a_j^{(r)}|\right)^+$ gerade die Auswahl des am nächsten liegenden Gitterpunkts übernimmt.

Die Ermittlung eines neuronalen Netzwerks zur Darstellung von $P^{k-1}f^*$ erfolgt durch Finden geeigneter neuronaler Netzwerke für die Summanden $P_a^{k-1}f^*$ mit $a\in D(M)$, was im Folgenden kurz motiviert wird:

Für einen weiteren Multiindex $\gamma\in\mathbb{N}_0^d$ schreiben wir $\gamma\leq\alpha$, falls für alle $j\in\{1,\ldots,d\}$ gilt: $\gamma_j\leq\alpha_j$. Durch Ausmultiplizieren des Polynoms $P_a^{k-1}f^*$ erhalten wir die Darstellung

$$P_a^{k-1}f^*(x)=\sum_{\gamma\in\mathbb{N}_0^d:0\leq\|\gamma\|_1\leq k-1}c_\gamma\cdot x^\gamma$$

mit

$$c_\gamma:=\sum_{\alpha\in\mathbb{N}_0^d:\|\alpha\|_1\leq k-1,\gamma\leq\alpha}(\partial^\alpha f^*)(a)\cdot\frac{(-a)^{\alpha-\gamma}}{\gamma!(\alpha-\gamma)!}.$$

Die Terme x^γ können nun mittels Lemma 7.22 diskutiert werden. Die Resultate dieser Überlegungen fassen wir in Satz 7.23 zusammen:

Satz 7.23 Ist $f^*:[0,1]^d\to\mathbb{R}$ k-mal partiell differenzierbar und gilt

$$\sup_{\alpha\in\mathbb{N}_0^d,\|\alpha\|_1=k}\sup_{x\in[0,1]^d}|\partial^\alpha f^*|\leq K,$$

so gibt es für alle $m, N \in \mathbb{N}$ mit $N \geq \max\{(k+1)^d, K+1\}$ ein

$$\tilde{f} \in \mathcal{F}(L, (r, 12rN, \ldots, 12rN, 1)^T), \qquad L = 8 + (m+5)(1 + \lceil \log_2(d) \rceil),$$

so dass

$$\forall x \in [0,1]^d : \quad |f^*(x) - \tilde{f}(x)| \leq (2K+1)3^{d+1}N2^{-m} + K2^k N^{-k/d}. \tag{7.12}$$

Bemerkung 7.24

- Der Satz zeigt, dass jede mindestens einmal differenzierbare Funktion durch ein neuronales Netzwerk geeigneter Größe angenähert werden kann. Tatsächlich zeigt [25], dass dies auch für allgemeinere, Hölder-stetige Funktionen der Fall ist.
- Der erste Summand auf der rechten Seite von Gl. (7.12) entsteht durch Approximation von $P^{k-1}f^*$ durch ein neuronales Netzwerk, vgl. Lemma 7.22. Je höher m, desto genauer wird $P^{k-1}f^*$ durch das Netzwerk dargestellt. Der zweite Summand in Gl. (7.12) ist Resultat der Näherung von f^* durch $P^{k-1}f^*$. Die Güte dieser Näherung wird durch N bestimmt, welches die Anzahl der Gitterpunkte von $D(M)$ beschreibt (M wird maximal gewählt, so dass noch $(M+1)^d = \#D(M) \leq N$ gilt). Durch geeignet große Wahl m, N und deren Balancierung untereinander kann die rechte Seite von Gl. (7.12) beliebig klein werden.

7.4.2 Statistische Resultate

Die Theorie aus [25] trifft Aussagen über neuronale Netzwerke, die globale Minimierer eines im Gegensatz zu Gl. (7.3) leicht abgewandelten Funktionals sind. Durch Anwendung der Gradientenmethode wird in der Praxis aber oft nur ein lokales Minimum von Gl. (7.3) gefunden. Güteaussagen über lokale Minimierer sind ein wesentlich schwierigeres Problem, das Gegenstand aktueller Forschung ist. In diesem Sinne gibt es hier noch eine Lücke zwischen Theorie und Praxis.

Die Zielfunktion aus Gl. (7.3) ähnelt mit dem $\|\cdot\|_2^2$-Bestrafungsterm $J(g)$ der Ridge-Regression aus Abschn. 2.3. In Hinsicht auf die dortigen Resultate ist klar, dass die vielen Parameter von $g \in \mathcal{F}(L, p)$ immer noch eine Überanpassung der Daten ermöglichen. Es ist daher nicht zu erwarten, dass der Minimierer $\hat{f}_{n,\lambda}^{FNN}$ für hochdimensionale Trainingsdaten kleine Generalisierungsfehler erzielt. In der Praxis wird durch das Einfügen von heuristisch motivierten Zwischenoperationen innerhalb der Layer (z. B. durch sogenannte *max-pooling layer*) des Netzwerks erreicht, dass die Überanpassung verringert wird und nur wenige Gewichte von null verschieden bzw. relevant für die Werte des Algorithmus sind. Um dies in der Theorie möglichst einfach nachzubilden, ändert [25] die Menge der im Optimierungsproblem betrachteten neuronalen Netzwerke ab. Wir definieren:

Definition 7.25 (Dünn besetzte neuronale Netzwerke)
Für $s \in \mathbb{N}$, $F \in \mathbb{R}$ und mit der Bezeichnung $\|v\|_0 := \#\{j \in \{1, \dots, d\} : v_j \neq 0\}$ für $v \in \mathbb{R}^d$ sei

$$\mathcal{F}(L, p, s, F) := \{f \in \mathcal{F}(L, p) : \sum_{l=1}^{L+1} \|W^{(l)}\|_0 + \sum_{l=1}^{L} |v^{(l)}|_0 \leq s, |f|_\infty \leq F\}$$

die Menge der *dünn besetzten neuronalen Netzwerke* mit höchstens s von null verschiedenen Gewichtsmatrizen und Verschiebungsvektoren und maximalem Funktionswert F. ♦

Im Englischen verwendet man den Begriff *sparse* für dünn besetzte Strukturen. Das Optimierungsproblem aus Definition 7.7 über beliebige neuronale Netzwerke wird nun abgewandelt zu folgender Definition:

Definition 7.26 (Neuronaler Netzwerkalgorithmus 2 für Regression)
Sei $L \in \mathbb{N}$, $p \in \mathbb{N}_0^{L+2}$ und $s \in \mathbb{N}$, $F \in \mathbb{R}$, $F > 0$ fest gewählt. Dann heißt

$$\hat{f}_{n,\lambda}^{FNN,sparse} :\in \operatorname*{arg\,min}_{g \in \mathcal{F}(L,p,s,F)} \hat{R}_n(g), \qquad \hat{R}_n(g) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - g(X_i))^2 \tag{7.13}$$

neuronaler Netzwerkalgorithmus für Regression mit *dünn besetztem Netzwerk.* ♦

Bemerkungen

- In Hinsicht auf Definition 7.7 entspricht die Optimierung über $\mathcal{F}(L, p, s, F)$ anschaulich der Änderung von $\|\cdot\|_2^2$ zu $\|\cdot\|_1$ im Bestrafungsterm J, vgl. Lemma 2.39.
- Wie bereits oben erklärt, wird $\hat{f}_{n,\lambda}^{FNN,sparse}$ nicht in der Praxis verwendet, sondern soll nur die in der Praxis verwendeten Netzwerke theoretisch nachbilden.

Tatsächlich gibt es auch in Satz 7.23 eine Aussage zur Anzahl der benötigten Nicht-Null-Komponenten des f^* approximierenden Netzwerks $\tilde{f}$. Dort gilt

$$s \leq 94 d^2 (k+1)^{2d} N (m+6)(1 + \lceil \log_2(d) \rceil). \tag{7.14}$$

Für beliebige Funktionen $f^* : [0,1]^d \to \mathbb{R}$ sind die oberen Schranken für $\tilde{f}$ aus Gl. (7.12) und (7.14) zu grob und erlauben keine besseren Aussagen für das Excess Bayes Risk als die Standardschätzer der nichtparametrischen Statistik in Kap. 6. Grund ist, dass der zweite Summand in Gl. (7.12) bei hoher Dimension d nur klein wird, falls eine starke Glattheit (k groß) von f^* vorliegt. In der Praxis entspricht die Forderung nach starker Glattheit jedoch anschaulich einer sich (fast) linear verhaltenden Funktion f^*, was bei komplexeren Zusammenhängen verletzt ist. Im Gegensatz zu den Schätzern aus Kap. 6 ist es mit neuronalen Netzwerken aber einfacher, strukturelle Zerlegungen von f^* zu formulieren und in der Schranke (7.12) auszunutzen. Als Beispiel betrachten wir das sogenannte additive Modell,

bei welchem angenommen wird, dass $f^*(x)$ in eine Summe zerfällt und jeder Summand nur noch von einer Komponente von x abhängt.

Beispiel 7.27 (Additives Modell und dessen Zerlegung) Es sei $K > 0$. Es sei $f^*(x) = \sum_{j=1}^{d} g_{0j}(x_j)$ mit stetig differenzierbaren Funktionen $g_{0j} : \mathbb{R} \to [-K, K]$. Dann gilt

$$f^* = g_1 \circ g_0,$$

wobei

$$g_0 : [0,1]^d \to \mathbb{R}^d, \quad g_0(x) = (g_{01}(x_1), \ldots, g_{0d}(x_d))^T,$$
$$g_1 : \mathbb{R}^d \to \mathbb{R}, \quad g_1(y) = \sum_{j=1}^{d} y_j.$$

Die einzelnen Komponenten von g_0 sind zwar nur einmal differenzierbar, aber haben auch nur den Definitionsbereich $\mathbb{R}$. Dahingegen besitzt g_1 zwar den Definitionsbereich $\mathbb{R}^d$, ist aber als Summenfunktion unendlich oft differenzierbar.

Anwendung von Satz 7.23 auf die Komponenten von g_0 mit $\tilde{d} = 1$, $k = 1$ und auf g_1 mit $\tilde{d} = d$, „$k = \infty$“ liefert neuronale Netzwerke $\tilde{g}_0 = (\tilde{g}_{01}, \ldots, \tilde{g}_{0d})^T$ und $\tilde{g}_1$ mit guten Approximationsraten, die sich auf $\tilde{f} := \tilde{g}_1 \circ \tilde{g}_0$ übertragen.

Motiviert durch dieses Beispiel fordern wir allgemeiner folgende strukturelle Zerlegung für f^*:

Modellannahme 7.28 (Zerlegung von f^*) Gegeben seien Trainingsdaten (X_i, Y_i) i. i. d. eines Regressionsproblems mit $X_i \in [0,1]^d$ f. s. Es seien ε_i standardnormalverteilt, d. h. $\varepsilon_i \sim N(0,1)$.

Es sei $K \in \mathbb{R}$, $q \in \mathbb{N}_0$. Für $l = 0, \ldots, q+1$ gebe es $a_l, b_l \in \mathbb{R}$ mit $|a_l|, |b_l| \leq K$ und $d_l \in \mathbb{N}$ (mit $d_0 = d$ und $d_{q+1} = 1$) sowie Funktionen $g_l : [a_l, b_l]^{d_l} \to [a_{l+1}, b_{l+1}]^{d_{l+1}}$, so dass die Bayes-Regel $f^* : [0,1]^d \to \mathbb{R}$ erfüllt:

$$f^* = g_q \circ q_{q-1} \circ \ldots \circ g_1 \circ g_0 \tag{7.15}$$

Für $l = 0, \ldots, q$ hänge die Funktion $g_l = (g_{lj})_{j=1,\ldots,d_{l+1}}^T$ nur von $t_l \in \{1, \ldots, d_l\}$ Variablen ab und sei k_l-mal stetig differenzierbar mit

$$\sup_{j=1,\ldots,d} \sup_{\alpha \in \mathbb{N}_0^d, \|\alpha\|_1 = k} \sup_{x \in [a_l, b_l]^{d_l}} |\partial^\alpha g_{lj}(x)| \leq K.$$

Bemerkungen

Die Funktionen g_i dürfen von beliebiger Struktur sein, müssen also nicht notwendig neuronale Netzwerke sein. Die fehlende Eindeutigkeit der Darstellung in Gl. (7.15) ist nicht

schädlich, da es nicht unser Ziel ist, die einzelnen g_i zu schätzen, sondern die Zerlegung als Ganzes für theoretische Resultate zu nutzen. In [25, Theorem 1 und Corollary 1] wird folgender Satz gezeigt:

Satz 7.29 Es gelte die Modellannahme 7.28. Für $l = 0, \ldots, q$ sei $k_l^* := k_l \cdot \prod_{\nu=l+1}^{q} \min\{k_\nu, 1\}$ und

$$\phi_n := \max_{l=0,\ldots,q} n^{-\frac{2k_l^*}{2k_l^*+t_l}}.$$

Es gebe eine Konstante $C > 0$, sodass die Ausdrücke $L = L(n)$, $p = p(n)$ und $s = s(n)$ in Definition 7.26 erfüllen:

(i) Die zur Schätzung verwendeten neuronalen Netzwerke erlauben Funktionswerte, die mindestens so groß wie die maximalen Funktionswerte der Bayes-Regel f^* sind: $F \geq \max\{K, 1\}$.
(ii) Die Anzahl der Layer geht mit Rate $\log(n)$ in n gegen unendlich: $\sum_{l=0}^{q} \log_2(4t_l) \log_2(n) \leq L \leq C \log(n)$
(iii) Die Größe der Layer muss mindestens mit Rate $n\phi_n$ in n gegen unendlich gehen: $n\phi_n \leq C \min_{i=1,\ldots,L} p_i$.
(iv) Die Anzahl der Nicht-Null-Einträge der Gewichtsmatrizen und Biasvektoren muss mit Rate $n\phi_n \log(n)$ in n gegen unendlich gehen: $\frac{1}{C} n\phi_n \log(n) \leq s \leq Cn\phi_n \log(n)$.

Dann gibt es eine Konstante $C' = C'(q, d_0, \ldots, d_{q+1}, t_0, \ldots, t_q, k_0, \ldots, k_q, F) > 0$ mit

$$\mathbb{E}R\left(\hat{f}_{n,\lambda}^{FNN,sparse}\right) - R(f^*) \leq C' \phi_n \log(n)^3.$$

Bemerkungen In der Definition der Konvergenzrate ϕ_n findet ein *tradeoff* statt: Falls t_l groß ist, kann dies durch entsprechend großes β_l kompensiert werden. Für bestimmte Strukturen von f^* kann damit auch eine von d unabhängige Konvergenzrate in n erreicht werden, vgl. das nachfolgende Beispiel 7.30.

Für das additive Modell liefert die Anwendung des Satzes das folgende Resultat:

Beispiel 7.30 Im additiven Modell aus Beispiel 7.27 gilt:

$$f^* = g_1 \circ g_0$$

Aufgrund der Differenzierbarkeit von g_{0j}, $j = 1, \ldots, d$ ist $g_0 : [0, 1]^d \to [-K, K]^d$ für K groß genug, und wir können $g_1 : [-K, K]^d \to \mathbb{R}$ wählen. Es gilt daher $d_0 = d$, $d_1 = d$, $d_2 = 1$.

Die einzelnen Komponenten von g_0 sind differenzierbar und hängen jeweils von einer Komponente ab, daher ist $t_0 = 1$, $k_0 = 1$.

g_1 ist als Summe von d Termen unendlich oft differenzierbar, daher gilt $k_1 > 0$ beliebig groß und $t_1 = d$.

Es folgt $k_0^* = 1$. Der Wert $k_1^* > 0$ kann beliebig groß gewählt werden, daher ist

$$\min_{l=0,1} \frac{2k_l^*}{2k_l^* + t_l} = \min\left\{\frac{2}{3}, \frac{2k_1^*}{2k_1^* + d}\right\} = \frac{2}{3}$$

und $\phi_n = n^{-2/3}$. Sind die restlichen Annahmen (i)–(iv) des Satzes 7.29 erfüllt, so gilt mit einer Konstanten $C' > 0$:

$$\mathbb{E}R\left(\hat{f}_{n,\lambda}^{FNN,sparse}\right) - R(f^*) \leq C' n^{-2/3} \log(n)^3$$

7.5 Ausblick: Faltende neuronale Netzwerke

Für hochdimensionale Trainingsdaten X_i sind einige ebenfalls hochdimensionale Layer notwendig, um f^* adäquat zu approximieren. Die Anzahl der Parameter eines neuronalen Netzwerks $g \in \mathcal{F}(L, p)$ beträgt

$$\sum_{l=0}^{L} p_l + \sum_{l=0}^{L} p_l p_{l+1}$$

und kann dementsprechend sehr hoch sein. Die zur Schätzung verwendete Zielfunktion $g \mapsto \{\hat{R}_n(g) + \lambda \cdot J(g)\}$ hat dann eine große Menge von lokalen Minima, und Gradientenverfahren liefern vor allem für hochdimensionale Trainingsdaten nur suboptimale Lösungen.

Beim neuronalen Netzwerk aus Definition 7.3 werden keine Einschränkungen an die Gewichtsmatrizen $W^{(l)}$ formuliert, anschaulich wird jedes Neuron aus Layer l mit jedem Neuron aus Layer $l-1$ verbunden. Solche Netzwerke heißen *fully connected neural networks*. Wir haben bereits an den theoretischen Resultaten des vorherigen Abschnitts gesehen (vgl. s in Gl. (7.14) und Satz 7.29), dass auch für die Beschreibung von komplexen Funktionen f^* nicht alle Einträge der Gewichtsmatrizen genutzt werden müssen. Aufgrund der hohen Anzahl von Parametern kann man aber nicht erwarten, dass ein Gradientenverfahren die richtigen Nicht-Null-Parameter findet.

Je nach Anwendung wurden daher viele neue Modelle definiert, welche ebenfalls als neuronale Netzwerke bezeichnet werden, sich aber teilweise wesentlich von der Struktur in Definition 7.3 unterscheiden.

Eine mögliche Anwendung ist die Klassifikation von Bildern. Dies schließt folgende Bereiche ein:

- Entscheide, ob auf einem Bild bestimmte Objekte dargestellt sind oder nicht,
- entscheide auf Basis eines Bilds, welche Aktion als Nächstes durchgeführt werden soll (z. B. bei Computerspielen).

In diesem Fall entsprechen die Trainingsdaten X_i hochaufgelösten Bildern B_i mit einer Höhe von h Pixeln und einer Breite von b Pixeln. Betrachten wir nur Graustufenbilder, so ist $B_i \in \mathbb{R}^{h\times b}$. Mittels der Vektorisierungsfunktion aus Gl. (7.4) können Bilder unmittelbar als Vektor aufgefasst werden mittels $X_i = \text{vec}(B_i) \in \mathbb{R}^{h\cdot b}$. Anschaulich entspricht dies einer spaltenweisen Aneinanderreihung der Pixel des Bilds:

$$B_i = \begin{array}{|c|c|c|}\hline 1 & 4 & 7 \\ \hline 2 & 5 & 8 \\ \hline 3 & 6 & 9 \\ \hline \end{array} \quad \rightarrow \quad X_i = \text{vec}(B_i) = (1,2,3,4,5,6,7,8,9)^T \in \mathbb{R}^9$$

In den folgenden Definitionen interpretieren wir $B \in \mathbb{R}^{h\times b}$ als ein Bild in Graustufen. Wir verwenden die Konvention $B_{x,y} = 0$ für $x \notin \{0,\ldots,h\}$ bzw. $y \notin \{0,\ldots,b\}$, d. h., Pixel „außerhalb" des Bildes haben den Wert 0. Erfahrungen aus der Bildverarbeitung zeigen, dass die Extraktion und Reduktion der Informationen eines Bilds besonders gut mittels Filtermatrizen und durch Verkleinern erfolgen kann. Durch Anwendung einer Filtermatrix auf ein Bild können beispielsweise Ränder von Strukturen im Bild erkannt oder eine Verwaschung bzw. Schärfung des Bilds erreicht werden.

Definition 7.31 (Faltungsmatrix und diskrete Faltung)
Eine Matrix $K \in \mathbb{R}^{n\times n}$ heißt *Faltungsmatrix,* falls $n \in \mathbb{N}$, $n \geq 3$ ungerade ist. Ist $b, h \in \mathbb{N}$ mit $b, h \geq n$ und $a := \frac{n+1}{2}$, so heißt die Abbildung

$$\tilde{\Phi}_K^{conv} : \mathbb{R}^{h\times b} \to \mathbb{R}^{h\times b}, \qquad \tilde{\Phi}_K^{conv}(B)_{x,y} := \sum_{i=1}^{n}\sum_{j=1}^{n} B_{x-i+a,y-j+a}\cdot K_{ij}$$

diskrete Faltung mit Faltungsmatrix K.

Die zugehörige Abbildung für vektorisierte Bilder bezeichnen wir mit $\Phi_K^{conv} := \text{vec} \circ \tilde{\Phi}_K^{conv} \circ \text{vec}^{-1}$. ♦

Bereits für $n = 3$ können mittels der diskreten Faltung wesentliche Informationen aus einem Bild B extrahiert werden. Typische Matrizen zeigt folgendes Beispiel:

Beispiel 7.32 (Faltungsmatrizen) Die folgenden Matrizen $K_1, K_2 \in \mathbb{R}^{3\times 3}$ entsprechen einem Kantenfilter (K_1) und einem Glättungsfilter (K_2):

$$K_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \qquad K_2 = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Für die Reduktion der Dimensionen eines Bilds und der darin enthaltenen Informationen gibt es viele Möglichkeiten. In neuronalen Netzwerken wird zum Beispiel *max pooling* verwendet, bei welchem je $a \times a$-Quadrate von Pixeln zu einem Pixel zusammengefasst werden:

Definition 7.33 (Max-pooling)
Sei $a \in \mathbb{N}$, $a \geq 2$ und $h, b \in \mathbb{N}$. Dann heißt

$$\tilde{\Phi}_a^{pool} : \mathbb{R}^{h \times b} \to \mathbb{R}^{\lceil \frac{h}{a} \rceil \times \lceil \frac{b}{a} \rceil},\ \tilde{\Phi}_K^{pool}(B)_{x,y} := \max\{B_{a \cdot x+i-1, a \cdot y+j-1} : i, j \in \{0, \ldots, a-1\}\}$$

max-pooling-Abbildung.

Die zugehörige Abbildung für vektorisierte Bilder bezeichnen wir mit $\Phi_a^{pool} := \text{vec} \circ \tilde{\Phi}_a^{pool} \circ \text{vec}^{-1}$. ♦

Bei neuronalen Netzwerken für die Klassifikation von Bildern verwendet man nun $d = b \cdot h$.

Definition 7.34 (Verallgemeinertes neuronales Netzwerk)
Sei $L \in \mathbb{N}$, $p = (p_0, \ldots, p_{L+1})^T \in \mathbb{R}^{L+2}$ und $\tau = (\tau_1, \ldots, \tau_{L+1})^T \in \{full, conv, pool\}^{L+1}$. Eine Abbildung $g : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}}$ heißt *verallgemeinertes neuronales Netzwerk* mit Netzwerkarchitektur (L, p, τ), falls

$$g = \Phi_{L+1} \circ \Phi_L \circ \ldots \circ \Phi_1,$$

wobei $\Phi_l : \mathbb{R}^{p_{l-1}} \to \mathbb{R}^{p_l}$, $l = 1, \ldots, L+1$ (die *Layer*) je nach Wert τ_l einer der folgenden Funktionen entsprechen:

- *fully connected layer*: $\Phi_l = \Phi_{v^{(l)}, W^{(l)}}^{full}$, wobei $\Phi_{v^{(l)}, W^{(l)}}^{full}(x) = v^{(l)} + W^{(l)} \cdot x$ mit $v^{(l)} \in \mathbb{R}^{p_l}$, $W^{(l)} \in \mathbb{R}^{p_l \times p_{l-1}}$,
- *convolution layer*: $\Phi_l = \Phi_{K^{(l)}}^{conv}$ mit $K^{(l)} \in \mathbb{R}^{n_l \times n_l}$, $n_l \in \mathbb{N}$,
- *max pooling layer*: $\Phi_l = \Phi_{a^{(l)}}^{pool}$ mit $a^{(l)} \in \mathbb{N}$, $a^{(l)} \geq 2$.

Enthält das Netzwerk mindestens einen *convolutional layer,* so nennt man g auch *faltendes neuronales Netzwerk.* ♦

Bemerkungen

- Es gibt noch viele weitere heuristisch motivierte Möglichkeiten für die einzelnen Layer Φ_l.
- Die Parameter des Netzwerks ergeben sich aus der Zusammenfassung der Parameter $v^{(l)}$, $W^{(l)}$ bzw. $K^{(l)}$ bzw. $a^{(l)}$ der jeweils verwendeten Layer. Die Anpassung des neuronalen Netzwerks an Trainingsdaten erfolgt weiterhin mit einer Gradientenmethode,

deren mathematische Komplexität aufgrund der verschiedenen Typen von Layern nun zunimmt.

- In der Praxis werden die einzelnen Layer von neuronalen Netzwerken noch komplexer aufgebaut. So wird beispielsweise in einem *convolution layer* nicht nur *eine* Filtermatrix auf die Eingangsdaten angewandt, sondern $r \in \mathbb{N}$ verschiedene Filtermatrizen. Formal entspricht dies der Verwendung von

$$\Phi_l = (\Phi^{conv}_{K_1^{(l)}}, \ldots, \Phi^{conv}_{K_r^{(l)}}) \tag{7.16}$$

  mit $K_1^{(l)}, \ldots, K_r^{(l)} \in \mathbb{R}^{n_l \times n_l}$ anstelle der in Definition 7.34 angegebenen Variante.
- Im Gegensatz zu einem *fully connected layer* mit $p_l(p_{l-1}+1)$ Parametern werden durch einen *convolution layer* nur n_l^2 Parameter eingeführt, wobei oft sogar nur $n_l \in \{3, 5\}$ gilt.
- Die Konzeption eines Netzwerks wird grafisch durch Angabe der verschiedenen Layer veranschaulicht. Ist ein konkretes Problem vorgegeben, so kann die damit verbundene Anschauung in die Modellierung des verwendeten neuronalen Netzwerks einfließen. Beispielsweise empfiehlt es sich, bei der Klassifikation von Bildern zunächst wesentliche strukturelle Eigenschaften eines Bilds mittels eines *convolution layers* im Stile von Gl. (7.16) zu extrahieren und diese dann weiterzuverarbeiten.
- Die Abbildung $\Phi_K^{falt} : \mathbb{R}^{h \cdot b} \to \mathbb{R}^{h \cdot b}$ ist linear, besitzt also eine Darstellung der Form $\Phi_K^{falt}(x) = W \cdot x$ mit geeignetem dünn besetztem $W \in \mathbb{R}^{(h \cdot b) \times (h \cdot b)}$. In diesem Sinne kann ein *convolution layer* auch als *fully connected layer* mit dünn besetztem $W^{(l)}$ betrachtet werden.

Eigenschaften solcher verallgemeinerten neuronalen Netzwerke werden aktuell in der Statistik erforscht, es gibt aber noch keine belastbaren Resultate.

Reinforcement Learning/Bestärkendes Lernen

8

Inhaltsverzeichnis

Typische Beispiele für die Anwendung des Reinforcement Learning sind:

a) Finde (gute) Spielstrategien für Spiele wie 4 Gewinnt, Tic-Tac-Toe, Schach, Go,
b) finde den schnellsten Weg durch ein Labyrinth,
c) lasse Roboter einfache Tätigkeiten selbst erlernen, z. B. das Ablegen von Dingen in die richtigen Container oder das Balancieren eines Stabs in vertikaler Ausrichtung,
d) entwickle einen Algorithmus für Autos, so dass diese auch ohne Fahrer von einem vorgegebenen Ort zu einem anderen Ort fahren, ohne Unfälle zu verursachen (autonomes Fahren).

Die Problemstellungen in den Beispielen haben Folgendes gemeinsam:

- Die gegebenen Systeme können sich in einer Menge von Zuständen befinden.
  Beispiel a) alle verschiedenen Konfigurationen des betrachteten Spiels, b) alle Orte, an welchen man sich im Labyrinth befinden kann, c) alle verschiedenen Zustände des Stabs, d) alle möglichen Sensoreneinstellungen des Autos.
- Es gibt mindestens einen günstigen Zustand, welcher erreicht und evtl. beibehalten werden soll.
  Beispiel a) gewinnen, b) Ausgang des Labyrinths erreicht, c) Stock wird vertikal balanciert, und dieser Zustand wird beibehalten, d) Zielort wird erreicht und auf dem Weg geschah kein Unfall.

S. Richter, *Statistisches und maschinelles Lernen,*
https://doi.org/10.1007/978-3-662-59354-7_8

- Regelmäßig muss basierend auf dem aktuellen Zustand eine Aktion aus einer Menge von Aktionen ausgewählt werden, um damit einen neuen Teilzustand zu erreichen.
  Beispiel a) ein Spielzug, b) einen Meter Weg in eine bestimmte Richtung gehen, c) bewege den Roboterarm 1mm nach oben/unten usw., d) beschleunigen, bremsen usw.
- Ziel ist die Ermittlung einer möglichst guten Strategie, welche auf Basis des aktuellen Zustands des Systems eine Aktion auswählt, durch welche man sich den günstigen Zuständen annähert.
- Im Allgemeinen gibt es keine direkte Information, ob die erreichten Teilzustände oder die gewählten Aktionen erfolgsversprechend sind; erst wenn man sich in günstigen oder ungünstigen Zuständen befindet, kann eine Bewertung erfolgen. Die Identifizierung der günstigen/ungünstigen Zustände ist recht einfach.

Beim Reinforcement Learning geht man von diesem allgemeinen Setting aus. Es wird als eigenes Teilgebiet neben Unsupervised und Supervised Learning gesehen, die theoretischen Grundlagen liegen in der dynamischen Programmierung (vgl. zum Beispiel [23]). In Abschn. 8.4 werden wir sehen, inwiefern Algorithmen des Supervised Learnings beim Reinforcement Learning verwendet werden können.

8.1 Die optimale Strategie

Die Abfolge der Zustände und der getätigten Aktionen kann mathematisch mittels eines Markov-Entscheidungsprozesses formalisiert werden. Der Einfachheit halber beschränken wir uns im Folgenden auf endliche Zustands- und Aktionsmengen.

Definition 8.1
Ein $(\mathcal{S}, A, P)$-*Markov-Entscheidungsprozess* $(S_t, A_t)_{t\in\mathbb{N}_0}$ ist eine Folge von Zufallsvariablen mit Werten in $\mathcal{S} \times A$ auf einem Wahrscheinlichkeitsraum $(\Omega, \mathcal{A}, \mathbb{P})$, wobei

- $\mathcal{S} \subset \mathbb{R}^d$ eine endliche Menge von Zuständen (der *Zustandsraum*) ist,
- A eine endliche Menge von Aktionen (der *Aktionsraum*) ist,
- $P = (P(s'|s, a))_{s,s'\in\mathcal{S},a\in A}$ eine Menge von Übergangswahrscheinlichkeiten ist. Es gelte für alle $t \in \mathbb{N}_0$:

$$\begin{aligned}&\mathbb{P}(S_{t+1} = s'|S_t = s, A_t = a, S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}..., S_0 = s_0, A_0 = a_0)\\&= \mathbb{P}(S_{t+1} = s'|S_t = s, A_t = a) = P(s'|s, a)\end{aligned} \tag{8.1}$$

♦

Bemerkung 8.2

- Die Größe $P(s'|s, a)$ gibt an, mit welcher Wahrscheinlichkeit $S_{t+1} = s'$ gilt, wenn der aktuelle Zustand $S_t = s$ und die ausgewählte Aktion $A_t = a$ ist. Diese Wahrscheinlichkeit ist üblicherweise vom betrachteten System vorgegeben, aber oft unbekannt. Der Namensteil „Markov“ drückt die Annahme aus, dass die Wahrscheinlichkeit nur vom aktuellen Zustand und der aktuellen Aktion abhängt, nicht jedoch von den vorherigen Zuständen und Aktionen, vgl. Gl. (8.1). Dadurch wird eine Vereinfachung der folgenden Theorie erreicht.
- Wahrscheinlichkeitstheoretisch betrachtet ist die Verteilung eines Markov-Entscheidungsprozesses durch Angabe der Übergangswahrscheinlichkeiten P noch nicht vollständig definiert. Es fehlt sowohl eine Beschreibung der Anfangsverteilung von (S_0, A_0) als auch eine Aussage über die Verteilung von A_{t+1} (der nächsten Aktion) gegeben den Zustand S_{t+1}. Im Gegensatz zu S_{t+1} ist die nächste Aktion jedoch von uns auszuwählen und nicht vom System vorgegeben. Im Folgenden ist unser Ziel daher eine sinnvolle Modellierung von A_{t+1}.

Die Bewertung der Zustände erfolgt mittels der sogenannten *reward*-Funktion. Um möglichst allgemeine Arten von Belohnungen beschreiben zu können, hängt diese nicht nur von dem aktuellen Zustand s ab, sondern beschreibt die erzielte Belohnung beim Übergang von Zustand s in Zustand s' mittels Aktion a.

Definition 8.3 Eine Abbildung $R : \mathcal{S} \times A \times \mathcal{S} \to \mathbb{R}$, $(s, a, s') \mapsto R(s, a, s')$ heißt *Belohnung/reward.* ♦

Wir geben nun zwei Beispiele, welche sich mit obiger Theorie einfach beschreiben lassen.

Beispiel 8.4 (Tic-Tac-Toe) Wir betrachten das Tic-Tac-Toe-Spiel, bei welchem zu Beginn ein leeres 3×3-Feld vorliegt. Zwei Spieler sind abwechselnd an der Reihe und füllen ein Feld mit einem Kreis $\circ$ (Spieler 1) oder mit einem Kreuz $\times$ (Spieler 2) aus. Spieler 2 beginnt. Erzeugt ein Spieler durch seinen Zug drei gleiche Symbole in einer Spalte, einer Diagonale oder einer Zeile, gewinnt dieser. Ist das Spielfeld ausgefüllt und kein Spieler hat gewonnen, endet das Spiel unentschieden.

Hier kann $\mathcal{S} \subset \{-1, 0, 1\}^9$ gewählt werden, wobei beispielsweise der Zustand $s = (-1, -1, -1, 0, 0, 1, 0, 1, 0) \in \mathcal{S}$ das Spielfeld

×		
×		○
×	○	

beschreibt. Hierbei steht -1 für ‚×‘, 1 für ‚○‘ und 0 für ein leeres Feld, und die Matrix des Spielfelds wird spaltenweise in den Vektor s geschrieben.

Die Menge der Aktionen ist $A = \{1, \ldots, 9\}$, wobei die Kästchen entsprechend der Tabelle

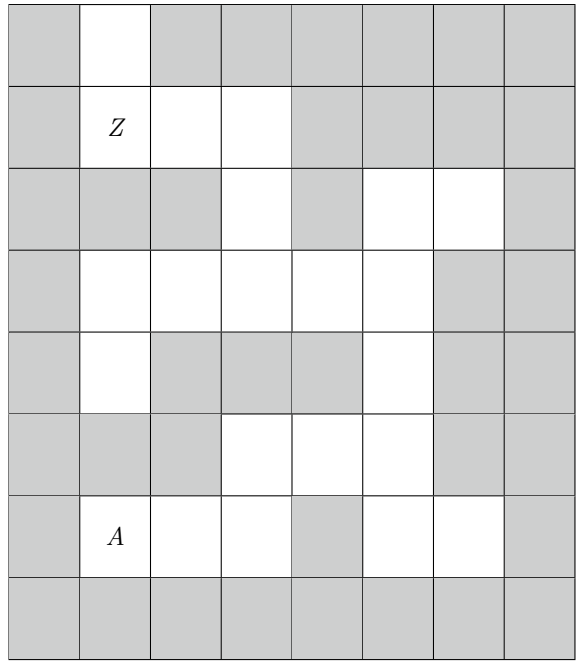
1	4	7
2	5	8
3	6	9

nummeriert sind.

Gegeben ein Spielfeld $s \in \mathcal{S}$, und eine Aktion $a \in A$ ist der nächste Zustand $s' \in \mathcal{S}$ aus Sicht von Spieler 1 zufällig, denn zwar ist der eigene Zug festgelegt (der zu einem nicht relevanten Zwischenzustand führt), der Zug von Spieler 2 aber nicht. Erst *nach* dem Zug von Spieler 2 liegt der nächste Zustand $s' \in \mathcal{S}$ vor, bei welchem Spieler 1 wieder agieren kann. Daher gibt es hier eine (unbekannte) Menge von Übergangs-Wahrscheinlichkeiten $P = (P(s'|s, a))_{s,s' \in \mathcal{S}, a \in A}$. Es ist zu beachten, dass wir durch diese Modellierung formal annehmen, dass der Spieler 2 immer nach dem gleichen Schema agiert. Starten wir also ein neues Spiel und führen wir so zweimal dasselbe Spielfeld herbei, so gehen wir davon aus, dass Spieler 2 jeweils mit den gleichen Wahrscheinlichkeiten auf die noch freien Felder setzt.

Die Funktion R ist frei wählbar und wird beispielsweise von außen (vom sogenannten Interpreter) vorgegeben. Eine mögliche Wahl ist die folgende: Jedes Tripel (s, a, s'), bei welchem s' sicher einen Zustand beschreibt, der den Gewinn von Spieler 1 oder von Spieler 2 bedeutet, führt zu bekannten Werten $R(s, a, s') = -1$ (verloren) oder $R(s, a, s') = 1$ (gewonnen). Ein unvollendetes Spiel s' erfährt zum Beispiel die Belohnung $R(s, a, s') = 0$ und ein Unentschieden $R(s, a, s') = 0{,}5$. Die letzte Setzung führt dazu, dass die später ermittelte Strategie auch ein Unentschieden als akzeptables Ergebnis sieht. Die Wahl $R(s, a, s') = 0$ oder sogar $R(s, a, s') = -1$ für „unentschieden“ würde zu einer Strategie führen, welche in jedem Fall versucht zu gewinnen. Für dieses kleine Beispiel ist diese Wahl nicht von praktischer Relevanz: Aufgrund der kleinen Dimensionen von $\mathcal{S}$ und A liefert der später vorgestellte Algorithmus in jedem Fall eine Strategie, welche nicht mehr verlieren wird. Sind die betrachteten Systeme aber größer, wie zum Beispiel bei ähnlich gelagerten Spielen wie Schach, erhält man in jedem Fall nur eine Approximation der bestmöglichen Strategie. Dann sind solche Überlegungen von großer Wichtigkeit, damit die zwanghafte Einstellung, gewinnen zu wollen, nicht zu schlechten Stellungen führt, die im Nachhinein nicht mehr korrigiert werden können und letztlich doch zur Niederlage führen.

Beispiel 8.5 (Labyrinth) Gegeben ist ein 8×8-Labyrinth der folgenden Form:



Der Startpunkt ist durch A markiert, der Zielpunkt durch Z. Die grau hinterlegten Felder symbolisieren Wände, die nicht betreten werden können. Von A beginnend soll durch Schritte der Länge 1 in eine beliebige Richtung das Ziel Z erreicht werden.

Hier kann $\mathcal{S} = \{(x_1, x_2) : x_1, x_2 \in \{1, ..., 8\}\}$ gewählt werden, wobei ein Zustand $s = (x_1, x_2) \in \mathcal{S}$ die Position der Person im Labyrinth beschreibt. Hierbei legen wir ein Koordinatensystem auf das Labyrinth, so dass der Punkt ganz unten links dem Punkt $(1, 1) \in \mathcal{S}$ und der Punkt ganz oben rechts $(8, 8) \in \mathcal{S}$ entspricht. Natürlich könnte $\mathcal{S}$ hier auch sparsamer gewählt werden, da viele Zustände in der Praxis nicht erreicht werden können.

Die Menge der Aktionen ist $A = \{1, 2, 3, 4\}$, wobei wir die Zahlen wie folgt identifizieren: $1 =$‚gehe $\uparrow$‘, $2 =$‚gehe $\rightarrow$‘, $3 =$‚gehe $\downarrow$‘, $4 =$‚gehe $\leftarrow$‘. Gegeben eine Position $s \in \mathcal{S}$, und eine Aktion $a \in A$ ist der nächste Zustand $s' \in \mathcal{S}$ hier deterministisch, denn dieser ist aufgrund der gewählten Richtung eindeutig bestimmt. Daher sind die Übergangswahrscheinlichkeiten $P = (P(s'|s, a))_{s,s' \in \mathcal{S}, a \in A}$ hier bekannt und nehmen nur die Werte 0 und 1 an.

Nicht jede Aktion ist in jedem Zustand $s \in \mathcal{S}$ erlaubt. Dies kann auf zwei Weisen eingearbeitet werden: Eine Methode ist die Setzung von $R(s, a, s') = -W$ mit einer großen Zahl $W > 0$, falls ein unzulässiger Zustand s' ausgehend von s mit Aktion a erhalten wurde. In diesem Fall wird also eine hohe Bestrafung für die Aktion ausgesprochen. Diese Variante hat den Vorteil, dass sie sich völlig in die nachfolgende Theorie einfügt und die Menge der

möglichen Aktionen $a \in A$ nicht von s abhängt. Da hier jedoch die unerlaubten Züge völlig klar sind, kann man in einer Implementation die Menge der Aktionen auch für jedes $s \in \mathcal{S}$ separat einschränken.

Für die restlichen Tripel (s, a, s') kann $R(s, a, s')$ zum Beispiel wie folgt gesetzt werden: Falls $s' = Z$ den Zielzustand beschreibt, wähle $R(s, a, s') = 1$, für jeden anderen Fall setze $R(s, a, s') = 0$. Für den letzteren Fall gibt es auch hier eine mögliche Anpassung: So kann man beispielsweise $R(s, a, s') = -w$ mit einer kleinen Zahl $w > 0$ setzen. Dies führt dazu, dass jeder Schritt eine kleine Bestrafung nach sich zieht und daher insgesamt Strategien bevorzugt werden, welche den Zielzustand in möglichst wenigen Schritten erreichen. In obigem Labyrinth ist dies nicht von großer Relevanz, da der optimale Weg relativ schnell gefunden werden kann und der nachfolgend vorgestellte Algorithmus eine optimale Strategie ermittelt. Bei größeren Labyrinthen wird aber oft nur eine Annäherung an den optimalen Weg ermittelt. Bei der Berechnung des Algorithmus kann dann eine Bestrafung langer Wege zu besseren Ergebnissen führen, da so mehr verschiedene Möglichkeiten ausprobiert werden.

Eine Entscheidungsstrategie muss jedem möglichen Zustand $s \in \mathcal{S}$ eine Aktion $a \in A$ zuordnen. Nutzt man über die gesamte Zeit hinweg dieselbe Strategie, so wird die Verteilung eines Markov-Entscheidungsprozesses vollständig beschrieben.

Definition 8.6 (Strategie)
Eine Abbildung $\pi : \mathcal{S} \to A$ heißt *Entscheidungsstrategie* bzw. *policy.*

Ein *Markov-Entscheidungsprozess* $(S_t, A_t)_{t\in\mathbb{N}_0}$ mit Strategie π ist ein Markov-Entscheidungsprozess $(S_t, A_t)_{t\in\mathbb{N}_0}$ mit

$$\forall t \in \mathbb{N}_0 : A_t = \pi(S_t).$$ ♦

Die Bestimmung einer möglichst guten Strategie ist eng verknüpft mit einer *Bewertung* der Zustände. Die Bewertung eines Zustands entsteht formal durch Aufsummieren aller in Zukunft erwartbaren Belohnungen. Für jeden Schritt in die Zukunft werden die Belohnungen allerdings um einen Faktor $\gamma \in [0, 1]$ abgewertet.

Definition 8.7
Sei $\gamma \in [0, 1]$. Sei (S_t, A_t) ein Markov-Entscheidungsprozess mit Strategie π.

- Die Abbildung

$$V_\pi : \mathcal{S} \to \mathbb{R}, \quad V_\pi(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1}) \Big| S_0 = s\right]$$

  heißt *(abgewertete) Bewertungsfunktion eines Zustands s*.
- Die Abbildung

$$Q_\pi : \mathcal{S} \times A \to \mathbb{R},$$

$$Q_\pi(s,a) := \mathbb{E}\left[R(s,a,S_1) + \sum_{t=1}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1}) \middle| S_0 = s, A_0 = a \right]$$

heißt *(abgewertete) Bewertungsfunktion für ein Zustands-Aktionspaar* (s,a). ♦

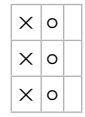
Bemerkung 8.8

- Es gilt $V_\pi(s) = Q_\pi(s, \pi(s))$.
- Die Wahl $\gamma = 1$ entspricht einer *weitsichtigen* Bewertung, die zukünftigen Zustände werden alle gleichrangig mit einbezogen. Dies kann in der Praxis problematisch sein, da ein weit in der Zukunft liegender optimaler Zustand bei einer aggressiven Spielweise des Gegners möglicherweise gar nicht erreicht wird. Die Setzung $\gamma = 1$ ist daher nur zu empfehlen, wenn ein Endzustand stets durch eine geringe Anzahl von Spielzügen erreicht wird.
- Die Wahl $\gamma = 0$ entspricht einer *kurzsichtigen* Bewertung, nur der nächste Zustand zählt. In diesem Fall wird keine weitsichtige Strategie verfolgt. Im Falle des Schachbeispiels wird dadurch (je nach Bewertungsfunktion) dem Ziel, im nächsten Zug keine Figur zu verlieren, oberste Priorität gegeben. Solch eine Strategie erkennt allerdings nur schwerlich, wenn der Gegner „Zwickmühlen" aufbaut.
- Typische Wahl von γ: $\gamma \approx 0{,}9$.
- Die einfache Modellierung der Abwertung durch einen geometrischen Abfall der Form γ^t ist zentral für die folgende Theorie. Andere Arten von Abwertungen können nicht einfach theoretisch behandelt werden und sind auch in der Praxis wesentlich schwieriger umzusetzen.

Bemerkung 8.9 (Ungenaue Modellierung und nicht praxisrelevante Zustände) Angenommen, es gibt eine Menge M von Zielzuständen, die in der Praxis zum Abbruch des beobachteten Markov-Entscheidungsprozesses führen (dies ist bei den oben vorgestellten Beispielen offensichtlich der Fall). Dann ist es für eine korrekte Bewertung der Zustände $s \in \mathcal{S}$ naheliegend, die Summation in der Definition von V_π bzw. Q_π nur bis zu dem Zeitpunkt τ durchzuführen, bei welchem das erste Mal $S_t \in M$ eintritt, d. h. formal:

$$\tau := \inf\{t \in \mathbb{N}_0 : S_t \in M\}, \qquad V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\tau} \gamma^t R(S_t, \pi(S_t), S_{t+1}) \middle| S_0 = s \right]$$

Dies führt jedoch zu einer wesentlich schwierigeren Theorie. Wir werden daher im Folgenden für alle Beispiele die Bewertung mittels der in Definition 8.7 gegebenen V_π, Q_π durchführen, auch wenn dies zu Ungenauigkeiten führt. So ist beispielsweise in der theoretischen Formulierung die Bewertung aller Zustände $s \in \mathcal{S}$ von Interesse, in der Praxis gibt es jedoch von vornherein eine Menge von Zuständen, die nie auftreten können, weil zuvor ein Zielzustand eingetreten ist. Im Kontext von Beispiel 8.4 ist beispielsweise der Zustand $s \in \mathcal{S}$ mit graphischer Interpretation



ein Zustand, der in der Praxis nicht auftritt, weil bereits vorher jemand gewonnen hat.

Mit Hilfe der Bellman-Gleichungen (vgl. [6]) kann ein Ein-Schritt-Zusammenhang zwischen den Werten von Q_π und V_π hergestellt werden. Der Beweis folgt direkt aus der Definition von Q_π, V_π:

Lemma 8.10 (Bellman-Gleichungen) In der Situation von Definition 8.7 gelten folgende Aussagen:

- Für die Bewertungsfunktion eines Zustands s:

$$\begin{aligned} V_\pi(s) &= \mathbb{E}\Big[R(s,\pi(s),S_1) + \gamma \cdot V_\pi(S_1)\Big|S_0 = s\Big] \\ &= \sum_{s'\in\mathcal{S}} P(s'|s,\pi(s)) \cdot [R(s,\pi(s),s') + \gamma \cdot V_\pi(s')] \end{aligned} \tag{8.2}$$

- Für die Bewertungsfunktion eines Paares (s,a):

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}\Big[R(s,a,S_1) + \gamma \cdot Q_\pi(S_1,\pi(S_1))\Big|S_0 = s, A_0 = a\Big] \\ &= \sum_{s'\in\mathcal{S}} P(s'|s,a) \cdot \big[R(s,a,s') + \gamma \cdot Q_\pi(s',\pi(s'))\big] \end{aligned} \tag{8.3}$$

Obige Gleichungen sind sogenannte Fixpunktgleichungen. Sie können benutzt werden, um aus gegebenen Übergangswahrscheinlichkeiten $P(s'|s,a)$ die Funktionen Q_π und V_π für eine gegebene Strategie π zu ermitteln. Für die folgende Betrachtung seien $\mathcal{S}, A$ endlich und $\mathcal{V} := \{V : \mathcal{S} \to \mathbb{R} \text{ messbar}\}$ und $\mathcal{Q} := \{Q : \mathcal{S} \times A \to \mathbb{R} \text{ messbar}\}$ die Menge aller möglichen (messbaren) Funktionen V, Q.

Satz 8.11 Sei π eine Strategie und $\gamma \in [0,1)$.

- Sei

$$T_v^\pi : \mathcal{V} \to \mathcal{V}, \quad (T_v^\pi V)(s) := \sum_{s'\in\mathcal{S}} P(s'|s,\pi(s)) \cdot \big[R(s,\pi(s),s') + \gamma \cdot V(s')\big].$$

  Dann gilt: Ist $V^{(0)} \in \mathcal{V}$ beliebig und $V^{(k+1)} := T_v^\pi V^{(k)}$ $(k \in \mathbb{N}_0)$, so gilt punktweise $V^{(k)} \to V_\pi$ $(k \to \infty)$.
- Sei

$$T_q^\pi : \mathcal{Q} \to \mathcal{Q}, \quad (T_q^\pi Q)(s,a) := \sum_{s'\in\mathcal{S}} P(s'|s,a) \cdot \big[R(s,a,s') + \gamma \cdot Q(s',\pi(s'))\big].$$

Dann gilt: Ist $Q^{(0)} \in \mathcal{Q}$ beliebig und $Q^{(k+1)} := T^\pi Q^{(k)}$ $(k \in \mathbb{N}_0)$, so gilt punktweise $Q^{(k)} \to Q_\pi$ $(k \to \infty)$.

Beweis Die Aussagen folgen aus dem Banach'schen Fixpunktsatz. Auf $\mathcal{V}$ definieren wir beispielsweise die Metrik $D(V_1, V_2) := \max_{s\in\mathcal{S}} |V_1(s) - V_2(s)|$. Da $\mathcal{S}$ endlich und $\mathbb{R}$ vollständig ist, ist $(\mathcal{V}, D)$ ein vollständiger metrischer Raum. Es gilt wegen $\sum_{s'\in\mathcal{S}} P(s'|s, \pi(s)) = 1$:

$$\begin{aligned} D(T_v^\pi(V_1), T_v^\pi(V_2)) &\leq \gamma \cdot \max_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} P(s'|s, \pi(s)) \cdot \underbrace{\left|V_1(s') - V_2(s')\right|}_{\leq D(V_1,V_2)} \\ &\leq \gamma \cdot D(V_1, V_2) \end{aligned}$$

Mit $\gamma \in [0, 1)$ folgt: T_v^π ist eine Kontraktion.

Durch die Aussage des Satzes kann V_π bzw. Q_π aus einer gegebenen Strategie π ermittelt werden, falls die Übergangswahrscheinlichkeiten $P(s'|s, a)$ bekannt sind. Eine gute Strategie π sollte $V_\pi(s)$ für möglichst viele $s \in \mathcal{S}$ maximieren. In der Praxis ist es jedoch aufgrund der Mächtigkeit der Menge aller Strategien nicht möglich, alle π durchzuprobieren und deren V_π miteinander zu vergleichen. Im Folgenden beschäftigen wir uns zunächst mit der theoretischen Existenz einer optimalen Strategie $\pi^* : \mathcal{S} \to A$ bzgl. der zuvor festgelegten Belohnung R und dem Abwertungsfaktor γ und geben danach effiziente Möglichkeiten der Berechnung an. Die optimale Strategie soll ausgehend von einem beliebigen Startzustand angeben, durch welche Aktionen man zur höchstmöglichen Bewertung gelangt. Formal fordern wir gemäß folgender Definition:

Definition 8.12 (Optimale Strategie)
Die *optimale Bewertungsfunktion* auf $\mathcal{S}$ ist

$$V^* : \mathcal{S} \to \mathbb{R}, \quad V^*(s) := \max_{\pi:\mathcal{S}\to A} V_\pi(s). \tag{8.4}$$

Analog definieren wir die *optimale Bewertungsfunktion* auf $\mathcal{S} \times \mathcal{A}$:

$$Q^* : \mathcal{S} \times A \to \mathbb{R}, \quad Q^*(s, a) := \max_{\pi:\mathcal{S}\to A} Q_\pi(s, a) \tag{8.5}$$

Eine Funktion $\pi^* : \mathcal{S} \to A$ heißt *optimale Strategie,* falls für alle $s \in \mathcal{S}$, $\pi : \mathcal{S} \to A$ gilt:

$$V_{\pi^*}(s) = V^*(s) \qquad \blacklozenge$$

Eine optimale Strategie liefert also für jeden Zustand $s \in \mathcal{S}$ die optimale Bewertung $V^*(s)$. Es ist nicht klar, ob solch eine optimale Strategie π^* existiert. Aufgrund der Endlichkeit von $\{\pi : \mathcal{S} \to A\}$ (da $\mathcal{S}$, A endlich) kann man zwar für jedes $s \in \mathcal{S}$ in Gl. (8.4) einen Maximierer $\pi_s : \mathcal{S} \to A$ finden. Die Definition der optimalen Strategie verlangt allerdings, dass für jedes $s \in \mathcal{S}$ *dieselbe* Funktion $\pi^* : \mathcal{S} \to A$ Maximierer ist. Der folgende Satz ist der erste Schritt hin zu einem Existenzbeweis von π^*.

Satz 8.13 Sei $\gamma \in [0,1)$. Sei $V^* : \mathcal{S} \to \mathbb{R}$ eine Abbildung. Für $V \in \mathcal{V}$, definiere den Operator

$$T_v^* : \mathcal{V} \to \mathcal{V}, \quad (T_v^* V)(s) := \max_{a \in A} \mathbb{E}\Big[R(s,a,S_1) + \gamma V(S_1)\Big| S_0 = s, A_0 = a\Big]. \tag{8.6}$$

Dann sind folgende Aussagen äquivalent:

(i) Für alle $s \in \mathcal{S}$ ist $V^*(s) = \max_\pi V_\pi(s)$, und es gibt eine optimale Strategie π^*.
(ii) V^* ist Lösung der Rekursionsgleichung

$$T_v^* V^* = V^*. \tag{8.7}$$

In diesem Falle ist jede Strategie der Form

$$\tilde{\pi}^* : \mathcal{S} \to A, \quad \tilde{\pi}^*(s) :\in \arg\max_{a \in A} \mathbb{E}\big[R(s,a,S_1) + \gamma V^*(S_1)\big| S_0 = s, A_0 = a\big] \tag{8.8}$$

eine optimale Strategie.

Beweis (i)⇒(ii): Es gilt für alle $s \in \mathcal{S}$:

$$\begin{aligned} V^*(s) = V_{\pi^*}(s) &= \mathbb{E}[R(s,\pi^*(s),S_1) + \gamma \cdot V_{\pi^*}(S_1)|S_0 = s] \\ &= \sum_{s' \in \mathcal{S}} P(s'|s,\pi^*(s)) \cdot \big[R(s,\pi^*(s),s') + \gamma \cdot V_{\pi^*}(s')\big] \\ &\le \max_{a \in A} \sum_{s' \in \mathcal{S}} P(s'|s,a) \cdot \big[R(s,a,s') + \gamma \cdot V^*(s')\big] = (T_v^* V^*)(s) \end{aligned}$$

Angenommen, es gibt ein $\tilde{s} \in \mathcal{S}$, bei welchem obige Ungleichung mit „<“ erfüllt ist. Definiere $\tilde{\pi}^*$ gemäß Gl. (8.8). Mit dieser Wahl gilt $(T_v^{\tilde{\pi}^*} V^*)(s) \ge V^*(s)$ für alle $s \in \mathcal{S}$ und $(T_v^{\tilde{\pi}^*} V^*)(\tilde{s}) > V^*(\tilde{s})$. Definiere $R_n(s) := |((T_v^{\tilde{\pi}^*})^n V^*)(s) - ((T_v^{\tilde{\pi}^*})^n V_{\tilde{\pi}^*})(s)|$. Wegen $\gamma \in [0,1)$ ist $T_v^{\tilde{\pi}^*}$ kontrahierend, d. h. für alle $s \in \mathcal{S}$ gilt $R_n(s) \to 0$ $(n \to \infty)$. Es folgt für alle $s \in \mathcal{S}$:

$$\begin{aligned} V_{\tilde{\pi}^*}(s) &= ((T_v^{\tilde{\pi}^*})^n V_{\tilde{\pi}^*})(s) = ((T_v^{\tilde{\pi}^*})^n V^*)(s) + R_n(s) \\ &\ge ((T_v^{\tilde{\pi}^*})^{n-1} V^*)(s) + R_n(s) \ge \ldots \ge (T_v^{\tilde{\pi}^*} V^*)(s) + R_n(s) \qquad (8.9) \\ &\ge V^*(s) + R_n(s) \qquad (8.10) \end{aligned}$$

Mit $n \to \infty$ folgt aus Gl. (8.10), dass $V_{\tilde{\pi}^*}(s) \ge V^*(s)$. Aus Gl. (8.9) folgt mit $n \to \infty$, dass $V_{\tilde{\pi}^*}(\tilde{s}) = (T_v^{\tilde{\pi}^*} V^*)(\tilde{s}) > V^*(\tilde{s})$. Damit ist $\tilde{\pi}^*$ eine bessere Strategie als π^*, Widerspruch zur Optimalität von π^*, d. h. zu $V^*(s) = \max_\pi V_\pi(s)$. Es folgt Gl. (8.7) und damit (ii).

Insbesondere wurde gezeigt: Ist π^* eine optimale Strategie, so gilt für $\tilde{\pi}^*$ auch $V_{\tilde{\pi}^*}(s) \ge V^*(s)$ für alle $s \in \mathcal{S}$, d. h., auch $\tilde{\pi}^*$ ist eine optimale Strategie.

(ii)⇒(i): Es gelte $V^* = T_v^* V^*$. Wir zeigen zunächst $V^*(s) = \max_\pi V_\pi(s)$ für alle $s \in \mathcal{S}$. Wir zeigen getrennt (a) $V^*(s) \le \max_\pi V_\pi(s)$ und (b) $V^*(s) \ge \max_\pi V_\pi(s)$.

Zu (a): Definiere $\tilde{\pi}^*$ gemäß Gl. (8.8). Wie oben folgt für alle $s \in \mathcal{S}$: $V_{\tilde{\pi}^*}(s) \geq V^*(s) + R_n(s)$. Der Grenzübergang $n \to \infty$ liefert für alle $s \in \mathcal{S}$: $\max_\pi V_\pi(s) \geq V_{\tilde{\pi}^*}(s) \geq V^*(s)$.

Zu (b): Sei $\pi : \mathcal{S} \to A$ beliebig. Sei $s \in \mathcal{S}$. Dann ist ähnlich wie zuvor mit $R'_n(s) := |((T_v^\pi)^n V_\pi)(s) - ((T_v^\pi)^n V^*)(s)|$:

$$\begin{aligned} V_\pi(s) &= ((T_v^\pi)^n V_\pi)(s) \\ &= ((T_v^\pi)^n V^*)(s) + R'_n(s) \leq \ldots \leq (T_v^\pi V^*)(s) + R'_n(s) \leq V^*(s) + R'_n(s), \end{aligned}$$

und mit $n \to \infty$ folgt $V_\pi(s) \leq V^*(s)$. Da π beliebig war, folgt $V^*(s) \geq \max_\pi V_\pi(s)$, und damit gilt insgesamt (i).

In (a) wurde gesehen, dass für alle $s \in \mathcal{S}$ gilt: $V_{\tilde{\pi}^*}(s) \geq V^*(s) = \max_\pi V_\pi(s)$, d. h., $\tilde{\pi}^*$ ist eine optimale Strategie.

Als Folgerung aus Satz 8.13 erhalten wir folgenden Satz:

Satz 8.14 (Existenz und Berechnung einer optimalen Strategie) Sei $\gamma \in [0, 1)$. Dann gilt:

(i) Für alle $s \in \mathcal{S}$ ist $V^*(s) = (T_v^* V^*)(s)$ (wobei T_v^* aus Gl. (8.6)), und

$$\pi^* : \mathcal{S} \to A, \quad \pi^*(s) \in \arg\max_{a \in A} \sum_{s' \in \mathcal{S}} P(s'|s, a) \cdot \big[R(s, a, s') + \gamma \cdot V^*(s')\big]$$

ist eine optimale Strategie.

(ii) Für alle $s \in \mathcal{S}$, $a \in A$ gilt $Q^*(s, a) = (T_q^* Q^*)(s, a)$, wobei

$$T_q^* : \mathcal{Q} \to \mathcal{Q}, \quad (T_q^* Q)(s, a) := \sum_{s' \in \mathcal{S}} P(s'|s, a) \cdot \big[R(s, a, s') + \gamma \cdot \max_{a' \in A} Q^*(s', a')\big], \tag{8.11}$$

und

$$\pi^*(s) \in \arg\max_{a \in A} Q^*(s, a) \tag{8.12}$$

ist eine optimale Strategie.

Beweis Wegen $\gamma \in [0, 1)$ ist T_v^* in Gl. (8.6) kontrahierend. Daher besitzt die Gleichung $T_v^* V^* = V^*$ nach dem Banach'schen Fixpunktsatz in Gl. (8.7) eine Lösung im endlichen Raum $\mathcal{V}$. Aussage (i) folgt nun direkt aus Satz 8.13.

Aussage (ii) folgt aus (i) durch Einsetzen der folgenden beiden Feststellungen:

a) Es gilt $Q^* = Q_{\pi^*}$, denn

$$Q_{\pi^*}(s, a) = \sum_{s' \in \mathcal{S}} P(s'|s, a) \cdot P(s'|s, a) \cdot \big[R(s, a, s') + \gamma \cdot V_{\pi^*}(s')\big],$$

und für alle $s \in \mathcal{S}$ gilt: $V_{\pi^*}(s) = V^*(s) = \max_\pi V_\pi(s)$.

b) Es gilt $V^*(s) = \max_{a \in A} Q^*(s,a)$ (vgl. Definition T_v^* in Gl. (8.6)).

Für den weiteren Verlauf des Kapitels ist Satz 8.14(ii) von fundamentaler Bedeutung, denn Gl. (8.11) und Gl. (8.12) erlauben eine explizite Berechnung der optimalen Strategie durch Iterationen, *ohne* dass verschiedene Strategien ausprobiert werden müssen.

8.2 Q-Value Iteration

Im Folgenden nutzen wir Satz 8.14(ii), um π^* zu ermitteln. Zur Berechnung von Q^* ist allerdings die Berechnung eines Erwartungswerts nötig, welche ohne Kenntnis der Übergangswahrscheinlichkeiten $P(s'|s,a)$ nicht möglich ist. Die Bestimmung von π^* kann daher nur erfolgen, wenn entweder $P(s'|s,a)$ oder direkt $Q^*(s,a)$ bekannt ist. Es ist daher unser Ziel, diese Funktionen aus Daten zu schätzen.

- Schätzt man $P(s'|s,a)$, so nennt man dies einen *model-based approach* (es wird versucht, die gesamten Parameter des Markov-Entscheidungsprozesses zu schätzen). In der Praxis wird dies oft nicht durchgeführt, weil $P(\cdot)$ eine Funktion mit Definitionsbereich $\mathcal{S} \times \mathcal{S} \times A$ ist und bereits die Dimension von $\mathcal{S}$ möglicherweise sehr hoch ist.
- Schätzt man $Q^*(s,a)$, so nennt man dies einen *model-free approach* oder *value-function based approach.* Diese Variante wird in der Praxis häufig angewandt, weil die Schätzung der Funktion $Q^* : \mathcal{S} \times A \to \mathbb{R}$ auch approximativ mit anderen Methoden des Supervised Machine Learnings durchgeführt werden kann, siehe Abschn. 8.4. Das Paradigma, Q^* zu schätzen und π^* darauf basierend zu ermitteln, nennt man *Q-Learning.*

Im Folgenden wollen wir uns mit Q-Learning beschäftigen. *Value iteration* beschreibt ein Verfahren, die optimale Strategie π^* durch Berechnung der Funktion Q^* mittels der Iterationsvorschrift Gl. (8.11) zu ermitteln. Hierbei wird angenommen, dass die Übergangswahrscheinlichkeiten $P(s'|s,a)$ bekannt sind. Dies ist in der Praxis üblicherweise nicht der Fall, weswegen die folgenden Konvergenzaussagen zunächst eher von theoretischem Interesse sind.

Satz 8.15 (Ermittlung von π^* mittels Q-Value Iteration) Sei $Q^{(0)} \in \mathcal{Q}$ beliebig. Für $k = 0, 1, 2, \ldots$ berechne

$$Q^{(k+1)} = T_q^* Q^{(k)}. \tag{8.13}$$

Für $K \in \mathbb{N}$ definiere

$$\tilde{\pi}^{*,K}(s) := \arg\max_{a \in A} Q^{(K)}(s,a).$$

Für $K \to \infty$ gilt dann $\tilde{\pi}^{*,K}(s) \to \pi^*(s)$, wobei π^* eine optimale Strategie ist.

Bemerkungen

- Die Gl. (8.13) bedeutet: Berechne für alle $s \in \mathcal{S}, a \in A$: $Q^{(k+1)}(s,a) = (T_q^* Q^{(k)})(s,a)$.
- In der Praxis kann das Eintreten der Konvergenz beispielsweise mittels

$$\sup_{s\in\mathcal{S}, a\in A} |Q^{(k)}(s,a) - Q^{(k+1)}(s,a)|$$

beurteilt werden. Ist dieser Ausdruck für ein $k \in \mathbb{N}$ kleiner als eine vorgegebene Toleranzschwelle $\varepsilon > 0$, wird die Iteration abgebrochen und $Q^{(k)}$ als Näherung für Q^* verwendet.

8.3 Q-Learning

Q-Learning beschreibt wie Q-Value Iteration ein Verfahren, die optimale Strategie π^* durch Berechnung der Funktion Q^* mittels der Iterationsvorschrift Gl. (8.11) zu ermitteln. Allerdings wird hier angenommen, dass die Übergangswahrscheinlichkeiten $P(s'|s,a)$ *unbekannt* sind.

Ohne die Kenntnis von $P(s'|s,a)$ ist die Iterationsvorschrift Gl. (8.11) nicht direkt verwendbar. Durch Beobachtungen des Markov-Entscheidungsprozesses kann jedoch eine Approximation von $P(s'|s,a)$ ermittelt werden: In Beispiel 8.4 (Tic-Tac-Toe) bedeutet beispielsweise der Start eines Spiels die Vorgabe eines Startzustands $S_0 = s$ und einer Aktion $A_0 = a$. Die Aktion von Spieler 2 führt dann zum nächsten beobachteten Zustand $s_{neu} := S_1$ und stellt somit eine mögliche Realisierung des Markov-Entscheidungsprozesses zum Zeitpunkt $t = 1$ dar. Basierend auf dieser Beobachtung können wir grob wie folgt approximieren:

$$P(s'|s,a) \approx \mathbb{1}_{s_{neu}}(s')$$

(wir nehmen also an, dass mit Wahrscheinlichkeit 1 nach (s,a) der Zustand $s' = s_{neu}$ eintritt). Da wir $P(s'|s,a)$ im Folgenden nicht separat schätzen und diese Schätzung abspeichern wollen, kann diese Approximation auch über die Zeit nicht verbessert werden und wird permanent angewandt. Die Approximation von Gl. (8.11) lautet dann:

$$Q^*(s,a) \approx R(s,a,s_{neu}) + \gamma \max_{a'\in A} Q^*(s_{neu}, a') \tag{8.14}$$

Wiederholung dieses Prinzips für verschiedene Startzustände $s \in \mathcal{S}$ und Aktionen $a \in A$ erlaubt dann eine Rekonstruktion der vollständigen Funktion Q^*. Wir erhalten folgenden Algorithmus zum Lernen der Funktion Q^*:

Algorithmus 8.16 (Ermittlung π^* mit Q-Learning; elementar) Sei $\alpha_t \in (0,1)$, $t \in \mathbb{N}_0$ (sogenannte *learning rate*) und $\delta_t \in [0,1]$, $t \in \mathbb{N}_0$ (sogenannte *exploration rate*).

Sei $Q^{(0)} \in \mathcal{Q}$ beliebig. Wiederhole für $t = 0, 1, 2, 3, \ldots$:

(1) Wähle gleichverteilt und zufällig einen Zustand $s \in \mathcal{S}$.
(2) Mit Wahrscheinlichkeit δ_t, wähle $a \in A$ zufällig; anderenfalls berechne $a := \arg\max_{a \in A} Q^{(t)}(s, a)$.
(3) Starte einen Markov-Entscheidungsprozess mit $S_0 = s$, $A_0 = a$ und erhalte $s' = S_1$.
(4) Aktualisiere $Q^{(t)}$ entsprechend Gl. (8.14), d. h., setze $Q^{(t+1)} = Q^{(t)}$ und

$$Q^{(t+1)}(s, a) := Q^{(t)}(s, a) + \alpha_t \cdot \left[R(s, a, s') + \gamma \cdot \max_{a' \in A} Q^{(t)}(s', a') - Q^{(t)}(s, a) \right]. \quad (8.15)$$

Sei $T \in \mathbb{N}$. Definiere

$$\hat{\pi}^{(T)}(s) := \arg\max_{a \in A} Q^{(T)}(s, a). \quad (8.16)$$

Bemerkung 8.17

1. Der oben vorgestellte Algorithmus ist zunächst von theoretischem Interesse (vgl. nachfolgenden Satz 8.18) und wird in der Praxis abgewandelt. Dies wird in Abschn. 8.3.1 besprochen.
2. In Schritt (1) wird ein zufälliger Zustand $s \in \mathcal{S}$ aus allen möglichen Zuständen ausgewählt. Für eine theoretische Aussage über die Konvergenz von $Q^{(t)}$ ist dies offensichtlich notwendig. In der Praxis ist jedoch häufig aus Gründen der einfacheren Implementation der Zustandsraum $\mathcal{S}$ zu groß gewählt (vgl. Beispiel 8.4). Dann ist eine zufällige Wahl von $s \in \mathcal{S}$ nicht sinnvoll, weil so viele irrelevante Zustände als Ausgangszustand genutzt werden. Mit „irrelevanten" Zuständen meinen wir überflüssige Zustände in $\mathcal{S}$, welche *in der Praxis* nicht auftreten können, da bereits zuvor ein Zielzustand eingetreten sein muss (vgl. Bemerkung 8.9). Hier zeigt sich also eine weitere Diskrepanz von Theorie und Praxis, die wir im Folgenden als gegeben hinnehmen müssen.
3. In Schritt (2) wird mit Wahrscheinlichkeit δ_t eine zufällige Aktion ausgewählt; ansonsten wird die im Moment vielversprechendste Aktion $a \in A$ durch Approximation von Gl. (8.12) genutzt. Die zufällige Auswahl von Aktionen garantiert, dass viele verschiedene Zustände und Möglichkeiten erkundet (engl. *explore*) werden. In der Praxis sollte $\delta_t \downarrow 0$ $(t \to \infty)$ gewählt werden, damit nach einer anfänglichen Phase des intensiven Erkundens die Konvergenz von $Q^{(T)}$ auf der Menge der relevanten Zustände gewährleistet werden kann. Die Bedingung $\delta_t \downarrow 0$ kann bei Systemen mit geringer Kardinalität von $\mathcal{S} \times A$ verletzt werden, da sich dort die korrekten Werte von Q^* innerhalb weniger Iterationsschritte auf $Q^{(T)}$ übertragen.
4. Der Algorithmus lernt Q^* von „hinten" nach „vorn". Startet man mit $Q^{(0)} \equiv 0$, so kann gemäß Gl. (8.15) für gegebene $s \in \mathcal{S}$ und $a \in A$ nur dann erstmals $Q^{(t+1)}(s, a) \neq 0$ eintreten, wenn entweder $R(s, a, s') \neq 0$ oder $Q^{(t)}(s', a') \neq 0$ gilt. Im ersten Fall muss s ein Zustand sein, aus welchem man direkt in einen Zustand s' mit nichttrivialer Belohnung gelangen kann. Der zweite Fall entspricht dem Regelfall: Für einen möglichen Folgezustand s' von s wurde bereits ein nichttriviales $Q^{(t)}(s', a')$ erreicht, und dieser nichttriviale Wert überträgt sich (mit Modifikationen) von $Q^{(t)}(s', a')$ auf $Q^{(t+1)}(s, a)$. Da üblicherweise Zustände s mit $R(s, a, s')$ erst ganz am Ende eines Spiels/einer Suche

auftreten, sind also einige Iterationen nötig, um das Wissen von den Endzuständen mit nichttrivialer Belohnung auf die Anfangszustände zu übertragen.

5. Schritt (4) nutzt die Approximation Gl. (8.14). Da die rechte Seite von Gl. (8.14) jedoch nur eine Näherung für den korrekten Iterationsschritt darstellt, wird $Q^{(t+1)}(s,a)$ nicht vollständig durch die rechte Seite (mit $Q^{(t)}$) ersetzt, sondern nur einen kleinen Schritt in diese Richtung verschoben. Die Schrittweite α_t nennt man *learning rate*. Um Konvergenz von $Q^{(T)}$ zu gewährleisten, muss $\alpha_t \to 0$ $(t \to \infty)$ gelten.
6. In der Praxis wählt man zum Beispiel $\alpha_t = \alpha_0 \cdot \frac{1}{1+t}$ mit geeignetem $\alpha_0 > 0$; dies ist begründet durch den nachfolgenden Satz 8.18.
7. In der Praxis kann dann $\hat{\pi}^{(T)}$ aus Gl. (8.16) als Prototyp für die optimale Strategie genutzt werden.

Für obigen Algorithmus konnte [39] ein theoretisches Konvergenzresultat beweisen. Die Bedingung (a) in Satz 8.18 wird dabei durch Schritt (2) gewährleistet.

Satz 8.18 Gilt in der Situation von Definition 8.16:

a) Alle $(s,a) \in \mathcal{S} \times A$ werden unendlich oft ausgewählt.
b) $\sum_{t=0}^{\infty} \alpha_t = \infty$, $\sum_{t=0}^{\infty} \alpha_t^2 = \infty$,

so gilt $\mathbb{P}$-f. s. punktweise:

$$Q^{(T)} \to Q^* \quad \textit{und} \quad \hat{\pi}^{(T)} \to \pi^* \quad (T \to \infty)$$

Besonders die Schritte (1) und (3) von Algorithmus 8.16 sind problematisch, denn sie sorgen dafür, dass „interessante“, in der Praxis häufig auftretende Zustände (für die es also im Vergleich zu anderen Zuständen besonders wichtig ist, dass eine sinnvolle Aktion getätigt wird) nur selten besucht werden. In Abschn. 8.3.1 besprechen wir dies detaillierter und formulieren eine Praxisversion von Algorithmus 8.16.

8.3.1 Durchführung in der Praxis mit menschlichem Akteur

Auf Basis des Konvergenzresultats aus Satz 8.18 entstanden die im Folgenden besprochenen Abwandlungen. In der Praxis wird der Q-Learning-Algorithmus aus Definition 8.16 in zwei Schritten abgewandelt. Zunächst modifizieren wir Schritt (1) wie folgt:

Bemerkung 8.19 (Modifikation 1: Änderung Schritt (1))

- Es wird nicht in jedem Iterationsschritt t ein neuer zufälliger Zustand $s \in \mathcal{S}$ erzeugt, sondern ein Startzustand $s_0 \in \mathcal{S}$ ausgewählt und der Markov-Entscheidungsprozess mit $S_0 = s_0$ nicht nur einen Schritt, sondern sehr lange beobachtet (der neue Zustand $s' = S_1$ wird dann also in Schritt (1) wieder als s gewählt). Der Zustand s', sofern zufällig, wird dabei durch einen menschlichen Akteur erhalten. Üblicherweise wird der Prozess beobachtet, bis ein Zielzustand erreicht wird (sofern existent). Einen solchen Durchgang von Zuständen nennt man *Episode*.
- Wenn der Prozess stoppt (z. B. Spielende), beginnt man wieder von vorn mit dem Startzustand s_0. Im Allgemeinen ergibt sich ein anderer Verlauf, da entweder in Schritt (2) des Algorithmus 8.16 aufgrund der Aktualisierungen bereits ein anderes a ausgewählt wird oder der Markov-Entscheidungsprozess in Schritt (3) ein anderes s' ausgibt.

Für die lange Beobachtung eines Markov-Entscheidungsprozesses statt eines ständigen Neustarts gibt es folgende Gründe:

- Durch die zufällige Auswahl aus $\mathcal{S}$ werden häufig Zustände betrachtet, die in der Realität nur sehr selten auftreten oder nicht aus der Evolution eines Markov-Entscheidungsprozesses entstehen können (Beispiel Schach: Ein zufälliger Zustand würde einer zufälligen Verteilung eines Teils aller Schachfiguren auf dem Spielbrett entsprechen. Viele dieser Konstellationen können in der Praxis nicht eintreten; deren Bewertung Q^* zu schätzen ist daher insbesondere in Hinsicht auf die Ermittlung einer optimalen Strategie sinnlos). Durch längere Beobachtung eines Markov-Entscheidungsprozesses gelangt man hingegen schnell zu Zuständen, welche auch in der Realität von Interesse sind.
- Die Konvergenz eines Algorithmus mit zufällig ausgewählten Startzuständen ist bei hochdimensionalem Zustandsraum viel langsamer, da viel seltener Zustände mit Belohnung $R(s, a, s') \neq 0$ erreicht werden. Dies führt dann zu keiner Aktualisierung von $Q^{(t+1)}$ in Gl. (8.15).

Wir wollen weiterhin guten Gewissens auf das Konvergenzresultat aus Satz 8.18 abstellen. Eine wesentliche Voraussetzung war, dass der Zustand $s \in \mathcal{S}$ im Q-Learning-Algorithmus 8.16 in Schritt (1) jeweils zufällig ausgewählt wird. Diese Voraussetzung wird durch obige Modifikation verletzt. Ein Markov-Entscheidungsprozess, welcher nicht in jedem Schritt neu gestartet wird, besucht Zustände S_0, S_1, S_2, ..., die stark voneinander abhängen. Beim Spiel Tic-Tac-Toe (vgl. Beispiel 8.4) beispielsweise besteht der Zustand S_2 aus den ausgefüllten Feldern von S_1 und zwei weiteren ausgefüllten Feldern. Offensichtlich sind diese Zustände nicht stochastisch unabhängig voneinander, sondern stark abhängig. Die folgende Modifikation versucht, diese Unabhängigkeit wieder herzustellen:

Bemerkung 8.20 (Modifikation 2: Änderung Schritt (4)) Speichere den Verlauf der letzten M ($M \in \mathbb{N}$) Beobachtungen/Übergänge (s, a, s') in einer Menge $\mathcal{M}$ (sogenannter *Replay Buffer*) ab und ersetze Schritt (4) wie folgt:

(4') Bestimme zufällig B Elemente $(\tilde{s}, \tilde{a}, \tilde{s}')$ aus $\mathcal{M}$ und führe (4) mit diesen Elementen aus.

Dies bezeichnet man als *Experience Replay*. Die Anzahl B nennt man *batch size* und M die *Größe des Replay Buffers.*

Die Herstellung von Unabhängigkeit hat noch einen anderen Beweggrund: Später wollen wir Methoden des Supervised Learnings auf die so in $\mathcal{M}$ erzeugten Beobachtungen anwenden; viele dieser Methoden funktionieren aber nur zuverlässig unter der Annahme von unabhängigen Beobachtungen.

Neben den theoretischen Vorteilen gibt es auch einen praktischen Grund für die Verwendung des *Experience Replay:* Die gesammelten Beobachtungen werden mehrmals und damit effizienter genutzt. Insbesondere wenn die Beschaffung neuer Beobachtungen aufwendig ist, kann so ein schnelleres Lernen der Funktion Q^* bzw. der optimalen Strategie π^* erreicht werden.

Mit den Änderungen aus Bemerkung 8.19 und 8.20 lautet der Algorithmus wie folgt:

Algorithmus 8.21 (Ermittlung π^* mit Q-Learning; Praxisversion) Sei $\alpha_k \in (0, 1)$, $k \in \mathbb{N}_0$ *(learning rate)* und $\delta_k \in [0, 1]$, $k \in \mathbb{N}_0$ *(exploration rate).*
Sei $\mathcal{M} = \emptyset$, $B \in \mathbb{N}$ *(batch size)* und $M \in \mathbb{N}$ (Größe des Replay Buffers).
Sei $Q^{(0)} \in \mathcal{Q}$ beliebig. Sei $k = 0$. Wiederhole:

- Ermittle einen Anfangszustand $s \in \mathcal{S}$ und starte einen Markov-Entscheidungsprozess mit $S_0 = s$.
- Wiederhole für $t = 0, 1, 2, 3, ...$, bis S_t den Zielzustand erreicht oder t größer als eine Schwelle ist (d. h. beobachte eine *Episode*):
  - (1') Sei $s = S_t$.
  - (2') Mit Wahrscheinlichkeit δ_k, wähle $a \in A$ zufällig. Mit Wahrscheinlichkeit $1 - \delta_k$, wähle $a := \arg\max_{a \in A} Q^{(k)}(s, a)$.
  - (3') Wähle $A_t = a$ und erhalte $s' = S_{t+1}$. Sei $\mathcal{M} = \mathcal{M} \cup \{(s, a, s')\}$. Falls $\#\mathcal{M} > M$, lösche die ältesten Beobachtungen, so dass wieder $\#\mathcal{M} = M$ gilt.
  - (4') *Experience Replay:* Wiederhole B-mal:
    *Wähle $(\tilde{s}, \tilde{a}, \tilde{s}') \in \mathcal{M}$ zufällig und führe das Update $Q^{(k)}$ entsprechend 8.14 aus, d. h. setze $Q^{(k+1)} = Q^{(k)}$ und

$$Q^{(k+1)}(\tilde{s},\tilde{a}) := Q^{(k)}(\tilde{s},\tilde{a}) + \alpha_k \cdot \left[R(\tilde{s},\tilde{a},\tilde{s}') + \gamma \cdot \max_{a' \in A} Q^{(k)}(\tilde{s}',a') - Q^{(k)}(\tilde{s},\tilde{a})\right]. \tag{8.17}$$

Erhöhe $k = k + 1$.

Sei $K \in \mathbb{N}$. Definiere

$$\hat{\pi}^{*,K,prax}(s) := \arg\max_{a \in A} Q^{(K)}(s,a).$$

In Hinsicht auf Punkt 4. in Bemerkung 8.17 werden in Anwendungen oft nicht alle Übergänge in (3') in $\mathcal{M}$ abgespeichert, sondern vor allem diejenigen mit $R(s,a,s') \neq 0$. Dadurch erhalten alle zu interessanten Zuständen gehörige Funktionswerte von Q^* schneller nichttriviale Werte. So kann beispielsweise festgelegt werden, dass nur jeder fünfte Übergang abgespeichert wird und zusätzlich jeder Übergang mit $R(s,a,s') \neq 0$.

Die Verwendung des Replay Buffers ist vor allem für komplexere Problemstellungen notwendig. Die gleichzeitige Verwendung einer *batch size* $B > 1$ führt vor allem beim Deep Q-Learning (vgl. Abschn. 8.4) zu stabileren Schätzungen von Q^*.

Für einfache Problemstellungen (mit kleinem $\mathcal{S} \times A$) hingegen kann auf den Replay Buffer verzichtet werden. In diesem Fall setzt man in Algorithmus 8.21 $M = B = 1$. Wir zeigen dies hier zunächst für das Labyrinthbeispiel (vgl. Beispiel 8.5).

Beispiel 8.22 (Fortführung Beispiel Labyrinth 8.5) Wir wenden den Algorithmus 8.21 beim Labyrinth aus Beispiel 8.5 an mit $\gamma = 0{,}9$ und ohne Replay Buffer $M = B = 1$, sowie den sechs verschiedenen Konfigurationen

(i) $\alpha_k = 0{,}5$, $\delta_k = 0{,}5$,

(ii) $\alpha_k = 0{,}5$, $\delta_k = 0{,}1$,

(iii) $\alpha_k = 0{,}1$, $\delta_k = 0{,}5$,

(iv) $\alpha_k = \frac{0{,}5}{1+0{,}0001k}$, $\delta_k = \frac{0{,}5}{1+0{,}001k}$,

(v) $\alpha_k = \frac{0{,}5}{1+0{,}0001k}$, $\delta_k = \frac{0{,}5}{1+0{,}1k}$,

(vi) $\alpha_k = \frac{0{,}5}{1+0{,}001k}$, $\delta_k = \frac{0{,}5}{1+0{,}001k}$.

Die Konvergenz von $Q^{(t)}$ kann wie folgt anhand von Schritt (4) in Algorithmus 8.16 überprüft werden: Vor dem Aktualisieren messen wir die absolute Schrittweite

$$d_k := \left|R(s,a,s') + \gamma \cdot \max_{a' \in A} Q^{(k)}(s',a') - Q^{(k)}(s,a)\right|.$$

Nach $N = 1000$ Durchläufen des Schritts (4') bilden wir den Mittelwert $D_k := \frac{1}{N}\sum_{j=1}^{N} d_{k-j}$ der letzten N Schrittweiten. Da $R(s,a,s')$ stets 0 oder 1 ist, kann man bereits bei $D_k \approx 0{,}1$ erwarten, dass $Q^{(k)}$ eine gute Approximation für Q^* darstellt. Es gibt in dieser Argumentation jedoch eine Schwachstelle: D_k ist nur ein Maß für die Konvergenz von $Q^{(k)}$ bei den

während der Iterationen *besuchten* Zustände. Besucht man also immer dieselben Zustände, wird Q^* auf diesen vielleicht gut durch $Q^{(k)}$ approximiert, möglicherweise aber nicht auf noch nicht besuchten Zuständen.

In Abb. 8.1 ist D_k logarithmisch über $\frac{k}{N}$ aufgetragen. Man kann sehen, dass die genutzten Schrittweiten d_t für wachsendes t immer kleiner werden. Es zeigen sich jedoch Unterschiede für die verschiedenen Fälle (i)–(vi) (dargestellt in Abb. 8.1a–f). Im Fall (i) wird bereits bei $k \approx 40 \cdot 10^3$ erreicht, dass $D_k \approx 0$ gilt. Da die *exploration rate* δ_k weiterhin konstant hoch ist und somit immer wieder zufällige Aktionen getätigt werden, ist hier auch sicher die vollständige Konvergenz von $Q^{(k)} \to Q^*$ eingetreten. Im Fall (ii) ist die *exploration rate* mit $\delta_k = 0{,}1$ wesentlich kleiner als in Fall (i). Der Algorithmus erreicht zwar genauso schnell $D_k \approx 0$, aber D_k bleibt nicht konstant null, sondern oszilliert zwischen null und höheren Werten. Der Grund ist, dass $Q^{(k)} \approx Q^*$ noch nicht für alle Zustände gilt, sondern gewisse Zustände (die „Sackgassen“) zu selten besucht wurden. Für diese Zustände gilt $Q^{(k)} \approx Q^*$ erst nach einer wesentlich höheren Anzahl von Iterationen. Im Fall (iii) ist die *learning rate* α_k wesentlich kleiner als in den Fällen (i) und (ii), deswegen tritt die Konvergenz wesentlich später ein. Da die *exploration rate* hier ausreichend hoch ist, tritt zwar erst für wesentlich höheres k die Konvergenz $Q^{(k)} \approx Q^*$ ein, diese verläuft aber relativ stabil. Dies zeigt bereits, dass ein korrektes Zusammenspiel zwischen *learning rate* und *exploration rate* sehr wichtig ist, damit die Konvergenz $Q^{(k)} \approx Q^*$ nach einer akzeptablen Anzahl von Schritten erwartet werden kann.

Das Labyrinth ist jedoch ein relativ einfaches Beispiel. Da der Zustandsraum $\mathcal{S}$ und auch die Menge der Aktionen A eine sehr geringe Kardinalität besitzen, müssen nicht viele Kombinationen von α_k und δ_k „probiert“ werden, um eine stabile Konvergenz $Q^{(k)} \approx Q^*$ zu erreichen. Bei komplexeren Problemstellungen und insbesondere beim Deep Q-Learning muss die Wahl von α_k und δ_k sehr gewissenhaft erfolgen (und man muss viel ausprobieren). Insbesondere müssen dann die Voraussetzungen von Satz 8.18 eingehalten werden, die wir hier zunächst ignoriert haben. Damit Konvergenz eintritt, muss dann α_k monoton fallen und gegen null konvergieren; in der Praxis muss dasselbe außerdem für δ_k gelten. Dadurch wird jedoch auch eine sinnvolle Wahl von α_k, δ_k schwieriger. Für die Fälle (iv)–(vi) ist das Verhalten des Algorithmus in Abb. 8.1d)–f) illustriert. Während im Fall (iv) eine relativ stabile Konvergenz eintritt, fällt in (v) die *exploration rate* zu schnell ab und es tritt dasselbe Problem wie in (ii) auf, nur wesentlich drastischer. In (vi) hingegen fällt die *learning rate* zu schnell ab, so dass Q^* auf den interessanten Zuständen nicht schnell genug gelernt wird.

Wir betrachten nun nur noch den Fall (i). Die nach $K \approx 50 \cdot 10^3$ Iterationen erhaltene approximierte Bewertungsfunktion $V^{(K)}(s) := \max_{a \in A} Q^{(K)}(s, a)$ für die einzelnen Zustände $s \in \mathcal{S}$ ist in Abb. 8.2 dargestellt. An jeder Stelle $s \in S$ können wir außerdem $Q^{(K)}(s, a)$ angeben. Dies erfolgt nur für die $a \in A$ mit $Q^{(K)}(s, a) \neq 0$; unter Nutzung der Bezeichnungen $\uparrow, \rightarrow, \downarrow, \leftarrow$ für die verschiedenen $a \in A$ erhalten wir die in Abb. 8.3 dargestellten Werte. Eine Approximation der optimalen Strategie ergibt sich aus Gl. (8.16), d. h., in jedem Zustand $s \in \mathcal{S}$ wähle die Richtung $a \in A$ mit dem höchsten beigeordneten Wert von $Q^{(K)}(s, a)$.

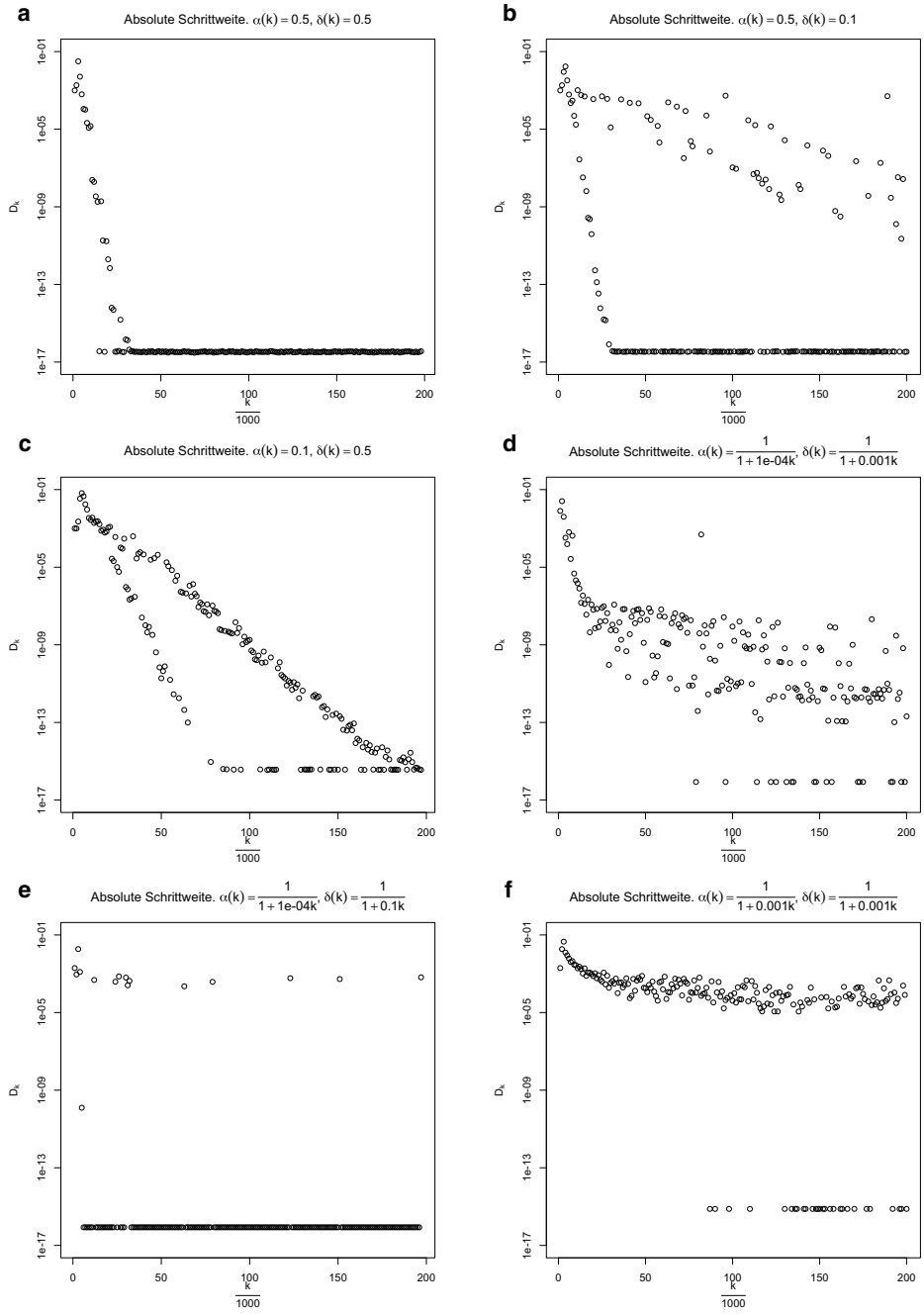


Abb. 8.1 Gemittelte absolute Schrittweiten D_k aufgetragen über der Anzahl der Iterationen k von Schritt (4') des Algorithmus 8.21 in Beispiel 8.22 für verschiedene Wahlen von α_k, δ_k. **a** Wahl (i), **b** Wahl (ii), **c** Wahl (iii)

	Z	1.0	0.9				
			0.81		0.53	0.48	
	0.59	0.66	0.73	0.66	0.59		
	0.53				0.53		
			0.39	0.43	0.48		
	0.28	0.31	0.35		0.43	0.39	

Abb. 8.2 $V^{(K)}(s)$ für die verschiedenen Zustände $s \in \mathcal{S}$ in Beispiel 8.22, Fall (i)

Bemerkung 8.23

- Das Beispiel des Labyrinths ist in dem Sinne „einfach“, als dass der Zustands- und Aktionsraum eine sehr geringe Kardinalität aufweisen. Während des Q-Learnings kann jeder Zustand ausreichend oft besucht werden, so dass eine vollständige Konvergenz $Q^{(K)} \approx Q^*$ erwartet werden kann. Bei komplexeren Problemstellungen kann man dies nicht erwarten. Dann müssen die *learning rate* α_k und die *exploration rate* δ_k geeignet angepasst werden, so dass zumindest die „interessanten“ Zustände ausreichend oft besucht werden.
- Da am Ende oft nur noch die Strategie $\hat{\pi}^{(K)}(s) = \arg\max_{a \in A} Q^{(K)}(s, a)$ von Interesse ist, ist es ausreichend, wenn $Q^{(K)}$ zumindest eine so gute Approximation von Q^* darstellt, dass die Verhältnisse von $Q^{(K)}(s, a)$ für verschiedene $a \in A$ korrekt abgebildet werden.

Das Beispiel des Tic-Tac-Toe-Spiels ist wesentlich komplexer. Hier gilt $\#\mathcal{S} = 3^9 = 19.683$ und $\#A = 9$, so dass Q^* auf insgesamt $3^9 \cdot 9 = 177.147$ Argumenten gelernt werden muss (natürlich sind es tatsächlich wesentlich weniger, da viele Zustands- und Aktionskombinationen nicht möglich sind; trotzdem ist die Anzahl dieser Kombinationen immer noch wesentlich höher als beim Labyrinth).

	Z	← 1.00 → 0.81	← 0.90 ↓ 0.73				
			↑ 0.81 ↓ 0.66		→ 0.43 ↓ 0.53	← 0.48	
	→ 0.59 ↓ 0.48	→ 0.66 ← 0.53	↑ 0.73 → 0.59 ← 0.59	→ 0.53 ← 0.66	↑ 0.48 ↓ 0.48 ← 0.59		
	↑ 0.53				↑ 0.53 ↓ 0.43		
			→ 0.39 ↓ 0.31	→ 0.43 ← 0.35	↑ 0.48 ↓ 0.39 ← 0.39		
	→ 0.28	→ 0.31 ← 0.25	↑ 0.35 ← 0.28		↑ 0.43 → 0.35	← 0.39	

Abb. 8.3 $Q^{(K)}(s,a)$ für alle möglichen Zustände $s \in \mathcal{S}$ und Aktionen $a \in A$ in Beispiel 8.22

Beispiel 8.24 (Fortführung Beispiel 8.4) Wir wenden den Algorithmus 8.21 beim Tic-Tac-Toe-Spiel aus Beispiel 8.4 an mit $\gamma = 0{,}9$, $M = B = 1$ und $\delta_k = \alpha_k = \frac{1}{1+0{,}0001k}$. Der menschliche Spieler 2, welcher uns die Folgezustände s' in Schritt (3') liefern muss, wird simpel simuliert: Spieler 2 wählt einfach einen zufälligen zulässigen Zug aus.

Die Konvergenz von $Q^{(k)}$ wird analog zu Beispiel 8.22 anhand der Schrittweite d_k und für jeweils $N = 1000$ aufeinanderfolgende Iterationen mit den Mittelwerten $D_k = \frac{1}{N}\sum_{j=1}^{N} d_{k-j}$ überprüft.

Zusätzlich zur absoluten Schrittweite (die als „Trainingsfehler" interpretiert werden kann) testen wir die Leistungsfähigkeit der aktuellen Approximation $Q^{(k)}$, indem wir nach jeweils N Iterationen die darauf basierende Strategie $\hat{\pi}^{(k)}(s) = \arg\max_{a \in A} Q^{(k)}(s,a)$ in 1000 Spielen gegen einen zufällig agierenden Gegner antreten lassen. Der prozentuale Anteil

der Nicht-Niederlagen (d. h. Gewinne und Unentschieden) kann als „inverser Validierungsfehler“ interpretiert werden und sollte idealerweise gegen 1 konvergieren.

Wir stoppen den Q-Learning-Algorithmus nach ca. $k = 200 \cdot 10^3$ Iterationen. In Abb. 8.4 ist D_k logarithmisch aufgetragen. Man kann sehen, dass die genutzten Schrittlängen d_k immer kleiner werden und der inverse Validierungsfehler tatsächlich gegen 1 konvergiert.

Im Gegensatz zu Beispiel 8.22 lässt sich die Funktion $Q^{(k)}(s, a)$ aufgrund der hohen Kardinalität des Zustandsraums $\mathcal{S} = \{-1, 0, 1\}^9$ nicht mehr für alle $s \in \mathcal{S}$ visualisieren. Wir zeigen exemplarisch die erhaltenen Werte für einen Spieldurchgang in Abb. 8.5.

An den nicht symmetrisch verteilten Werten von $Q^{(k)}(s, a)$ in Abb. 8.5a ist ersichtlich, dass die Konvergenz $Q^{(k)} \to Q^*$ noch nicht eingetreten ist. Trotzdem liefert der Algorithmus bereits eine gute Approximation für die optimale Strategie: Das mittlere Kästchen wird als Erstes ausgewählt, und im Laufe des Spieles werden Zwickmühlen konstruiert sowie Niederlagen erkannt (negative Zahl in Abb. 8.5d). Die Werte von Q^* sind überwiegend positiv, da ein Unentschieden s' mit $R(s, a, s') = 0{,}5$ bewertet wurde.

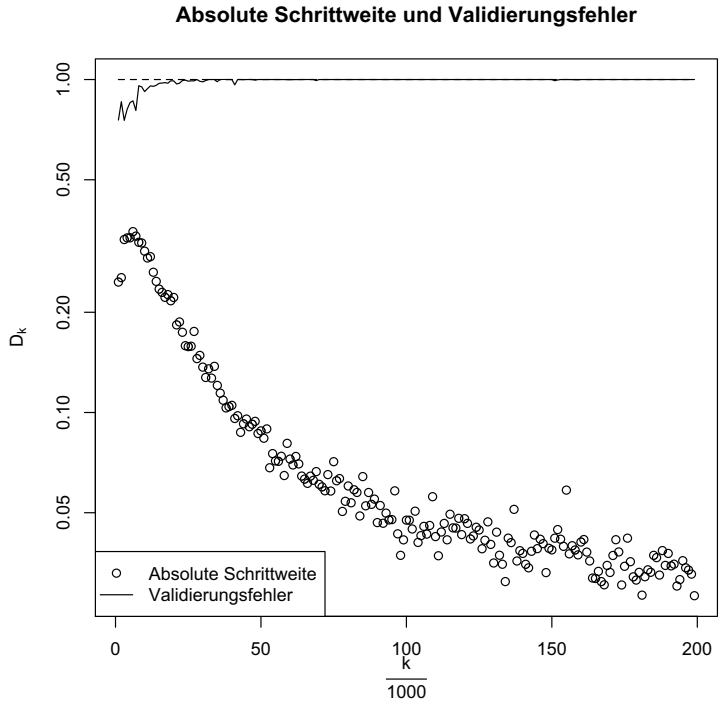


Abb. 8.4 Absolute Schrittweiten D_k aufgetragen über der Anzahl der Iterationen k von Schritt (4') des Algorithmus 8.21 beim Tic-Tac-Toe-Spiel

a)

0.78	0.57	0.59
0.48	0.80	0.57
0.68	0.65	0.67

→

	○	

b)

0.82	0.84	0.74
×	○	0.55
0.68	0.89	0.77

→

×	○	
	○	

c)

0.42	×	0.36
×	○	0.18
0.66	○	0.91

→

	×	
×	○	
	○	○

d)

×	×	0.19
×	○	-0.76
1.00	○	○

→

×	×	
×	○	
○	○	○

Abb. 8.5 $Q^{(k)}(s,a)$ für die Zustände $s \in \mathcal{S}$ eines Spieldurchgangs von Tic-Tac-Toe. Die Werte $Q^{(k)}(s,a)$ für alle möglichen Aktionen sind in die freien Kästchen eingetragen. Die Kreuze wurden manuell im Spielverlauf gesetzt

Ein gutes Abstimmen der Parameter γ, δ_k und α_k ist notwendig, um eine stabile Konvergenz zu erreichen. Dies liegt in der hohen Kardinalität von $\mathcal{S}$: Für ein vollständiges Lernen von Q^* müssen $3^9 \cdot 9$ Parameter korrekt approximiert werden (auch wenn natürlich viele Konfigurationen nicht von Relevanz sind). Eine Beschleunigung ist in mehrerlei Hinsicht möglich:

- Aufstellen eines stärker an das Problem angepassten Zustandsraumes: Nutze die Symmetrien im Spiel, um Zustände und entsprechend darauf folgende Aktionen in der Programmierung zusammenzufassen.
- Verwende einen „intelligenteren“ Gegner, welcher ebenfalls Gewinnabsichten verfolgt (vgl. Abschn. 8.3.2).
- Versuche von vornherein, die Funktion Q^* durch eine geeignete Funktionenklasse zu approximieren, welche weniger Parameter hat und trotzdem eine gute Annäherung an Q^* erlaubt (vgl. Abschn. 8.4, Deep Q-Learning).

Bemerkung 8.25 Wie bereits in Beispiel 8.22 bemerkt ist die Überprüfung der Konvergenz mittels D_k, $i = 1, ..., N$ in Beispiel 8.24 mit Vorsicht zu genießen. Der angewandte Algorithmus 8.21 ist nur in der Lage, $Q^*(s,a)$ für die Zustände s richtig zu schätzen, welche innerhalb der Iterationen genügend oft besucht wurden. Sobald die *exploration rate* δ_k sehr

nahe bei 0 liegt, werden nur noch sehr selten neue Zustände besucht. Der Algorithmus lernt dann nur noch mit den bereits besuchten Zuständen, insbesondere mit den besuchten Zielzuständen, und schätzt Q^* auf Basis dieser. Wurden noch nicht alle relevanten Endzustände genügend oft besucht, wird $Q^{(k)}$ zwar konvergieren, aber nicht gegen das optimale Q^*. In dem Sinne liefert die absolute Schrittweite in Abb. 8.4 bloß, dass nach $200 \cdot 10^3$ Iterationen keine wesentliche Verbesserung von $Q^{(k)}$ mehr zu erreichen ist. Für bessere Ergebnisse muss die Prozedur mit anderen Parametern α_k, δ_k neu gestartet werden. Die Abweichungen $D_k \neq 0$ stammen daher, dass hin und wieder noch neue Zustände durch $\delta_k \neq 0$ besucht werden und der Lernprozess (die Übertragung der Werte $Q^{(k)}$ für Zustände nahe den Zielzuständen an die Werte $Q^{(k)}(s, \cdot)$ für Anfangszustände s) noch nicht vollständig abgeschlossen ist.

8.3.2 Durchführung in der Praxis ohne menschlichen Akteur

Insbesondere beim Entwickeln von optimalen Strategien für Spiele mit zwei oder mehr Akteuren ist eine gute Praxisumsetzung des Schritts (3') in Algorithmus 8.21 nötig. Die Qualität des erhaltenen Zustands s' muss hoch sein, damit schnell starke Strategien gegen starke Gegner entwickelt werden. Gegen schwache Gegner muss ein so ermittelte Strategie trotzdem nicht schlecht sein; mittels der *exploration steps* werden auch Spielzüge solcher Gegner berücksichtigt.

Ein naheliegendes Verfahren ist das simultane Lernen von Q^* aus Sicht des Spielers und des Gegners. Im Schritt (3') wird zur Erzeugung von s' der Algorithmus gespiegelt. Eventuell ist hierfür eine Erweiterung des Zustandsraums $\mathcal{S}$ nötig, da das Spiel nun auch aus der Sicht des Gegners betrachtet wird. Benötigt wird hierfür eine Funktion invert : $\mathcal{S} \to \mathcal{S}$, welche einen Zustand $s_{zwischen}$, aus Sicht des Spielers in denselben Zustand $s^g = \text{invert}(s_{zwischen})$ aus Sicht des Gegners umwandelt. $s_{zwischen}$ ist hierbei der Zustand, welcher direkt nach der Aktion des Spielers und noch vor der Aktion des Gegners eintritt. Die Menge $\mathcal{M}$ aller gesammelten Zustände enthält nun auch die (entsprechend invertierten) Züge aus Sicht des Gegners.

Algorithmus 8.26 (Ermittlung π^* mit Q-Learning; Praxisversion 2) Sei $\alpha_k \in (0, 1)$, $k \in \mathbb{N}_0$ *(learning rate)* und $\delta_k \in [0, 1]$, $k \in \mathbb{N}_0$ *(exploration rate)*.
Sei $\mathcal{M} = \emptyset$, $B \in \mathbb{N}$ *(batch size)* und $M \in \mathbb{N}$ (Größe des Replay Buffers).
Sei $Q^{(0)} \in \mathcal{Q}$ beliebig. Sei $k = 0$. Wiederhole:

- Ermittle einen Anfangszustand $s \in \mathcal{S}$.
- Wiederhole für $t = 0, 1, 2, 3, ...$, bis ein Zielzustand erreicht wird oder t größer als eine Schwelle ist (d. h. beobachte eine *Episode*):

(1S) Falls $t > 1$, setze $s = s'$.
(2S) (Spieler) Mit Wahrscheinlichkeit δ_k, wähle $a \in A$ zufällig. Mit Wahrscheinlichkeit $1 - \delta_k$, wähle $a := \arg\max_{a \in A} Q^{(k)}(s, a)$.
(3S) Erzeuge den ‚Zwischenzustand' $s_{zwischen} \in \mathcal{S}$ aus Sicht des Spielers. Falls $t > 1$, sei $\mathcal{M} = \mathcal{M} \cup \{s^g, a^g, \text{invert}(s)\}$. Falls $\#\mathcal{M} > M$, lösche die ältesten Beobachtungen, so dass wieder $\#\mathcal{M} = M$ gilt.
(1G) Setze $s^g = \text{invert}(s_{zwischen})$.
(2G) (Gegner) Mit Wahrscheinlichkeit δ_k wähle $a^g \in A$ zufällig. Mit Wahrscheinlichkeit $1 - \delta_k$ wähle $a^g := \arg\max_{a \in A} Q^{(k)}(s^g, a)$.
(3G) Erzeuge den ‚Zwischenzustand' $s^g_{zwischen} \in \mathcal{S}$ aus Sicht des Gegners. Sei $s' = \text{invert}(s^g_{zwischen})$. Sei $\mathcal{M} = \mathcal{M} \cup \{(s, a, s')\}$.
(4') Wie in Algorithmus 8.21 (insbesondere die Erhöhung von k um 1).

Sei $K \in \mathbb{N}$. Definiere

$$\hat{\pi}^{*,K,prax2}(s) := \arg\max_{a \in A} Q^{(K)}(s, a).$$

Dieser Algorithmus kann in der Praxis in viele verschiedene Richtungen abgewandelt werden, was die Verwaltung von $\mathcal{M}$ und $Q^{(k)}$ betrifft. So können beispielsweise für Spieler und Gegner auch verschiedene $Q^{(k)}$ genutzt werden.

Beispiel 8.27 (Fortführung Beispiel 8.5) Hier wählt man $\text{invert} : \mathcal{S} \to \mathcal{S}, s = (s_1, ..., s_9) \mapsto -s := (-s_1, ..., -s_9)$, d. h., ‚$\circ$' wird zu ‚$\times$' umgewandelt und umgekehrt; leere Felder bleiben unberührt.

Wir wenden den Algorithmus 8.26 beim Tic-Tac-Toe-Spiel aus Beispiel 8.4 an mit $\gamma = 0{,}9$, Replay Buffer $M = 100$, *batch size* $B = 1$ und $\delta_k = \alpha_k = \frac{1}{1+0{,}0001k}$. Wie in Beispiel 8.24 messen wir die absolute Schrittweite sowie den „inversen Validierungsfehler" alle $N = 1000$ Iterationen von Schritt (4'). In Abb. 8.6a ist D_k sowie der inverse Validierungsfehler logarithmisch über $\frac{k}{N}$ aufgetragen. D_k suggeriert hier, dass Konvergenz von $Q^{(k)}$ eintritt. Betrachtet man allerdings den inversen Validierungsfehler (dieser beträgt für $\frac{k}{N} > 100$ jeweils ca. 0,87), so stellt man fest, dass Q^* und π^* nicht für alle Zustände korrekt gelernt werden. Dies liegt daran, dass die Berechnung des inversen Validierungsfehlers auf eine völlig andere Art erfolgt, als der Lernprozess stattfindet. Da Spieler 2 nun ebenfalls einen Sieg erringen möchte, werden nach einer anfänglichen Erkundungsphase (mit hoher *exploration rate*) vor allem Zustände besucht, die auftreten, wenn beide Spieler Gewinnabsichten verfolgen. Ist die Erkundungsphase zu kurz (d. h., die *exploration rate* fällt zu schnell ab), so kann das gelernte $Q^{(k)}$ nur gut mit Gegnern umgehen, die ebenfalls gewinnen wollen. Der inverse Validierungsfehler wird jedoch mit einem zufällig agierenden Gegner ermittelt. Dabei treten eventuell Zustände s auf, welche $Q^{(k)}$ „noch nicht kennt" bzw. für die $Q^{(k)}(s, \cdot)$ noch nicht ausreichend gut gelernt wurde. Mit anderen Worten, unsere gelernte Strategie $\hat{\pi}^{(k)}$ kann mit der zufälligen Reaktion des Gegners nicht umgehen.

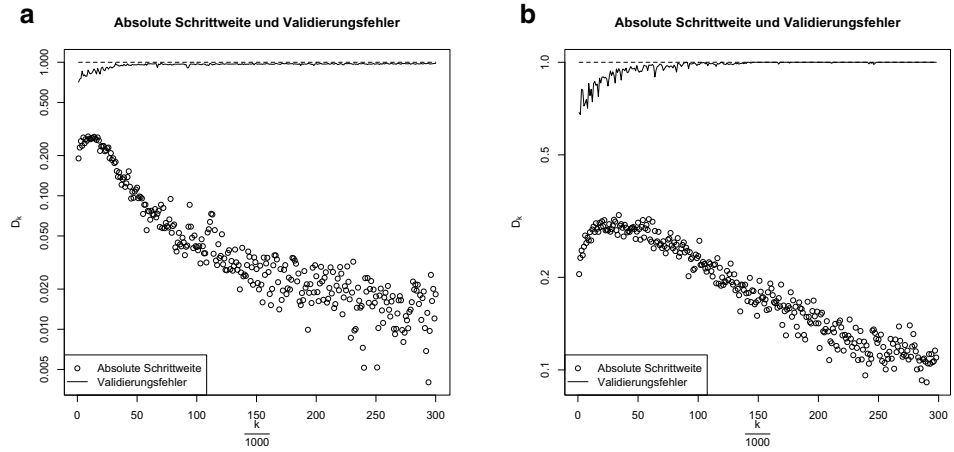


Abb. 8.6 Absolute Schrittweiten D_k aufgetragen über der Anzahl der Iterationen k von Schritt (4') des Algorithmus 8.26 beim Tic-Tac-Toe-Spiel. a): Mit $\delta_k = \frac{1}{1+0{,}0001k}$, b): mit $\delta_k = \frac{1}{1+0{,}00001k}$

Wenn wir Algorithmus 8.26 stattdessen mit einer langsam abfallenderen *exploration rate* $\delta_k = \frac{1}{1+0{,}00001k}$ starten, kann dieses Problem weitgehend behoben werden, indem die Erkundungsphase zu Beginn verlängert wird (vgl. Abb. 8.6b).

Die Anwendung von Algorithmus 8.26 ist daher stets mit Bedacht zu verwenden. Bei komplexeren Problemstellungen kann die Erkundungsphase beispielsweise nicht beliebig lang ausgedehnt werden.

8.4 Approximation durch neuronale Netzwerke: Deep Q-Learning

Bei komplexen Problemen ist eine exakte Darstellung und Ermittlung von $Q^*(s,a)$ aufgrund der hohen Kardinalität von $\mathcal{S} \times A$ oft nicht möglich. In Beispiel 8.4 entspricht $Q^*(s,a)$ einer $3^9 \times 9$-Matrix, d. h., es müssen $3^9 \cdot 9 \approx 177.000$ Parameter gelernt werden. Mindestens dieselbe Anzahl Iterationen wäre nötig, um jeden Eintrag mindestens einmal sinnvoll aktualisiert zu haben. Bei noch größeren Zustandsräumen $\mathcal{S}$ ist die dafür benötigte Berechnungszeit zu lang.

Für eine möglichst gute Schätzung muss daher die Anzahl an Parametern reduziert werden, von welchen $Q^* : \mathcal{S} \times A \to \mathbb{R}$ abhängt, sodass die verbleibenden Parameter mit einer höheren Genauigkeit geschätzt werden können. Formal geschieht dies durch die Annahme, dass Q^* in geeigneten Funktionenräumen liegt. Hierbei fasst man Q^* nicht länger als Abbildung $\mathcal{S} \times A \to \mathbb{R}$ auf, sondern repräsentiert Q^* durch

$$\tilde{Q}^* : \mathcal{S} \to \mathbb{R}^{\#A}, \quad \tilde{Q}^*(s) = (Q^*(s,a))_{a \in A}.$$

Durch die in Definition 8.1 geforderte Einbettung $\mathcal{S} \subset \mathbb{R}^d$ kann $\tilde{Q}^*$ dann aufgefasst werden als eine Funktion

$$\tilde{Q}^* : \mathbb{R}^d \to \mathbb{R}^{\#A}.$$

Auch wenn die Kardinalität von $\mathcal{S}$ sehr groß ist, so ist d selbst für komplexe Probleme nur von moderater Größe. Damit bietet es sich an, $\tilde{Q}^*$ durch Modelle des Supervised Learnings zu approximieren.

Definition 8.28 (Modellannahme: Reinforcement Learning) Es gilt $\tilde{Q}^* \in \mathcal{F}$, wobei $\mathcal{F}$ eine Modellklasse des Supervised Learnings beschreibt. ♦

Da die meisten Modelle des Supervised Learnings stetige Abbildungen (oder zumindest Abbildungen mit beschränkter Variation) liefern, führt man durch diesen Schritt implizit die Annahme ein, dass die verschiedenen Zustände von $\mathcal{S}$ miteinander vergleichbar sind und Algorithmen auf „ähnlichen" Zuständen auch ähnlich agieren sollten. Die Qualität der folgenden Theorie hängt damit wesentlich von der Qualität der Einbettung $\mathcal{S} \subset \mathbb{R}^d$ in Hinsicht auf das zu lösende Problem ab, d. h. mit welchen reellen Vektoren die verschiedenen Zustände bezeichnet werden. Zur Verdeutlichung geben wir zwei kleine Beispiele:

Beispiel 8.29 (Wahl von $\mathcal{S}$)

- Tic-Tac-Toe, vgl. Beispiel 8.4: $\mathcal{S} = \{-1, 0, 1\}^9 \subset \mathbb{R}^9$, d. h. $d = 9$ (mit -1 =‚×', 0 =, ', 1 =‚○'). ‚1' entspricht hier also dem eigenen Symbol und ‚-1' dem gegnerischen Symbol, während die Null die Neutralität eines leeren Feldes ausdrückt. Mit dieser Einbettung wird die Vorstellung ausgedrückt, dass die Strategie auf gegnerischen Symbolen und örtliche Akkumulationen davon auf dem Spielfeld wesentlich anders behandeln soll als ähnliche Ausprägungen mit den eigenen Symbolen.
- Schach: Wähle zum Beispiel $\mathcal{S} = \{-6, ..., -1, 0, 1, ..., 6\}^{64} \subset \mathbb{R}^{64}$, d. h. $d = 64$ (mit 0 = leer, 1 = eigener Bauer, 2 = eigener Springer, 3 = eigener Läufer, 4 = eigener Turm, 5 = eigener Dame, 6 = eigener König, negative entsprechend Gegnerfiguren).
- Negativbeispiel Labyrinth: Betrachte das einfache ‚Labyrinth' in Abb. 8.7, wobei wir jeweils $\mathcal{S} = \{s_1, ..., s_5\}$ nutzen, aber einmal die Zustände von links nach rechts (A zu Z) mit $s_1, ..., s_5 = 1, ..., 5$ identifizieren und einmal mit $s_1, ..., s_5 = 4, 2, 5, 1, 3$. Die Aktionen seien jeweils $A = \{-1, 1\}$, wobei 1 =‚gehe →' und -1 =‚gehe ←' entspricht. Wir brechen den Prozess ab, wenn Z erreicht wird.
  Q^* lautet in diesem Fall:

$Q^*(s, a)$	$a = -1$	$a = 1$
$s = s_1$		0,73
$s = s_2$	0,66	0,81
$s = s_3$	0,73	0,90
$s = s_4$	0,81	1,00
$s = s_5$	0,90	

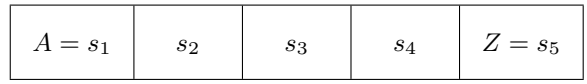


Abb. 8.7 Einfaches Labyrinth mit Startzustand A und Zielzustand Z

Ist nun $s_1, ..., s_5 = 1, ..., 5$, so ist $\tilde{Q}^* : \mathcal{S} \subset \mathbb{R} \to A \subset \mathbb{R}^2$ in jeder Komponente monoton wachsend und einfach zu approximieren; dies wäre eine gute Wahl der Einbettung $\mathcal{S} \subset \mathbb{R}$. Eine schlechte Wahl ist $s_1, ..., s_5 = 4, 2, 5, 1, 3$, so ergibt sich eine wesentlich kompliziertere Abbildung $\tilde{Q}^*$.

Im Folgenden sei zur Vereinfachung der Notation $A = \{1, ..., \#A\}$. Aus Satz 8.18 wissen wir, dass die Iteration im Schritt (4) von Algorithmus 8.16,

$$(\tilde{Q}^{(k+1)}(s))_a := (\tilde{Q}^{(k)}(s))_a + \alpha_k \cdot \left[R(s,a,s') + \gamma \cdot \max_{a' \in A} (\tilde{Q}^{(k)}(s'))_{a'} - (\tilde{Q}^{(k)}(s))_a\right], \quad (8.18)$$

oft genug für verschiedene (s, a) ausgeführt erfüllt: $\tilde{Q}^{(k)} \to \tilde{Q}^*$. Die Vorschrift 8.18 wurde erhalten als Approximation von

$$(\tilde{Q}^{(k+1)}(s))_a \approx R(s,a,s') + \gamma \cdot \max_{a' \in A} (\tilde{Q}^{(k)}(s'))_{a'}. \quad (8.19)$$

Im Folgenden nutzen wir zur Approximation von $\tilde{Q}^*$ ein Modell des Supervised Learnings, ausgedrückt durch parametrische Funktionenklassen

$$\mathcal{F} = \{f_\theta : \mathbb{R}^d \to \mathbb{R}^{\#A} | \theta \in \Theta\}$$

mit einem geeigneten Parameterraum Θ. Dies entspricht der folgenden Modellannahme:

Modellannahme 8.30 (Approximatives Q-Learning) Es gilt $\tilde{Q}^* \in \mathcal{F}$ (d. h., es gibt $\theta^* \in \Theta$ mit $\tilde{Q}^* = f_{\theta^*}$).

Wir wollen weiterhin die Iterationsvorschrift Gl. (8.18) zur Bestimmung von $\tilde{Q}^*$ nutzen; in jedem Iterationsschritt muss nun allerdings eine Darstellung $\tilde{Q}^{(k)} = f_{\theta^{(k)}}$ mit geeignetem $\theta^{(k)} \in \Theta$ existieren. Damit ist ein Update entsprechend Gl. (8.18) nicht mehr möglich: Wir können nicht direkt die Funktionswerte von $f_{\theta^{(k)}}$ verändern, sondern nur die zugehörigen Parameter $\theta^{(k)}$. An dieser Stelle muss die Iterationsvorschrift Gl. (8.18) mit den Schätzmethoden des verwendeten Supervised-Learning-Modells fusioniert werden. Dies geschieht durch folgende Interpretation von Gl. (8.19):

Bemerkung 8.31 (Interpretation durch ein Supervised-Learning-Modell) In jedem Schritt erhalten wir ein Tripel (s, a, s') bestehend aus Ausgangszustand s, getätigter Aktion a und Folgezustand s'. Hierbei kann s aufgefasst werden als Trainingsdatum $X_i = s \in \mathbb{R}^d$, und gemäß Gl. (8.19) ist

$$(Y_i)_a := R(s,a,s') + \gamma \cdot \max_{a' \in A}(\tilde{Q}^{(k)}(s'))_{a'} = R(s,a,s') + \gamma \cdot \max_{a' \in A}(f_{\theta^{(k)}}(s'))_{a'}$$

der Zielwert für die a-te Komponente der Entscheidungsregel $f_{\theta^{(k)}}$ bei $X_i = s$. Für alle anderen Komponenten $\tilde{a} \neq a$ liegt uns keine Information vor, daher setzen wir einfach

$$(Y_i)_{\tilde{a}} := \begin{cases} R(s,a,s') + \gamma \cdot \max_{a' \in A}(f_{\theta^{(k)}}(s'))_{a'}, & \tilde{a} = a, \\ (f_{\theta^{(k)}}(s))_{\tilde{a}}, & \tilde{a} \neq a \end{cases}.$$

In diesem Sinne liefert uns jedes neue beobachtete Tripel (s,a,s') eine neue Beobachtung (X_i, Y_i).

Hat man im k-ten Schritt die Beobachtung (X_k, Y_k) erhalten, so kann nun Schritt (4) in Algorithmus 8.16 wie folgt abgeändert werden: Lerne den Parameter $\theta^{(k+1)}$ (bzw. den zugehörigen Algorithmus $f_{\theta^{(k+1)}}$) neu auf Basis der Trainingsdaten $(X_i, Y_i)_{i=1,\ldots,k}$ und einer Verlustfunktion L. Dies zeigt, dass theoretisch jedes beliebige Verfahren des Supervised Learnings für eine Implementation in das Reinforcement Learning verwendet werden kann. Aufgrund der großen Menge an Iterationen wächst die Datenmenge $(X_i, Y_i)_{i=1,\ldots,k}$ jedoch schnell und entsprechend auch die Dauer der Parameterschätzung.

In der Praxis bevorzugt man daher Methoden, welche für ein (sinnvolles) Lernen des Parameters $\theta^{(k+1)}$ nur das aktuell beobachtete Trainingsdatum (X_k, Y_k) benötigen. Oft wird dies mit der Verwendung der multivariaten quadratischen Verlustfunktion $L(y,s) := \sum_{a \in A}(y_a - s_a)^2$ kombiniert. Die Durchführung einer Gradientenmethode zur Aktualisierung von $\theta^{(k+1)}$ aus $\theta^{(k)}$ lautet dann

$$\theta^{(k+1)} = \theta^{(k)} - \alpha_k \cdot \partial_\theta \{L(f_{\theta^{(k)}}(X_i), Y_i) + \lambda \cdot J(\theta^{(k)})\},$$

wobei $J(\cdot)$ ein differenzierbarer Regularisierungsterm passend zum jeweiligen Modell und λ ein Tuningparameter ist.

Der Übersicht halber lassen wir im Folgenden die in Algorithmus 8.21 eingeführten Verbesserungen weg und erweitern nur den Grundalgorithmus 8.16. Dies führt zu folgender *naiver* Methode:

Algorithmus 8.32 (Q-Learning mit allgemeiner Supervised-Learning-Methode) Sei $\alpha_k \in (0,1), k \in \mathbb{N}_0$ (learning rate).
Sei $f_{\theta^{(0)}} \in \mathcal{F}$ beliebig.
Wiederhole für $k = 0, 1, 2, 3, \ldots$:

(1) Wähle gleichverteilt und zufällig Zustand $s \in \mathcal{S}$.
(2) Berechne $a := \arg\max_{a \in A}(f_{\theta^{(k)}}(s))_a$.
(3) Starte einen Markov-Entscheidungsprozess mit $S_0 = s$, $A_0 = a$ und erhalte $s' = S_1$.
(4) Gradientenmethode (mit Regularisierer J)

$$\theta^{(k+1)} = \theta^{(k)} - \alpha_k \cdot \partial_\theta \Big\{ \big[R(s,a,s') + \gamma \cdot \max_{a' \in A} (f_{\theta^{(k)}}(s'))_{a'} - f_{\theta^{(k)}}(s) \big]^2 + \lambda \cdot J(\theta^{(k)}) \Big\}. \tag{8.20}$$

Sei $K \in \mathbb{N}$. Definiere

$$\hat{\pi}^{*,K,sv}(s) := \arg\max_{a \in A} (f_{\theta^{(K)}}(s))_a.$$

Es ist zu beachten, dass in Gl. (8.20) nun eine doppelte Approximation der ursprünglichen Iteration Gl. (8.11) stattfindet:

(A1) Gl. (8.19) liefert eine Approximation des nächsten Iterationsschritts. Das bedeutet, bereits das zum Anpassen verwendete Y_k ist nur eine Approximation. Daher muss bei der Verwendung von *Experience Replay* auch darauf geachtet werden, die alten Übergänge aus der Menge $\mathcal{M}$ regelmäßig zu entfernen, d. h., M darf nicht zu groß gewählt werden (vgl. Algorithmus 8.21).

(A2) Durch Gl. (8.20) geht man nicht „direkt" auf Y_k zu, sondern nähert sich diesem entlang des vorgegebenen Funktionsraums $\mathcal{F}$ mit einer Gradientenmethode.

Das erhaltene $f_{\theta^{(k+1)}}$ ist also nur eine Approximation der nächsten Funktion $\tilde{Q}^{(k+1)}$, welche zusätzlich noch mit einem nur approximierten Y_k erhalten wurde.

Aufgrund der doppelten Approximation ist 8.20 sehr instabil und konvergiert entweder gar nicht oder nur sehr langsam gegen das wahre $\tilde{Q}^*$. Diese Instabilität wird in der Praxis verringert, indem *Experience Replay* mit einer moderat großen *batch size* verwendet wird (vgl. Algorithmus 8.21). Zusätzlich hält man den Parameter $\theta^{(k)}$ zumindest für die Erzeugung der Beobachtungen Y_k einige Zeit fest. Dadurch wird der durch (A2) bedingte Fehler verringert.

Algorithmus 8.33 (Q-Learning mit allgemeiner Supervised-Learning-Methode 2) Sei $\alpha_k \in (0,1)$, $k \in \mathbb{N}_0$ (learning rate), $C \in \mathbb{N}$ beliebig.

Sei $f_{\bar{\theta}^{(0)}} \in \mathcal{F}$ beliebig.

Wiederhole für $m = 0, 1, 2, 3, \ldots$:

(0) Sei $\theta^{(0)} := \bar{\theta}^{(m)}$.
Wiederhole für $k = 0, 1, \ldots, C-1$:

(1) Wähle gleichverteilt und zufällig Zustand $s \in \mathcal{S}$.
(2) Berechne $a := \arg\max_{a \in A} (f_{\theta^{(k)}}(s))_a$.
(3) Starte einen Markov-Entscheidungsprozess mit $S_0 = s$, $A_0 = a$ und erhalte $s' = S_1$.
(4) Gradientenmethode (mit Regularisierer J)

$$\theta^{(k+1)} = \theta^{(k)} - \alpha_k \cdot \partial_\theta \Big\{ \big[R(s,a,s') + \gamma \cdot \max_{a' \in A} (f_{\bar{\theta}^{(m)}}(s'))_{a'} - f_{\theta^{(k)}}(s) \big]^2 + \lambda \cdot J(\theta^{(k)}) \Big\}. \tag{8.21}$$

Setze $\bar{\theta}^{(m)} = \theta^{(C)}$.

Sei $M \in \mathbb{N}$. Definiere

$$\hat{\pi}^{*,M,svstab}(s) := \arg\max_{a \in A}(f_{\bar{\theta}^{(M)}}(s))_a.$$

Obiger Algorithmus muss nun noch wie in Algorithmus 8.21 modifiziert werden (insbesondere muss *Experience Replay* eingefügt werden). In der Praxis verwendet man für die Funktionenklasse $\mathcal{F}$ häufig neuronale Netzwerke, $\mathcal{F} = \mathcal{F}(L, p)$. In diesem Fall nennt man das obige Verfahren auch *Deep Q-Learning*. Die Gradientenmethode 8.21 wird dann wie in Abschn. 7.3.1 beschrieben ausgeführt.

Beispiel 8.34 (Fortführung Beispiel 8.4) Wir wenden den Algorithmus 8.33 beim Tic-Tac-Toe-Spiel aus Beispiel 8.4 an mit $\gamma = 0{,}9$, $C = 10$ sowie einem *Replay Buffer* $M = 100$ und *batch size* $B = 16$. Die Schritte (1), (4) werden dabei entsprechend Algorithmus 8.21 modifiziert; Spieler 2 lassen wir hier wieder zufällig Folgezustände s' erzeugen.

Als approximierende Funktionenklasse wählen wir ein neuronales Netzwerk $\mathcal{F} = \mathcal{F}(L, p)$ mit $L = 2$, $p = (9, 18, 18, 9)$ mit Bestrafungsparameter $\lambda = 0{,}001$. Während der Durchführung des Algorithmus speichern wir nur jeden fünften Übergang (s, a, s') im *Replay Buffer* ab, und zusätzlich jeden Übergang mit $R(s, a, s') \neq 0$ (vgl. Bemerkung unter Algorithmus 8.21).

Die Konvergenz der zum neuronalen Netzwerk assoziierten Funktion $\tilde{Q}^{(k)} = f_{\theta^{(k)}}$ wird wie zuvor auf Basis von Gl. (8.21) alle $N = 1000$ Iterationen gemessen. In Abb. 8.8 sind die absoluten Schrittweiten D_k sowie der inverse Validierungsfehler logarithmisch aufgetragen. Wir wählen dabei die *learning rate* und *exploration rate* wie folgt:

$$\alpha_k = \begin{cases} 0{,}001, & k \leq 85 \cdot 10^3, \\ 0{,}0001, & 85 \cdot 10^3 < k \leq 188 \cdot 10^3, \\ 0{,}00001, & 188 \cdot 10^3 < k \end{cases} \qquad \delta_k = \begin{cases} 1, & k \leq 85 \cdot 10^3, \\ 0{,}1, & 85 \cdot 10^3 < k \leq 188 \cdot 10^3, \\ 0, & 188 \cdot 10^3 < k \end{cases}$$

Die Umbrüche $k = 85 \cdot 10^3$ und $k = 188 \cdot 10^3$ sind in Abb. 8.8 als senkrechte Linien eingetragen. Die schrittweise Absenkung von α_k dient dazu, den Konvergenzprozess über die Zeit zu stabilisieren. δ_k wird abgesenkt, damit $\tilde{Q}^*$ nach Erforschung vieler möglicher Spiele vor allem an den Zuständen gelernt wird, die bei Verwendung der Strategie $\hat{\pi}^{(k)}$ auftreten.

Wie man sieht, ist nach ca. $k \approx 250 \cdot 10^3$ Schritten die Funktion $\tilde{Q}^{(k)}$ eine sinnvolle Approximation von $\tilde{Q}^*$, es werden kaum noch Spiele gegen einen zufällig agierenden Gegner verloren (inv. Validierungsfehler ≈ 1). Das bedeutet, mit Hilfe der Approximation von $\tilde{Q}^*$ durch ein neuronales Netzwerk konnten wir eine ähnliche Qualität wie in Beispiel 8.24 erreichen, aber mit wesentlich weniger Parametern, d. h., mit wesentlich weniger benötigtem Speicherplatz.

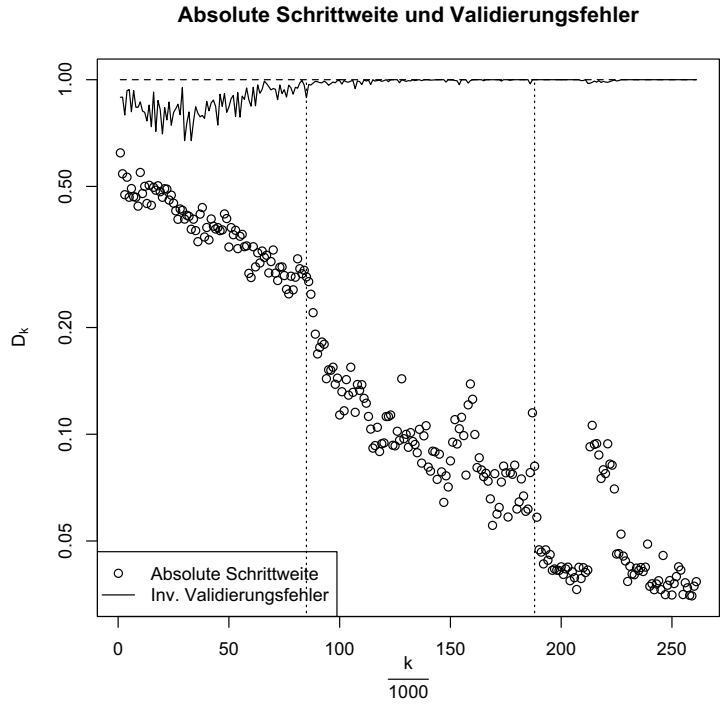


Abb. 8.8 Absolute Schrittweiten D_k und inverser Validierungsfehler aufgetragen über der Anzahl der Iterationen k von Schritt (4) des Algorithmus 8.33 beim Tic-Tac-Toe-Spiel

Unsupervised Learning: Bestimmung von Repräsentanten

9

Inhaltsverzeichnis

Im Gegensatz zum Supervised Learning, bei dem man Trainingsdaten $(X_i, Y_i), i = 1, \ldots, n$ beobachtet, sind beim Unsupervised Learning nur Trainingsdaten $X_i \in \mathcal{X} \subset \mathbb{R}^d$ gegeben. Ziel ist es, Muster in den X_i zu erkennen, die *nicht allein* durch ein gleichmäßig auf den Raum verteiltes, zufälliges Rauschen entstanden sind. Mathematisch entspricht dies der Ermittlung struktureller Aussagen über *die Verteilung* von X.

Jeder Algorithmus zum Auffinden solcher Strukturen stellt eine Modellannahme an die Verteilung von X oder deren Eigenschaften. Aufgabe der Statistik ist es zu prüfen, wie gut die in der Modellannahme vorkommenden Größen durch den Algorithmus geschätzt werden und beispielsweise Konvergenzraten anzugeben. Im Gegensatz zum Supervised Learning (mit der Bewertung von Algorithmen durch das Risiko bzgl. 0-1-Verlust bzw. quadratischem Verlust) hat sich bisher kein einheitliches Qualitätsmaß etabliert, weshalb die Ergebnisse jeweils auf das Verfahren angepasst formuliert werden.

Typische Modellannahmen haben als Ziel eine *Komprimierung* der Trainingsdaten, zum Beispiel wie folgt:

(P1) Reduktion der Vielfalt der Trainingsdaten: Wähle aus oder bilde aus $X_i, i = 1, \ldots, n$ geeignete Repräsentanten $\hat{m}_1, \ldots, \hat{m}_K \in \mathcal{X}$, welche einen Großteil der Daten möglichst gut beschreiben. Formal ist dies verbunden mit der Modellannahme, dass die Verteilung von X um gewisse Punkte $m_1^*, \ldots, m_K^* \in \mathcal{X}$ konzentriert ist. Wir lernen dies anhand des k-means-Clustern und des Clusterns mit Mischungsverteilungen kennen.

(P2) Reduktion der Dimension: Finde geeignete Entsprechungen $\tilde{X}_i \in \mathbb{R}^{\tilde{d}}, i = 1, \ldots, n$ der Trainingsdaten $X_i, i = 1, .., n$ in einem niedrigdimensionaleren Raum $(\tilde{d} < d)$,

S. Richter, *Statistisches und maschinelles Lernen,*
https://doi.org/10.1007/978-3-662-59354-7_9

welche die Lage und Abstände der X_i untereinander möglichst gut nachbilden. Dies entspricht der Modellannahme, dass die Verteilung $\mathbb{P}^{X_1}$ im Wesentlichen um eine k-dimensionale Mannigfaltigkeit

$$M = \{x \in \mathbb{R}^d : h(x) = 0\}$$

konzentriert ist (vgl. *Spektrales Clustern* oder *Hauptkomponentenanalyse*). Neben Anwendungen im Supervised Learning kann diese Methode auch als Vorbereitung für Methoden vom Typ (1) gesehen werden (für Details vgl. Kap. 10).

Die Annahmen (P1) und (P2) sind in der obigen Form noch zu allgemein formuliert und erlauben insbesondere in hohen Dimensionen noch nicht die Formulierung gut generalisierender Komprimierungsalgorithmen. Je nach Anforderung müssen konkretere Modellannahmen gestellt werden. In (P1) muss so präzisiert werden, was unter „Repräsentanten" verstanden wird (z. B. räumliche Nähe); in (P2) muss ein konkretes (parametrisches) Modell für M bzw. die Funktion h und die mit M verbundene Dimensionsreduktion gefordert werden.

In Praxisanwendungen und für ein tieferes Verständnis ist es wichtig, dass Verfahren die folgende Eigenschaft erfüllen:

$$\begin{aligned} &\text{Ein } \textit{neues} \text{ Trainingsdatum } X \in \mathbb{R}^d \text{ muss auf dieselbe Weise} \\ &\text{wie für } X_1, \ldots, X_n \text{ geschehen komprimiert werden können.} \end{aligned} \tag{9.1}$$

Das heißt, in (P1) muss X einem der ermittelten Repräsentanten zugeordnet werden können; in (2) muss eine dimensionsreduzierte Variante $\tilde{X}$ in derselben Weise wie für $X_1, \ldots, X_n$ geschehen ermittelt werden können.

In diesem Kapitel behandeln wir zunächst Ansätze vom Typ (P1), in Kap. 10 betrachten wir Ansätze vom Typ (P2). Eine einfache Anwendung von (P1) in der Praxis ist die Komprimierung von Bilddaten (die Idee dieser Darstellung ist entnommen aus [17]).

Beispiel 9.1 Ein Graustufenbild bestehend aus hb Pixeln ($h \in \mathbb{N}$ Höhe, $b \in \mathbb{N}$ Breite) kann als Matrix $B = (B_{m,j})_{m,j} \in [0,1]^{h \times b}$ aufgefasst werden. Durch

$$X_{m \cdot b + j} := (B_{m,j}, B_{m+1,j}, B_{m,j+1}, B_{m+1,j+1})^T \in [0,1]^4, \quad \begin{matrix} m = 1, \ldots, h-1, \\ j = 1, \ldots, b-1, \end{matrix}$$

erhält man „Beobachtungen" X_i, $i = 1, \ldots, n$ mit $n = (h-1) \cdot (b-1)$, die jeweils der Zusammenfassung der vier Pixel an den Positionen (m, j), $(m+1, j)$, $(m, j+1)$, $(m+1, j+1)$ entsprechen. Wir gehen davon aus, dass alle Komponenten von X_i in etwa dieselben Werte erhalten (*). Mittels K-means Clustering (dies wird in Abschn. 9.1 erklärt) können beispielsweise $K = 6$ Werte $m_1^*, \ldots, m_K^* \in [0,1]^4$ ermittelt werden, welche jeweils einen der Werte X_i möglichst gut repräsentieren bzw. approximieren. Ersetzen wir nun jeweils X_i durch den repräsentierenden Wert $\tilde{X}_i := m_k^*$ ($k \in \{1, \ldots, K\}$), so kann ein neues Bild $B' = (B'_{m,j})_{m,j} \in \mathbb{R}^{(h-1)\cdot(b-1)}$ erhalten werden durch

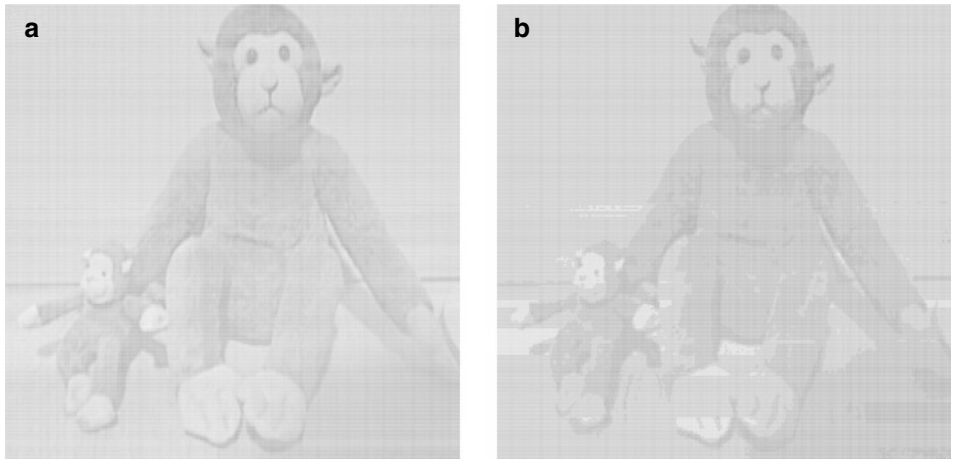


Abb. 9.1 Beispiel für die Komprimierung eines Bildes mit Hilfe des Verfahrens aus Beispiel 9.1. **a** Das ursprüngliche Graustufenbild, **b** Eine komprimierte Version mit nur noch $K = 6$ verschiedenen Graustufenwerten

$$B'_{m,j} := \tilde{X}_{m \cdot b + j, 1}, \quad m = 1, \ldots, h-1, \quad j = 1, \ldots, b-1.$$

Hierbei wählen wir jeweils nur die erste Komponente von $\tilde{X}_i$ wegen (*). In Abb. 9.1 ist ein Beispiel für dieses Verfahren zu sehen. Das zunächst detaillierte Graustufenbild a) wird dadurch komprimiert zu einem Bild b), welches nur noch aus $K = 6$ verschiedenen Graustufenwerten besteht, weiterhin wesentliche Strukturelemente enthält, aber wesentlich effizienter und mit weniger Platz abgespeichert werden kann.

9.1 k-means Clustering

In diesem Abschnitt starten wir zunächst nicht mit einer konkreten Modellannahme an die Verteilung von X, sondern leiten einen Algorithmus nur auf Basis des in (P1) formulierten Ansatzes her. Als Ergebnis erhalten wir den k-means-Clustering-Algorithmus 9.10. Die zugehörige Modellannahme wird in Definition 9.15 herausgearbeitet.

9.1.1 Messung des Abstands

Wir betrachten in diesem Abschnitt nur den Fall $X_i \in \mathbb{R}^d$, d. h. wir verlangen, dass die eingegebenen Daten quantitativ messbar sind (wir geben gleich noch eine Motivation, wie ansonsten vorzugehen ist). Zur Messung der „Verschiedenheit“ zweier Datenpunkte X_i, X_j benötigen wir eine *Abstandsfunktion.*

Definition 9.2 (Abstandsfunktion)
Eine Abbildung $D : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ nennen wir *Abstandsfunktion.* ♦

Häufig verwendete Abstandsfunktionen sind

- der euklidische Abstand: $D(x, x') := \sum_{j=1}^{d} (x_j - x'_j)^2$,
- der absolute Abstand: $D(x, x') := \sum_{j=1}^{d} |x_j - x'_j|$.

Je nach Skalierung und Bedeutung der Daten kann aber auch eine völlig andere Wahl von D sinnvoll sein.

Sind die Daten nicht quantitativ messbar, so kann der im Folgenden eingeführte Algorithmus 9.10 trotzdem anwendbar sein. Liegt beispielsweise die J-te Komponente der Beobachtungen X_i in einer nicht sinnvoll in $\mathbb{R}$ einbettbaren Menge M_J, so gibt es die folgenden beiden Möglichkeiten:

- Erstelle eine sinnvolle Einbettung von M_J in $\mathbb{R}^k$ mit $k \geq 2$. Ist beispielsweise $M_J = \{a, b, c\}$, so kann die Einbettung $\iota : M_J \to \mathbb{R}^3$ mit $\iota(a) = (1, 0, 0)$, $\iota(b) = (0, 1, 0)$, $\iota(c) = (0, 0, 1)$ gewählt werden. Ein Datenpunkt $X_i \in \mathbb{R}^d$ wird also vor der Bearbeitung in einen Datenpunkt $\tilde{X}_i \in \mathbb{R}^{\tilde{d}}$ umgewandelt, welcher in einem höherdimensionalen Raum liegt. Die im Folgenden vorgestellte Theorie kann dann auf $\tilde{X}_i$, $i = 1, \ldots, n$ mit geeigneter Abstandsfunktion $\tilde{D} : \mathbb{R}^{\tilde{d}} \times \mathbb{R}^{\tilde{d}} \to \mathbb{R}_{\geq 0}$ angewandt werden.
- Existiert für zwei Elemente $m, m' \in M_J$ eine sinnvolle Definition der ‚Ungleichheit' $d_J(m, m') \in \mathbb{R}_{\geq 0}$ und sind für die restlichen Komponenten sinnvolle Abstandsfunktionen $d_j : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ wählbar (zum Beispiel $d_j(z, z') := (z - z')^2$), so kann

$$\tilde{D}(x, x') := \sum_{j=1}^{d} d_j(x_j, x'_j)$$

  gewählt und die im Folgenden vorgestellte Theorie auf X_i, $i = 1, \ldots, n$ mit Abstandsfunktion $\tilde{D}$ angewandt werden.

Je nach gewählter Abstandsfunktion ändern sich die Schritte in Bemerkung 9.9 und Algorithmus 9.10 entsprechend.

9.1.2 Motivation k-means und Zuweisungsfunktionen

Gegeben eine natürliche Zahl $K \in \mathbb{N}$ (die *Anzahl der Cluster*) sollen die gegebenen Trainingsdaten X_i, $i = 1, \ldots, n$ in K Mengen (sogenannte *Cluster*) eingeteilt werden, welche jeweils möglichst „ähnliche" X_i beinhalten. Die Einteilung kann formalisiert werden mittels einer Zuweisungsfunktion.

Definition 9.3 (Zuweisungsfunktion)
Eine Abbildung $C : \{1, \ldots, n\} \to \{1, \ldots, K\}$ heißt *Zuweisungsfunktion.* Die Menge aller Zuweisungsfunktionen sei $\mathcal{C}$. ♦

Damit möglichst „ähnliche" X_i in den einzelnen Clustern liegen, muss in jedem Cluster die Summe der Abstände der Elemente untereinander minimiert werden. Damit ergibt sich folgende Definition für eine *optimale* Zuweisungsfunktion.

Definition 9.4 (Optimale Zuweisungsfunktion)
Für eine Zuweisungsfunktion $C \in \mathcal{C}$ nennen wir

$$W_n(C) := \frac{1}{2} \sum_{k=1}^{K} \sum_{i,i':C(i)=C(i')=k} D(X_i, X_{i'}). \tag{9.2}$$

die *Ungleichheit innerhalb der Cluster.* $\hat{C}_n^* = \hat{C}_n^*(X_1, \ldots, X_n) \in \mathcal{C}$ heißt *optimale Zuweisungsfunktion,* falls

$$\hat{C}_n^* :\in \arg\min_{C \in \mathcal{C}} W_n(C). \tag{9.3}$$

♦

Die direkte Ermittlung von $\hat{C}_n^*$ mittels Gl. (9.3) heißt *kombinatorisches Clustering.* Dieses Optimierungsproblem ist für große n und K nicht effizient berechenbar, da die Menge $\mathcal{C}$ der Zuweisungsfunktionen zu groß ist. Wir suchen deswegen im Folgenden nach zu Gl. (9.3) alternativen Darstellungen von $\hat{C}_n^*$, welche eine sinnvolle Approximation von $\hat{C}_n^*$ erlauben.

Wir bemerken hier noch eine äquivalente Formulierung des Optimierungsproblems, um etwas mehr Einsicht in die Definition von $W_n(\cdot)$ zu gewinnen.

Lemma 9.5 Für $C \in \mathcal{C}$ gilt

$$W_n(C) = T_n - B_n(C),$$

wobei

$$T_n := \frac{1}{2} \sum_{i,i'=1}^{n} D(X_i, X_{i'}),$$

$$B_n(C) := \frac{1}{2} \sum_{k=1}^{K} \sum_{i:C(i)=k} \sum_{i':C(i')\neq k} D(X_i, X_{i'}).$$

T_n hängt nicht von C ab und wird auch als *gesamte Ungleichheit* bezeichnet, $B_n(C)$ heißt die *Ungleichheit zwischen den Clustern.*

Beweis Es gilt $T_n = \frac{1}{2} \sum_{k=1}^{K} \sum_{i:C(i)=k} \big(\sum_{i':C(i')=k} D(X_i, X_{i'}) + \sum_{i':C(i')\neq k} D(X_i, X_{i'}) \big)$.

Bemerkung Äquivalent zu Gl. (9.3) ist daher das Maximierungsproblem

$$\hat{C}_n^* :\in \arg\max_{C \in \mathcal{C}} B_n(C). \tag{9.4}$$

Es ist also egal, ob wir die Ungleichheit *zwischen* den einzelnen Clustern maximieren oder die Ungleichheit *innerhalb* der Cluster minimieren. Aufgrund der komplizierten Struktur von $B_n(C)$ werden wir jedoch weiterhin Gl. (9.3) anstelle von Gl. (9.4) betrachten.

In Hinsicht auf Gl. (9.1) und aus statistischer Sicht ist selbst die Kenntnis der Funktion $\hat{C}_n^*$ nicht zufriedenstellend, denn:

- $\hat{C}_n^*$ gibt nur eine Zuordnung der bisherigen Trainingsdaten zu den K Clustern an, erlaubt es aber nicht direkt, weitere Beobachtungen X einem der Cluster zuzuordnen.
- Für immer mehr Trainingsdaten $n \to \infty$ konvergiert $\hat{C}_n^*$ nicht gegen einen festen Ausdruck, sondern die Abbildung $\hat{C}_n^*$ erhält immer mehr Argumente. Mit $\hat{C}_n^*$ lässt sich also nicht unmittelbar asymptotische Theorie betreiben.

Beide Probleme lösen wir auf, indem wir das Optimierungsproblem Gl. (9.3) umformulieren. Wir wollen erreichen, dass die Lösung nicht länger eine optimale Zuweisungsfunktion $\hat{C}_n^*$ ist, sondern eine andere Größe, für welche wir eine Modellannahme formulieren können und die für wachsendes n dieselbe Struktur behält. Um dies möglichst einfach zu erreichen, betrachten wir ab jetzt den Spezialfall $D(x, y) = \|x - y\|_2^2$.

Lemma 9.6 (Ungleichheit innerhalb der Cluster für $D(x, y) = \|x-y\|_2^2$) Ist $D(x, y) = \|x - y\|_2^2$, so gilt mit

$$n_k(C) := \#\{i \in \{1, \ldots, n\} : C(i) = k\}, \qquad m_k(C) := \frac{1}{n_k(C)} \sum_{i:C(i)=k} X_i$$

die Darstellung

$$W_n(C) = \sum_{k=1}^{K} n_k(C) \cdot \sum_{i:C(i)=k} \|X_i - m_k(C)\|_2^2. \tag{9.5}$$

Beweis Es gilt $D(X_i, X_{i'}) = \|X_i - X_{i'}\|_2^2 = \|X_i - m_k(C)\|_2^2 - 2\langle X_i - m_k(C), X_{i'} - m_k(C)\rangle + \|X_{i'} - m_k(C)\|_2^2$. Einsetzen in Gl. (9.2) liefert

$$W_n(C) = 2n_k(C) \sum_{i:C(i)=k} \|X_i - m_k(C)\|_2^2 + \Big\| \sum_{i:C(i)=k} (X_i - m_k(C)) \Big\|_2^2.$$

Wegen $\sum_{i:C(i)=k}(X_i - m_k(C)) = 0$ folgt die Behauptung.

Interpretation von Lemma 9.6 Jede Zuweisungsfunktion $C \in \mathcal{C}$ erzeugt K Cluster mit *Mittelpunkten* $m_k(C)$, $k = 1, \ldots, K$. Zum Cluster $k \in \{1, \ldots, K\}$ gehören diejenigen X_i, welche zu $m_k(C)$ den geringsten Abstand haben. Diese Darstellung erlaubt nun auch eine sinnvolle Zuweisung neuer Beobachtungen X: Diese werden dem Cluster $k \in \{1, \ldots, K\}$ zugeordnet, bei welchem der Abstand $D(X, m_k(\hat{C}_n^*)) = \|X_0 - m_k(\hat{C}_n^*)\|_2^2$ am geringsten ist.

9.1.3 Iterationsverfahren

Mathematische Vereinfachung: Im Folgenden lassen wir die Gewichtung der einzelnen Cluster mittels $n_k(C)$ in 9.5 weg, d. h. wir setzen

$$\tilde{W}_n(C) := \sum_{k=1}^{K} \sum_{i:C(i)=k} \|X_i - m_k(C)\|_2^2 = \sum_{i=1}^{n} \|X_i - m_{C(i)}(C)\|_2^2 \tag{9.6}$$

und wollen Minimierer von $\tilde{W}_n(C)$ finden. Die Gewichtung $n_k(C)$ forcierte, dass die Elemente großer Cluster (d. h. mit vielen Elementen) im Durchschnitt kleinere Abstände untereinander besitzen als bei kleinen Clustern. An dem folgenden Beispiel wird die Veränderung deutlich:

Beispiel 9.7 Wir betrachten $n - 2 = 50$ auf $[-0{,}1, 0{,}1]$ unabhängig gleichverteilte Werte $X_i \in \mathbb{R}$, $i = 1, \ldots, n$ und $X_{51} := -1$, $X_{52} := 1$. In Abb. 9.2 sind die folgenden beiden Zuweisungsfunktionen grafisch dargestellt:

$$C_1(i) := \begin{cases} 1, & i \in \{1, \ldots, 50\}, \\ 2, & i \in \{51, 52\}, \end{cases}, \qquad C_2(i) := \begin{cases} 1, & X_i < 0, \\ 2, & X_i > 0 \end{cases}$$

Anschaulich ordnet C_1 die beiden extremen Beobachtungen einem Cluster zu und die restlichen dem anderen Cluster; C_2 teilt die Beobachtungen in der Mitte auf die beiden Cluster auf. Hier gilt $W_n(C_1) \approx 11{,}2 < 46{,}4 \approx W_n(C_2)$ und $\tilde{W}_n(C_1) \approx 2{,}15 > 1{,}78 \approx \tilde{W}_n(C_2)$. W_n bevorzugt eine „Auslagerung" der extremen Beobachtungen X_{51}, X_{52} in einen Cluster,

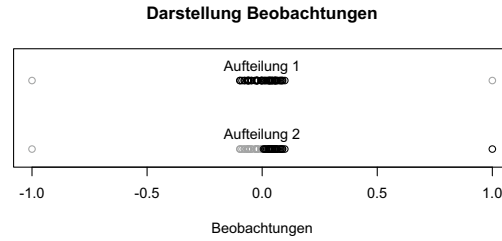


Abb. 9.2 Grafische Darstellung der Zuweisungsfunktionen C_1, C_2 aus Beispiel 9.7 durch die Farben Grau und Schwarz

da beispielsweise die Hinzunahme von X_{51} zu einem Cluster mit vielen Elementen mit dem Faktor $n_k(C)$ bestraft wird. Bei $\tilde{W}_n$ erfolgt solch eine Multiplikation nicht mehr, weswegen hier eine Zuordnung von X_{51} und X_{52} zu verschiedenen Clustern möglich ist und zur Reduktion des Wertes von $\tilde{W}_n(C)$ bevorzugt wird.

Die anschauliche Bedeutung der $m_k(C)$ wird durch den Übergang von $W_n(C)$ zu $\tilde{W}_n(C)$ nicht verändert. Die Betrachtung von $\tilde{W}_n(C)$ anstelle von $W_n(C)$ erfolgt nicht aus Gründen der Anschauung, sondern weil $\tilde{W}_n(C)$ eine einfachere näherungsweise Minimierung mit stabilen Verfahren erlaubt. Dazu verallgemeinert man das auf 9.6 basierende Optimierungsproblem zunächst, um auf bereits bekannte Techniken für eine approximative Lösung zurückgreifen zu können.

Lemma 9.8 (Verallgemeinertes k-means Optimierungsproblem) Sei

$$\breve{W}_n : \mathcal{C} \times \mathbb{R}^d \times \ldots \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}, \qquad \breve{W}_n(C, m_1, \ldots, m_K) := \sum_{k=1}^{K} \sum_{i:C(i)=k} \|X_i - m_k\|_2^2.$$

Dann gilt: $\hat{C}_n^* :\in \arg\min_{C \in \mathcal{C}} \tilde{W}_n(C)$ genau dann, wenn

$$(\hat{C}_n^*, \hat{m}_1^*, \ldots, \hat{m}_K^*) :\in \operatorname*{arg\,min}_{C \in \mathcal{C}, m_1, \ldots, m_K \in \mathbb{R}^d} \breve{W}_n(C, m_1, \ldots, m_K) \tag{9.7}$$

Die Lösung $\hat{C}_n^*$ heißt *optimale k-means-Clusterzuweisung,* die $\hat{m}_k^*$ heißen *optimale k-means-Clusterzentren.*

Beweis Wir ermitteln zunächst das Minimum von $(m_1, \ldots, m_K) \mapsto \breve{W}_n(C, m_1, \ldots, m_K)$ für festes $C \in \mathcal{C}$. Da K, C fixiert sind, verändert sich auch $n_k = n_k(C)$ $(k = 1, \ldots, K)$ nicht bei Variation von $m_1, \ldots, m_K$. Es gilt daher

$$\operatorname*{arg\,min}_{(m_1, \ldots, m_K)} \breve{W}_n(C, m_1, \ldots, m_K) = \Big(\operatorname*{arg\,min}_{m_k} \sum_{i:C(i)=k} \|X_i - m_k\|_2^2 \Big)_{k=1,\ldots,K}.$$

Sind x_i, $i \in A$ gegeben, so lautet der Minimierer von $z \mapsto \sum_{i \in A} \|x_i - z\|_2^2$ über $z \in \mathbb{R}$ gerade $z = \frac{1}{\#A} \sum_{i \in A} x_i$. Daher ist

$$\Big(\operatorname*{arg\,min}_{m_k} \sum_{i:C(i)=k} \|X_i - m_k\|_2^2 \Big)_{k=1,\ldots,K} = \big(m_k(C)\big)_{k=1,\ldots,K}.$$

Damit gilt

$$\min_{C \in \mathcal{C}, m_1, \ldots, m_K \in \mathbb{R}^d} \breve{W}_n(C, m_1, \ldots, m_K) = \min_{C \in \mathcal{C}} \breve{W}_n(C, m_1(C), \ldots, m_K(C)) = \min_{C \in \mathcal{C}} \tilde{W}_n(C).$$

Lemma 9.8 sagt aus, dass das neue Minimierungsproblem Gl. (9.7) zwar von mehr Parametern abhängt, der optimale Wert für $C \in \mathcal{C}$ aber mit demjenigen des ursprünglichen Optimierungsproblems $\hat{C}_n^* :\in \arg\min_{C\in\mathcal{C}} \breve{W}_n(C)$ übereinstimmt. In der folgenden Bemerkung betrachten wir die Struktur von Gl. (9.7) genauer.

Bemerkung 9.9 Die simultane Lösung von Gl. (9.7) in $C \in \mathcal{C}$, $m_1, \ldots, m_K \in \mathbb{R}^d$ ist schwierig. Allerdings existiert eine Zerlegung in zwei Teilprobleme, die leicht zu lösen sind:

a) Sind $m_1, \ldots, m_K$ fest, so ist ein Minimierer von $C \mapsto \breve{W}_n(C, m_1, \ldots, m_K)$ gegeben durch:
$$\tilde{C}(i) := \operatorname*{arg\,min}_{k\in\{1,\ldots,K\}} \|X_i - m_k\|_2^2,$$
d. h. weise jedes X_i dem Cluster k zu, bei welchem der Abstand zum Clusterzentrum m_k am geringsten ist.
(*Beweis:* Folgt direkt aus der Darstellung $\breve{W}_n(C, m_1, \ldots, m_K) = \sum_{i=1}^n \|X_i - m_{C(i)}\|_2^2$.)
b) Ist $C \in \mathcal{C}$ fest, so sind $\tilde{m}_k = m_k(C)$ $(k = 1, \ldots, K)$ Minimierer von $(m_1, \ldots, m_K) \mapsto \breve{W}_n(C, m_1, \ldots, m_K)$ (vgl. Beweis zu Lemma 9.8).

Für Probleme solcher Struktur existiert eine Standardmethode zur Ermittlung eines *lokalen* Minimums: Dieses erhält man, indem die Schritte (a),(b) wiederholt hintereinander ausgeführt werden. Damit erhalten wir folgendes Verfahren:

Algorithmus 9.10 (k-means-Iterationsverfahren) Seien $\hat{m}_k^{(0)} \in \mathbb{R}^d$ $(k = 1, \ldots, K)$ und $\hat{C}^{(0)} \in \mathcal{C}$ und $T \in \mathbb{N}$.
Für $t = 1, 2, \ldots, T$ (oder bis Konvergenz erreicht ist) wiederhole:

a) Für $i = 1, \ldots, n$, bestimme $\hat{C}^{(t)}(i) := \arg\min_{k\in\{1,\ldots,K\}} \|X_i - m_k^{(t-1)}\|_2^2$.
b) Für $k = 1, \ldots, K$, bestimme $\hat{m}_k^{(t)} = m_k(\hat{C}^{(t)}) = \frac{1}{\#\{i:\hat{C}^{(t)}(i)=k\}} \sum_{i:\hat{C}^{(t)}(i)=k} X_i$.

$\hat{C}_n^{K,kmeans} := \hat{C}^{(T)}$ heißt dann *lokal optimale k-means-Clusterzuweisung*, $\hat{m}_k^{K,kmeans} := \hat{m}_k^{(T)}$ *lokal optimale k-means-Clusterzentren.*
Jedes neu beobachtete $x \in \mathbb{R}^d$ wird dem Cluster
$$\hat{k}_n^{K,kmeans}(x) :\in \operatorname*{arg\,min}_{k\in\{1,\ldots,K\}} \left\|\hat{m}_k^{K,kmeans} - x\right\|_2^2 \tag{9.8}$$
zugeordnet.

Die Zuordnung Gl. (9.8) zerlegt den Raum $\mathbb{R}^d$ anschaulich in konvexe Polygone

$$P_k := \left\{ x \in \mathbb{R}^d : \|\hat{m}_k^{K,kmeans} - x\|_2 = \min_{j\in\{1,\ldots,K\}} \|\hat{m}_j^{K,kmeans} - x\|_2 \right\}, \quad k = 1, \ldots, K,$$

die sogenannten *Voronoi-Zellen;* jeder Cluster $k \in \{1, \ldots, K\}$ entspricht dann einem solchen Polygon, vgl. Abb. 9.3f oder Abb. 9.4.

Satz 9.11 In der Situation von Algorithmus 9.10 gilt:

$$\breve{W}_n(\hat{C}^{(t)}, \hat{m}_1^{(t)}, \ldots, \hat{m}_K^{(t)}) \downarrow \breve{W}_n(z^*) \quad (t \to \infty),$$

wobei z^* ein lokales Minimum von $\breve{W}_n$ ist.

Beweis In Bemerkung 9.9 haben wir gesehen, dass

$$\begin{aligned} \breve{W}_n(\hat{C}^{(t)}, \hat{m}_1^{(t)}, \ldots, \hat{m}_K^{(t)}) &\overset{(b)}{\leq} \breve{W}_n(\hat{C}^{(t)}, \hat{m}_1^{(t-1)}, \ldots, \hat{m}_K^{(t-1)}) \\ &\overset{(a)}{\leq} \breve{W}_n(\hat{C}^{(t-1)}, \hat{m}_1^{(t-1)}, \ldots, \hat{m}_K^{(t-1)}), \end{aligned}$$

d. h. $t \mapsto \breve{W}_n(\hat{C}^{(t)}, \hat{m}_1^{(t)}, \ldots, \hat{m}_K^{(t)})$ ist monoton fallend. Wegen $\breve{W}_n(\ldots) \geq 0$ folgt die Konvergenz mit dem Satz von Bolzano-Weierstraß. Für $R > 0$ groß genug gilt $m_1^*, \ldots, m_K^*, X_1, \ldots, X_n \in [-R, R]^d$. Sei $Z := \mathcal{C} \times ([-R, R]^d)^K$. Ausgestattet mit der Metrik $\tilde{D} : Z \times Z \to \mathbb{R}_{\geq 0}$, $\tilde{D}((C, m_1, \ldots, m_K), (C', m_1', \ldots, m_K')) = \sum_{i=1}^n |C(i) - C'(i)| + \sum_{k=1}^K D(m_k, m_k')$ ist Z folgenkompakt. Daher gibt es eine Teilfolge von $z_t := (\hat{C}^{(t)}, \hat{m}_1^{(t)}, \ldots, \hat{m}_K^{(t)}) \in Z$ mit Grenzwert $z^* \in Z$. Stetigkeit von $\breve{W}_n(\cdot)$ impliziert $W^* = \breve{W}_n(z^*)$. z^* ist ein lokales Minimum, sonst entsteht ein Widerspruch zur Optimalität in Schritt (a), (b).

Bemerkung 9.12

- Der Vorteil des k-means-Iterationsverfahrens aus Algorithmus 9.10 ist dessen schnelle Berechenbarkeit und dessen stabile Konvergenz, allerdings wird nur ein lokales Minimum von $\breve{W}_n$ ermittelt, das evtl. stark von den gewählten Startwerten $m_k^{(0)}$, $k = 1, \ldots, K$ abhängt.
- In der Praxis versucht man ein globales Minimum zu finden, indem der Algorithmus 9.10 mehrmals mit verschiedenen (zufällig oder heuristisch bestimmten) Startwerten $\hat{m}_k^{(0)}$, $k = 1, \ldots, K$ ausgeführt wird und anschließend die Lösungen $\hat{m}_k^{K,iter}$, $\hat{C}_n^{K,iter}$ mit den kleinsten $\breve{W}_n(\hat{C}_n^{K,iter}, \hat{m}_1^{K,iter}, \ldots, \hat{m}_K^{K,iter})$ zurückgegeben werden.
- Verallgemeinerung: Auch wenn die Herleitung von $\breve{W}_n$ auf Basis von $D(x, y) = \|x - y\|_2^2$ durchgeführt wurde, kann der Algorithmus für beliebige Abstandsfunktionen D durchgeführt werden mittels

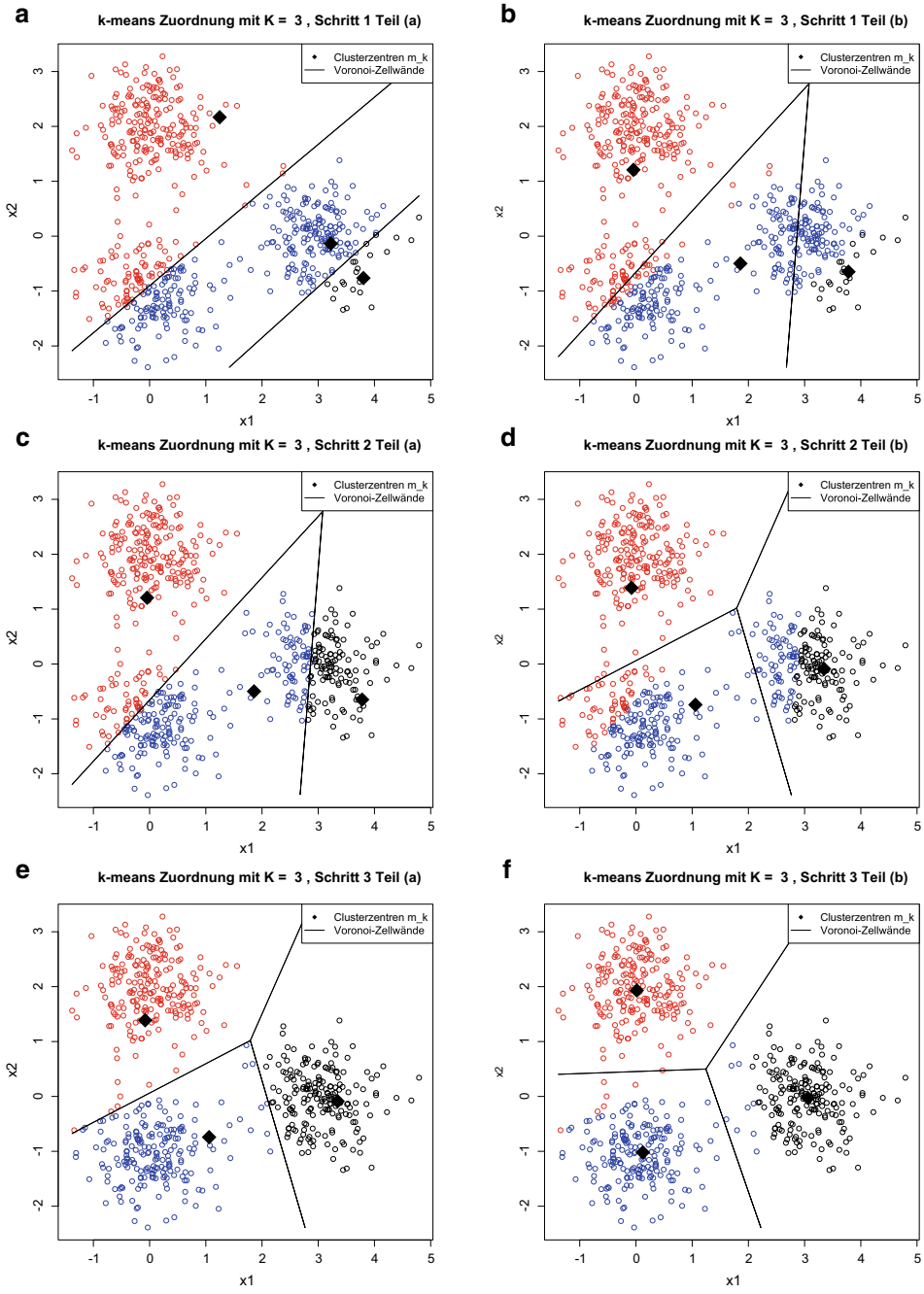


Abb. 9.3 k-means Clustering Algorithmus 9.10. **a**–**f** Darstellung der ersten drei Iterationsschritte mit $K = 3$ angewandt auf die Trainingsdaten von Beispiel 9.13 und der zugehörigen Voronoi-Zellen

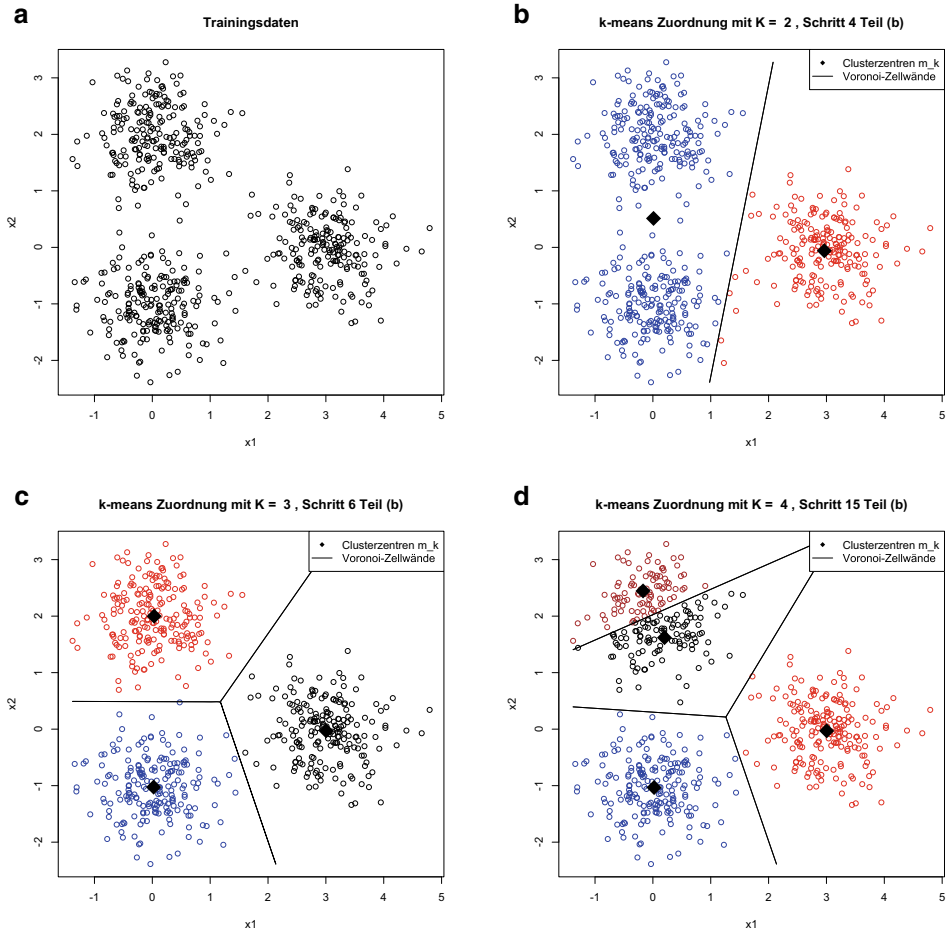


Abb. 9.4 k-means Clustering Algorithmus 9.10. **a** Darstellung der Trainingsdaten. **b**, **c**, **d** Darstellung des jeweils finalen Iterationsschritts des Algorithmus 9.10 angewandt mit $K \in \{2, 3, 4\}$ sowie der zugehörigen Voronoi-Zellen

$$\breve{W}_n(C, m_1, \ldots, m_K) = \sum_{k=1}^{K} \sum_{i:C(i)=k} D(X_i, m_k).$$

Die Schritte (a),(b) in Bemerkung 9.9 und Definition 9.10 ändern sich dann entsprechend zu

$$\tilde{C}(i) := \underset{k\in\{1,\ldots,K\}}{\arg\min}\ D(X_i, m_k), \qquad \tilde{m}_k = \underset{m}{\arg\min} \sum_{i:C(i)=k} D(X_i, m).$$

Eine Abstandsfunktion D verschieden von der quadrierten euklidischen Norm wird zum Beispiel verwendet, wenn Robustheit der Clustering-Prozedur gegenüber Ausreißern

erreicht werden soll; man wählt dann $D(x, x') = \|x - x'\|_1$ oder $D(x, x') = \min\{\|x - x'\|_1, 1\}$. Im ersten Fall ist dann $\tilde{m}_k$ ein mehrdimensionaler Median von $\{X_i : C(i) = k\}$.

Zur Illustration von Algorithmus 9.10 betrachten wir das nachstehende Beispiel.

Beispiel 9.13 Sei $K^* = 3$. Es seien (X_i, A_i), $i = 1, \ldots, n$ i.i.d. mit $\mathbb{P}(A_i = k) = \frac{1}{K^*}$, $k = 1, \ldots, K^*$ und

$$X_i \in \mathbb{R}^2, \quad X_i | Y_i \sim N(\mu_i, \Sigma),$$

wobei $\Sigma = 0{,}3 I_{2\times 2}$ und $\mu_1 = (0, 2)^T$, $\mu_2 = (0, -1)^T$, $\mu_3 = (3, 0)^T$. Wir generieren insgesamt $n = 600$ Beobachtungen. In Abb. 9.3 sind die ersten drei Iterationsschritte des Algorithmus 9.10 angewandt auf $(X_i)_{i=1,\ldots,n}$ mit $K = 3$ (dies entspricht der korrekten Anzahl an Clustern) und zufälligen Startwerten für m_1, m_2, m_3 dargestellt. Hier sieht man, dass eine schnelle Konvergenz gegen eine korrekte Einteilung der Daten in drei Cluster erfolgt. In Abb. 9.4 sind die ursprünglichen Trainingsdaten (ungefärbt) und die Finalzustände des Algorithmus 9.10 für jeweils zufällig bestimmte Startzentren für $K = 2$, $K = 3$ und $K = 4$ darstellt. Für $K = 2$ und $K = 4$ wird der Algorithmus mit einer „falschen" Anzahl an gewünschten Clusterzentren ausgeführt, konvergiert aber dennoch gegen eine Clusterzuweisung. Diese Clusterzuweisung hängt aber viel stärker von den Startzentren ab als bei einer korrekten Spezifikation mit $K = 3$, vgl. Abb. 9.5.

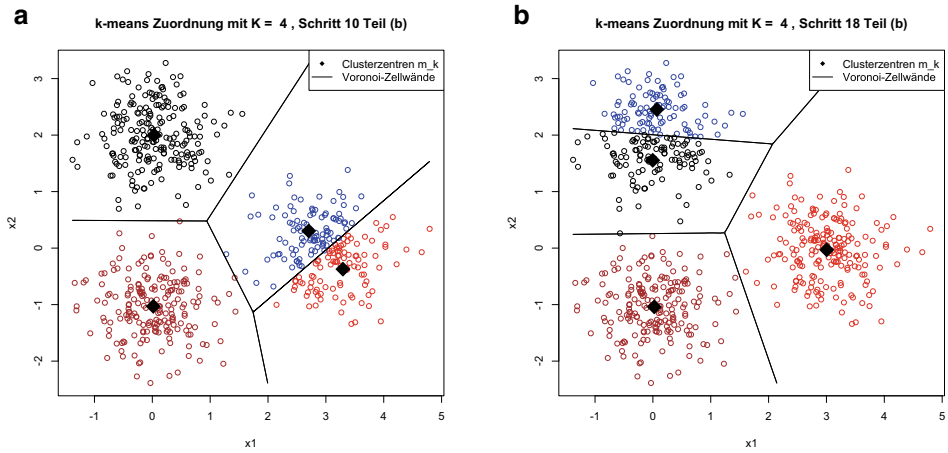


Abb. 9.5 k-means Clustering Algorithmus 9.10. **a**, **b** Darstellung des finalen Iterationsschritts des Algorithmus 9.10 angewandt mit $K = 4$ mit zwei verschiedenen, zufällig bestimmten Startzentren

9.1.4 Theoretische Resultate

In diesem Abschnitt wollen wir die Modellannahme und Konvergenzaussagen für die Clusterzentren $\hat{m}_k^*$, $k = 1, \ldots, K$ aus Lemma 9.8 formulieren. Aus Lemma 9.9(a) können wir folgende alternative Darstellung des Optimierungsproblems herleiten:

Lemma 9.14 (Verallgemeinertes k-means Optimierungsproblem 2) Sei

$$\bar{W}_n : \mathbb{R}^d \times \ldots \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}, \qquad \bar{W}_n(m_1, \ldots, m_K) := \sum_{i=1}^{n} \min_{k \in \{1,\ldots,K\}} \|X_i - m_k\|_2^2. \quad (9.9)$$

Dann gilt $(\hat{m}_1^*, \ldots, \hat{m}_K^*) \in \arg\min_{(m_1,\ldots,m_K) \in (\mathbb{R}^d)^K} \bar{W}_n(m_1, \ldots, m_K)$ genau dann, wenn

$$(\hat{C}_n^*, \hat{m}_1^*, \ldots, \hat{m}_K^*) :\in \operatorname*{arg\,min}_{(C,m_1,\ldots,m_K) \in \mathcal{C} \times (\mathbb{R}^d)^K} \breve{W}_n(C, m_1, \ldots, m_K).$$

Beweis Sei $m_1, \ldots, m_K$ fest. Laut Bemerkung 9.9(a) minimiert

$$\tilde{C}(i) := \operatorname*{arg\,min}_{k \in \{1,\ldots,K\}} \|X_i - m_k\|_2^2$$

den Ausdruck $C \mapsto \breve{W}_n(C, m_1, \ldots, m_K)$. In diesem Fall gilt

$$\begin{aligned} \breve{W}_n(\tilde{C}, m_1, \ldots, m_K) &= \sum_{i=1}^{n} \|X_i - m_{\tilde{C}(i)}\|_2^2 = \sum_{i=1}^{n} \min_{k \in \{1,\ldots,K\}} \|X_i - m_k\|_2^2 \\ &= \bar{W}_n(m_1, \ldots, m_K). \end{aligned}$$

Diese dritte Formulierung Gl. (9.9) des ursprünglichen Optimierungsproblems $\hat{C}_n^* \in \arg\min_{C \in \mathcal{C}} \tilde{W}_n(C)$ hat aus statistischer Sicht den Vorteil, dass die schlecht asymptotisch behandelbare Zuweisungsfunktion $C \in \mathcal{C}$ vollständig aus dem Optimierungsproblem entfernt ist. Die Argumente $m_1, \ldots, m_K \in \mathbb{R}^d$ sind von fester Dimension. Unter geeigneten Annahmen an die zugrunde liegende Verteilung $\mathbb{P}^X$ kann man erwarten, dass es „wahre" Clusterzentren $m_1^*, \ldots, m_K^* \in \mathbb{R}^d$ gibt, gegen die $\hat{m}_1^*, \ldots, \hat{m}_K^*$ für eine steigende Anzahl an Trainingsdaten (d. h. $n \to \infty$) konvergieren.

Definition 9.15 (Modellannahme: k-means Clustering)
Es gelte $\mathbb{E}[\|X\|_2^2] < \infty$ und es gebe eindeutig bestimmte $m_1^*, \ldots, m_K^*$ mit

$$\mathbb{E}\Big[\min_{k=1,\ldots,K} \|X - m_k^*\|_2^2\Big] = \inf_{m_1,\ldots,m_K \in \mathbb{R}^d} \mathbb{E}\Big[\min_{k=1,\ldots,K} \|X - m_k\|_2^2\Big].$$

♦

Die Modellannahme trifft für sehr viele Verteilungsmodelle von X zu. Dies bedeutet jedoch nicht automatisch, dass k-means-Clustering in jedem Fall gute Ergebnisse liefert. Der Grund ist, dass nur eine Aussage über die „wahren" *Clusterzentren* m_k^*, $k = 1, \ldots, K$, getroffen wird, nicht jedoch über die darauf basierende *Zuweisung* aus Gl. (9.8). Details geben wir in Bemerkung 9.17. Für die Qualität der Schätzung von $\hat{m}_k^*$ gilt das folgende Resultat aus [22].

Satz 9.16 Unter der Modellannahme aus Definition 9.15 gilt für

$$(\hat{m}_1^*, \ldots, \hat{m}_K^*) :\in \operatorname*{arg\,min}_{(m_1,\ldots,m_K)\in(\mathbb{R}^d)^K} \bar{W}_n(m_1, \ldots, m_K)$$

nach geeigneter Vertauschung der $\hat{m}_k^*$ untereinander die Eigenschaft

$$\forall k = 1, \ldots, K : \qquad \hat{m}_k^* \overset{n\to\infty}{\to} m_k^* \qquad \mathbb{P}\text{-}f.s.$$

Das bedeutet, die optimalen Clusterzentren basierend auf den Trainingsdaten konvergieren gegen die „wahren" optimalen Clusterzentren, falls immer mehr Trainingsdaten zur Verfügung stehen. Der besprochene Algorithmus 9.10 liefert mit $\hat{m}_k^{K,kmeans}$ jedoch nur eine Approximation für $\hat{m}_k^*$.

Bemerkung 9.17 (Nachteile des k-means-Clustering)

(P1) Die Wahl der Clusteranzahl K in Algorithmus 9.10 ist oft nur heuristisch möglich, da im Gegensatz zum Supervised Learning eine Kontrolle der Anpassungsqualität mittels Y_i nicht möglich ist.

(P2) Die Modellannahme aus Definition 9.15 erwartet lediglich, dass sich die Verteilung $\mathbb{P}^X$ um *eindeutig bestimmte* $m_1^*, \ldots, m_K^* \in \mathbb{R}^d$ konzentriert. Wie genau und in welcher Form sich die Verteilung um diese Punkte konzentriert, wird *nicht* spezifiziert (und müsste durch eine zusätzliche, speziellere Modellannahme geklärt werden, welche dann auch zur Erstellung des Algorithmus und der Zuweisung genutzt werden muss). Die Aufteilung des Raumes durch Gl. (9.8) führt notwendigerweise zu konvexen Clustern und kann sehr stark von der erwarteten Aufteilung (wie beispielsweise sich überlappende Gruppen oder ineinanderliegende Kreise, vgl. Abb. 9.6) abweichen, falls eine speziellere Struktur vorliegt. Das Problem ist dann, dass k-means-Clustering das zusätzliche Wissen nicht gewinnbringend in die Schätzung der Cluster bzw. Clusterzuweisungen einbringt. Insbesondere für hochdimensionale Daten kann dies problematisch sein.

Die optimale Zuweisungsfunktion basierend auf dem ursprünglichen Ansatz Gl. (9.3) würde nicht nur konvexe Cluster liefern. Dass k-means-Clustering nur noch konvexe Cluster liefert, liegt an der mathematischen Vereinfachung von $W_n(\cdot)$ zu $\tilde{W}_n(\cdot)$ in Gl. (9.6), vgl. auch Beispiel 9.7.

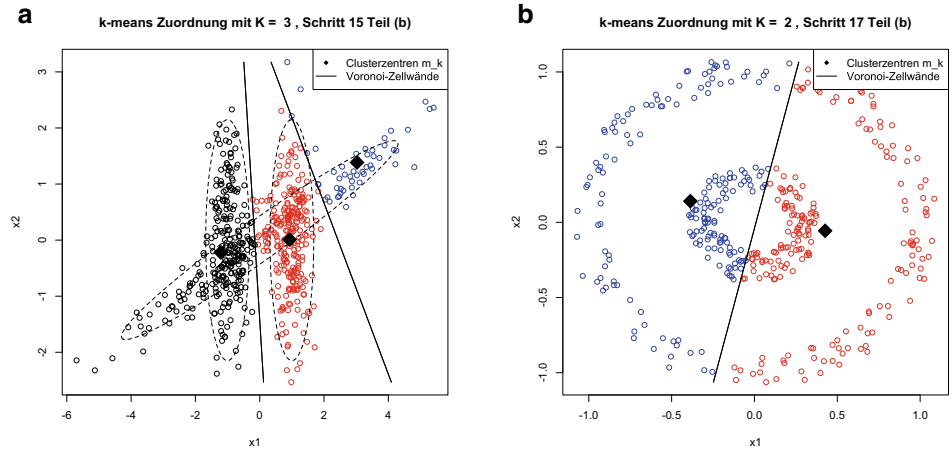


Abb. 9.6 k-means Clustering Algorithmus 9.10: Darstellung des finalen Iterationsschritts des Algorithmus 9.10 angewandt mit $K = 3$ bzw. $K = 2$ bei Trainingsdaten kommend von nicht-konvexen Clustern. **a** drei gemischte Normalverteilungen, vgl. Beispiel 9.27. Die gestrichelte Linie gibt jeweils eine Höhenlinie der drei zugrunde liegenden Normalverteilungen an und drückt damit die „erwarteten" Clusterzuweisungen aus. **b** Ineinanderliegende Kreise, vgl. Beispiel 10.23 mit $M^* = 2$ und $n = 400$ Trainingsdaten. Erwartet wird eine Aufteilung in zwei Cluster; den inneren und den äußeren Kreis

Wir konzentrieren uns im Folgenden auf (P2). In Abschn. 9.2 betrachten wir einen Lösungsansatz, welcher konkretere strukturelle Annahmen an die zugrunde liegende Verteilung $\mathbb{P}^X$ trifft und dadurch eine Schätzung von auf die Modellannahme angepassten Clustern erlaubt, die nicht notwendig konvex sind.

9.2 Clustering mit Mischungsverteilungen

Ansatz Wir geben eine parametrische Klasse von Wahrscheinlichkeitsverteilungen für $\mathbb{P}^X$ vor und ermitteln die passenden Parameter anhand der vorgegebenen Trainingsdaten. Im Gegensatz zu k-means-Clustering können sich hier die geschätzten Cluster auch überlappen und je nach gewählter Wahrscheinlichkeitsverteilung auch eine nichtkonvexe Form besitzen. In diesem Abschnitt verwenden wir aus Übersichtsgründen eine etwas laxe Bezeichnung für Dichten von Wahrscheinlichkeitsverteilungen. Eine Wahrscheinlichkeitsdichte wird grundsätzlich durch $f(\cdot)$ gekennzeichnet. Schreiben wir $f_p(z)$, so meinen wir damit die Wahrscheinlichkeitsdichte einer Zufallsvariable Z, welche eventuell von Parametern p abhängt. Das Argument von f gibt also Auskunft darüber, um die Dichte welcher Zufallsvariable es sich handelt, und die Subskripte drücken Parameterabhängigkeiten aus.

Modellannahme 9.18 (Parametrische Mischungsverteilungen) $\mathbb{P}^X$ sei stetig bzgl. des Lebesgue-Maßes auf $\mathbb{R}^d$ mit Dichte $f(x)$. Es gebe

- eine Familie von Dichten $\{p(x,\theta) : \theta \in \Theta\}$ bzgl. des Lebesgue-Maßes auf $\mathbb{R}^d$, wobei $\Theta \subset \mathbb{R}^s$ ein geeigneter Parameterraum sei,
- einen ausgezeichneten Parameter $\underline{\theta}^* = (\theta_1^*, \ldots, \theta_K^*) \in \Theta^K$,
- Wahrscheinlichkeiten $\underline{\pi}^* := (\pi_1^*, \ldots, \pi_K^*) \in \Delta_K := \{(\pi_1, \ldots, \pi_K) \in [0,1]^K : \sum_{k=1}^K \pi_k = 1\}$,

so dass

$$\forall x \in \mathbb{R}^d : \quad f(x) = f_{\underline{\theta}^*, \underline{\pi}^*}(x) = \sum_{k=1}^K \pi_k^* \cdot p(x, \theta_k^*).$$

Interpretation der Modellannahme: Wir vermuten, dass die Trainingsdaten X_i, $i = 1, \ldots, n$, aus K verschiedenen Verteilungen stammen. Das i-te Trainingdatum X_i entsteht durch eine Zwei-Schritt-Prozedur:

1. Mit Wahrscheinlichkeit π_k^* wird die Nummer $k \in \{1, \ldots, K\}$ der Verteilung ausgewählt, d. h. der Cluster, aus welchem die Realisierung gezogen wird.
2. X_i ist dann eine zufällige Realisierung der ausgewählten Verteilung $p(x, \theta_k^*)$.

Das prominenteste Beispiel ist der Fall, dass die Beobachtungen aus K verschiedenen Normalverteilungen stammen. Formal entspricht diese Modellannahme derjenigen der Linearen Diskriminanzanalyse aus Abschn. 4.1, jedoch mit unbeobachteten Klassen.

Beispiel 9.19 (Gemischte Normalverteilungen) In der Situation von Modell 9.18 wählen wir den Parameterraum

$$\Theta = \{\theta = (\mu, \Sigma) : \mu \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d \times d} \text{ symmetrisch positiv definit}\}$$

und die Familie der Dichten $\{p(\cdot, \theta) : \theta \in \Theta\}$ mit

$$p(x,\theta) = N(x, \mu, \Sigma) := \frac{1}{(2\pi \det(\Sigma))^{d/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right).$$

Im Fall gemischter Normalverteilungen nimmt man an, dass jede einzelne zu bestimmende Gruppe eine ellipsenförmige Struktur hat, deren Ort durch μ und deren Form durch Σ bestimmt wird.

Ansatz zur Ermittlung von $\hat{\theta}_1^*, \ldots, \hat{\theta}_K^*$: Die (unbekannten) Parameter werden aus den Trainingsdaten X_i mittels einer Maximum-Likelihood-Methode aus der Statistik bestimmt. Dazu sei $\underline{X}_n := (X_1, \ldots, X_n)$. $\underline{X}_n$ besitzt die Wahrscheinlichkeitsdichte

$$f_{\underline{\theta}^*,\underline{\pi}^*}(\underline{x}_n) = \prod_{i=1}^{n} f_{\underline{\theta}^*,\underline{\pi}^*}(x_i)$$

bzgl. des Lebesgue-Maßes. Mit den Abkürzungen $\underline{\theta} := (\theta_1, \ldots, \theta_K) \in \Theta^K$, $\underline{\pi} := (\pi_1, \ldots, \pi_K) \in \Delta_K$ nennen wir

$$L(\underline{\theta}, \underline{\pi}) := \log f_{\underline{\theta},\underline{\pi}}(\underline{X}_n) = \sum_{i=1}^{n} \log f_{\underline{\theta},\underline{\pi}}(X_i) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \cdot p(X_i, \theta_k) \right)$$

die *Likelihood-Funktion*. Bei der Maximum-Likelihood-Methode suchen wir die Parameter $\bar{\theta} \in \Theta^K$, $(\pi_1, \ldots, \pi_K) \in \Delta_K$, bei welchen die gemachten Beobachtungen $\underline{X}_n$ am wahrscheinlichsten gewesen wären. Mit der Interpretation von $L(\underline{\theta}, \underline{X}_n)$ als Wahrscheinlichkeit ergibt sich nachstehende Definition.

Definition 9.20
Der Maximierer

$$(\hat{\underline{\theta}}^*, \hat{\underline{\pi}}^*) :\in \underset{(\theta_1,\ldots,\theta_K)\in\Theta^K, (\pi_1,\ldots,\pi_K)\in\Delta_K}{\arg\max} L(\underline{\theta}, \underline{\pi})$$

mit $\hat{\underline{\theta}}^* = (\hat{\theta}_1^*, \ldots, \hat{\theta}_K^*)$, $\hat{\underline{\pi}}^* = (\hat{\pi}_1^*, \ldots, \hat{\pi}_K^*)$ heißt *Maximum-Likelihood-Schätzer* im Mischungsmodell. ♦

Im Gegensatz zur logistischen Regression (vgl. Abschn. 5.2) ist die Funktion $(\underline{\theta}, \underline{\pi}) \mapsto L(\underline{\theta}, \underline{\pi})$ nicht konkav. Zwar sind in einigen Modellen die Logarithmen der einzelnen Dichten $\theta \mapsto \log p(x, \theta)$ konkav, aber nicht die logarithmierten Linearkombinationen der Dichten, aus welchen L zusammengesetzt ist. Es gibt daher keine Garantien für ein globales Maximum von L. Zur Bestimmung von $(\hat{\underline{\theta}}^*, \hat{\underline{\pi}}^*)$ kann ein Gradientenverfahren genutzt werden, über dessen Konvergenz sind aber keine unmittelbaren Aussagen verfügbar. In der Praxis können solche Verfahren also beliebig instabil sein.

Im Folgenden versuchen wir daher nicht mehr, $(\hat{\underline{\theta}}^*, \hat{\underline{\pi}}^*)$ zu ermitteln, sondern suchen stattdessen nach näherungsweisen Lösungen, die gleichzeitig aber eine stabilere Berechnung erlauben. Ähnlich wie beim iterativen Algorithmus für k-means Clustering zerlegen wir das Optimierungsproblem in Teilprobleme.

Ansatz Für die theoretischen Berechnungen täuschen wir vor, dass mehr Informationen als nur die Trainingsdaten $X_i, i = 1, \ldots, n$ vorliegen, und entferne diese am Ende der theoretischen Berechnung wieder. Hier fügen wir Beobachtungen hinzu, welche uns Informationen darüber geben, aus welcher Verteilung X_i ausgewählt wurde. Anschaulich entkoppeln wir die Auswahl des Clusters $k \in \{1, \ldots, K\}$ und das Ziehen der Realisierung $X_i \sim p(\cdot, \theta_k^*)$ und nehmen an, dass wir anstatt nur X_i auch den ausgewählten Cluster durch eine Zufallsvariable Z_i beobachten.

Formal seien Z_i, $i = 1, \ldots, n$ Zufallsvariablen mit $\mathbb{P}(Z_i = z) = \pi_z^*$, $z \in \{1, \ldots, K\}$. Die zugehörige Dichte von Z_i bzgl. des Zählmaßes auf $\{1, \ldots, K\}$ ist dann $f_{\underline{\pi}^*}(z) = \pi_z^*$. Wir nehmen an, dass die X_i gegeben Z_i entstehen durch

$$X_i | Z_i \sim p(\cdot, \theta_{Z_i}^*).$$

Sei $\underline{Z}_n := (Z_1, \ldots, Z_n)$. Dann gilt für die gemeinsame Dichte von $(\underline{X}_n, \underline{Z}_n)$:

$$f_{\underline{\theta},\underline{\pi}}(\underline{x}_n, \underline{z}_n) = \prod_{i=1}^{n} \underbrace{f_{\underline{\theta},\underline{\pi}}(x_i, z_i)}_{= f_{\underline{\theta}}(x_i|z_i) \cdot f_{\underline{\pi}}(z_i)} = \prod_{i=1}^{n} p(x_i, \theta_{z_i}) \cdot \pi_{z_i} \tag{9.10}$$

Im Gegensatz zu $L(\underline{\theta}, \underline{\pi})$ ist hier eine wesentliche Vereinfachung der Struktur durch Logarithmierung möglich:

Lemma 9.21 Es gilt für beliebiges $\hat{\underline{\theta}} \in \Theta^K$, $\hat{\underline{\pi}} \in \Delta_K$:

$$L(\underline{\theta}, \underline{\pi}) \geq \sum_{\underline{z}_n} f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\underline{z}_n | \underline{X}_n) \cdot \log\left(\frac{f_{\underline{\theta},\underline{\pi}}(\underline{X}_n, \underline{z}_n)}{f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\underline{z}_n | \underline{X}_n)}\right) =: Q_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\underline{\theta}, \underline{\pi}) \tag{9.11}$$

mit Gleichheit genau dann, wenn $\underline{\theta} = \hat{\underline{\theta}}$, $\underline{\pi} = \hat{\underline{\pi}}$.

Beweis Es ist

$$\begin{aligned} L(\underline{\theta}, \underline{\pi}) &= \log f_{\underline{\theta},\underline{\pi}}(\underline{X}_n) \\ &= \log\Big(\sum_{\underline{z}_n \in \{1,\ldots,K\}^n} f_{\underline{\theta},\underline{\pi}}(\underline{X}_n, \underline{z}_n)\Big) = \log\Big(\sum_{\underline{z}_n} f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\underline{z}_n | \underline{X}_n) \cdot \frac{f_{\underline{\theta},\underline{\pi}}(\underline{X}_n, \underline{z}_n)}{f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\underline{z}_n | \underline{X}_n)}\Big). \end{aligned}$$

Da $f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\cdot | \underline{X}_n)$ stets eine Wahrscheinlichkeitsdichte ist, ergibt sich die Ungleichung (9.11) nun aus der Jensen-Ungleichung.

Bemerkung Lemma 9.21 liefert eine untere Schranke $Q_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\underline{\theta}, \underline{\pi})$ für die zu maximierende Funktion $L(\underline{\theta}, \underline{\pi})$ aus Definition 9.20. Diese untere Schranke ist vollständig bekannt, falls $\underline{X}_n = (X_1, \ldots, X_n)$ bekannt ist. In vielen Fällen lässt sie sich gut berechnen und leichter maximieren als $L(\underline{\theta}, \underline{\pi})$ selbst, da sie in den bekanntesten Modellen eine konkave Form besitzt (vgl. Algorithmus 9.22 bzw. Beispiel 9.26).

Die untere Schranke wird nun genutzt, um iterativ ein (lokales) Maximum von $L(\underline{\theta}, \underline{\pi})$ zu approximieren.

Definition 9.22 (EM-Algorithmus)
Sei $\hat{\underline{\theta}}^{(0)} \in \Theta^K$, $\hat{\underline{\pi}}^{(0)} \in \Delta_K$ beliebig.
Für $t = 0, 1, 2, \ldots$, wiederhole bis zur Konvergenz:

1. (E-Schritt) Berechne die Abbildung $(\underline{\theta}, \underline{\pi}) \mapsto Q_{\hat{\underline{\theta}}^{(t)}, \hat{\underline{\pi}}^{(t)}}(\underline{\theta}, \underline{\pi})$
2. (M-Schritt) Bestimme

$$(\hat{\underline{\theta}}^{(t+1)}, \hat{\underline{\pi}}^{(t+1)}) :\in \underset{\underline{\theta}, \underline{\pi}}{\arg\max}\, Q_{\hat{\underline{\theta}}^{(t)}, \hat{\underline{\pi}}^{(t)}}(\underline{\theta}, \underline{\pi}).$$

Sei $T \in \mathbb{N}$. Der *EM-Schätzer* von $(\underline{\theta}^*, \underline{\pi}^*)$ aus dem Modell 9.18 ist $(\hat{\underline{\theta}}^{EM,T}, \hat{\underline{\pi}}^{EM,T}) := (\hat{\underline{\theta}}^{(T)}, \hat{\underline{\pi}}^{(T)})$.
Jedes neu beobachtete $x \in \mathbb{R}^d$ wird dem Cluster

$$\hat{k}_n^{K,EM}(x) :\in \underset{k \in \{1,\ldots,K\}}{\arg\max}\, p(x, \hat{\theta}_k^{EM,T}) \tag{9.12}$$

zugewiesen, d. h. dem Cluster, bei welchem die zugehörige Wahrscheinlichkeitsdichte den größten Wert hat. ♦

Das Vorgehen ist in Abb. 9.7 für die ersten drei Schritte grafisch dargestellt. Jedes Paar $(\hat{\underline{\theta}}^{(t)}, \hat{\underline{\pi}}^{(t)})$ erzeugt eine untere Schranke für $L(\underline{\theta}, \underline{\pi})$, deren Maximierung eine weitere Näherung für ein lokales Maximum von $L(\underline{\theta}, \underline{\pi})$ liefert. Dabei gilt stets $Q_{\hat{\underline{\theta}}^{(t)}, \hat{\underline{\pi}}^{(t)}}(\hat{\underline{\theta}}^{(t)}, \hat{\underline{\pi}}^{(t)}) = L(\hat{\underline{\theta}}^{(t)}, \hat{\underline{\pi}}^{(t)})$, d. h. an der Stelle $(\hat{\underline{\theta}}^{(t)}, \hat{\underline{\pi}}^{(t)})$ stimmt die untere Schranke mit L überein.

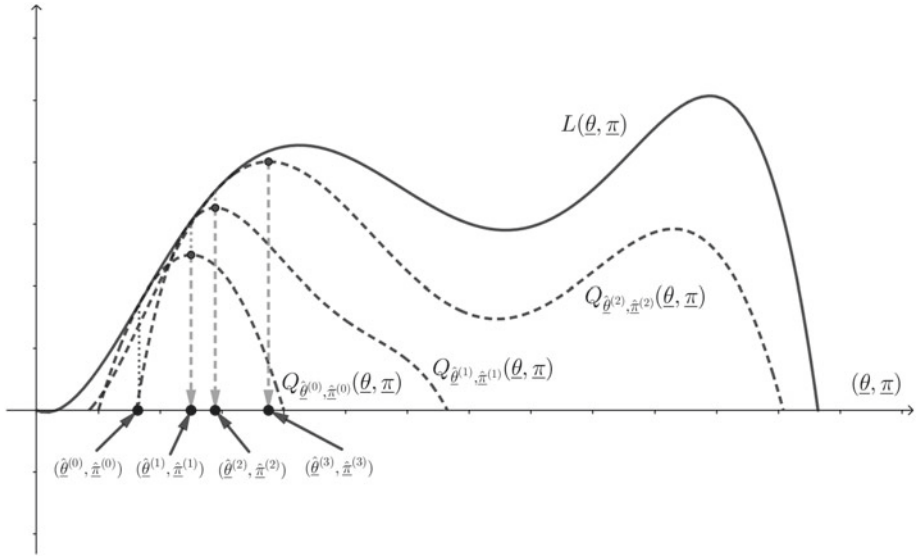


Abb. 9.7 Grafische Veranschaulichung des EM-Algorithmus 9.22 für die ersten drei Iterationsschritte

Bemerkung 9.23 (Zum EM-Algorithmus)

- „E“ steht für Erwartungswert *(expectation):* Die Berechnung der unteren Schranke Q ist die Berechnung einer Summe, welche als Erwartungswert bzgl. $\underline{Z}_n$ gegeben $\underline{X}_n$ aufgefasst werden kann. „M“ steht für Maximierung *(maximization).*
- Aufgrund der Gleichheitsaussage in Lemma 9.21 ist die untere Schranke in einer Umgebung von $\hat{\underline{\theta}}, \hat{\underline{\pi}}$ nahe an der tatsächlichen Funktion $\log f_{\underline{\theta},\underline{\pi}}(\underline{X}_n)$. Somit ist davon auszugehen, dass ein Iterationsschritt $\underline{\theta}^{(t+1)}, \underline{\pi}^{(t+1)}$ in die „richtige Richtung“ schiebt (vgl. Abb. 9.7).

Es gilt das folgende allgemeine Konvergenzresultat (vgl. [20, Section 6.4, Theorem 4.12]):

Satz 9.24 Ist $(\hat{\underline{\theta}}, \hat{\underline{\pi}}, \underline{\theta}, \underline{\pi}) \mapsto Q_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\underline{\theta}, \underline{\pi})$ stetig, so gilt

$$(\hat{\underline{\theta}}^{EM,T}, \hat{\underline{\pi}}^{EM,T}) \to (\hat{\underline{\theta}}^{\circ}, \hat{\underline{\pi}}^{\circ}) \qquad (T \to \infty),$$

wobei $(\hat{\underline{\theta}}^{\circ}, \hat{\underline{\pi}}^{\circ})$ ein lokales Maximum oder ein Sattelpunkt von $L(\underline{\theta}, \underline{\pi})$ ist.

Beachte $(\hat{\underline{\theta}}^{EM,T}, \hat{\underline{\pi}}^{EM,T})$ ist in diesem Sinne eine ‚doppelte‘ Approximation des wahren Werts $(\underline{\theta}^*, \underline{\pi}^*)$: $(\hat{\underline{\theta}}^{EM,T}, \hat{\underline{\pi}}^{EM,T})$ approximiert ein lokales Maximum der Funktion $L(\underline{\theta}, \underline{\pi})$, welche selbst nur eine Näherung einer Funktion ist, die $(\underline{\theta}^*, \underline{\pi}^*)$ als globales Maximum besitzt.

Die Berechnung der unteren Schranke im E-Schritt von Algorithmus 9.22 kann mittels der folgenden Formel durchgeführt werden:

Lemma 9.25 Es gilt:

$$Q_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\underline{\theta}, \underline{\pi}) = \sum_{i=1}^{n} \sum_{z_i=1}^{K} \Big(\log p(X_i, \theta_{z_i}) + \log \pi_i \Big) \cdot f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(z_i | X_i) - \text{const.}(\hat{\underline{\theta}}, \hat{\underline{\pi}})$$

und

$$f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(z_i | X_i) = \frac{p(X_i, \hat{\theta}_{z_i}) \hat{\pi}_{z_i}}{\sum_{z=1}^{K} p(X_i, \hat{\theta}_z) \hat{\pi}_z}$$

Beweis Zunächst ist

$$Q_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\underline{\theta}, \underline{\pi}) = \sum_{\underline{z}_n} f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\underline{z}_n | \underline{X}_n) \cdot \log f_{\underline{\theta},\underline{\pi}}(\underline{X}_n, \underline{z}_n) - \text{const.}(\hat{\underline{\theta}}, \hat{\underline{\pi}}),$$

es genügt also, den ersten Teil zu berechnen. Es gilt

$$f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\underline{z}_n|\underline{X}_n) = \prod_{j=1}^{n} f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(z_j|X_j)$$

Zusammen mit Gl. (9.10) folgt:

$$\begin{aligned}
&\sum_{\underline{z}_n} f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(\underline{z}_n|\underline{X}_n) \cdot \log f_{\underline{\theta},\underline{\pi}}(\underline{X}_n, \underline{z}_n) \\
&= \sum_{\underline{z}_n} \prod_{j=1}^{n} f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(z_j|X_j) \cdot \sum_{i=1}^{n} \log\Big(p(x_i, \theta_{z_i}) \cdot \pi_{z_i}\Big) \\
&= \sum_{i=1}^{n} \sum_{z_i=1}^{K} \log\Big(p(X_i, \theta_{z_i}) \cdot \pi_{z_i}\Big) \cdot f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(z_i|X_i) \prod_{j=1, j\neq i}^{n} \underbrace{\sum_{z_j=1}^{K} f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(z_j|X_j)}_{=1} \\
&= \sum_{i=1}^{n} \sum_{z_i=1}^{K} \Big(\log p(X_i, \theta_{z_i}) + \log \pi_i\Big) \cdot f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(z_i|X_i)
\end{aligned}$$

Die Darstellung von $f_{\hat{\underline{\theta}},\hat{\underline{\pi}}}(z_i|X_i)$ folgt aus der Bayes-Regel für bedingte Wahrscheinlichkeitsdichten.

Für das Beispiel 9.19 der gemischten Normalverteilungen können die Iterationsschritte des EM-Algorithmus explizit berechnet werden:

Beispiel 9.26 (Gemischte Normalverteilungen, Fortsetzung Beispiel 9.19) In der Notation von Satz 9.22 gilt mit den zusätzlichen eingeführten Hilfsvariablen $\hat{w}_{k,i}^{(t)}$, welche anschaulich die Einflüsse der einzelnen Trainingsdaten X_i auf die Variablen $\hat{\mu}_k^{(t)}$, $\hat{\Sigma}_k^{(t)}$ der k-ten Mischungsverteilung darstellen:

$$\begin{aligned}
\hat{w}_{k,i}^{(t+1)} &= \frac{N(X_i, \hat{\mu}^{k,(t)}, \hat{\Sigma}^{k,(t)}) \cdot \hat{\pi}_k^{(t)}}{\sum_{l=1}^{K} N(X_i, \hat{\mu}^{l,(t)}, \hat{\Sigma}^{l,(t)}) \cdot \hat{\pi}_l^{(t)}}, \\
\hat{\pi}_k^{(t+1)} &= \frac{1}{n} \sum_{i=1}^{n} \hat{w}_{k,i}^{(t+1)}, \\
\hat{\mu}_k^{(t+1)} &= \frac{\sum_{i=1}^{n} \hat{w}_{k,i}^{(t+1)} \cdot X_i}{\sum_{i=1}^{n} \hat{w}_{k,i}^{(t+1)}}, \\
\hat{\Sigma}_k^{(t+1)} &= \frac{\sum_{i=1}^{n} \hat{w}_{k,i}^{(t+1)} \cdot (X_i - \hat{\mu}_k^{(t+1)}) \cdot (X_i - \hat{\mu}_k^{(t+1)})}{\sum_{i=1}^{n} \hat{w}_{k,i}^{(t+1)}}
\end{aligned}$$

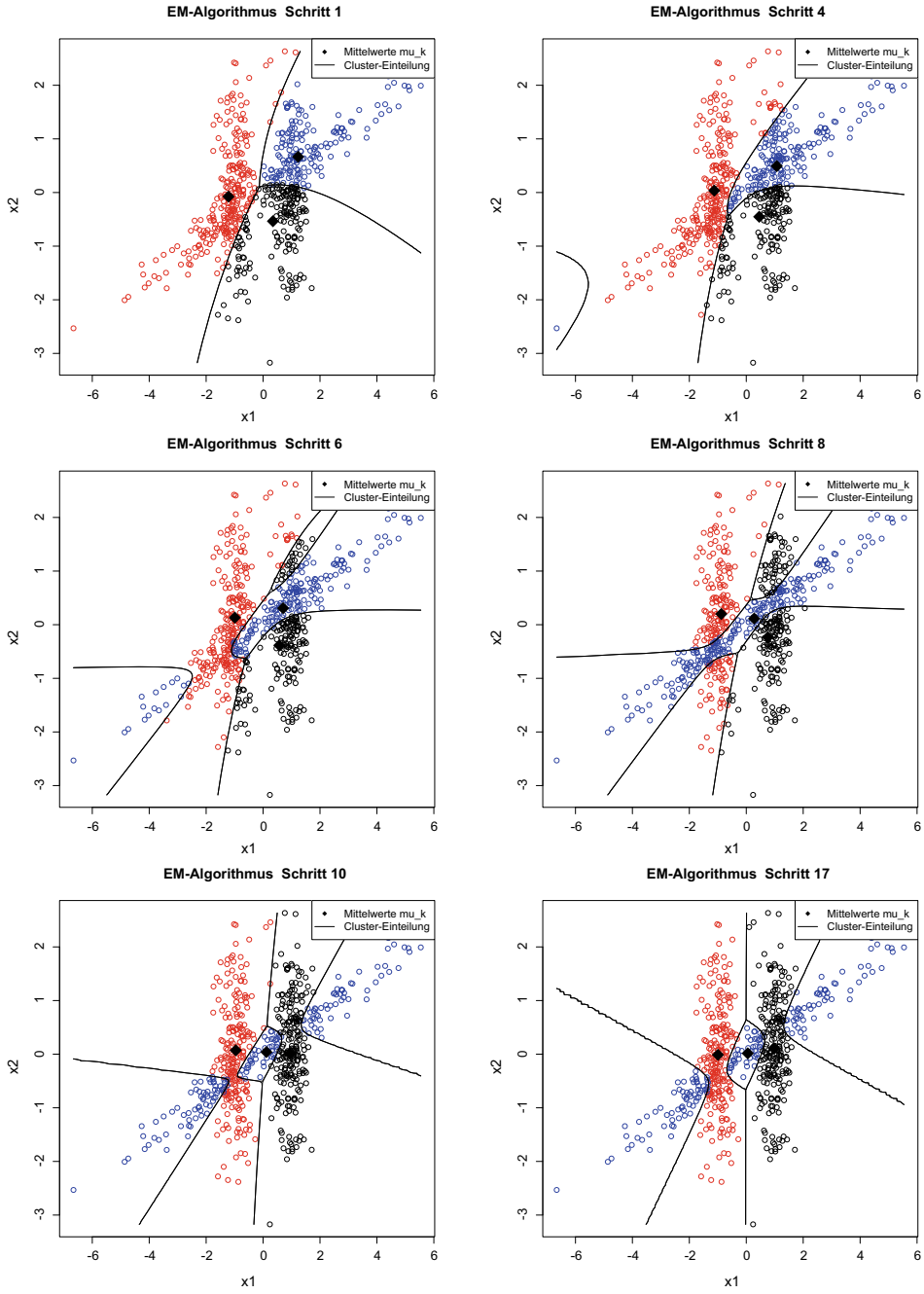


Abb. 9.8 EM-Algorithmus: Darstellung einiger Iterationsschritte mit $K = 3$ angewandt auf die Trainingsdaten von Beispiel 9.27 und die zugehörige Aufteilung des Raums in Cluster mit drei verschiedenen Farben

Für kompliziertere Modelle für $p(x, \theta)$ muss die Funktion Q im E-Schritt (vgl. Definition 9.22) durch ein Monte-Carlo-Verfahren angenähert werden. Wir betrachten nun ein Beispiel für den Spezialfall gemischter Normalverteilungen (vgl. 9.19).

Beispiel 9.27 (Gemischte Normalverteilungen mit konkreten Werten) Wir betrachten $K^* = 3$ gemischte Normalverteilungen mit

$$\mu_1^* = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \quad \mu_2^* = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \mu_3^* = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \Sigma_1^* = \Sigma_2^* = \begin{pmatrix} 0.1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \Sigma_3^* = \begin{pmatrix} 5 & 2 \\ 2 & 0.85 \end{pmatrix}$$

In Abb. 9.8 sind einige Iterationsschritte des EM-Algorithmus dargestellt sowie die Aufteilung des Raums $\mathbb{R}^d$ in K Cluster gemäß Gl. (9.12). Im Gegensatz zum k-means-Clustering (vgl. Abb. 9.6) werden die Cluster hier korrekt erkannt.

Bemerkung 9.28 (Nachteile des EM-Algorithmus)

- Der EM-Algorithmus basiert auf parametrischen Wahrscheinlichkeitsverteilungen, d. h. man muss bereits eine grobe Vorstellung haben, wie die Form der Cluster bzw. die Verteilung der Daten in den einzelnen Clustern aussieht. Verwendet man eine zu restriktive Klasse von Verteilungen, können die Cluster nicht richtig erkannt werden (vgl. Abb. 9.9b).

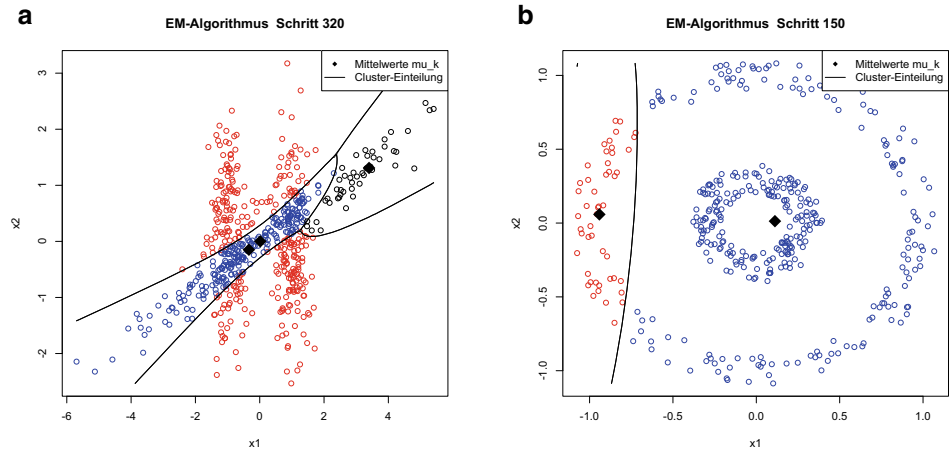


Abb. 9.9 Darstellung des finalen Iterationsschritts des EM-Algorithmus, basierend auf der Annahme mischender Normalverteilungen, und die zugehörige Aufteilung des Raums in Cluster mit drei verschiedenen Farben. **a** Trainingsdaten aus Beispiel 9.27. Hier wurden Startwerte gewählt, bei welchen der EM-Algorithmus mit $K = 3$ nur gegen ein lokales (nicht globales) Maximum konvergiert und die Cluster nicht korrekt identifiziert werden. **b** $n = 400$ Trainingsdaten aus Beispiel 10.23 mit $M^* = 2$. Anwendung des EM-Algorithmus mit $K = 2$ liefert nicht die „erwarteten" zwei Cluster (innerer Kreis und äußerer Kreisring)

- Selbst im Falle einer korrekten Spezifikation liefert der EM-Algorithmus laut Satz 9.24 nur Konvergenz gegen ein lokales Maximum von L (vgl. Abb. 9.9a). Wie beim k-means-Clustering empfiehlt es sich daher, den EM-Algorithmus mit mehreren verschiedenen Startwerten durchzuführen und den besten Schätzer anhand des größten erreichten Funktionswerts $L(\hat{\underline{\theta}}^{EM,T}, \hat{\underline{\pi}}^{EM,T})$ zu wählen.
- Flexible Clusterstrukturen sind nur mit relativ komplizierten Verteilungen mit vielen Parametern möglich; dann ist die untere Schranke Q evtl. nicht einfach berechenbar und muss approximiert werden; dies führt im Allgemeinen zu instabileren Iterationsschritten.
- Bereits aus der Modellannahme ist ersichtlich, dass die Anzahl K^* der Cluster bekannt sein muss.

Unsupervised Learning: Dimensionsreduktion 10

Inhaltsverzeichnis

In diesem Kapitel seien stets i. i. d. Trainingsdaten X_i, $i = 1, ..., n$ gegeben.

Im Allgemeinen arbeiten sowohl Verfahren des Supervised Learnings als auch die in Kap. 9 vorgestellten Verfahren des Unsupervised Learnings besser, wenn die Trainingsdaten X_i, $i = 1, ..., n$ eine *geringe Dimension d* besitzen und die einzelnen Komponenten *stochastisch unabhängig voneinander* sind. Beide Eigenschaften führen zu schnelleren Konvergenzraten, da die Bayes-Regeln (beim Supervised Learning) eine einfachere Gestalt aufweisen bzw. die Struktur der Daten einfacher und die Anzahl der vorliegenden Parameter geringer ist (bei den Verfahren aus Kap. 9). In diesem Kapitel stellen wir mit der Hauptkomponentenanalyse und dem spektralen Clustern zwei Methoden vor, welche die ursprünglichen Trainingsdaten X_i, $i = 1, ..., n$ transformieren und ggf. in der Dimension reduzieren, so dass eine Anwendung mit den bisher eingeführten Verfahren des Supervised/Unsupervised Learnings erfolgsversprechender ist.

10.1 Hauptkomponentenanalyse (PCA)

In Abb. 10.1a, b sind Trainingsdaten (vgl. Beispiel 10.6 und Beispiel 10.15 für das mathematische Modell) dargestellt, bei welchen sich eine besondere Struktur in deren Lage erkennen lässt. Diese sind jeweils gut durch Repräsentanten einer Teilmenge $M \subset \mathbb{R}^d$ geringerer Dimension approximierbar; hier konkret mit $d = 2$ lauten diese

S. Richter, *Statistisches und maschinelles Lernen*,
https://doi.org/10.1007/978-3-662-59354-7_10

$$
\begin{aligned}
M_1 &= \{x = \alpha_1 \cdot (1,1)^T : \alpha_1 \in \mathbb{R}\},\\
M_2 &= \{x = (x_1, x_2)^T \in \mathbb{R}^2 : x_1^2 + x_2^2 = 1\}\\
&= \left\{ \begin{pmatrix} \cos(\alpha_1) \\ \sin(\alpha_1) \end{pmatrix} : \alpha_1 \in [-\pi, \pi] \right\}.
\end{aligned} \tag{10.1}
$$

Wir sprechen im Folgenden von einer Menge M „geringerer Dimension“, falls sich die Punkte auf M mit $k < d$ Parametern eindeutig beschreiben lassen; eine mathematisch exakte Definition erfolgt durch den Begriff der *Mannigfaltigkeit,* auf den wir hier nicht weiter eingehen. Für M_1 und M_2 gilt jeweils $k = 1$. Mit der Hauptkomponentenanalyse (engl. *principal component analysis, PCA*) verfolgen wir das Ziel, diese Teilmengen M auf Basis der Trainingsdaten zu erkennen. Eine natürliche Reduktion der Trainingsdaten wird dann erreicht, indem jedes X_i durch den am nächsten liegenden Punkt $Z_i \in M$ (d. h. durch eine Projektion auf M) ersetzt wird. Wie aus Gl. (10.1) ersichtlich ist es dann möglich, Z_i durch Punkte $\alpha_{i1} \in \mathbb{R}^1$ in einer niedrigeren Dimension zu beschreiben. In Abb. 10.1c, d sind die projizierten Punkte Z_i dargestellt; in Abb. 10.1e, f sind die auf eine Dimension reduzierten Trainingsdaten α_{i1} dargestellt.

Für sich betrachtet stellt die PCA damit nur eine *Vorbearbeitung* der Daten dar, auf welche dann Methoden des Supervised/Unsupervised Learnings angewandt werden. Die Hoffnung ist, dass die reduzierten Daten α_{i1} immer noch die wesentlichen Informationen (d. h. die Lage) der ursprünglichen X_i in sich tragen (nur in komprimierter Form), so dass die darauf angewandten Methoden sich nicht oder nur noch sekundär um die Auswahl geeigneter Komponenten aus den Trainingsdaten α_{i1} konzentrieren müssen. In Abschn. 10.1.1 leiten wir mit Hilfe dieser Motivation einen Ansatz zur Berechnung der α_{i1} aus den ursprünglichen Trainingsdaten X_i her.

10.1.1 Ansatz

Wir betrachten hier den Fall, dass sich die Trainingsdaten gut durch Punkte auf einer Geraden

$$g(\alpha) = \mu + \alpha \cdot v, \alpha \in \mathbb{R}, \qquad (\mu \in \mathbb{R}^d, v \in \mathbb{R}^d)$$

approximieren lassen. Damit der Vektor v eindeutig bestimmt ist, fordern wir die Normierung $\|v\|_2^2 = 1$. Wir leiten nun eine mathematische Formulierung her, um diese Gerade zu identifizieren und die Trainingsdaten auf diese zu projizieren (vgl. Abb. 10.1a, c, e). Messen wir die Qualität der Projektion auf die Gerade mit Hilfe des euklidischen Abstands, so entspricht das Finden der Gerade der Lösung des folgenden Optimierungsproblems:

Definition 10.1 (PCA: Optimierungsproblem 1. Hauptkomponente)
Das Optimierungsproblem zur Bestimmung der 1. Hauptkomponente $v \in \mathbb{R}^d$ lautet

$$\min_{\alpha = (\alpha_1, \ldots, \alpha_n)^T \in \mathbb{R}^n, \mu \in \mathbb{R}^d, v \in \mathbb{R}^d} \hat{R}_n(\mu, \alpha, v) \quad \text{unter NB} \quad \|v\|_2^2 = 1, \tag{10.2}$$

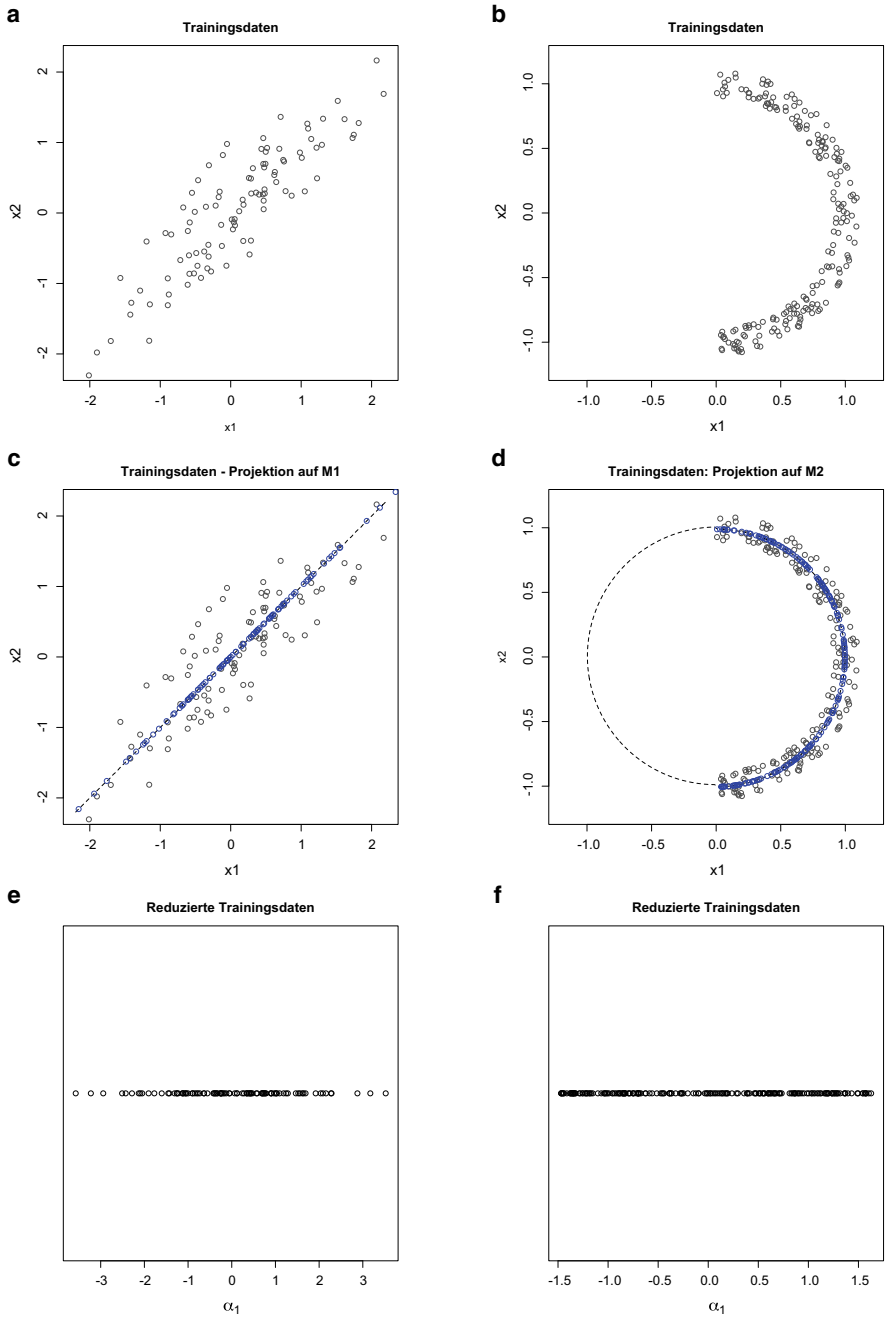


Abb. 10.1 **a**, **b** $n = 200$ Trainingdaten aus den Beispielen 10.6 **a** und 10.15 **b**. **c**, **d** Darstellung der Mengen M_1, M_2 aus Gl. (10.1) als gestrichelte Linie und (Orthogonal-)Projektion der Trainingsdaten auf diese Mengen (blau). **e**, **f** Erfolgte Reduktion α_{i1} der Trainingsdaten X_i auf eine Dimension

wobei

$$\hat{R}_n(\mu, \alpha, v) := \frac{1}{n} \sum_{i=1}^{n} \|X_i - (\mu + \alpha_i v)\|_2^2$$

den *empirischen Rekonstruktionsfehler* von X_i aus α_i bezeichnet. ♦

Halten wir $v \in \mathbb{R}^d$ zunächst fest, so können durch Ableiten die Extrema von $\hat{R}_n(\mu, \alpha, v)$ in μ, α bestimmt werden; der Ausdruck wird minimal für

$$\hat{\alpha}_i := v^T (X_i - \hat{\mu}_n), \quad \hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} X_i. \tag{10.3}$$

Dann gilt wegen $\|v\|_2^2 = 1$:

$$\begin{aligned}
\hat{R}_n(\hat{\mu}_n, \hat{\alpha}, v) &= \frac{1}{n} \sum_{i=1}^{n} \|(X_i - \hat{\mu}_n) - \alpha_i v\|_2^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} \|X_i - \hat{\mu}_n\|_2^2 - 2 \sum_{i=1}^{n} \alpha_i \underbrace{(X_i - \hat{\mu}_n)^T v}_{=\alpha_i} + \frac{1}{n} \sum_{i=1}^{n} \alpha_i^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} \|X_i - \hat{\mu}_n\|_2^2 - \frac{1}{n} \sum_{i=1}^{n} \alpha_i^2
\end{aligned}$$

Wegen $\frac{1}{n} \sum_{i=1}^{n} \alpha_i^2 = v^T \hat{\Sigma} v$ mit $\hat{\Sigma} := \frac{1}{n} \sum_{i=1}^{n} (X_i - \hat{\mu}_n)(X_i - \hat{\mu}_n)^T$ (die *empirische Kovarianzmatrix* oder *scatter matrix*) vereinfacht sich das Optimierungsproblem aus Gl. (10.2) zu:

$$\max_{v \in \mathbb{R}^d} v^T \hat{\Sigma} v \quad \text{unter NB} \quad \|v\|_2^2 = 1 \tag{10.4}$$

Dies ist ein konvexes Optimierungsproblem mit existierendem globalem Minimum. Eine Lösung bestimmen wir durch Aufstellen der Lagrange-Funktion

$$\ell(v, \lambda) := -v^T \hat{\Sigma} v + \lambda(\|v\|_2^2 - 1).$$

Hier gilt

$$\partial_v \ell(v, \lambda) = 0 \iff \hat{\Sigma} v = \lambda v,$$

d. h. der Minimierer von $\ell(v, \lambda)$ in v ist ein Eigenvektor $\hat{v}(\lambda)$ mit Länge 1 der Matrix $\hat{\Sigma}$ zum Eigenwert λ. Für $\lambda \in \mathbb{R}$ erhalten wir wegen $\|v\|_2^2 = 1$:

$$\hat{v}(\lambda)^T \hat{\Sigma} \hat{v}(\lambda) = \lambda,$$

d. h. die Zielfunktion in Gl. (10.4) wird maximal für den größtmöglichen Eigenwert λ von $\hat{\Sigma}$. Wir halten fest:

Lemma 10.2 Ein globaler Minimierer $(\hat{\alpha}, \hat{\mu}, \hat{v})$ des Optimierungsproblems aus Gl. (10.2) ist wie folgt gegeben:

- $\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^n X_i$ der Mittelwert der Beobachtungen,
- $\hat{v}$ ist der Eigenvektor mit Länge 1 zum größten Eigenwert von $\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^n (X_i - \hat{\mu}_n)(X_i - \hat{\mu}_n)^T$,
- $\hat{\alpha}_i := \hat{v}^T(X_i - \hat{\mu}_n)$ ist die Projektion von X_i auf die Gerade $\{\hat{\mu}_n + \alpha \cdot \hat{v} : \alpha \in \mathbb{R}\}$, vgl. Gl. (10.3).

Möglicherweise werden die Trainingsdaten X_i durch $\hat{\alpha}_i$ nicht ausreichend gut repräsentiert, d. h. der empirische Rekonstruktionsfehler ist zu hoch. Für hohe Dimensionen d ist es dann naheliegend, X_i stattdessen auf k-dimensionale Hyperebenen

$$g(\alpha) = \mu + \alpha_1 \cdot v^{(1)} + \ldots + \alpha_k \cdot v^{(k)}, \quad \alpha = (\alpha_1, \ldots, \alpha_k)^T \in \mathbb{R}^k$$

zu projizieren, wobei $\mu \in \mathbb{R}^d$ der *Stützvektor* ist und die *Richtungsvektoren* $v^{(1)}, \ldots, v^{(k)} \in \mathbb{R}^d$ auf Länge 1 normiert sind.

Definition 10.3 (PCA: Optimierungsproblem k Hauptkomponenten)
Sei $k \in \mathbb{N}$, $1 \leq k \leq d$. Das Optimierungsproblem zur Bestimmung der ersten k Hauptkomponenten $v^{(1)}, \ldots, v^{(k)}$ lautet

$$\min_{\alpha \in \mathbb{R}^{n\times k}, \mu \in \mathbb{R}^d, v^{(1)}, \ldots, v^{(k)} \in \mathbb{R}^d} \hat{R}_n(\mu, \alpha, v) \quad \text{unter NB} \quad \text{Für alle } j = 1, \ldots, k: \ \|v^{(j)}\|_2^2 = 1, \tag{10.5}$$

wobei

$$\hat{R}_n(\mu, \alpha, v) := \frac{1}{n}\sum_{i=1}^n \left\| X_i - \left(\mu + \sum_{j=1}^d \alpha_{ij} v^{(j)}\right)\right\|_2^2, \qquad \alpha = (\alpha_{ij})_{i\in\{1,\ldots,n\}, j\in\{1,\ldots,k\}}$$

den *empirischen Rekonstruktionsfehler* von X_i aus α bezeichnet. ♦

Man kann ähnlich wie bei Lemma 10.2 zeigen, dass dieser Ansatz die folgenden Lösungen besitzt.

Lemma 10.4 Sei $k \in \mathbb{N}$, $1 \leq k \leq d$. Ein globaler Minimierer $(\hat{\alpha}, \hat{\mu}_n, \hat{v}^{(1)}, \ldots, \hat{v}^{(k)})$ des Optimierungsproblems aus Gl. (10.5) ist wie folgt gegeben:

- $\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^n X_i$ ist der Mittelwert der Beobachtungen,
- sind $\hat{\lambda}_1 \geq \ldots \geq \hat{\lambda}_k$ die k größten Eigenwerte von $\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^n (X_i - \hat{\mu}_n)(X_i - \hat{\mu}_n)^T$, so sind $\hat{v}^{(1)}, \ldots, \hat{v}^{(k)}$ die zugehörigen Eigenvektoren mit Länge 1 (diese bilden ein Orthonormalsystem),

- $\hat{\alpha}_{ij} := (\hat{v}^{(j)})^T (X_i - \hat{\mu}_n)$ ist die Projektion von X_i auf die Gerade $\{\hat{\mu}_n + \alpha \cdot \hat{v}^{(j)} : \alpha \in \mathbb{R}\}$, vgl. Gl. (10.3).

Bemerkungen

- Ist X_1 auf $\mathbb{R}^d$ stetig verteilt, so gilt f.s. $\text{Rang}(\hat{\Sigma}) = \min\{d, n-1\}$ und die $\min\{d, n-1\}$ größten Eigenwerte von $\hat{\Sigma}$ sind f.s. verschieden. Solange also $k \leq \min\{d, n-1\}$ gilt, ist es im Allgemeinen möglich, die k größten Eigenwerte eindeutig der Größe nach zu ordnen.
- Die Eigenvektoren $\hat{v}^{(j)}$ sind orthogonal, weil $\hat{\Sigma}$ symmetrisch ist. Aufgrund der Orthonormalität der $\hat{v}^{(j)}$ bleibt die Funktion zur Bestimmung der $\hat{\alpha}_{ij}$ aus den X_i mit $\hat{\alpha}_{ij} = (\hat{v}^{(j)})^T (X_i - \hat{\mu}_n)$ von einfacher Gestalt.

Wir fassen die Erkenntnisse in einem Algorithmus zusammen.

Definition 10.5 (PCA-Algorithmus)
Sei $k \in \mathbb{N}$, $1 \leq k \leq d$. Gegeben i.i.d. Trainingsdaten X_i, $i = 1, ..., n$, setze

$$\hat{\Sigma} := \frac{1}{n} \sum_{i=1}^{n} (X_i - \hat{\mu}_n)(X_i - \hat{\mu}_n)^T, \qquad \hat{\mu}_n := \frac{1}{n} \sum_{i=1}^{n} X_i.$$

Seien $\hat{\lambda}_1 \geq ... \geq \hat{\lambda}_k$ die k größten Eigenwerte von $\hat{\Sigma}$ und $\hat{v}^{(1)}, ..., \hat{v}^{(k)}$ die zugehörigen Eigenvektoren mit Länge 1. Dann heißen $\hat{v}^{(1)}, \ldots, \hat{v}^{(k)}$ die geschätzten ersten k *Hauptkomponenten* mit zugehöriger Projektionsabbildung

$$\hat{A}_n^k : \mathbb{R}^d \to \mathbb{R}^k, \quad \hat{A}_n^k(x) := ((\hat{v}^{(1)})^T (x - \hat{\mu}_n), ..., (\hat{v}^{(k)})^T (x - \hat{\mu}_n))^T.$$

Die Werte $\hat{\alpha}_i^{PCA,k} = \hat{A}_n^k(X_i)$ heißen die *auf die ersten k Hauptkomponenten projizierten Trainingsdaten.* ♦

Bemerkung Mit Hilfe $\hat{A}_n^k$ können auch *neue* Beobachtungen auf den niedrigdimensionaleren Raum $\mathbb{R}^k$ projiziert werden. Dies ist nützlich, wenn der PCA-Algorithmus nur als Vorbearbeitung der Daten genutzt wird und danach ein Verfahren auf die reduzierten Daten $\hat{\alpha}_i$, $i = 1, ..., n$ angewandt wurde und nun dasselbe für die neue Beobachtung geschehen soll, vgl. Modellannahme 10.8.

Im Folgenden geben wir ein Beispiel für die Anwendung im Falle $d = 2$.

Beispiel 10.6 (Multivariate Normalverteilung) Gegeben seien i.i.d. $X_i \sim N(0, \Sigma)$ (Dimension $d = 2$) mit

$$\Sigma = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}.$$

Σ hat die Eigenwerte $\lambda_1 = 1.9, \lambda_2 = 0.1$ mit zugehörigen Eigenvektoren $v^{(1)} = \frac{1}{\sqrt{2}}(1,1)^T$, $v^{(2)} = \frac{1}{\sqrt{2}}(-1,1)^T$. In Abb. 10.2 sind $n = 100$ Trainingsdaten zusammen mit den Schätzungen $\hat{v}^{(1)}, \hat{v}^{(2)}$ sowie $\hat{\alpha}_i^{PCA,2}, \hat{\alpha}_i^{PCA,1}$ und die anschauliche Interpretation der Projektionen $\hat{\alpha}_i^{PCA,1}$ dargestellt. Wie man sehen kann, wird die Lage von X_i bereits relativ gut durch $\hat{\alpha}_i^{PCA,1}$ beschrieben.

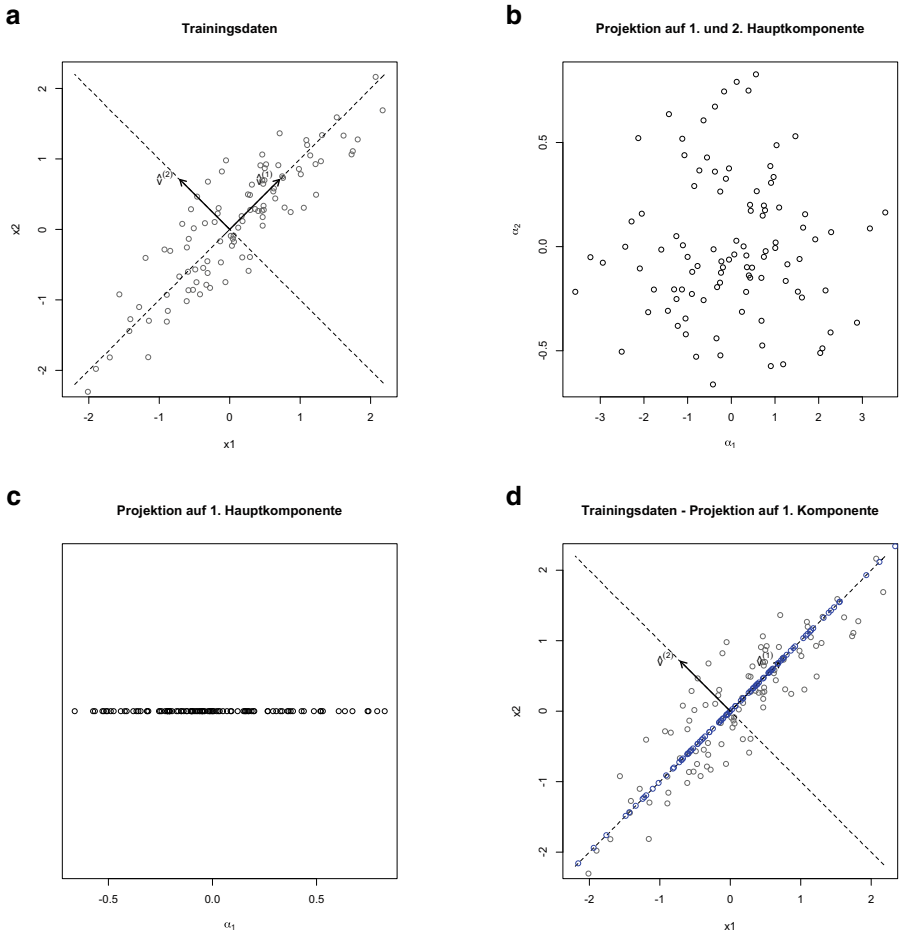


Abb. 10.2 Anwendung der Hauptkomponentenanalyse auf $n = 100$ Trainingsdaten X_i aus Beispiel 10.6. **a** Trainingsdaten und geschätzte Hauptkomponenten $\hat{v}^{(1)}, \hat{v}^{(2)}$, **b** Projektionen $\hat{\alpha}_i^{PCA,2}$ auf 1. und 2. geschätzte Hauptkomponente, **c** Darstellung der Projektion auf die 1. Hauptkomponente $\hat{\alpha}_i^{PCA,1}$, **d** anschauliche Entsprechung (blau) von $\hat{\alpha}_i^{PCA,1}$ durch Projektion der Trainingsdaten X_i auf die von $\hat{v}^{(1)}$ erzeugte Gerade

10.1.2 Statistische Einordnung und Modellannahme

Mit dem PCA-Algorithmus aus Definition 10.5 werden Eigenvektoren und Eigenwerte der empirischen Kovarianzmatrix $\hat{\Sigma}$ der Beobachtungen bestimmt. Für eine genauere Analyse untersuchen wir zunächst das Verhalten der zugehörigen theoretischen Größe Σ und leiten daraus Modellannahmen ab.

Lemma 10.7 (Zerlegung von X_1) Es gelte $\mathbb{E}[\|X_1\|_2^2] < \infty$. Die zu X_1 gehörige Kovarianzmatrix

$$\Sigma = \mathbb{E}[(X_1 - \mathbb{E}X_1)(X_1 - \mathbb{E}X_1)^T] \in \mathbb{R}^{d\times d}$$

sei symmetrisch positiv definit mit paarweise verschiedenen Eigenwerten $\lambda_1 > ... > \lambda_d > 0$. Es sei $v^{(1)}, ..., v^{(d)}$ die eindeutig bestimmte zugehörige Orthonormalbasis aus Eigenvektoren und $S := (v^{(1)}, ..., v^{(d)})$. Dann gilt für $\alpha_1 = (\alpha_{11}, ..., \alpha_{1d})^T$ mit $\alpha_1 := S^t \cdot (X_1 - \mathbb{E}X_1)$:

$$\mathrm{Cov}(\alpha_{1j}, \alpha_{1k}) = \begin{cases} \lambda_j, & j = k \\ 0, & j \neq k \end{cases} \tag{10.6}$$

und

$$X_1 - \mathbb{E}X_1 = \alpha_{11} v^{(1)} + ... + \alpha_{1d} v^{(d)} \tag{10.7}$$

Die Richtungen $v^{(j)}$, $j = 1, ..., d$ werden als *Hauptkomponenten* bezeichnet. Sie geben an, in welche (unkorrelierten) Richtungen sich der Vektor X_1 mit welcher Stärke entwickelt.

Beweis Definieren wir $\Lambda := \mathrm{diag}(\lambda_1, ..., \lambda_d)$, so gilt $\Sigma = S\Lambda S^t$ bzw. $S^t \Sigma S = \Lambda$. Mit Hilfe dieser Identität und $\mathrm{Cov}(\alpha_{1j}, \alpha_{1k}) = (S^t \Sigma S)_{jk}$ folgt Gl. (10.6). Wegen $SS^t = I_{d\times d}$ folgt $S\alpha_1 = \tilde{X}_1$ und damit Gl. (10.7).

Bemerkung Wegen Satz 2.8 ist bekannt, dass Σ immer symmetrisch positiv semidefinit ist. Es wird also nur gefordert, dass Σ vollen Rang besitzt und die Eigenwerte verschieden groß sind. Tatsächlich funktionieren die im Folgenden vorgestellten Algorithmen auch, wenn diese Annahmen verletzt sind, aber die Eindeutigkeit der Bezeichnungen ist nicht mehr gewährleistet.

Die Hauptaussage von Lemma 10.7 ist, dass eine Beobachtung X_1 als Linearkombination der d Hauptkomponenten $v^{(1)}, ..., v^{(d)}$ dargestellt werden kann. Die zugehörigen Linearfaktoren sind durch die Zufallsvariablen $\alpha_{11}, ..., \alpha_{1d}$ bzw. durch den Vektor $\alpha_1 = (\alpha_{11}, ..., \alpha_{1d})^T$ gegeben. Dieser erfüllt:

- Die Komponenten von α_1 sind unkorreliert, vgl. Gl. (10.6). Anschaulich bedeutet das, dass die Information über die Lage von X_1 in den verschiedenen Komponenten von α_i disjunkt sind, d. h. jede Komponente „neue“ Informationen über die Lage enthält.
- $\alpha_1 = (\alpha_{11}, ..., \alpha_{1d})$ enthält zunächst „alle“ Informationen über die Lage von X_1, aber der „Informationsgehalt“ sinkt aufgrund der monoton fallenden Varianz $\lambda_j = \mathrm{Var}(\alpha_{1j})$

monoton in den Komponenten von α_1 ab. α_{i1} enthält die „meisten“ Informationen über die Lage von X_i, α_{i2} etwas weniger usw. Das bedeutet, dass vor allem die Komponenten α_{1j} mit kleinem j wichtig sind, um den Wert von X_1 zu erklären (die Vektoren $v^{(j)}$, $j = 1, ..., d$ haben als Elemente einer Orthonormalbasis Länge 1 und nehmen keinen Einfluss auf die „Größe“ des Summanden $\alpha_{1j}v^{(j)}$).

- Aufgrund Gl. (10.7) gilt

$$\sum_{j=1}^{d} \mathrm{Var}(X_{1j}) = \mathbb{E}[\|X_1 - \mathbb{E}X_1\|_2^2] = \sum_{j=1}^{d} \mathrm{Var}(\alpha_{1j}) = \sum_{j=1}^{d} \lambda_j. \tag{10.8}$$

Anschaulich geben die Eigenwerte λ_j damit an, welcher Anteil der Entfernung von X_1 zu $\mathbb{E}X_1$ durch α_{1j} erklärt wird; dieser beträgt

$$\frac{\lambda_j}{\lambda_1 + ... + \lambda_d}.$$

Falls also $\frac{\lambda_j}{\lambda_1+...+\lambda_d} > \frac{1}{d}$ ist, enthält α_{1j} durchschnittlich „mehr Information“ über die Lage von X_1 als eine zufällig ausgewählte Komponente von X_1. Der prozentuale Anteil an der Varianz, welcher durch die Projektion auf die ersten k Hauptkomponenten erklärt wird, beträgt entsprechend

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{d} \lambda_j}.$$

In Beispiel 10.6 gilt $\frac{\lambda_1}{\lambda_1+\lambda_2} = 0.95$, d. h. mit Hilfe der Projektion auf die erste Hauptkomponente werden 95 % der Varianz erklärt.

Lemma 10.7 ordnet jedem Trainingsdatum X_i den Vektor $\alpha_i = (\alpha_{i1}, ..., \alpha_{id})^T$ zu. Es bezeichne $\alpha_i^k := (\alpha_{i1}, ..., \alpha_{ik})^T$ die ersten k Komponenten dieses Vektors. Der PCA-Algorithmus aus Definition 10.5 ermittelt mit $\hat{\alpha}_i^{PCA,k}$ eine natürliche Schätzung von α_i^k. Aufgrund des oben festgestellten absteigenden Informationsgehalts der Komponenten von α_i über die Lage von X_i stellt $\hat{\alpha}_i^{PCA,k}$ damit eine Kompression von X_i dar, und Verfahren des Supervised/Unsupervised Learnings können auch direkt auf $\hat{\alpha}_i^{PCA,k}$ anstatt X_i angewandt werden. Dieses Vorgehen entspricht mathematisch einer geänderten Modellannahme an das verwendete Verfahren, welches wir hier zur Verdeutlichung kurz mit V bezeichnen.

Modellannahme 10.8 (PCA: Typ 1 (Weiterverarbeitung)) *Weiterverarbeitung der Projektionen auf die ersten k Hauptkomponenten:*

- Supervised Learning: Die Modellannahme von V gelte für (α_1^k, Y_1) (und nicht für (X_1, Y_1)).
- Unsupervised Learning: Die Modellannahme von V gelte für α_1^k (und nicht für X_1).

Bemerkung Falls bei einem Problem des Supervised Learning Y im *wahren* Modell nur über die entfernten Komponenten $(\alpha_{1(k+1)}, ..., \alpha_{1d})$ von X abhängt (und nicht wie in obiger Modellannahme postuliert von $\alpha_{11}, ..., \alpha_{1k}$), wird also durch $\hat{\alpha}_i^{PCA,k}$ sämtliche für Y_i relevante Information aus X_i entfernt. Auch wenn dies im Allgemeinen nicht auftritt, sollte man sich dieser Möglichkeit bewusst sein und die Anwendung des PCA-Algorithmus stets kritisch hinterfragen.

Rekonstruktionsfehler

Ist das Ziel wie eingangs motiviert, dass die Lage der Trainingsdaten X_i mit Hilfe der Projektionen $\hat{\alpha}_i^{PCA,k}$ auf die ersten k Hauptkomponenten rekonstruiert werden kann, so kann die Qualität von $\hat{\alpha}_i^{PCA,k}$ mittels des sogenannten Rekonstruktionsfehlers erfolgen. Ist $(w^{(1)}, ..., w^{(d)})$ ein Orthonormalsystem von $\mathbb{R}^d$, so erhält man eine Darstellung

$$X_1 - \mathbb{E}X_1 = \sum_{j=1}^{d} \alpha_{1j} w^{(j)}$$

und $\alpha_{1j} = (w^{(j)})^T (X_1 - \mathbb{E}X_1)$, $j = 1, ..., d$. Lassen wir nun auf der rechten Seite die Summanden für $j = k+1, ..., d$ weg, bildet diese nur noch eine Approximation von $X_1 - \mathbb{E}X_1$. Die rechte Seite kann dann geschrieben werden als

$$\sum_{j=1}^{k} \alpha_{1j} w^{(j)} = \sum_{j=1}^{k} w^{(j)} \cdot (w^{(j)})^T (X_1 - \mathbb{E}X_1) = \Pi_k (X_1 - \mathbb{E}X_1),$$

wobei $\Pi_k := \sum_{j=1}^{k} w^{(j)} \cdot (w^{(j)})^T$. Der entstehende Fehler kann durch den Abstand

$$\left\| (X_1 - \mathbb{E}X_1) - \sum_{j=1}^{k} \alpha_{1j} w^{(j)} \right\|_2^2 = \| (X_1 - \mathbb{E}X_1) - \Pi_k (X_1 - \mathbb{E}X_1) \|_2^2$$

ausgedrückt werden. Der so im Mittel erwartete Fehler bildet den Rekonstruktionsfehler.

Definition 10.9
Ist $(w^{(1)}, ..., w^{(d)})$ ein Orthonormalsystem von $\mathbb{R}^d$, $k \in \mathbb{N}$ und $\Pi_k := \sum_{j=1}^{d} w^{(j)} (w^{(j)})^T$, so heißt

$$R(\Pi_k) := \mathbb{E} \left\| (X_1 - \mathbb{E}X_1) - \Pi_k (X_1 - \mathbb{E}X_1) \right\|_2^2$$

der *Rekonstruktionsfehler* zur Projektion Π_k. ♦

Es gibt eine explizite Formel für den Rekonstruktionsfehler und dessen Minimum über die Menge der k-elementigen Orthonormalsysteme von $\mathbb{R}^d$. Dieses wird gerade erreicht durch die Eigenvektoren der Kovarianzmatrix $\Sigma = \mathbb{E}[(X_1 - \mathbb{E}X_1)(X_1 - \mathbb{E}X_1)^T]$.

Lemma 10.10 Mit den Bezeichnungen aus Definition 10.9 gilt:

$$R(\Pi_k) = \mathrm{Sp}(\Sigma(I_{d\times d} - \Pi_k))$$

Es gilt

$$\inf_{w^{(1)},\ldots,w^{(k)}\ orthonormal} R(\Pi_k) = R(\Pi_k^*) = \sum_{j=k+1}^{d} \lambda_j$$

wobei $v^{(1)}, \ldots, v^{(d)}$ die Eigenvektoren zu den Eigenwerten $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d \geq 0$ von Σ bezeichnen und $\Pi_k^* = \sum_{j=1}^{k} v^{(j)}(v^{(j)})^T$.

Beweis Wegen $(I_{d\times d} - \Pi_k)^T(I_{d\times d} - \Pi_k) = (I_{d\times d} - \Pi_k)$ gilt:

$$\begin{aligned} R(\Pi_k) &= \mathbb{E}\|(I_{d\times d} - \Pi_k)(X_1 - \mathbb{E}X_1)\|_2^2 \\ &= \mathbb{E}[(X_1 - \mathbb{E}X_1)^T(I_{d\times d} - \Pi_k)(X_1 - \mathbb{E}X_1)] = \mathrm{Sp}((I_{d\times d} - \Pi_k)\Sigma) \end{aligned}$$

Die zweite Behauptung kann elementar nachgerechnet werden, indem ein beliebiges Orthonormalsystem $w^{(1)}, \ldots, w^{(k)}$ in der Basis $v^{(1)}, \ldots, v^{(d)}$ geschrieben wird durch $w^{(j)} = \sum_{l=1}^{d} \beta_{jl} v^{(l)}$ mit $\beta_{jl} \in \mathbb{R}$.

Das bedeutet, der Rekonstruktionsfehler unter Nutzung von nur k orthonormalen Vektoren kann nie unter $R(\Pi_k^*) = \sum_{j=k+1}^{d} \lambda_j$ sinken (insbesondere nicht bei Nutzung der k Hauptkomponenten erhalten durch den PCA-Algorithmus). Im Extremfall, dass alle Eigenwerte von Σ in etwa gleich groß sind (d. h. X_1 ist „in alle Richtungen“ im Raum verteilt) entsteht schon durch Weglassen eines Vektors aus einem vollständigen Orthonormalsystem $(w^{(1)}, \ldots, w^{(d)})$ ein großer Rekonstruktionsfehler.

Analog zum Supervised Learning kann der Rekonstruktionsfehler zur Messung der Qualität eines Algorithmus genutzt werden, der die ersten k Komponenten des Orthonormalsystems $v^{(1)}, \ldots, v^{(d)}$ von Σ schätzt (wie beispielsweise der PCA-Algorithmus):

Definition 10.11
Sei $1 \leq k \leq d$. Liefert ein Algorithmus eine Schätzung $\hat{v}^{(1)}, \ldots, \hat{v}^{(k)}$ der ersten k Elemente eines Orthonormalsystems, so nennen wir

$$R(\hat{\Pi}_k), \qquad \hat{\Pi}_k := \sum_{j=1}^{k} \hat{v}^{(j)}(\hat{v}^{(j)})^T$$

den *Rekonstruktionsfehler* des Algorithmus (mit k Hauptkomponenten). $\mathbb{E}R(\hat{\Pi}_k)$ heißt *Generalisierungsfehler* und

$$\mathbb{E}R(\hat{\Pi}_k) - R(\Pi_k^*)$$

das *Excess Bayes Risk* des Algorithmus (mit k Hauptkomponenten). ♦

Offensichtlich gilt $R(\Pi_k^*) = 0$ genau dann, wenn $\lambda_{k+1} = ... = \lambda_d = 0$. In diesem Fall ist automatisch auch die Modellannahme 10.8 erfüllt. In praktischen Beispielen gilt jedoch häufig sogar noch $\lambda_d > 0$. Man kann zeigen, dass auch die Konvergenzgeschwindigkeit des Excess Bayes Risks $\mathbb{E}R(\hat{\Pi}_k) - R(\Pi_k^*)$ maßgeblich von den übrigen Werten $\lambda_{k+1}, ..., \lambda_d$ abhängt. Zur Abschätzung des Excess Bayes Risks trifft man daher typischerweise Modellannahmen an die Größe der unbekannten λ_j. Ist $(c(j))_{j=1,...,d}$ eine vorgegebene monoton fallende nichtnegative Folge reeller Zahlen, so fordert man:

Definition 10.12 (Modellannahme PCA: Typ 2 (Rekonstruktion))
Es gelte $\lambda_j \leq c(j)$ $(j = 1, ..., d)$. ♦

Standardisierung

Wie aus dem ursprünglichen Optimierungsproblem in Definition 10.3 ersichtlich ist, erfolgt die Ermittlung der Vektoren $\hat{v}^{(j)}$ und $\hat{\alpha}_i^{PCA,k}$ im PCA-Algorithmus mit dem Ziel, die Lage der X_i möglichst gut nachzubilden. Aufgrund von verschiedenen Skalierungen (zum Beispiel entstanden durch Umrechnung von Einheiten) in den Komponenten von X_i kann dies jedoch zu aus statistischer Sicht „unerwünschten" Ergebnissen führen, indem eigentlich notwendige Hauptkomponenten entfernt werden.

Der Grund ist, dass die „statistische Information" einer Zufallsvariablen in Hinblick auf eine andere Zufallsvariable anders gemessen wird als durch einen reinen Vergleich der Lage. Sind beispielsweise die Komponenten X_{11}, X_{12} unabhängig, so enthalten sie statistisch betrachtet genau gleich viel (unabhängige) Information über das zugrunde liegende Ereignis $\omega \in \Omega$ des Wahrscheinlichkeitsraums, welches bei einer Beobachtung eintritt. Sind diese jedoch unterschiedlich stark skaliert, so liefert die PCA einen klaren Favoriten. Zur Verdeutlichung betrachten wir das folgende Beispiel.

Beispiel 10.13 Gegeben seien i.i.d. Trainingsdaten $X_i \sim N(0, \Sigma)$ der Dimension $d = 2$ mit

$$\Sigma = \Sigma_1 := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \textit{oder} \quad \Sigma = \Sigma_2 := \begin{pmatrix} 100 & 0 \\ 0 & 1 \end{pmatrix}.$$

Die Eigenvektoren beider Matrizen sind jeweils $(1, 0)^T$ und $(0, 1)^T$; die Eigenwerte sind jedoch $\lambda_1(\Sigma_1) = 1, \lambda_2(\Sigma_1) = 1$ und $\lambda_1(\Sigma_2) = 100, \lambda_2(\Sigma_2) = 1$. Im zweiten Fall gibt es wegen $\lambda_1(\Sigma_2) \gg \lambda_2(\Sigma_2)$ eine deutliche Präferenz für die erste Hauptkomponente; man neigt dazu, nur die Projektionen auf die erste $\hat{\alpha}_i^{PCA,1}$ in weiteren Verfahren zu nutzen, obwohl beide Komponenten von X_1 gleich viel unabhängige statistische Information bereitstellen. Im ersten Fall geschieht dies nicht, da die Daten auf die gleiche Größe skaliert sind.

Soll die Auswahl der Hauptkomponenten also auf Basis der statistischen Information und nicht auf Basis der Lage erfolgen, müssen die Daten vor der Anwendung der PCA einheitlich skaliert werden.

Bemerkung 10.14 (PCA-Algorithmus mit Standardisierung) Führe den PCA-Algorithmus wie in Definition 10.5 durch, aber ersetze alle Terme $X_i - \hat{\mu}_n$ und $x - \hat{\mu}_n$ durch

$$\hat{N}_n^{-1}(X_i - \hat{\mu}_n) \quad bzw. \quad \hat{N}_n^{-1}(x - \hat{\mu}_n),$$

wobei

$$\hat{N}_n := \mathrm{diag}(\hat{\sigma}_{n,1}, ..., \hat{\sigma}_{n,d}), \qquad \hat{\sigma}_{n,j}^2 := \frac{1}{n}\sum_{i=1}^{n}(X_{ij} - \hat{\mu}_{n,j})^2$$

die Standardabweichungen der einzelnen Komponenten von X_1 schätzt. Die entsprechend erhaltenen Eigenwerte und Eigenvektoren bezeichnen wir mit $\hat{\lambda}_j^{std}$ bzw. $\hat{v}^{(j),std}$. Die zugehörigen Projektionen nennen wir $\hat{\alpha}_i^{PCA,k,std}$.

Werden vom PCA-Algorithmus ohne Standardisierung die gleichen Hauptkomponenten ausgewählt wie beim PCA-Algorithmus mit Standardisierung, so liefern diese (bis auf Skalierung) die gleichen Resultate $\hat{\alpha}_i^{PCA,k}$ bzw. $\hat{\alpha}_i^{PCA,k,std}$. Ein Unterschied in der Qualität der komprimierten Information tritt erst auf, wenn aufgrund der unterschiedlich großen $\hat{\lambda}_j$ bzw. $\hat{\lambda}_j^{std}$ unterschiedliche Hauptkomponenten ausgewählt werden.

In Lemma 10.7 entspricht die Modifikation aus Bemerkung 10.14 der Berechnung der *Korrelationsmatrix*

$$\Sigma^{std} = \mathbb{E}[X_1^{std} \cdot (X_1^{std})^T]$$

mit den standardisierten Zufallsvariablen $X_1^{std} = (X_{11}^{std}, ..., X_{1d}^{std})^T$, wobei

$$X_{1j}^{std} := \frac{X_{1j} - \mathbb{E}X_{1j}}{\sqrt{\mathrm{Var}(X_{1j})}}, \quad j = 1, ..., d.$$

Entsprechend erhält man neue Eigenwerte und Eigenvektoren λ_j^{std}, $v^{(j),std}$ und $\alpha_1^{std} := S^t \cdot X_1^{std}$. Anstelle von Gl. (10.8) gilt dann wegen $X_1^{std} = \sum_{j=1}^{d} \alpha_{1j}^{std} v^{(j),std}$:

$$d = \sum_{j=1}^{d} \mathrm{Var}(X_{1j}^{std}) = \mathbb{E}[\|X_1^{std}\|_2^2] = \sum_{j=1}^{d} \mathrm{Var}(\alpha_{1j}^{std}) = \sum_{j=1}^{d} \lambda_j^{std},$$

und die Eigenwerte λ_j^{std} geben nun unabhängig voneinander mit $\frac{\lambda_j^{std}}{d}$ den Anteil an, den $\hat{\alpha}_{1j}$ an der Lage des standardisierten X_1^{std} erklärt. Ist insbesondere $v \in \mathbb{R}^d$ ein Vektor der Form $v = \frac{1}{d}\sum_{j=1}^{d} v^{(j),std}$, so gilt für die Korrelation

$$\rho(v^T X_1, \alpha_{1j}^{std}) = \sqrt{\frac{\lambda_j^{std}}{d}},$$

d. h. $\frac{\lambda_j^{std}}{d}$ ist auch statistisch gesehen ein Maß für den Einfluss von α_{1j} auf X_1.

Wir gehen im weiteren Verlauf des Kapitels nicht weiter auf die Standardisierung und die damit verbundene Interpretation ein.

10.1.3 Nichtlineare PCA

In der bisher eingeführten Form in Definition 10.5 ist der PCA-Algorithmus nur in der Lage, die Trainingsdaten zur Dimensionsreduktion auf Hyperebenen zu projizieren. In der Praxis sind die Daten aber häufig in komplexeren Strukturen angeordnet. Zur Motivation betrachten wir das folgende Beispiel:

Beispiel 10.15 Die Verteilung von $X_1 = (X_{11}, X_{12})$ sei wie folgt definiert:

$$X_1 = U_{11} \cdot \begin{pmatrix} \cos(U_{11}) \\ \sin(U_{12}) \end{pmatrix},$$

wobei $U_1 = (U_{11}, U_{12})^T$ und U_{11} gleichverteilt auf $[-0.9, 1.1]$ und U_{22} gleichverteilt auf $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

In Abb. 10.3a sind $n = 200$ i. i. d. Beobachtungen X_i, $i = 1, ..., n$ dieses Modells zu sehen. Offensichtlich können die Daten hier nicht gut durch eine Gerade approximiert werden, der PCA-Algorithmus aus Definition 10.5 schlägt fehl und erkennt keine hilfreichen Hauptkomponenten zur Dimensionsreduktion (vgl. Abb. 10.3b). Führen wir allerdings eine geeignete Koordinatentransformation von X_1 durch, können die Daten wieder durch Punkte auf einer Geraden angenähert werden. Hier gilt jeweils $X_i = \Phi_0(U_i)$ mit

$$\Phi_0 : [0, \infty) \times [-\pi, \pi) \to \mathbb{R}^2, \quad \Phi(u) = (u_1 \cdot \cos(u_2), u_2 \cdot \sin(u_1))^T.$$

Nutzen wir die inverse Funktion $h_0 := \Phi_0^{-1}$ gegeben durch

$$h_0(x) = ((x_1^2 + x_2^2)^{1/2}, \arctan2(x_2, x_1))^T. \tag{10.9}$$

In Abb. 10.3c sind die transformierten Daten U_i aufgetragen sowie die Ermittlung der Hauptkomponenten mittels des PCA-Algorithmus auf Basis der Trainingsdaten U_i (anstelle von X_i). Die erhaltenen Projektionen $\hat{\alpha}_i^{PCA,1}$ auf die 1. Hauptkomponente sind in Abb. 10.3d zu sehen. In Abb. 10.3e ist die anschauliche Entsprechung der Projektionen im Raum der ursprünglichen Trainingsdaten X_i dargestellt, welche durch

$$P_i := \Phi_0(\hat{\mu}_n + \alpha_i^{PCA,1} \hat{v}^{(1)}), \quad i = 1, ..., n$$

erhalten wurden.

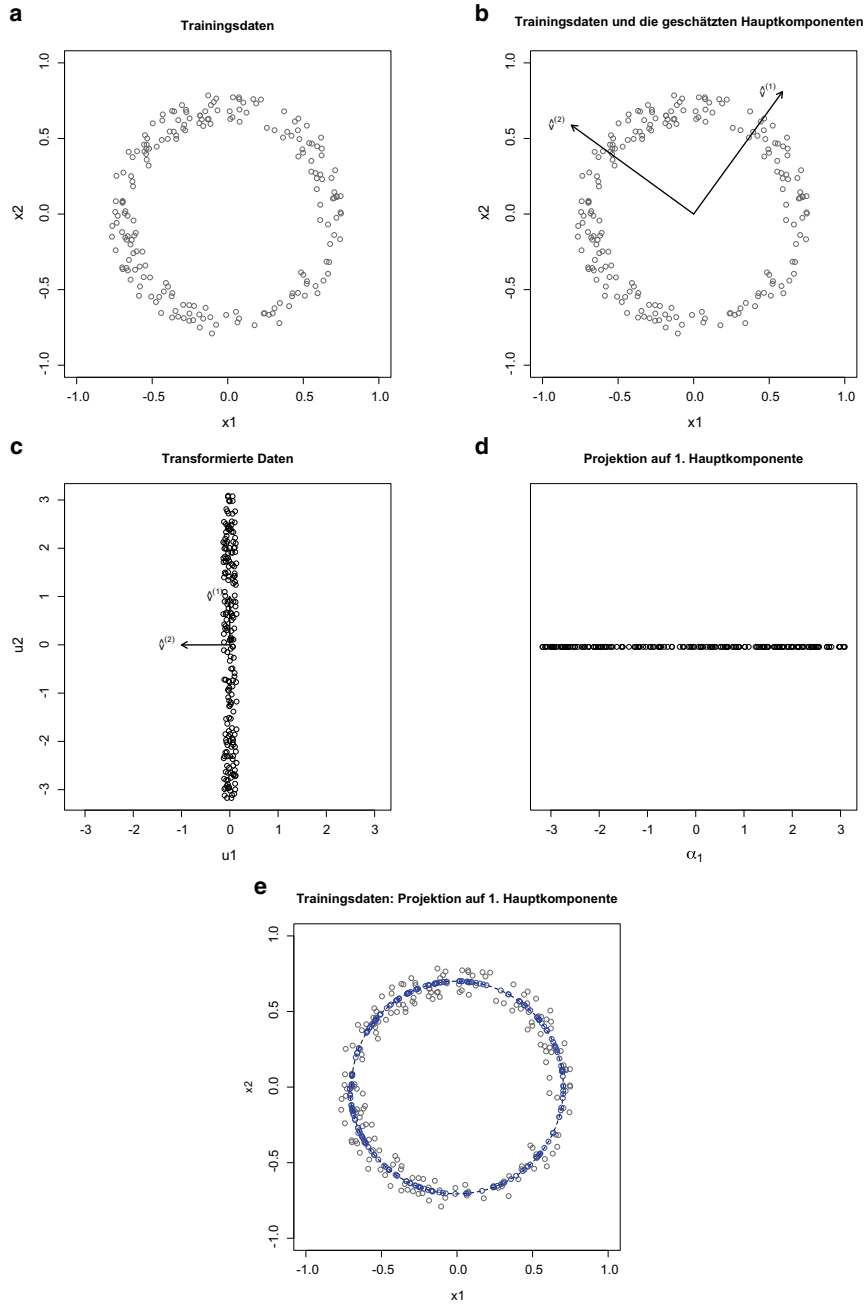


Abb. 10.3 **a** $n = 200$ Trainingsdaten X_i aus Beispiel 10.15, **b** erfolglose Anwendung des PCA-Algorithmus aus Definition 10.5 auf X_i, **c** transformierte Trainingsdaten U_i und geschätzte Hauptkomponenten, **d** Projektion von U_i auf die 1. Hauptkomponente, **e** anschauliche Entsprechung (blau) von $\hat{\alpha}_i^{PCA,1}$ im ursprünglichen Raum der Trainingsdaten

Bemerkung Die erhaltene Dimensionsreduktion in Beispiel 10.15 ist mit Vorsicht zu genießen: Zwar wurden die Daten in der Dimension reduziert, allerdings fand durch die Koordinatentransformation eine *Änderung der Geometrie* im Raum statt: Während beispielsweise zwei Punkte $x = (-1, -\varepsilon)$, $y = (-1, +\varepsilon)$ ($\varepsilon > 0$ klein) im Raum der Trainingsdaten nahe beieinander lagen, liegen die zugehörigen Projektionen $\alpha_1 = \arctan2(x) \approx -\pi$, $\alpha_2 = \arctan2(y) \approx +\pi$ nun sehr weit auseinander. Da die Hauptkomponentenanalyse im Allgemeinen als Vorbearbeitung der Daten genutzt wird, um weitere Algorithmen mit einem bestimmten Ziel anzuwenden, kommt es maßgeblich auf dieses verfolgte Ziel an (und damit letztlich auf die Gültigkeit der Modellannahme 10.8), ob die neu erhaltenen, reduzierten Trainingsdaten $\hat{\alpha}_i^{PCA,1}$, $i = 1, ..., n$ nützlich sind.

Insbesondere im Kontext des Findens von Repräsentanten (vgl. Kap. 9) und der Klassifizierung ist es wichtig, dass die Hauptkomponentenanalyse Daten, die sich um verschiedene, niedrigdimensionale Mengen akkumulieren, in ihrer Struktur durch Dimensionsreduktion zu vereinfachen. Daher müssen die Daten X_i auch nicht unbedingt aus den Projektionen $\hat{\alpha}_i^{PCA,k}$ vollständig rekonstruierbar sein, sondern es müssen lediglich die wesentlichen, trennenden Eigenschaften der Daten gut extrahiert werden. Zur Verdeutlichung betrachten wir das folgende Beispiel.

Beispiel 10.16 Die Verteilung von $X_1 = (X_{11}, X_{12})$ sei wie folgt definiert: Es sei Z_1 eine Zufallsvariable mit $\mathbb{P}(Z_1 = 1) = \mathbb{P}(Z_2 = 2) = \frac{1}{2}$. Ist $Z_1 = 1$, so wird X_1 wie in Beispiel 10.15 erzeugt. Ist $Z_1 = 2$, so seien X_1 unabhängig gleichverteilt auf $M = \{x \in \mathbb{R}^2 : |x_1| + |x_2| = 1\}$. In Abb. 10.4a ist ein Beispiel für $n = 400$ Trainingsdaten X_i, $i = 1, ..., n$ zu sehen. Für einen besseren Überblick sind die Daten für $Z_1 = 1$ schwarz und die Daten für $Z_1 = 2$ rot dargestellt; für die im Folgenden angewandten Algorithmen ist diese Zusatzinformation nicht vorhanden.

Für die gegebenen Trainingsdaten wären zwei verschiedene Koordinatentransformationen notwendig, um die Daten so zu transformieren, dass die Variation unabhängig voneinander in zwei verschiedene Richtungen erfolgt wie nach der Transformation in Beispiel 10.15.

Hier allerdings wollen wir keine vollständige Rekonstruktion der Daten erreichen. Stattdessen ist unser Ziel, wesentliche Eigenschaften zu extrahieren, die eine Trennung der Daten X_i mit verschiedenen (unbekannten) $Z_i = \pm 1$ einfacher machen. Hierzu nutzen wir den Übergang in einen höherdimensionalen Raum, der eine neue Sicht bzw. Perspektive auf die Daten erlaubt. Wir setzen

$$h_\beta : \mathbb{R}^2 \to \mathbb{R}^3, \quad h_\beta(x) = (x_1, x_2, \beta \cdot (x_1^2 + x_2^2)), \quad x = (x_1, x_2) \tag{10.10}$$

mit $\beta = 2$, und betrachten die (zentrierten) Daten $\tilde{X}_i := h_\beta(X_i) - \frac{1}{n}\sum_{l=1}^n h_\beta(X_l)$, $i = 1, ..., n$, vgl. Abb. 10.4b. Durch die Wahl von β haben wir erreicht, dass die Daten $\tilde{X}_i$ vor allem entlang der dritten Komponente eine starke Streuung in der Lage aufweisen. Die Daten $\tilde{X}_i$ sind zwar immer noch nicht gut durch eine Gerade approximierbar, aber der PCA-Algorithmus erkennt, dass er zur Beschreibung der Daten vor allem die dritte Komponente

nutzen muss (diese wird die erste Hauptkomponente), um die Lage von $\tilde{X}_i$ zu erklären. In Abb. 10.4b sind zusätzlich die drei geschätzten Hauptkomponenten des PCA-Algorithmus (angewandt auf $\tilde{X}_i$) eingezeichnet. In Abb. 10.4c, d sind die Projektionen $\hat{\alpha}_i^{PCA,2}$ bzw. $\hat{\alpha}_i^{PCA,1}$ auf die erste und zweite bzw. nur auf die erste Hauptkomponente dargestellt.

Man sieht, dass die Projektionen $\hat{\alpha}_i^{PCA,2}$, obgleich von derselben Dimension wie die Ausgangsdaten, nun eine wesentlich einfachere Bearbeitungsgrundlage beispielsweise für den k-means-Clustering-Algorithmus oder den EM-Algorithmus bieten als die ursprüng-

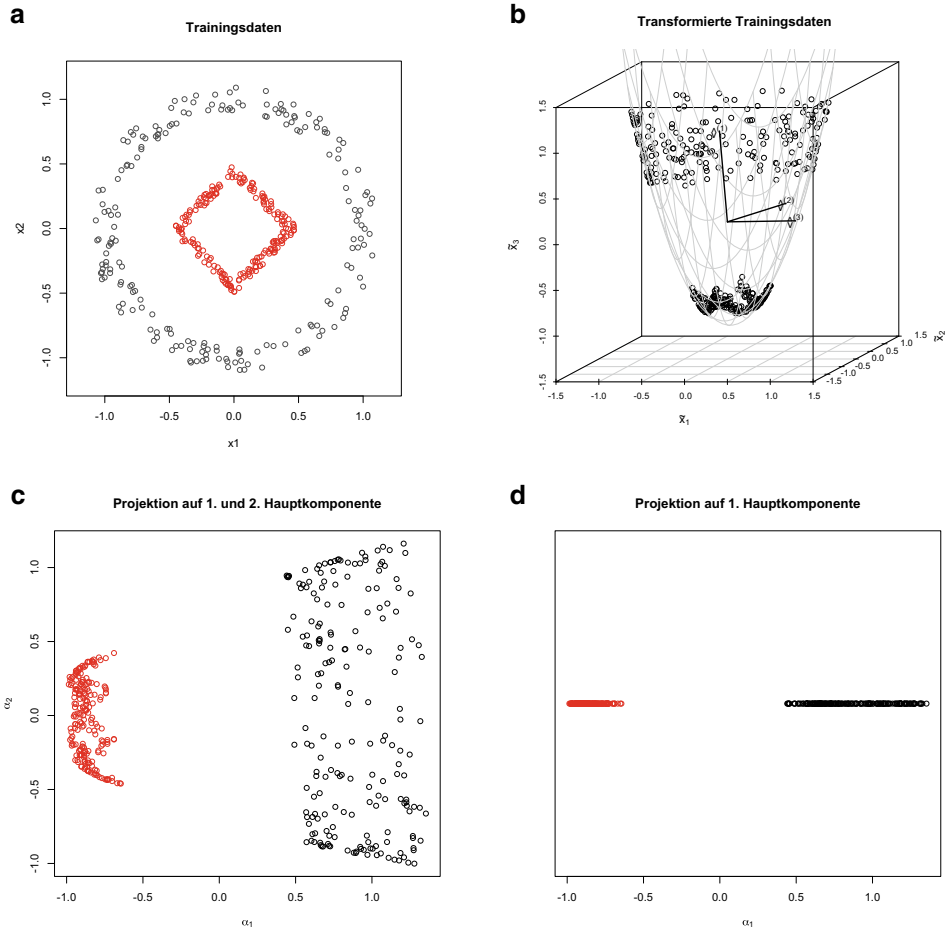


Abb. 10.4 **a** $n = 400$ Trainingsdaten X_i aus Beispiel 10.16, **b** Transformation $\tilde{X}_i$ der Daten in $\mathbb{R}^3$ mittels h_2 aus Gl. (10.10), Darstellung der durch h_2 erzeugten Fläche (grau) und der Hauptkomponenten des PCA-Algorithmus angewandt auf $\tilde{X}_i$, **c** Darstellung der Projektionen $\hat{\alpha}_i^{PCA,2}$ auf die ersten beiden Hauptkomponenten, **d** zugehörige Projektionen $\hat{\alpha}_i^{PCA,1}$ auf die erste Hauptkomponente

lichen Daten; die Modellannahmen dieser Verfahren werden durch die neue Struktur von $\hat{\alpha}_i^{PCA,2}$ besser erfüllt als durch die ursprünglichen Trainingsdaten X_i. Anschaulich entsteht $\hat{\alpha}_i^{PCA,2}$ durch Betrachtung der dreidimensionalen Daten $\tilde{X}_i$ aus der Richtung, in welcher laut PCA die geringste Änderung der Lage der $\tilde{X}_i$ erfolgt (und somit durch einen Blick von dieser Richtung der geringste Informationsverlust auftritt). Mathematisch entspricht dies der Richtung der dritten Hauptkomponente.

Der Erfolg dieser Methode hängt maßgeblich von einer günstigen Skalierung der Daten durch die Transformation h_β ab. In Abb. 10.5 wurden die Daten stattdessen mit h_β und $\beta = 1.2$ transformiert. Wie man sieht, ist nicht länger die dritte Komponente von $\tilde{X}_i$ die wichtigste Richtung zur Beschreibung der Daten; entsprechend ergeben sich andere Projektionen $\hat{\alpha}_i^{PCA,2}$, die nicht optimal für eine Weiterverarbeitung mit dem EM-Algorithmus oder k-means-Clustering geeignet sind.

Die mutwillige Skalierung von h_β in der dritten Komponente ist gewissermaßen ein „Missbrauch" der ursprünglichen Rekonstruktionsidee der Hauptkomponentenanalyse: Durch Vergrößerung der Streuung der transformierten Trainingsdaten $\tilde{X}_i \in \mathbb{R}^{\tilde{d}}$ in einer von uns vorgegebenen Richtung zwingen wir den PCA-Algorithmus dazu, die erste Hauptkomponente in $\mathbb{R}^{\tilde{d}}$ in diese Richtung zu legen um die transformierten Daten gut beschreiben zu können, unabhängig davon, ob auch die ursprünglichen Trainingsdaten X_i mit Hilfe dieser ersten Hauptkomponente gut rekonstruiert werden können.

Wir halten zunächst eine formale Beschreibung des oben durchgeführten Algorithmus fest.

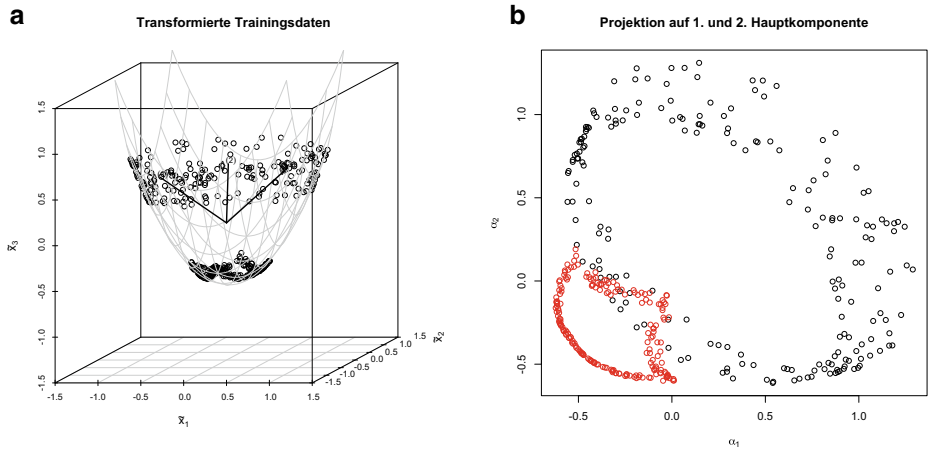


Abb. 10.5 **a** Transformation $\tilde{X}_i$ der Trainingsdaten aus Beispiel 10.16 in $\mathbb{R}^3$ mittels $h_{1.2}$ aus Gl. (10.10), Darstellung der durch $h_{1.2}$ erzeugten Fläche (grau) und der Hauptkomponenten des PCA-Algorithmus angewandt auf $\tilde{X}_i$, **b** Darstellung der Projektionen $\hat{\alpha}_i^{PCA,2}$ auf die ersten beiden Hauptkomponenten

Definition 10.17 (Algorithmus: PCA nichtlinear)
Sei $k \in \mathbb{N}$, $1 \leq k \leq \tilde{d}$ und $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ eine (nichtlineare) Funktion mit $\tilde{d} \in \mathbb{N}$. Gegeben i.i.d. Trainingsdaten X_i, $i = 1, ..., n$, setze

$$\hat{\Sigma}^h := \frac{1}{n} \sum_{i=1}^{n} (h(X_i) - \hat{\mu}_n)(h(X_i) - \hat{\mu}_n)^T, \qquad \hat{\mu}_n := \frac{1}{n} \sum_{i=1}^{n} h(X_i).$$

Seien $\hat{\lambda}_1^h \geq ... \geq \hat{\lambda}_k^h \geq 0$ die k größten Eigenwerte von $\hat{\Sigma}^h$ und $\hat{v}^{(1)}, ..., \hat{v}^{(k)}$ die zugehörigen Eigenvektoren mit Länge 1. Die zu den ersten k Hauptkomponenten gehörige Projektionsabbildung lautet

$$\hat{A}_n^{h,k} : \mathbb{R}^d \to \mathbb{R}^k, \quad \hat{A}_n^{h,k}(x) := ((\hat{v}^{(1)})^T (h(x) - \hat{\mu}_n), ..., (\hat{v}^{(k)})^T (h(x) - \hat{\mu}_n))^T.$$

Die Werte $\hat{\alpha}_i^{PCA,h,k} = \hat{A}_n^{h,k}(X_i), i = 1, ..., n$ heißen die auf *die ersten k Hauptkomponenten projizierten Trainingsdaten.* Zusätzlich sei $\hat{\alpha}_{\cdot j}^h := (\hat{\alpha}_{1j}^h, ..., \hat{\alpha}_{nj}^k)^T \in \mathbb{R}^n$, $j = 1, ..., k$ die Projektion der Trainingsdaten auf die j-te Hauptkomponente. ♦

Bemerkung 10.18

- Verwendet man die nichtlineare Hauptkomponentenanalyse, so wählt man im Allgemeinen eine Funktion $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ mit $\tilde{d} \gg d$. Dann entspricht die Darstellung der Punkte $\hat{\alpha}_i^{PCA,h,2}$ anschaulich einer Betrachtung der zentrierten Punkte $h(X_i) - \frac{1}{n} \sum_{j=1}^{n} h(X_j)$ in die beiden Richtungen mit der stärksten Streuung in der Lage (Entsprechendes gilt natürlich auch für die k-dimensionalen Projektionen).
- Nichtlineare PCA und Trennung von Daten: Da die Punkte $h(X_i)$ zentriert werden, entspricht die Bestimmung der ersten Hauptkomponente gleichzeitig der Bestimmung einer Hyperebene, auf der auf beiden Seiten (abhängig von ihrem Abstand zum Ursprung) in etwa gleich viele Punkte liegen. Im Kontext vom Unsupervised Learning kann daher PCA auch immer als „Trennung" verschiedener Punktmengen verstanden werden, vgl. Beispiel 10.16. Eine erfolgreiche Trennung mit der ersten Hauptkomponente erfolgt dann, wenn die die Punktmengen trennenden Eigenschaften durch h so stark betont werden, dass die erste Hauptkomponente genau in die Richtung der beiden verschiedenen Punktmengen im höherdimensionalen Raum $\mathbb{R}^{\tilde{d}}$ zeigt. Die Bedeutung der weiteren Hauptkomponenten kann dann iterativ verstanden werden: Aufgrund der Orthogonalität versucht die zweite Hauptkomponente jeweils die beiden entstandenen Mengen wieder möglichst günstig zu trennen usw.
- Darstellung der Projektionen im Ursprungsraum $\mathbb{R}^d$: Durch den Übergang in einen höherdimensionalen Raum $\mathbb{R}^{\tilde{d}}$ kann nicht mehr dargestellt werden, was die Projektionen auf die Hauptkomponenten $\hat{v}^{(j)} \in \mathbb{R}^{\tilde{d}}$ im ursprünglichen Raum $\mathbb{R}^d$ bedeuten. Der Grund ist, dass die Projektion im hochdimensionalen Raum $(\hat{v}^{(1)}, ..., \hat{v}^{(k)})^T \cdot \hat{\alpha}_i^{PCA,h,k}$ im Allgemeinen nicht mehr im Bild $h(\mathbb{R}^d) \subset \mathbb{R}^{\tilde{d}}$ der Funktion h liegen.
- Zum Informationsverlust/Rekonstruktionsfehler: Im Grunde läuft die Einbettung der Trainingsdaten X_i in einen höherdimensionalen Raum mittels einer Funktion $h : \mathbb{R}^d \to$

$\mathbb{R}^{\tilde{d}}$ zunächst der Idee der Dimensionsreduktion zuwider. Durch eine geeignete Betrachtung in einem höherdimensionalen Raum können aber komplexere Datenstrukturen erkannt und vereinfacht werden (vgl. Beispiel 10.16). Die Messung des Rekonstruktionsfehlers findet dann jedoch nicht mehr in $\mathbb{R}^d$, sondern im erweiterten Raum $\mathbb{R}^{\tilde{d}}$ statt; dieser lautet entsprechend Definition 10.9 nun

$$R^h(\Pi_k) := \mathbb{E}\left\|(h(X_1) - \mathbb{E}h(X_1)) - \Pi_k(h(X_1) - \mathbb{E}h(X_1))\right\|_2^2, \tag{10.11}$$

wobei $\Pi_k := \sum_{j=1}^{k} w^{(j)}(w^{(j)})^T$ und $(w^{(1)}, ..., w^{(\tilde{d})})$ ein Orthonormalsystem von $\mathbb{R}^{\tilde{d}}$ ist. Verwenden wir also die nichtlineare Hauptkomponentenanalyse mit $\tilde{d} > d$, so ist im Allgemeinen auch mit einer d-dimensionalen Projektion $\hat{\alpha}_i^{PCA,h,d}$ noch keine vollständige Rekonstruktion der Ausgangsdaten möglich. Zu sehen ist dies an Beispiel 10.16 bzw. Abb. 10.4c mit $d = 2, \tilde{d} = 3$. Durch das Betrachten des Paraboloids „von der Seite" geht der Informationsgehalt der 3. Hauptkomponente verloren, welche die „Tiefe" des Punktes im Raum angibt. Dadurch kann aus $\hat{\alpha}_i^{PCA,h,d}$ nicht mehr rekonstruiert werden, ob ein Punkt mit $\hat{\alpha}_i^{PCA,h,d} \approx (1,0)^T$ einem Punkt $X_i \approx (1,0)$ oder $X_i \approx (-1,0)$ entsprach (d. h. auf welcher Seite des äußeren Kreises der Punkt lag). Erst durch Angabe der vollständigen Koordinaten $\hat{\alpha}_i^{PCA,h,\tilde{d}}$ ist eine eindeutige Zuordnung möglich, denn dann ist $h(X_i) = (\hat{v}^{(1)}, ..., \hat{v}^{(\tilde{d})})^T \cdot \hat{\alpha}_i^{PCA,h,\tilde{d}}$ und somit

$$X_i = h^{-1}\Big((\hat{v}^{(1)}, ..., \hat{v}^{(\tilde{d})})^T \cdot \hat{\alpha}_i^{PCA,h,\tilde{d}}\Big).$$

Die bessere Zuordnung zu einer der beiden Strukturen, welche $\hat{\alpha}_i^{PCA,h,d}$ ermöglicht, erfolgt also auf Kosten der genauen Positionsbestimmung der ursprünglichen Punkte X_i auf den Strukturen.

Bei Verwendung des nichtlinearen PCA-Algorithmus ist also eine vollständige Rekonstruktion von X_i im Allgemeinen nur aus $\hat{\alpha}_i^{PCA,h,\tilde{d}}$ möglich. In diesem Sinne misst auch der Rekonstruktionsfehler in Gl. (10.11), wie hoch der Informationsverlust der Lage von X_i durch Nutzung von $\hat{\alpha}_i^{PCA,h,k}$, $k \in \{1, ..., \tilde{d}\}$ ist.

- Interpretation der nichtlinearen PCA im Ursprungsraum: Eine andere Perspektive auf die nichtlineare PCA bietet das ursprüngliche Optimierungsproblem aus Definition 10.1: Dass nicht mehr die Beobachtungen X_i in $\mathbb{R}^d$, sondern die Beobachtungen $h(X_i)$ in $\mathbb{R}^{\tilde{d}}$ mit Hilfe einer Hyperebene approximiert werden sollen, entspricht formal einer Änderung des verwendeten Abstandsmaßes: Im Raum $\mathbb{R}^{\tilde{d}}$ liegen die Daten $h(X_i)$ auf der Menge $\{h(x) : x \in \mathbb{R}^d\}$; zwei Elemente dieser Menge haben die Abstände $\|h(x) - h(x')\|_2^2$ zueinander. Anschaulich ändert sich der Abstand der Trainingsdaten X_i im Ursprungsraum von $D(x, x') = \|x - x'\|_2^2$ zu

$$D^h(x, x') = \|h(x) - h(x')\|_2^2.$$

In Beispiel 10.16 lautet dieser Abstand

$$D^{h_\beta}(x, x') = \|x - x'\|_2^2 + \beta^2 \cdot \left| \|x\|_2^2 - \|x'\|_2^2 \right|^2.$$

Zusätzlich zu $\|x - x'\|_2^2$ wird nun auch dem Abstand der Radien $\left| \|x\|_2^2 - \|x'\|_2^2 \right|$ (mit Skalierungsfaktor β) bei der Bestimmung der ersten Hauptkomponente Gewicht eingeräumt. Ist β groß, so liegen Punkte mit verschiedenen Radien für den PCA-Algorithmus weiter auseinander als dies durch die Darstellung der Trainingsdaten X_i in Abb. 10.4a suggeriert wird. Im höherdimensionalen Raum $\mathbb{R}^{\tilde{d}}$ geschieht dies dadurch, dass der zugehörige Paraboloid $h_\beta : \mathbb{R}^2 \to \mathbb{R}^3$ die Trainingsdaten für großes β weit in Richtung der dritten Komponente (des „Radius") auseinanderzieht und damit die Variation der transformierten Daten $\tilde{X}_i$ in diese Richtung erhöht.

Gemäß dieser Interpretation entsprechen die Punkte $\hat{\alpha}_i^{PCA,h,k}$, $i = 1, ..., n$ der bestmöglichen Darstellung der Punkte $X_i \in \mathbb{R}^d$ mit Abständen gegeben durch D^h im k-dimensionalen Raum $\mathbb{R}^k$.

Wir kennen damit zwei mögliche Interpretationen der Projektionen $\hat{\alpha}_i^{PCA,h,k}$:

(I1) der „k-dimensionale Blick" auf die zentrierten Punkte $h(X_i)$ im höherdimensionalen Raum $\mathbb{R}^{\tilde{d}}$, der am wenigsten Informationsverlust bedeutet,
(I2) die bestmögliche Darstellung der Punkte $X_i \in \mathbb{R}^d$ mit Abständen gegeben durch $D^h : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ in einem k-dimensionalen Raum.

Die Formulierung von h für gegebene Trainingsdaten in höherdimensionalen Räumen ($d > 3$) kann schwierig sein, da man nicht weiß, wie die Daten geometrisch im Raum verteilt sind, welche Komponentenkombinationen überhaupt interessante Blickrichtungen auf die Daten bieten könnten und wie die Komponenten von h untereinander skaliert werden müssen. Im Gegensatz zur SVM (vgl. Abschn. 4.4) ist es jedoch nicht ausreichend, die Dimension $\tilde{d}$ einfach zu erhöhen und $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ mit beliebigen nichtlinearen Komponenten zu füllen. Da beim Unsupervised Learning im Gegensatz zum Supervised Learning das Korrektiv der Daten Y_i fehlt, womit man sich „automatisch" vor Überanpassung schützen kann, muss man beim Unsupervised Learning viel genauer darauf achten, was die einzelnen Komponenten von h in Hinsicht auf das danach auf die reduzierten Daten $\hat{\alpha}_i^{PCA,h,k}$ angewandte Verfahren bewirken sollen und wie diese Komponenten zueinander skaliert sind. In Beispiel 10.16 etwa war es wichtig, dass β genügend groß gewählt wurde, damit bereits die erste Hauptkomponente eine wesentliche Vereinfachung für die Unterscheidung der beiden verschiedenfarbig eingezeichneten Mengen ermöglichte. In diesem Sinne ist (I1) keine geeignete Interpretation zur Durchführung der nichtlinearen PCA in der Praxis.

Eine Motivation mit (I2) ist vielversprechender. Falls D^h einfach genug ist, können die Resultate des nichtlinearen PCA-Algorithmus auch für hochdimensionale Trainingsdaten $X_i \in \mathbb{R}^d$ erklärt werden. Wird ein Abstandsmaß D^h im Raum $\mathbb{R}^d$ vorgegeben, so kann die Ermittlung der zugehörigen Funktion $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ jedoch schwierig sein. Wir zeigen nun, warum die Funktionen h nicht unmittelbar angegeben werden müssen. Wegen

$$D^h(x,x') = \|h(x) - h(x')\|_2^2 = h(x)^T h(x) - 2h(x)^T h(x') + h(x')^T h(x') \tag{10.12}$$

ist ersichtlich, dass das durch h erzeugte neue Abstandsmaß nur von $K(x,x') := h(x)^T h(x')$ abhängt. Es ist daher intuitiver, das gewünschte neue Abstandsmaß mittels $K(x,x')$ vorzugeben.

Das folgende Lemma zeigt, dass der nichtlineare PCA-Algorithmus tatsächlich nur mit der Angabe von K anstelle von dem zugehörigen h durchgeführt werden kann. Insbesondere zeigt es die Überflüssigkeit der Berechnung der Eigenvektoren $\hat{v}^{(j)}$ im hochdimensionalen Raum $\mathbb{R}^{\tilde{d}}$ und gibt trotzdem explizite Formeln für die Berechnung der zugehörigen Projektionen $\hat{\alpha}_i^{PCA,h,k}$ und $\hat{A}_n^{h,k}(x)$ an.

Lemma 10.19 (Der Kern-Trick beim nichtlinearen PCA-Algorithmus) Sei $k \in \mathbb{N}$ mit $1 \le k \le \min\{\tilde{d}, n-1\}$ und $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$. Definiere $K(x,x') := h(x)^T h(x')$. Sei $\hat{W} := (\hat{W}_{ij})_{i,j=1,\dots,n} \in \mathbb{R}^{n\times n}$ mit

$$\hat{W}_{ij} := \hat{K}_0(X_i, X_j),$$

wobei

$$\hat{K}_0(x,y) := K(x,y) - \frac{1}{n}\sum_{l=1}^n K(x,X_l) - \frac{1}{n}\sum_{l=1}^n K(X_l,y) + \frac{1}{n^2}\sum_{i,l=1}^n K(X_i,X_l).$$

Dann gilt mit den Bezeichnungen aus Definition 10.17:

(i) $\frac{1}{n}\hat{W}$ besitzt dieselben von null verschiedenen Eigenwerte $\hat{\lambda}_j^h$ wie $\hat{\Sigma}^h$ mit zugehörigen Eigenvektoren $\hat{\alpha}_{\cdot j}^h$ (mit $\|\hat{\alpha}_{\cdot j}\|_2^2 = n\hat{\lambda}_j^h$).

(ii) Falls $\hat{\lambda}_k^h > 0$, so gilt für $x \in \mathbb{R}^d$ und $1 \le j \le k$:

$$\hat{A}_n^{h,k}(x)_j = \frac{1}{n\hat{\lambda}_j^h}\sum_{i=1}^n \hat{\alpha}_{ij}^h \hat{K}_0(x,X_i) \tag{10.13}$$

Beweis Es ist leicht zu sehen, dass

$$\hat{W}_{i,j} = (h(X_i) - \hat{\mu}_n)^T (h(X_j) - \hat{\mu}_n), \qquad \hat{\mu}_n := \frac{1}{n}\sum_{i=1}^n h(X_i).$$

Zur Abkürzung schreiben wir $\tilde{X}_i := h(X_i) - \hat{\mu}_n$ und

$$\mathbb{X} := \begin{pmatrix} \tilde{X}_1^T \\ \vdots \\ \tilde{X}_n^T \end{pmatrix} \in \mathbb{R}^{n\times\tilde{d}}.$$

Dann gilt $\hat{\Sigma}^h = \frac{1}{n}\mathbb{X}^T\mathbb{X}$ und $\hat{W} := \mathbb{X}\mathbb{X}^T$. Insbesondere folgt $\text{Rang}(\hat{\Sigma}^h) = \text{Rang}(\hat{W}) \leq \min\{\tilde{d}, n-1\}$ (*). Wie in Definition 10.17 bezeichne $\hat{v}^{(j)}$ die Eigenvektoren mit Länge 1 von $\hat{\Sigma}^h$ zu den (absteigend sortierten) Eigenwerten $\hat{\lambda}^h_j$.

Sei $1 \leq j \leq \min\{\tilde{d}, n-1\}$. Dann ist $\hat{\lambda}^h_j$ von null verschieden. Wegen $\hat{\alpha}^h_{i,j} = \tilde{X}_i^T \hat{v}^{(j)}$ gilt $\hat{\alpha}^h_{\cdot j} = \mathbb{X}\hat{v}^{(j)}$. Mit $\hat{\Sigma}^h \hat{v}^{(j)} = \hat{\lambda}^h_j \hat{v}^{(j)}$ folgt $\|\hat{\alpha}^h_{\cdot j}\|_2^2 = n(\hat{v}^{(j)})^T \hat{\Sigma}^h \hat{v}^{(j)} = n\hat{\lambda}^h_j \|\hat{v}^{(j)}\|_2^2 = n\hat{\lambda}^h_j$ und

$$\frac{1}{n}\hat{W}\hat{\alpha}^h_{\cdot j} = \mathbb{X} \cdot \underbrace{\frac{1}{n}\mathbb{X}^T\mathbb{X}}_{=\hat{\Sigma}^h} \hat{v}^{(j)} = \hat{\lambda}^h_j \mathbb{X}\hat{v}^{(j)} = \hat{\lambda}^h_j \hat{\alpha}^h_{\cdot j},$$

das bedeutet, auch $\hat{W}$ hat den Eigenwert $\hat{\lambda}^h_j$, allerdings zum Eigenvektor $\hat{\alpha}^h_{\cdot j}$. Aufgrund von (*) folgt damit Aussage (i). Mit $\tilde{x} := h(x) - \hat{\mu}_n$ gilt:

$$\tilde{x}^T \hat{v}^{(j)} = \frac{1}{\hat{\lambda}^h_j} \tilde{x}^T \hat{\Sigma} \hat{v}^{(j)} = \frac{1}{n\hat{\lambda}^h_j} \tilde{x}^T \mathbb{X}^T \hat{\alpha}^h_{\cdot j} = \frac{1}{n\hat{\lambda}^h_j} \sum_{i=1}^{n} \hat{\alpha}^h_{ij} \cdot \tilde{x}^T \tilde{X}_i$$

Damit folgt (ii).

Bemerkung Ist $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ mit sehr hohem $\tilde{d} \gg n$, so ergeben sich die Projektionen $\hat{\alpha}_i^{PCA,h,k} = (\hat{\alpha}^h_{i1}, ..., \hat{\alpha}^h_{ik})^T$ trotzdem als Eigenvektoren einer quadratischen Matrix $\hat{W}$ mit „nur“ Dimension $n \times n$.

In einem nächsten Schritt liegt es nahe, die unterliegende Funktion h vollständig zu vergessen und nur noch mit der Angabe der Funktion K zu arbeiten. Dies führt zur Kern-basierten Hauptkomponentenanalyse.

10.1.4 Kern-basierte Hauptkomponentenanalyse

Lemma 10.19 erlaubt die Berechnung des nichtlinearen PCA-Algorithmus aus Definition 10.20 nur durch Angabe einer Funktion $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$, welche eine Dekomposition der Form

$$K(x, x') = h(x)^T h(x'), \quad x, x' \in \mathbb{R}^d \tag{10.14}$$

mit geeigneter Funktion $h : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ (möglicherweise $\tilde{d} = \infty$) besitzt. Anschaulich wird der nichtlineare PCA-Algorithmus dann im Raum $\mathbb{R}^{\tilde{d}}$ mit den eingebetteten Trainingsdaten $h(X_i)$, $i = 1, ..., n$ durchgeführt. Wir haben in Abschn. 4.4 gesehen, dass ein Mercer-Kern K die Gl. (10.14) mit geeignetem h erfüllt (vgl. Definition 4.32 und Satz 4.36). Einige häufig genutzte Kerne sind der Gauß-Kern und der Polynomkern (vgl. Beispiel 4.34). Im Gegensatz zur SVM wird bei der Kern-basierten Hauptkomponentenanalyse auch der zentrierte Polynomkern vom Grad $p \in \mathbb{N}$,

$$K_p^{poly,zentriert}(x, x') = (x^T x')^p, \quad x, x' \in \mathbb{R}^d, \tag{10.15}$$

genutzt. Ist ein Kern K gegeben, so erhalten wir den Kern-basierten PCA-Algorithmus (engl. *kernel-based PCA*).

Definition 10.20 (Algorithmus: Kern-basierte PCA)
Sei $k \in \mathbb{N}$, $1 \leq k \leq n-1$. Gegeben seien i.i.d. Trainingsdaten X_i, $i = 1, ..., n$ und ein Mercer-Kern $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$. Sei

$$\hat{K}_0(x, y) := K(x, y) - \frac{1}{n}\sum_{l=1}^{n} K(x, X_l) - \frac{1}{n}\sum_{l=1}^{n} K(X_l, y) - \frac{1}{n^2}\sum_{i,l=1}^{n} K(X_i, X_l)$$

die *zentrierte Kernfunktion*. Definiere $\hat{W} = (\hat{W}_{ij})_{i,j=1,...,n} \in \mathbb{R}^{n\times n}$ durch

$$\hat{W}_{ij} := \hat{K}_0(X_i, X_j).$$

Seien $\hat{\lambda}_1^K \geq ... \geq \hat{\lambda}_k^K \geq 0$ die k größten Eigenwerte von $\frac{1}{n}\hat{W}$ und $\hat{\alpha}_{\cdot j}^K = (\hat{\alpha}_{1j}^K, ..., \hat{\alpha}_{nj}^K)^T$, $j = 1, ..., k$ die zugehörigen Eigenvektoren mit $\|\hat{\alpha}_{\cdot j}^K\|_2^2 = n\hat{\lambda}_j^K$. Die zu den ersten k Hauptkomponenten gehörige Projektionsabbildung lautet

$$\hat{A}_n^{K,k} : \mathbb{R}^d \to \mathbb{R}^k, \quad \hat{A}_n^{K,k}(x) := \left(\hat{A}_n^K(x)_1, ..., \hat{A}_n^K(x)_k\right)^T,$$
$$\hat{A}_n^K(x)_j := \frac{1}{n\hat{\lambda}_j^K}\sum_{i=1}^{n} \hat{\alpha}_{ij} \cdot \hat{K}_0(x, X_i).$$

Die Werte $\hat{\alpha}_i^{PCA,K,k} = \hat{A}_n^{K,k}(X_i)$, $i = 1, ..., n$ heißen die auf *die ersten k Hauptkomponenten projizierten Trainingsdaten.* ♦

Bemerkungen

- Für den linearen Kern $K(x, x') = x^T x'$ erhalten wir den ursprünglichen PCA-Algorithmus.
- Für hohes n ist die Berechnung der Eigenwerte und Eigenvektoren von $\hat{W}$ sehr aufwendig: in der aktuellen Forschung werden je nach beabsichtigter Anwendung Methoden untersucht, welche beispielsweise annehmen, dass die Eigenvektoren $v^{(j)}$ dünn besetzt sind (d. h. viele Komponenten sind 0).
- Falls der Kern-basierte PCA-Algorithmus einen möglichst geringen Rekonstruktionsfehler liefern soll, d. h. die Lage von X_i gut mittels $\hat{\alpha}_i^{PCA,K,k}$ beschrieben werden kann, so sollte die Wahl von k anhand der geschätzten Eigenwerte $\hat{\lambda}_k^K$ getroffen werden: Wähle k so groß, dass
$$\frac{\sum_{j=1}^{k} \hat{\lambda}_j^K}{\sum_{j=1}^{n-1} \hat{\lambda}_j^K} \geq 1 - \varepsilon$$
mit vorher vorgegebenem $\varepsilon > 0$ gilt (zur Legitimation betrachte Bemerkung 10.26).

Im Allgemeinen ist man jedoch nicht unmittelbar an einem geringen Rekonstruktionsfehler interessiert, sondern eine geeignete Wahl von k hängt vor allem von dem danach auf $\hat{\alpha}_i^{PCA,K,k}$ angewandten Verfahren ab. Auch wenn $\hat{\alpha}_i^{PCA,K,k}$ keine gute Rekonstruktion der Lage von X_i ermöglicht, kann bereits durch geringes k eine gute Vereinfachung der Daten erreicht worden sein (vgl. Beispiel 10.16: Dort liefert $\hat{\alpha}_i^{PCA,h,1}$ eine gute Vorbereitung für das k-means-Clustering, aber aus $\hat{\alpha}_i^{PCA,h,1}$ können die ursprünglichen Trainingsdaten X_i nicht gut rekonstruiert werden).

Für die Kern-basierte PCA mit Kern K erhält man wie in Gl. (10.12) motiviert die folgende Modifikation des ursprünglichen euklidischen Abstands $D(x, x')$ im Raum der Trainingsdaten:

$$D^K(x, x') = h(x)^T h(x) - 2h(x)^T h(x') + h(x')^T h(x') = K(x, x) - 2K(x, x') + K(x', x')$$

Beispiel 10.21 (Geometrie des Gauß-Kerns) Für $\gamma > 0$ betrachten wir den Gauß-Kern $K_\gamma^{gauss}(x, x') = \exp(-\gamma \|x - x'\|_2^2)$. Hier ergibt sich

$$D^K(x, x') = 2\big[1 - \exp(-\gamma \|x - x'\|_2^2)\big].$$

Durch Modifikation von $\gamma \in (0, \infty)$ kann dieser Abstand verändert werden.

Ist $\gamma \ll 1$, so gilt wegen $e^z \approx 1 + z$ für $|z| \ll 1$:

$$D^K(x, x') \approx 2\gamma \|x - x'\|_2^2,$$

d. h. $D^K(x, x')$ imitiert den ursprünglichen euklidischen Abstand $\|x - x'\|_2^2$ im Raum der Trainingsdaten X_i (bis auf den Faktor 2γ, der jedoch keine Veränderung der Abstände der Punkte untereinander bewirkt). Ist hingegen $\gamma \gg 1$, so ist

$$\frac{D^K(x, x')}{\|x - x'\|_2^2} \begin{cases} > 1, & \|x - x'\|_2 \ klein, \\ < 1, & \|x - x'\|_2 \ groß \end{cases}$$

und vormals kleine Abstände werden von Kern-basierten PCA-Algorithmus als „groß“ angesehen und umgekehrt.

Die erste Hauptkomponente beim Kern-basierten PCA-Algorithmus kann nun Streuungen normal zu allen möglichen Kurven $\{x \in \mathbb{R}^d : v^T h(x) = 0\}$ ($v \in \mathbb{R}^\infty$) bewerten (d. h., eine „Trennung“ ist nicht nur entlang einer Hyperebene möglich) und liefert so die Richtung der stärksten Streuung der Trainingsdaten.

Im Falle, dass es zwei klar voneinander separierte niedrigdimensionale Mengen $M_1, M_2 \subset \mathbb{R}^d$ gibt, um welche die Trainingsdaten X_i akkumuliert sind, gibt es zwei Möglichkeiten:

(1) Die Trainingsdaten, welche zu einer der beiden Mengen gehören (ohne Beschränkung der Allgemeinheit M_1), haben eine wesentlich höhere Streuung als die von M_2. In

diesem Fall werden die Projektionen auf die ersten k Hauptkomponenten priorisiert versuchen, die zur Menge M_1 gehörigen Trainingsdaten gut zu beschreiben.

(2) Die jeweils zu M_1, M_2 gehörigen Trainingsdaten haben ungefähr die gleiche Streuung. In diesem Fall wird von den Projektionen auf die erste Hauptkomponente keine der beiden Mengen priorisiert; stattdessen wird stärker versucht, die Streuung der Trainingsdaten *normal* zu den Mengen M_1, M_2 zu beschreiben. Dieser Fall ist besonders interessant, denn hier können die Projektionen zur *Trennung* der Trainingsdaten verschiedener Mengen genutzt werden.

Da die Abstände der Trainingsdaten während der Kern-basierten PCA anschaulich mittels D^K beschrieben werden, kann durch geeignete Wahl von γ möglicherweise die Situation (2) hergestellt werden. Damit die Nutzung des Gauß-Kerns für die Trennung von Daten vielversprechend ist, ist also eine gute Wahl von γ nötig.

Für mehr als zwei Mengen ist solch eine Wahl von γ entsprechend schwieriger oder vielleicht gar nicht möglich.

Beispiel 10.22 (Geometrie des zentrierten Polynomkerns) Der zentrierte Polynomkern aus Gl. (10.15) mit $d = p = 2$ lautet $K(x, x') = (x^T x')^2$ und liefert

$$D^K(x, x') = \|x\|_2^4 - 2(x^T x')^2 + \|x\|_2^4.$$

Im Gegensatz zum Gauß-Kern ist hier keine einfache Interpretation des Abstandsmaßes möglich. Wir nutzen stattdessen die Interpretation aus (I1). Hier ist die zugehörige Funktion $h : \mathbb{R}^2 \to \mathbb{R}^3$ leicht identifizierbar (vgl. Beispiel 4.35):

$$K(x, x') = h(x)^T h(x'), \quad mit \quad h(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$$

Anschaulich kann man sich daher eine Anwendung des Polynomkerns vorstellen wie bei der Durchführung des Beispiels 10.16: Die Daten werden in einen dreidimensionalen Raum projiziert; die ersten beiden Hauptkomponenten liefern eine Perspektive des entstehenden dreidimensionalen Gebildes $\{h(X_i) - \frac{1}{n}\sum_{j=1}^n h(X_j) : i \in \mathbb{N}\}$ der zentrierten Trainingsdaten in Richtung der niedrigsten Variation der Lage. Es fehlt allerdings ein Parameter zur Skalierung wie zum Beispiel in Gl. (10.10) oder beim Gauß-Kern, womit der Polynomkern sehr unflexibel und eine erfolgreiche Anwendung „Glückssache“ ist; die vorliegenden Trainingsdaten müssen genau zu den vom Polynomkern betrachteten Richtungen passen.

Zum Verdeutlichung betrachten wir die Anwendung der Kern-basierten PCA auf ein Beispiel mit dem Ziel, die vorliegenden Trainingsdaten möglichst gut für Algorithmen zur Bestimmung von Repräsentanten (vgl. Kap. 9) vorzubereiten. Das heißt, das Ziel ist die Bereitstellung von Projektionen $\hat{\alpha}_i^{PCA,K,k}$, welche die Modellannahme dieser Algorithmen möglichst gut erfüllen.

Beispiel 10.23 Sei $M^* \in \{2, 3\}$. Der Vektor $X_1 \in \mathbb{R}^2$ sei zufällig verteilt gemäß

$$X_1 = R \cdot \begin{pmatrix} \cos(U) \\ \sin(U) \end{pmatrix},$$

wobei U gleichverteilt auf $[0, 2\pi]$ und

$$falls\ M^* = 2: \quad R = \Delta + \begin{cases} 0.3, & Z = 1, \\ 1, & Z = 2, \end{cases} \qquad falls\ M^* = 3: \quad R = \Delta + \begin{cases} 0.5, & Z = 1, \\ 1, & Z = 2, \\ 1.5, & Z = 3 \end{cases}$$

wobei Δ unabhängig von U gleichverteilt auf $[-0.1, 0.1]$, und Z eine von U, Δ unabhängige Zufallsvariable mit $\mathbb{P}(Z = i) = \frac{1}{M^*}$ $(i = 1, ..., M^*)$ ist.

Wir betrachten zunächst den Fall $M^* = 2$ und erzeugen $n = 400$ Trainingsdaten. Die unterschiedlichen Farben stellen die Zugehörigkeit der Trainingsdaten zu den Realisierungen $Z_i \in \{1, 2\}$ dar und dienen dem besseren Verständnis der Wirkung der Algorithmen. Sie sind während der Anwendung unbekannt. In Abb. 10.6 sind die Ergebnisse für $\hat{\alpha}_i^{PCA,K,2}$ für den zentrierten Polynomkern $K = K_2^{poly}$ und den Gauß-Kern $K = K_\gamma^{gauss}$ für verschiedene $\gamma > 0$ dargestellt. Wie man sieht, liefert der Polynomkern kein zufriedenstellendes Ergebnis; es wird keine günstige Richtung zur Betrachtung der Daten ausgewählt. Beim Gauß-Kern mit $\gamma = 10$ werden die kleinen Abstände zu stark vergrößert und die erste Hauptkomponente legt zu viel Wert auf eine gute Beschreibung des inneren Kreisrings. Bei $\gamma = 0.1$ werden die Abstände durch Verwendung des Gauß-Kerns kaum verändert und die Beschreibung des großen Kreisrings wird priorisiert. Mit $\gamma = 3$ jedoch haben die Daten beider Kreisringe in etwa die gleiche Variation untereinander und die erste Hauptkomponente versucht die Variation der Trainingsdaten *normal* zu den beiden Kreisringen gut zu beschreiben, womit eine Separierung der zu verschiedenen Z_i gehörigen Trainingsdaten nur mit Hilfe der ersten Hauptkomponente möglich ist.

Wir betrachten nun noch das Modell aus Beispiel 10.23 im Falle $M^* = 3$ mit $n = 600$ Trainingsdaten. Die Projektionen $\hat{\alpha}_i^{PCA,K,2}$ des Kern-basierten PCA-Algorithmus mit $K = K_2^{poly}$ und $K = K_{1.3}^{gauss}$ sind in Abb. 10.7 dargestellt. Wie man sieht, liefert der Polynomkern hier eine leicht bessere Perspektive auf die Trainingsdaten (verursacht durch die zusätzliche Streuung der Daten des äußersten Kreises); mit dem Gauß-Kern können die Daten zu verschiedenen Z_i nicht mittels der ersten oder zweiten Hauptkomponente getrennt werden (wir haben hier mit $\gamma \in \{0.3, 1, 5, 30\}$ all diejenigen γ ausgewählt welche wesentlich verschiedene Resultate $\hat{\alpha}_i^{PCA,K,2}$ liefern).

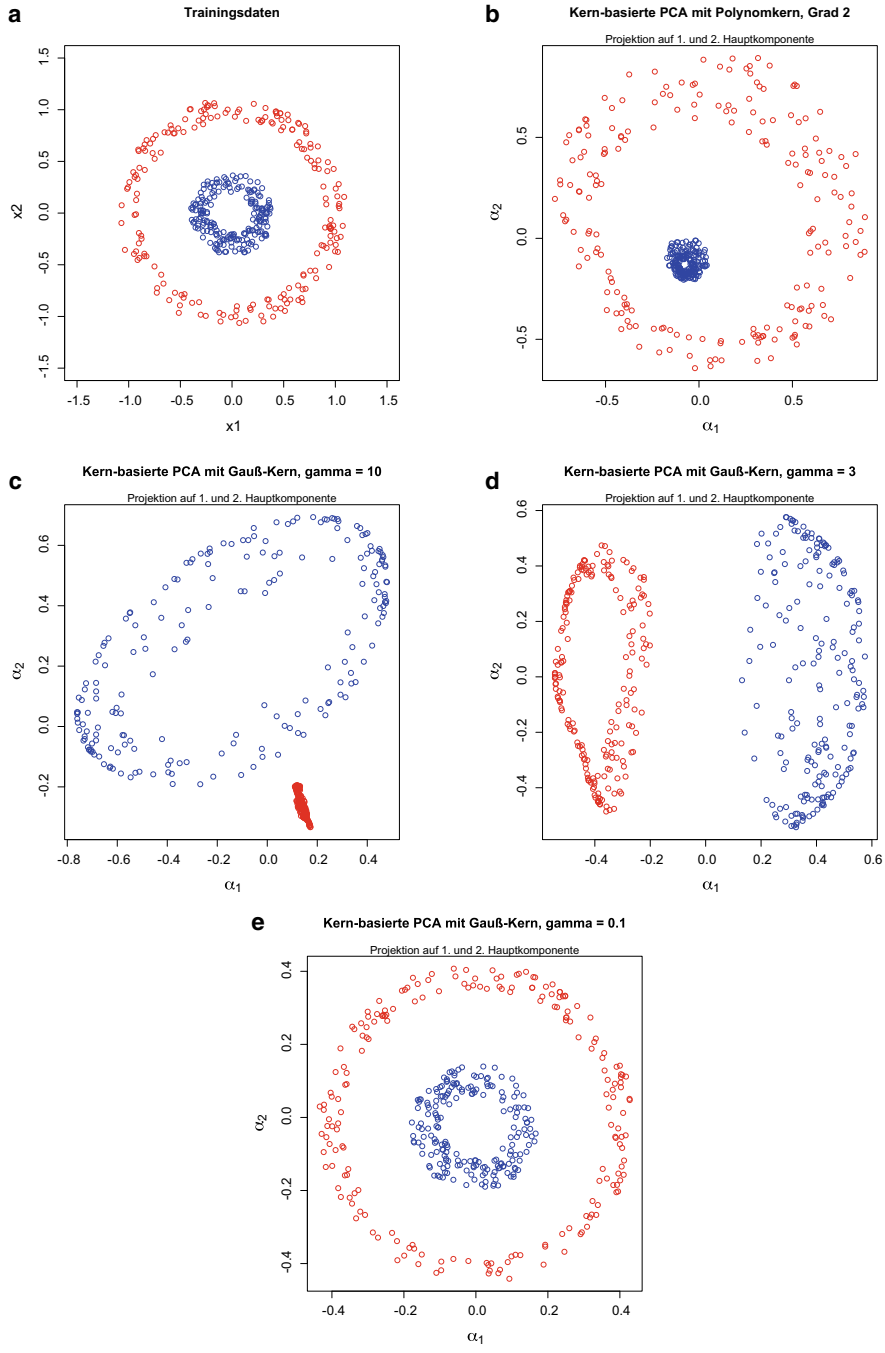


Abb. 10.6 Anwendung der Kern-basierten PCA aus Definition 10.20 auf die Trainingsdaten aus Beispiel 10.23 mit $M^* = 2$. **a** Trainingsdaten, **b** $\hat{\alpha}_i^{PCA,K,2}$ mit zentriertem Polynomkern $K = K_2^{poly}$, **c, d, e** $\hat{\alpha}_i^{PCA,K,2}$ mit Gauß-Kern $K = K_\gamma^{gauss}$ mit $\gamma \in \{0.1, 3, 10\}$

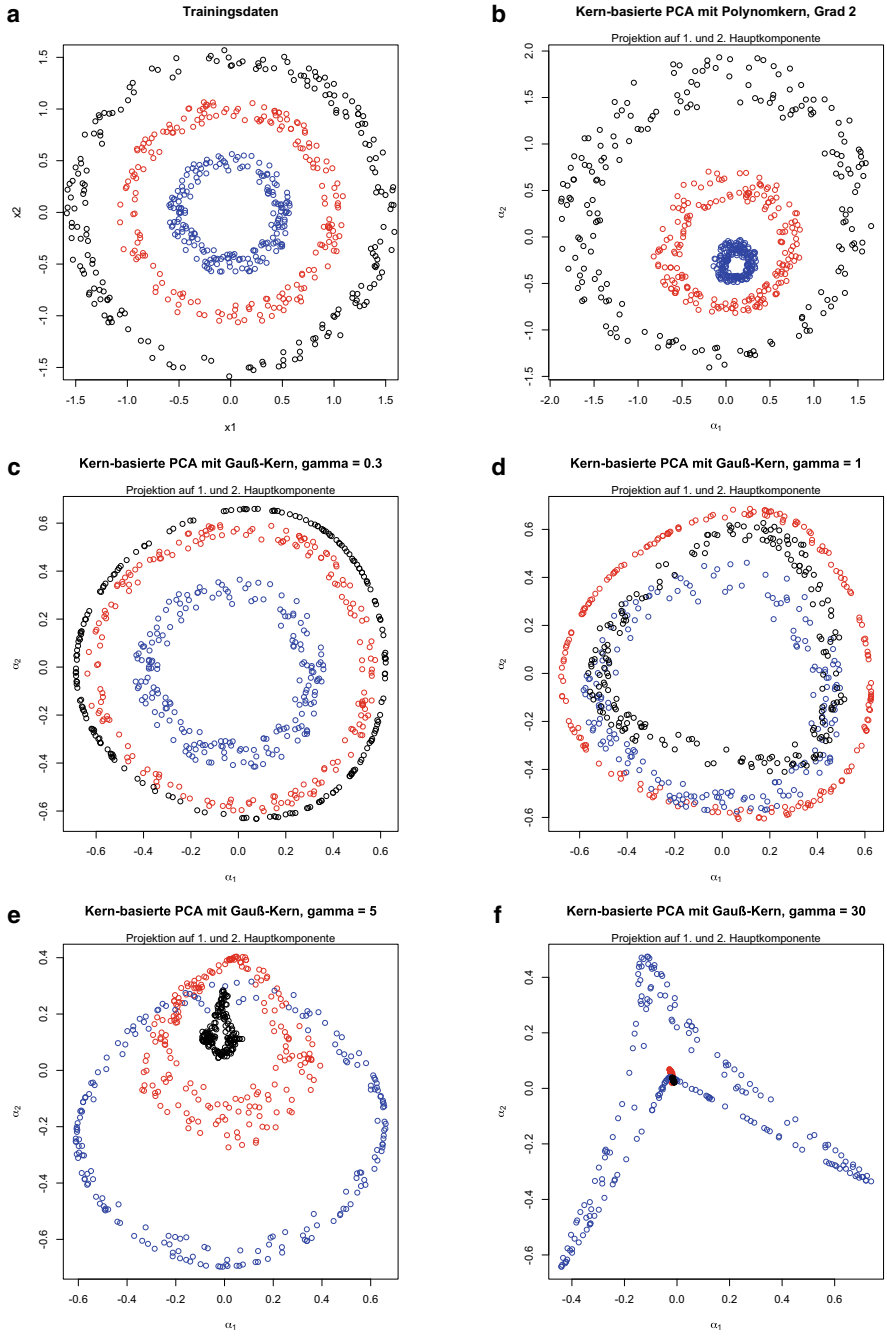


Abb. 10.7 Anwendung der Kern-basierten PCA aus Definition 10.20 auf die Trainingsdaten aus Beispiel 10.23 mit $M^* = 3$. **a** Trainingsdaten, **b** $\hat{\alpha}_i^{PCA,K,2}$ mit zentriertem Polynomkern $K = K_2^{poly}$, **c**–**f** $\hat{\alpha}_i^{PCA,K,2}$ mit Gauß-Kern $K = K_\gamma^{gauss}$ mit $\gamma \in \{0.3, 1, 5, 30\}$

10.1.5 Theoretische Resultate

Bereits in Gl. (10.11) haben wir den Rekonstruktionsfehler für den nichtlinearen PCA-Algorithmus motiviert. Für die Kern-basierte PCA mit Kern K leiten wir daraus eine Definition des Rekonstruktionsfehlers her. Sei $h : \mathbb{R}^d \to Z_{\tilde{d}}$ eine Funktion mit der Eigenschaft aus Gl. (10.14), wobei

$$Z_{\tilde{d}} = \begin{cases} \mathbb{R}^{\tilde{d}}, & \tilde{d} < \infty, \\ \ell^2, & \tilde{d} = \infty. \end{cases}$$

Im Falle $\tilde{d} = \infty$ nutzen wir zur formal korrekten Beschreibung der auftauchenden „unendlichdimensionalen“ Vektoren den Folgenraum ℓ^2 aus Gl. (4.36):

$$\ell^2 = \{(a_k)_{k\in\mathbb{N}} | \text{ Für alle } k \in \mathbb{N} \text{ ist } a_k \in \mathbb{R} \text{ und } \sum_{k=1}^{\infty} a_k^2 < \infty\}$$

Mit dem Skalarprodukt $\langle a, b\rangle := a^T b := \sum_{k\in\mathbb{N}} a_k b_k$ wird ℓ^2 ein Hilbert-Raum. Mit

$$\Sigma^h := \mathbb{E}[(h(X_1) - \mathbb{E}h(X_1))(h(X_1) - \mathbb{E}h(X_1))^T]$$

meinen wir dann $\Sigma^h_{ij} := \mathbb{E}[(h(X_1) - \mathbb{E}h(X_1))_i (h(X_1) - \mathbb{E}h(X_1))_j], i, j \in \mathbb{N}$ und für $v \in \ell^2$ ist die Matrixmultiplikation $\Sigma^h v \in \ell^2$ definiert durch

$$(\Sigma^h v)_j = \sum_{l\in\mathbb{N}} \Sigma^h_{jl} \cdot v_l, \quad j \in \mathbb{N}.$$

Genauso ergibt sich die Definition von Eigenwerten und Eigenvektoren. Eine geeignete Definition für den Rekonstruktionsfehler der Kern-basierten PCA ergibt sich nun wie in Definition 10.9.

Definition 10.24 (Rekonstruktionsfehler bei der Kern-basierten PCA)
Sei $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ ein Mercer-Kern und $h : \mathbb{R}^d \to Z_{\tilde{d}}$ eine Abbildung mit der Eigenschaft Gl. (10.14). Ist $(w^{(j)})_{1\leq j\leq \tilde{d}}$ ein Orthonormalsystem in $Z_{\tilde{d}}$, $k \in \mathbb{N}$ und $\Pi_k := \sum_{j=1}^{k} w^{(j)}(w^{(j)})^T$, so heißt

$$R^h(\Pi_k) := \mathbb{E}\left\|(h(X_1) - \mathbb{E}h(X_1)) - \Pi_k(h(X_1) - \mathbb{E}h(X_1))\right\|_2^2$$

Rekonstruktionsfehler zur Projektion Π_k. ♦

Wie in Lemma 10.10 ergibt sich, dass der Rekonstruktionsfehler minimal wird, falls Π_k aus den ersten k Eigenvektoren von Σ^h entsteht.

Lemma 10.25 Es gilt

$$\inf_{w^{(1)},\ldots,w^{(k)}\ orthonormal} R^h(\Pi_k) = R^h(\Pi_k^*) = \sum_{j=k+1}^{\tilde{d}} \lambda_j^K,$$

wobei $(v^{(j)})_{1\le j\le\tilde{d}}$ die Eigenvektoren zu den Eigenwerten $\lambda_1^K \ge \lambda_2^K \ge \ldots \ge 0$ von Σ^h bezeichnen und $\Pi_k^* := \sum_{j=1}^k v^{(j)}(v^{(j)})^T$.

Das Excess Bayes Risk des Rekonstruktionsfehlers für den Kern-basierten PCA-Algorithmus, welcher (implizit) eine Schätzung $\hat{v}^{(1)}, \ldots, \hat{v}^{(k)} \in Z_{\tilde{d}}$ der ersten k Eigenvektoren von Σ^h zur Bestimmung von $\hat{\alpha}_i^{PCA,K,k}$, $i = 1, \ldots, n$ nutzt, lautet entsprechend

$$\mathbb{E}R(\hat{\Pi}_k) - R(\Pi_k^*), \qquad \hat{\Pi}_k := \sum_{j=1}^{k} \hat{v}^{(j)}(\hat{v}^{(j)})^T.$$

Bemerkung 10.26 Aufgrund Lemma 10.25 ist klar, dass die relative Größe der Eigenwerte λ_j^K, $1 \le j \le \tilde{d}$ von Σ^h zueinander die „Schwierigkeit" des Problems der Rekonstruktion von X_1 aus den Projektionen von $h(X_1) - \mathbb{E}h(X_1)$ auf die ersten k Hauptkomponenten codieren:

- Ist $R(\Pi_k^*) = 0$ oder äquivalent dazu $\sum_{j=k+1}^{\tilde{d}} \lambda_j^K = 0$, so kann $h(X_1) - \mathbb{E}h(X_1)$ (und damit auch X_1) nur auf Basis der Projektionen auf die ersten k Hauptkomponenten vollständig rekonstruiert werden.
- Ist hingegen $R(\Pi_k^*) = \sum_{j=k+1}^{\tilde{d}} \lambda_j^K$ im Verhältnis zu $\sum_{j=1}^k \lambda_j^K$ sehr groß, so liefern die ersten k Hauptkomponenten nur sehr eingeschränkte Informationen über die Lage von $h(X_1) - \mathbb{E}h(X_1)$ und viele weitere Hauptkomponenten wären nötig, um die Lage genauer zu beschreiben.
- Allgemein gibt das Verhältnis
$$\frac{\sum_{j=1}^k \lambda_j^K}{\sum_{j\in\mathbb{N}} \lambda_j^K}$$
ein Maß dafür an, wie gut X_1 aus den Projektionen auf die ersten k Hauptkomponenten ermittelt werden kann.

Man kann zeigen (vgl. Satz 10.28(i)), dass im Allgemeinen *alle* Eigenwerte λ_j^K, $j = 1, \ldots, \tilde{d}$ Einfluss auf das Excess Bayes Risk $\mathbb{E}R(\hat{\Pi}_k) - R(\Pi_k^*)$ nehmen. Analog zu Definition 10.12 formuliert man daher für eine vorgegebene, monoton fallende nichtnegative Folge $c(j))_{j=1,\ldots,\tilde{d}}$ reeller Zahlen:

Modellannahme 10.27 (Kern-basierte PCA: Typ 2 (Rekonstruktion)) Es gelte $\lambda_j^K \leq c(j)$ $(j = 1, ..., \tilde{d})$.

In [24] werden Aussagen über die Konvergenzgeschwindigkeit von $\mathbb{E}R(\hat{\Pi}_k) - R(\Pi_k^*)$ in Abhängigkeit von k, n und $\tilde{d}$ getroffen. Es gilt (vgl. [24, Theorem 2.9 und Section 2.4]):

Satz 10.28 Es sei $\mathbb{E}K(X_1, X_1) < \infty$ und es gebe eine Konstante $C_1 > 0$, so dass für alle $u \in Z_{\tilde{d}}$ gilt:

$$\sup_{l \in \mathbb{N}} l^{-1/2} \mathbb{E}[|K(X_1, u)|^k]^{1/k} \leq C_1 \mathbb{E}[K(X_1, u)^2]^{1/2} \tag{10.16}$$

Es sei weiter $k \leq \frac{n}{16C_3^2}$ mit einer Konstanten $C_3 > 0$, und es gelte die Modellannahme 10.27.

(i) Dann gibt es eine Konstante $c > 0$ (nur abhängig von C_1, C_3), so dass

$$\mathbb{E}R(\hat{\Pi}_k) - R(\Pi_k^*) \leq c \cdot n^{-1/2} \Big(\sum_{j=1}^{\infty} c(j) \Big)^{1/2} \cdot \Big[\sum_{j=1}^{k} \sqrt{c(j)} + k^{1/2} \sum_{j=k+1}^{\tilde{d}} c(j) \Big] \tag{10.17}$$

und $R(\Pi_k^*) \leq \sum_{j=k+1}^{\tilde{d}} c(j)$.

(ii) Ist speziell $c(j) = e^{-\rho j}$ mit einem $\rho > 0$, so gibt es eine Konstante $c > 0$ (nur abhängig von C_1, C_3, ρ), so dass

$$\mathbb{E}R(\hat{\Pi}_k) - R(\Pi_k^*) \leq c \frac{\min\{k, \log(e \cdot n)\}}{n}, \qquad R(\Pi_k^*) \leq \frac{e^{-\rho}}{1 - e^{-\rho}} \cdot e^{-\rho k}.$$

Bemerkungen

- Die tatsächliche Qualität des Kern-basierten PCA-Algorithmus wird durch den erwarteten Generalisierungsfehler $\mathbb{E}R(\hat{\Pi}_k)$ gemessen. Für diesen gilt

$$\mathbb{E}R(\hat{\Pi}_k) = \big[\mathbb{E}R(\hat{\Pi}_k) - R(\Pi_k^*)\big] + R(\Pi_k^*), \tag{10.18}$$

  womit durch die Aussagen (i),(ii) jeweils eine Abschätzung möglich ist. Eine zentrale von Satz 10.28 ist, dass der Term nicht *unmittelbar* von der Dimension des Raums $Z_{\tilde{d}}$, in welchen die Trainingsdaten eingebettet werden abhängt, sondern nur über die Terme $\sum_{j=1}^{\tilde{d}} \lambda_j^*$, $\sum_{j=1}^{k} \sqrt{\lambda_j^*}$ und $\sum_{j=k+1}^{\tilde{d}} \lambda_j^*$. Die durchgeführte Einbettung in einen möglicherweise unendlichdimensionalen Raum führt also im Allgemeinen nicht zu einer drastischen Verschlechterung der Konvergenzrate.
- Gl. (10.16) ist eine Momentenbedingung an die Zufallsvariable $K(X_1, u)$. Sie ist beispielsweise erfüllt, wenn
  - $K(x, x') = x^T x'$ der lineare Kern und X_1 normalverteilt ist oder
  - K beschränkt ist, z. B. falls K der Gauß-Kern ist.

Mathematische Bedeutung der Projektionen

In diesem Abschnitt leiten wir heuristisch her, welches Objekt durch die Projektionen $\hat{\alpha}_i^{PCA,K,k} = (\hat{\alpha}_{i1}^K, ..., \hat{\alpha}_{ik}^K)^T$ geschätzt wird bzw. wie sich die Güte von $\hat{\alpha}_i^{PCA,K,k}$ bewerten lässt. Gemäß Definition 10.20 gilt

$$\hat{\alpha}_{ij}^K = \hat{A}_n^K(X_i)_j, \qquad i = 1, ..., n, \quad j = 1, ..., k,$$

wobei

$$\hat{A}_n^K(x)_j = \frac{1}{n\hat{\lambda}_j^K} \sum_{i=1}^n \hat{\alpha}_{ij}^K \cdot \hat{K}_0(x, X_i). \tag{10.19}$$

Mit anderen Worten, $\hat{\alpha}_{\cdot j}^K = (\hat{\alpha}_{1j}^K, ..., \hat{\alpha}_{nj}^K)^T$ entsteht durch sukzessives Einsetzen der Trainingsdaten X_i, $i = 1, ..., n$ in die Projektionsabbildung $\hat{A}_n^K(x)_j : \mathbb{R}^d \to \mathbb{R}$.

Im Kern-basierten PCA-Algorithmus werden die Eigenvektoren $\hat{v}^{(j)}$ nicht mehr explizit ermittelt, sie sind aber theoretisch als Eigenvektoren von $\hat{\Sigma}^h$ zu den Eigenwerten $\hat{\lambda}_j^K$ berechenbar. Für großes $n \in \mathbb{N}$ erwarten wir $\hat{\Sigma}^h \approx \Sigma^h$ und daher $\hat{v}^{(j)} \approx v^{(j)}$, $\hat{\lambda}_j^K \approx \lambda_j^K$ (Notation aus Lemma 10.25). Weiter gilt

$$\hat{A}_n^K(x)_j = (h(x) - \frac{1}{n}\sum_{i=1}^n h(X_i))^T \hat{v}^{(j)},$$

entsprechend erwarten wir

$$\hat{A}_n^K(x)_j \to (h(x) - \mathbb{E}h(X_1))^T v^{(j)} =: A^K(x)_j$$

Setzen wir $A^{K,k}(x) := (A^K(x)_1, ..., A^K(x)_k)^T$, so folgt: *Die Werte*

$$\hat{\alpha}_i^{PCA,K,k} = \hat{A}_n^{K,k}(X_i), \quad i = 1, ..., n$$

entsprechen Schätzungen der „optimalen" Projektion $A^{K,k}(X_i)$ von X_i auf die ersten k Hauptkomponenten, und wir erwarten, dass

$$\sup_{i=1,...,n} |\hat{\alpha}_i^{PCA,K,k} - A^{K,k}(X_i)| = \sup_{i=1,...,n} |\hat{A}_n^{K,k}(X_i) - A^{K,k}(X_i)| \to 0 \quad (n \to \infty).$$

Durch immer mehr Trainingsdaten ($n \to \infty$) wird also die „wahre" Projektionsabbildung $A^{K,k}$ immer genauer bestimmt und die Schätzungen $\hat{\alpha}_{ij}^K$ für $A^K(X_i)_j$ werden immer besser. In Abschn. 10.2.4 wird im Kontext des spektralen Clusterns ein theoretisches Resultat für diese Aussage präsentiert.

Alternative Definition der „wahren" Projektion

Eine mathematische Definition von $A^K(\cdot)_j$ ($j = 1, ..., k$) ohne die explizite Verwendung der Eigenvektoren $v^{(j)}$ ist wie folgt möglich: Gl. (10.19) ist äquivalent zu

$$\hat{\lambda}_j^K \cdot \hat{A}_n^K(x)_j = \frac{1}{n}\sum_{i=1}^{n} \hat{A}_n^K(X_i)_j \cdot \hat{K}_0(x, X_i), \quad x \in \mathbb{R}^d,$$

d. h. für $n \to \infty$ erwarten wir

$$\lambda_j^K \cdot A^K(x)_j = \mathbb{E}\big[K_0(x, X_1)A^K(X_1)_j\big], \quad x \in \mathbb{R}^d,$$

wobei

$$K_0(x, y) := K(x, y) - \mathbb{E}K(X_1, y) - \mathbb{E}K(x, X_1) - \mathbb{E}K(X_1, X_2).$$

Das bedeutet: $A^K(\cdot)_j$ ist Eigenfunktion des Operators

$$T_K : L^2(\mathbb{P}^{X_1}) \to L^2(\mathbb{P}^{X_1}), \quad (T_K g)(x) := \mathbb{E}\big[K_0(x, X_1)g(X_1)\big] \tag{10.20}$$

zum Eigenwert λ_j^K, wobei

$$L^2(\mathbb{P}^{X_1}) := \{g : \mathbb{R}^d \to \mathbb{R} \text{ messbar} : \mathbb{E}[g(X_1)^2] < \infty\}. \tag{10.21}$$

Wegen $\|\hat{\alpha}_{\cdot j}^K\|_2^2 = n\hat{\lambda}_j^K$ erwarten wir außerdem

$$\mathbb{E}[A^K(X_1)_j^2] \leftarrow \frac{1}{n}\sum_{i=1}^{n}(\hat{A}_n^K(X_i)_j)^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{\alpha}_{ij}^K)^2 = \hat{\lambda}_j^K \to \lambda_j^K \quad (n \to \infty),$$

was eine Normierung der Funktionen $A^K(x)_j$ und damit eine eindeutige Eigenfunktion charakterisiert. Wir erhalten:

Definition 10.29
Sei $k \in \mathbb{N}$. Es seien $\lambda_1^K > \lambda_2^K > ... > \lambda_k^K \geq 0$ die k größten Eigenwerte des Operators aus Gl. (10.20) mit zugehörigen Lösungen g_j, $j = 1, ..., k$ mit $\mathbb{E}[g_j(X_1)^2] = \lambda_j^K$. Dann heißt

$$A^{K,k} : \mathbb{R}^d \to \mathbb{R}^k, \quad A^{K,k}(x) := \big(A^K(x)_1, ..., A^K(x)_k\big)^T$$

mit $A^K(\cdot)_j := g_j$ die zur Verteilung $\mathbb{P}^{X_1}$ und zum Kern K assoziierte *wahre Projektionsabbildung der Kern-basierten PCA.* ♦

Werden die reduzierten Trainingsdaten $\hat{\alpha}_i^{PCA,K,k}$ zur Verarbeitung mit einem weiteren Verfahren V des maschinellen Lernens genutzt, so können wir nun mit Hilfe von $A^{K,k}$ die entsprechende Modellannahme analog zu Modellannahme 10.8 kompakt formulieren.

Modellannahme 10.30 (Kern-basierte PCA: Typ 1 (Weiterverarbeitung)) *Weiterverarbeitung der Projektionen auf die ersten k Hauptkomponenten:*

- Supervised Learning: Die Modellannahme von V gelte für $(A^{K,k}(X_1), Y_1)$ (und nicht für (X_1, Y_1)).
- Unsupervised Learning: Die Modellannahme von V gelte für $A^{K,k}(X_1)$ (und nicht für X_1).

10.2 Spektrales Clustern

Bei der Kern-basierten Hauptkomponentenanalyse stand im Vordergrund, die Trainingsdaten in der Dimension zu reduzieren, so dass der Rekonstruktionsfehler möglichst gering war. Indem bestimmte Komponentenkombinationen oder Eigenschaften der Daten durch Skalierung o. Ä. besonders herausgehoben werden, werden diese vom PCA-Algorithmus als nötig für eine gute Rekonstruktion erachtet und finden auch Eingang in die reduzierten Trainingsdaten. Daher kann durch geeignete Wahl des Kerns auch der PCA-Algorithmus als Vorbearbeitung der Daten für Verfahren des Unsupervised Learnings genutzt werden.

Beim *spektralen Clustern* wählt einen anderen Ansatz: Aus den gegebenen Trainingsdaten $X_i \in \mathbb{R}^d$, $i = 1, ..., n$ sollen ebenfalls reduzierte Daten $\alpha_i \in \mathbb{R}^k$, $i = 1, ..., n$ mit $k < d$ ermittelt werden, die allerdings keine möglichst gut Rekonstruktion *der Lage* von X_i liefern sollen, sondern von deren *Abständen zueinander*. Die Anwendung erfolgt mit dem Ziel, insbesondere Algorithmen des Unsupervised Learnings zum Trennen der Daten bzw. zum Zuordnen zu Repräsentanten auf α_i anwenden zu können. Wir werden das spektrale Clustern daher zunächst mit Hilfe eines graphentheoretischen Problems zur Trennung von Daten motivieren. Im zweiten Teil des Abschnitts formalisieren wir das Problem des Findens geeigneter niedrigdimensionaler Daten, welche die Abstände zwischen den ursprünglichen Trainingsdaten näherungsweise erhalten. Eine wichtige Rolle für die Interpretation der Optimierer spielt hierbei die verwendete Normierungsbedingung, die wir aus der grafischen Motivation ableiten.

Nach der vollständigen Formalisierung des Problems werden wir sehen, dass bis auf die Normierungsbedingung nahezu dasselbe Problem wie bei der Kern-basierten Hauptkomponentenanalyse gelöst wird, was die Übernahme der jeweiligen Interpretationen der Optimierer ermöglicht.

10.2.1 Optimale Graph Cuts

Wir geben im Folgenden zunächst einen grafisch motivierten Ansatz, wie gegebene Trainingsdaten X_i, $i = 1, ..., n$ anhand ihrer Abstände zueinander möglichst sinnvoll in zwei Gruppen eingeteilt werden können. Die Trainingsdaten X_i und deren „Ungleichheiten“ untereinander können anschaulich mit einem gewichteten Graphen dargestellt werden. Wir führen zunächst die formale Definition ein:

Definition 10.31 (Gewichteter Graph)
Ein gewichteter Graph ist ein Tripel $G = (V, E, W)$, wobei

- V die Menge der Knoten *(vertices),*
- $E \subset V \times V$ die Menge der Kanten *(edges),*
- $W \in \mathbb{R}^{\#V \times \#V}$ eine symmetrische Matrix ist, deren Komponenten W_{ij} das Gewicht *(weights)* zwischen zwei Knoten $i, j \in E$ beschreibt.

♦

Sind Trainingsdaten X_i und eine Abstandsfunktion $D : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ gegeben, so induzieren diese einen gewichteten Graphen:

Definition 10.32 (Induzierter reduzierter ε-Nachbarschaftsgraph)
Sei $\varepsilon > 0$. Sei $V := \{1, ..., n\}$ die Menge der Indizes der Trainingsdaten, und setze

$$(i, j) \in E \quad \Longleftrightarrow \quad D(X_i, X_j) < \varepsilon, \quad i \neq j. \tag{10.22}$$

Setze für ein $\gamma \geq 0$:

$$W_{ij} := \exp\big(-\gamma \cdot D(X_i, X_j)\big) \cdot \mathbb{1}_{\{(i,j)\in E\}} \tag{10.23}$$

Der Graph $G_n := (V, E, W)$ heißt *induzierter reduzierter ε-Nachbarschaftsgraph.* ♦

Bemerkung 10.33

- Durch die Charakterisierung der Kanten mittels Gl. (10.22) werden Verbindungen zwischen X_i, X_j mit sehr großen Abständen $D(X_i, X_j)$ entfernt.
- Ist $\gamma > 0$, so sind die Gewichte W_{ij} ein Maß für die *„Gleichheit"* zweier Beobachtungen X_i, X_j. Ist W_{ij} also sehr *klein,* so sind X_i, X_j sehr *verschieden.* Im Extremfall $D(X_i, X_j) \geq \varepsilon$ ist $W_{ij} = 0$.

Beispiel 10.34 Wir betrachten sechs Punkte $X_1, ..., X_6 \in \mathbb{R}^2$, siehe Abb. 10.8. Der ε-Nachbarschaftsgraph hat dann die Knotenmenge $V = \{1, ..., 6\}$. Für die Wahl $\varepsilon = 6$ ergibt sich die Kantenmenge

$$E = \{(1, 3), (1, 4), (2, 4), (2, 5), (2, 6), (3, 4), (4, 5), (5, 6)\}.$$

Die symmetrische Gewichtsmatrix W für den Fall $\gamma = 0$ lautet:

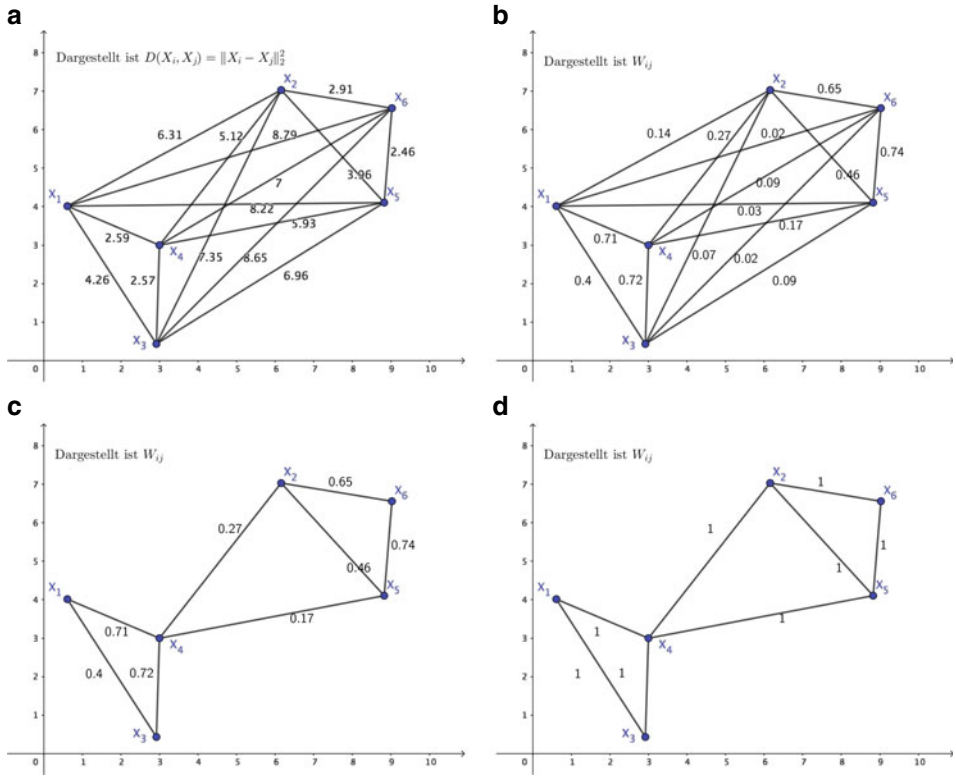


Abb. 10.8 **a**, **b** Darstellung der Abstände $D(X_i, X_j) = \|X_i - X_j\|_2^2$ und Gewichte W_{ij} für ein Beispiel in $\mathbb{R}^2$ mit sechs Punkten. **c**, **d** Darstellung der ε-Nachbarschaftsgraphen für $\varepsilon = 6$, **c** mit $\gamma = \frac{1}{20}$, **d** mit $\gamma = 0$

$$W^{(\varepsilon=6,\gamma=0)} = \begin{pmatrix} 0&0&1&1&0&0 \\ 0&0&0&1&1&1 \\ 1&0&0&1&0&0 \\ 1&1&1&0&1&0 \\ 0&1&0&1&0&1 \\ 0&1&0&0&1&0 \end{pmatrix}$$

In Anwendungen wird häufig $\varepsilon = \infty$ und $0 < \gamma < \infty$ gewählt, so dass durch die Definition (10.22) keine Kanten gelöscht werden, der Graph demzufolge fast noch alle Trainingsdaten miteinander verbindet und die Informationen über deren „Ungleichheiten" vollständig aus der Gewichtsmatrix W_{ij} gewonnen werden.

Für die Herleitung des spektralen Clusterns aus einem graphentheoretischen Problem betrachten wir aber zunächst den Fall $0 < \varepsilon < \infty$, $\gamma = 0$ und haben das Ziel, die Daten in $K^* = 2$ Mengen aufzuteilen. Im Folgenden gehen wir also davon aus, dass die Gleichheit der

X_i im Nachbarschaftsgraphen G_n allein durch das Vorhandensein einer Kante ausgedrückt wird und stets $W_{ij} \in \{0, 1\}$ gilt. Wir gehen zum besseren Verständnis des Parameters γ im Folgenden aber auch jeweils auf den Fall $\gamma > 0$ ein.

Ansatz Wir teilen die Menge der Knoten V so in zwei Mengen $S, \overline{S} := V \backslash S$ auf, dass die Gewichtssumme der dadurch gelöschten Kanten (d. h. Kanten mit Knoten in S und $\overline{S}$) möglichst klein ist. Formal bedeutet dies,

$$\sum_{i \in S, j \in \overline{S}} W_{ij} \tag{10.24}$$

über alle Teilmengen $S \subset V$ zu minimieren. In der Literatur nennt man die Bestimmung einer solchen Aufteilung *optimalen Graph Cut*. Ist $\gamma > 0$, so werden durch dieses Vorgehen vor allem Kanten mit sehr kleinen Gewichten entfernt, d. h. es werden Verbindungen zwischen sehr verschiedenen X_i, X_j gelöscht.

Obiger Ansatz hat noch ein Problem: Es gibt eine triviale Lösung $S = \emptyset$, $\overline{S} = V$, denn dann ist die Summe in Gl. 10.24 leer. Dieses Problem wird üblicherweise umgangen, indem man fordert, dass beide Teile S, $\overline{S}$ in etwa gleich viele Kanten enthalten, d. h.

$$\text{Gewicht in } S \approx \text{Gewicht in } \overline{S}. \tag{10.25}$$

Im Folgenden wollen wir diesen Ansatz formalisieren. Die Menge der Teilmengen $S \subset V$ ist für großes n sehr groß, so dass eine exakte Lösung im Allgemeinen nicht möglich ist. Wir konzentrieren uns daher von vornherein auf eine geeignete approximative Lösung. Solch eine Approximation lässt sich natürlicher im Raum $\mathbb{R}^n$ finden, da Zahlen algorithmisch leichter approximiert werden können als Mengen. Eine Beschreibung der Aufteilung $S, \overline{S}$ kann wie folgt mit einem Vektor $\alpha \in \mathbb{R}^n$ ausgedrückt werden.

Definition 10.35 (Induzierte Zerlegung)
Jedes $\alpha \in \{-1, +1\}^{\#V}$ induziert eine Zerlegung $V = S \cup \overline{S}$ mittels

$$\alpha_i = 1 \iff i \in S$$

und die Zerlegung heißt die von α induzierte Zerlegung.

Die Indikatorvektoren $\frac{1}{2}(\mathbb{1} + \alpha), \frac{1}{2}(\mathbb{1} - \alpha) \in \{0, 1\}^{\#V}$ geben dann die Zugehörigkeit von $i \in V$ an mittels

$$\left[\frac{1}{2}(\mathbb{1} + \alpha)\right]_i = 1 \iff i \in S, \qquad \left[\frac{1}{2}(\mathbb{1} - \alpha)\right]_i = 1 \iff i \in \overline{S}.$$

♦

Zur kompakten Beschreibung der Bedingung aus Gl. (10.25) benötigen wir noch die folgende Definition:

Definition 10.36 (Gradmatrix, Volumen)
Für einen gewichteten Graphen $G = (V, E, W)$ heißt die Diagonalmatrix $\mathbb{D} = \mathbb{D}(G) \in \mathbb{R}^{\#V \times \#V}$ mit

$$\mathbb{D}_{ii} := \sum_{j \in V} W_{ij}$$

die *Gradmatrix* von G. Das *Volumen* einer Menge $S \subset V$ von Knoten ist

$$\mathrm{vol}(S) := \sum_{i \in S} \mathbb{D}_{ii}.$$

♦

Im Falle $\gamma = 0$ zählt $\mathbb{D}_{ii}$, mit wie vielen anderen Knoten i verbunden ist (der *Grad* des Knotens i. Das Volumen $\mathrm{vol}(S)$ drückt dann aus, wie viele Kanten insgesamt von allen Knoten der Menge S ausgehen.

Im Folgenden formalisieren wir die durch Gl. (10.24) und Gl. (10.25) aufgestellten Größen und Bedingungen mittels des Indikatorvektors:

Bemerkung 10.37 Sei $L := \mathbb{D} - W$ die *Laplace-Matrix* und $\alpha \in \{-1, +1\}^{\#V}$. Dann gilt: (10.24) hat die Form

$$\sum_{i \in S, j \in \overline{S}} W_{ij} = \frac{1}{4} \langle \alpha, L\alpha \rangle, \tag{10.26}$$

und die Bedingung (10.25) kann formalisiert werden als

$$vol(S) = vol(\overline{S}) \quad \Longleftrightarrow \quad \langle \mathbb{D}\mathbb{1}, \alpha \rangle = 0. \tag{10.27}$$

Beweis Es gilt:

$$\begin{aligned}
&\sum_{i \in S, j \in \overline{S}} W_{ij} \\
&= \sum_{i,j \in V} W_{ij} \cdot [\frac{1}{2}(\mathbb{1} + \alpha)]_i \cdot [\frac{1}{2}(\mathbb{1} - \alpha)]_j = \frac{1}{4} \sum_{i,j \in V} W_{ij} \cdot (\underbrace{1}_{\alpha_i^2} + \alpha_i - \alpha_j - \alpha_i \alpha_j) \\
&= \frac{1}{4} \Big(\sum_{i \in V} \alpha_i^2 \underbrace{\sum_{j \in V} W_{ij}}_{\mathbb{D}_{ii}} + \underbrace{\sum_{i \in V} \alpha_i \sum_{j \in V} W_{ij} - \sum_{j \in V} \alpha_j \sum_{i \in V} W_{ij}}_{=0 \quad (da\ W = W^T)} - \langle \alpha, W\alpha \rangle \Big) \\
&= \frac{1}{4} \Big(\langle \alpha, \mathbb{D}\alpha \rangle - \langle \alpha, W\alpha \rangle \Big)
\end{aligned}$$

Weiter ist

$$\begin{aligned} & vol(S) = vol(\overline{S}) \\ \iff & \sum_{i \in V} \mathbb{D}_{ii} \cdot \frac{1}{2}(1+\alpha_i) = \sum_{i \in V} \mathbb{D}_{ii} \cdot \frac{1}{2}(1-\alpha_i) \\ \iff & 0 = \sum_{i \in V} \mathbb{D}_{ii} \cdot \Big[\frac{1}{2}(1+\alpha_i) - \frac{1}{2}(1-\alpha_i)\Big] = \sum_{i \in V} \mathbb{D}_{ii}\alpha_i. \end{aligned}$$

Wir können nun einen optimalen Graph Cut mittels des Indikatorvektors $\alpha \in \{-1, +1\}^{\#V}$ beschreiben:

Definition 10.38 (Optimaler Graph Cut)
Sei $G = (V, E, W)$ ein gewichteter Graph. Sei $\hat{\alpha}^{OGC} \in \{-1, +1\}^{\#V}$ eine Lösung von

$$\min_{\alpha \in \{-1,+1\}^{\#V}} \frac{1}{4}\langle \alpha, L\alpha \rangle \quad \text{unter NB} \quad \langle \mathbb{D}\mathbb{1}, \alpha \rangle = 0.$$

Dann heißt $\hat{\alpha}^{OGC}$ *optimaler Graph Cut* von G. ♦

Beispiel 10.39 (Fortsetzung von Beispiel 10.34) Hier ist

$$\mathbb{D} = \mathrm{diag}(2, 3, 2, 4, 3, 2).$$

Die Nebenbedingung $\langle \mathbb{D}\mathbb{1}, \alpha \rangle = 0 \iff \mathrm{vol}(S) = \mathrm{vol}(\overline{S})$ erlaubt insgesamt zwölf Aufteilungen für $S, \overline{S}$. Ohne Beschränkung der Allgemeinheit kann $\{2\} \in S, \{5\} \in \overline{S}$ angenommen werden, so dass noch folgende sechs Kombinationen relevant sind: Jede dieser Aufteilungen entspricht also einem optimalen Graph Cut. Die naheliegende Aufteilung $S = \{1, 3, 4\}$, $\overline{S} = \{2, 5, 6\}$ ist allerdings aufgrund der zu erfüllenden Nebenbedingung nicht enthalten.

S	$\overline{S}$	$\frac{1}{4}\langle \alpha, L\alpha \rangle = \sum_{i \in S, j \in \overline{S}} W_{ij}$
$\{1, 2, 4\}$	$\{3, 5, 6\}$	7
$\{2, 3, 4\}$	$\{1, 5, 6\}$	7
$\{2, 4, 6\}$	$\{1, 3, 5\}$	7
$\{2, 3, 6\}$	$\{1, 4, 5\}$	7
$\{1, 2, 6\}$	$\{3, 4, 5\}$	7
$\{1, 2, 3\}$	$\{4, 5, 6\}$	7

Würde man stattdessen den ε-Nachbarschaftsgraphen mit $\varepsilon = 5.5$ betrachten, so würde die Kante $(4, 5)$ im Graphen fehlen und es ergäbe sich der optimale Graph Cut $S = \{1, 3, 4\}$, $\overline{S} = \{2, 5, 6\}$ mit $\frac{1}{4}\langle \alpha, L\alpha \rangle = 2$.

Wie an Beispiel 10.34 gesehen werden kann, hat die Formulierung noch mehrere Nachteile:

- Die Anzahl der zu überprüfenden Indikatorvektoren $\alpha \in \{-1, +1\}^{\#V}$ beträgt $2^{\#V}$, wächst also exponentiell in $\#V$. Im Falle des induzierten ε-Nachbarschaftsgraphen der Beobachtungen X_i ist $\#V = n$ die Anzahl der Trainingsdaten.
- Die Bedingung $\langle \mathbb{D}\mathbb{1}, \alpha \rangle = 0$ bzw. $\mathrm{vol}(S) = \mathrm{vol}(\overline{S})$ ist sehr strikt und schließt möglicherweise die „visuell erwartete“ Lösung aus.

Durch die im Folgenden besprochene Approximation werden beide Probleme simultan gelöst.

Ansatz Ähnlich wie beim Ansatz zur Klassifizierung (vgl. Abschn. 3.3) erweitern wir den Raum der möglichen Indikatorvektoren α von $\{-1, +1\}^{\#V}$ auf $\mathbb{R}^{\#V}$. Während zuvor alle Möglichkeiten für α einzeln geprüft werden mussten, wandelt sich das Problem nun in ein stetiges Optimierungsproblem in $\mathbb{R}^{\#V}$ um. Zusätzlich wird die Bedingung $\langle \mathbb{D}\mathbb{1}, \alpha \rangle = 0$ durch die mögliche Abweichung von α_i zu ± 1 abgeschwächt.

Durch die Erweiterung auf $\alpha \in \mathbb{R}^{\#V}$ wird allerdings die Normierung von α aufgegeben. Insbesondere wäre dann $\alpha \equiv 0$ eine Lösung. Wir müssen daher wieder eine Normierung von α einführen (d. h. *eine* Gleichung für die Komponenten von α). Eine sinnvolle Bedingung lautet

$$\langle \mathbb{D}\alpha, \alpha \rangle = \sum_{i \in V} \mathbb{D}_{ii} \alpha_i^2 \stackrel{!}{=} \sum_{i \in V} \mathbb{D}_{ii}, \tag{10.28}$$

denn diese wäre im Falle von $\alpha_i = \pm 1$ wegen $\alpha_i^2 = 1$ immer erfüllt. Tatsächlich verwenden wir die Normierung „$= \#V$“ auf der rechten Seite von Gl. (10.28) (in der Praxis wird teilweise auch „$= 1$“ genutzt). Dies ist keine Einschränkung; die ursprünglichen α können dann durch Multiplikation mit $\left(\frac{\sum_{i \in V} \mathbb{D}_{ii}}{\#V}\right)^{1/2}$ erhalten werden. Die Einführung dieser Normierungsbedingung führt dazu, dass die Lösung weiterhin versucht, die Punkte in der Nähe von $\pm\left(\frac{\#V}{\sum_{i \in V} \mathbb{D}_{ii}}\right)^{1/2}$ auszurichten.

Es ergibt sich die folgende Approximation der Lösung aus Definition 10.38.

Definition 10.40 (Approximation: Optimaler Graph Cut)
Sei $G = (V, E, W)$ ein gewichteter Graph. Sei $\hat{\alpha} \in \mathbb{R}^{\#V}$ eine Lösung von

$$\min_{\alpha \in \mathbb{R}^{\#V}} \frac{1}{4} \langle \alpha, L\alpha \rangle \quad \text{unter NB} \quad \langle \mathbb{D}\mathbb{1}, \alpha \rangle = 0, \qquad \langle \mathbb{D}\alpha, \alpha \rangle = \#V. \tag{10.29}$$

Die Zuordnung

$$\hat{\alpha}_i^{OGC,\approx} = \mathrm{sign}(\hat{\alpha}_i), \quad i = 1, ..., \#V, \qquad \text{wobei} \quad \mathrm{sign}(z) = \begin{cases} 1, & z > 0, \\ -1, & z \leq 0 \end{cases}$$

liefert $\hat{\alpha}^{OGC,\approx} \in \{-1,+1\}^{\#V}$ und heißt *approximativ optimaler Graph Cut* von G. ♦

Für dieses Optimierungsproblem existieren explizit berechenbare Lösungen, die durch die Eigenwerte und Eigenvektoren eines verallgemeinerten Eigenwertproblems gegeben sind.

Satz 10.41 Seien $\rho_j \in \mathbb{R}$, $\alpha^{(j)} \in \mathbb{R}^{\#V}$, $j = 1, ..., \#V$ Lösungen der Gleichung

$$L\alpha = \rho \mathbb{D}\alpha \tag{10.30}$$

mit $0 \leq \rho_1 \leq \rho_2 \leq ... \leq \rho_{\#V}$ und $\langle \mathbb{D}\alpha^{(j)}, \alpha^{(j)} \rangle = \#V$. Dann ist $\hat{\alpha} := \alpha^{(2)}$ eine Lösung des Optimierungsproblems (10.29).

Beweis Man erhält ein lokales Minimum durch Ableiten der zu diesem Optimierungsproblem assoziierten Lagrange-Funktion. Dass dieses lokale Minimum sogar ein globales Minimum ist, kann mit dem Courant-Fischer-Theorem bewiesen werden.

Bemerkungen

- Sei $\mathbb{1} = (1, ..., 1)^T \in \mathbb{R}^n$. Wegen $L \cdot \mathbb{1} = 0$ ist $\rho_1 = 0$ und $\alpha^{(1)} = \left(\frac{\#V}{\sum_{i \in V} \mathbb{D}_{ii}}\right)^{1/2} \mathbb{1}$. Die Lösung $\alpha^{(1)}$, welche zum kleinsten Eigenwert ρ_1 gehört, ist also allein schon aufgrund ihrer Vorzeichen nicht interessant.
- Das Optimierungsproblem in Gl. (10.30) heißt verallgemeinertes Eigenwertproblem und die Lösungen $\alpha^{(j)}$ heißen verallgemeinerte Eigenvektoren. Durch die Definition $L_N := \mathbb{D}^{-1/2} L \mathbb{D}^{-1/2}$ und $y := \mathbb{D}^{1/2}\alpha$ kann stattdessen das „normale“ Eigenwertproblem

$$L_N y = \rho y$$

  gelöst werden. Die Lösungen α werden dann erhalten durch $\alpha = \mathbb{D}^{-1/2} y$.
- Im Zusammenhang mit Gl. (10.30) nennt man den Vektor $\alpha^{(2)}$ in der Graphentheorie *Fiedler-Vektor*.

Beispiel 10.42 (Fortsetzung von Beispiel 10.34) Im Falle $\varepsilon = 6$, $\gamma = 0$ erhalten die Lösung $\alpha^{(2)} = (0.87, -0.50, 0.87, 0.27, -0.50, -0.77)^T$ und damit

$$\hat{\alpha}^{OGC,\approx} = (1, -1, 1, 1, -1, -1)^T. \tag{10.31}$$

Dies entspricht der visuell erwarteten Aufteilung $S = \{1, 3, 4\}$, $\overline{S} = \{2, 5, 6\}$.

Durch die Relaxierung des Problems ist es nun auch möglich, Gewichte $W_{ij} \in [0, 1]$ und damit $\gamma > 0$ zu verwenden. Als Beispiel betrachten wir den voll verbundenen ε-Nachbarschaftsgraphen mit $\varepsilon = \infty$, $\gamma = \frac{1}{20}$, welcher in Abb. 10.8b dargestellt ist. Dann ist

$$W^{(\varepsilon=\infty,\gamma=\frac{1}{20})} = \begin{pmatrix} 0.00 & 0.14 & 0.40 & 0.71 & 0.03 & 0.02 \\ 0.14 & 0.00 & 0.07 & 0.27 & 0.46 & 0.65 \\ 0.40 & 0.07 & 0.00 & 0.72 & 0.09 & 0.02 \\ 0.71 & 0.27 & 0.72 & 0.00 & 0.17 & 0.09 \\ 0.03 & 0.46 & 0.09 & 0.17 & 0.00 & 0.74 \\ 0.02 & 0.65 & 0.02 & 0.09 & 0.74 & 0.00 \end{pmatrix}$$

In diesem Fall erhalten wir $\alpha^{(2)} = (0.88, -0.62, 0.88, 0.70, -0.82, -0.96)$ und wieder $\hat{\alpha}^{OGC,\approx}$ wie in Gl. (10.31). Man sieht, dass die Verwendung von stetigen Gewichten (ohne vorheriges Reduzieren des Graphen) in diesem Fall zum gleichen Ergebnis führt. Hier unterscheiden sich die einzelnen Werte von $\alpha^{(2)}$ auch deutlicher voneinander, was zu stabileren Graph Cuts führt.

10.2.2 Anwendung und Interpretation: Spektrales Clustern

Das zuvor besprochene Finden eines optimalen Graph Cuts erlaubt es, Trainingsdaten in $K^* = 2$ Mengen („Cluster“) zu zerlegen. Eine Verallgemeinerung dieser Technik auf mehr als zwei Cluster erscheint allerdings schwierig, insbesondere in Hinsicht auf die Verwendung der Indikatorvektoren $\alpha \in \{-1, +1\}^{\#V}$. Bisher fehlt auch noch eine mathematische Beschreibung, neue Datenpunkte $x \in \mathbb{R}^d$ einem der Cluster zuzuordnen.

Für eine Verallgemeinerung beginnen wir zunächst mit einem neuen Ansatz und erkennen dann dessen Verbindung zur bisher besprochen Theorie. Das ursprüngliche Ziel war die Projektion der hochdimensionalen Trainingsdaten X_i auf einen niedrigdimensionalen Raum, wobei die Abstände zwischen X_i möglichst auch im niedrigdimensionalen Raum beibehalten werden sollen.

Setzen wir für $\gamma > 0$

$$\tilde{W}_{ij} := \exp\big(-\gamma \cdot D(X_i, X_j)\big), \tag{10.32}$$

so erhalten wir mit $\tilde{W}_{ij}$ ein Maß für die Gleichheit zwischen zwei X_i, X_j: $\tilde{W}_{ij}$ ist klein, falls X_i, X_j sehr verschieden bezüglich der Abstandsfunktion D sind.

Bei einer 1-dimensionalen Projektion ist unser Ziel, einen Vektor $\hat{\alpha} = (\hat{\alpha}_1, ..., \hat{\alpha}_n) \in \mathbb{R}^n$ zu finden, sodass dessen i-te Komponente $\hat{\alpha}_i \in \mathbb{R}$ das Trainingsdatum X_i im eindimensionalen Raum repräsentiert. Ein sinnvoller Ansatz für die Wahl von $\hat{\alpha}$ ist:

$$\hat{\alpha} \in \operatorname*{arg\,min}_{\alpha \in \mathbb{R}^n} J(\alpha), \qquad J(\alpha) := \frac{1}{8} \sum_{i,j=1}^{n} \tilde{W}_{ij} (\alpha_i - \alpha_j)^2, \tag{10.33}$$

denn dann sind $\hat{\alpha}_i, \hat{\alpha}_j$ nahe beinander genau dann, wenn $\tilde{W}_{ij}$ groß ist, d. h. wenn X_i, X_j einen geringen Abstand zueinander haben. Der Faktor $\frac{1}{8}$ ändert die Lösung nicht und ist in Hinsicht auf die gleich folgenden Resultate gewählt. Das obige Optimierungsproblem muss allerdings wieder zwei Nebenbedingungen unterworfen werden:

- Die triviale Lösung $\hat{\alpha} = \mathbb{1}$ muss ausgeschlossen werden.
- Wir benötigen eine Normierung für $\hat{\alpha}$. Wäre nämlich $\hat{\alpha}$ Minimierer, so würde $\frac{1}{2}\hat{\alpha}$ noch einen kleineren Wert für J liefern; daher besitzt J im Moment nur den trivialen Minimierer $\hat{\alpha} = 0$.

Eine sinnvolle Wahl der Nebenbedingungen ist hier nicht offensichtlich. Es gilt jedoch das folgende zentrale Resultat, welches eine Verbindung des Ansatzes aus Gl. (10.33) mit dem Finden eines optimalen Graph Cuts erlaubt.

Lemma 10.43 Es sei $\tilde{\mathbb{D}} \in \mathbb{R}^{n\times n}$ eine Diagonalmatrix mit $\tilde{\mathbb{D}}_{ii} := \sum_{j=1}^{n} \tilde{W}_{ij}$, $i = 1, ..., n$ und $\tilde{L} = \tilde{\mathbb{D}} - \tilde{W}$. Dann gilt für alle $\alpha \in \mathbb{R}^n$:

$$J(\alpha) = \frac{1}{4}\langle \alpha, \tilde{L}\alpha \rangle$$

Beweis Es ist

$$\begin{aligned} J(\alpha) &= \frac{1}{8}\Big(\underbrace{\underbrace{\sum_{i=1}^{n}\underbrace{\sum_{j=1}^{n} \tilde{W}_{ij}}_{=\tilde{\mathbb{D}}_{ii}} \alpha_i^2}_{=\langle \alpha, \tilde{\mathbb{D}}\alpha\rangle} - 2 \underbrace{\sum_{i,j=1}^{n} \tilde{W}_{ij}\alpha_i\alpha_j}_{=\langle \alpha, \tilde{W}\alpha\rangle} + \underbrace{\sum_{j=1}^{n}\underbrace{\sum_{i=1}^{n} \tilde{W}_{ij}}_{=\tilde{\mathbb{D}}_{jj}} \alpha_j^2}_{=\langle \alpha, \tilde{\mathbb{D}}\alpha\rangle}}_{} \Big) \\ &= \frac{1}{4}\langle \alpha, (\tilde{\mathbb{D}} - \tilde{W})\alpha \rangle = \frac{1}{4}\langle \alpha, \tilde{L}\alpha \rangle. \end{aligned}$$

Damit erhalten wir die Äquivalenz des Problems vom approximativen Graph Cut und der Bestimmung der optimalen eindimensionalen Projektion:

Bemerkung 10.44 Die Funktion $J(\alpha)$ entspricht *fast genau* der Optimierungsfunktion beim Finden eines approximativ optimalen Graph Cuts mit einem Nachbarschaftsgraphen mit Gewichtsmatrix $W = \tilde{W}$ und $\varepsilon = \infty$, vgl. Definition 10.40. Der einzige Unterschied entsteht durch die Tatsache, dass in Gl. (10.33) $\tilde{W}_{ii} \neq 0$ ist, aber $W_{ii} = 0$ gilt, d. h. die Diagonalelemente der Gewichtsmatrizen stimmen nicht überein. Offensichtlich hat dies keinen Einfluss auf die Laplace-Matrizen (es gilt $L = \tilde{L}$), sondern nur auf die Gradmatrizen ($\mathbb{D} \neq \tilde{\mathbb{D}}$). Da für große Anzahlen von Trainingsdaten der Einfluss von W_{ii} auf die Komponenten von $\mathbb{D}$ gering ist, werden wir diesen Unterschied im Folgenden ignorieren.

Für die Bestimmung des optimalen Graph Cuts kennen wir sinnvolle Nebenbedingungen; diese sind (nun übertragen auf das obige Problem) gegeben durch Gl. (10.29):

$$\langle \tilde{\mathbb{D}}\mathbb{1}, \alpha \rangle = 0, \quad \text{und} \quad \langle \tilde{\mathbb{D}}\alpha, \alpha \rangle = n \tag{10.34}$$

Andererseits erhalten wir so auch eine stärkere anschauliche Interpretation der Lösung eines approximativen Graph Cuts: Sie liefert den Vektor $\hat{\alpha} \in \mathbb{R}^n$, welcher die Trainingsdaten bezüglich der Abstandsfunktion D und der Gewichtsmatrix W am besten in den eindimensionalen Raum projiziert. Unter Nutzung der Nebenbedingungen aus Gl. (10.34) erhalten wir:

Definition 10.45 (Eindim. abstandserhaltende Projektion der Trainingsdaten)
Es seien $\tilde{W}$, $\tilde{\mathbb{D}}$ und $\tilde{L}$ wie in Lemma 10.43. Sei $\hat{\alpha} = (\hat{\alpha}_1, ..., \hat{\alpha}_n)^T \in \mathbb{R}^n$ Lösung von

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{4} \langle \alpha, \tilde{L}\alpha \rangle \qquad \text{unter NB} \qquad \langle \tilde{\mathbb{D}}\mathbb{1}, \alpha \rangle = 0, \quad \text{und} \quad \langle \tilde{\mathbb{D}}\alpha, \alpha \rangle = n.$$

Dann heißt $\hat{\alpha}$ *optimale eindimensionale abstandserhaltende Projektion* von $X_1, ..., X_n$. Eine Aufteilung in zwei Cluster ist gegeben durch

$$\hat{C}(i) := \begin{cases} 1, & \hat{\alpha}_i < 0, \\ 2, & \hat{\alpha}_i > 0. \end{cases} \tag{10.35}$$

♦

Bemerkung 10.46

- Die Berechnung erfolgt wie in Satz 10.41 durch Lösung eines generalisierten Eigenwertproblems.
- Die Wahl der Zuweisungsfunktion $\hat{C}(i)$ ist motiviert durch die Interpretation von $\hat{\alpha}$ als approximativ optimaler Graph Cut.
- Falls man die Daten in mehr als zwei Cluster aufteilen möchte, so kann man nun auf $\hat{\alpha}_1, ..., \hat{\alpha}_n \in \mathbb{R}$ auch eine der vorher besprochenen Clustering-Methoden nutzen, beispielsweise k-means-Clustering. Dies ist möglich durch die Interpretation von $\hat{\alpha}$ als Projektion der Daten in die niedrigere Dimension 1.
- Allgemeines Verhalten von $\hat{\alpha}$: Besonders praxisrelevant ist das Trennungsverhalten von α für $\gamma \gg 1$: In diesem Fall sind die Einträge von $\tilde{W}_{ij}$ aus Gl. (10.32). In diesem Fall ist $\tilde{W}_{ij}$ nur wesentlich verschieden von null, wenn $D(X_i, X_j)$ sehr klein ist; ansonsten ist stets $\tilde{W}_{ij} \approx 0$. Es wird also eine imaginäre Situation erschaffen, in der nur Trainingsdaten X_i, X_j mit kleinem $D(X_i, X_j)$ noch nah beieinander liegen; alle anderen Paare haben „unendlich großen“ Abstand. In Hinsicht auf das ursprüngliche Minimierungsproblem Gl. (10.33) legt $\hat{\alpha}$ also das Hauptaugenmerk darauf, dass die Daten so in zwei Mengen getrennt werden, dass sehr nah beieinander liegende Trainingsdaten in der gleichen Menge liegen. $\hat{\alpha}$ sucht in den Trainingsdaten also nach zwei Punktmengen, die deutlich voneinander separiert, aber untereinander eng verbunden sind. Mit anderen Worten, $\hat{\alpha}$ findet die zwei größten zusammenhängenden Punktmengen in den Trainingsdaten und weist der einen Punktmenge negative, der anderen Punktmenge positive Werte zu.

Wir untersuchen nun das Verhalten von $\hat{\alpha}$ anhand eines Beispiels.

Beispiel 10.47 Wir betrachten $n = 400$ Trainingsdaten aus Beispiel 10.23 mit $M^* = 2$. In Abb. 10.9a sind die Trainingsdaten dargestellt, wobei die verschiedenen Farben die Zugehörigkeit zu den verschiedenen Kreisringen ($Z_i \in \{1, 2\}$) symbolisieren. Diese dienen hier nur zum besseren Verständnis der Resultate und sind während der Berechnungen unbekannt. In Abb. 10.9b, c, d sind die Vektoren $\hat{\alpha}$ aus Definition 10.45 für $\gamma \in \{0.1, 10, 100\}$ unter Verwendung des euklidischen Abstands $D(x, x') = \|x - x'\|_2^2$ dargestellt. Wie man sieht, ist die durch Gl. (10.35) gegebene Zuordnung erst für $\gamma \in \{10, 100\}$ korrekt. Für

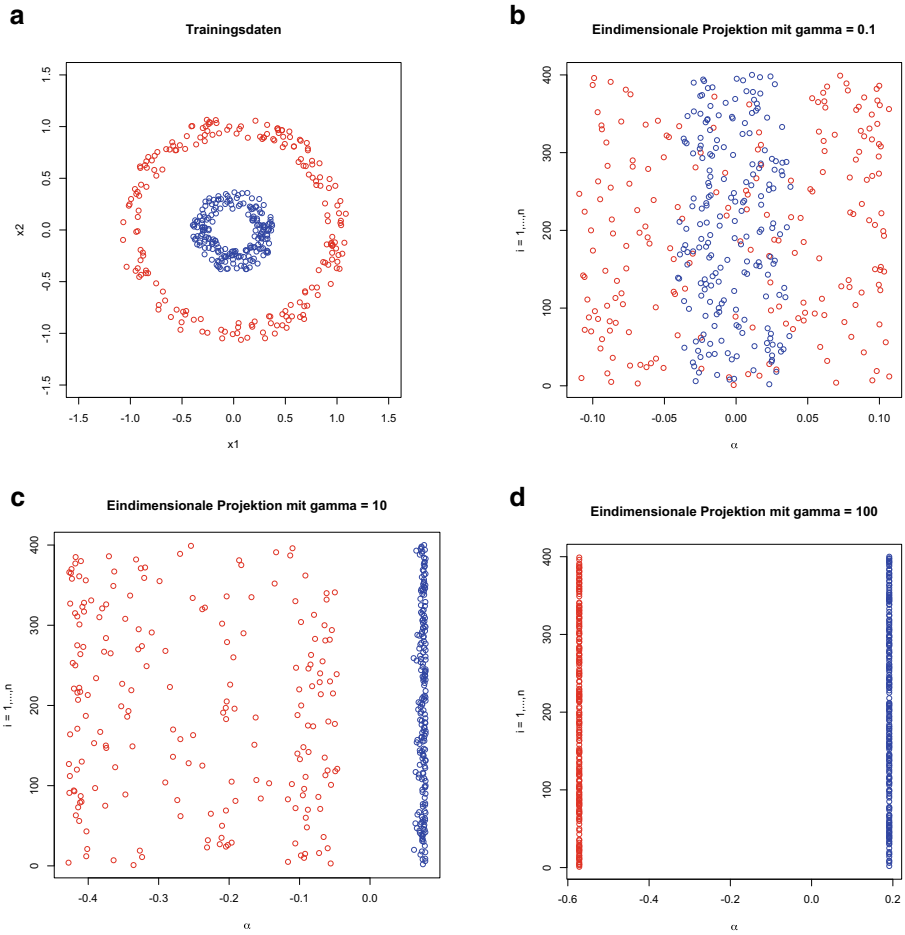


Abb. 10.9 Darstellung der optimalen eindimensionalen abstandserhaltenden Projektionen erhalten aus $n = 400$ Trainingsdaten aus Beispiel 10.23 mit $M^* = 2$. **a** Trainingsdaten, **b, c, d** Darstellung von $\hat{\alpha}$ aus Definition 10.45 für die verschiedenen Beobachtungen $i \in \{1, ..., n\}$ (y-Achse) mit verschiedenen $\gamma \in \{0.1, 10, 100\}$

größeres γ werden die zu verschiedenen Kreisringen gehörigen Trainingsdaten immer deutlicher separiert, wie in der Bemerkung nach Definition 10.45 beschrieben.

In praktischen Beispielen muss die Wahl von γ jedoch mit mehr Bedacht erfolgen. Im Allgemeinen sind die Trainingsdaten verrauschter und klare Aufteilungen in zwei Punktmengen gehen aus diesen nicht exakt hervor. Als Beispiel betrachten wir die leicht abgeänderten Trainingsdaten in Abb. 10.10a. In Abb. 10.10b, c ist die eindimensionale Projektion $\hat{\alpha}$ unter Verwendung von $\gamma = 1000$ und die entsprechende Zuordnung gemäß (10.35) farblich dargestellt; In Abb. 10.10d, e dasselbe für $\gamma = 10$. Ein zu groß gewähltes γ führt hier zu einer nicht korrekten Erkennung der Trainingsdaten aus den beiden verschiedenen Kreisringen, da es durch die Kontamination eine günstigere Aufteilung in zwei leichter trennbare Punktmengen gibt. Mit $\gamma = 10$ werden auch „mittelgroße“ Abstände zwischen den Trainingsdaten bei der Berechnung von $\hat{\alpha}$ berücksichtigt, was dazu führt, dass $\hat{\alpha}$ eine Trennung entlang der „Verbindungsbrücke“ als günstiger erachtet, um die Abstände möglichst adäquat in einer Dimension wiederzugeben.

Im Allgemeinen kann auch nicht erwartet werden, dass auch mehr als zwei zusammenhängende Punktmengen (vgl. zum Beispiel Abb. 10.12a) mit Hilfe von $\hat{\alpha}$ aus Definition 10.45 und genügend großem γ getrennt werden können, da dies der ursprünglichen Motivation von $\hat{\alpha}$ als optimaler Graph Cut zuwiderläuft. Für komplexere und/oder hochdimensionale Strukturen in den Trainingsdaten ist es im Allgemeinen nicht möglich, mit nur einer Dimension die Abstände der Daten untereinander adäquat wiedergeben zu können.

Es besteht allerdings die Hoffnung, dass eine Nachbildung der Abstände der Trainingsdaten X_i durch Punkte in einem höherdimensionalen Raum $\mathbb{R}^k$, $1 < k \leq d$ eine bessere Vorverarbeitung der Daten durch Algorithmen aus Kap. 9 ermöglicht, so dass beispielsweise auch mehrere zusammenhängende Punktmengen separiert werden können. Hierbei scheint es zunächst nicht plausibel, auch $k = d$ bzw. $k > d$ zuzulassen. Wie wir aber bereits nach Definition 10.45 bemerkt haben, sucht $\hat{\alpha}$ nicht nur nach einer direkten Nachbildung der Abstände zwischen den X_i im Raum $\mathbb{R}$, sondern manipuliert diese vorher entsprechend der Wahl von γ. Diese Manipulation kann genauso im Falle $k = d$ eine völlig neue Perspektive auf die Trainingsdaten bieten.

Im Folgenden leiten wir die Theorie für Projektionen in einen k-dimensionalen Raum ($1 \leq k \leq d$) her.

Ansatz Wir wollen $X_1, ..., X_n \in \mathbb{R}^d$ durch $\hat{\alpha}_1, ..., \hat{\alpha}_n \in \mathbb{R}^k$ repräsentieren. Dazu schreiben wir

$$\hat{\alpha} := (\hat{\alpha}_1, ..., \hat{\alpha}_n) = \begin{pmatrix} \hat{\alpha}_{\cdot 1}^T \\ \vdots \\ \hat{\alpha}_{\cdot k}^T \end{pmatrix} \in \mathbb{R}^{k \times n}.$$

Wir verwenden denselben Ansatz wie in Gl. (10.33), d. h. wir minimieren

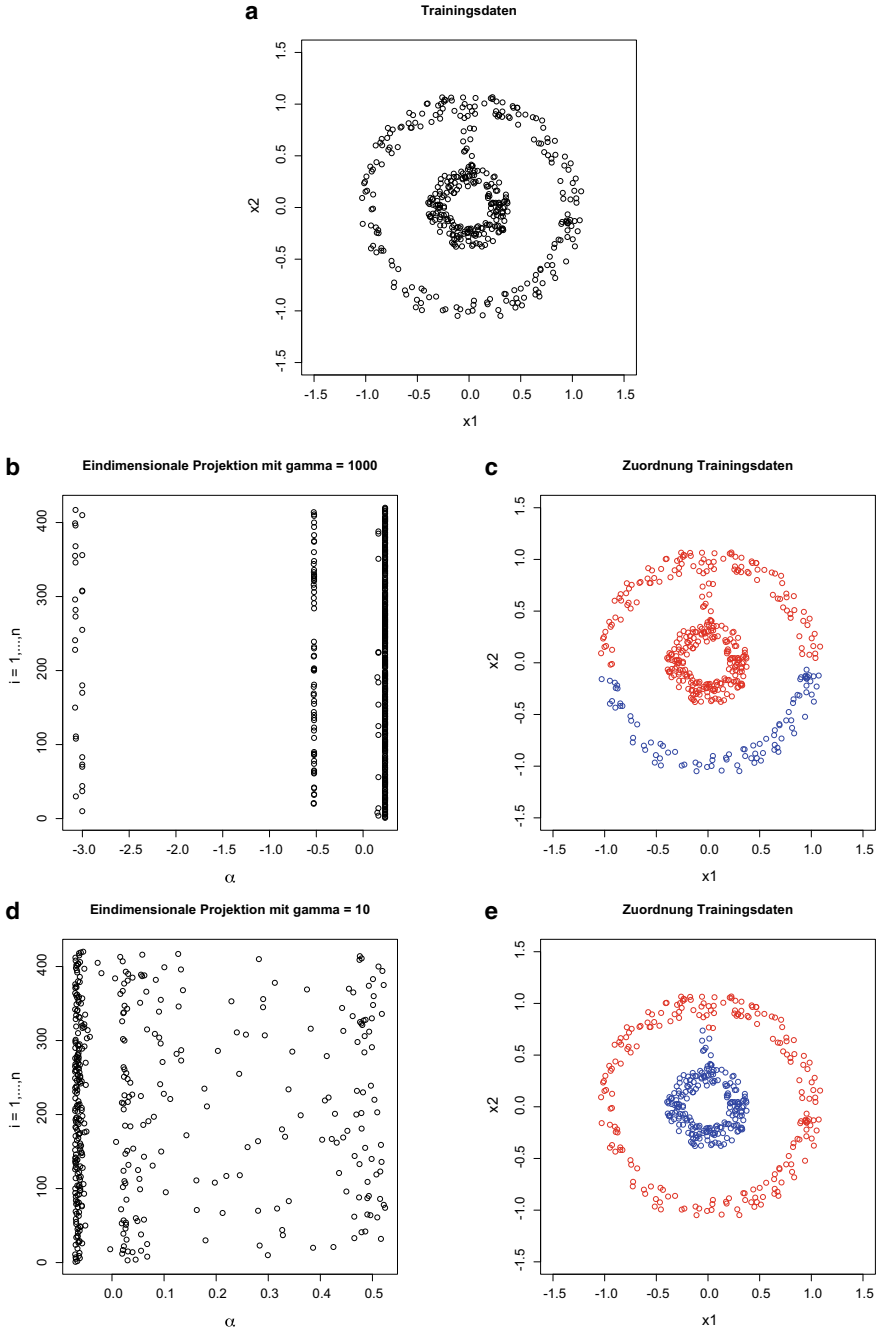


Abb. 10.10 Darstellung der optimalen eindimensionalen abstandserhaltenden Projektionen erhalten aus leicht abgeänderten Beobachtungen aus Beispiel 10.23 mit $M^* = 2$. **a** Trainingsdaten, **b, c** $\hat{\alpha}$ aus Definition 10.45 mit $\gamma = 1000$ und die farbliche Zuordnung der Trainingsdaten gemäß $\hat{C}(i)$, **d, e** analoge Darstellung mit $\gamma = 10$

$$J(\alpha) := \sum_{i,l=1}^{n} \tilde{W}_{il} \|\alpha_i - \alpha_l\|_2^2 = \sum_{i,l=1}^{n} \tilde{W}_{il} \sum_{j=1}^{k} (\alpha_{ij} - \alpha_{lj})^2$$
$$= \sum_{j=1}^{k} \sum_{i,l=1}^{n} \tilde{W}_{il} (\alpha_{ij} - \alpha_{lj})^2 = 2 \sum_{j=1}^{k} \alpha_{\cdot j}^T \tilde{L} \alpha_{\cdot j} = 2\mathrm{Sp}(\alpha \tilde{L} \alpha^T).$$

Wie im eindimensionalen Fall führen wir Nebenbedingungen zum Vermeiden der trivialen Lösung und zur Einführung einer Normierung der Lösung α ein und erhalten:

Definition 10.48 (Optimale k-dimensionale Projektion)
Es seien $\tilde{W}$, $\tilde{\mathbb{D}}$ und $\tilde{L}$ wie in Lemma 10.43. Sei $1 \leq k \leq d$. Sei $\hat{\alpha}^{SC,k} \in \mathbb{R}^{k \times n}$ Lösung von

$$\min_{\alpha \in \mathbb{R}^{p \times n}} J(\alpha) \qquad \text{unter NB} \qquad \forall j = 1, ..., k: \ \langle \tilde{\mathbb{D}} \mathbb{1}, \alpha_{\cdot j} \rangle = 0, \quad \text{und}$$
$$\langle \tilde{\mathbb{D}} \alpha_{\cdot j}, \alpha_{\cdot j} \rangle = n. \tag{10.36}$$

Dann heißen $\hat{\alpha}_1^{SC,k}, ..., \hat{\alpha}_n^{SC,k}$ gegeben durch die Spalten von $\hat{\alpha}^{SC,k}$ *optimale k-dimensionale Projektionen* von $X_1, ..., X_n$. ♦

Analog zum Resultat von Satz 10.41 ist auch hier eine explizite Berechnung mittels der Lösung eines generalisierten Eigenwertproblems möglich.

Satz 10.49 Es seien $\tilde{W}$, $\tilde{\mathbb{D}}$ und $\tilde{L}$ wie in Lemma 10.43. Sei $1 \leq k \leq d$. Seien $\hat{\rho}_j \in \mathbb{R}$, $\hat{\beta}^{(j)} \in \mathbb{R}^n$, $j = 1, ..., n$ Lösungen der Gleichung

$$\tilde{L}\beta = \rho \tilde{\mathbb{D}} \beta \tag{10.37}$$

mit $0 \leq \hat{\rho}_1 \leq \hat{\rho}_2 \leq ... \leq \hat{\rho}_n$ und $(\hat{\beta}^{(j)})^T \tilde{\mathbb{D}} \hat{\beta}^{(j)} = n$. Dann ist

$$(\hat{\alpha}_1, ..., \hat{\alpha}_n) = \hat{\alpha} = \begin{pmatrix} (\hat{\beta}^{(2)})^T \\ \vdots \\ (\hat{\beta}^{(k+1)})^T \end{pmatrix},$$

eine Lösung des Optimierungsproblems (10.36).

Zusammengefasst erhalten wir damit den Algorithmus des „spektralen Clusterns“ zur Reduktion von gegebenen n-dimensionalen Trainingsdaten. Der Name begründet sich wie folgt: Einerseits liefern die Projektionen mit ihren Vorzeichen bereits eine Voreinteilung der Daten in Cluster (vgl. Definition 10.45) und werden bevorzugt als Vorbearbeitung der Daten zur Anwendung von Clustering-Algorithmen wie aus Kap. 9 eingesetzt; andererseits muss zur Ermittlung der Projektionen ein Eigenwertproblem gelöst werden (die Menge der Eigenwerte einer Matrix wird auch als Spektrum bezeichnet).

In der Praxis wird $\tilde{W}$ anstatt wie in Gl. 10.32 mit Hilfe eines Kerns definiert, d. h.

$$\tilde{W}_{ij} := K(X_i, X_j)$$

für einen beliebigen Mercer-Kern K (vgl. Definition 4.32). Da für die Herleitung des optimalen Graph Cuts und der optimalen k-dimensionalen Projektionen nur die Interpretation von W_{ij} bzw. $\tilde{W}_{ij}$ als Maß für die „Gleichheit" zweier Beobachtungen eine Rolle gespielt hat, bleiben alle Aussagen sinngemäß gültig. Im Falle von $D(x, x') = \|x - x'\|_2^2$ erhält man die bisherige Theorie durch den Gauß-Kern $K(x, x') = \exp(-\gamma \|x - x'\|_2^2)$.

Definition 10.50 (Algorithmus: Spektrales Clustern)
Gegeben seien i.i.d. Trainingsdaten $X_1, ..., X_n \in \mathbb{R}^d$. Weiter sei K ein Mercer-Kern und $k \in \mathbb{N}$. Definiere $\tilde{W} := (\tilde{W}_{ij})_{i,j=1,...,n} \in \mathbb{R}^{n \times n}$ durch

$$\tilde{W}_{ij} := K(X_i, X_j)$$

und die Diagonalmatrix $\tilde{\mathbb{D}} \in \mathbb{R}^{n \times n}$ durch $\tilde{\mathbb{D}}_{ii} := \sum_{j=1}^{n} K(X_i, X_j)$. Setze $\tilde{L} := \tilde{\mathbb{D}} - \tilde{W}$. Seien $0 \leq \hat{\rho}_1^K \leq \hat{\rho}_2^K \leq ... \leq \hat{\rho}_n^K$ die Eigenwerte und $\hat{b}^{(j)}$, $j = 1, ..., n$ die zugehörigen Eigenvektoren der Matrix $\tilde{\mathbb{D}}^{-1/2} \tilde{L} \tilde{\mathbb{D}}^{-1/2}$ mit $\|\hat{b}^{(j)}\|_2^2 = 1$. Setze $\hat{\beta}^{(j)} := n^{1/2} \mathbb{D}^{-1/2} \hat{b}^{(j)}$ und

$$(\hat{\alpha}_1^{SC,K,k}, ..., \hat{\alpha}_n^{SC,K,k}) := \begin{pmatrix} (\hat{\beta}^{(2)})^T \\ \vdots \\ (\hat{\beta}^{(k+1)})^T \end{pmatrix}.$$

Dann heißen $\hat{\alpha}_i^{SC,K,k}$, $i = 1, ..., n$ *optimale k-dimensionale Projektionen* von X_i, $i = 1, ..., n$ *erhalten durch spektrales Clustern.* ♦

Wir zeigen zunächst das Verhalten für die Trainingsdaten aus Beispiel 10.47:

Beispiel 10.51 (Fortführung Beispiel 10.47) Wir betrachten $n = 400$ Trainingsdaten aus Beispiel 10.23 mit $M^* = 2$. In Abb. 10.11 sind die Ergebnisse $\hat{\alpha}_i^{SC,K,2}$ des spektralen Clusterns (Definition 10.50) mit dem Gauß-Kern $K = K_\gamma^{gauss}$ und $\gamma \in \{0.1, 7.5, 10, 100\}$ dargestellt. Wie zuvor deuten die verschiedenen Farben die Zugehörigkeit zu den verschiedenen Kreisringen ($Z_i \in \{1, 2\}$) an und sind während der Durchführung des Algorithmus nicht bekannt. Während die auf eine Dimension reduzierten Trainingsdaten $\hat{\alpha}_i^{SC,K,1}$ nur für die Fälle $\gamma \in \{10, 100\}$ eine Trennung der beiden Kreisringe erlauben (vgl. Beispiel 10.23), ist dies mit den auf zwei Dimensionen reduzierten Trainingsdaten $\hat{\alpha}_i^{SC,K,2}$ zusätzlich auch für $\gamma = 7.5$ noch möglich, indem auf $\hat{\alpha}_i^{SC,K,2}$ beispielsweise ein k-means-Clustering oder ein EM-Algorithmus angewandt wird. Wir sehen also:

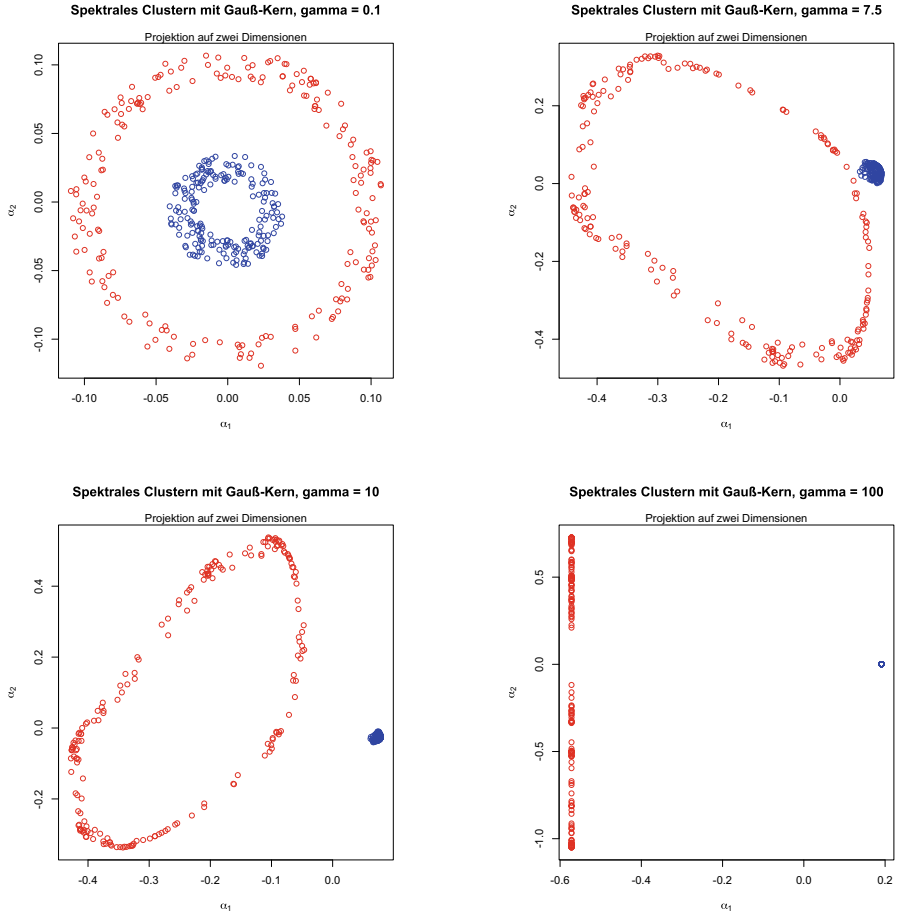


Abb. 10.11 Darstellung von $\hat{\alpha}_i^{SC,K,2}$ aus Definition 10.50 angewandt auf $n = 400$ Trainingsdaten aus Beispiel 10.23 mit $M^* = 2$ für verschiedene $\gamma \in \{0.1, 7.5, 10, 100\}$

- $\hat{\alpha}_i^{SC,K,2}$ kann wesentlich verschieden von X_i sein (die Erklärung haben wir nach Beispiel 10.47 geliefert) und insbesondere eine bessere Grundlage für Algorithmen aus Kap. 9 bieten.
- Eine Projektion auf mehr als eine Komponente (auch wenn dies nicht unbedingt nötig ist) kann die Abhängigkeit der Verwendbarkeit der Resultate von der Wahl von γ verringern.

Zur Illustration der Leistungsfähigkeit des spektralen Clusterns und zur Unterscheidung vom Verhalten der Kern-basierten PCA betrachten wir die Trainingsdaten aus Beispiel 10.23 mit $M^* = 3$:

Beispiel 10.52 Wir betrachten $n = 600$ Trainingsdaten aus Beispiel 10.23 mit $M^* = 3$. In Abb. 10.12a sind diese dargestellt (erneut mit verschiedenen Farben für die Zugehörigkeit zu verschiedenen Kreisringen). In Abb. 10.12b, c, d ist $\hat{\alpha}_i^{PCA,K,2}$ des spektralen Clusterns (Definition 10.50) mit dem Gauß-Kern $K = K_\gamma^{gauss}$ und $\gamma \in \{20, 30, 100\}$ dargestellt. Im Gegensatz zur Kern-basierten PCA (vgl. Abb. 10.7) liefert $\hat{\alpha}_i^{PCA,K,2}$ ab $\gamma = 30$ eine zweidimensionale Projektion, mit der die Punkte zu verschiedenen Kreisringen mit Hilfe von k-means-Clustering getrennt werden können. Wird der Algorithmus mit $\gamma = 100$ angewandt, ist sogar eine Trennung mit k-means-Clustering angewandt auf die *eindimensionale* Projektion $\hat{\alpha}_i^{PCA,K,1}$ möglich.

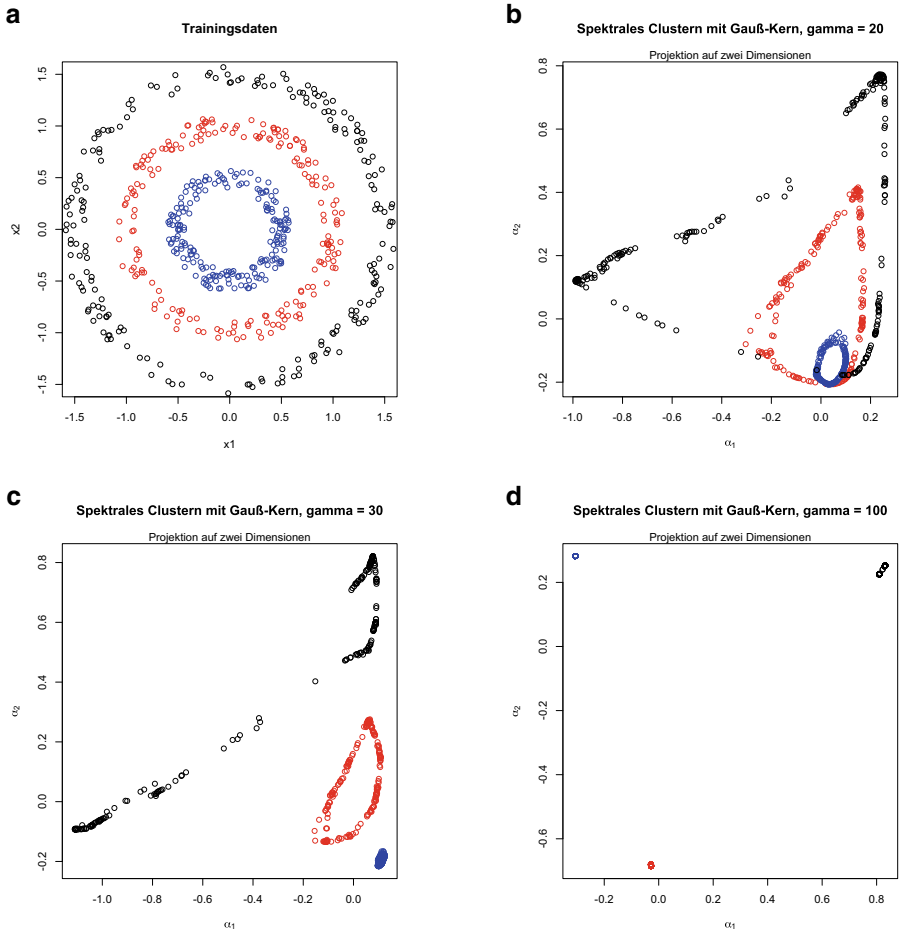


Abb. 10.12 Anwendung des spektralen Clusterns auf $n = 600$ Trainingsdaten aus Beispiel 10.23 mit $M^* = 3$. **a** Trainingsdaten, **b, c, d** Darstellung von $\hat{\alpha}_i^{PCA,K,2}$ mit dem Gauß-Kern K und $\gamma \in \{20, 30, 100\}$

Auch wenn die Verwendung des spektralen Clusterns von $\gamma = 100$ bei diesem einfachen Beispiel funktioniert, so würde dies in der Praxis bei einer ähnlichen Situation mit stärker verrauschten Daten im Allgemeinen nicht zu sinnvollen Ergebnissen führen (vgl. Beispiel 10.47). In diesem Fall sollte dann $\gamma < 100$ in Kombination mit höherdimensionalen Projektionen $\hat{\alpha}_i^{PCA,K,2}$ genutzt werden, um eine korrekte Zuordnung durch k-means-Clustering zu erhalten.

Die Daten $\hat{\alpha}_i^{SC,K,k}$ entstehen anschaulich aus den X_i durch eine „Projektion" $\hat{A}_n^{SC,K,k} : \mathbb{R}^d \to \mathbb{R}^k$ mittels $\hat{\alpha}_i^{SC,K,k} = \hat{A}_n^{SC,K,k}(X_i)$, $i = 1, ..., n$. Aufgrund der bisherigen Herleitung ist jedoch unklar, wie diese Abbildung $\hat{A}_n^{SC,K,k}$ aussieht. Daher fehlt uns bisher die mathematische Grundlage, um *neue* Beobachtungen $x \in \mathbb{R}^d$ auf genau die gleiche Weise wie $X_i, i = 1, ..., n$ in den Raum $\mathbb{R}^k$ abzubilden. Aus statistischer Sicht ist außerdem unklar, welche nichtzufällige Größe mit $\hat{\alpha}_i^{SC,K,k}$ geschätzt wird und wie entsprechend die Güte von $\hat{\alpha}_i^{SC,K,k}$ zu bewerten ist.

Indem wir den Algorithmus des spektralen Clusterns in Beziehung zur Kern-basierten Hauptkomponentenanalyse setzen, können die obigen beiden Fragestellungen gelöst werden. Insbesondere erhalten wir durch Definition 10.55 eine geeignete Abbildung $\hat{A}_n^{SC,K,k}$.

10.2.3 Verbindung zum Kern-basierten PCA-Algorithmus

Im Folgenden formulieren den Algorithmus des spektralen Clusterns aus Definition 10.50 so um, dass er dieselbe Struktur wie der Kern-basierte PCA-Algorithmus aus Definition 10.20 besitzt. Insbesondere wollen wir erreichen, dass die Einträge der Matrix, von welcher Eigenwerte und Eigenvektoren bestimmt werden, durch einen Kern gegeben sind. Wir stellen zunächst fest, dass

$$\tilde{\mathbb{D}}^{-1/2}\tilde{L}\tilde{\mathbb{D}}^{-1/2} = I_{n\times n} - \tilde{\mathbb{D}}^{-1/2}\tilde{W}\tilde{\mathbb{D}}^{-1/2}.$$

Damit gilt für $\rho \in \mathbb{R}$, $b \in \mathbb{R}^n$:

$$\left[\tilde{\mathbb{D}}^{-1/2}\tilde{L}\tilde{\mathbb{D}}^{-1/2}\right]b = \rho b \quad \Longleftrightarrow \quad \left[\tilde{\mathbb{D}}^{-1/2}\tilde{W}\tilde{\mathbb{D}}^{-1/2}\right]b = (1-\rho)b$$

Mit anderen Worten, $\tilde{\mathbb{D}}^{-1/2}\tilde{L}\tilde{\mathbb{D}}^{-1/2}$ hat dieselben Eigenvektoren wie $\tilde{\mathbb{D}}^{-1/2}\tilde{W}\tilde{\mathbb{D}}^{-1/2}$, aber genau in der umgekehrten Reihenfolge.

Wir wissen bereits, dass $\tilde{\mathbb{D}}^{-1/2}\tilde{L}\tilde{\mathbb{D}}^{-1/2}$ den Eigenwert $\rho = 0$ mit Eigenvektor $b = n^{-1/2}\mathbb{1}$ besitzt (vgl. Bemerkung nach Satz 10.41). Entsprechend hat $\tilde{\mathbb{D}}^{-1/2}\tilde{W}\tilde{\mathbb{D}}^{-1/2}$ den größten Eigenwert $\lambda = 1$ zum (auf Länge 1 normierten) Eigenvektor $v := \frac{1}{\|\tilde{\mathbb{D}}^{1/2}\mathbb{1}\|_2}\tilde{\mathbb{D}}^{1/2}\mathbb{1}$. Setzen wir

$$\hat{W} := n \cdot (\tilde{\mathbb{D}}^{-1/2}\tilde{W}\tilde{\mathbb{D}}^{-1/2} - 1 \cdot vv^T),$$

so besitzt $\frac{1}{n}\hat{W}$ genau dieselben Eigenwerte und Eigenvektoren wie die Matrix $\tilde{\mathbb{D}}^{-1/2}\tilde{W}\tilde{\mathbb{D}}^{-1/2}$, aber v ist nun ein Eigenvektor zum Eigenwert 0. Wegen $(\tilde{\mathbb{D}}^{1/2}\mathbb{1})_i =$

$[\tilde{\mathbb{D}}^{1/2}]_{ii} = \tilde{\mathbb{D}}_{ii}^{1/2} = \left(\sum_{j=1}^n K(X_i, X_j)\right)^{1/2}$ gilt $\|\tilde{\mathbb{D}}^{1/2}\mathbb{1}\|_2^2 = \sum_{i,j=1}^n K(X_i, X_j)$ und damit

$$\begin{aligned}\hat{W}_{il} &= n \cdot \left[(\mathbb{D}^{-1/2})_{ii}\tilde{W}_{il}(\mathbb{D}^{-1/2})_{ll} - \frac{1}{\|\tilde{\mathbb{D}}^{1/2}\|_2^2}(\tilde{\mathbb{D}}^{1/2}\mathbb{1})_i(\tilde{\mathbb{D}}^{1/2}\mathbb{1})_l\right] \\ &= \hat{K}_{1,scal}(X_i, X_l), \qquad i, l \in \{1, ..., n\},\end{aligned}$$

wobei

$$\begin{aligned}\hat{K}_{1,scal}(x, y) &:= \frac{K(x, y)}{\left(\frac{1}{n}\sum_{i=1}^n K(x, X_i)\right)^{1/2} \cdot \left(\frac{1}{n}\sum_{i=1}^n K(X_i, y)\right)^{1/2}} \\ &\quad - \frac{\left(\frac{1}{n}\sum_{i=1}^n K(x, X_i)\right)^{1/2} \cdot \left(\frac{1}{n}\sum_{i=1}^n K(X_i, y)\right)^{1/2}}{\frac{1}{n^2}\sum_{i,l=1}^n K(X_i, X_l)} \\ &= \frac{K(x, y) - \frac{\left(\frac{1}{n}\sum_{i=1}^n K(x, X_i)\right)\cdot\left(\frac{1}{n}\sum_{i=1}^n K(X_i, y)\right)}{\frac{1}{n^2}\sum_{i,l=1}^n K(X_i, X_l)}}{\left(\frac{1}{n}\sum_{i=1}^n K(x, X_i)\right)^{1/2} \cdot \left(\frac{1}{n}\sum_{i=1}^n K(X_i, y)\right)^{1/2}}.\end{aligned} \tag{10.38}$$

Zusammengefasst erhalten wir:

Bemerkung 10.53 (Algorithmus: Spektrales Clustern, alternative Formulierung) Sei $1 \leq k \leq d$. Gegeben seien i.i.d. Trainingsdaten X_i, $i = 1, ..., n$ und ein Mercer-Kern $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$. Sei $\hat{K}_{1,scal}$ gegeben durch Gl. (10.38) die *normierte Kernfunktion.* Definiere $\hat{W} = (\hat{W}_{ij})_{i,j=1,...,n} \in \mathbb{R}^{n\times n}$ durch

$$\hat{W}_{ij} := \hat{K}_{1,scal}(X_i, X_j).$$

Seien $\hat{\lambda}_1^K \geq ... \geq \hat{\lambda}_k^K \geq 0$ die k größten Eigenwerte von $\frac{1}{n}\hat{W}$ und $\hat{b}^{(j)}$, $j = 1, ..., k$ die zugehörigen Eigenvektoren mit Länge 1. Dann erfüllt $\hat{\alpha}_{\cdot j}^K = (\hat{\alpha}_{1j}^K, ..., \hat{\alpha}_{nj}^K)^T$ mit $\hat{\alpha}_{ij}^K := \left(\frac{1}{n}\sum_{l=1}^n K(X_l, X_i)\right)^{-1/2}\hat{b}_i^{(j)}$ die Gleichung

$$(\hat{\alpha}_1^{SC,K,k}, ..., \hat{\alpha}_n^{SC,K,k}) = \begin{pmatrix}\hat{\alpha}_{\cdot 1}^T \\ \vdots \\ \hat{\alpha}_{\cdot k}^T\end{pmatrix}$$

mit den Größen aus dem Algorithmus des spektralen Clusterns (Definition 10.50).

Wir fassen kurz die Unterschiede der erhaltenen Größen von Kern-basierter PCA und spektralem Clustern zusammen.

Bemerkung 10.54 (Vergleich: Spektrales Clustern und Kern-basierte PCA) Durch die Zentrierung der Daten und durch die Forderung, dass die Projektionen $\hat{\alpha}_i^{PCA,K,k}$ die Varianz

der Daten in Richtung der Hauptkomponenten nachbilden sollen, stellt die Kern-basierte PCA implizit die Nebenbedingungen

$$\langle \mathbb{1}, \alpha_{\cdot j} \rangle = \sum_{i=1}^{n} \alpha_{ij} = 0, \qquad \frac{1}{n\lambda_j} \langle \alpha_{\cdot j}, \alpha_{\cdot j} \rangle = \frac{1}{n\lambda_j} \sum_{i=1}^{n} \alpha_{ij}^2 = 1$$

an die Projektionen auf die j-te Hauptkomponente. Beim spektralen Clustern hingegen lauten die Nebenbedingungen an die gefundenen Lösungen

$$\langle \tilde{\mathbb{D}}, \alpha_{\cdot j} \rangle = \sum_{i=1}^{n} \tilde{D}_{ii} \alpha_{ij} = 0, \qquad \frac{1}{n} \langle \tilde{\mathbb{D}} \alpha_{\cdot j}, \alpha_{\cdot j} \rangle = \frac{1}{n} \sum_{i=1}^{n} \tilde{\mathbb{D}}_{ii} \alpha_{ij}^2 = 1.$$

Im Falle, dass mit den Algorithmen zwei gegebene Punktmengen möglichst gut getrennt werden sollen, legt also die Kern-basierte PCA mehr Wert darauf, dass die Projektionen die Lage der Punkte korrekt abbilden, beim spektralen Clustern wird mehr Wert auf die Nachbildung der Abstände gelegt. Mathematisch äußert sich dies in der Verwendung eines anderen Kerns bei der Erstellung der Matrix $\hat{W}$ (vgl. Definition 10.20 bzw. Bemerkung 10.53). Selbst bei Verwendung des gleichen „Basiskerns“ K können sich daher $\hat{\alpha}_i^{PCA,K,k}$ und $\hat{\alpha}_i^{SC,K,k}$ stark voneinander unterscheiden.

Vom spektralen Clustern kann im Gegensatz zur PCA nicht erwartet werden, dass die Daten $\hat{\alpha}_i^{SC,K,k}$ eine gute Rekonstruktion der Lage der ursprünglichen Trainingsdaten X_i ermöglichen.

Durch die Darstellung des spektralen Clusterns in der Form der Kern-basierten PCA in Bemerkung 10.53 kann die Interpretation der Eigenwerte und $\hat{\alpha}_i^{SC,K,k}$ übernommen werden. Insbesondere gilt:

(1) Der Quotient

$$\frac{\sum_{j=1}^{k} \hat{\lambda}_j^K}{\sum_{j=1}^{n} \hat{\lambda}_j^K}$$

gibt ein Maß dafür an, wie gut die Abstände zwischen den Trainingsdaten mit den Projektionen $\hat{\alpha}_i^{SC,K,k}$ in $\mathbb{R}^k$ nachgebildet werden konnten.

(2) Der ursprünglich verwendete Ansatz aus Gl. (10.33) ist von komplexerer Natur als einfach nur eine Nachbildung der Abstände der Trainingsdaten $X_i \in \mathbb{R}^d$, $i = 1, ..., n$ in einem niedrigdimensionaleren Raum $\mathbb{R}^k$. Stattdessen werden die Abstände vor der Projektion gemäß des verwendeten Kerns K noch transformiert; dies geschieht durch Einbettung der Trainingsdaten in einen höherdimensionalen Raum $\mathbb{R}^{\tilde{d}}$ ($\tilde{d} > d$). Anschaulich ist $\hat{\alpha}_i^{SC,K,k}$ damit eine Projektion aus $\mathbb{R}^{\tilde{d}}$. Entsprechend kann beim spektralen Clustern auch eine Projektion auf $k > d$ Komponenten sinnvoll sein.

(3) Die Punkte $\hat{\alpha}_i^{SC,K,k}$ entstehen durch Anwendung einer Abbildung $\hat{A}_n^{SC,K,k} : \mathbb{R}^d \to \mathbb{R}^k$ der Form $\hat{\alpha}_i^{SC,K,k} = \hat{A}_n^{SC,K,k}(X_i)$, die einer Projektion aus einem höherdimensionalen Raum entspricht und mathematisch aus der Eigenwertgleichung für $\hat{W}$ gewonnen werden kann (vgl. Beweis von Lemma 10.19).

Unter Nutzung von (3) bestimmen wir nun die Abbildung $\hat{A}_n^{SC,K,k}$. Ist $j \in \{1, ..., k\}$, so folgt mit den Notationen aus Bemerkung 10.53 aus der Eigenwertgleichung $\frac{1}{n}\hat{W}\hat{b}^{(j)} = \hat{\lambda}_j^K \hat{b}^{(j)}$ für $i = 1, ..., n$:

$$\hat{b}_i^{(j)} = \hat{B}_n(X_i)_j, \qquad \hat{B}_n(x)_j := \frac{1}{n\hat{\lambda}_j^K} \sum_{l=1}^{n} \hat{K}_1(X_l, x)\hat{b}_l^{(j)}$$

bzw. für $\hat{\alpha}_{ij}^K$:

$$\hat{\alpha}_{ij}^K = \hat{A}_n^{SC,K}(X_i)_j, \qquad \hat{A}_n^{SC,K}(x)_j := \Big(\frac{1}{n}\sum_{l=1}^{n} K(X_l, x)\Big)^{-1} \cdot \frac{1}{n\hat{\lambda}_j^K} \sum_{l=1}^{n} \hat{K}_{1,scal}(X_l, x)\hat{\alpha}_{lj}^K,$$

wobei

$$\hat{K}_1(x, y) := K(x, y) - \frac{\big(\frac{1}{n}\sum_{i=1}^{n} K(x, X_i)\big) \cdot \big(\frac{1}{n}\sum_{i=1}^{n} K(X_i, y)\big)}{\frac{1}{n^2}\sum_{i,l=1}^{n} K(X_i, X_l)}. \tag{10.39}$$

Damit erhalten wir:

Definition 10.55
Es seien die Bezeichnungen wie in Definition 10.50. Es sei $\hat{K}_1$ gegeben durch Gl. (10.39). Die Abbildung

$$\hat{A}_n^{SC,K,k} : \mathbb{R}^d \to \mathbb{R}^k, \quad \hat{A}_n^{SC,K,k}(x) := \big(\hat{A}_n^{SC,K}(x)_1, ..., \hat{A}_n(x)_k^{SC,K}\big)^T$$

mit

$$\hat{A}_n^{SC,K}(x)_j := \Big(\frac{1}{n}\sum_{l=1}^{n} K(X_l, x)\Big)^{-1} \cdot \frac{1}{n\hat{\lambda}_j^K} \sum_{l=1}^{n} \hat{K}_1(X_l, x)\hat{\alpha}_{lj}^K, \qquad j = 1, ..., k$$

heißt die zu den ersten k Dimensionen gehörige Projektionsabbildung des spektralen Clusterns. ♦

10.2.4 Theoretische Resultate

In diesem Abschnitt verfolgen wir ein ähnliches Ziel wie in Abschn. 10.1.5 („Mathematische Bedeutung der Projektionen“).

Interpretieren wir die Projektionsabbildungen $\hat{A}_n^{SC,K}(\cdot)_j$, $j = 1, ..., k$ aus Definition 10.55 als Schätzungen von (uns unbekannten) „wahren" Projektionsabbildungen $A^{SC,K}(\cdot)_j$ und setzen wir

$$A^{SC,K,k} : \mathbb{R}^d \to \mathbb{R}^k, \quad A^{SC,K,k}(x) := \big(A^{SC,K}(x)_1, ..., A^{SC,K}(x)_k\big)^T,$$

so können wir $\hat{\alpha}_i^{SC,K,k} = \hat{A}_n^{SC,K,k}(X_i)$ als Schätzung der „wahren" Projektion $A^{SC,K,k}(X_i)$ interpretieren, und die Güte von $\hat{\alpha}_i^{SC,K,k}$ kann mittels

$$\sup_{i=1,...,n} \|\hat{A}_n^{SC,K,k}(X_i) - A^{SC,K,k}(X_i)\|_\infty$$

bewertet werden.

Da $\alpha_i^{K,k} := A^{SC,K,k}(X_i)$ anschaulich den „wahren" Projektionen der Daten X_i entsprechen, können mit Hilfe von $A^{SC,K,k}$ auch Modellannahmen an Verfahren formuliert werden, welche die aus dem spektralen Clustern erhaltenen Werte $\hat{\alpha}_i^{SC,K,k}$, $i = 1, ..., n$ weiterverarbeiten (vgl. Definition 10.57 unten).

Die Funktion $A^{SC,K,k}$ ist uns jedoch zunächst unbekannt; eine vage Definition als Limes von $\hat{A}_n^{SC,K}(\cdot)_j$ kann nicht als mathematische Definition genutzt werden. Im Folgenden leiten wir daher eine stringente Definition von $A^{SC,K,k}$ her. Sei $j \in \{1, ..., k\}$. Die Abbildung $\hat{A}_n^{SC,K}(\cdot)_j$ aus Definition 10.55 erfüllt

$$\hat{A}_n^{SC,K}(x)_j := \Big(\frac{1}{n}\sum_{l=1}^n K(X_l, x)\Big)^{-1} \cdot \frac{1}{n\hat{\lambda}_j^K} \sum_{l=1}^n \hat{K}_1(X_l, x)\hat{A}_n^{SC,K}(X_l)_j, \quad x \in \mathbb{R}^d.$$

Gilt $\hat{\lambda}_j^K \to \lambda_j^K$ $(n \to \infty)$ für ein geeignetes $\lambda_j^K \in \mathbb{R}$, so erwarten wir für $n \to \infty$

$$\lambda_j^K \cdot \mathbb{E}[K(X_1, x)] \cdot A^K(x)_j = \mathbb{E}\big[K_1(x, X_1)A^K(X_1)_j\big], \quad x \in \mathbb{R}^d,$$

wobei

$$K_1(x, y) := K(x, y) - \frac{\mathbb{E}K(X_1, y) \cdot \mathbb{E}K(x, X_1)}{\mathbb{E}K(X_1, X_2)}.$$

Das bedeutet: $g = A^K(\cdot)_j$ ist eine Lösung des verallgemeinerten Eigenwertproblems

$$T_K^{SC} g = \lambda \cdot T_K^{SC,diag} g \tag{10.40}$$

von Operatoren zum Eigenwert λ_j^K, wobei

$$\begin{aligned} T_K^{SC} &: L^2(\mathbb{P}^{X_1}) \to L^2(\mathbb{P}^{X_1}), \quad (T_K^{SC} g)(x) := \mathbb{E}\big[K_1(x, X_1)g(X_1)\big], \\ T_K^{SC,diag} &: L^2(\mathbb{P}^{X_1}) \to L^2(\mathbb{P}^{X_1}), \quad (T_K^{SC,diag} g)(x) := \mathbb{E}\big[K(x, X_1)] \cdot g(x) \end{aligned}$$

($L^2(\mathbb{P}^{X_1})$ wurde in Gl. (10.21) definiert). Wegen $\|\tilde{D}^{1/2}\hat{\alpha}_{\cdot j}^K\|_2^2 = n$ erwarten wir

$$\begin{aligned} 1 &= \frac{1}{n}\sum_{i=1}^{n}\Big((\frac{1}{n}\sum_{i=1}^{n}K(X_l,X_i))^{1/2}\hat{\alpha}_{ij}^{K}\Big)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\Big(\frac{1}{n}\sum_{i=1}^{n}K(X_l,X_i)\Big)\cdot(\hat{A}_n^{SC,K}(X_i)_j)^2 \\ &\to \mathbb{E}[K(X_1,X_2)A^{SC,K}(X_2)_j^2] \quad (n\to\infty), \end{aligned}$$

was eine Normierung der Funktionen $A^{SC,K}(\cdot)_j$ liefert und damit eine eindeutige Lösung des verallgemeinerten Eigenwertproblems Gl. (10.40) charakterisiert.

Im Gegensatz zur Kern-basierten PCA ist uns die mathematische Bedeutung der Eigenwerte λ_j^K unbekannt und daher eine formale Definition von $A^K(\cdot)_j$ noch nicht möglich. Aufgrund der Annahme $\hat{\lambda}_j^K \to \lambda_j^K$ $(n\to\infty)$ erwarten wir aber, dass die Ordnung der Eigenwerte λ_j^K für n groß genug derjenigen von $\hat{\lambda}_j^K$ entspricht. Das bedeutet, die Funktionen $A^K(\cdot)_j$ müssen als die Lösungen zu den k größten Eigenwerten $\lambda_1^K \geq ... \geq \lambda_k^K$ des Problems Gl. (10.40) definiert werden. Um eindeutige Lösungen zu erhalten, fordern wir, dass alle Eigenwerte in Gl. (10.40) verschieden sind. Damit erhalten wir die folgende formale Definition von $A^{SC,K,k}$:

Definition 10.56
Sei $k\in\mathbb{N}$. Es seien $\lambda_1^K > \lambda_2^K > ... > \lambda_k^K \geq 0$ die k größten Eigenwerte des verallgemeinerten Eigenwertproblems aus Gl. (10.40) mit zugehörigen Lösungen g_j, $j=1,...,k$ mit $\mathbb{E}[K(X_1,X_2)g_j(X_2)^2]=1$. Dann heißt

$$A^{SC,K,k}:\mathbb{R}^d\to\mathbb{R}^k,\quad A^{SC,K,k}(x):=\big(A^{SC,K}(x)_1,...,A^{SC,K}(x)_k\big)^T$$

mit $A^{SC,K}(\cdot)_j := g_j$ die zur Verteilung $\mathbb{P}^{X_1}$ und zum Kern K assoziierte *wahre Projektionsabbildung des spektralen Clusterns.* ♦

Werden die reduzierten Trainingsdaten $\hat{\alpha}_i^{SC,K,k}$ zur Verarbeitung mit einem weiteren Verfahren V des maschinellen Lernens genutzt, so können wir nun mit Hilfe von $A^{SC,K,k}$ die entsprechende Modellannahme kompakt formulieren.

Modellannahme 10.57 (Spektrales Clustern: Weiterverarbeitung) *Weiterverarbeitung der k-dimensionalen Projektionen:*

- Supervised Learning: Die Modellannahme von V gelte für $(A^{SC,K,k}(X_1),Y_1)$ (und nicht für (X_1,Y_1)).
- Unsupervised Learning: Die Modellannahme von V gelte für $A^{SC,K,k}(X_1)$ (und nicht für X_1).

Gehen wir davon aus, dass die Modellannahme mit einem festen $k \in \mathbb{N}$ gültig ist, so muss für gute Resultate des Verfahrens V nur gewährleistet sein, dass $\hat{\alpha}_i^{SC,K,k}$ eine gute Schätzung von $A^{SC,K,k}(X_i)$ liefert. In [36, Theorem 16 und Example 1] wird genau dies gezeigt, solange die Anzahl n der Trainingsdaten groß genug ist.

Satz 10.58 Es sei $k \in \mathbb{N}$, $\mathcal{X} \subset \mathbb{R}^d$ kompakt und $X_i \in \mathcal{X}$, $i = 1, ..., n$ i.i.d. Weiter sei K ein Mercer-Kern und es gebe $c_0 > 0$ mit $\inf_{x \in \mathcal{X}} K(x, x) \geq c_0$. In der Situation von Definition 10.56 (verschieden große Eigenwerte) gilt dann

$$\sup_{i=1,...,n} \|\hat{\alpha}_i^{SC,K,k} - A^{SC,K,k}(X_i)\|_\infty \to 0 \quad (n \to \infty). \tag{10.41}$$

Ist $K = K_\gamma^{gauss}$ der Gauß-Kern mit fest gewähltem $\gamma > 0$, so gibt es eine Konstante $c = c(k) > 0$ mit

$$\sup_{i=1,...,n} \|\hat{\alpha}_i^{SC,K,k} - A^{SC,K,k}(X_i)\|_\infty \leq \frac{c(k)}{\sqrt{n}}. \tag{10.42}$$

Bemerkungen

- Gl. (10.41) bedeutet, dass sich die Projektionen $\hat{\alpha}_i^{SC,K,k}$ des spektralen Clusterns mit steigender Anzahl an Trainingsdaten immer mehr den „wahren“ Projektionen $A^{SC,K,k}(X_i)$ annähern.
- Speziell für den Gauß-Kern kann durch Gl. (10.42) eine explizite Rate in n für die Annäherung angegeben werden. Diese ist (monoton wachsend) abhängig von k, d. h. für eine schnelle Konvergenzrate darf k nicht mit der Anzahl der Trainingsdaten wachsen.

Literatur

1. Akaike, H.: Information theory and an extension of the maximum likelihood principle. In Selected papers of Hirotugu Akaike, S. 199–213. Springer (1973)
2. Barron, A. R., Klusowski, J. M.: Approximation and estimation for high-dimensional deep learning networks. *arXiv e-prints*, page arXiv:1809.03090, Sep (2018)
3. Bartlett, P. L., Jordan, M. I., McAuliffe, J. D.: Convexity, classification, and risk bounds. J. Amer. Statist. Assoc. **101**(473), 138–156 (2006)
4. Bartlett, P. L., Traskin, M.: AdaBoost is consistent. J. Mach. Learn. Res. **8**, 2347–2368 (2007)
5. Bauer, B., Kohler, M.: On deep learning as a remedy for the curse of dimensionality in nonparametric regression. Ann. Statist. **47**(4), 2261–2285 (2018)
6. Bellman, R.: A markovian decision process. J. Math. Mech. **6**, 679–684 (1957)
7. Blanchard, G., Bousquet, O., Massart, P.: Statistical performance of support vector machines. Ann. Statist. **36**(2), 489–531 (2008)
8. Blanchard, G., Lugosi, G., Vayatis, N.: On the rate of convergence of regularized boosting classifiers. J. Mach. Learn. Res. **4**(5), 861–894 (2004)
9. Blanchard, G., Schäfer, C., Rozenholc, Y.: Oracle bounds and exact algorithm for dyadic classification trees. In International Conference on Computational Learning Theory, S. 378–392. Springer (2004)
10. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge University Press, Cambridge (2004)
11. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J.: Classification and regression trees. Wadsworth Statistics/Probability Series. Wadsworth Advanced Books and Software, Belmont (1984)
12. Bühlmann, P., Van De Geer, S.: Statistics for high-dimensional data: methods, theory and applications. Springer Science & Business Media, New York (2011)
13. Bühlmann, P., Yu, B.: Analyzing bagging. Ann. Statist. **30**(4), 927–961 (2002)
14. Chang, C.-C., Lin, C.-J.: Libsvm: A library for support vector machines. ACM transactions on intelligent systems and technology (TIST) **2**(3), 27 (2011)
15. Fan, R.-E., Chen, P.-H., Lin, C.-J.: Working set selection using second order information for training support vector machines. J. Mach. Learn. Res. **6**, 1889–1918 (2005)
16. Freund, Y., Schapire, R. E.: Experiments with a new boosting algorithm. Icml. **96,** 148–156 (1996)

S. Richter, *Statistisches und maschinelles Lernen,*
https://doi.org/10.1007/978-3-662-59354-7

17. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning. Springer Series in Statistics, 2. Aufl. Springer, New York (2009) (Data mining, inference, and prediction)
18. Hsu, D., Kakade, S. M., Zhang, T.: Random design analysis of ridge regression. Found. Comput. Math. **14**(3), 569–600 (2014)
19. Kohler, M., Langer, S.: Deep versus deeper learning in nonparametric regression. *Submitted for publication* (2018)
20. Lehmann, E. L., Casella, G.: Theory of point estimation. Springer Science & Business Media, New York (2006)
21. Andy, L., Wiener, M.: Classification and regression by randomforest. R News **2**(3), 18–22 (2002)
22. Pollard, D.: Strong consistency of k-means clustering. Ann. Statist. **9**(1), 135–140 (1981)
23. Puterman, M. L.: Markov decision processes: Discrete stochastic dynamic programming, 1. Aufl. Wiley, New York(1994)
24. Reiß, M., Wahl, M.: Non-asymptotic upper bounds for the reconstruction error of PCA. *arXiv e-prints*, page arXiv:1609.03779, Sep (2016)
25. Schmidt-Hieber, J.: Nonparametric regression using deep neural networks with ReLU activation function. *arXiv e-prints*, page arXiv:1708.06633, Aug (2017)
26. Scornet, E., Biau, G., Vert, J.-P.: Consistency of random forests. Ann. Statist. **43**(4), 1716–1741 (2015)
27. Scott, C.: Tree pruning with subadditive penalties. IEEE Trans. Signal Process. **53**(12), 4518–4525 (2005)
28. Scott, C., Nowak, R.: Dyadic classification trees via structural risk minimization. In Advances in Neural Information Processing Systems, S. 375–382 (2003)
29. Scott, C., Nowak, R.: Near-minimax optimal classification with dyadic classification trees. In Advances in neural information processing systems, S. 1117–1124 (2004)
30. Scott, C., Nowak, R. D.: Minimax-optimal classification with dyadic decision trees. IEEE Trans. Inform. Theory **52**(4), 1335–1353 (2006)
31. Steinwart, I., Scovel, C.: Fast rates for support vector machines using Gaussian kernels. Ann. Statist. **35**(2), 575–607 (2007)
32. Tseng, P., Center for Intelligent Control Systems (U.S.), Massachusetts Institute of Technology. Laboratory for Information, and Decision Systems.: Coordinate Ascent for Maximizing Non-differentiable Concave Functions. CICS (Series). Center for Intelligent Control Systems, M.I.T. (1988)
33. Tsybakov, A. B.: Optimal aggregation of classifiers in statistical learning. Ann. Statist. **32**(1), 135–166 (2004)
34. Tsybakov, A. B.: Introduction to Nonparametric Estimation, 1. Aufl. Springer Publishing Company, Incorporated, New York (2008)
35. van der Vaart, A.W.: Asymptotic statistics. Cambridge Series in Statistical and Probabilistic Mathematics, Bd. 3. Cambridge University Press, Cambridge (1998)
36. von Luxburg, U., Belkin, M., Bousquet, O.: Consistency of spectral clustering. Ann. Statist. **36**(2), 555–586 (2008)
37. Wager, S., Walther, G.: Adaptive Concentration of Regression Trees, with Application to Random Forests. *arXiv e-prints*, page arXiv:1503.06388, Mar (2015)
38. Wathen, A. J., Zhu, S.: On spectral distribution of kernel matrices related to radial basis functions. Numer. Algorithms. **70**(4), 709–726 (2015)
39. Christopher, J. C. H.: Watkins and Peter Dayan. Q-learning. Mach. Learn. **8**(3), 279–292 (1992)
40. Werner, D.: Funktionalanalysis. Springer-Lehrbuch. Springer, Berlin (2007)
41. Zhang, T.: Statistical behavior and consistency of classification methods based on convex risk minimization. Ann. Statist. **32**(1), 56–85 (2004)

42. Zhang, T., Bin, Y.: Boosting with early stopping: Convergence and consistency. Ann. Statist. **33**(4), 1538–1579 (2005)
43. Zhang, Y., Wainwright, M. J., Jordan, M. I.: Optimal prediction for sparse linear models? Lower bounds for coordinate-separable M-estimators. Electron. J. Stat. **11**(1), 752–799 (2017)
44. Zhou, S.: Restricted eigenvalue conditions on subgaussian random matrices. arXiv preprint arXiv:0912.4045 (2009)

Stichwortverzeichnis

S. Richter, *Statistisches und maschinelles Lernen,*
https://doi.org/10.1007/978-3-662-59354-7